KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittely

Sami Karhunen

LOHKOKETJU JA ÄLYKKÄÄT SOPIMUKSET

Opinnäytetyö
Helmikuu 2018

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Helmikuu 2018
Tietojenkäsittely

Tikkarinne 9
80220 JOENSUU
(013) 260 600

Tekijä(t)
Sami Karhunen

Nimeke
Lohkoketju ja älykkäät sopimukset

Toimeksiantaja
CGI Suomi

Tiivistelmä

Tässä opinnäytetyössä on perehdytty lohkoketjun teoreettiseen toimintaperiaatteeseen ja tutustuttu teknologioihin, jotka käyttävät lohkoketjua tai mahdollistavat lohkoketjusovellusten toteutuksen. Erityisesti tarkastellaan lohkoketjuteknologian mahdollistamaa älykästä sopimista, jossa toisille tuntemattomat ihmiset tai yritykset voivat keskenään tehdä sopimuksia ilman luotettavaa tahoa, kuten viranomaista, joka varmistaisi sopimuksen. Lisäksi tarkastellaan lohkoketjuteknologian soveltuvuutta eri käyttötarkoituksiin ja annetaan käytännön esimerkkejä sovelluksista, jotka hyödyntävät lohkoketjua.

Osana tätä opinnäytetyötä oli myös testata lohkoketjun kehittämistyökaluja käytännössä ja rakentaa sovellus, joka hyödyntää lohkoketjua älykkään sopimuksen rakentamiseen. Opinnäytetyö sisältää ohjeet, joita seuraamalla voi alustaa oman älykkään sopimuksen testiympäristöineen. Myös sovelluksen lähdekoodi ja asennusohjeet sisältyvät opinnäytetyöhön.

Opinnäytetyössä käydään läpi keskeiset lohkoketjun toimintaperiaatteet: tiiviste, konsensus ja lohkoketjun hajautettu luonne. Lohkoketjuteknologioista käsitellään mm. virtuaalivaluutta Bitcoinia ja älykkäiden sopimusten alustaa Ethereumia. Lohkoketjusovellusten kehittämiseen löytyy paljon työkaluja ja ne ovat toimivia ja usein hyvin dokumentoituja. Opinnäytetyön lopputuloksena on, että lohkoketjuteknologia on toimivaa, mutta sopivien sovelluskohteiden löytäminen ei ole helppoa.

Kieli	Sivuja 43
suomi	

Asiasanat
lohkoketju, älykkäät sopimukset, bitcoin, ethereum

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
February 2018
Business Information Technology

Tikkarinne 9
80220 JOENSUU
FINLAND
(013) 260 600

Author (s)
Sami Karhunen

Title
Blockchain and Smart Contracts

Commissioned by
CGI Suomi

Abstract

This thesis focuses on the theoretical principles of blockchain and explores technologies that utilize or enable implementation of blockchain applications. Particularly smart contracts enabled by blockchain technology are examined, in which unknown parties can conclude contracts without a trusted party, for example, an authority that would secure the agreement. In addition, the suitability of blockchain technology for different uses is studied and examples of applications that utilize the blockchain are given.

As part of this thesis blockchain development tools are tested in practice. Moreover, an application that utilizes blockchain to create and deploy smart contracts is built. Instructions are provided and they can be used to initialize your smart contract and testing environment. Also source code for the built application and instructions how to use are included.

The thesis discusses the main principles of blockchain: hash, consensus and decentralization. As examples of blockchain technologies viral currency Bitcoin and the smart contract platform Ethereum are presented. There is a variety of tools available for the development of blockchain applications which are functional and well documented. The result of this thesis is that blockchain technology works well, but finding suitable applications is not easy.

Language
Finnish

Pages 43

Keywords
blockchain, smart contracts, bitcoin, ethereum

Sisältö

1 Johdanto 5
2 Lohkoketju 6
2.1 Lohkoketjun toimintaperiaate 6
2.2 Konsensusprotokollat 8
3 Lohkoketjuteknologioita 12
3.1 Bitcoin 12
3.2 Ethereum 13
3.3 Solidity 15
3.4 Oraclize 16
4 Sopimuksen tietojen julkisuus 17
5 Tietoturva 20
6 Yksityiset lohkoketjut 21
6.1 Quorum 22
6.2 Hyperledger 23
6.3 Corda 25
7 Käytännön esimerkkejä 26
8 Sovellus 27
9 Älykkään sovelluksen kehittäminen 30
9.1 TestRPC 32
9.2 Sovelluksen kääntäminen 35
9.3 Sovelluksen migraatio 35
9.4 Sovelluksen testaaminen 36
9.5 web3.js 38
9.6 Sovelluksen asennusohje 39
10 Pohdintaa lohkoketjusovellusten tulevaisuudesta 41
Lähteet 44

1 Johdanto

Internet tarjoaa mahdollisuuden jakaa tietoa helposti ja halutessaan anonyymisti, tai esiintymällä jonakin muuna kuin kuka oikeasti on. Tämän vuoksi internetissä tarvitaan tahoja, joiden tarjoamaan tietoon voi luottaa. Pankit tarjoavat yrityksille vahvaa henkilön tunnistuspalvelua, tai kuluttajalle turvaa siitä, että verkkokauppa ei laskuta luottokorttia sovittua enempää. Nykyään yleistyvät jakamistalouden markkinat perustuvat usein luottamusta tarjoaviin yrityksiin, kuten Airbnb ja Uber. Nämä yritykset eivät omista huoneistoja, tai autoja joita välittävät, vaan tarjoavat luottamusta vuokralle tarjoajan ja vuokraajan välille. Lohkoketjuteknologian tavoitteena on tuoda tekninen ratkaisu luottamuksen synnyttämiseen luotettujen tahojen sijasta.

Lohkoketju on teknologia, jonka avulla toisilleen tuntemattomat tahot pystyvät yhdessä ylläpitämään luotettavaa, hajautettua tietokantaa, jonka sisältöä ei ole jälkikäteen mahdollista muokata. Lohkoketjun sisältö sijaitsee kaikilla varmistamiseen osallistuvilla solmuilla (engl. node). Siinä ei ole palveluntarjoajaa, tai keskushallintoa, joka varmistaisi ketjun sisällön aitouden, eikä näin ollen myöskään sellaista solmukohtaa, jonne vihamielinen taho voisi kohdistaa hyökkäystä. Lohkoketjuteknologian ensimmäinen sovellus on Bitcoin-virtuaalivaluutta. (Iansiti & Lakhani 2017.)

Lohkoketjua voi hyödyntää tiedon tallennuksen lisäksi myös tallentamalla siihen ohjelmakoodia, joka toteuttaa itse itsensä, kun sopimukselle määritellyt raja-arvot täyttyvät. Seuraavassa on kaksi esimerkkiä: kuljetuspalvelu, jossa asiakkaan kuitatessa lähetyksen saapuneeksi, siirtyy maksu automaattisesti asiakkaalta toimittajalle, tai sovellus joka maksaa myöhästyneestä lennosta korvausta asiakkaan tilille automaattisesti ja välittömästi myöhästymisen tapahtuessa.

Lohkoketjuteknologia on tällä hetkellä tutkimusyhtiö Gartnerin hypekäyrän huipulla. Gartner ennustaa, että teknologia arkipäiväistyy viidestä kymmeneen vuoden kuluttua (Panetta 2017). Vaikka teknologia onkin vielä varsin tuoretta, on konsepti esitelty jo vuonna 1991 Stuart Haberin ja W. Scott Stornettan toimesta (Haber & Stornetta 1991, 99).

Tässä opinnäytetyössä käydään läpi lohkoketjun toimintaperiaatetta, tutustutaan erilaisiin lohkoketjuteknologioihin ja toteutetaan sovellus, joka käyttää nykyaikaisia lohkoketjujen kehittämiseen soveltuvia työkaluja.

2 Lohkoketju

Termi lohkoketju muodostuu kahdesta asiasta: lohkosta ja ketjusta. Lohko sisältää tallennettavaa dataa. Tämä voi olla Bitcoinin tapauksessa transaktioita, joissa virtuaalivaluutta vaihtaa omistajaa, tai älykkäiden sopimuksien kohdalla ohjelmakoodia, joka suoritetaan lohkoa muodostettaessa.

Ketjulla taas viitataan rakenteeseen, jossa uusi lohko lisätään ketjuun aiempien lisäksi. Ylläpidon muodostavat solmut, jotka käytännössä ovat tietokoneita, tai nykyään yleisemmin koneiden yhteenliittymiä. Tämän luvun alaluvuissa käydään läpi lohkoketjun toimintaa periaatetasolla.

2.1 Lohkoketjun toimintaperiaate

Mikään osa lohkoketjusta ei sijaitse vain yhdessä paikassa verkossa, vaan kaikilla lohkoketjua ylläpitävillä solmuilla on koko lohkoketjun sisältö ensimmäisestä tapahtumasta viimeisimpään. Tämä myös takaa sen, että lohkoketjussa ei ole sellaista heikkoa kohtaa, johon hakkeri voisi murtautua ja väärentää tietoa. Jos yksi, tai vaikka useampikin verkon solmuista lakkaa toimimasta, muut jatkavat verkon ylläpitoa normaalisti. (Iansiti & Lakhani 2017.)

Kaikki verkon solmut ovat tasa-arvoisia (Harper 2018). Verkko on myös avoin kaikille, jotka haluavat liittyä toimijaksi verkkoon. Pääsyoikeuksia ei tarvitse, tai pysty valvomaan.

Lohkoketjun jatkoksi lisättävä uusi lohko sisältää dataa, transaktioita, tai älykkään sopimuksen suorittamaa koodia. Lisäksi lohkoon lisätään aina aikaleima ja tiiviste

edellisen lohkon sisällöstä. Uusi lohko jaetaan kaikille ylläpitoon osallistuville solmuille.

Tiiviste (engl. hash) on tiivistealgoritmin datasta tuottama, satunnaiselta vaikuttava merkkijono. Merkkijonosta ei pysty palauttamaan alkuperäistä dataa, mutta vertaamalla alkuperäistä dataa tiivisteeseen, voidaan varmistua siitä, että tiiviste on luotu juuri kyseisestä datasta. Mikäli alkuperäisessä datassa muutetaan edes yhtä bittiä, niin tiivisteestä muodostuu täysin erilainen merkkijono. Esimerkiksi Bitcoin käyttää tiivisteenä SHA256-algoritmia, joka tuottaa aina 256-bittisen tiivisteen. (Bitcoinwiki 2015.)

Tiivisteen tarkoitus on siis todentaa lohkoketjun historian muuttumattomuus. Esimerkiksi jos olisin ostanut tuotteen verkkokaupasta, joka hyväksyy Bitcoinin maksuvälineenä ja tuotteen saatuani haluaisin perua tuon Bitcoinin siirron, niin voisin etsiä Bitcoinin historiasta kyseisen lohkon jossa siirto on tapahtunut ja perua sen. Tämä muuttaisi lohkon sisältöä, minkä seurauksena lohkosta muodostettavan tiivisteen sisältö myös muuttuisi. Tämä myös tarkoittaisi, että kaikkien myöhempien lohkojen sisältö muuttuisi, koska jokainen lohko sisältää myös edellisen lohkon tiivisteen. (Faife 2017.)

Periaatteessa kuka vain voi yrittää lisätä uuden lohkon lohkoketjun jatkeeksi, mutta vasta verkon konsensus siitä, että kyseinen lohko hyväksytään ketjuun, validoi lohkon sisällön (Lewis 2015). Kuten jo aiemmin todettiin, että lohkoketju on tasavertaisista solmuista koostuva vertaisverkko eikä siihen kuulu sellaista yksittäistä tahoa, jolla olisi valtaa varmistaa lohkoja aidoksi, tai väärennetyksi. Sen sijaan verkon solmujen täytyy päästä yhdessä konsensukseen siitä, mitkä lohkot ovat aitoja ja mitkä väärennettyjä.

Uuden lohkon lisääminen lohkoketjuun edellyttää lohkoketjun sääntöjen noudattamista. Tätä käsitellään tarkemmin seuraavassa, konsensusprotokollia koskevassa alaluvussa, mutta lähtökohtana on, että uuden lohkon lisäävällä solmulle on enemmän hyötyä lisätä todenmukaista, kuin väärennettyä dataa. Esimerkiksi Bitcoinin tapauksessa tämä tarkoittaa sitä, että väärennetyn datan

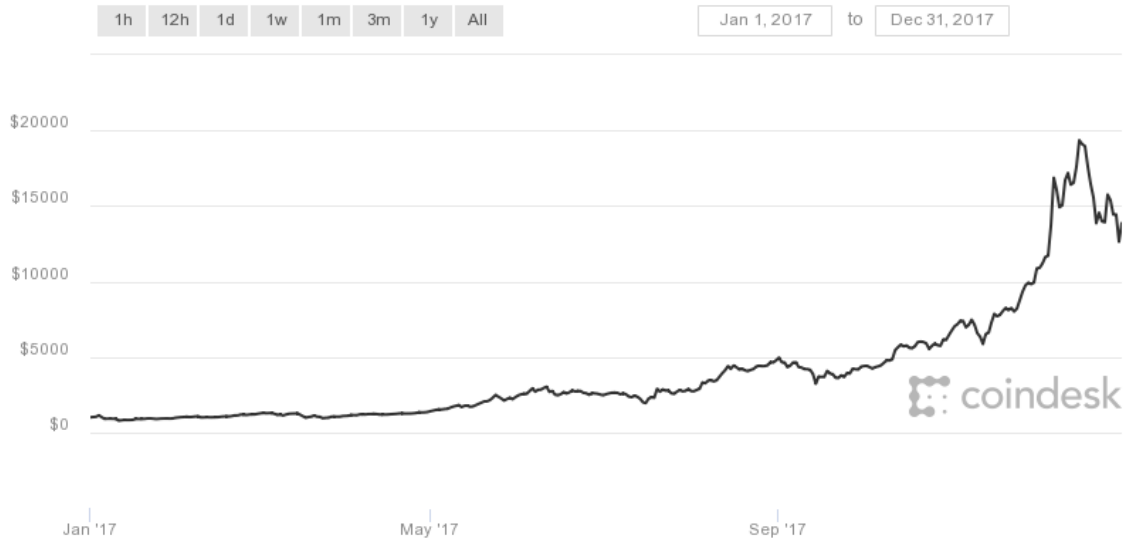
lisääminen vaatisi niin paljon prosessoritehon hankkimista, ettei se ole taloudellisesti kannattavaa. (Lewis 2015.)

