Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Petteri Piironen

Johdanto älysopimuksiin

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

26.4.2020

Tekijä Otsikko Sivumäärä Aika	Petteri Piironen Johdanto älysopimuksiin 45 sivua + 1 liite 26.4.2020
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Mediatekniikka
Ohjaaja	Yliopettaja Kari Aaltonen

Insinöörityössä tutkittiin älysopimuksia uutena teknologiana ja perehdyttiin niiden käyttötarkoituksiin, hyötyihin ja haasteisiin. Työ toteutettiin itsenäisenä tutkimuksena.

Insinöörityön tavoite on tarjota lukijalle kattava käsitys lohkoketjuteknologioiden perusteista, älysopimusteknologiasta ja älysopimusten kehittämisestä. Insinöörityössä tehtyä tutkimusta voidaan hyödyntää oppimateriaalina lohkoketjuteknologioiden ja älysopimusten opetuksessa.

Työssä perehdytään lohkoketjuteknologian perusteisiin, älysopimuksiin ja niiden käyttötarkoituksiin, hyötyihin sekä tämänhetkisiin rajoitteisiin. Älysopimukset ovat kehittyneet lohkoketjuteknologian pohjalta, ja ne ovat toistaiseksi uusi ja vilkkaan kehityksen alla oleva teknologia. Älysopimukset ovat sopimuksia koodin muodossa, johon voidaan määritellä lukuisia ehtoja ja määritteitä, joilla saavutetaan nykyisiä sopimusmalleja luotettavampi malli, jossa ei tarvita välikäsiä tai sovittelijoita.

Insinöörityössä toteutettiin myös mallisopimus Solidity-ohjelmointikielellä Remix web -kehitysympäristössä. Mallisopimuksen on tarkoitus havainnollistaa älysopimusten kehityksen kulku ja niihin vaadittaviin työkaluihin ja teknologioihin tutustuminen.

Työn tuloksena saatiin kattava esittely lohkoketjuteknologian perusteista, älysopimuksista ja niiden käyttötarkoituksista, hyödyistä ja rajoitteista sekä kehitettiin mallisopimus, jonka avulla havainnollistetaan ja opastetaan älysopimusten kehitys vaihe vaiheelta.

Avainsanat	älysopimus, lohkoketju, Ethereum, kryptografia, automaatio, bitcoin

Author Title Number of Pages Date	Petteri Piironen Introduction to Smart Contract technologies 45 pages + 1 appendix 26 April 2020
Degree	Bachelor of Engineering
Degree Programme	Information and Communications Technology
Professional Major	Media Technology
Instructor	Kari Aaltonen, Principal Teacher

In this thesis, an extensive independent research into smart contract and blockchain technologies was conducted. The purpose of this final year project was to provide answers to smart contract benefits, use cases and to bring forth the current issues that smart contract technologies need to overcome.

The study also aims at giving a broad understanding into blockchain and smart contract technologies and to also provide instructions for smart contract development using modern development tools. The research carried out in this thesis also serves as material for any future courses in these technologies. Smart contracts have been emerging from rapidly evolving blockchain technology which is under constant development. Smart contracts are programmable, deterministic and automatic forms of contracts that do not require a middleman, as they can be fully trusted to execute on the terms and functions written in the contract.

This thesis also includes an example contract written in Solidity in Remix-web integrated development environment. The goal of this example contract is to provide the required technologies and steps to develop a functioning smart contract on the Ethereum blockchain.

The results of this study offer an extensive introduction to blockchain and smart contract technologies and smart contract development. This thesis also provides answers to smart contract benefits, use cases and some of the current issues that smart contracts are currently facing.

Keywords	smart contract, blockchain, Ethereum, cryptography, automation, bitcoin

Sisällys

Lyhenteet

BTC	*Bitcoin*. Vuonna 2009 Satoshi Nakamoton julkaisema kryptovaluutta.
ETH	*Ethereum.* Lohkoketjuteknologiaan perustuva hajautettu avoimen lähdekoodin tietojenkäsittely-ympäristö, jolla on oma, erillinen lohkoketju.
dapp	*Decentralized App.* Hajautettu sovellus, jonka palvelinpuolen koodi pyörii hajautetulla vertaisverkolla. Hajautettu sovellus voi muuten näyttää ja toimia täysin muiden sovellusten tavoin.
EVM	*Ethereum Virtual Machine.* Ethereumin tarjoama hajautettu virtuaalikone, joka pystyy suorittamaan ohjelmakoodia kansainvälisen tietoverkon avulla.
P2P	*Peer-to-Peer.* Vertaisverkko, jolla ei ole kiinteitä palvelimia tai käyttäjiä, vaan jokainen verkkoon kytketty taho toimii palvelimena ja asiakkaana muille tahoille.
PoW	*Proof-of-Work*. Proof-of-work konsensusmekanismi, jolla kryptovaluuttojen kontekstissa louhijat palkitaan sen perusteella, kuka on löytänyt ratkaisun ja pystynyt lisäämään lohkonsa lohkoketjuun. Tätä ratkaisua kutsutaan nimellä “proof of work”.
PoS	*Proof-of-Stake.* toisenlainen konsensusmekanismi, jolla kryptovaluutojen kontekstissa seuraavan lohkon luoja valitaan erinäisillä kriteereillä, kuten satunnaisuudella ja osuuden vauraudella sekä iällä.
ICO	*Initial Coin Offering.* Englanninkielinen termi joukkorahoituksen muodolle, joka keskittyy kryptovaluuttoihin kasvuyritysten pääoman lähteenä.
IDE	*Integrated Development Environment.* Ohjelmointiympäristö, jolla ohjelmoija toteuttaa ja suunnittelee ohjelmistoa. Ohjelmointiympäristössä yhdistyvät tyypillisesti tekstieditori ja ohjelmointikielen kääntäjä.
HTTPS	*Hypertext Transfer Protocol Secure.* HTTP-protokollan ja TLS/SSL-protokollan yhdistelmä, jota käytetään suojattuun tiedonsiirtoon internetissä.

SHA-256	*Secure Hash Algorithm.* Yhdysvaltain kansallisen turvallisuusviraston kehittämä salausalgoritmi, jota käytetään mm. bitcoinin transaktioiden vahvistamiseen.
SAT	*Satoshi. B*itcoin-kryptovaluutan rakenteellinen osa, joka on sadasmiljoonasosa yhtä kokonaista bitcoinia.
ERC20	*Ethereum Request For Comments 20*. ERC20, Ethereum-verkon protokolla joka määrittää tiettyjä sääntöjä ja standardeja Ethereum-verkon rahakkeisiin. Luku 20 viittaa tiettyyn standardiin.
API	*Application Programming Interface.* API, ohjelmointirajapinta, määritelmä, jonka mukaan eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja keskenään.

1. Johdanto

Insinöörityön tavoitteena on perehtyä älysopimuksiin uutena teknologiana ja arvioida niiden tuomia vaikutuksia tulevaisuuden kannalta sekä tarjota kattava esittely lohkoketjuteknologiasta, kryptografiasta, älysopimusten perusteista ja käyttötarkoituksista. Työssä toteutetaan myös malliälysopimus Solidity-ohjelmointikielellä.

Sopimukset ovat perinteisesti käsitettynä kahden osapuolen välille asetettuja määritelmiä tai lupauksia, joita käytetään pääasiassa kaupankäynnissä ja taloudessa, mutta niille on käyttötarkoituksia myös ihmissuhteissa, kuten avioliitoissa. Sopimukset ovat tärkeitä valtiotasolla, sillä sopimukset ovat kautta aikojen olleet kapitalististen valtioiden peruspiirre. Sopimukset ovat vapaan markkinatalouden perusrakenne riippumatta siitä, paneeko ne täytäntöön hallitus vai jokin muu hallitseva taho. [1.]

Älysopimus on termi, jolla kuvataan koodia, joka automaattisesti suorittaa kaiken tai osia sopimuksesta, joka on tallennettu lohkoketjupohjaiselle alustalle. Historiallisesti sopimuksen jokaista vaihetta ovat valvoneet ja hallinnoineet ihmiset, oli kyseessä sopimuksen luominen, sovittelu tai täytäntöönpano. Älykkäiden sopimusten konsepti mahdollistaa luotettavien sopimusluontoisten tapahtumien, kuten rahallisten transaktioiden suorittamisen ilman kolmannen osapuolen valvontaa tai osallistumista. Lohkoketjuteknologian myötä nämä sopimukset ovat esillä julkisesti jaetussa tilikirjassa, josta sopimus voidaan varmentaa koska tahansa, eikä sen sisältöä voi lohkoketjuteknologian luonteen mukaan muuttaa. Älysopimusten myötä ollaan siirtymässä omistajuuden määrittelyn ylle, sillä omistajuuden lisäksi voidaan määritellä omistukseen liittyviä ehtoja, jotka tulevat tietynlaisten sopimustyyppien mukana, kuten arvopapereiden tai obligaatioiden. Obligaatiot sisältävät tietynlaisia ehtoja, kuten koron tuottoa tai ajoittaista uudelleenmaksua tai voi seurata rangaistuksia, mikäli tiettyjä velvoitteita ei noudateta. Näitä ehtoja voidaan määritellä älysopimuksiin ja automatisoida tarpeen mukaan. Älysopimuksiin voidaan määritellä koodilla erinäisiä sääntöjä liittyen sopimuksen luonteeseen, kuten varainsiirto taholta A taholle B tai sopimuksen yksityisyyteen liittyviä ominaisuuksia, joihin perehdytään tässä työssä tarkemmin.

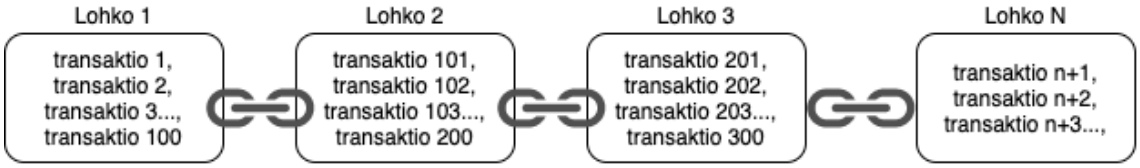
2. Lohkoketjuteknologian perusteet

Tässä luvussa käydään läpi lohkoketjuteknologian peruskäsitteet ja siihen liittyvät teknologiat ja protokollat.

2.1. Lohkoketjun konsepti

Lohkoketju (engl. blockchain) on monelle tuttu termi, vaikkei sen teknisestä puolesta välttämättä ymmärretä tarkemmin. Lohkoketjuteknologian merkittävin toteutus julkaistiin vuonna 2008 artikkelissa *Bitcoin: A Peer-to-Peer Electronic Cash System*, jossa kehittäjä Satoshi Nakamoto (alias) julkisti bitcoin avoimen lähdekoodin [2].

Lohkoketjuteknologialla kuvataan konseptia, jolla voidaan tallentaa tietoa niin sanottuihin lohkoihin, jotka yhdessä muodostavat julkisen rekisterin, tietynlaisen tilikirjan kaikista ketjussa tapahtuneista tapahtumista ja tiedoista. Lohkoketju koostuu lohkoista, jotka ovat yhdistettynä toisiinsa kryptografisin menetelmin muodostaen listamaisen rakenteen, kuten kuvassa 1.

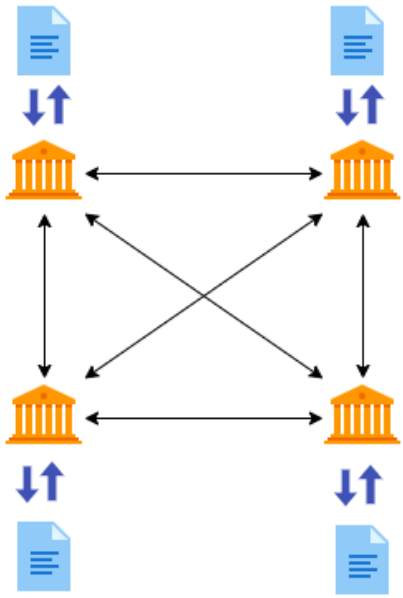


Kuva 1. Lohkoketju visualisoituna.

Tämä lista jaetaan kaikkien osallistujien kesken, jolloin tuloksena on hajautettu vertaisverkko, joka ei luota keskitettyyn palvelimeen tai toimijaan, johon kaikki osallistujat olisivat yhdistettynä. Jaetun listan voi täten tarkastaa mistä tahansa lähteestä, mikä mahdollistaa anonyymin luottamuksen osallistujien välillä. Hajautettu malli on luotettavampi kuin keskitetty malli, jossa kaikki keskitettyyn toimijaan yhdistetyt tahot ovat riippuvaisia keskitetyn toimijan luotettavuudesta. Hyvä esimerkki perinteisestä keskitetystä mallista on esimerkiksi pankkien toiminta, jossa kaikki maksuliikenne kulkee pankin kautta, kuten kuvassa 2.

Kuva 2. Keskitetty malli, jossa kaikki toimijat nojaavat yhteen keskeiseen tahoon.

Hajautetussa mallissa jokaisella osapuolella on sama tilikirjan versio, joka päivittyy automaattisesti silloin, kun ketjuun kirjataan uusia tapahtumia. Tämän myötä jokaisen transaktion pätevyys voidaan varmentaa kaikkien osallistuvien tahojen kautta, ja mikäli transaktio vastaa sitä, mitä kaikkien tilikirjaan on kirjattu, voidaan turvallisesti olettaa sen olevan pätevä. Hajautettu malli ei nojaa keskitettyyn toimijaan ja on täten luotettavampi ei pelkästään moraalisesti, vaan myös toiminnallisesti, sillä yhden toimijan kaatuessa mikään informaatio ei enää liiku verkossa. Jos taas hajautetussa mallissa yksi tahoista kaatuu, se ei vaikuta verkon toimintaan millään tavalla (kuva 3).



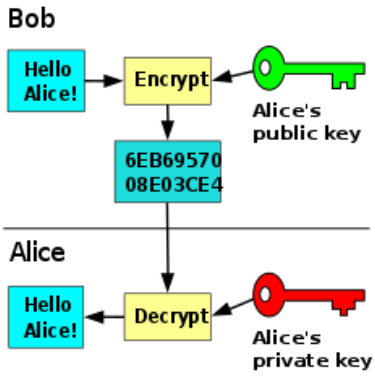
Kuva 3. Hajautettu malli, jossa yhdistetyt tahot eivät nojaa keskeiseen toimijaan, vain jakavat kaiken tiedon keskenään.

Jokainen lohko sisältää dataa edeltävästä lohkosta, kuten kryptografisen hajautusalgoritmin eli tiivisteen, aikaleiman sekä transaktiodataa. Lohkoketju voi haarautua uusiin ketjuihin, jolloin voimaan jää se haara, jonka lohkojen yhteenlaskettu vaikeustaso on suurempi. Poisjäävän haaran lohkoja kutsutaan orpolohkoiksi. Lohkoketjuteknologia mahdollistaa myös sen, ettei lohkoketjun historiaa voi muuttaa, sillä se vaatisi lohkon uudelleenluomista ja siihen liittyvien aiempien lohkojen uusiksi luomista. Tämä on yksi lohkoketjuteknologian vahvimmista puolista, sillä se estää mahdollisen datan väärentämisen.

Bitcoinin lohkoketjussa jokainen kahden käyttäjän välinen transaktio tallennetaan lukuisten muiden samantyyppisten transaktioiden kanssa yhteen lohkoon, joka liitetään jatkuvaan lohkoketjuun. Bitcoinin transaktioluontoisessa kontekstissa lohkoketjun hyödyt tulevat parhaiten esille, sillä transaktioissa ei tarvita välikäsiä ja jokainen transaktio jää muiden todistettavaksi ja julkiseksi lohkoketjussa olevaan tilikirjaan.

2.2. Lohkoketjujen kryptografia

Yleisesti lohkoketjujen yhteydessä kryptografiaan viitattaessa tarkoitetaan kryptaamista eli salausta, joka on prosessi, jossa viesti tai tieto muunnetaan muotoon, jossa vain oikeutetut osapuolet voivat lukea viestin. Yleisin ja tunnetuin näistä on Public-key encryption, jossa kahden osapuolen välinen kommunikaatio tapahtuu salatusti julkisen ja yksityisen avaimen avulla. Tätä menetelmää on helppo lähestyä esimerkillä, jossa Bob lähettää Alicelle viestin, jonka hän kryptaa omalla julkisella avaimella ja Alicen yksityisellä avaimella. Alice purkaa viestin kryptauksen omalla julkisella avaimella ja Bobin yksityisellä avaimella. Milläkään muulla taholla, jolla ei ole pääsyä yksityisiin avaimiin, ei myöskään ole pääsyä viestin sisältöön (kuva 4).



Kuva 4. Public-key encryption, jossa kahden osapuolen välinen kommunikaatio voidaan salata ja avata osapuolten omistamien avainten avulla [26].

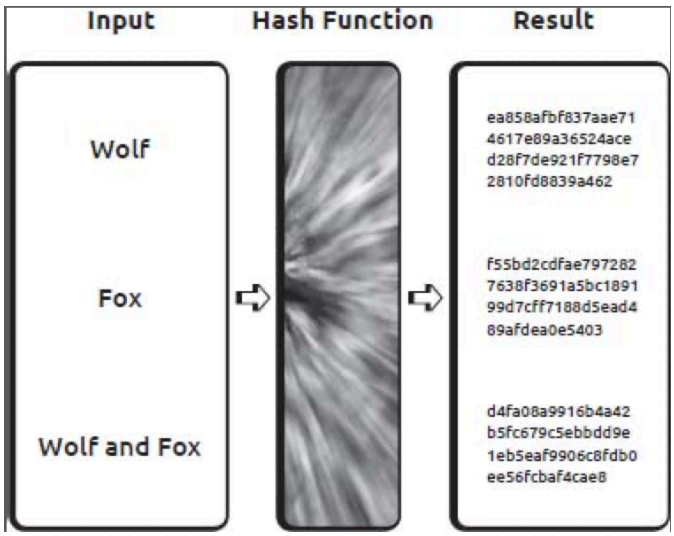
Hajautusfunktio

Lohkoketjuille yleisin kryptaamisen muoto on Hash Function eli hajautusfunktio. Hajautusfunktio on matemaattinen algoritmi, jota hyödynnetään jonkin konseptin, kuten sanan, muuttamiseksi ennaltamäärättyyn koodiin. Mikäli jokin osa sanasta muuttuu, sitä mukaa myös koodi muuttuu. Täten jokaiselle muunnetulle objektille on oma hash eli tiiviste. Kuva 5 havainnollistaa erilaisten merkkijonojen tiivisteiden luomisen.

Syöte	Tiiviste
Merkkijono	48fbfc83242b688fb60fad0f82b5cf65fef4a573d9b0c13955b84a9ebbc6798c
merkkijono	d72149fdc8b2eaa2fbd1eb7a23b89d6ba42abaa09c19fca483908875311500f1
Merkkijooono	92467eea9dbcfeb6dcd0c562f54bc8192b209ec688aa42ed8d9c279b0d8a5a90

Kuva 5. Esimerkkiarvoja SHA-256-tiivisteille. Yksikin muutos merkkijonossa saa koko tiivisteen muuttumaan.

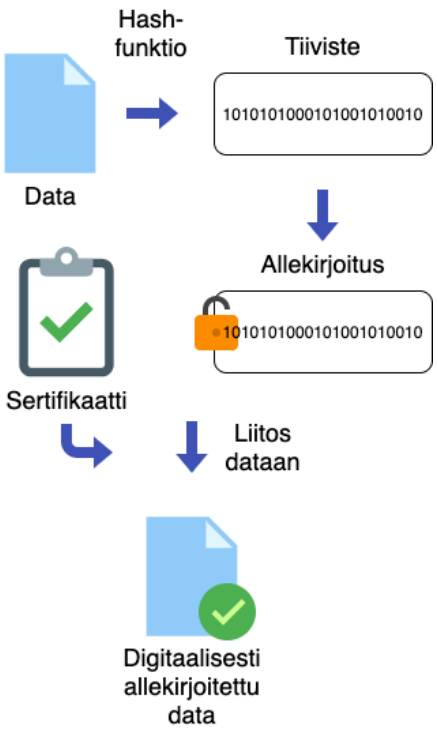
Lohkoketjuissa yleisin hajautusfunktio on Yhdysvaltojen kansallisen turvallisuusviraston eli NSAn kehittämä Secure Hash Algorithm, eli tiivistefunktio josta käytetään lyhennettä SHA. SHA-perheeseen kuuluu useita eri iteraatioita algoritmista, mutta tässä tapauksessa perehdytään SHA-256 -algoritmiin. SHA-256 käytetään mm. bitcoinin transaktioissa ja Proof-of-Work-ja Proof-of-Stake-protokollissa. SHA-256 -algoritmin perusominaisuus on se, että riippumatta syötetyn konseptin pituudesta, tulos on aina 256 bittiä pitkä. Tulos usein esitetään heksadesimaalina, jossa tapauksessa pituus on 64 merkkiä. [4.]



Kuva 6. Hajautusfunktio visualisoituna. Syötetyn merkkijonon pituudella ei ole väliä tuloksen kannalta, tulos on aina yhtä pitkä. [5, s. 75.].

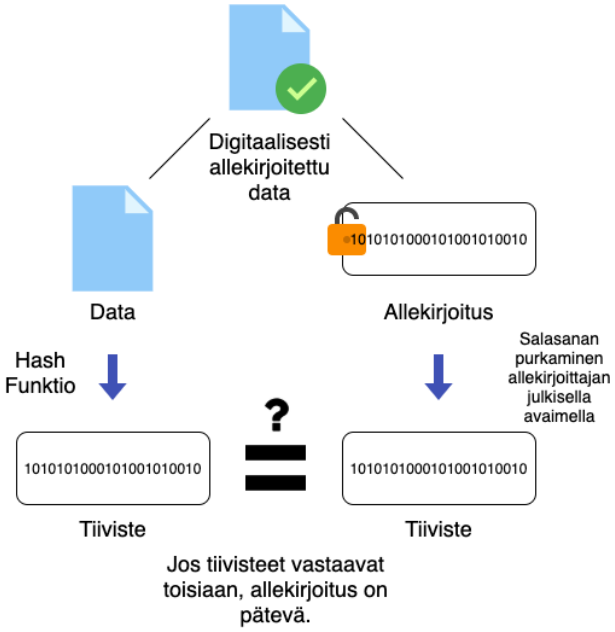
Digitaaliset allekirjoitukset

Digitaalisella allekirjoituksella tarkoitetaan sähköiseen viestiin tehtyä aiemmin mainittuun julkisen avaimen formaattiin perustuvaa allekirjoitusta, jolla varmennetaan viestin tai datan sisältö ja allekirjoittajan henkilöllisyys [6]. Mikäli viestiä muutettaisiin, allekirjoitus paljastaisi väärennöksen. Viestiä muutattaessa sen tallentamisen tai siirtämisen aikana tai väärää henkilöllisyyttä käytettäessä matemaattinen allekirjoitus ei enää täsmää. Digitaalisia allekirjoituksia kuten kuvassa 6 hyödynnetään yritysten asiakirjoissa ja sähköposteissa, ja ne ovat laillisesti päteviä. Digitaalisia allekirjoituksia voidaan soveltaa älysopimuksissa osapuolten varmistamiseen.



Kuva 7. Digitaalisen allekirjoituksen prosessi [27].

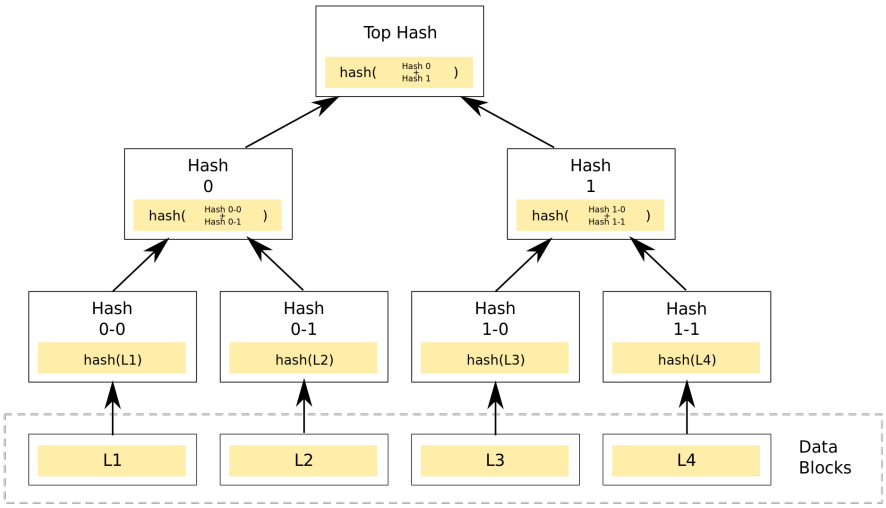
Digitaaliset allekirjoitukset, aivan kuten käsin kirjoitetut allekirjoitukset, ovat ainutlaatuisia. Kun allekirjoittaja digitaalisesti allekirjoittaa dokumentin, allekirjoitus luodaan käyttäen allekirjoittajan yksityistä avainta. Matemaattinen algoritmi toimii salauksena ja luo dataa, joka vastaa allekirjoitettua dokumenttia, jota kutsutaan tiivisteeksi ja joka salaa allekirjoitetun datan. Tulokseksi jäänyt tiiviste on digitaalinen allekirjoitus, jolle on annettu lisäksi aikaleima, joka kertoo, milloin se on luotu. Tämä varmistaa sen, että jos dokumenttia muutetaan allekirjoittamisen jälkeen, digitaalinen allekirjoitus on epäpätevä kuten kuva 8 havainnollistaa.



Kuva 8. Digitaalisen allekirjoituksen varmistaminen [27].

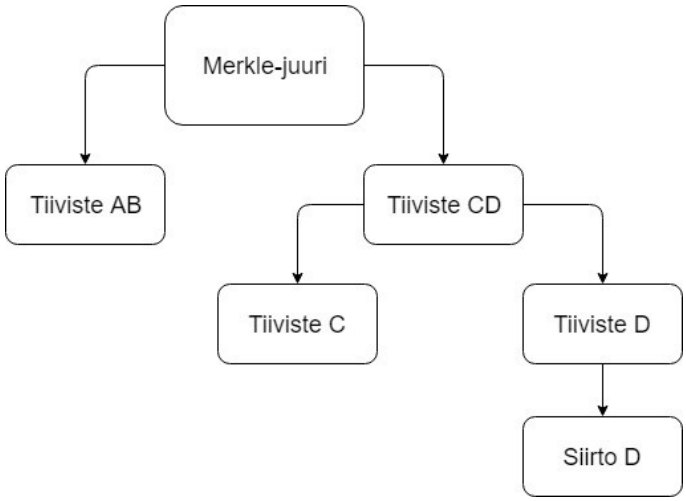
Merkle-puu

Kyrptografiassa ja lohkoketjuissa esiintyvä tiivistepuu tai Merkle-puu on puuta muistuttava rakenne, jonka avulla voidaan tallentaa suuria määriä dataa tehokkaasti ja data voidaan varmentaa turvallisesti. Merkle-puu mahdollistaa sen, että lohkoketjuun ei tarvitse luoda yhtä jättimäistä ylätunnistetta, johon kaikki lohkon transaktiot tallentuisivat. Merkle-puu mahdollistaa transaktioiden varmistamisen ns. ylhäältä alas, jolloin syntyy useita puun haaroja, jotka ovat kooltaan pienempiä ja täten vaativat pienempää laskentatehoa. Rakenteessa tiivisteitä luodaan alhaalta ylöspäin, jolloin ensimmäiset tiivisteet ovat yksittäisten siirtojen tiivisteitä. Merkle-puun rakenteessa näitä tiivisteitä kutsutaan lehdiksi. Myöhemmät ei-lehtitiivisteet luodaan yhdistämällä kaksi aiempaa tiivistettä, joita kutsutaan oksiksi. Lopulta jäljelle jää yksi tiiviste, jota kutsutaan juureksi. Kuvassa 9 havainnollistuu Merkle-puun rakenne. L-alkuiset datalohkot muodostavat tiivisteet 0-0, 0-1, 1-0 ja 1-1, joita kutsutaan lehdiksi. Seuraavaksi tiivisteistä 0-0 ja 0-1 muodostetaan tiiviste 0, ja tiivisteistä 1-0 ja 1-1 luodaan tiiviste 1, joita kutsutaan oksiksi. Lopulta tiivisteet 0 ja 1 muodostavat Merkle-juuren.



Kuva 9. Merkle-juuren tiivisterakenne. Data muodostaa lehdet, jotka muodostavat oksat jotka yhdessä muodostavat puun juuren [28].

Merkle-puun avulla voidaan yksinkertaisesti todentaa, onko jokin tieto puussa. Todennus kulkee ylhäältä alas. Tämä mahdollistaa sen, ettei koko puuta tarvitse ladata, kuten kuvasta 10 voidaan todeta.

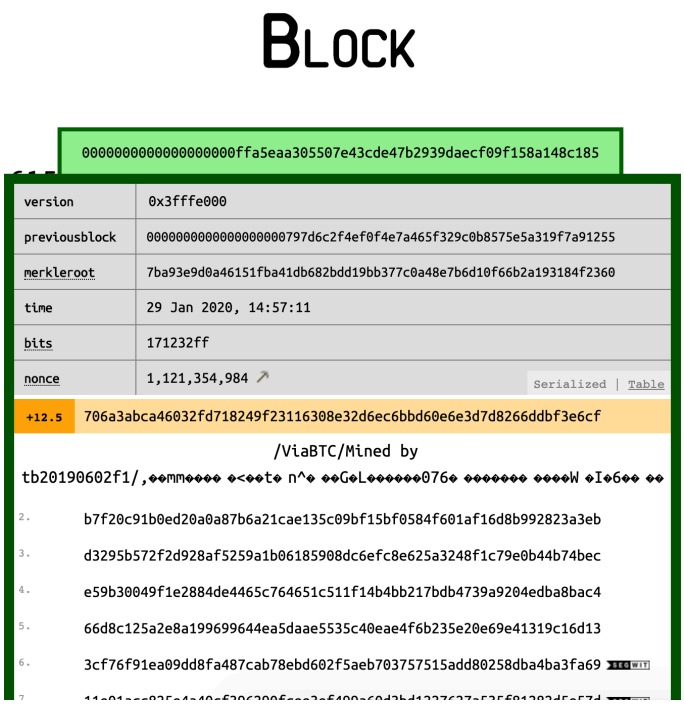


Kuva 10. Merkle-juuren rakenteesta yksittäisen tiedon varmentaminen [8].

Merkle-puulle on myös ominaista sen muuttumattomuus, eli mikäli jonkin rakenteessa olevan tiivisteen tietoa muutetaan, muuttuvat kaikki puun ylemmät oksat. Merkle-puun suurimmat hyödyt ovat sen nopeus, tehokkuus ja vähäinen tallennustilan käyttö.

2.3. Lohkon datarakenne

Kaikki transaktiodata on pysyvästi tallennettu tiedostoihin, joita kutsutaan lohkoiksi. Ne voidaan kuvitella yksittäisiksi sivuiksi pöytäkirjassa tai osakkeen tilikirjassa. Nämä lohkot järjestetään lineaariseen järjestykseen suhteutettuna aikaan ja ne muodostavat lohkoketjun. Uusien lohkojen transaktioiden prosessoiminen syntyy louhinnan yhteydessä, samoin kuin itse lohkojen luominen. Jokainen lohko sisältää oletukseltaan aikaleiman, viimeaikaisten transaktioiden tallenteet ja viittauksen edelliseen lohkoon. Lohkosta löytyy myös vastaus vaikeaan laskennalliseen matemaattiseen ongelmaan, joka on uniikki jokaiselle lohkolle. Ilman oikeaa vastausta ei uutta lohkoa voi lisätä ketjuun. Louhinnan konsepti on ratkoa näitä matemaattisia ongelmia, löytää uusia lohkoja ja tätä myötä saada uuden lohkon löytämisestä palkkio.



Kuva 11. Bitcoin-lohkoketjun yksittäisen lohkon rakenne, joka koostuu aiemman lohkon tiivisteestä, Merkle-juuresta, aikaleimasta ja bittiarvosta, joka ilmaisee lohkon laskennallisen vaikeustason sekä satunnaisluvun noncen [29].

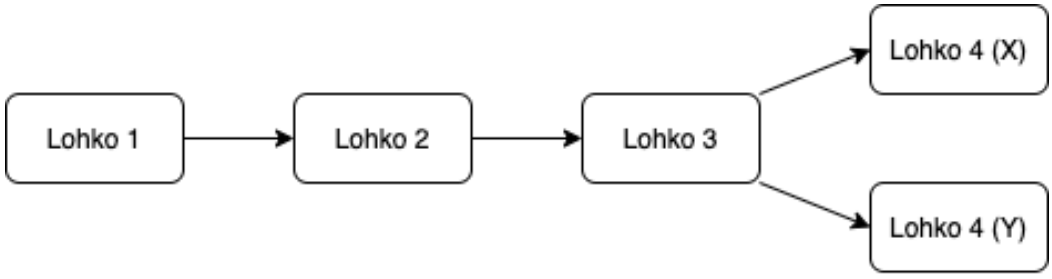
Jokainen lohko koostuu pääosin kahdesta tekijästä, ns. block headerista, johon tallennetaan erinäistä metadataa, ja pitkästä listasta, johon on listattu kaikki viimeaikaiset transaktiot. Kuvassa 11 näkyy block headerin rakenne, johon sisältyy previousblock–arvo, joka on aiemman lohkon tiiviste, Merkle root eli lohkon Merkle-juuri, joka on luotu lohkon transaktioiden tiivisteistä, time eli aikaleima, joka ilmastaan Unix-muodossa. ja bits, joka ilmaisee lohkon matemaattisen vaikeuden. Nonce on satunnainen luku, jonka louhijat löytävät uuden lohkon luonnin yhteydessä ja jonka mukaan louhija saa palkinnon.

Block headerista ja Merkle-puusta voidaan havaita, kuinka lohkoketju ei ole yksinkertaisesti pelkkään datan turvalliseen säilyttämiseen perustuva teknologia, vaan erittäin tehokas ja muuttumaton, sillä pienikin muutos lohkossa vaikuttaa aiempiin ja on välittömästi havaittavissa.

2.4. Proof-of-Work-protokolla

Proof-of-Work (PoW), eli todiste työstä, on protokolla, jolla alun perin vuonna 1993 julkaistussa artikkelissa "*On Memory-Bound Functions For Fighting Spam*" havainnollistettiin roskapostin estämistä tuomalla sähköposteihin prosessin, joka vaatii lähettäjää prosessoimaan jokaista viestiä määritetyn ajan verran ennen lähettämistä. Tämä todistaa, että postin lähettämiseen oli käytetty määritetyn verran laskentaa ja aikaa, eli se ei ole roskapostia. PoW on konsensusalgoritmi, joka on tiedon tuottamiseen kallis ja aikaa vievä, mutta helposti muiden todennettavissa. [10.]

Bitcoinin PoW vaatii louhijoiden ratkaisevan matemaattisia laskuja löytääkseen uusia lohkoja ketjuun. Laskujen vaikeus ei aina ole sama, ja se mukautuu jatkuvasti, jotta uusia lohkoja voitaisiin luoda tasaisin väliajoin. Protokollan perusidea on, että kalliista ja aikaa vievästä työstä louhijat saavat palkkion joka kerta, kun uusi lohko löydetään. Tämä kuitenkin johtaa tilanteeseen, jossa yhdellä lohkoketjulla on tuhansittain louhijoita, jotka kilpailevat uusista lohkoista itsenäisesti. Seurauksena voi olla, että kaksi louhijaa löytävät uuden lohkon, jolla on eri Merkle-juuren tiiviste, mikä johtaa kokonaan erilaiseen lohkon tiivisteeseen. Tämän vuoksi lohkoketju jaetaan kahteen eri haaraan, ja se haara jonka alussa olevan lohkon ratkaiseminen oli haastavampaa, saa jatkaa viraallisena lohkoketjuna, jolle louhijat siirtyvät.



Kuva 12. Lohkoketjun haarauma.

Yksi merkittävistä PoW-protokollan ongelmista on louhinnan keskittäminen. [8.] Jokainen taho voi halutessaan louhia itsenäisesti, mutta kilpailun kasvaessa on syntynyt suuria louhintajoukkoja, joiden laskentateho on marginaalisesti suurempi kuin yksittäisten toimijoiden, jolloin niiden tuotto on suurempaa yksittäisiin toimijoihin verrat-

tuna. Tämä on johtanut siihen, että monet yksittäiset louhijat eivät enää louhi itselleen, vaan antavat laskentatehoaan isojen organisaatioiden louhinta“altaaseen”, jota organisaatio hallinnoi. Täten kaikki altaaseen liittyvät louhijat saavat oman laskentatehonsa edestä palkkiota sitä mukaa, kun organisaatio löytää uusia lohkoja. Näin yksittäisten toimijoiden ei tarvitse kilpailla suurta laskentatehoa vastaan, vaan voivat liittyä osaksi sitä. Karkeasti 80 % koko verkon laskentatehosta tulee seitsemästä isosta louhinta-altaasta, joista kuusi on peräisin Kiinasta. Teoreettisesti nämä tahot voisivat tehdä yhteistyötä ja koordinoida hyökkäyksiä verkkoa kohtaan tai manipuloida transaktioita oman tarpeensa mukaisesti, mikä johtaisi esimerkiksi “double-spending”-ongelmaan. [11.]

Louhijoiden suorituskykyä mitataan heidän laskentakykyjen mukaan, ja tätä suorituskykyä voi kuvata termillä hash rate. Koska bitcoin on hajautettu järjestelmä kaikille sen käyttäjille, myös sen louhiminen tapahtuu maailmanlaajuisesti ja hajautetusti. Teoreettisesti jokin taho, joka kykenee saamaan yli 50 % kokonaissuorituskyvystä,kykenisi manipuloimaan transaktioiden hyväksymistä ja järjestelyä. [11.] Tämä taho voisi jopa peruuttaa transaktioita, mikä johtaisi “douple-spending”-ongelmaan, jossa hyökkääjän valuuttasumma palautuisi, mikä mahdollistaisi valuutan uudelleen käytön, ja tämä olisi koko bitcoinin perusideaa vastaan. Onnistunut hyökkääjä voisi myös estää muita louhijoita louhimasta, mikä johtaisi monopoliasemaan louhinnassa. Tämänkaltaista teoreettista hyökkäystä kutsutaan nimellä 51 % -hyökkäys. [11.]