Konsensusta ei saavuteta reaaliajassa, vaan toisinaan voidaan päätyä tilanteeseen, jossa lohkoketjuun lisätään samanaikaisesti kaksi uutta lohkoa (jotka voivat olla molemmat valideja), jolloin ketju haarautuu. Lopulta voimaan jää se haara, jonka jatkeeksi enemmistö solmuista jatkaa uusien lohkojen lisäämistä, eli näin verkon enemmistö määrää konsensuksesta. Kun toinen haara saa puolelleen enemmistön, ei muidenkaan solmujen enää kannata jatkaa vähemmistöksi jääneen haarautuneen ketjun ylläpitoa koska tämä haarautunut ketju ei koskaan saavuttanut konsensusta. (Lewis 2015.)

2.2 Konsensusprotokollat

Tällä hetkellä sekä Bitcoin, että Ethereum käyttävät konsensusprotokollana todistusta tehdystä työstä (engl. proof of work) (Rosic 2017a). Käytännössä tämä toimii niin, että verkkoa ylläpitävät solmut kilpailevat toisiaan vastaan siitä kuka saa lisätä seuraavan lohkon lohkoketjuun. Tästä keskinäisestä kilpailusta käytetään yleisesti termiä louhiminen. Lohkon lisäyksestä maksetaan palkkio, joka on erityisesti Bitcoinin osalta taloudellisesti hyvin merkittävä, 12,5 Bitcoinia (Bitcoinwiki 2017). Bitcoinin kurssi on tällä hetkellä (29.1.2018) lähes puolittunut siitä huipusta, jossa kurssi kävi vuoden 2017 loppupuolella. Vuoden 2017 huipussaan kurssi oli toisaalta n. 20 kertaa korkeampi kuin vuoden 2017 alussa. 12,5 bitcoinin louhimispalkkio vastaa tämän hetken kurssilla yli 100 000 euron summaa (Coindesk 2018).

Bitcoinin kurssikehitys vuodelta 2017 (kuva 2.)



Kuva 2. Bitcoin kurssi (Coindesk 2018).

Louhimisessa on kyse prosessoritehoa vaativan algoritmin ratkaisusta. Bitcoinin tapauksessa tämä toimii niin, että louhija pyrkii muodostamaan tiivisteen, johon sisältyy edellisen lohkon tiiviste, uuteen lohkoon lisättävät transaktiot, aikaleima ja ns. nonce-arvo. Näistä muodostettavan SHA256-tiivisteen alussa tulee olla tietty määrä nollia. Tähän pyritään sijoittamalla nonce-arvoksi sellainen merkkijono, että lohkon tiivisteen alkuun muodostuu tavoiteltava määrä nollia. Bitcoinin tapauksessa nollien määrä on tällä hetkellä 18 (Tämän voi varmentaa katsomalla uusimpia lohkoja esimerkiksi sivustolla www.blockexplorer.com), ja vaikeustaso pyritään pitämään sellaisena, että uusia lohkoja louhitaan n. 10 minuutin välein. (Rosic 2017a.)

18 peräkkäistä nollaa SHA256-tiivisteen alussa on erittäin harvinaista, joten louhija joutuu käymään miljoonia mahdollisia nonce-arvoja läpi ennen sopivan löytymistä. Käytännössä louhiminen on nykyään erityisten louhimiseen erikoistuneiden koneyhteenliittymien hallinnassa, eikä yksittäisten kotikoneiden prosessoritehoilla pärjää kilpailussa. Vaikka louhiminen vaatii paljon prosessoritehoa, niin valmiin tiivisteen tarkastaminen on kotikoneellakin helppoa. Muodostettua tiivistettä verrataan lohkon sisältämään dataan ja nähdään välittömästi vastaako tiiviste tätä dataa.

Mikäli joku siis haluaisi väärentää dataa proof-of-workin tapauksessa, tulisi tällä olla riittävästi prosessoritehoa käytössään, niin että hän pystyisi muodostamaan väärennetyn transaktion sisältävän proof-of-workin algoritmin mukaisen tiivisteen nopeammin, kuin kukaan muu louhijoista. Edes tämä yksi onnistunut väärennetty lohko ei kuitenkaan riittäisi. Muut louhijat pystyisivät välittömästi toteamaan, tiivistettä vertaamalla, että lohko on väärennetty. Tällöin louhijan ei kannattaisi jatkaa louhimista tämän väärennetyn lohkon päälle, koska enemmistö ei koskaan tulisi hyväksymään lohkon jatkeeksi tehtyä lohkoa. Väärentäjän pitäisi pystyä louhimaan useita peräkkäisiä lohkoja onnistuakseen saamaan väärennöksen läpi. On arveltu, että edes valtiollisilla toimijoilla ei ole niin paljon prosessoritehoa käytössä, että tämä olisi mahdollista.

Työtodistus-protokollassa on myös se etu, että kukaan ei pysty etukäteen ennustamaan, kuka ja missä seuraavan lohkon ratkaisee. Ja näin ollen kukaan ei pysty kohdistamaan esim. palveluestohyökkäystä mihinkään tiettyyn tahoon, koska tällöin vain joku toinen louhija ratkaisee lohkon, eikä lohkoketjun toiminta häiriinny lainkaan.

Työtodistus-protokolla on toimiva ja Bitcoinin tapauksessa jo lähes 10 vuoden ajan käytännössä testattu ja luotettavaksi todettu konsensusprotokolla. Sen heikoin puoli toisaalta liittyy valtaisaan energiankulutukseen. Kun valtavat määrät konetehoa kilpailee siitä, kuka seuraavan lohkon saa ratkaistua, kuluu sähköä maailmanlaajuisesti erittäin massiivisia määriä. https://digiconomist.net/bitcoin-energy-consumption sivuston arvio Bitcoinin nykyisestä vuosikulutuksesta on hieman yli 20 TWh, joka on suunnilleen samaa luokkaa kuin Azerbaidzanin (väkiluku n. 9,7 miljoonaa) valtion sähkönkulutus.

Energian hinta vaihtelee runsaasti eri maissa, mutta jos esimerkkinä lasketaan sähköön kulunutta varallisuutta verrattuna syntyneiden Bitcoinien arvoon. https://www.sahkonhinta.fi sivuston mukaan sähkön hinta on vuoden 2018 alkupuolella ollut suunnilleen 6 senttiä kilowattitunnilta. Bitcoinin louhintaan kulutettu sähköenergia olisi siis maksanut n. 1,2 miljardia euroa. Sähkönsiirtohinta tekisi vielä suunnilleen toisen 1,2 miljardia lisää, yhteensä siis 2,4 miljardia euroa. Uusia Bitcoineja syntyy 12,5 kpl noin 10 minuutin välein eli

75 kappaletta tunnissa. Tämä tekee 1800 kappaletta vuorokaudessa ja vuodessa Bitcoineja siis syntyy hieman karkeasti arvioiden 650 000 kpl. Tämän hetken Bitcoinin kurssilla uuden syntyneen valuutan arvo on 6 miljardia euroa, eli hieman yli kaksinkertaisesti käytetyn sähkön hintaan nähden. Laskelma on suurpiirteinen, koska sähkönhinta vaihtelee ja Bitcoineja voi syntyä nopeammin, tai hitaammin kuin 10 minuutin välein. Myös Bitcoinin kurssi vaihtelee rajusti. Tästä laskelmasta on nähtävissä, että Bitcoinin tuo arvonlisää käytetyn sähkön hintaan nähden, mutta kurssivaihtelu ja sähkönhinnan nousu voi toisaalta kääntää tilanteen myös negatiiviseksi.

Virtuaalivaluutat kuten Nxt, BlackCoin ja Peercoin käyttävät vaihtoehtoista konsensusprotokollaa, todistetta panoksesta (eng. proof-of-stake) (Castor 2017). Myös Ethereum lohkoketju on vaihtamassa tulevassa Casper julkaisussa konsensusprotokollan nykyisen työtodistuksen sijasta todisteeseen panoksesta. Tässä protokollassa osallistuakseen lohkoketjun varmentamiseen tulee louhijan laittaa kyseistä virtuaalivaluuttaa syrjään panokseksi ketjun varmentamisesta. Palkkio lohkon louhimisesta maksetaan suhteessa syrjään laitetusta panoksesta.

Todiste panoksesta-toteutuksia on kahta erilaista. Yksinkertaisemmassa louhija valitaan satunnaisesti panoksen laittaneiden joukosta. Monimutkaisemmassa (ns. Byzantine Fault Tolerance) versiossa satunnaiselle joukolle panoksen laittaneista annetaan oikeus ehdottaa seuraavaa lohkoa. Konsensus saavutetaan louhijoiden antamilla äänillä, joiden enemmistön perusteella päätetään validit lohkot. Mikäli joku taho selvästi tahallisesti käyttää panostaan väärinkäytöksiin, voivat äänestäjät päättää, että tämän tahon panokseksi laittamat varat tuhotaan. (Kronovet 2017.)

Tämän protokollan selkein etu on, että se ei kuluta valtavia määriä energiaa hukkaan lohkoketjun louhimiseen. Myös maksettuja palkkioita voidaan pienentää, koska käytännössä kaikkien virtuaalivaluuttaa omistavien kannattaa pitää varansa panoksena louhimisessa. Tällaisessa protokollassa ei tarvita niin suurta taloudellista motivaattoria, kuin työtodistuksessa, jossa louhijat sijoittavat valtavia rahasummia laitteistoon ja sähköön. Koska palkkiona käytettävät virtuaalivaluutat ovat aina uutta, tyhjästä luotua virtuaalivaluutta, ovat ne myös

omiaan aiheuttamaan inflaatiota, mutta kun palkkiosummaa voidaan pienentää, myös paine valuutan arvon laskuun pienenee. (Rosic 2017a.)

3 Lohkoketjuteknologioita

Tässä kappaleessa käsitellään lohkoketjua käytäntöön soveltavia teknologioita: virtuaalivaluutta Bitcoinia, älykkäiden sopimusten lohkoketju-alusta Ethereumia ja ohjelmointikieli Soliditya, sekä Oraclizea, jonka avulla alykkäät sopimukset voivat olla yhteydessä ulkoisiin verkkopalveluihin.

3.1 Bitcoin

Lohkoketjun ensimmäinen merkittävä toteutus on Bitcoin ja usein hieman virheellisesti nämä kaksi asiaa jopa liitetään synonyymeiksi toisilleen. Bitcoin on virtuaalivaluutta, jota pystytään siirtämään omistajalta toiselle ilman, että osapuolien täytyy tuntea tai luottaa toisiinsa. Taustalla oleva teknologia varmistaa myös, ettei samaa valuuttaa voi käyttää kuin kerran. Bitcoinin konseptin esitteli vuonna 2008 nimimerkkiä Satoshi Nakamoto käyttänyt henkilö, tai ryhmä (Nakamoto 2008). Satoshi Nakamoton henkilöllisyyttä ei ole varmuudella pystytty todentamaan tähän päivään mennessä (Feins 2017). Ensimmäiset bitcoinit laskettiin liikkeelle tammikuussa 2009, samalla kun ensimmäinen Bitcoin lohko syntyi (Blockexplorer 2017). Bitcoin on epävirallinen, mutta laajasti käytetty digitaalinen valuutta. Bitcoineilla maksaessa ostajan ja myyjän on mahdollista pysyä toisilleen anonyymeina, mutta varmistua siitä, että valuutta siirtyy toiselle. Siirtoa varten ei tarvita luotettua välikättä, kuten pankkia. Bitcoin ei ole ainoa virtuaalivaluutta, mutta ylivoimaisesti tunnetuin ja eniten käytetty.

Bitcoin siirrot ja koko siirtojen historia on väärentämätön. Bitcoin lohkoketjusta voi lukea kaikki siirrot, mistä osoitteesta ja mihin osoitteeseen on siirretty. Se ei tarkoita sitä, että kaikki bitcoinit olisivat automaattisesti turvassa. Bitcoinin omistajalla on salattu avain, jolla tämä voi todistaa omistavansa tietyn bitcoinin. Tätä avainta käyttämällä voi tehdä siirron toiselle taholle. Kuitenkin salausavain

voi hukkua, tai se voidaan hakkeroida aivan kuten mikä tahansa salasana ja näin ollen myös bitcoinit voidaan varastaa. Tämän jälkeen varastetun bitcoinin siirto voidaan todentaa bitcoinin lohkoketjusta. Koska omistajat ovat anonyymeja, niin käytännössä varastettuja bitcoineja on mahdotonta saada takaisin. Toisaalta jos salausavaimiaan säilyttää yhdessä paikassa, vaikka ulkoisella kovalevyllä, joka rikkoutuu, niin bitcoinit ovat tämän jälkeen menetetty kokonaisuudessaan ikuisesti. (Brito & Castillo 2013.)