51 % -hyökkäys on kuitenkin monissa tapauksissa erittäin epätodennäköistä, ja teoreettisesti lähes mahdotonta. Bitcoinin arvon nousun myötä se on houkuttanut enemmän ja enemmän louhijoita ketjulle, jonka ansiosta järjestelmä on päässyt kasvamaan hyvin laajaksi. Tämänhetkinen lohkopalkkio louhijalle on 12,5 BTC lohkoa kohden, joka 210 000 lohkon välein puolittuu jatkuvasti, kunnes uusia lohkoja ei voida enää löytää. Tällainen kilpailullinen asettelu takaa louhijoiden rehellisyyden, kun he saavuttavat lohkopalkkioita. [30.]

Bitcoinin tämänhetkinen hash rate on noin 120 000 000 TH/s (triljoonaa hashia per sekunti) 28.1.2020. Jos jokin taho haluaa manipuloida tätä, tulisi hänen tuottaa 60 000 000 TH/s, mikä ei teoriassa ole mahdotonta, mutta erittäin epätodennäköistä käytännössä. [31.]

2.5. Proof-of-Stake-protokolla

Lohkoketjun perusperiaatteeseen kuuluu hajauttaminen. Tässä mallissa yhteen yksittäiseen toimijaan tai keskukseen ei tarvitse luottaa vaan koko järjestelmä on hajautettu useiden osallistujien kesken ja uusien lohkojen luonti on jaettu tasaisesti kaikkien kesken. PoW-protokolla kuitenkin tuo mukanaan omia ongelmia, kuten louhinnan keskittäminen, korkea energiankulutus sekä 51 % -hyökkäykset. Näiden ongelmien inspiroimana kehittäjät ovat luoneet toisenlaisen protokollan nimeltä Proof-of-Stake eli todiste osuudesta. Laskennallisen työn sijaan osallistujat näyttävät todisteen omasta osuudestaan ja laittavat sen ns. pantiksi. Mitä suurempi pantti, sitä suurempi mahdollisuus on päästä muodostamaan seuraava lohko. Tämän ansiosta osallistujat eivät kilpailisi laskennallisesta tehosta saadakseen uuden lohkon ketjuun, vaan protokolla valitsisi seuraavan kirjoittajan tiettyjen kriteerien mukaan, kuten osuuden iän, määrän sekä satunnaisuuden perusteella. [12.]

Taistellakseen suuria osuuksia vastaan protokolla hyödyntää satunnaisuutta ja osuuden ikää kriteereinä, jolloin suuret osuudet eivät pelkästään takaa voittoa. Protokollassa osallistujat laittavat pantiksi rahakkeita (tai kryptovaluuttoja), mistä syntyy rahallinen sitoutuminen. Mitä pidempään osuus on ollut ylhäällä, sitä suurempi sen mahdollisuus on päästä muodostamaan seuraava lohko. Satunnaisessa valinnassa muodostajat valitaan pienimmän tiivistearvon ja suurimman pantin mukaan. PoS-protokolla yhdistää näitä kolmea edellä mainittua kriteeriä, ja niiden yhteisen konsensuksen perusteella valitsee seuraavan muodostajan. Protokollassa ei lasketa matemaattisia ongelmia eikä uusia lohkoja löydetä louhinnan kautta vaan niitä muodostetaan.

PoS-protokolla tarjoaa potentiaalisia ratkaisuja louhinnan keskittymiselle ja korkealle energiankulutukselle, mutta PoS ei tule kuitenkaan ongelmitta. Vaikka se käyttää erinäisiä kriteerejä taistellakseen suuria osuuksia vastaan, ne ovat silti yksi kriteereistä, joten mitä suurempi osuus, sitä enemmän painoarvoa sillä kriteerillä on. Tämä voi johtaa keskittymiseen, mikäli suurimmat osuudet olisivat vain muutamalla taholla. Toiseksi PoW-protokollassa uusien haarojen muodostuessa valitaan aina se, kumpaan on käytetty enemmän laskennallista energiaa, mutta PoS-protokollassa uusien haarojen muodostuessa se jää tahojen päätettäväksi. Tämä johtaa siihen, että molempia haaroja voidaan jatkaa, sillä lohkojen muodostamiseen ei mene kuluja energian kannalta, joten se on taloudellisesti kannattavaa lohkoja validoiville tahoille. Joku haitallinen taho voisi hyödyntää tätä luomalla uuden haaran ennen varojensa käyttämistä ja jatkaa oman haaransa louhimista, jolloin aiemman mainitun periaatteen mukaan muut louhijat jatkavat hänen haaraansa, sillä siitä ei synny lisäkuluja. Täten

hyökkääjän haarasta tulisi ajan myötä pisin haara, vaikka hänellä olisi hyvin pieni osuus siinä. [13.]

Tätä ongelmaa kutsutaan ‘nothing-at-stake’ -ongelmaksi. Vaikka tämänkaltaiset ongelmat ovat hyvin vaikeita toteuttaa eikä niistä välttämättä ole suurta taloudellista hyötyä mahdolliselle hyökkääjälle, ne ovat silti varteenotettavia ongelmia vertailtaessa erilaisia protokollia.

2.6. Kryptovaluutta

Kryptovaluutat, eli erinäiset digitaalisen valuutan muodot, kuten bitcoin, hyödyntävät lohkoketjuteknologiaa. Kryptovaluuttojen keskeinen konsepti perustuu vapaaseen, hajautettuun valuutanmuotoon, jossa sitä eivät määrää pankit tai niiden asettamat korkoprosentit, vaan kysynnän ja tarjonnan laki. Bitcoinien teoreettinen maksimimäärä on 21 miljoonaa kolikkoa, joista osa on ajan myötä kadonnut inhimillisiin virheisiin, kuten yksityishenkilöiden lompakoiden yksityisten avainten katoamiseen. Bitcoineja on siis rajattu määrä olemassa, ja kun viimeiset louhinnan yhteydessä luodut bitcoinit on jaettu, niitä ei tule enempää. Kullan tavoin bitcoinin arvo perustuu sen rajallisuuteen sekä muihin ominaisuuksiin, kuten sen nopeaan ja globaaliin siirtämiseen, sen yksityiseen luonteeseen ja välikäsien tarpeettomuuteen. Sen käyttämistä helpottaa myös se, että bitcoin voidaan jakaa pienempiin yksikköihin eli satosheihin (SAT), joilla käyttäjät voivat suorittaa pienempiä maksuja. [32.]

Perinteisessä maksuliikenteessä transaktiot menisivät kolmannen osapuolen kautta A:n ja B:n välillä, missä nousee erinäisiä kysymyksiä esiin, kuten kolmannen osapuolen luotettavuus, transaktion suoriutuminen ongelmitta, transaktion käsittelymaksu jne. Kryptovaluutan tapauksessa transaktio osapuolten välillä on suora, luotettava ja joidenkin kryptovaluutojen tapauksessa myös anonyymi. Nämä ominaisuudet tekevät kryptovaluutoista arvokkaan valuutan muodon henkilöille, jotka eivät tarvitse perinteistä valuuttaa tai usko siihen.

Kryptovaluutoja on bitcoinin julkaisun jälkeen kehitetty moniin eri käyttötarkoituksiin, joista osa tarjoaa ominaisuuksia, joita bitcoin ei tarjoa, kuten anonyymiä transaktiota, nopeampia suoritusaikoja, eri lohkoarkkitehtuuria yms. Myös perinteisiä omistuksia on alettu muuttaa digitaaliseen muotoon, jolloin esimerkiksi osake muunnetaan digitaaliseen valuuttaan tai rahakkeeseen, joka edustaa osaketta.

Kryptorahakkeet

Kryptorahakkeet ovat erikoisempi digitaalisen valuutan muoto, jossa jokainen rahake edustaa jotain omaisuutta tai hyödykettä oikeassa maailmassa, kuten osakkuutta tai arvopaperia. Yleisin kryptorahakkeen muoto on ERC20-rahake, joita kuka tahansa voi luoda Ethereum-lohkoketjulla. ERC20 on protokolla, joka määrittelee tiettyjä sääntöjä ja standardeja Ethereum-lohkoketjun rahakkeisiin. Yksinkertaisesti selitettynä, mikäli rahakkeen luoja sisällyttää tiettyjä funktioita rahakkeen älysopimuksessa, on rahake tällöin ERC20-standardia noudattava. [33.]

Rahakkeita usein tarjotaan ICO- eli Initial Coin Offering -yhteydessä, jossa yhtiöt tarjoavat mahdollisuuden osallistua rahakkeiden jakamiseen joukkorahoituksen kautta. Joukkorahoitusta tarjoavat yhtiöt määrittelevät, kuinka monta rahaketta kukin sijoittaja saa esimerkiksi yhtä Etheriä vastaan. Rahakkeilla voidaan tarjota ratkaisuja yhtiön ekosysteemeihin, luoda kannustimia yksilöille osallistua yhtiön palveluun tai luoda esimerkiksi Proof-of-Stake -järjestelmä. Rahakkeille on useita eri käyttötarkoituksia, ja niiden markkina-arvo on kasvanut viime vuosina merkittävän kokoiseksi, mikä on johtanut yritysmaailman ja lainsäätäjien nousevaan mielenkiintoon rahakkeita ja kryptovaluuttoja kohtaan. [34.]

3. Älysopimusten perusteet

Tässä luvussa käydään läpi älysopimusten perusteita, niiden keskeinen idea, käyttötarkoituksia, niiden rajoitteita sekä tulevaisuuden mahdollisuuksia.

3.1 Konsepti

Älysopimuksen termin kehitti 1990-luvun alkupuolella tietojenkäsittelytieteilijä, juristi ja kryptograafikko Nick Szabo. Hän kuvaili sopimuksia “…paljon käytännöllisemmäksi kuin paperiset esi-isänsä. Älysopimus on ennalta määrätty joukko lupauksia digitaalisessa muodossa, mukaan lukien protokollat, joiden sisällä osapuolet toimivat täyttääkseen sopimukseen määritetyt lupaukset.” Älysopimus on toisin sanoen deterministinen digitaalinen sovinto, kahden osapuolen välinen tai tiettyihin toimintoihin liittyvä sopimustyyppi. Jos jotain tapahtuu sopimuksessa, sen voidaan taata tapahtuvan, automaattisesti.

Lohkoketjuteknologian ja sen mukana tulleen bitcoinin avulla lohkoketjuteknologiaa on alettu kehittää vuoden 2008 julkaisunsa jälkeen, ja tämän myötä myös älysopimusten idea on päässyt realisoitumaan ja ne ovat tulleet vilkkaan kehityksen alle viime vuosi-

na. Älysopimukset ovat tulleet käytäntöön Ethereumin kautta, joka tarjoaa käyttäjälle alustan älysopimusten suorittamiseksi ja kehittämiseksi. Älysopimukset voidaan nähdä itse itsensä toteuttavaksi autonomiseksi ohjelmaksi, jota voidaan käyttää esimerkiksi valuutan, omaisuuden tai minkä tahansa arvon siirtämiseen taholta toiselle. Älysopimukseen voidaan määritellä koodilla erinäisiä ehtoja ja toimintoja, joita suoritetaan tiettyjen ehtojen täyttyessä. Toimintaa voidaan lähestyä kiinteistösopimuksen muodossa. Mikäli taho A haluaisi ostaa maata tai kiinteistön, hän voisi käyttää älysopimusta, johon kiinteistön myyjä on määritellyt tietyn summan kiinteistölle, ja mikäli sopimuksen ehdot, jotka tarkastavat esimerkiksi ostohalukkuuden ja maksuvalmiuden, täyttyvät, transaktio voidaan automaattisesti suorittaa. Ihanteellisessa tapauksessa kiinteistön ostamisen yhteydessä ei tarvittaisi kolmansia osapuolia tarkastamaan ostajan valmiuksia varallisuuden suhteen tai nimittämään ostajaa uudeksi omistajaksi, vaan se voitaisiin digitaalisesti määritellä sopimuksen toteutuessa.

Lohkoketjun ja kryptovaluutoiden kontekstissa älysopimukset ovat siis ennalta määritettyä logiikkaa koodin muodossa. Ne tallennetaan ja kopioidaan jaetulle tilikirjalle eli lohkoketjulle, jossa hajautettuun vertaisverkkoon liittyneet tahot suorittavat, valvovat ja todentavat näitä sopimuksia. Toisin sanoen älysopimukset ovat pieniä ohjelmia, jotka suorittavat “jos x tapahtuu, tee y” -tyyppisiä ehtolausekkeita, joita todentavat muut tahot samalla hajautetulla verkolla. [23.]

3.2 Hyödyt

Verrattuna nykyisiin paperisiin sopimuksiin, älysopimukset tuovat mukanaan lukuisia hyötyjä, joihin perehdytään tässä luvussa.

Tarkkuus ja automaatio

Kaikki älysopimusta koskevat tiedot ilmaistaan sopimuksessa ehdollisessa muodossa käyttäen jos–niin -tyyppisiä ehtolausekkeita koodissa. Esimerkiksi tilatessaan palvelun tai tuotteen asiakas maksaa x määrään y-valuuttaa, ja sopimus välittömästi palauttaa asiakkaalle tuotteen tai palvelun tämän maksua vastaan. Kaikkien älysopimuksessa olevien ehtojen on oltava selkeitä ja tarkkoja. Tämä on kriittinen vaatimus, sillä transaktioissa voi ilmetä vikatilanteita, joiden sattuessa tulee olla tarkat ehdot määriteltynä. Automaattisuuden takia älysopimukset välttävät suurimman osan perinteisiin sopimuksiin liittyvistä ongelmista, kuten mahdolliset epäselvyydet ja vikatilanteiden aikaa vievät sovittelut. Mikäli sopimus on tarpeellisen kehittyneeksi ohjelmoitu, se voi automatisoida

lähes kaikki vikatilanteet ja niitä johtavat tapahtumat. Tällainen tarkkuus ja automaatio tekee älysopimuksista paperisia sopimuksia huomattavasti tehokkaampia. [23.]

Nopeus ja tehokkuus

Älysopimuksen perusperiaate on, ettei sopimuksen tarvitse nojata ihmisen valvontaan, vaan sen implementaatiota ohjaavat ja todentavat muut tahot lohkoketjulla. Siinä vaiheessa, kun jokin sopimuksen ehdoista täyttyy, sopimus suorittaa itse itsensä koodin määritelmien mukaisesti. Sopimukseen voidaan laatia useita ehtoja, joiden täyttyessä jokin osa koodista suoritetaan automaattisesti. Esimerkiksi kuukausimaksun tyyppisissä tilauksissa käyttäjän maksettua sopimukseen siihen määritetyn summan kryptovaluuttaa asiakkaan tilaus uudistettaisiin välittömästi. Perinteisiin sopimuksiin verrattuna, jotka ovat hitaampia ja vaativat jonkin asteen varmennuksen ihmiseltä, älysopimuksen ehtoja, kuten maksetun summan määrän, palvelun toteutumisen ja ehdon täyttymisen, hoitavat lohkoketjulla olevat muut tahot. Jokainen sopimus kohdistetaan erillisenä kokonaisuutena, ja jokainen tapahtuma validoidaan ensin alkuperästään riippumatta. Tämä johtaa nopeaan, joustavaan ja vankkaan tapaan toteuttaa sopimusluontoisia asioita. [23.]

Turvallisuus ja yksityisyys

Marinon ja Juelsin [15] julkaisemassa tutkimuksessa on todettu älysopimusten omaavan korkeimman luokan turvallisuuden. Älysopimukset luodaan lohkoketjulle, jonka mukana tulevat lohkoketjun tarjoamat turvallisuudet, kuten hajautettu vertaisverkosto, joka koostuu ei-luotettavista osapuolista. Tämä perusperiaate, jossa yksikään taho ei ole keskeinen luotettava taho, pakottaa jokaisen tahon pitämään toista silmällä ja todentamaan toisten osapuolten luotettavuutta. Tällöin jokainen hajautetun verkon osallistuja aktiivisesti pitää muiden osapuolten luotettavuutta yllä varmistaakseen, että kukin transaktio toteutetaan tehokkaasti ja että kaikkien tapahtumien tilanteesta on yhtenäinen kuva. Toisin sanoen järjestelmässä jokainen pitää jokaista silmällä.

Lohkoketjuteknologia on jo itsessään kryptografisiin menetelmiin perustuvaa, mikä takaa tietyntasoisen turvallisuuden jo ruohonjuuritasolla. Tämä teknologia pitää sisällään mm. transaktionaalisen datan salausta sekä julkisten ja yksityisten avainten käyttöä transaktioiden lukemiseen ja suorittamiseen. Ennen kuin mikään taho suorittaa transaktiota hajautetussa verkossa, transaktio tulee jokaisen osallistujan todennettavaksi ja varmistettavaksi, ja näin koko verkosto pääsee yhtenäiseen käsitykseen transaktion luotettavuudesta, mikä vahvistaa turvallisuutta. Nämä periaatteet on implementoitu älysopimuksiin niiden lohkoketjuteknologiaan perustuvan luonteen takia.

Älysopimusten muuttumattomuus on kiistanalainen ominaisuus, sillä siitä on sekä hyötyä että haittaa osapuolille. Älysopimusten muuttumattomuus ja niiden hyödyntämä lohkoketjuteknologia takaa sen, etteivät ulkopuoliset tahot kykene hyökkäämään sopimusta vastaan. Tämä takaa myös suojaa mahdolliselta korruptiolta osapuolien välille, sillä kumpikaan osapuoli, ilman etukäteen sovittuja tarkkoja ehtoja, ei voi tehdä vapaasti muutoksia sopimukseensa. Väärennykseltä suojaava ja yksityisyyttä tarjoavaa älysopimusta voitaisiin hyödyntää esimerkiksi äänestyksissä, joissa äänestäjän yksityisyys on taattu ja mahdollinen väärentäminen tai korruptio täysin ulkopuolisten tavoittamattomissa.

Yksi älysopimusten keskeisistä ongelmista on niiden yksityisyys; kuinka saada arkaluontoista tietoa piiloon sopimuksesta kuitenkin pitäen sopimuksen julkisella lohkoketjulla? Vuonna 2019 julkaistussa tutkimuksessa *Mixicles: Simple Private Decentralized Finance* [17] tuodaan esiin teknologia nimeltä Mixicles. Nazarov haastattelussa [18] selittää, että Mixiclet “toimivat erottamalla eri osia sopimuksesta useampiin sopimuksiin, joista osa on lohkoketjulla ja osa sen ulkopuolella. Tämä tarkoittaa, että maksutapahtumien tulokset eroavat sopimuksen tuloksista, mutta ketjun ulkopuolisen osan yksityisyyttä käytetään luomaan yksityisyyttä koko sopimukselle. Ulkopuolinen ei siis voi päätellä, mitä sopimuksessa tarkalleen tapahtui, vaikka se suoritettiin.” Mixicle-teknologia tarjoaa ratkaisua yhteen keskeisimpään ongelmaan älysopimuksissa ja täten tehostaa älysopimusten turvallisuutta ja yksityisyyttä entisestään.

Kustannustehokkuus

Kustannustehokkuus on monilla yrityksillä ja tahoilla yksi suurimmista prioriteeteista, ja yrityksiä hallinnoivat tahot pyrkivät aina löytämään tapoja, joilla parantaa kustannustehokkuuttaan, sillä yritysten ja muiden tahojen päätavoite on usein tehdä liikevoittoa. Älysopimusten implementaatio vie pois kolmannen osapuolen, joka on perinteisten sopimusten peruspiirre. Kolmannen osapuolen poistaminen tarkoittaa ylimääräisten kulujen leikkaamista niin yritysten kuin yksittäisten tahojen sopimusluontoisissa transaktioissa. Ottaen huomioon kansainvälisten yritysten sopimuskulut, jotka voivat nousta tuhansiin dollareihin sopimukselta, ja koostua lukuisista kuluista, kuten laillisesta konsultoinnista, hankinta-ajasta, toiminta-, suunnittelu- tai projektinhallinnasta, rahoituksesta ja riski- ja sääntelytoiminnasta, näiden kulujen leikkaaminen tehostaa yrityksen kustannustehokkuutta huomattavasti. [19.]

3.3 Ethereum-ekosysteemi

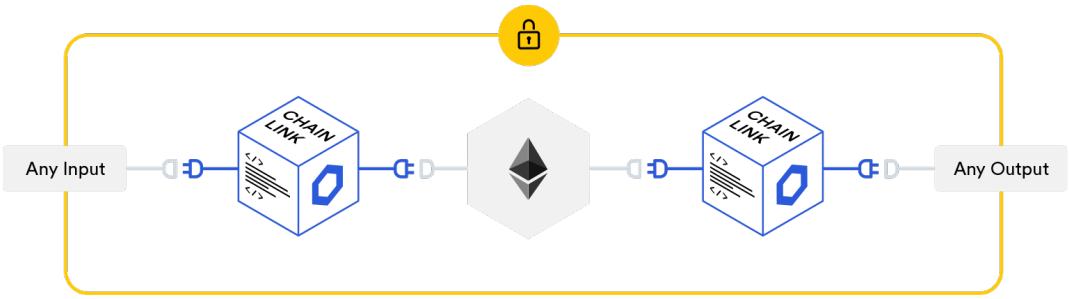
Ethereumin kehitti kryptovaluuttatutkija ja ohjelmoija Vitalik Buterin yhteistyössä englantilaisen ohjelmoijan Gavin Woodin kanssa vuonna 2013. Ethereum on avoimen lähdekoodin julkinen lohkoketjuteknologiaan perustuva hajautettu tietojenkäsittelyalusta, verkosto ja käyttöjärjestelmä älysopimuksille ja dApp-sovelluksille. Vertailuksi bitcoin on lohkoketjuteknologiaa hyödyntävä kryptovaluutta ja maksujärjestelmä, joten kyse on siis varsin eri asiasta. Sitä kuvaillaan “uuden ajan internetiksi”, joka on neutraali, jota ei hallinnoi mikään yritys tai henkilö, ja jonka avulla kaikilla osallistujilla on pääsy avoimeen taloudelliseen järjestelmään, johon raha ja maksut ovat sisäänrakennettuja. [21.]

Vuonna 2015 julkaistu Ethereum on ollut maailman johtava ohjelmoitava lohkoketju, mikä tarkoittaa että kehittäjät voivat luoda erinäisiä hajautettuja sovelluksia (engl. decentralized applications, lyhennetty dapps), joiden pääpiirre on luotettavuus, mikä tarkoittaa, että kun sovellus julkaistaan Ethereumilla, se toimii aina, niin kuin se on ohjelmoitu toimimaan eikä sitä hallinnoi mikään taho. Kehittäjät ovat luoneet Ethereumille jo lukuisia kryptovaluuttalompakoita, taloudellisia sovelluksia, hajautettuja markkinoita, pelejä ja useita muita käytännön sovelluksia. Ethereumin lohkoketjun natiivi valuutta on ether. Jokaisesta tapahtuneesta suoritteesta, kuten älysopimuksen suorittamisesta, vaaditaan Ethereum-verkossa suoritusmaksu, riippuen siitä, kuinka paljon laskentatehoa, kaistaa ja tilaa suorite vie. Tämän suoritusmaksun ansiosta verkossa olevalle validoijalle taataan alkuperäisen ennakkomaksun saaminen, vaikka suoritus epäonnistuisi, mikä kannustaa verkkoon osallistuvia tahoja toimimaan validoijana. Toiseksi, kaikkia suoritteita ajetaan niin kauan, kuin niihin varattu suoritusmaksu riittää. Tämä estää liian pitkien suoritteiden ja turhien kulujen aiheutumisen verkostossa. Suoritusmaksut maksetaan gas-yksikkönä, jota varten suorittava taho varaa tarpeellisen määrän etheriä ennen suoritusta. [21.]

3.4 Oraakkelit datan varmentamisen ratkaisuna

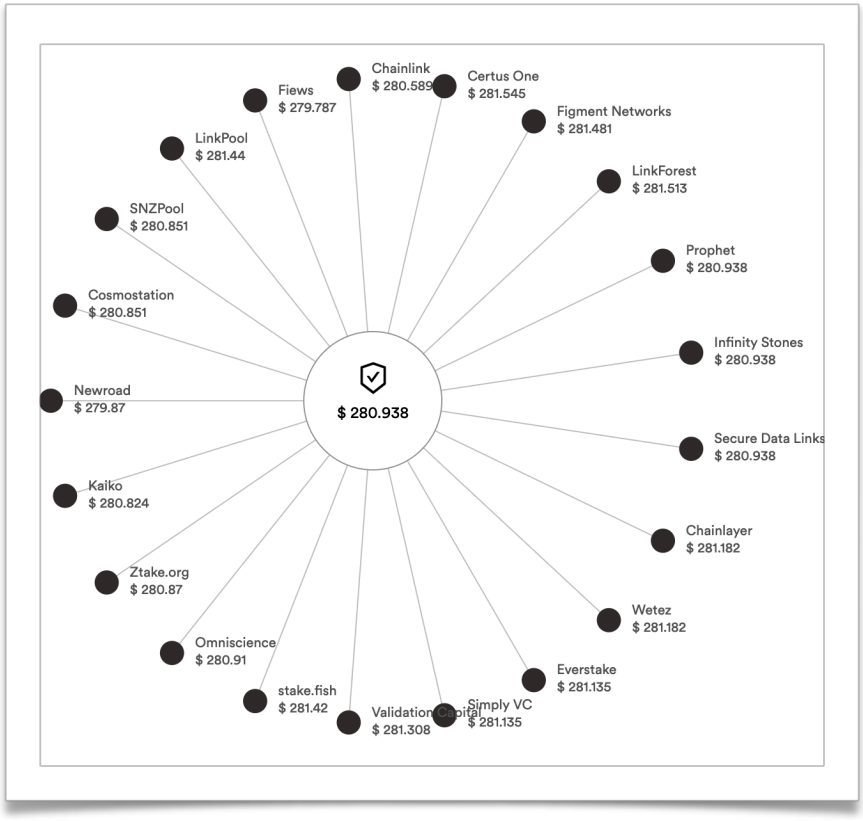
Älysopimukset eivät voi natiivisti hakea dataa ulkoisista tietosyötteistä, sillä se olisi turvallisuusriski ja mahdollistaisi ulkoisten lähteiden vaikuttamisen sopimuksen sisällä. Väliohjelmistoja (engl. Middleware) on kehitetty tehostamaan älysopimusten luotettavuutta ja tehokkuutta, ja tässä työssä keskitytään niistä yhteen nimeltä Chainlink. Termi väliohjelmisto viittaa ohjelmistokomponenttiin, joka toimii osien tai sovelluksien välisenä rajapintana tai palveluna. Chainlink on hajautettu oraakkeliverkosto, jonka tarkoitus on mahdollistaa älysopimusten yhdistäminen ulkopuolisiin tietosyötteisiin lu-

otettavasti ja turvallisesti. Chainlink toimii ohjelmistona, joka tarjoaa monitasoisen turvallisuuden tiedon todennukseen, ennen kuin sitä syötetään sopimukseen. Väliohjelmiston tarkoitus on sallia tietosyötteen tuloa mistä tahansa lähteestä, ja sen turvallista, ja tarkkaa syöttöä mihin tahansa päätteeseen, kuten kuva 13 havainnollistaa.



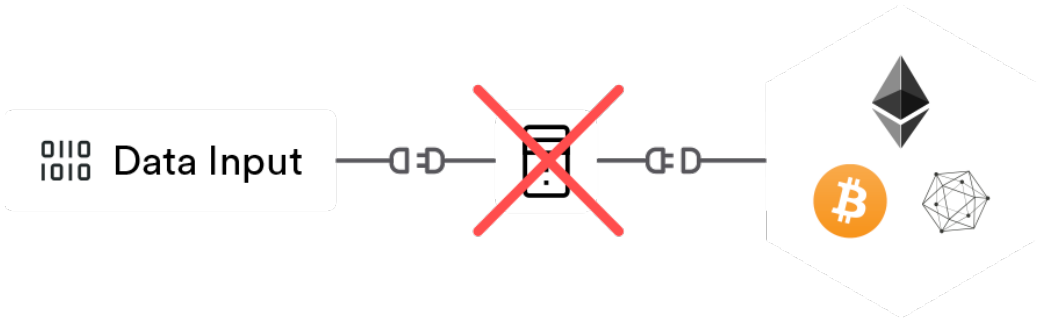
Kuva 13. Chainlink-väliohjelmisto toimii turvallisena datan varmentajana minkä tahansa syötteen ja päätteen välillä [35].

Hajautettu oraakkeliverkosto tarkoittaa käytännössä hajautettua verkostoa, jota ylläpitää useampi Chainlink-noodi, jotka kaikki varmentavat tietosyötteiden luotettavuuden demokraattisin menetelmin. Kalliissa sopimuksessa virhemarginaalin tulisi olla pieni, sillä pienikin virhe voi maksaa jollekin osapuolelle tuhansia ellei jopa miljoonia tappioita, joten siihen sopimukseen syötetyn tiedon tulee olla tarkkaa ja luotettavaa. Mikäli haluttaisiin esimerkiksi tietää bitcoinin hinta tietyllä hetkellä sopimusta varten, se riippuisi monesta eri tekijästä, kuten havaitsijan maantieteellisestä sijainnista, pörssien hinnoitteluista ja muista mahdollisista muuttujista. Saadakseen mahdollisimman tarkan keskiarvon bitcoinin hinnasta, voi käyttää lukuisia väliohjelmistoa ja noodeja, jotka hakevat tietosyötteistä dataa bitcoinin hinnasta. Validoituaan ja päästyään tarpeeksi tarkkaan keskiarvoon tietosyötteiden tarjoamien hintojen välillä voi saada luotettava n keskiarvon bitcoinin hinnasta, joka voidaan lyödä lukkoon sopimusta varten. Käytännössä siis noodit käyvät läpi jokaisen tietosyötteen antamaa dataa, todentavat keskenään sen olevan oikeaa ja luotettavaa, minkä jälkeen prosessi toistetaan jokaiselle käytettävälle tietosyötteelle ja lopulta lasketaan näiden laskettu keskiarvo, kuten kuva 14 havainnollistaa. [liite 1.]



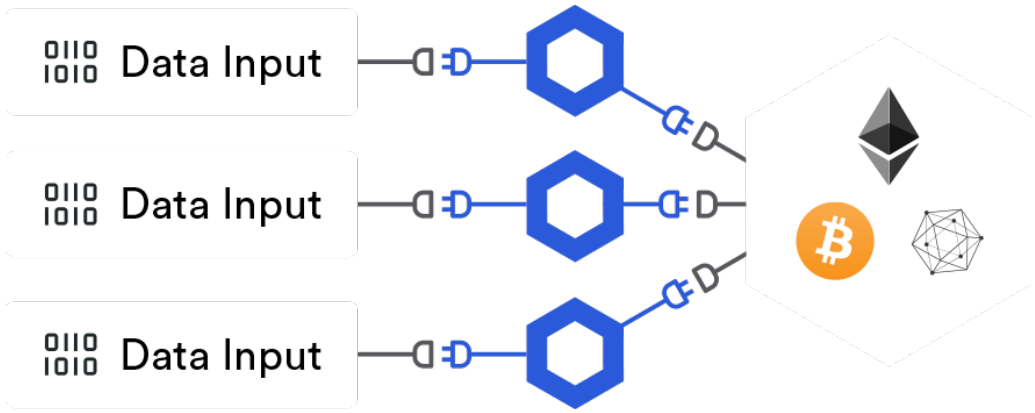
Kuva 14. Hajautettu verkosto, jossa jokainen noodi palauttaa Ethereum-kryptovaluutan hinnan Yhdysvaltain dollarin muodossa (19.2.2020) ja laskee noodien tietosyötteiden keskiarvon [36].

Sopimukset eivät voi nojata yhteen tietosyötteeseen, sillä jos ainut sopimukseen liitetty tietosyöte kaatuu tai sen tarjoama data on saavuttamattomissa, koko sopimuksen toiminta kaatuu. Tätä heikkoutta kutsutaan termillä “Single point of failure”, SPOF (kuva 15).



Kuva 15. Yksittäisen tietosyötteen malli, joka havainnollistaa SPOF-ongelman [35]. https://chain.link/features/

Ratkaisu yksittäisen tietosyötteen malliin perustuvaan järjestelmään on käyttää hajautettua oraakkeliverkkoa, kuten Chainlink-väliohjelmistoa, joka tarjoaa samantasoista turvallisuutta kuin älysopimukset itse. Tämä malli ratkaisee SPOF-ongelman ja tarjoaa tarkempaa ja luotettavampaa dataa lukuisista tietosyötteistä (kuva 16).



Kuva 16. Chainlink-oraakkeliverkosto. Malli ratkaisee SPOF-ongelman ja mahdollistaa useamman tietosyötteen käytön [35].

3.5 Käyttötarkoituksia

Ymmärtääkseen älysopimusten käyttötarkoituksien mahdollisuuksia täytyy tarkastella, missä kaikkialla on sopimusluontoisia suhteita. Jokainen varallisuuden siirto on sopimusluontoinen transaktio, jossa määritellään tiettyjä ehtoja, kuten osapuolet, siirron tarkoitus ja määrä sekä muita mahdollisia laillisia piirteitä, kuten perintöön tai vakuutukseen liittyviä. Insinöörityössä haastateltiin myös Chainlink-yhtiön yhteisömanageria Rory Piantia älysopimusten käyttötarkoituksista ja niiden implementaatiosta. (liite 1).

Kaupankäynti

Haastattelussa Rory Piant [37] havainnollisti älysopimusten käyttöä osakekaupassa, jossa esimerkiksi halukas taho haluaa ostaa jonkun yhtiön osaketta 800 Yhdysvaltain dollarin hintaan. Hän voisi luoda digitaalisen sopimuksen, johon halukas ostaja syöttäisi varat kryptovaluutan muodossa, ja sopimus tarkastaisi tietyin väliajoin osakemarkkinoilta kyseisen yhtiön osakkeen arvoa. Ja kun osakkeen hinta osuisi 800 Yhdysvaltain dollarin kohdalle, sopimuksen ehdot täyttyisivät ja ostajalle siirtyisi yksi osake. Tähän ei tarvittaisi perinteistä pörssimeklaria eikä välttämättä edes ostajan läsnäoloa,

sillä sopimusehtojen täyttyessä transaktio suoritettaisiin automaattisesti. Samaa käytäntöä voitaisiin käyttää useammissa kaupankäynnin muodoissa kuin vain osakekaupoissa.

Pankki- ja rahoituspalvelusopimukset

Älysopimusten konseptia voidaan hyödyntää pankki -ja rahoitusalalla laajalti mm. lainasopimuksissa, pankkimaksuissa, obligaatioissa ja monissa muissa pankin toimissa, sillä lähes kaikki pankin toiminta perustuu osapuolten väliseen valuutan siirtoon tai hallinnointiin. Pankin toiminta on aina perustunut luottamukseen ja turvallisuuteen pankin ja asiakkaan välillä. Älysopimusten tuoma ehdollinen määrittely tuo tarkkuutta, turvallisuutta ja luottamusta asiakkaille siitä, että pankin maksu tai laina toteutuu täysin sopimuksessa määriteltyjen ehtojen mukaisesti, mikä poistaa mahdolliset inhimilliset virheet tai väärinkäsitykset. Sopimuksen määrittelyn jälkeen pankin ei tarvitse käyttää resursseja sopimuksen valvomiseen, sillä voidaan luottaa siihen, että sopimus toimii määritettyjen funktioiden mukaisesti. [37.]

Vakuutussopimukset

Piant mainitsi haastattelussa [37] myös käyttötarkoitusten laajentuvan muuhunkin kuin vain finanssialan tarkoituksiin, kuten vakuutuksiin. Piantin mukaan on kehitteillä mm. lentovakuutuksiin perustuvia älysopimuksia. Lentovakuutukset ovat muodollisuutensa takia helppoa siirtää älysopimuksiin, sillä lentoliikenteestä on saatavilla suuria määriä avointa dataa tietosyötteiden kautta. Dataa, kuten lentojen aikatauluja, lähtöjä ja saapumisia, käyttäen älysopimukseen voitaisiin luoda henkilön lennolle vakuutus, ja josta sopimus voisi tarkastaa lennon mahdolliset myöhästymiset. Mikäli henkilön lento myöhästyisi esimerkiksi kolme tuntia eikä vakuutettu henkilö ehtisi jatkolennolle, sopimus voitaisiin automaattisesti suorittaa ehtojen täyttyessä ja vakuutetulle henkilölle voitaisiin tarjota sopimukseen määritetyt korvaukset, kuten tilapäinen majoittuminen tai lippujen hyvitys.

Kansainvälinen kuljetus ja geofencing

Kansainvälinen kuljetus sekä geofencing eli sellaisten virtuaalisten rajojen määrittäminen, joiden ylittäminen aiheuttaa jonkin tapahtuman, kuten maksun suorittamisen, ovat Piantin [37] mukaan sektoreita, jotka voisivat käyttää älysopimusteknologiaa. Jokaisen tuotteen kansainväliseen kuljetukseen liittyy eri vaihdeiden aikana useita maksuja, joita voitaisiin älysopimuksessa suorittaa niiden vaiheiden tapahtuessa ilman ihmisen varmennusta. Koko prosessi tavaran lähettämisestä alkuperäisestä pisteestä asiakkaan

ovelle vientiin saakka voidaan digitaalisesti varmentaa ja täten nopeuttaa kuljetukseen liittyviä transaktioita. Sopimukseen voitaisiin määritellä esimerkiksi, että kun tavara lähtee satamasta, osa summasta menee satamalle. Kuljetuksen aikana osa summasta menisi laivan vaatimiin kuluihin, ja tavaran saapuessa loput menisivät käsittely -ja mahdollisiin tullikuluihin. Tavaran saapuessa jokaiseen pisteeseen matkalla se kuitattaisiin digitaalisesti saapuneeksi tiettyyn pisteeseen, ja siihen pisteeseen liittyvä maksu voidaan todennuksen ohella suorittaa. Saapuessaan päämäärään kuljetusyhtiö voisi ottaa vastuun kuljetettavasta tuotteesta, kuljettaa se asiakkaalle ja asiakas voisi kuitata digitaalisesti tuotteen saapuneen, jolloin sopimuksen viimeinen vaihe suoritettaisiin. Näin kaikki maksutapahtumat tavaran kuljetuksessa automatisoitaisiin eikä niihin tarvitsisi ihmisen todennusta. Asiakkaat voisivat näin muodostaa suoria yhteyksiä kuljettajaan, ja yritykset voisivat GPS-datan avulla paikantaa kuljetuksen tilannetta ja siihen liittyviä maksuja.