Bitcoinin keskimääräinen transaktiomaksu on tällä hetkellä 28 dollaria ja siirron varmistaminen kestää keskimäärin 78 minuuttia (Browne 2017). Transaktiomaksua ei ole pakko maksaa, mutta silloin transaktio on vähemmän houkutteleva louhijoille ja sen varmistaminen kestää pidempään. Tähän verrattuna suomalaisen pankin käyttäminen vaikuttaa ketterältä, luotettavalta ja edulliselta. Toisaalta kansainväliset rahansiirrot voivat olla pankkien välityksellä Bitcoinia hitaampia ja kalliimpia.

3.2 Ethereum

Ethereum on hajautettu, lohkoketjuun pohjautuva alusta älykkäille sopimuksille, jonka konseptin Vitalik Buterin esitteli 2013 (Ethereum Foundation 2018). EVM eli Ethereum Virtual Machine on Turing-täydellinen virtuaalikone, joka pystyy ajamaan skriptejä kansainvälisesti hajautetussa louhijoiden verkossa. Lohkoketju vaatii taloudellisen kannusteen louhijoille, ja osa Ethereumia on sen oma virtuaalivaluutta Ether. Etherin arvo on ollut rajussa nousussa, vielä kesällä 2017 sen arvo oli 10 dollarin luokkaa, mutta tällä hetkellä (31.1.2018) arvo on hieman yli
1 000 dollaria (Coindesk 2018). Louhijan palkkio varmistetusta lohkosta on 5 etheria ja uusia lohkoja ratkaistaan keskimäärin 15 sekunnin välein (Ethereum 2018a).

Koska Ethereumissa ajetaan scriptejä, on erityisen tärkeää suunnitella ohjelmakoodi niin, että se kuluttaa mahdollisimman vähän resursseja. Ethereumissa jokainen ajettu scripti kuluttaa louhijan resursseja, ja kulutettujen

resurssien yksikkönä on gas. Mitä enemmän scripti vaatii resursseja, sitä useampi gas-yksikkö kuluu sen toteuttamiseen.

Gas ei ole virtuaalivaluutta, eikä sille ole suoraa rahallista arvoa määritettynä. Arvo ei myöskään ole suorassa suhteessa Etherin hintaan, vaikka Etheria käytetäänkin gasin maksuvälineenä. Tämä siksi, että virtuaalivaluutat, Ether mukaan luettuna saattavat vaihdella arvoltaan hyvinkin rajusti, mutta prosessoritehon hinta pysyy varsin vakaana, joten gasin arvo määräytyy markkinan perusteella. (Madeira 2018.)

Käytännössä siis taho joka haluaa sopimuksensa louhittavaksi voi määritellä kuinka paljon tästä on valmis maksamaan. Sopimukselle määritellään sekä hinta, jonka sen louhimisesta on valmis maksamaan gas-yksikköä kohti sekä määritellään maksimi kulutetuille gas-yksiköille. Louhijat taas voivat päättää onko sopimus houkutteleva louhittavaksi. Näin ollen mikäli sopimuksen louhimisesta ei ole valmis maksamaan riittävää määrää gas yksiköltä, se voi jäädä louhimatta kokonaan, tai ainakin louhiminen kestää paljon pidempään. (Madeira 2018.)

Louhijat saavat siis louhinnasta aiheutuvat louhintapalkkiot. Louhijan näkökulmasta on toki houkuttelevaa ottaa mahdollisimman paljon louhittavaa mukaan, jopa pienempää hintaa tarjoavatkin, mutta tämä johtaa toisaalta siihen, että louhittavaan lohkoon tulee paljon varmistettavia sopimuksia ja tämä vaatii paljon prosessoritehoa, tai vastaavasti kestää pidempään ja tämän vuoksi louhija ei enää pärjää kilpailussa niitä louhijoita vastaan, jotka louhivat pienemmän määrän sopimuksia tinkien samalla omasta palkkiostaan.

Gas on siis kannustamassa sitä, että julkaistavat sopimukset suunniteltaisiin käyttämään resursseja säästellen ja toisaalta rajoittamassa sitä, että ketjuun ei tulisi turhaan kokeilu- tai leikkimielisiä älykkäitä sopimuksia, tai tahattomia sovellusvirheitä, jossa älykkään sopimuksen skripti esimerkiksi jäisi ikuiseen for-looppiin.

Älykkään sopimuksen vaatimaa prosessoritehoa voi toki ennakoida etukäteen suunnittelemalla sen vaatimat skriptit kuluttamaan mahdollisimman vähän

resursseja, mutta tarkkaa määrää kulutetuille gas-yksiköille voi olla mahdoton sanoa etukäteen. Älykkäälle sopimukselle voi määritellä sekä hinnan gas-yksikölle, myös sen kuinka paljon sopimuksen louhiminen voi maksimissaan maksaa. Mikäli sopimus kuluttaa paljon resursseja, mutta yläraja on liian alhainen jää sopimus suorittamatta. Louhija saa kuitenkin pitää siihen saakka kertyneet palkkiot itsellään, koska resurssit ovat jo käytetty. Louhija palauttaa sopimuksen siihen tilaan kuin se oli ennen suorituksen aloittamista ja lisää sen lohkoketjuun statuksena ”epäonnistunut transaktio”. Toisaalta mikäli louhiminen kuluttaa vähemmän resursseja, kuin mitä ylärajaksi on määritelty, niin sopimuksen omistajalle palautuu käyttämättä jääneet varat. (Tomshwom 2017.)

Tämä siis vaatii älykkään sopimuksen suunnittelijalta hyvää ymmärrystä siitä mikä gasin merkitys on. Ja kuinka paljon oma sopimus kuluttaa prosessointitehoa. Kehittäjän tulee arvioida sitä, että haluaako sopimuksen toteutuvan ripeämmin ja onko valmis maksamaan siitä. Gas-ylärajan määrittäminen turhan korkeaksi ei nosta sopimuksen louhimisen haluttavuutta louhijoiden keskuudessa. Itseasiassa erityisen korkeaksi nostettu yläraja voi aiheuttaa sen, että sopimus näyttäytyy louhijalle niin resurssisyöpöltä, ettei se mahdu enää tämän lohkoketjulle asetettujen resurssien puitteisiin ja jää sen vuoksi louhimatta.

3.3 Solidity

Solidity on älykkäitä sopimuksia varten kehitetty ohjelmointikieli, jota voi ajaa Ethereumin virtuaalisessa koneessa (Allison 2016). Solidity on Gavin Woodin vuonna 2014 esittelemä ohjelmointikieli (stackedit.io 2014), jonka kehityksestä vastaa Ethereumin projekti Solidityn työryhmä Christian Reitwiessnerin johdolla (Reitwiessner 2018). Soliditya voi käyttää Ethereumin lisäksi ohjelmointikielenä Hyperledger Burrowsin lohkoketjualustalla (Hyperledger Burrows 2018).

Solidity on korkean-tason ohjelmointikieli. Kieli on staattisesti tyypitetty, tukee perintää ja kirjastojen käyttöä. Ennen ajoa Ethereumin virtuaalikoneessa, tulee koodi kääntää virtuaalikoneen ymmärtämään bytecode-muotoon.

Kehitykseen on otettu vaikutteita C++, Python ja Javascript-ohjelmointikielistä (Solidity 2017). Syntaksiltaan kieli muistuttaa Javascriptia. Osoitteesta http://solidity.readthedocs.io/en/latest/ löytyy Solidityn dokumentaatio.

Tuki Soliditylle löytyy useista ohjelmointiympäristöistä. Intellij IDEA:an ja SublimeTextiin saa pluginin, joka ymmärtää Solidityn syntaksin ja tekee koodin värityksen ja muotoilun oikein. Visual Studioon ja Visual Studio Codeen saa koodin värityksen lisäksi Solidityn kääntäjän. https://remix.ethereum.org/ tarjoaa selaimessa toimivan kehitysympäristön, johon sisältyy kääntäjä ja ajoympäristö.

3.4 Oraclize

Älykkäät sopimukset eivät voi olla suoraan yhteydessä ulkoisiin verkkosivustoihin. Tämä voisi olla usein hyödyllistä, esimerkiksi osana älykkään sopimuksen koodia voitaisiin hakea ulkoisesta verkkosivusta tietoa valuuttakursseista, tai vaikkapa tietoa säätilasta. Tämä rajoitus johtuu siitä, että älykkään sopimuksen lopputulos täytyy olla todennettavissa kaikkina aikoina. Jos sopimuskoodi ajetaan identtisillä parametreillä, on sen kaikkina aikoina palautettava sama lopputulos. Ulkoinen verkkosivu voi kuitenkin palauttaa eri arvon eri aikoina, tai lakata kokonaan vastaamasta, joten sopimuksen lopputulos voisi kahden eri testikerran välillä erota toisistaan. Sopimuskoodi myös ajetaan jokaisen louhijan koneella, joten tällöin kilpaillessaan seuraavan lohkon ratkaisusta, jokainen louhija kutsuisi ulkoista verkkosivustoa lähes samanaikaisesti ja tämä aiheuttaisi turhaa kuormaa, ehkä jopa kaataisi verkkosivuston. (Greenspan 2016.)

Tätä rajoitusta voi kuitenkin kiertää käyttämällä luotettua tahoa, joka hoitaa kyselyn ja tarjoaa datan älykkäälle sopimukselle. Oraclize tarjoaa palvelua verkko-apien ja älykkäiden sopimusten välille. Oraclize tarjoaa noudetun datan lisäksi aitoustodistuksen siitä, että Oraclize on noutanut datan alkuperäisestä datalähteestä ja data on alkuperäinen ja muokkaamaton. (Oraclize 2018.)

Tämän kiertotien heikkous on siinä, että älykkään sopimuksen laatija joutuu luottamaan kolmannen osapuolen palveluun. Jos palveluntarjoajan palvelu

hakkeroidaan, tai se ei tuota muusta syystä luotettavaa tulosta, älykäs sopimus silti luottaa Oraclizen noutamaan tietoon. Jos palvelu näyttäytyy Oraclizelle validina, palauttaa Oraclize datan ja aitoustodistuksen, ja vaikka sopimuksen laatija tietäisi Oraclizen datan olevan väärennetystä lähteestä, ei hänellä olisi keinoa näyttää tätä toteen. Käytännössä sopimuksen laatijan tulisi vähintään toteuttaa mekanismi, jolla oman sopimuksen pystyy laittamaan tauolle mahdollisten virhetilanteiden varalta. Oraclize on maksullinen palvelu, joten siitä myös aiheutuu lisäkuluja älykkään sopimuksen ylläpitäjälle. Useat ulkopuoliset API:t ovat suojattuja salasanalla. Oraclize mahdollistaa API-avaimen kryptauksen, jolloin API-avainta ei tarvitse sopimuksella paljastaa selväkielisenä ja vain Oraclize voi salausavaimellaan purkaa avaimen kryptauksen. Tässä edelleen käyttäjä joutuu luottamaan Oraclizeen avaimen osalta. Jotkut palvelut myös rajoittavat API-avaimen jakamista kolmannelle osapuolelle, riippumatta siitä käytetäänkö kryptausta. (Greenspan 2016.)

4 Sopimuksen tietojen julkisuus

Julkisen Ethereumin luonteeseen kuuluu, että kaikki älykkään sopimuksen sisältämä informaatio on julkista ja helposti kaikkien saatavilla. Näin ollen tiedosta on myös mahdollista päätellä sen taustalla oleva tarkoitus ja logiikka ja hankkia esimerkiksi tietoa toisen osapuolen sopimushistoriasta. (Tual 2015.)

https://etherscan.io/ on verkkosivusto, josta voi selata Ethereumin louhittuja lohkoja. Tapahtumia voi katsella lohko kerrallaan ja palvelu näyttää kaikki lohkoon sisältyneet Ether-transaktiot sekä älykkäillä sopimuksilla tapahtuneet transaktiot. Lisäksi sieltä voi myös katsoa sopimuskoodin sisällön. Palvelu näyttää sopimuskoodin Ethereumin omassa binäärimuodossa.

Tarjolla on kuitenkin kääntäjiä joilla binäärikoodista saadaan käännettyä alkuperäinen sopimuskoodi Solidity syntaksilla. Porosity tarjoaa tällaista palvelua sivustolla https://github.com/comaeio/porosity. Ethereumin sopimuskoodi on helposti tarkastettavissa ja analysoitavissa ja näin ollen siitä on myös mahdollista

etsiä puutteita, tai aukkoja, jotka mahdollisesti mahdollistavat sopimuksen käyttämisen myös alkuperäisen kehittäjän suunnitelmien vastaisesti.

Merkittävä esimerkki tästä saatiin vuonna 2016. The DAO, eli Decentralized Autonomous Organisation on projekti, jonka tarkoituksena on antaa yrityksen päätösvalta suoraan osakkeenomistajille ja jättää perinteinen johtoryhmä ja palkattu toimitusjohtaja kokonaan pois yrityksen hallinnosta. Tarkoitus on, että toisilleen tuntemattomat ihmiset voivat yhdessä sijoittaa yritykseen ja olla mukana päätöksen teossa. Kaikki toiminnot ovat läpinäkyviä ja todistettavissa sekä jäljitettävissä. (Ethereum 2018b.)