Omistajuuden määrittely

Lohkoketjuteknologian myötä älysopimusten ominaisuutena on jokaisen transaktion luotettava ja turvallinen tallennus. Kaikki älysopimuksessa tapahtuneet transaktiot tallennetaan lohkoketjuun kronologisessa järjestyksessä, josta tapahtumia voi auditoida tarpeen mukaan. Omaisuuden omistajuus voidaan myös täysin määritellä uudella tavalla älysopimusten kontekstissa, ja sen siirto taholta toiselle on tehokkaampaa kuin perinteisin menetelmin. Mitä useampia omaisuuden muotoja digitalisoidaan (ks. kryptovaluutat), kuten kiinteistöjä, osuuksia tai vaikkapa fyysistä kultaa, sitä enemmän turvalliselle ja helposti siirrettävälle omistajuudelle on tarvetta. Älysopimukset tarjoavat ympäristön, jossa omistajuuden nimittäminen on yksinkertaista ja josta sen voi kuka tahansa myöhemmin tarkistaa.

3.6 Älysopimusten haasteet

Vaikka älysopimukset ovat idean tasolla vallankumouksellinen teknologia, niiden on silti ylitettävä tiettyjä haasteita saadakseen laajaa käyttöä ja standardisointia. Tässä luvussa käydään läpi muutamia älysopimusten haasteita ja niiden mahdollisia ratkaisuja.

Luotettavan tiedon ongelma

Älysopimukset ovat epäilemättä vallankumouksellinen teknologia, jolla on kuitenkin laajentumisen kannalta isoja haasteita ratkaistavana. Suurin kysymys älysopimuksiin liittyen on, kuinka saadaan oikean maailman dataa älysopimuksiin, jotka ovat lohkoketjul-

la “eristettynä”. Miten saadaan luotettavaa tietoa siitä, mikä oli tietyn osakkeen hinta maaliskuun kolmantena päivänä kello 16.48 vuonna 2019? Tätä ongelmaa kutsutaan nimellä “The oracle problem” eli oraakkelin ongelma. Niin kutsutut oraakkelit valvovat ja validoivat reaaliaikaisia tietosyötteitä, jotka palauttavat tietoa lämpötilasta, kurssivaihteluista, urheilutuloksista tai vaikkapa lentoyhtiön lennoista. Ilman oikean maailman dataa älysopimukset ovat kuin kaupunki ilman sähköjä.

Viimeaikainen kehitys on kuitenkin tarjonnut ratkaisuja tähän ongelmaan, muun muassa Ari Juelsin kehittämän Town Crierin [25], jonka tarkoitus on toimia niinsanottuna siltana Ethereum-lohkoketjun ja HTTPS-kykenevien tietosyötteiden välillä. HTTPS on sitä edeltävien HTTP-protokollan ja TLS/SSL-protokollan yhdistelmä, jota käytetään turvalliseen tiedonsiirtoon internetissä. Town Crierin pääpiirre on ohjelma, jota suoritetaan turvallisella, eristetyllä laitteistolla, jota kutsutaan turvalliseksi erillisalueeksi. Erillisalueen tarkoitus on suojella ohjelmaa mahdollisilta hyökkäyksiltä ja pitää ohjelman laskenta yksityisenä. Ohjelma vastaanottaa pyyntöjä älysopimukselta, joka voi pyytää dataa liittyen siihen, peruttiinko lento, ja sen jälkeen hakea vastauksia tietosyötteistä pyyntöön liittyen ja palauttaa vastaus lohkoketjuun. Salauksen ja turvallisen laitteiston avulla ohjelma palauttaa todisteen siitä, että lennon vakuutussopimukseen palautettiin dataa Town Crieriltä ja dataa ei ole päässyt kukaan väärentämään.

Town Crier on luotettavampi tietosyöte kuin useat muut, mutta tarjoaa rajoitetun, keskitetyn palvelun. Toisin sanoen Town Crier toimii keskeisenä toimijana tiedonsyötölle, joka ei ole täydellinen luottamusmalli, sillä ympäristössä luotetaan yhteen keskeiseen toimijaan ja sen todenmukaisuuteen. Tähän ongelmaan tarjoaa ratkaisua Chainlink, jonka keskeinen tarkoitus on toimia hajautettuna tietoverkkona oraakkelien datalle, jottei palveluiden tarvitse nojata yhteen keskitettyyn toimijaan. Kryptografian avulla Chainlink-palvelu tarjoaa todisteen lohkoketjulle siitä, että data on sitä, mitä oli alun perin pyydetty. Asiakkaat voivat maksaa eritasoisesta hajautuksesta, ja verkostoa pyörittävät noodit tuottavat voittoa datan palauttamisesta, mihin Chainlinkin toimintamalli perustuu.

Hidas omaksuminen

Vaikka älysopimukset pyrkivät poistamaan mahdollisimman useissa tapauksissa välikäden tai kolmannen osapuolen tarpeen, monimutkaisen ja teknisen luonteen takia jokainen älysopimus halutaan kuitenkin auditoida, mielellään jonkun niihin erikoistuneen toimesta, kuten perinteisissä sopimuksissa asianajajat toimivat. Rory Piantin mukaan [37] älysopimusten auditoiminen on toiseksi tärkein vaihe niiden kehittämisessä, ja ala on vielä toistaiseksi melko rajallinen, sillä älysopimuksiin erikoistunei-

ta kehittäjiä ei ole vielä paljoa. Auditoijan tulee ymmärtää sopimuksen kehitetyn sopimuksen toiminnallisuus ja osata määritellä, vastaako sen toteutus sopimuksen osapuolten vaatimuksia. Tekninen vaikeus ja monimutkaisuus ovat ehdottomia esteitä älysopimusten omaksumiselle, ja niiden kehittämisen ja auditoimisen suoraviivaistaminen on tärkeää laajan käyttöönoton kannalta.

Älysopimusten implementointiin liittyviä haasteita ovat mm. sopimuslakeihin ja lohkoketjuihin liittyvät kansainväliset lainsäädännöt. Todennäköisesti lohkoketjuteknologian edetessä ja älysopimusten yleistyessä on mahdollista luoda älysopimuksiin ja lohkoketjuihin kohdistuvaan lainsäädäntöön tietynlaisia standardeja, jotka helpottavat niiden käyttöönottoa. Merkittävä haaste on myös tämänhetkisten älysopimusten tekninen implementointi, sillä niihin liittyvä teknologia on toistaiseksi uutta ja jatkuvasti kehittyvää eikä älysopimusten tuomia seurauksia vielä täysin ymmärretä.

Muuttumattomuus

Älysopimusten muuttumattomuus on yleisesti ongelmapiirre, vaikka se myös tarjoaa turvaa sille, ettei mikään taho voi muuttaa sopimuksen tietoja sen jälkeen, kun se on luotu. Tähän on kuitenkin luotu tilapäisiä ratkaisuja, kuten uuden, muokatun sopimuksen luominen ja tietosyötteen reitittäminen uuteen sopimukseen, mutta mitään keskeistä tapaa ei ole kehitetty. Tällä hetkellä älysopimusta ei voida terminoida tai muuttaa ilman, että siitä toiminnosta on sovittu sopimusta laadittaessa, mikä tuo esiin ongelmia. Esimerkiksi kahden osapuolen jatkuvassa sopimuksessa, kuten kuukausimaksusopimuksessa, palvelua tarjoava osapuoli ei välttämättä halua terminoida sopimusta yhden myöhästyneen maksun takia, sillä suhde voi olla pitkällä aikavälillä hyödyllinen, mutta mikäli myöhästynyttä maksua ei ole erikseen määritelty käsiteltävän jollain muulla kuin automaattisella terminoinnilla, sopimus luultavasti terminoituisi. Älysopimusten automaattinen suoritus ei siis välttämättä ole aina sovussa sen suhteen miten monet palveluntarjoajien ja asiakkaitten väliset suhteet toimivat.

Julkaisemassaan tutkimuksessa “*Setting Standards for Altering and Undoing Smart Contracts*” [15] Ari Juels ehdottaa implementoimaan perinteisten sopimusten muokkaamisen ja mitätöinnin standardeja myös älysopimuksiin. Nämä standardit voidaan jakaa kolmeen kategoriaan: mitätöinnin tai muokkaamisen oikeuttamiseen, sovinnolliseen mitätöintiin ja tuomariston mitätöintiin. Mitätöinnin oikeuttamisella tarkoitetaan ennalta määritettyä tahoa, jolla on oikeus sopimuksen mitätöintiin, mikäli sitä edellyttävät ehdot täyttyvät. Sovinnollinen mitätöinti tai muokkaaminen tarkoittaa sopimukseen kirjattujen osapuolten yhteistä ymmärrystä sopimuksen mitätöinnistä, mikäli sitä edellyttävät ehdot täyttyvät. Tuomariston mitätöinti tai muokkaaminen tarkoit-

taa sopimukseen määritettyä kolmatta osapuolta, joka toimii sovittelijana ja oikeudellisena toimijana sopimuksen muokkaamiselle tai mitätöinnille riitatilanteessa, jossa esimerkiksi toinen osapuolista on käsittänyt sopimuksen toiminnan väärin.

Sopimuksen muokkaamista tarvitaan tilanteissa, joissa koko sopimusta ei ole tarkoitus mitätöidä, vaan ainoastaan siihen määriteltyjä ehtoja. Muokkaaminen toimisi tehokkaana mekanismina, mikäli sopimuksen olosuhteet muuttuvat yllättäen, kuten sen laillisuus tai osapuolten ehtojen muutos, jolloin alkuperäinen sopimus voitaisiin säilyttää. Kuten mitätöinnissä, sopimuksen muokkaamiseen pätisivät samat kolme edellä mainittua standardia. Sopimusten mitätöinti ja niiden muokkaus ovat vilkkaan kehityksen alla ja todennäköisesti tulevat olemaan perusominaisuuksia tulevaisuudessa. Vaikka sopimusten muuttumattomuus toimiikin älysopimusten luonnetta vastaan, se heijastaa sitä tosiasiaa, että älysopimukset saavat kaupallista hyväksyntää vain, jos ne heijastavat sopimusten liiketoiminnan todellisuutta.

Moniselitteisyyden sisällyttäminen

Älysopimusten tarkkuus ja automaatio voivat olla rajoittava tekijä osapuolten neuvotellessa sopimuksen laatimisesta, sillä ne rajoittavat moniselitteisiä määritelmiä, joita osapuolet mahdollisesti haluaisivat tai ei haluaisi sisällyttää sopimuksessa. [16.] Sen sijaan, että osapuolet laatisivat tarkkoja määritelmiä jokaiselle mahdolliselle tapahtumalle, he saattavat päättää asiasta vasta, kun jotain odottamatonta tapahtuu sopimuksen ollessa jo voimassa, ja muuttaa sen hetken tarpeiden mukaan. Osapuolet saattavat tarkoituksellisesti päättää jättää määritelmän epäselväksi sopimukseen voidakseen antaa itselleen mahdollisuuden väittää, että määritelmä olisi tulkittava heidän hyväkseen. Tällainen menettely on huomattavasti vaikeampaa älysopimuksessa kuin paperisessa sopimuksessa, sillä älysopimus ei voi sisältää epäselviä ehtoja eikä kaikkia mahdollisia odottamattomia tapahtumia voida määritellä. Riittävän objektiivisten ehtojen määrittelemiseen menee vielä aikaa monilla älysopimuksia hyödyntävillä aloilla. Tämänhetkiset älysopimukset suorittavat suhteellisen yksinkertaisia toimintoja, joissa jos–niin-määritelmät ovat selkeitä. Kun älysopimukset kehittyvät monimutkaisemmiksi, osapuolet voivat olla erimielisiä siitä, voidaanko jokin ehto määritellä tarpeeksi tyydyttävästi objektiiviseen sopimukseen.

3.7 Tulevaisuuden näkymät

Toistaiseksi useiden älysopimusten toiminnallisuus on vielä yksinkertaist,a kuten varainsiirtoa tahojen välillä kryptovaluuttojen muodossa, kun tietyt ehdot täyttyvät.

Lohkoketjuteknologian kasvaessa ja omaksuessa laajemmalle myös älysopimusten hyödyt ja käyttömahdollisuudet kasvavat, kun useampia omistuksia saadaan ns. lohkoketjulle. Joka tapauksessa älysopimusteknologia on vielä vuosien päässä siitä, että ne voisivat määritellä subjektiivisia laillisia kriteerejä, kuten osapuolen tyytyväisyys suoritettuihin standardeihin tai se, pitäisikö tietty hyvitys suorittaa ja maksaa.

Ottaen huomioon, kuinka monentyyppisiä sopimusluontoisia asioita esiintyy useilla eri teollisuuden sektoreilla aina logistiikasta maksuliikenteeseen, voidaan alkaa käsittää älysopimusten mahdollista vaikutusta tulevaisuuteen. Maksuliikenteen kuluista tulisi huomattavasti halvempia ja transaktioista luotettavampia sekä automaattisia. Mitään välikäsiä tai todentajia ei enää tarvittaisi, joten myös maksuliikenteen tehokkuus ja nopeus parantuisi. Logistiikassa toimitusketju automatisoituisi, millä olisi suuri vaikutus globaaliin kaupankäyntiin, sillä jokaista toimitusketjun sisäistä transaktiota ei tarvitsisi ihmisen todentaa ja hyväksyä. Kaikki tämä kuitenkin edellyttää erittäin kehittyneitä sopimuksia, jotka ovat laillisesti päteviä ja joihin on koodin muodossa määritelty satoja tai jopa tuhansia eri ehtoja kehittyneen automaation saavuttamiseksi.

Piantin mukaan [37] yritysten omaksumisen kannalta tärkeintä on kannustaa yrityksiä näkemään älysopimusten kustannushyödyt. Älysopimusten automaation ja turvallisuuden takia yritykset voivat jo säästää kuluissa poistamalla kolmansien osapuolten valvonnan, mutta yritysten pitää kuitenkin punnita riskit ja hyödyt älysopimusten suhteen. Mitä enemmän älysopimukset alkavat standardisoitua, sitä vähemmän riskejä niihin liittyy. Jos jotain sopimusmallia on käytetty jo satoja tai tuhansia kertoja, yritykset tietävät, että niihin voi luottaa, mikä alentaa yritysten havaintoa niihin liittyvistä riskeistä. Piantin mukaan monet vanhat yritykset, kuten pankit, ovat alkaneet implementoida pilottiohjelmia ja konseptitodistuksia, ja hänen mukaansa on vain ajan kysymys, että useat instituutiot siirtyvät perinteisistä sopimuksista älykkäisiin.

4. Älysopimuksen toteutus Solidity-kielellä

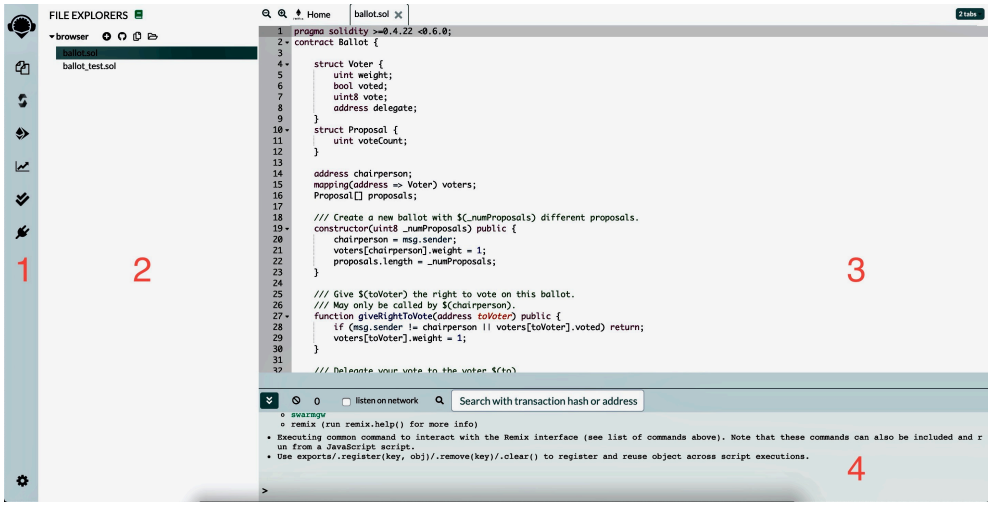
Solidity-ohjelmointikieli

Solidity on olioperustainen korkean tason ohjelmointikieli älysopimusten kehitykseen. Sen kehittivät Gavin Wood ja Christiam Reitwiessner vuonna 2014. Solidityllä kehitetyt älysopimukset ovat Ethereum Virtual Machinella. Solidity kääntyy bittikoodiksi, joka voidaan suorittaa Ethereum-virtuaalikoneella. [21.]

Remix-web-kehitysympäristö

Remix on tehokas, avoimen lähdekoodin työkalu, jolla käyttäjä voi kirjoittaa Solidity-kielellä älysopimuksia suoraan verkkoselaimessa. Remix on selainpohjainen Integrated Development Environment (IDE) eli kehitysympäristö. Remix on kehitetty JavaScript-ohjelmointikielellä, joten se tukee paikallista ja selainpohjaista käyttöä. Remixin ominaisuuksiin kuuluu myös testaus, älysopimusten käyttöön ottaminen ja lukuisia muita ominaisuuksia. [38.]

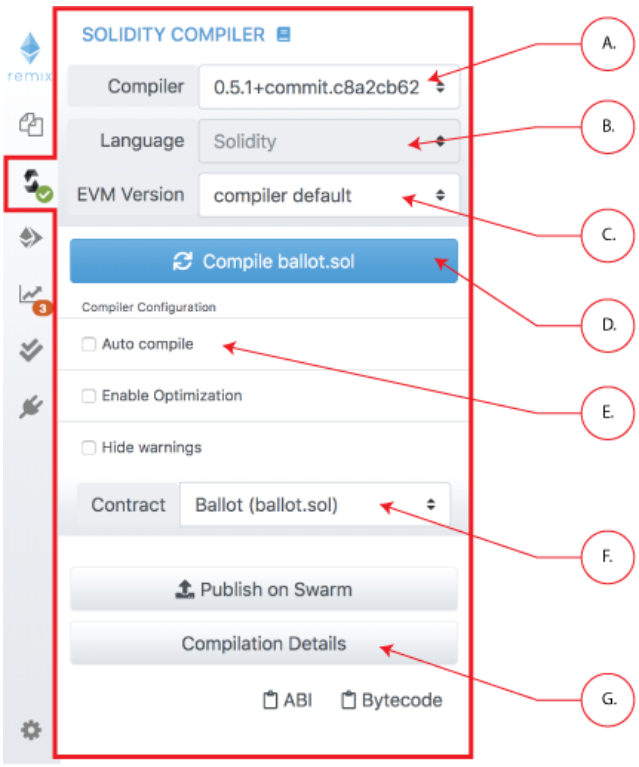
Remix-kehitysympäristön päivitetty näkymä (kuva 17) koostuu neljästä eri pääosasta: kuvakepaneelista (1), sivupaneelista (2), pääpaneelista (3) ja terminaalista (4).



Kuva 17. Remix-kehitysympäristön oletusnäkymä.

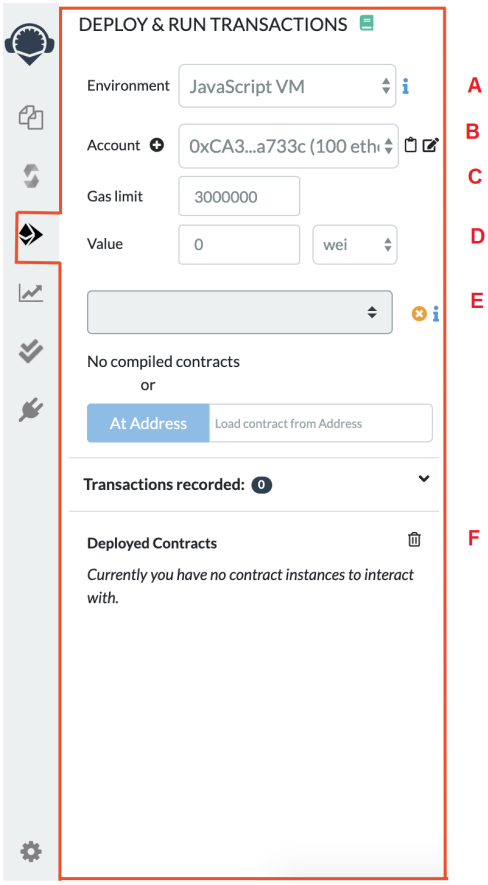
Kun käyttäjä lataa Remix-kehitysympäristön, kuvakepaneeli näyttää kolme vakiokuvaketta: File Explorers, Plugin Manager ja Settings. Remixin uusimmassa versiossa suurin osa ominaisuuksista on lisäosia, joten Plugin Manager on käyttäjälle tärkeä. Plugin Manager mahdollistaa käyttäjän lataavan vain niitä toiminnallisuuksia, mitä hän tarvitsee. Ladattuja lisäosia voi kytkeä päälle ja pois päältä tarpeen mukaan. Remix alustaa jokaisen session muutamalla oletustiedostolla: Storage.sol, Owner.sol, Ballot.sol ja Ballot_test.sol. Sol -tiedostopääte viittaa Solidity-kielen tiedostoon. Nämä tiedostot toimivat yksinkertaisina malleina uusille käyttäjille, mutta tämän työn kontekstissa niitä ei tarvita.

Solidity-kuvaketta painaessa käyttäjälle avautuu Solidityn kääntäjän näkymä ja sen hallinnointipaneeli (kuva 18).



Kuva 18. Solidity-kääntäjän hallinnointipaneeli.

Kääntäjän version käyttäjä voi valita ensimmäisestä valikosta (A). Kääntäjä kääntää koodin, kun käyttäjä painaa Compile-painiketta (D). Käyttäjä voi halutessaan automaattisesti suorittaa kääntäjän joka kerta, kun tiedosto tallennetaan tai kun toinen tiedosto valitaan aktivoimalla auto compile -valintaruutu (E). Remix-kääntäjä tukee kaikissa 0.5.7-versiota uudemmissa versioissa Yul-ohjelmointikieltä ja sen tiedostoja, jonka käyttäjä voi valita alasvetovalikosta kohdassa Language (B). EVM Version -valikosta käyttäjä voi valita eri versioita Ethereum-haaroista, mikä mahdollistaa kääntämisen toisille versioille (C). Compilation Details -painikkeesta avautuvasta näkymästä käyttäjä voi tarkistaa valitun Ethereum-haaran Metadata-osiosta (G). Jokaisen käännöksen jälkeen sopimusten listavalikko päivitetään ja siihen lisätään uudet sopimukset. Käännetyn sopimuksen voi valita Contract-valikosta (F). Kuvassa 19 näkyy deploy & run transactions-näkymä.



Kuva 19. Solidity deploy & run -hallinnointipaneeli

Kun sopimus on käännetty onnistuneesti ilman virheitä, se ilmestyy valikon alempaan välilehteen Deploy & Run Transactions. Välilehden valikosta voidaan valita suoritusympäristö, jolla sopimus suoritetaan (A). Vaihtoehtoina ovat Javascript VM, Injected web3 ja Web3 provider. Jos ympäristöksi on asetettu JsVM, Remix luo Account-valikkoon (B) oletuksena viisi eri osoitetta, joille on asetettu tilin arvoksi 100 etheriä testaamisen tarkoituksessa. Nämä etherit ovat päteviä vain testiympäristössä, eikä niillä ole oikeaa arvoa. Gas limit -kohdassa (C) asetetaan maksimimäärä gasia, jota transaktiot saavat käyttää. Value (D) määrittää, kuinka paljon valuutaa ja missä muodossa lähetetään sopimukseen tai funktioon, jonka tyyppi on payable. Onnistuneesti käännetyt sopimukset ovat valittavissa valikosta (E). Kun sopimus on valittu se tulee esille Deployed Contracts (F) -osioon, josta sopimuksen toimintaa voidaan testata. Osion ylle tallennetaan myös kaikki tapahtuneet transaktiot.

Remix pitää sisällään näiden lisäksi muita ominaisuuksia, mutta tämän työn kannalta edellä mainitut näkymät ovat olennaisia.

Ethereum-virtuaalikone

Ethereum Virtual Machine (lyhennettynä EVM) on tehokas, hiekkalaatikoitu (engl. sandboxed) käyttöjärjestelmä, joka on rakennettu jokaisen Ethereum-verkoston noodin sisään. EVM on vastuussa sopimusten bittikoodin suorittamisesta. Sopimukset kirjoitetaan usein korkean tason ohjelmointikielellä, kuten Solidityllä, minkä jälkeen se käännetään EVM-bittikoodiksi. Hiekkalaatikoitu tarkoittaa, että konekoodi on täysin eristetty isäntätietokoneen verkosta, tiedostojärjestelmästä tai muista prosesseista.

Jokainen noodi Ethereumin-verkossa pyörittää samaa instanssia EVM:stä, mikä mahdollistaa niiden suorittavan koodia samojen määritelmien mukaisesti. EVM on Turing-vahva, mikä tarkoittaa järjestelmää, joka on kykenevä suorittamaan mitä tahansa loogista vaihetta laskennallisessa funktiossa. EVM on välttämätön osa Ethereum-protokollaa ja keskeinen asia Ethereumin konsensusjärjestelmän kannalta. Se mahdollistaa kenen tahansa suorittaa koodia luottamuksesta riippumattomassa ekosysteemissä, jossa suorituksen lopputulos voidaan taata ja joka on täysin deterministinen, kuten älysopimusten tapauksessa. Mallityö toteutetaan käyttäen EVM-järjestelmää älysopimuksen kehityksessä. [21.]

4.3 Sopimuksen rakenne

Solidityllä kirjoitetut älysopimukset ovat verrattavissa olio-ohjelmoinnissa esiintyviin luokkiin. Jokainen sopimus sisältää tyypillisesti muuttujia, funktioita, funktion modifikaattoreita, tapahtumia, tietuetyyppejä ja listatyyppejä. Lisäksi sopimukset voivat periytyä muista sopimuksista, mikä on tyypillistä olio-ohjelmoinnissa. Solidity tarjoaa laajalti muita ominaisuuksia näiden lisäksi, mutta tässä työssä keskitytään vain esimerkin kannalta olennaisiin ominaisuuksiin. Tässä osiossa käydään läpi tyypillisen sopimuksen rakenne ja esimerkkejä edellä mainituista ominaisuuksista.

Tyypillinen rakenne Solidityssä kirjoitetulle sopimukselle on seuraavanlainen:

Sopimus alustetaan ilmaisemalla pienin Solidity-versio, jota sopimus tukee. Tämän jälkeen luodaan sopimus, jolle hyvien standardien mukaan tulee antaa mahdollisimman kuvaava nimi, sillä yhdessä tiedostossa voi olla useita, jopa satoja eri sopimuksia. Sopimuksen sisälle alustetaan kaikki siihen liittyvä, kuten muuttujat, konstruktorit, funktiot ja niiden modifikaattorit (esimerkkikoodi 1).

```
pragma solidity >=0.4.22 <0.6.0;
```

```
contract ContractName {
      <muuttujat>
      <muuttujien kartoittaminen>
      <konstruktori>
      <funktiot>
      <funktioiden modifikaattorit>
}
```

Esimerkkikoodi 1. Sopimuksen rakenteen malli.

Muuttujat

Solidityssä muuttujiin asetetaan arvoja, jotka tallentuvat pysyvästi sopimuksen muistiin. Älysopimuksissa useimmin käytetyt muuttujien tyypit ovat seuraavat:

Address, joka on kahdessa eri muodossa: address ja address payable, jotka molemmat pitävät 20-bittisen merkkijonon (Ethereum-osoitteen pituisen) sisällään, mutta address payable pystyy käsittelemään transfer- ja send-ominaisuuksia. Erottelun tarkoitus on muodostaa kaksi erilaista osoitetyyppiä, joista toinen voi vastaanottaa etheriä ja toinen ei, mikä pienentää virhemarginaalia silloin, kun sopimuksissa käsitellään omaisuutta.

Integer, joka voi sisältää merkittyjä ja etumerkittömiä kokonaislukuja 8 bitistä 256 bittiin asti (esimerkkikoodi 2). Kirjoitetaan muuttujana muodossa int/uint. Bittimäärä määritetään aina muuttujan lopuksi, esim. uint256. Int-muuttujat voivat olla negatiivisia, kun uint-tyypin muuttujat voivat olla minimiarvoltaan 0.

String-, Hex-, ja Enum-tyypit pitävät sisällään merkkijonoja. String-muuttuja pitää lainausmerkkien sisällä merkkijonoja, Hex-muuttuja ilmaisee heksadesimaalisia merkkijonoja ja Enum mahdollistaa käyttäjän luomien tyyppien määrittämisen.

```
pragma solidity >=0.4.22 <0.6.0;

contract SimpleStorage {
      uint storedData; // uint tyypin muuttuja
      // …
}
```

Esimerkkikoodi 2. Muuttujan alustaminen sopimuksessa.

Funktiot

Funktiot ovat sopimuksen koodissa suoritettava osa (esimerkkikoodi 3).

```
pragma solidity >=0.4.22 <0.6.0;
```

```
contract SimpleAuction [
      function bid() public payable { // Funktio
            // …
      }
}
```

Esimerkkikoodi 3. Funktion alustaminen sopimuksessa.

Funktioiden kutsut voivat tapahtua sisäisesti tai ulkoisesti, ja niillä on eri tason näkyvyyksiä muihin sopimuksiin verrattaessa. Funktion näkyvyysmäärityksiä ovat public, private, internal ja external. Funktioilla on myös erilaisia tiloja: pure, view ja payable. Jokaisen funktion loppuun tulee määritellä return-tyyppi. Funktiot voivat palauttaa useita eri arvoja.

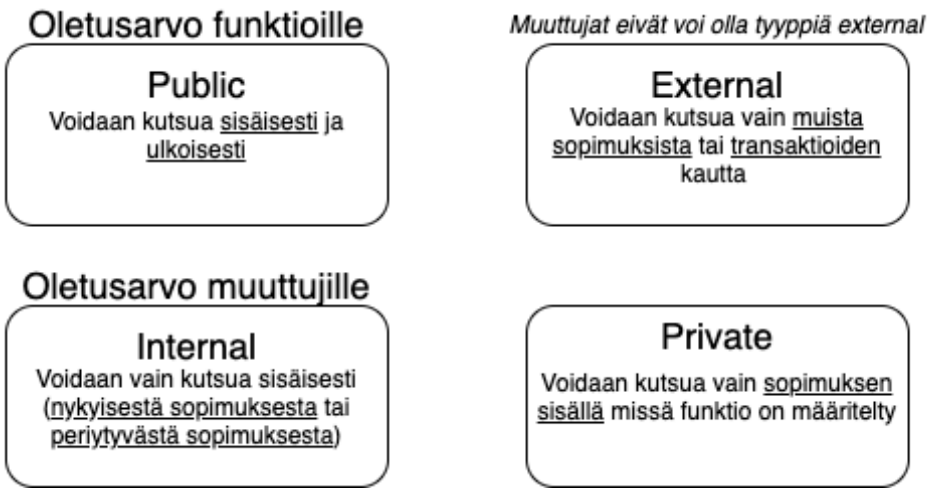
Mikäli funktion näkyvyyttä ei määritellä, sen oletusarvoksi asetetaan Public. Solidityä kirjoittaessa on hyvä muistaa funktioiden näkyvyyksien määrittäminen, sillä se auttaa hahmottamaan sopimuksen toimintaa silloin, kun sitä joudutaan arvioimaan.

Public-funktiota voidaan kutsua sisäisesti ja ulkoisesti, kun taas Private-funktioita voidaan kutsua vain niiden sopimusten sisällä, jossa ne on määritetty.

Internal on oletusnäkyvyys sopimuksen muuttujille. Internal-funktioita voidaan kutsua vain sisäisesti, joko sopimuksen sisällä, jossa se on määritelty, tai muussa sopimuksessa, josta se on periytynyt.

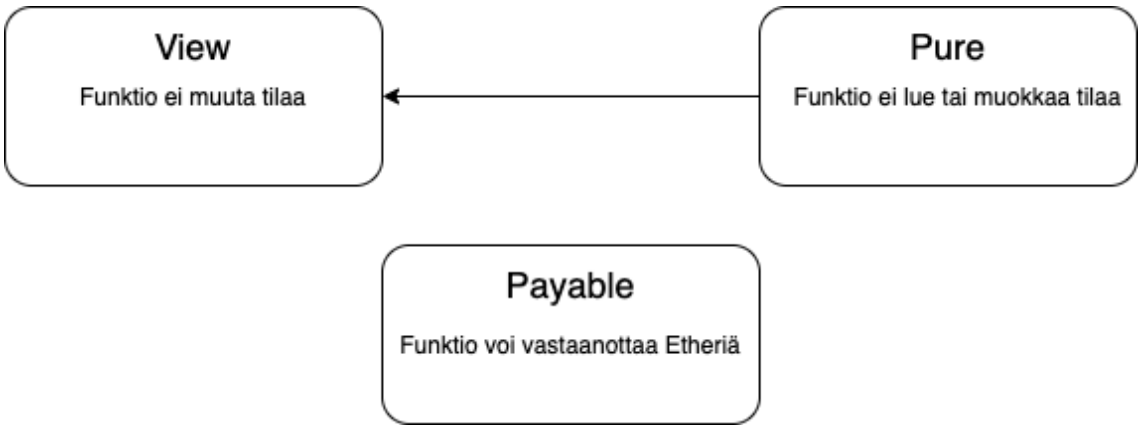
External-funktiot ovat osa sopimusrajapintaa, mikä tarkoittaa, että niitä voidaan kutsua muista sopimuksista ja transaktioiden kautta. External-funktiota f ei voida kutsua sisäisesti (esim. f() ei toimi, mutta this.f() toimii). External-funktiot ovat usein tehokkaampia isojen datamäärien käsittelyssä, koska dataa ei kopioida call datasta muistiin. Call data on EVM:n rekisteröimä tapahtuma, johon kirjataan funktioiden lähettämää dataa. Tämä myös tarkoittaa, että External-funktiot ovat kustannustehokkaampia.

Private-funktiot ja -muuttujat ovat näkyviä vain sen sopimuksen sisällä, jossa ne on määritelty, eikä niitä voi periä muissa sopimuksissa. Kuva 20 havainnollistaa erilaiset Solidity-ohjelmointikielen näkyvyysmääritteet.



Kuva 20. Solidity-funktioiden näkyvyysmääritteet.

Näkyvyystyyppien lisäksi funktioihin voidaan lisätä View-, Pure- tai Payable-muoto, jolla voidaan pidemmälle määritellä funktion toiminnallisuutta. View-tyypin funktiot eivät muuta tilaa, kun Pure-tyypin funktiot eivät luo eivätkä muokkaa tilaa. Kolmas tyyppi on Payable, joka tulee määrittää jokaiseen funktioon, johon halutaan lähettää etheriä. Ilman Payable-tyypin määritystä jokainen funktio hylkää automaattisesti siihen lähetetyt etherit. Kuva 21 havainnollistaa Solidity-ohjelmointikielen funktioiden tyypit ja niiden ominaisuudet.



Kuva 21. Solidity-funktioiden tyypit.

Funktion modifikaattorit

Modifikaattoreita käytetään funktioiden toiminnan muuttamiseen deklaratiivisella tavalla. Modifikaattoria voidaan esimerkiksi käyttää ehdon tarkastamiseen ennen funktion suorittamista. Modifikaattorit ovat periytyviä ominaisuuksia, ja niitä perivät sopimukset voivat ylikirjoittaa niiden toimintaa, mikäli ne on merkitty Virtual-tyypiksi.

Tapahtumat

Tapahtumat ovat periytyviä ominaisuuksia sopimuksissa. Niitä kutsuttaessa ne tallentavat argumentit tapahtumalokiin, tietynlaiseen tietorakenteeseen lohkoketjussa. Nämä lokit assosioidaan sopimuksen osoitteeseen ja liitetään lohkoketjuun, ja ne pysyvät siellä teoriassa ikuisesti, niin kauan kuin lohkoon on pääsy. Loki ja sen sisältämä tapahtumadata ei ole muiden sopimusten käytettävissä.

Ulkopuolisten lähdetiedostojen tuominen

Solidity tukee import-lausekkeita, mikä mahdollistaa muiden lähdetiedostojen ja kirjastojen käytön sopimuksissa. Globaalilla tasolla import-lauseketta voidaan käyttää muodossa

```
import “tiedostonimi”;
```

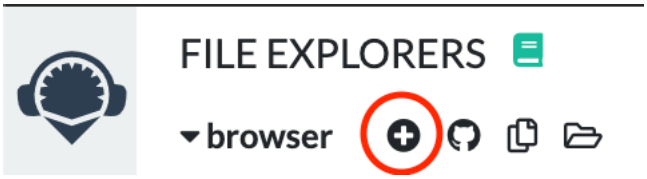
Remix tarjoaa automaattisen ohjauksen GithHubille ja hakee tiedoston automaattisesti verkon välityksellä. Tuotaessa tiedostoja GitHubin kautta tarvitsee laittaa haettavan tiedoston URL-osoite import-lausekkeeseen. Esimerkiksi Chainlink-väliohjelmiston tuomiseksi import-lauseke olisi muodossa

```
import “github.com/smartcontractkit/chainlink/solidity/contracts/Chainlinked.-
sol";
```

4.4 Mallisopimuksen toteutus

Paremman käsityksen saamiseksi älysopimusten teknisestä toteutuksesta insinöörityössä toteutettiin sopimusmalli, joka havainnollistaa yksinkertaisen älysopimuksen toteutuksen.