DAO keräsi joukkorahoituksella yli 100 miljoona US dollaria toukokuussa 2016 (Vigna 2016). Toukokuussa 2016 joukko tutkijoita tekivät julkaisun, jossa toivat esille mahdollisen haavoittuvuuden The DAO:n Ethereum-koodissa, jonka vuoksi sijoitukset voidaan varastaa, tai jäädyttää (Popper 2016.)

17.6.2016 The DAO joutui hyökkäyksen kohteeksi, jossa hyökkääjä hyödynsi sopimukseen jääneitä heikkouksia ja kaappasi n. 1/3 The DAO:n varoista, joiden arvo tuolloin oli n. 50 miljoonaa dollaria. Kaappaaja ei saanut varoja suoraan itselleen vaan ne siirtyivät tilille, jossa niitä sopimuksen sääntöjen mukaan pidettiin 27 päivää ennen vapauttamista. (Price 2016.)

Tänä aikana Ethereum-yhteisö keskusteli siitä mitä hyökkäyksen seurauksena tulisi tehdä. 20.7.2016 yhteisö päätyi siihen, että Ethereum lohkoketju haarautetaan ja uudessa haarassa kaapatut varat palautetaan alkuperäisille omistajille. Tätä voi hieman yksinkertaistaen verrata varmuuskopion palautukseen. Pohjaksi otetaan sellainen lohko historiasta, jossa varat ovat vielä alkuperäisellä omistajalla ja tämän pohjalta ratkaistaan seuraava lohko. Mikäli riittävä määrä louhijoita jatkaa louhimista näiden lohkojen jatkoksi, voidaan katsoa uuden validin lohkoketjun syntyneen.

Osa louhijoista oli kuitenkin sitä mieltä, että vaikka hyökkäys oli epäeettinen, niin kyse ei silti ollut kirjaimellisesti hakkeroinnista. Hyökkääjä vain noudatti erittäin kirjaimellisesti sopimuksen logiikkaa ja näin ollen voitiin katsoa varojen siirtyneen

omistajalta toiselle älykkään sopimuksen sääntöjen mukaisesti. Myös ajatus siitä, että lohkoketjun historiaan kajotaan, ei ollut kaikkien mieleen. Lohkoketjun keskeinen ajatus on kuitenkin siinä, että historia on muuttumaton ja tämän periaatteen uhraaminen ei sopinut kaikille. (Finley 2016.)

Niinpä osa louhijoista myös jatkoi alkuperäisen muuttumattoman lohkoketjun louhimista ja tämän seurauksena Ethereum lohkoketju on haarautunut kahdeksi. Alkuperäistä muuttumatonta ketjua kutsutaan Ethereum Classiciksi (ETC) ja haaraa, jossa The DAO:n kaapatut varat palautettiin alkuperäisille omistajille, kutsutaan pelkäksi Ethereumiksi (ETH).

Tässä tapauksessa siis alkuperäiset sijoittajat saivat virtuaalivaluuttansa takaisin. Hyökkääjä sai myös pitää kaappaamansa varat. Tosin hänen varat olivat nyt haarautumisen jälkeen Ethereum Classicissa, jonka kurssi on selvästi alhaisempi, kuin Ethereumin. Hyökkäyksen tekijää ei ole saatu jäljitettyä. (Leising 2017.)

Cornellin yliopiston tutkijan Philip Daianin mukaan syy The DAO:n hyökkäyksen mahdollisti osittain myös Solidity-ohjelmointikielen puutteet, eikä syy ollut kokonaan The DAO:n sopimuksen ohjelmoijien puutteellisuudet koodissa. Vaikka ohjelmoijat olisivat noudattaneet täydellisesti kaikkia Solidityn parhaita käytäntöjä, olisi sopimus silti ollut haavoittuvainen. (Quentson 2016.)

Solidity ohjelmointikieltä on kehitetty ja säännöllisesti päivitetty, mutta silti The DAO:n esimerkki kuvaa sitä, kuinka älykkäitä sopimuksia kehitettäessä on ymmärrettävä erittäin tarkasti käyttämiensä työkalujen mahdollisia haavoittuvuuksia ja myös oman tuottaman ohjelmakoodin haavoittuvuuksia.

5 Tietoturva

Ethereum-yhteisö on kerännyt parhaita käytäntöjä turvallisen älykkään sopimuksen kehittämiseen. Ohjeet löytyvät osoitteesta https://consensys.github.io/smart-contract-best-practices/general_philosophy/. Tässä on kuitenkin tiivistetysti muutamia huomioon otettavia asioita. Normaalissa

sovelluskehittämisessa testaaminen pyrkii siihen, että sovellus toimii halutusti. Älykkäiden sopimusten osalta pitäisi pystyä ottamaan myös huomioon, että jos sopimuskoodi toteuttaa halutut asiat, pystyykö sitä väärinkäyttämään tavalla jota sovelluskehittäjä ei ajatellut.

Varaudu epäonnistumiseen. Sopimuskoodin osalta on aina mahdollisuus siihen, että siihen on jäänyt virheitä. Tällaisen tilanteen toteutuessa sopimus tulisi olla mahdollista laittaa tauolle. Myös siirrettävien varojen osalta tulisi miettiä rajoitukset, kuinka paljon ja kuinka usein siirtoja voi tehdä. Sopimuksen päivittäinen virhetilanteessa tulisi myös suunnitella jo etukäteen.

Pidä sopimukset yksinkertaisina. Suunnittele sopimuksen logiikka yksinkertaiseksi. Laajemmissa kokonaisuuksissa jaa sopimus pienempiin paloihin ja pidä funktiot mahdollisimman pieninä. Käytä lohkoketjua vain niiltä osin kuin se ratkaisussasi on välttämätöntä.

Pysy ajantasalla. Tarkasta sopimukset säännöllisesti mahdollisten yleisesti tiedossa olevien bugilöydösten varalta. Päivitä sopimukseen liittyvät kirjastot ja työkalut. Ota käyttöön uudet ja hyödylliset turvallisuutta parantavat tekniikat.

6 Yksityiset lohkoketjut

Julkisen, avoimen lohkoketjun lisäksi on olemassa yksityisiä lohkoketjuja. Julkisen lohkoketjun varmennus perustuu aiemmin esitettyihin kryptograafisiin konsensusmenetelmiin kuten proof-of-work tai proof-of-stake. Kuka vain voi lukea lohkoketjun tapahtumia. Kuka vain voi tehdä transaktioita, jotka muuttavat lohkoketjun tilaa. Ja kuka vain myös pystyy ottamaan osaa lohkoketjun validointiin, kunhan vain noudattaa ennalta asetettuja sääntöjä. (Buterin 2015.)

Yksityinen lohkoketju taas perustuu siihen, että käyttäjien oikeuksia rajoitetaan. Käyttäjillä on rajattu oikeus muokata lohkoketjun sisältöä ja lukuoikeuskin voi olla rajattu. Tämä luonnollisesti edellyttää myös, että käyttäjät tunnistetaan. Luku- ja kirjoitusoikeus voivat olla rajattuja vain osaan lohkoketjun tiedoista. Yksityinen lohkoketju tarjoaa etuja, joihin avoin lohkoketju ei pysty: Yksityistä lohkoketjua ylläpitävä ja varmennuksesta huolehtiva taho pystyy halutessaan peruuttamaan transaktioita, jotka ovat virheellisiä, tai epärehellisiä. Ylläpitäjä pystyy myös tarvittaessa muuttamaan lohkoketjun sääntöjä. (Buterin 2015.)

Yksityisessä lohkoketjussa, jossa louhijat tunnetaan, ei myöskään ole mahdollisuutta aiemmin kuvatulle 51 % hyökkäykselle, jossa vihamielinen taho hankkisi niin paljon prosessoritehoa, että pystyisi dominoimaan lohkoketjua. Yksityinen lohkoketju voi myös olla nopeampi ja skaalautuvampi kuin julkinen lohkoketju, koska lohkoketjun varmentaminen ei tapahdu tuhansilla kilpailevilla koneilla, vaan se voidaan tehdä muutamalla louhintakoneella. Tosin proof-of-stake on lähtökohtaisesti huomattavasti proof-of-workia nopeampi ja skaalautuvampi ja tämän odotetaan kaventavan eroa yksityisen ja julkisen lohkoketjun välillä. (Buterin 2015.)

Siinä missä julkisen lohkoketjun tapauksessa joudutaan aina odottamaan useampi ratkaistu lohko, ennen kuin konsensusta voidaan pitää lopullisena, voidaan yksityisen lohkoketjun tapauksessa luottaa siihen, että konsensus syntyy automaattisesti lohkon ratkaisun jälkeen. (Buterin 2015.)

Yksityisen lohkoketjun konsepti toisaalta lähtökohtaisesti on hieman ristiriidassa niiden lupauksien kanssa, joita julkinen lohkoketju ratkaisee: Tieto ei olekaan enää ainakaan täysin läpinäkyvää ja yksityisen lohkoketjun tapauksessa ylläpitäjänä on yksittäinen, tai useampi taho, joiden hyvään tahtoon ja luotettavan tiedon ylläpitoon käyttäjän tulee luottaa. (Buterin 2015.)

Julkisen lohkoketjun luonteeseen kuuluu ideologiaa avoimuudesta ja vapautta painostuksesta. Kun käyttäjää ei tunneta, ei tämän tekemisiä voida rajoittaa. Eikä julkisen lohkoketjun osalla ole sellaista ylläpitäjää, tai kehittäjää, jota voitaisiin painostaa muokkaamaan lohkoketjun sisältämää dataa, tai lohkoketjun sääntöjä. Näin ollen julkisen lohkoketjun avulla on mahdollista tehdä palveluita, joiden käyttö on esimerkiksi valtion painostuksen ulottumattomissa. Yksityinen lohkoketju taas on käytännössä suljettu hajautettu tietokanta. (Buterin 2015.)

Yksityinen lohkoketju soveltuu yritysten, tai organisaatioiden väliseen käyttöön tilanteissa, joissa lohkoketjun tiedot halutaan pitää yritysten välisinä. Lohkoketjun avulla hajautettu tietokanta on kaikkien mukana olevien yritysten käytössä reaaliaikaisesti. Kun yritys A kirjoittaa lohkoketjuun tietoa, on se myös yrityksen B ja C käytössä välittömästi.

6.1 Quorum

JPMorgan Chase yrityksen esittelemä Quorum on heidän omien sanojen mukaisesti yrityksille suunnattu versio Ethereumista. Quorum pohjautuu Ethereumiin ja se mahdollistaa Ethereumista poiketen tiedon yksityisyyden sopimuksen, tai transaktion tasolla. Quorum on suunniteltu käyttöön, jossa nopeus ja sopimuksen nopea läpimeno ovat tärkeitä. Quorum on GPL/LGPL-lisensoitu, kuten Ethereum eli molemmat ovat ilmaisia käyttää. (Quorum 2018.)

Ohjelmointikielenä toimii Solidity. Truffle Framework, jota käytän oman sovelluksen kehityksessä, tukee myös Quorum-kehitystä (Truffle 2018). Quorum on suunniteltu kehittymään samaa tahtia Ethereumin kanssa. Se sisältää vain pieniä muokkauksia Ethereumin ytimeen, joten samat päivitykset ovat nopeasti saatavilla molemmille alustoille. (Lobban 2017a.)

Kun Ethereumissa kaikki louhijat louhivat kaikki transaktiot, niin Quorumissa kaikki louhivat julkiseksi määritellyt transaktiot, mutta yksityisten transaktioiden osalta julkiseen ketjuun tallennetaan ainoastaan kryptatun sopimuksen tiiviste. Varsinainen sopimuskoodi sijoitetaan lohkoketjun ulkopuolella sijaitsevaan komponenttiin nimeltä Transaction Manager. Sopimukselle tulee määritellä, kenelle sopimuksen sisältö on sallittu. Nämä sallitut tahot saavat sopimuksen sisällön Transaction Managerista dekryptatussa muodossa. (Lobban 2017b.)

Ainakin teoriassa Quorum vaikuttaa hyvältä alustalta. Kehittäjä jolle Ethereumin työkalut ovat tuttuja, tuntee olonsa kotoisaksi Quorumin kanssa. Quorumista, toisin kuin Ethereumista, löytyy kuitenkin erittäin vähän käyttäjäkokemuksia. Lisäksi Transaction Managerin osalta sopimuksen laatija joutuu luottamaan siihen, että Quorum on huolehtinut riittävästä tietoturvasta ja että sopimuksen tietoja ei väärinkäytetä. Tai ettei lohkoketjun sääntöjä myöhemmin muuteta niin, että ne ovat kehittäjälle epäsuotuisat.

6.2 Hyperledger

Hyperledger on Linux Foundationin, vuonna 2015 aloittama yhteistyöprojekti, jonka alla kehitetään useita lohkoketjuun perustuvia alustoja. Hyperledgerin kehityksessä on mukana yli 100 tunnettua yritystä, hyvin erilaisilta toimialoilta. Mukana on matkustukseen liittyviä yrityksiä kuten Airbus ja Daimler. IT-yrityksiä kuten IBM, Fujitsu, SAP, Huawei, Nokia, Intel ja Samsung, finanssialan yrityksiä kuten Deutsche Börse, American Express, JPMorgan, BBVA, BNP Paribas ja Wells Fargo. Mukana on myös lohkoketju startuppeja kuten Blockstream, Netki, Lykke, Factom ja Consensys. (Rosic 2017b.)

Hyperledger Burrow on Monaxin kehittämä suljettu lohkoketju, joka suorittaa sopimuskoodia Ethereumin standardin mukaisesti. Burrow käyttää konsensusmetodina Tendermintia https://tendermint.com/, jonka luvataan kykenevän tuhansiin transaktioihin sekunnissa. Konsensus syntyy tunnettujen validaattoreiden joukon kesken. Jokainen lohko on kryptograafisesti allekirjoitettu valtaosan validaattoreista toimesta. Konsensusmekanismissa ei ole samanlaisia

tuntemattomia validaattoreita, kuten Ethereumin proof-of-workissa. (Hyperledger Burrows 2018.)

Monax tarjoaa kehittäjille toimialakohtaisia kehittämiskittejä (SDK – Software Development Kit). Kehittämiskittien luvataan nopeuttavan kehitystä ja näihin säännöllisesti tulevat päivitykset tuovat mukaan uusia ominaisuuksia. Kehittämiskittejä on tarjolla finanssi- sekä vakuutusalalle ja logistiikkaan. (Monax 2018.)

Hyperledger Fabric on myös suljettu lohkoketju, jota mm. IBM käyttää tarjoamassaan lohkoketjujen sovellusalustassa. Älykkäitä sopimuksia kutsutaan Fabricissa chaincodeksi. Sitä voi kirjoittaa Go-ohjelmointikielellä. Javalle ja muille kielille tukea on luvattu tulevissa versioissa. Fabricissa ei ole sisäänrakennettua valuuttaa. (Hyperledger Fabric 2018.)

Hyperledger Fabric on modulaarinen järjestelmä, jossa konsensus ja käyttäjähallinta on toteutettu plug-and-play periaatteella joten käyttäjä voi valita itse mitä teknologioita haluaa käyttää. Hyperledger Fabricissa on mahdollista luoda kanavia, joiden tiedot jaetaan vain niiden kesken joilla on annettu oikeudet kyseiseen kanavaan. Kanavat tulevat tarpeeseen, kun kilpailevat yritykset ovat sopineet tiettyjen tietojen jakamisesta, mutta haluavat pitää osan tiedoista myös kilpailijoiden ulottumattomissa. (Hyperledger 2018b.)

Ethereumissa lohkoketjun ylläpito on hajautettu louhijoille, mutta Hyperledgerin tapauksessa lohkoketjua ylläpitävät yritykset yhteistyössä. Hyperledger Composer on työkalu, jolla voi esimerkiksi määritellä millaista omaisuutta lohkoketjussa voi vaihtaa. Composerilla voi lisäksi määritellä millaisia liiketoimintasääntöjä transaktioihin sisältyy. Lisäksi sen avulla voi määrittää osallistujat, näiden identiteetin ja hoitaa käyttöoikeusvalvonnan ja määrittää roolit jotka saavat oikeuksia suorittaa eri tyyppisiä transaktioita. (Hyperledger 2018a.)

Hyperledger Fabricin tapauksessa kehityksen voi aloittaa omalla koneella, joko lataamalla Hyperledger Fabricin lähdekoodin Githubista, tai IBM:n sertifioimat Hyperledger Fabric imaget Dockerhubista. Näihin IBM myös tarjoaa maksullista

tukea. Lisäksi IBM tarjoaa kehitykseen pilvipohjaisia vaihtoehtoja. IBM Bluemix Container Service mahdollistaa lohkoketjun lanseerauksen Docker kontin sisällä. Lisäksi IBM tarjoaa myös omaa lohkoketjualustaa, jota se kuvaa yrityskäyttöön suunnitelluksi alustaksi, jossa on helppo aktivoida ja ylläpitää turvallista verkkoa yritysten välillä. (Parzygnat & Gupta 2016.)

6.3 Corda

Corda on R3 yrityksen lohkoketjuteknologia, joka on suunnattu finanssialan yritysten käyttöön. R3 on yli 70:n maailman suurimman pankin yhteisyritys, joka vastaa Cordan kehityksestä. Suomessa toimivista pankeista mukana ovat ainakin Nordea, OP ja Danske Bank. (R3 2015). Cordan lähdekoodi löytyy Githubista https://github.com/corda/corda ja se on Apache 2.0 lisensoitu, eli lähdekoodi on vapaasti käytettävissä.

Corda, kuten Hyperledgerkin on suljettu lohkoketju, eikä siinäkään ole sisäänrakennettua valuuttaa. Cordassa tiedot eivät ole jaettuna koko systeemin kesken. Tiedot ovat jaettu vain niihin oikeutettujen tahojen kesken. Myös konsensus syntyy tietoihin oikeutettujen tahojen välillä, eikä koko systeemin laajuisesti. Sopimuskoodi kirjoitetaan JVM ohjelmointikielillä, eli mm. Java ja Kotlin. Cordassa sopimuksiin on mahdollista liittää lakitekstiä jota sopimuksen toteutuminen noudattaa. Näin sopimukset ovat yhteensopivia perinteisen lakijärjestelmän kanssa ja lakitekstiin voidaan viitata riitatapauksessa. (Corda 2018.)

7 Käytännön esimerkkejä

Elinkeinoelämän tutkimuslaitos on yhteistyössä Fortumin kanssa tutkimusraportin 'A Blockchain Application in Energy' (https://www.etla.fi/julkaisut/a-blockchain-application-in-energy/) joka esittelee energiasektorille suunnatun lohkoketjusovelluksen. Sovelluksen tarkoituksena on mahdollistaa kuluttajien ja piensähköntuottajien välinen kauppa ilman nykyisiä markkinatoimijoita. Sovellus on kehitetty Ethereum alustalle. (Etla 2017.)

Ascribe tuottaa taiteilijoille suunnattua palvelua, jossa taiteilija voi rekisteröidä teoksensa. Palvelun kautta rekisteröinti on seurattavissa ja vahvistettavissa ikuisesti. Teoksen voi myydä eteenpäin ja uuden omistajan tiedot tallennetaan palveluun. Teokselle voi muodostaa sertifikaatin, joka sisältää koko omistushistorian ja sertifikaatin voi halutessaan tulostaa. Palvelu toimii lohkoketjussa, palveluntarjoaja ei tosin kerro tarkempia teknisiä tietoja esimerkiksi käytetystä alustasta. (Ascribe 2018.)

Wave pyrkii siirtämään kansainvälisen toimitusketjun paperityöt lohkoketjuun. Näin dokumentit ovat suoraan jaettavissa koko ketjun kaikkien osapuolien kesken. Paperiset dokumentit ovat aina väärennettävissä, mutta lohkoketjuun tallennetut dokumentit ovat identtisiä kaikilla katselijoilla. Wave ei myöskään kerro tarkempia tietoja mille alustalle heidän lohkoketjusovellus on toteutettu. (Wavebl 2018.)

Augur on beta-vaiheessa oleva palvelu, joka tarjoaa lohkoketjun päälle rakennettua työkalua erilaisten ennusteiden luomiseen, joka toisaalta muistuttaa paljon perinteistä vedonlyöntiä. Kuka vain voi perustaa palveluun uuden kyselyn, johon haluaa ennusteen. Esimerkiksi kuka voittaa seuraavat presidentinvaalit, tai mikä maa voittaa jääkiekon maailmanmestaruuden. Tämän jälkeen muut kävijät voivat käyttää tuntemustaan aiheesta ja sijoittaa rahaa sille näkemykselle, jonka uskovat toteutuvan. Mikäli ennuste osuu oikeaan, saa sijoittamalleen rahalle tuottoa ja vastaavasti mikäli oma ennuste meni pieleen, menettää sijoittamansa rahat. Varat ovat sijoitettuna älykkäälle sopimukselle, joten takaisinmaksu

tapahtuu automaattisesti älykkään sopimuksen logiikan mukaisesti, kun ennustamisen elinkaari päättyy. (Augur 2018.)

Everledger pyrkii suojaamaan arvokasta omaisuutta tallentamalla lohkoketjuun tuotteen tunnistetietoja, historiaa ja omistajuussuhdetta. Everledger on tallentanut 1,6 miljoonan timantin tiedot lohkoketjuun. Timanteista on tallennettu tietoa niiden väristä, karaateista ja sertifikaatista, jonka ammattilainen on antanut tutkittuaan timantin yksilökohtaiset ominaisuudet. Everledgerin pyrkimys on luoda timanteista digitaalinen kopio. Palvelu on tarkoitus julkaista vuoden 2018 aikana kuluttajille ja tavoitteena on, että ennen pitkää kuluttaja voisi älypuhelimen avulla tarkastaa esimerkiksi jalokiven aitouden ja varmistua sen oikeasta omistajasta. Everledger on rakennettu Hyperledger alustalle. (Everledger 2018.)

8 Sovellus

Testasin älykkään sopimuksen kehittämistä ja kehittämiseen tarkoitettuja työkaluja käytännössä. Suunnittelin pienimuotoisen sovelluksen joka hyödyntää älykästä sopimista. Kyseessä on kuvitteellinen sovellus, jossa käyttäjäosapuolina ovat vakuutusyhtiö sekä vakuutusyhtiön sairauskuluvakuutettu asiakas. Vakuutusyhtiö tarjoaa vakuutusta, jossa asiakas voi saada alennusta hinnasta urheilemalla sovitun määrän kuukausittain.

Asiakas tallentaa urheiludataansa älypuhelimen applikaatiolla. Vakuutusyhtiö taas tarjoaa verkkosivuston, jossa nämä älypuhelimen applikaation tiedot voi tallentaa älykkäälle sopimukselle ja mikäli ennalta asetettu määrä urheilua ylittyy myöntää älykäs sopimus automaattisesti alennuksen vakuutusmaksusta, eikä vakuutusyhtiön työntekijän tarvitse manuaalisesti tarkastaa täyttyvätkö ehdot alennukseen.

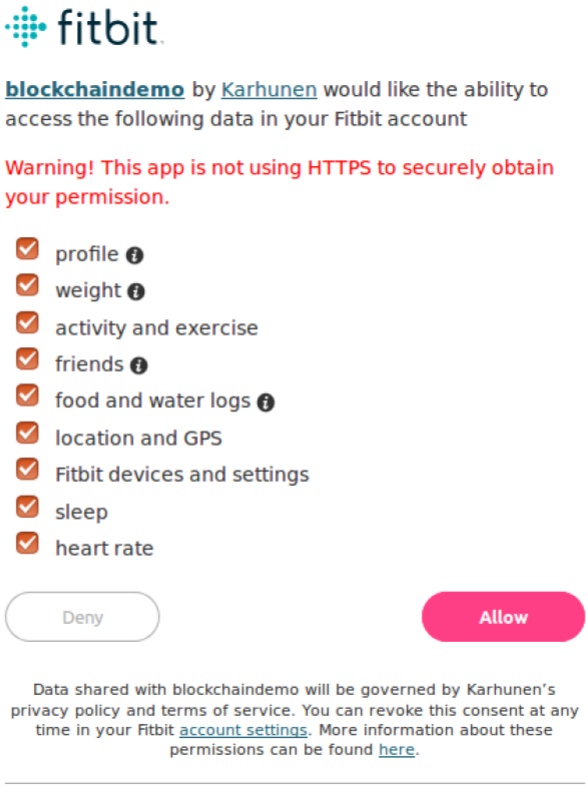
Älykkään sopimuksen tila on tallennettu lohkoketjuun luotettavasti ja sieltä sen voi aina myöhemmin tarkastaa tarpeen vaatiessa. Vakuutusyhtiön kannalta data on myös näin ollen luotettavampaa, koska käyttäjä ei voi suoraan ilmoittaa

liikuntamääriä vaan se tapahtuu suoraan urheilusuorituksia seuraavasta applikaatiosta älykkään sopimuksen välityksellä. Tämä helpottaa myös asiakkaan työtä, kun ei tarvitse erillistä omaa kirjanpitoa liikunnasta, tai manuaalisesti ilmoittaa liikuntamääriä esimerkiksi verkkopalvelun kautta. Sisällytän älykkääseen sopimukseen myös funktion, jolla vakuutusyhtiö voisi ohjelmallisesti tarkastaa onko asiakas oikeutettu saamaan alennusta vakuutusmaksusta.

Urheilusuoritusten seurantaan olen valinnut Fitbit sovelluksen, joka on saatavilla Android ja iOS puhelimiin ja Fitbit valmistaa myös aktiivisuusrannekkeita sekä älykelloja, jotka seuraavat käyttäjän urheilusuorituksia. Lisäksi Fitbit tarjoaa sovelluskehittäjälle dokumentoidun ohjelmointirajapinnan, josta käyttäjädataa voi hakea.

Käyttäjällä tulee olla käyttäjätunnus Fitbit-palveluun ja tämän täytyy myöntää käyttöoikeus kehittämälleni sovellukselle hakea hänen tietoja.