Kuten aiemmin mainittu, käytetään älysopimuksen toteutukseen Remix web IDE:tä, joka tukee Solidity-ohjelmointikieltä ja kääntää sitä vaivattomasti verkkoselaimessa. Remix ei vaadi erillistä alustamista, joten käyttäjä voi suoraan aloittaa älysopimuksen kehittämisen navigoimalla https://remix.ethereum.org/ -osoitteeseen. Käyttäjä voi luoda uuden tiedoston Remixiin File Explorers-välilehdestä löytyvästä plus-painikkeesta (kuva 22). Luotu tiedosto tulee File Explorers -välilehteen ja pääpaneelin aktiivisiin välilehtiin.



Kuva 22. Remix File Explorer-välilehden uuden tiedoston luonti.

Mallisopimuksesta kirjoitetaan yksinkertainen sopimus, joka pystyy lähettämään ja vastaanottamaan viestejä. Kuten aiemmin mainittu, jokainen Solidityllä kirjoitettu älysopimus tulee aloittaa alustamalla Solidityn versio, jota sopimus tukee. Tämä voidaan ilmoittaa staattisena eli yhtenä versiona, kuten

```
pragma solidity 0.4.25
```

tai dynaamisena kahden version välille, kuten

```
pragma solidity >=0.4.25 <0.6.0
```

jolloin vanhin versio, jota sopimus tukee, on 0.4.25 ja uusin versio, jota sopimus tukee, on 0.6.0.

Versioinnin jälkeen luodaan sopimus, jolle annetaan sen toimintoa kuvaava nimi, sillä isoissa projekteissa voi olla useita sopimuksia ja noudattaen yleisiä ohjelmoinnin standardeja funktioiden ja muuttujien nimien tulisi olla usein mahdollisimman kuvaavia niiden toiminnan kannalta. Mallisopimus nimetään HelloWorld nimellä (esimerkkikoodi 4).

```
contract HelloWorld {
```

Esimerkkikoodi 4. Mallisopimuksen luonti.

Sopimuksen sisälle kirjoitetaan kaikki sopimukseen liittyvä koodi. Sopimuksen alussa asetetaan muuttujat ja kutsutaan konstruktoria, joka asettaa tiettyjä arvoja muuttujille sopimuksen suorittaessa (esimerkkikoodi 5).

```
address public Vastaanottaja;
address public Vastaaja;
string public VastaanottoViesti;
string public VastausViesti;
```

Esimerkkikoodi 5. Mallisopimuksen muuttujien alustaminen.

Muuttujat alustetaan public-tyypiksi, sillä tässä kontekstissa niitä ei tarvitse tarkemmin määritellä. Muuttujien ja ominaisuuksien alustamisen jälkeen kutsutaan konstruktori-

funktiota, jolle annetaan parametriksi viesti, joka on tyyppiä string. Parametrille annetaan myös memory-argumentti, joka mahdollistaa string-parametrin tallentumisen sopimuksen muistiin. Konstruktorifunktion sisään asetetaan vastaanottajaksi msg.sender, joka tarkoittaa funktion kutsujaa. Asetetaan myös VastaanottoViesti-muuttujan konstruktori viestimuuttujaan ja tilamuuttuja Vastaanottajaksi (esimerkkikoodi 6).

```
constructor(string memory message) public {
      Vastaanottaja = msg.sender;
      VastaanottoViesti = message;
}
```

Esimerkkikoodi 6. Mallisopimuksen konstruktori.

Konstruktorin jälkeen luodaan sopimuksen funktiot, jotka ovat vastuussa sopimuksen toiminnallisuudesta. Mallin toiminnallisuuteen tarvitaan kaksi eri lähetysfunktiota, vastaanottajalle ja vastaajalle. Funktioille annetaan parametrina niitä vastaavat viestityypit, vastaanottoViesti ja vastausViesti. Funktiolle määritellään parametriksi string- sekä memory-argumentti, jotta funktioon syötetty vastaanottoViesti-muuttuja tallentuu oikeassa muodossa (esimerkkikoodi 7).

```
function LahetaViesti(string memory vastaanottoViesti) public {
      if (Vastaanottaja != msg.sender) {
            revert();
      }

      VastaanottoViesti = vastaanottoViesti;
}
```

Esimerkkikoodi 7. Mallisopimuksen LahetaViesti-funktio.

Funktio LahetaViesti suorittaa if-lausekkeella tarkistuksen, jossa varmistetaan Lahettaja-muuttujan olevan sama kuin msg.sender eli funktion suorittaja. Msg.sender kuuluu Solidityn globaaleihin muuttujiin, jotka ovat yleiskäyttöisiä aputoimintoja. Tämä suoritetaan, koska viestin lähettäjän täytyy olla sama kuin lähettäjä, kun taas LahetaVastaus -funktiossa kuka tahansa voi olla vastaajana (esimerkkikoodi 8).

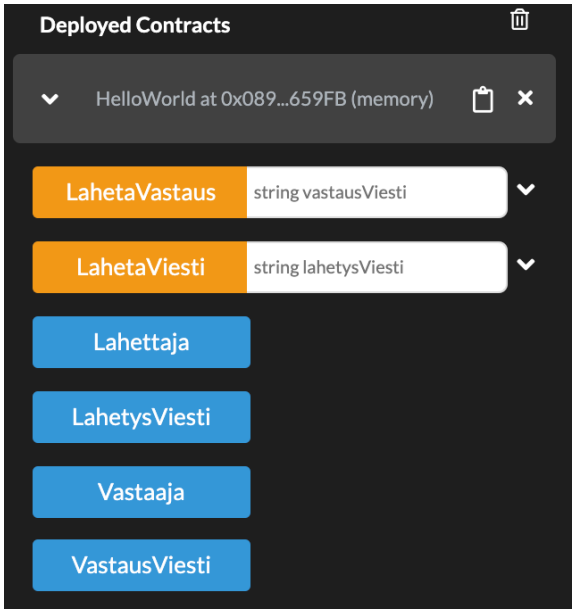
```
function LahetaVastaus(string memory vastausViesti) public {
      Vastaaja = msg.sender;

      VastausViesti = vastausViesti;
}
```

Esimerkkikoodi 8. Mallisopimuksen LahetaVastaus-funktio.

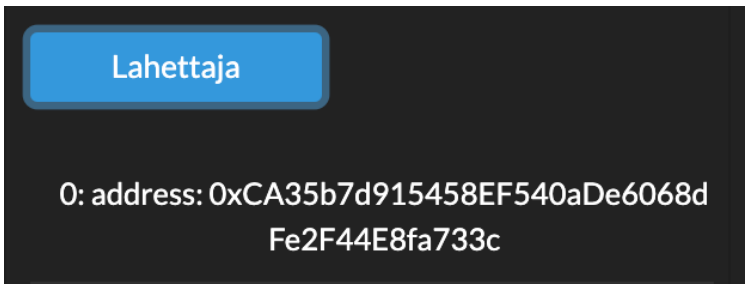
4.5 Tulokset

Sopimuksen toiminnan testaamista varten, se tulee ottaa käyttöön Remixin Solidity -kääntäjän puolelta valitsemalla oikean kääntäjän versio ja painamalla Compile test.sol. Tämän jälkeen käyttöön otettu sopimus ilmestyy Deploy & Run Transactions -välilehteen, josta se voidaan ottaa käyttöön valitsemalla käännetyn sopimuksen valikosta ja painamalla Deploy. Käyttöön otettu sopimus ilmestyy Deployed Contracts -osion alle, josta mallitapauksessa ilmestyy kuvan 23 kaltainen näkymä.



Kuva 23. Käyttöön otettu sopimus.

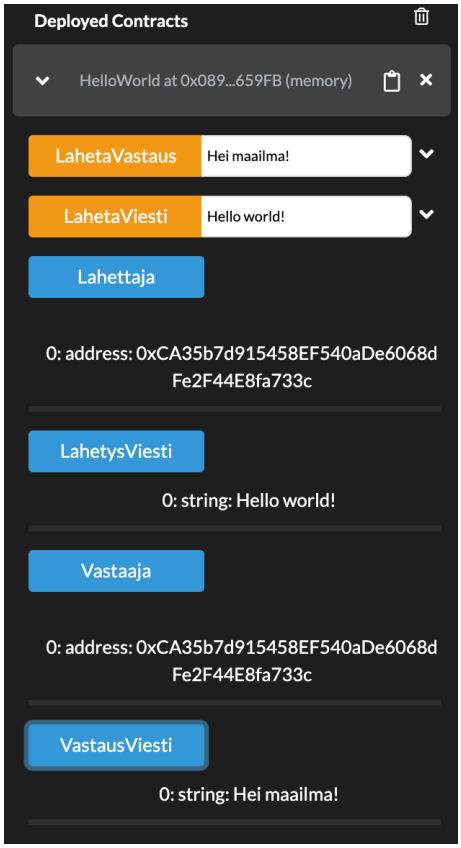
Näkymä toimii myös käyttöliittymänä sopimuksen toiminnoille. LahetaVastaus ja LahetaViesti ovat tekstikenttiä, joihin käyttäjä voi funktioiden toiminnallisuutta testatessa syöttää merkkijonoja parametreiksi. Tätä voidaan kokeilla syöttämällä tekstikenttiin merkkijonoja ja sen jälkeen painamalla LahetaViesti- tai LahetaVastaus-painikkeita. Käyttöliittymässä näkyvät myös sopimuksessa alustetut muuttujat ja niitä painamalla niiden senhetkiset arvot. Mikäli mitään funktiota ei ole kutsuttu, kaikkien paitsi Lahettajan arvot ovat 0. Lahettajan arvo alustetaan oletuksena msg.sender-arvolla sopimuksen konstruktorifunktiossa, joten sen osoitearvoksi periytyy testiympäristön tilin osoite kuten kuvassa 24 näkyy.



Kuva 24. Lahettaja-muuttujan arvo sopimuksessa.

Julkisessa ympäristössä sopimusten funktioita voi kutsua mikä tahansa taho, jolla on pääsy julkiseen sopimukseen, joten on tärkeää määritellä tarkasti funktiokutsujen ehdot ja se, kuka niitä voi kutsua.

Kun sopimus on käännetty, sen toimintaa voidaan testata syöttämällä merkkijonoja funktioiden parametreihin ja kutsumalla niitä. Tämän jälkeen käyttöliittymästä voidaan painaa LahetysViesti- ja VastausViesti-painikkeita, jotta nähdään niiden uudet arvot. Esimerkissä syötettiin merkkijonot “Hei maailma!” ja “Hello world!”, minkä jälkeen viestien arvot vaihtuivat annettuihin parametreihin. Sopimuksen Lahettaja ja Vastaaja osoitteet ovat samat, sillä tässä tapauksessa kutsut molempiin funktioihin tapahtuivat samasta testiosoitteesta, kuten kuvassa 25.



Kuva 25. Käyttöön otetun sopimuksen testaus.

Merkkijonot tallentuvat sopimukseen onnistuneesti, ja ne voidaan varmentaa LahetysViesti- ja VastausViesti-osioista, jossa ne näkyvät string-muodossa. Julkisessa julkaistussa sopimuksessa kuka tahansa voisi kutsua sopimuksen LahetaVastaus-funktiota, jolloin osoite voisi olla eri kuin lähettäjän.

Funktiota kutsuessa Remixin konsoliin ilmestyy call-tapahtuma (kuva 26), josta näkyy funktion kutsun tarkempia tietoja, kuten transaktion tiiviste, kutsujan osoite, kutsuttavan osoite, transaktiomaksu, suoritusmaksu, kutsun tiiviste ja mahdollisia parametreja sekä lokeja.

Kuva 26. Remix-konsolin tapahtuma funktiokutsusta.

Kuvassa näkyy mallisopimuksen LahetaVastaus-funktion call-tapahtuma, jolle annettiin parametrina merkkijono “Hei maailma!”. Parametri näkyy decoded output -osiossa. Funktion transaktion hinta on 24 546 gas, ja sen suoritusmaksu on 3 274 gas-yksikköä.

Transaktio ja suoritusmaksut vaihtelevat sopimuksissa sen perusteella, kuinka vaativia funktioita joudutaan suorittamaan. Tämän takia on hyvä pyrkiä pitämään sopimusten funktiot mahdollisimman yksinkertaisina ja kustannustehokkaina, kuten käyttämällä External-näkyvyystyyppiä, joka vaatii vähemmän gasia suorittaakseen, sillä External -tyypin funktiot käsittelevät laajoja määriä dataa tehokkaammin kuin muut tyypit.

Deployed Contracts -osion yllä Remix näyttää myös kaikki transaktiot, jotka ovat tapahtuneet sopimuksessa. Käyttäjä voi tarkastella transaktioita painamalla tallennuskuvaketta ja luomalla uuden scenario.json-tiedoston, josta transaktiolokia pääsee tarkastelemaan. Scenario.json sisältää tarkempaa dataa kuin Remix-konsolin tapahtumat ja tallentaa ne pysyvään tiedostoon, josta niitä voi tarkastella.

5. Yhteenveto

Insinöörityön tavoite oli antaa lukijalle kattava käsitys lohkoketjuteknologiasta, älysopimuksista ja älysopimusten kehittämisen perusteista. Työssä toteutettiin tutkimus älysopimusteknologiasta uutena teknologiana ja käytiin läpi älysopimuksen kehitys vaihe vaiheelta. Koodin toteutus ja sen testaus onnistuivat, ja tuloksena saatiin toimiva mallisopimus, johon osapuolet voivat lähettää viestejä puolin ja toisin. Työn tutkiva osa tarjoaa vastauksia siihen, miten älysopimusteknologiaa voidaan hyödyntää ja kuinka älysopimukset tarjoavat ratkaisuja moniin sopimusluontoisiin ongelmiin. Työn käytännön puolessa on kehitettävää, ja mallisopimusta voisi kehittää pidemmälle esimerkiksi laajentamalla sen toiminnallisuutta.

Työssä tehtyjä havaintoja lohkoketjuteknologian perusteista, älysopimusten perusteista ja niiden kehityksestä voidaan hyödyntää oppimateriaalina lohkoketjuteknologian tai älysopimusten kehityksessä. Työn tutkiva osa perustuu kuitenkin jatkuvan kehityksen alla oleviin teknologioihin, joten työssä tehdyt havainnot päivittyvät ajan myötä kehityksen takia. Älysopimukset ovat vielä toistaiseksi uusi teknologia, joten niiden tutkiminen ja oppiminen on haastavaa rajoitetun materiaalin takia.

Nyt älysopimukset ovat vielä siinä vaiheessa, että ihmiset yliarvioivat uuden teknologian lyhyellä aikavälillä, mutta aliarvioivat sen vaikutuksen pitkällä tähtäimellä. Vaikka älysopimuksia on kehitettävä ennen niiden laajaa omaksumista monimuotoisissa kaupallisissa suhteissa, niiden on mahdollista mullistaa palkkio- ja kannustinrakenteita, jotka muodostavat osapuolten väliset sopimukset tulevaisuudessa. Tämän takia ei ole oleellista ajatella, kuinka nykyiset konseptit ja rakenteet voidaan siirtää tälle uudelle teknologialle. Pikemminkin älysopimusten todellinen vallankumous tulee todennäköisesti täysin uusista hyödyistä, joita ei ole osattu vielä kuvitella.

Lähteet

1. Szabo, Nick. 1996. Smart Contracts: Building Blocks for Digital Markets. Verkkoaineisto. <http://pdfs.semanticscholar.org/9b6c/d3fe0bf5455d-d44ea31422d015b003b5568f.pdf?_ga=2.97799128.612124525.1578685599-2080643238.1578685599> Luettu 13.1.2020.

2. Nakamoto, Satoshi. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. Verkkoaineisto. <https://bitcoin.org/bitcoin.pdf> Luettu 16.1.2020.

3. Adamchik, S. Victor, CMU. 2009. Concept of Hashing. Verkkoaineisto. <https://www.cs.cmu.edu/~adamchik/15-121/lectures/Hashing/hashing.html> Luettu 20.1.2020.

4. Sahu, Aradhana & Ghosh, Samarendra Mohan. 2017. Review Paper on Secure Hash Algorithm With Its Variants. Verkkoaineisto. <https://www.researchgate.net/publication/326009898_Review_Paper_on_Secure_Hash_Algorithm_With_Its_-Variants> Luettu 1.3.2020.

5. Pagliery, Jose. 2014. Bitcoin and the Future of Money. E-kirja. Triumph books.

6. Understanding digital signatures. Verkkoaineisto. Docusign. <https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq> Luettu 27.1.2020.

7. Aydar, Mehmet; Cetin Cemil, Salih; Ayvaz, Serkan & Betul, Aygun. 2019. Private key encryption and recovery in blockhain. Verkkoaineisto. <https://www.researchgate.net/publication/334361184_Private_key_encryption_and_recovery_in_blockchain> Luettu 1.3.2020.

8. Hyytiäinen, Teemu. 2018. Lohkoketju Laboratorio: Merkle-puu. Verkkoaineisto. <https://medium.com/lohkoketju/merkle-puu-5d039617381f> Luettu 26.1.2020.

9. Nguyen, Anh. 2019. Smart Contract - A Promising Use Case For The Adoption Of Blockchain. Verkkoainesto. <https://www.theseus.fi/bitstream/handle/10024/263968/Thesis.pdf?sequence=2&isAllowed=y> Luettu 26.1.2020.

10. What is Proof of Work?. 2019. Verkkoaineisto. Ledger. <https://www.ledger.com/academy/blockchain/what-is-proof-of-work> Luettu 26.1.2020.

11. What is a 51% attack? 2019. Verkkoaineisto. Binance Academy. <https://www.binance.vision/security/what-is-a-51-percent-attack> Luettu 26.1.2020.

12. Proof of Stake explained. 2019. Binance Academy. Verkkoaineisto. <https://www.binance.vision/blockchain/proof-of-stake-explained> Luettu 26.1.2020.

13. Martinez, Julian. 2018. Understanding Proof of Stake: The Nothing at Stake Theory Verkkoaineisto. <https://medium.com/coinmonks/understanding-proof-of-stake-the-nothing-at-stake-theory-1f0d71bc027> Luettu 27.1.2020.

14. Grincalaitis, Merunas. 2018. Can a Smart Contract be upgraded/modified? Is CPU mining even worth the Ether? The Top questions answered here… Verkkoaineisto. <https://medium.com/ethereum-developers/can-a-smart-contract-be-upgraded-modified-1393e9b507a> Luettu 27.1.2020.

15. Marino, Bill & Juels, Ari. 2016. Setting Standards for Altering and Undoing Smart Contracts. Cornell Tech (Jacobs Institute), New York, United States.

16. Nzuva, Silas. 2019. Smart Contracts Implementation, Applications, Benefits and Limitations. School of Computing and Information Technology, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya.

17. Juels, Ari; Nazarov, Sergey; Breidenbach, Lorenz; Coventry, Alex; Ellis, Steve & Magauran, Brendan. 2019. Mixicles: Simple Private Decentralized Finance. Chainlink.