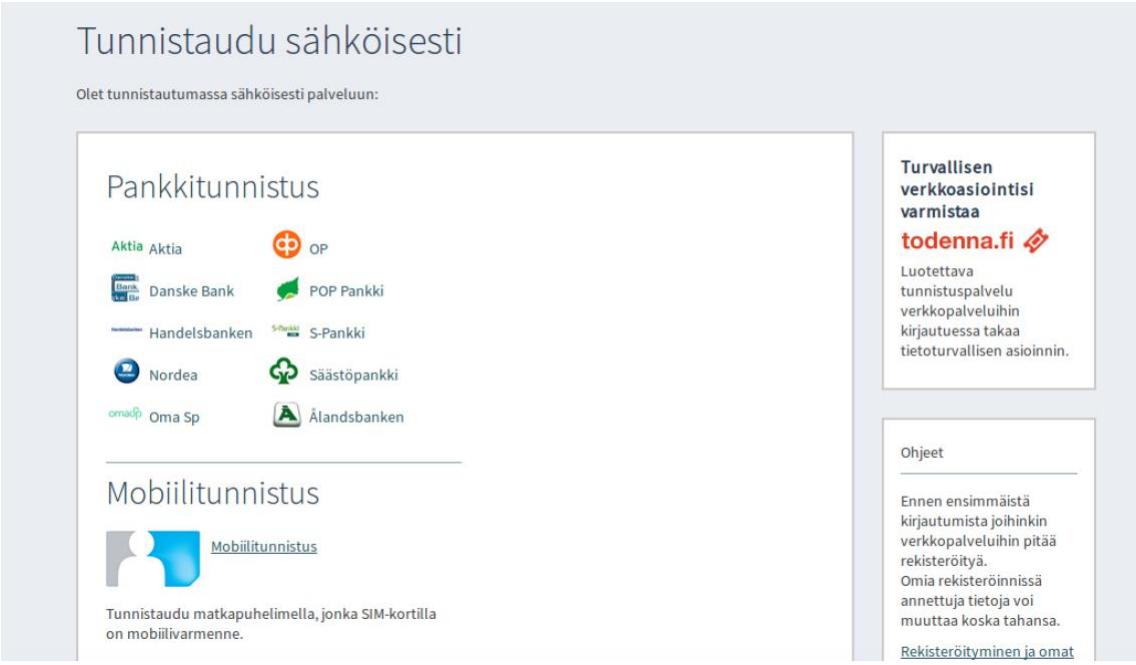
Verkkosivu, jossa käyttäjä antaa luvan käyttää Fitbit-dataansa. (kuva 2.)



Kuva 2. Kuvakaappaus Fitbit sovelluksesta.

Käyttääkseen palvelua tulee käyttäjän jokaisella käyttökerralla tunnistautua luotettavasti. Tämä tapahtuu pankkitunnuksilla ja käytän tähän minulle työelämästä tuttua todenna.fi-palvelua, joskin tässä tapauksessa vain palvelun testiympäristöä, joka toimii pankkien ilmoittamilla testitunnuksilla, jotka löytyvät verkosta http://docs.maksuturva.fi/fi/html/pages/4_2_1_verkkopankkien_testitunnukset_print.html

Todenna.fi-sivu, jossa käyttäjä tunnistautuu palveluun. (kuva 3.)



Kuva 3. Kuvakaappaus todenna.fi sivusta

9 Älykkään sovelluksen kehittäminen

Kehitysympäristönä käytin Linux-käyttöjärjestelmää. Valitsin älykkään sovellukseni alustaksi avoimen Ethereumin. Älykkään sovelluksen toteutukseen käytin Truffle frameworkia (http://truffleframework.com/). Truffle tarjoaa kehitys- ja testausympäristön Ethereumiin ja on lisäksi hyvin dokumentoitu. Ohjelmointikielenä Ethereumin älykkäissä sovelluksissa käytin Solidium-ohjelmointikieltä. Käyttäjän selainsovelluksen osalta käytin palvelinpuolella node.js:aa. Selainsovelluksen teen käyttämällä HTML:aa, Vue.js-komponentteja, Javascriptia ja CSS-frameworkia nimeltä Bulma. Bulman avulla selainsovelluksesta saa helposti responsiivisen, eli se soveltuu myös kännykän näytölle.

Seuraavia ohjeita seuraamalla voit alustaa älykkään sovelluksen ja testikäyttöön tarkoitetun TestRPC ohjelman, joka simuloi lohkoketjua. Älykkään sovelluksen kehittäminen alkaa Truffle frameworkin avulla. Linuxissa asennus on helpointa Node package managerin avulla komentoriviltä:

```
npm install -g truffle
```

Tämän jälkeen luodaan tyhjä kansio uudelle sovellukselle ja Trufflen avulla alustetaan sovellus luodussa kansiossa:

```
truffle init
```

Tämä komento luo esimerkkiprojektin, jossa on mukana esimerkkisopimukset joiden avulla voi myös ensikertalaisena tutustua ohjelmointiin. Tässä tapauksessa poistin esimerkkisopimukset kokonaan pois. Seuraavaksi loin uuden sopimuksen vakuutussopimukselle nimeltä insuranceContract. Käytin yleisesti hyväksi todettua tapaa nimetä sopimukset, funktiot ym. englanniksi. Uuden sopimuksen luominen onnistuu komennolla:

```
truffle create contract insuranceContract
```

Tämän jälkeen sopimuskoodia voi muokata haluamallaan tekstieditorilla. Käytin Sublime2 editoria, johon saa tuen Solidityn syntaksille, eli editori osaa värein korostaa koodista esimerkiksi funktiot, parametrit jne. joka helpottaa ohjelmoijan työtä. Tuen saa asennettua Sublime2 editorissa avaamalla tools -> command palette ja kirjoittamalla Package control (package control täytyy olla asennettuna, ohje löytyy osoitteesta https://packagecontrol.io/installation). Tämän jälkeen kirjoita command paletteen Install package ja Sublime2 listaa tarjolla olevat paketit. Hakusanalla Ethereum löytyy oikea paketti, josta lisätietoa löytyy sivulta https://packagecontrol.io/packages/Ethereum. Asennuksen jälkeen Sublime2 täytyy vielä käynnistää uudelleen, jotta tuen Soliditylle saa käyttöön.

Sovelluksella voi olla kahden tyyppisiä funktioita: Transaktioita sisältäviä funktioita, tai ”vain-luku” funktioita. Funktiot jotka ovat vain luettavissa, merkitään constant avainsanalla. Nimensä mukaisesti nämä funktiot ovat tarkoitettu vain sopimuksen tilan lukemiseen ja niissä ei voi tapahtua funktioita, jotka muuttaisivat sopimuksen tilaa. Nämä funktiot eivät myöskään kuluta gasia, toisin kuin transaktioita sisältävät funktiot, jotka siis ovat funktioita joissa sopimuksen tilaa

muokataan ja nämä funktiot voivat kuluttaa runsaastikin prosessoritehoa, niin niiden suorituksesta myös koituu kuluja gasin muodossa.

9.1 TestRPC

Älykkään sopimuksen kehittämistä varten asensin koneelleni testrpc-clientin (https://github.com/ethereumjs/testrpc), jolla pystyin testaamaan älykkäitä sopimuksia omalla koneellani, enkä tarvinnut siirtää sopimuksia julkiseen lohkoketjuun. Testrpc on node.js pohjainen client testaamiseen ja kehittämiseen ja se simuloi Ethereumin toimintaa paikallisesti. Etuna on myös se, että sopimuksen louhiminen tapahtuu välittömästi, eikä tarvitse odottaa normaalien lohkoketjun mukaisten sääntöjen mukaista louhimisaikaa. Testrpc:n asennus tapahtuu noden package managerilla antamalla komento:

```
$ npm install -g ethereumjs-testrpc
```

Testrpc käynnistetään komennolla:

```
$ testrpc
```

Käynnistys kannattaa tehdä terminaali-ikkunassa jonka jättää auki kehityksen ja testaamisen ajaksi. Komento luo oletuksena 10 uutta osoitetta, joille on määritelty simuloidut varat, nämä kaikki ovat tietenkin keinotekoisia tilejä ja varoja, joten riskiä varojen kadottamisesta ei ole. Oletuksena sovellus kuuntee porttia 8545, mutta esimerkiksi tilien määrää ja oletusporttia voi muuttaa antamalla testrpc komennolle käynnistysparametreja. Nämä parametrit ovat listattuna testrpc:n sivuilla.

Älykäs sopimus voi olla näennäisesti yksinkertainen toteuttaa ja koodimäärä sopimuksella voi jäädä vähäiseksi, pari funktiota voi jo riittää. Kuitenkin sopimuksen luontiin ja suunnitteluun kannattaa käyttää huolellisuutta, jotta siitä tulisi tietoturvallinen ja toisaalta se ei kuluta turhaan prosessoritehoa, koska siitä koituu myös turhia kuluja.

Sovellustani varten tarvitsin tallettaa sopimukselle käyttäjäkohtaisesti ja kuukausikohtaisesti tiedon siitä, onko asiakkaalla oikeus alennukseen vai ei. Lisäksi tarvitsin funktion, jolla vakuutusyhtiö voi tarkastaa asiakkaan oikeuden alennukseen. Minun täytyi pystyä tallettamaan sopimukselle jokaista asiakasta varten joku yksilöivä tieto. Koska Ethereumin luonteeseen kuuluu se, että kaikki sopimuksen data on julkista, en halunnut käyttää tunnistamiseen henkilötunnusta, tai käyttäjän nimeä. Käytin tässä tapauksessa asiakasnumeroa, joka luodaan asiakkaalle tämän rekisteröityessä sovellukseen. Mikäli kyseessä olisi oikea vakuutusyhtiön käyttöön tuleva sovellus, olisi ehkä vielä hyvä luoda tätä tarkoitusta varten joku muu tunniste kuin asiakasnumero, ainakin mikäli asiakasnumeroa hyödyntämällä voisi syntyä riski asiakkaan vakuutustietojen hyväksikäytöstä.

Tämä on seikka, jota julkiseen Ethereumiin toteutettavassa sopimuksessa kannattaa miettiä tarkoin. Henkilötunnuksen voi kyllä tallentaa sopimukselle kryptattuna, jolloin kukaan ei suoraan näe sopimukselta henkilötunnuksia, mutta toisaalta jos joku on saanut henkilötunnuksen haltuun, voisi hän tarkastaa sopimukselta tämän henkilön tilanteen. Kryptauksen osalta kannattaa myös huomioida se, että prosessoritehon jatkuva lisääntyminen yleensä johtaa siihen, että nykyisillä tietokoneilla tietoturvalliset algoritmit ovat todennäköisesti tulevaisuudessa murrettavissa jopa kotikoneilla ja älykkäällä sopimuksella tieto on tallennettu ainakin teoriassa ikuisesti.

Tässä kyseisessä esimerkkitapauksessa tieto siitä onko asiakas oikeutettu alennukseen vai ei, ei ole ehkä kovin vaarallinen asia tultuaan julki, mutta lähtökohta on oltava se, että kaikki asiakkaan ja vakuutusyhtiön väliseen suhteeseen liittyvät asiat pysyvät vain niiden osapuolien tiedossa joilla on oikeus tarkastella tietoja. Toisaalta henkilötunnus on joitakin harvinaisia poikkeustapauksia lukuun ottamatta sellainen pysyvä tunniste, joka voidaan aina tarkastaa ja löytää sillä oikea tieto sopimukselta. Asiakasnumerot taas voivat vaihtua ja tällöin on oltava jossain rekisteri siitä, että mikä älykkäältä sopimukselta löytyvä tunnistetieto kuuluu kullekin asiakkaalle. Mikäli tämä rekisteri katoaa, on kaikki älykkäällä sopimuksella tieto tämän jälkeen käyttökelvotonta.

Koska sopimukselle luonnollisesti täytyy pystyä tallentamaan useamman kuin yhden asiakkaan tietoja, sijoitetaan tiedot sopimukselle jonkun tietorakenteen mukaisesti. Tämän tietorakenteen valintaan kannattaa kiinnittää huomiota, koska mikäli tiedot esimerkiksi olisivat sopimuksella jonomuodossa ja transaktiofunktiossa joutuisimme iteroimaan jonoa läpi, siitä tulisi raskasta jonon kasvaessa isoksi. Tästä myös seuraisivat suuremmat transaktiokustannukset gasin muodossa.

Tällaisessa tapauksessa parempi tietorakenne olisi Solidityn mappings, jota Solidity tukee. Tällöin tietorakenteeseen voisi suoraan tallentaa sijainnin josta kyseinen haettu tieto löytyy. Esimerkiksi jos haluaisin omassa esimerkkisovelluksessa tarkastaa, onko asiakkaalla kyseiseltä kuukaudelta jo merkintä sopimuksella, löytäisin map tietorakenteesta suoraan asiakasnumerolla eikä sopimuksella tarvitsisi iteroida läpi kaikkia tietoja.

Lisäksi halusin sopimuksellani rajoittaa sitä, kuka voi kutsua funktioita. Määrittelin sopimuksella omistajan jonka osoite tallennetaan sopimukselle aloitusvaiheessa. Tämän jälkeen vain tämän osoitteen omistaja voi kutsua sopimuksen transaktionaalisia funktioita. Halusin myös säilyttää mahdollisuuden muuttaa tätä osoitetta myöhemmin, joten mahdollistan funktiolla sopimuksen omistajan osoitteen muokkaamisen. Tässä tapauksessa sopimuksella voi olla vain yksi omistaja kerrallaan. Funktiossa jossa sopimuksen nykyinen omistaja voi luovuttaa käyttöoikeuden toiselle taholle, nykyinen omistaja antaa uuden omistajan osoitteen. On huomioitava, että jos tuo osoite menee väärin, menetetään käyttöoikeus sopimukseen lopullisesti, koska omistajaa voi muuttaa enää vain se taho jonka osoite juuri sopimukselle merkittiin virheellisesti.

9.2 Sopimuksen kääntäminen

Kun sopimuksen koodi on kirjoitettu, se käännetään trufflen avulla komennolla:

```
truffle compile
```

Tämä kertoo heti, jos sopimuskoodi ei käänny. Pääosin kääntäjä antaa myös selvän vihjeen mikä sopimuksella aiheutti virheen, mutta joskus kävi niin, että kääntäjä antoi jotain omaa sisäistä virhelokia ja piilotti tekemäni sopimuksen virheen ja nämä virheet vain täytyi itse löytää sopimukselta. Esimerkiksi puuttuvat puolipisteet aiheuttivat tämänkaltaisia tilanteita. Mikäli virheitä ei tule, on sopimus käännetty onnistuneesti.

9.3 Sopimuksen migraatio

Trufflen luoma rakenne sisältää kansion nimeltä migrations. Migraatiot ovat Javascript-tiedostoja, jotka auttavat auttavat lähettämään älykkään sopimuksen Ethereum-verkkoon. Näiden avulla lähetys voidaan tarvittaessa vaiheistaa. Omassa toteutuksessa poistin Trufflen luomat esimerkkitiedostot ja loin vain yhden Javascript-tiedoston, jonka sisältö on seuraavanlainen:

```
var insuranceContract =
artifacts.require("./insuranceContract.sol");

module.exports = function(deployer) {
  deployer.deploy(insuranceContract);
};
```

Halutessani lähettää oman sopimukseni paikallisesti pyörivään testRPC verkkoon on varmistuttava, että testRPC on päällä. Tämän jälkeen annoin komennon:

```
truffle migrate
```

Onnistuneen migrate komennon jälkeen komentoriville tulee tekstit:

```
Using network 'development'.

Network up to date.
```

Tämä jälkeen voin tarkastaa testRPC:n konsolista tapahtumat, esimerkkitapahtumani näyttää tältä:

```
Transaction:
0x80b8c52bebfc3e7e6dd593880003166b70896655cea9dc61444f05fc7
c69a3c3
  Contract                                          created:
0x6ed98c4b6abc3af206b5009f05f132a403114ade
  Gas usage: 0x034e78
  Block Number: 0x01
  Block Time: Sat Nov 18 2017 04:45:33 GMT-0800 (PST)
```

Mikäli sopimuksen migraatio on jo kerran tehty ja sitä muokataan, niin komento tulee antaa muodossa:

```
truffle migrate --reset
```

Muuten migraatio katsotaan jo tehdyksi ja uusia muutoksia ei huomioida. Tämän tarkoituksena on, että älykkään sopimuksen kokonaisuutta voidaan kehittää inkrementaalisesti ja migraatio tehdään vain uusille sopimuksille ja olemassa oleville sopimuksille vain tarpeen mukaan.

9.4 Sopimuksen testaaminen

Testasin sopimusta Trufflen konsolityökalulla, joka avataan komennolla:

```
truffle console
```

Oletuksena truffle console käyttää development verkkoa, mutta tämän voi muuttaa käyttämällä --network parametria käynnistyksessä.

```
var insuranceContract =
insuranceContract.at(insuranceContract.address)

truffle(development)>
insuranceContract.addDiscountStruct(12345, 01, true)

{ tx:
'0x7c577a5e6b8601da2caa154b0f1291f9e50ec4f5da905a89fb59b073
d848f69a',
  receipt:
   { transactionHash:
'0x7c577a5e6b8601da2caa154b0f1291f9e50ec4f5da905a89fb59b073
d848f69a',
         transactionIndex: 0,
         blockHash:
'0x40a6830544606098e327d5ff982c6560777bf7d9e147b25b74f105ca
d58a9886',
         blockNumber: 2,
         gasUsed: 82597,
         cumulativeGasUsed: 82597,
         contractAddress: null,
         logs: [] },
  logs: [] }
```

Kutsumalla funktiota ja antamalla parametrina samat arvot kuin alennustietoa lisätessä, vastaa älykäs sopimus konsoliin: true. Jos taas asiakasnumeroa, tai kuukautta vaihtaa, tulee vastauksena konsoliin: false.

```
insuranceContract.checkDiscount(12345, 01)
```

Testasin myös funktiota, jolla voin muuttaa sopimuksen omistajaa antamalla parametriksi mielivaltaisen osoitteen:

```
truffle(development)>
insuranceContract.changeOwner(3242535325)
```

Funktio toimii ja sain vastauksena tiedon onnistuneesta transaktiosta. Jos tämän jälkeen yritän kutsua addDiscountStruct funktiota, sain konsoliin vastauksen epäonnistuneesta funktiosta. Tällöin voin todeta, ettei sopimukselle voi tallentaa tietoja kuin sopimuksen omistaja. Luonnollisesti en myös enää pystynyt kutsumaan sopimuksen changeOwner funktiota.

Sen sijaan checkDiscountia pystyin edelleen kutsumaan ja parametreistä riippuen vastauksena tulee true tai false, koska checkDiscountia on avoin muillekin kuin sopimuksen omistajalle. Testasin sopimusta manuaalisesti tallentamalla sinne useampia asiakkaita ja eri näiden alennustietoja eri kuukausille ja hakemalla näitä alennustietoja ja näin testaamalla sopimus toimi.

Sopimukselle ei ole sisällytetty tietojen validointia, ja sinne voi syöttää myös vaikka kuukausi kohtaan numeron 13, joka ei loogisesti vastaa enää mitään kuukautta. Tämän validoinnin myös hoitaa sovelluksessa, joka kutsuu sopimusta ja näin samalla säästämme prosessointitehoa ja sopimus kuluttaa näin ollen vähemmän gasia. Toinen huomioitava asia sopimuksella on, että sopimus palauttaa asiakkaan tietoja hakiessa vastauksena falsen myös silloin kun vastaavalla asiakasnumerolla, tai kuukaudella ei ole tallennettu tietoa. Eli tätä kautta ei pysty varmistumaan siitä, onko tietoa sopimukselle koskaan todellisuudessa syötetty.

9.5 web3.js

Web3.js on Javascript kirjasto, jolla web-sovelluksen saa integroitua älykkääseen sopimukseen. Kirjaston avulla voi käyttää älykkään sovelluksen funktioita suoraan käyttöliittymältä. Dokumentaatio löytyy osoitteesta https://github.com/ethereum/wiki/wiki/JavaScript-API. Kirjastoa voi käyttää

suoraan HTML:n joukossa selaimessa, tai node.js palvelimella. Dokumentaatio olisi voinut olla selkeämpi ja kattavampi, mutta sain kirjastoa hyödyntämällä kaikki haluamani toiminnot tehtyä, eli tallennettua älykkäälle sopimukselle dataa ja lukemaan dataa sopimukselta. Kirjastoa hyödyntäen toteuttamani funktiot löytyvät lähdekoodista, josta tarkemmin seuraavassa luvussa.

9.6 Sovelluksen asennusohje

Älykäs sopimus on vain osa tätä sovellusta. Koska käyttöliittymät, kirjautuminen, API-kutsut, tai tietokannat eivät sinänsä ole enää tämän opinnäytetyön aihepiiriä, en käy niitä tarkemmin läpi, mutta lähdekoodi on saatavilla ja sovellusta voi testata itse.

Tämän ohjeen avulla sovellustani voi helposti testata itse. Käytin itse kehitykseen Linux Ubuntua, joten tämä ohje on tehty sen mukaan. Sovellus käyttää MongoDB Community Edition tietokantaa, joka täytyy olla asennettuna. Asennusohjeet löytyvät https://docs.mongodb.com/manual/installation/ osoitteesta. Sovelluksen käyttöönottoa varten tulee myös olla asennettuna versionhallintatyökalu Git (asennusohje https://git-scm.com/book/en/v2/Getting-Started-Installing-Git) ja Node Package Manager (asennusohje https://www.npmjs.com/get-npm).

Lähdekoodi sovellukseeni löytyy https://gitlab.com/samikarhunen/blockchain.git Tee tyhjä kansio sovellusta varten ja kopioi lähdekoodit git clone https://gitlab.com/samikarhunen/blockchain.git -komennolla. Käyttämäni tietokannan määritykset löytyvät projektin juuresta tiedostosta config.js. MongoDB käyttää oletuksena porttia 27017, joka asetuksissa on määritelty. Jos kuitenkin tietokantayhteys ei toimi, tarkasta että MongoDB:n ja asetustiedoston asetukset täsmäävät. Asetustiedostossa tietokannan nimeksi on määritelty testibase, joten luo MongoDB:hen samanniminen kanta, tai jos haluat käyttää muun nimistä, muokkaa asetustiedostoa.

Tämän jälkeen anna uuden kansion juuressa komento ”sudo npm install” joka asentaa automaattisesti kaikki sovellukseni kaipaamat riippuvuudet. Tämän jälkeen sovellus käynnistetään antamalla komento ”node eka.js”. Asenna ja

käynnistä myös TestRPC, ohjeet löytyvät tämän opinnäytetyön TestRPC-luvusta. Älykäs sopimus löytyy kansiosta /smartcontract ja sopimus tulee kääntää ja migratoida. Näihin löytyvät ohjeet tämän opinnäytetyön luvuista sopimuksen kääntäminen ja sopimuksen migraatio.

Tämän jälkeen sovellus on valmis käytettäväksi paikallisesti. Alkuun pääset kirjoittamalla selaimeen osoitteen localhost:8080/auth. Täältä pääset kirjautumaan palveluun. Käytä verkkopankkikirjautumisessa pankkien testitunnuksia http://docs.maksuturva.fi/fi/html/pages/4_2_1_verkkopankkien_testitunnukset_print.html. Tämän jälkeen sinun täytyy valtuuttaa sovellus lukemaan Fitbit-sovelluksen tietojasi, eli tarvitset Fitbit-tilin ja kirjautumalla voit antaa oikeuden sovelluksen lukea tietojasi.

Tämä testisovellus ei tallenna mitään dataa koneesi ulkopuolelle ja älykäs sopimuskin pyörii vain testiversiona omalla koneellasi, joten tietoja ei luovuteta ulkopuolisille. Valtuutuksen jälkeen sovellus palaa perusnäkymään, jossa sovellus ensin lukee edeltävän kokonaisen kuukauden askelmääräsi Fitbit-sovelluksen tallennusten mukaisesti. Sen jälkeen sovellus automaattisesti tallentaa tiedon älykkäälle sopimukselle. Tämän jälkeen sovellus myös lukee automaattisesi tiedon älykkäältä sopimukselta oletko edellisen kuukauden osalta oikeutettu alennukseen. Alennuksen rajaksi on sopimuksella määritelty 7500 askelta päivässä, eli 225000 kuukausittaista askelta oikeuttavat alennukseen.

Sovellus on erittäin käyttäjän näkökulmasta erittäin yksinkertainen, mutta se toteuttaa seuraavat asiat:

1. Vahva kirjautuminen pankkitunnuksilla
2. Integraatio Fitbit sovelluksen kanssa
3. Tiedon tallentaminen ja lukeminen älykäälle sopimukselle
4. Käyttöliittymä asiakkaalle

10 Pohdintaa lohkoketjujen tulevaisuudesta

Tulevaisuuden ennustaminen on vaikeaa ja lohkoketjuteknologian hyödyllisyydestä ja yleistymisen todennäköisyydestä löytyy mielipiteitä äärilaidasta toiselle.
Yksi näkemys on Forbesin artikkelista ”From Tech tech titans to blockchain billionaires”. Artikkelissa ennustetaan, että blockchain on teknologiana nyt kuin internet oli vuonna 1994. Tuolloin Jeff Bezos perusti nykyään hurjan menestyksekkään Amazon verkkokaupan ja esimerkiksi Sears, yli 100 vuotta vanha kivijalkaketju jäi tuolloin sijoilleen pitäen kiinni vanhoista tavoista toimia ja nykyään ketju kamppailee konkurssin partaalla. Vain yritykset, jotka ovat mukana uuden lohkoketjuteknologian käyttöönotossa, ovat kilpailukykyisiä tulevaisuudessa.

Tehdessäni tätä opinnäytetyötä luin paljon vastaavan tyyppisiä artikkeleita, joissa pidettiin lohkoketjua mullistavana teknologiana, ja lueteltiin toimialoja jotka tulevat hyötymään teknologiasta. Kuitenkin usein jäi myös sellainen tunne, ettei artikkeleissa pystytty ihan tarkkaan nimeämään ongelmaa, jonka lohkoketju ratkaisee. Jos esitettiin ongelma, niin ei perusteltu, että miksi, tai miten lohkoketju ratkaisee ongelman.

Kai Stinchcomben artikkelissa ”Ten years in, nobody has come up with a use for blockchain” (Stinchcombe 2017) esitetään pessimistisemmin, että vaikka Bitcoinin ja lohkoketjuteknologian esittelystä alkaa olla kymmenen vuotta, ei teknologialle ole löytynyt kunnollista käyttökohdetta. Bitcoinin esittelyn myötä esitettiin, että kukaan ei jatkossa haluaisi maksaa esimerkiksi luottokorttiyhtiölle mitään rahan välittämisestä, koska se hoituisi ilmaiseksi virtuaalivaluutalla, joka olisi myös valtion sääntelyn ulottumattomissa. Kuitenkin käytäntö on osoittanut, että kuluttaja on valmis maksamaan luottokorttiyhtiön luotettavuudesta.

Luottokorttiyhtiön tarjoama varmuus esimerkiksi siitä, että mikäli myyjä ei pysty toimittamaan tuotetta, saa maksaja rahansa takaisin on kaivattu ominaisuus. Bitcoinin salausavaimet ovat käyttäjän vastuulla. Mikäli salausavaimet katoavat,

tai ne joku onnistuu varastamaan ne tietomurron yhteydessä, ei ole sellaista tahoa joka korvaisi käyttäjän menetykset. Perinteinen pankki vastaa säilyttämistään varoista ja on valtaosalle mieluisampi paikka säilyttää varojaan. Artikkelissa kiinnitetään huomiota siihen, että usein lohkoketjuratkaisuissa pyritään etsimään ratkaisua siihen, kuinka esimerkiksi rahaliikenne henkilöltä A henkilölle B pyritään järjestämään ilman luotettua tahoa välissä, tai yritetään tuoda ratkaisu jonkunlaiseen rekisterin ylläpitämiseen ilman, että rekisteriä kuitenkaan ylläpitäisi mikään luotettu, tunnettu taho. Käytännössä kuitenkin luotetuista tahoista koituvat kulut koetaan näistä koituvia hyötyjä pienemmiksi.

Oma ajatukseni lohkoketjun tulevaisuudesta on, että teknologiaa hyödynnetään jatkossa enenevässä määrin, mutta teknologia ei todennäköisesti aiheuta sellaista anarkistista murrosta, missä teknologian avulla voitaisiin ohittaa viranomaiset, pankit ja lakimiehet. Sen sijaan markkinoille tulee sekä uusia toimijoita, jotka hyödyntävät lohkoketjua liiketoiminnassaan, mutta myös perinteiset toimijat hyödyntävät lohkoketjua. Tästä oiva esimerkki on, että vaikka Bitcoinia on ennustettu pankkien kaatajaksi, niin todellisuudessa pankit ovatkin omaksumassa lohkoketjuteknologiaa omaan käyttöön, kuten Cordasta kertovassa luvussa käsiteltiin.

Tärkein kysymys lohkoketjun käyttöönottoa pohdittaessa on se käytännön ongelma, mitä lohkoketjulla halutaan korjata. Ja onko lohkoketju ainoa ja paras ratkaisu kyseiseen ongelmaan. Julkisen lohkoketjun etu on, että tieto on siellä hajautetusti ja väärentämättömästi tallennettuna. Julkisuus voi toisaalta aiheuttaa paljon ongelmia, koska hyvin usein liiketoimintaan liittyy halu pitää tietoa vain tiettyjen tahojen saatavana. Jos ratkaisu nimenomaan edellyttää pitkäaikaista, luotettavaa tietojen säilyttämistä eikä julkisuus ole ongelma, on julkinen lohkoketju hyvä ratkaisu.

Jos julkiseen lohkoketjuun usein mielikuvissa liitetään juuri anarkistia mielikuvia siitä, kuinka ihmiset tulevaisuudessa hoitavat rahansiirron, sopimukset ja rekisterien ylläpidot suoraan ilman välikäsiä, niin yksityinen lohkoketju on arkipäiväisempi työkalu, joka on suunniteltu yrityselämän tarpeisiin.

Yksityinen lohkoketju tarjoaa mielestäni vaihtoehdon nykyisille hajautetuille tietokannoille. Siinä tapauksessa tahot tuntevat toisensa, joten myös perinteisillä tietokantateknologioilla on mahdollista rakentaa useamman yrityksen kesken jaettu tietovarasto, joka on myös hajautettu useaan paikkaan. Mikäli olisin perustamassa useamman yrityksen yhteiseen käyttöön tulevaa tietokantaa, joka olisi tarpeen olla hajautettu, tutkisin soveltuuko yksityinen lohkoketju omaan käyttötarkoitukseen. Yksityinen lohkoketju voi tuoda helpotusta integraatioiden tekemiseen, koska kaikilla tahoilla on sama tieto reaaliaikaisesti käytössään. Nykyisissä tietokantaratkaisuissa voi olla esimerkiksi niin, että tiedot synkronoidaan vain kerran vuorokaudessa.

Osana opinnäytetyötä toteuttamastani sovelluksesta voin todeta, että työkalut ovat toimivia ja verkosta löytyy hyvin dokumentaatiota ja apuja kehittämiseen. Tuntui, että siinä missä vaikka Java-sovelluskehitykseen on vuosien saatossa vakiintunut paljon hyviä käytäntöjä, niin lohkoketjusovelluksen kehittämisen parhaat käytännöt odottavat vielä laajempaa kypsymistä. Ja kokemusta ja rohkeutta kehittäjiltä ja konseptoijilta.

Kokonaisuudessaan lohkoketjusovelluksista jäi positiivinen kuva ja uskon käytännön toteutuksia tulevan jatkuvasti enemmän, kun teknologia arkipäiväistyy ja sen mahdollistamat uudet liiketoimintamallit löytyvät.

Lähteet

Allison, I. 2016. Microsoft adds Ethereum language Solidity to Visual Studio. http://www.ibtimes.co.uk/microsoft-adds-ethereum-language-solidity-visual-studio-1552171. 31.1.2018

Ascribe. 2018. Ascribe. https://www.ascribe.io/ 12.2.2018

Augur. 2018. Augur. http://www.augur.net/ 12.2.2018

Bitcoinwiki. 2015. Block Hashing Algorithm. https://en.bitcoin.it/wiki/Block_hashing_algorithm. 6.12.2017

Bitcoinwiki. 2017. Controlled Supply. https://en.bitcoin.it/wiki/Controlled_supply 6.12.2017

Blockexplorer. 2017. Bitcoin Block #0. https://blockexplorer.com/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f. 30.1.2018

Brito, J & Castello A. 2013. Bitcoin. A Primer For Policymakers. s. 19

Browne, R. 2017. Big Transaction Fees Are A Problem For Bitcoin – But There Could Be A Solution. https://www.cnbc.com/2017/12/19/big-transactions-fees-are-a-problem-for-bitcoin.html. 30.1.2018

Buterin, V. 2015. On public and private blockchain. https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/. 31.1.2018.

Castor A. 2017. A (Short) Guide To Blockchain Consensus Protocols. https://www.coindesk.com/short-guide-blockchain-consensus-protocols/ 6.12.2017

Coindesk. 2017. Bitcoin (USD) Price. https://www.coindesk.com/price/ 6.12.2017

Corda. 2018. Contracts. https://docs.corda.net/key-concepts-contracts.html. 31.1.2018.

Disparte D. 2017. From Tech Titans To Blockchain Billionaires. https://www.forbes.com/sites/dantedisparte/2017/08/30/from-tech-titans-to-blockchain-billionaires/#2371b22b2428 30.1.2018

Ethereum. 2018a. FAQ. https://www.ethereum.org/ether. 31.1.2018

Ethereum. 2018b. How to build a democracy on the blockchain. https://ethereum.org/dao. 31.1.2018

Ethereum Foundation. 2018. Foundation Team. https://www.ethereum.org/foundation 30.1.2018

Elinkeinoelämän Tutkimuslaitos ETLA. 2017. Etlan ensimmäinen koodi julki: energia-alalle älykkäitä sopimuksia. https://www.sttinfo.fi/tiedote/etlan-ensimmainen-koodi-julki-energia-alalle-alykkaita-sopimuksia?publisherId=3695&releaseId=59786191. 12.2.2018

Everledger. 2018. Everledger. https://www.everledger.io/ 12.2.2018

Faife, C. 2017. Bitcoin hash functions explained. https://www.coindesk.com/bitcoin-hash-functions-explained/. 12.2.2018.

Feins, W. 2017. Satoshi Nakamoto – Creator Of Bitcoin. http://www.eurocheddar.com/profiles-in-europe/satoshi-nakamoto-creator-bitcoin/ 30.1.2018

Finley, K. 2016. A $50 million hack just showed that the Dao was all too human. https://www.wired.com/2016/06/50-million-hack-just-showed-dao-human/. 31.1.2018.

Greenspan, G. 2016. Why many smart contract use cases are simply impossible. https://www.coindesk.com/three-smart-contract-misconceptions/. 31.1.2018.

Haber, S. & Stornetta, W.S. J. 1991 How to time-stamp a digital document. Cryptology (3) 99-111

Harper, B. 2018. All Active Nodes are Equal in a Decentralized Peer to Peer trust—What is Bitcoin, and Legitimate Attacks Against it? https://medium.com/@Ben_Harper/all-active-nodes-are-equal-in-a-decentralised-peer-to-peer-trust-bitcoin-387da598bdce 12.2.2018

Hyperledger Burrows. 2018. Hyperledger Burrows v0.16. https://github.com/hyperledger/burrow. 31.1.2018.

Hyperledger Fabric. 2018. Hyperledger Fabric FAQ. http://hyperledger-fabric.readthedocs.io/en/release/Fabric-FAQ.html. 31.1.2018.

Hyperledger. 2018a. Hyperledger Composer Overview. https://www.hyperledger.org/wp-content/uploads/2017/05/Hyperledger-Composer-Overview.pdf 31.1.2018.

Hyperledger. 2018b. Hyperledger Fabric. https://hyperledger.org/projects/fabric. 31.1.2018.

Iansiti, M. & Lakhani K. 2017. The Truth About Blockchain. Harward Business Review. https://hbr.org/2017/01/the-truth-about-blockchain 5.12.2017

Leising, M. 2017 The Ether Thief. https://www.bloomberg.com/features/2017-the-ether-thief/. 31.1.2018.

Lewis, A. 2015. A gentle introduction to blockchain technology. https://bitsonblocks.net/2015/09/09/a-gentle-introduction-to-blockchain-technology/. 12.2.2018.

Lobban, T. 2017a. Quorum overview. https://github.com/jpmorganchase/quorum/wiki/Quorum-Overview. 31.1.2018.

Lobban, T. 2017b. Transaction processing. https://github.com/jpmorganchase/quorum/wiki/Transaction-Processing. 31.1.2018.

Madeira, A. 2018. What is the gas in Ethereum. https://www.cryptocompare.com/coins/guides/what-is-the-gas-in-ethereum/. 31.1.2018

Monax. 2018. Software Developments Kits. https://monax.io/library/. 31.1.2018.

Nakamoto, S. 2008. Bitcoin P2P e-cash paper. http://article.gmane.org/gmane.comp.encryption.general/12588/ 30.1.2018

Oraclize. 2018. How it works. http://www.oraclize.it/. 31.1.2018.

Panetta, K. 2017. Top Trends in the Gartner Hype Cycle for Emerging Technologies, 2017. https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/ 5.12.2017

Parzygnat, M & Gupta, N. IBM Blockchain 101: Quick-start guide for developers. https://www.ibm.com/developerworks/cloud/library/cl-ibm-blockchain-101-quick-start-guide-for-developers-bluemix-trs/index.html 31.1.2018.

Popper, N. 2016. Paper points up flaws in venture fund based on virtual money. https://www.nytimes.com/2016/05/28/business/dealbook/paper-

points-up-flaws-in-venture-fund-based-on-virtual-money.html. 31.1.2018

Price, R. 2016. Digital currency Ethereum is cratering because of a $50 million hack. http://www.businessinsider.com/dao-hacked-ethereum-crashing-in-value-tens-of-millions-allegedly-stolen-2016-6?r=UK&IR=T&IR=T. 31.1.2018.

Quentson, Andrew. 2016. Ethereum's Solidity flaw exploited in DAO attack say Cornell Researcher. https://www.ccn.com/ethereum-solidity-flaw-dao/. 31.1.2018.

Quorum. 2018. What is Quorum?. https://www.jpmorgan.com/global/Quorum. 31.1.2018.

R3. 2015. Press release. https://static1.squarespace.com/static/55f73743e4b051cfcc0b02cf/t/56727a13e0327c81fa9361ab/1450342931449/PRESS+RELEASE+R3+distributed+ledger+initiative+grows+to+42+bank+members+and+extends+reach+FINAL+.pdf. 31.1.2018.

Reitwiesner, C. 2018. Dr. Christian Reittwiesner profile on Linkedin. https://www.linkedin.com/in/dr-christian-reitwiessner-594b0982/ 31.1.2018

Rosic A. 2017a. Proof of Work vs Proof of Stake: Basic Mining Guide. https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/ 6.12.2017

Rosic, A. 2017b. What is Hyperledger? How the Linux Foundation builds an open platform around the blockchain project of Intel. https://blockgeeks.com/guides/what-is-hyperledger/. 31.1.2018.

Solidity. 2017. Solidity documentation. https://solidity.readthedocs.io/en/develop/. 31.1.2018.

Stackedit.io. 2014. Solidity-original. https://stackedit.io/viewer#!url=https://gist.githubusercontent.com/gavofyork/31b35cd2252a00d0d057/raw/16de06189d2175d2e31b300f1f8531e20c927635/solidity-original. 31.1.2018

Stinchcombe, K. 2017. Ten Year In, Nobody Has Come Up With A Use For Blockchain. https://hackernoon.com/ten-years-in-nobody-has-come-up-with-a-use-case-for-blockchain-ee98c180100 30.1.2018

Tomshwom. 2017. Ethereum “gas” - how it works. https://steemit.com/ethereum/@tomshwom/ethereum-gas-how-it-works. 31.1.2018

Truffle. 2018. Building Dapps for Quorum: Private Enterprise Blockchains. http://truffleframework.com/tutorials/building-dapps-for-quorum-private-enterprise-blockchains. 31.1.2018.

Tual, S. 2015. Is all the code inside of an Ethereum contract public? https://www.quora.com/Is-all-the-code-inside-of-an-Ethereum-contract-public. 31.1.2018.

Vigna, P. 2016. Chiefless company rakes in more than $100 million. https://www.wsj.com/articles/chiefless-company-rakes-in-more-than-100-million-1463399393. 31.1.2018

Wavebl. 2018. Wavebl. http://wavebl.com 12.2.2018