18. Hamacher, Adriana. 2019. Chainlink CEO: How “Mixicles” can change the game for smart contract privacy. Verkkoaineisto. <https://decrypt.co/9037/chainlink-ceo-sergey-nazarov-mixicles-smart-contract-defi-privacy?amp=1> Luettu 7.2.2020.

19. Cummins, Tim. 2017. The cost of a contract. Verkkoaineisto. <https://blog.iac-cm.com/commitment-matters-tim-cummins-blog/the-cost-of-a-contract> Luettu 9.2.2020.

20. Hayes, Adam. 2020. Is Ethereum More Important Than Bitcoin? Verkkoaineisto .<https://www.investopedia.com/articles/investing/032216/ethereum-more-important-bitcoin.asp> Luettu 19.1.2020.

21. What is the “Unstoppable world computer?”. 2019. Verkkoaineisto. Bitrates. <https://www.bitrates.com/guides/ethereum/what-is-the-unstoppable-world-computer> Luettu 15.2.2020.

22. Orcutt, Mike. 2019. Blockchain smart contracts are finally good for something in the real world. Verkkoaineisto. <https://www.technologyreview.com/s/612443/blockchain-smart-contracts-can-finally-have-a-real-world-impact/> Luettu 20.2.2020.

23. Levi, Stuart & Lipton, Alex. 2018. An Introduction to Smart Contracts and Their Potential and Inherent Limitations. Verkkoaineisto. <https://corpgov.law.harvard.edu/2018/05/26/an-introduction-to-smart-contracts-and-their-potential-and-inherent-limitations/> Luettu 20.2.2020.

24. Mikä on Ethereum? 2019. Verkkoaineisto. Bitcoinkeskus. <https://bitcoinkeskus.com/ethereum-opas/> Luettu 27.2.2020.

25. Juels, Ari. 2016. Town Crier: An Authenticated Data Feed for Smart Contracts. Verkkoaineisto. <https://www.researchgate.net/publication/308691770_Town_Crier_An_Authenticated_Data_Feed_for_Smart_Contracts> Luettu 18.1.2020.

26. Public Key Cryptography. Verkkomateriaali. Wikipedia. <https://en.wikipedia.org/wiki/Public-key_cryptography#/media/File:Public_key_encryption.svg>.

27. Mehmet, Aydar; Salil, Cetin; Serkaz, Ayvan & Aygun, Betul. 2019. Private key encryption and recovery in blockchain. Verkkoaineisto. <https://www.researchgate.net/publication/334361184_Private_key_encryption_and_recovery_in_blockchain> Luettu 1.4.2020.

28. Merkle Tree. Verkkomateriaali. Wikipedia. <https://en.wikipedia.org/wiki/Merkle_tree#/media/File:Hash_Tree.svg>.

29. Learn me a bitcoin. Walker, Greg. Verkkomateriaali. < https://learnmeabitcoin.com/explorer/block/00000000000000000ffa5eaa305507e43cde47b2939daecf09f158a148c185>.

30. Bitcoin block reward halving countdown. Verkkomateriaali. <https://www.bitcoinblockhalf.com/>. Luettu 1.4.2020.

31. Blockchain Hash Rate. Verkkomateriaali. Blockchain.com. <https://www.blockchain.com/charts/hash-rate>. Luettu 1.4.2020.

32. Perez, Yessi Bello. Understanding Bitcoin's smallest unit, the Satoshi. Verkkomateriaali. <https://thenextweb.com/hardforkbasics/2019/10/16/understanding-bitcoins-smallest-unit-the-satoshi/>. Luettu 1.4.2020.

33. What is an ERC20 token?. 2019. Verkkomateriaali. Coinsutra. <https://coinsutra.com/what-is-erc20-token/>. Luettu 1.4.2020.

34. Regulation of Cryptocurrency around the world. 2018. Verkkomateriaali. The Law Library of Congress. <https://www.loc.gov/law/help/cryptocurrency/world-survey.php>.

35. Features. 2020. Verkkomateriaali. Chainlink. <https://chain.link/features>. Luettu 1.4.2020.

36. ETH / USD aggregation. 2020. Verkkomateriaali. Chainlink. <https://feeds.chain.link/eth-usd>. Luettu 1.4.2020.

37. Piant, Rory. Community Manager, Chainlink, Grand Cayman. Haastattelu 11.2.2020.

38. Documentation. 2020. Verkkomateriaali. Remix, Ethereum-IDE. <https://remix-ide.readthedocs.io/en/latest/>.

Haastattelu

Interview with Rory Piant from Chainlink

11.2.2020

P = Petteri Piironen

R = Rory Piant

P: So just to give a brief introduction as to who you are, your role in the company and what you do.

R: Yeah, so my name is Rory Piant i'm the community manager for Chainlink. What that really means is my job is kind of globally sort of helping, manage both kind of community voice, community engagements, developing and helping people evangelize for Chainlink inside the community. So not really unlike the business development side per say, but really the day to day, giving the community the tools, the educational resources to go out and talk about chainlink and do meetups, events things like that. So that's what my main job is like.

P: Okay. So for anyone who has heard of these upcoming technologies like blockchain and smart contracts who doesn't really know about them, how would you go about describing smart contracts to them?

R: That's a great question. Basically the high level description is that it's a deterministic digital agreement. So it's a type of contract between two parties or related to a set of functions. So when something happens, something is guaranteed to happen. It will automatically happen.

Taking something like Ethereum the idea is that in a perfect world, Ethereum will be around essentially around forever, or could be around forever. So its a digital agreement, that will always execute based on the input or functions that we set, forever. So a human being doesn't have to be there watching the price of a stock, or physically ensuring that things are happening. Once this contract is created, they know that it's going to execute as soon as specific functions or parameters are met.

So how I would describe it is like a real estate deal. Say I wanna purchase land or a home, I can use a smart contract, where I could as a buyer deposit a stable coin, ethereum or something like that into a smart contract. That person (seller) could then basically be like “Yep, Rory has the funds and wants to purchase this” and as soon as those funds are available, they can be transferred into that person (sellers) account. And in a perfect world, this is probably kinda like the next step in technology, is that when something like a title could be put into a blockchain and could be instantly transferred to me. So I wouldn’t need like a title company or any middlemen that sit there and make sure that Rory does officially own the property and Rory does officially have the money. All of that could be taken care of in this digital agreement, meaning I could do a deal with someone in Saudi Arabia, someone in Africa, someone in the UK without technically even having to know who they are. They could simply say “Heres this land, I can prove I own it” and I could say “That's great, i’m willing to pay this price for it.” and we create a smart contract for it. And it could be seamlessly done in a single block, which is relatively only 12 seconds. So I could essentially buy a property anywhere around the world in 12 seconds, and have digital proof that I own the property.

P: Okay, so was that the more technical explanation of it?

R: Yeah well that's what I would do in a conference to explain what a smart contract is, and the technical explanation would be a deterministic digital agreement where basically I know that it's going to execute no matter what as long as the parameters are met.

P: Okay, so based on that what is the purpose of smart contracts? What are they for generally?

R: Yeah we always say “creating a trustless environment” which is kind of a funky word. But basically I don’t have to trust another individual that they are going to follow through on an agreement. If those input parameters are met, once that contract is created, I know, they know, everyone involved in the smart contract knows it will be executed. So an example would be that Amazon stock hits 800 usd and I wanted to buy some. We could basically create a digital agreement right away, where I would put the funds, right now they need to be in cryptocurrency, I could put the funds in the smart contract and basically it would, let's say as often as 10 seconds, or could do it at the open or close of the stock market. But basically it would check that price using an oracle, and it would basically say “Okay, the price is 800, and Rory now wants to buy and wants the contract to be executed”. I wouldn’t have to be awake or have a broker, it would automatically do that for me.

P: Right. So this is in the field of finance mostly, but could this be extended to other industries?

R: Yeah, insurance is another big one that people like to talk about. Flight insurance for example, we just recently worked with Etherisk who has a team that does flight insurance. So basically if your flight is delayed by 3 hours, right? You created a smart contract, flights delayed by 3 hours, an oracle confirms that the flight is delayed by 3 hours and a certain amount of money could be put into your account to rebook a ticket, pay for hotel or lodging because you can't make that travel, maybe you're stuck somewhere. So that would be one way of humans never having to be involved in that. It's just automatically going to happen when that flight is delayed by a certain amount of time, that's one way. International shipping and geofencing is very interesting as well. So basically a product is shipped around the world, payments could be made around the world based on where the shipping containers are. Right, so they've left the port, 25% gets paid to them so they get to pay their employees. The people onboard the ship, it's there halfway, another 25% gets paid. it goes through customs, they digitally sign that and verify that its passed through the customs and then another payment could be made. And this goes all the way to the delivery of the goods. So another company steps up, they say "we'll deliver the goods to Rory" and basically as soon as they deliver those goods, I can electronically sign that the goods have been received, they sign that they have been delivered and the payment for the delivery can be made.

So it's a really cool idea that you could remove all the middle human beings involved in that process and basically I could create a direct relationship with the shipper. If I was a business we could basically use cool things like GPS data so I would always know where the shipment was and they would always know they were going to get paid as long as they met the agreed upon parameters at the time that we set.

P: Okay. So how secure is all this? How can we trust the information that is being input to these contracts?

R: Yeah that's a great question. I would say it's as secure as both the smart contract itself, right, so writing the smart contract right now it does typically take someone that's familiar with it. So that usually wants to be audited in a perfect world. So you'd most likely want to have an audited smart contract for something that involves quite a bit of money. So step one is making sure the smart contract is coded correctly, that has to be done correctly. The next thing is the blockchain it occurs on. Right now Ethereum is the number one smart contract platform. But basically you would have to know that Ethereum will keep functioning. So with Ethereum there's thousands of nodes decen-

trally operated all over the world, but if for whatever reason everybody turned that node off or something occurred, or the network stopped, then obviously my smart contract couldn't execute. So that's another factor that comes into play. And then the last factor is if i'm using an oracle or a middleware to fetch outside data so I need to deliver that GPS data to my smart contract. I need a piece of software that does that. So basically I need to make sure that that piece of software or middleware is secure and reliable and also in a perfect world decentralized so that I can make sure that there's a lot of money at stake. So that a simple API, a web API going down doesn't stop my smart contract from working, because if i'm using an external input as a function inside that smart contract. If that fails, the API simply goes down, or that single oracle goes down, then my smart contract is worthless. So there's a couple of steps that have to occur properly for everyone to likely agree that a smart contract is suitable for that use case.

P: Okay. So there's some sort of governance in that it needs to be agreed upon that it's reliable.

R: Yes exactly. You need to agree that the smart contract is reliable, you need to agree that the blockchain is executing the smart contract is decentralized enough that will likely not fail, right? And then if using an outside input like an oracle to deliver data you need to deliver that that's also secure, that that's also decentralized enough so that data will get delivered when you need it.

P: Okay. So how would the transition from traditional contracts to smart contracts work, for a company for example?

R: Yeah absolutely. I think there's a lot of things there that were seeing a lot of work happening, but there's still work that needs to be done. So the first thing is I think you have to say is how can they save money? Flat out. It's not interesting to them unless they can likely save money or time, right? And smart contracts have the ability to do both of those things for them, typically removing the physical individuals who have to monitor such things that have to execute bank transfers and payments. This can now or very quickly be done on the blockchain by using smart contracts. So what they have to do is they have to evaluate their personal risk versus the reward. So at some point we would like to think that as the technology gets developed, the risk of using it will go down, in other words methods of creating smart contracts get developed and accepted. We're already kind of there now, but people accepting that this is a good smart contract and we're going to copy this and we're all comfortable with using this copy because it's been used a million times before. So that kind of eliminates that sort of programmatic risk as we like to call it. And then basically they have to be able to show that with a little

bit of effort, long term you could save a lot of money, time and resources. So right now we're making the case for that and I think you see a lot of institutions dabbling with it doing proof of concepts and pilot programs. I'd like to say all of them come back to say it's good, but if you're a publicly traded company, for instance, on an exchange then you have to be able to go back to your stockholders and say "I'm doing something where the risk is low and the reward is high". I think most of us can agree that's a good idea. Right now I think some people see it to be medium or high risk, depending upon the amount of money involved. In transition to this we have find a way to reduce that risk both in terms of programmatically like better coding and in terms of enough people developing enough use cases and using it to where the risk of not using it is higher than the risk of using it, you know. So you need those couple of first banks using and creating automated payments and things like that. Once those first couple start it makes it easy for the large legacy businesses to say "okay, well they're creating smart contracts worth a billion dollars, it might be okay for me to put mine on that's worth 500 million". So that's the kind of stepping stones that have to occur. Right now we are in the kind of like the pilot, proof of concept, some teams are dabbling with that. What we're really looking for is the next step where someone is really taking a financial product worth a lot of money and creating a smart contract that has a lot of value in there.

P: Okay. So you mentioned the development of smart contracts and it takes a long time right now. So what is the development process for a smart contract like?

R: Right, so the first step is to find someone who has the skills and the ability to develop that. So number one they have to create the smart contract to know what they are doing. The second thing is that it needs to be audited. So even though someone has developed it, you most likely want a third party to take a look at the contract, similar to how a lawyer functions today, we sign an agreement. I would like to have an educated third party to take a look at what we want the contract to do, and does that actually match up with what the contract actually does? So using a third party to say "okay, every time this GPS coordinate is hit, the smart contract is going to make a payment". We likely want a third party who is relatively trusted who's experienced to be able to look at the smart contract to say "Yep, that's exactly how it was designed, this will execute based on the way it was written". So those are the two challenges that we deal with now; you've got to create it and for most big use cases of smart contracts you want it to be audited by a 3rd party.

P: Sure. So you mentioned the term middleware, and as far as I know Chainlink is some sort of a middleware, right?

R: Yes.

P: So could you explain what Chainlink is and what role does it play in this smart contract ecosystem?

R: Yeah sure. So chainlink is referred to as a middleware or its a decentralized oracle network. So it's not a blockchain. It's an infrastructure tool. And what this software allows you to do, is to connect your smart contract like something on Ethereum, to external data. Smart contracts can not natively retrieve external data, and that's a good thing. The reason that they can't is because you're basically creating a security vulnerability, right? If you can allow an external source to do something inside your smart contract, that can be scary. Unless you're using a piece of software or an oracle solution that's created in a way that gives you the levels of security and decentralization that you need to basically be able to safely assume that the data you need will be delivered when you need it. So how that works with Chainlink is we have a bunch of node operators, 28 right now, and basically what you can do is you can use them to deliver you any sort of external data. So if you want the price of a stock, you can do that. The big thing right now is decentralized finance, where people are basically using these protocols to track the price of an asset at any given time, so this is a great use case for oracles because you need a trusted source that delivers that data. Because if you use like a single centralized oracle or the company that created the financial product used an oracle, if they lose control of that single oracle or it fails, then this whole group of financial products that's basically built right now on blockchain will fail to execute. So let's say you need to know the price of Bitcoin. Well if you looked right and I asked you right what the price of Bitcoin is, it would likely depend upon on the geographical location that you're in, or would likely depend upon a certain exchange. What we say is that's not the best way to look at it. If we want to execute a trillion dollar contract on the price of Bitcoin, understanding that a small percentage difference could impact that contract in a meaningful way, someones gonna lose ultimately. So what we say is, well lets use APIs that basically aggregate data from multiple exchanges all over the world, lets use multiple node operators or basically pieces of software that retrieve this data, all pushing it into a single smart contract and typically aggregating that. So if I want to know the price of Bitcoin, I can use these oracles and these data providers to basically get a wide view of the global price of bitcoin all over the world. They can take an average or whatever I wanna use, average being the most common. And basically say right now the average price of bitcoin around the world is this, therefore we agreed to do

some kind of transaction on the price of bitcoin. We can both look at the data itself and be able to say when we execute this contract, the price of bitcoin was x, because all these sources agreed that the price of bitcoin was x. And so that's really the importance of oracles, you want it to be as decentralized as the actual network that you're on. So Ethereum has thousands of nodes, basically agreeing using consensus, so our goal is to potentially create thousands of oracles or blockchain middleware in this sort of network that you can reach out to and say I need this data and I know its just as decentralized as the blockchain that you're operating on.

P: Okay, so there's some sort of consensus mechanism that approves the average of the data and that is then trusted.

R: That's exactly right. So the good news is that if a single node or oracle goes down, your data is still delivered, because you're using 28, you're using 15 or however many you're using. You know that if that single oracle goes down, you're okay, your smart contract is still going to function. If your data source or your API provider goes down, which happens sometimes, right? They have some sort of failure, well you know that you're pulling from all those sources so even if one goes down, you're likely going to get a price that's close to the truth because you're pulling from all these different data feeds and oracles. So the idea is if I've a trillion dollar contract, everythings done perfectly, but i'm not using a quality oracle, that whole contract would essentially be made worthless or worst case scenario executed on the wrong data because I relied on a single source or a single oracle for that information.

P: Okay, so while this is all very complex for the common person, how do you think the adoption process will go? How long will it take for this to become an industry standard?

R: Right. So now we're already kind of starting to see that, especially around decentralized finance. So people are building a lot of new interesting technology that allows anyone around the world to access complex financial instruments, basically be able to long or short an asset.Basically be able to create a basket of tokens so in other words "I don't know anything about these projects, I just wanna buy a token that represents the top ten, similar to like an index on the stock market".

P: Like an ETF?

R: That's exactly right. So I can basically right now create my own ETF on the blockchain and create a single token that maybe represents multiple projects or multiple tokens on the blockchain so that I can vary my risk or exposure to certain assets.

What's great about is that I don't have to be around, I don't have to monitor it, I don't have to watch it, it will either raise or lower based on the performance of those assets, so we're seeing that now kind of explode. So basically what we're doing is creating oracles that are specifically designed for them, creating multiple price feeds, using multiple oracles. A smart creator can basically say "I want to create a new financial instrument, I need external data, I know that Chainlink has an oracle solution that provides this trusted source for me." In ten minutes I can connect my smart contract to all these oracles and nodes who can aggregate this data for me, and I can tell my users that I'm using a decentralized source so that you can trust that whatever data is delivered, is likely to be the truth. Because when you're dealing a lot of money, a certain error inside that data could end up costing someone a lot of money.

P: Right. You mentioned some limitations, but what do you think right now is the most important thing that needs to be tackled in terms of smart contracts?

R: I think its connecting legacy business or industries to this new technology. So you know I remember reading an article about something like when electricity was invented we were all using steam. It took roughly 40 years for most factories to transition from steam to electricity. Electricity was better, nobody disputed it. But it took factories and companies a long time because they had spent a lot of time, money and resources on steam technology, so it was very costly to go to the new transition. So what we have to do is to create a structure, and this is in blockchain in general, where A it makes it very easy for these legacy industries to see the value, to see the use cases and to basically drop the risk for them, the programmatic risk as well as the risks for oracles and stuff. We basically have to develop them to high enough level that they can take a look at it and say this is a risk that i'm willing to accept. There are some businesses that are more risk averse than others. Banks typically are extremely risk averse, so although this technology may be great and it may be useful and it may save them money and time, the reality is that there's some risk involved. So they're likely wanting to see other industries starting to play, experiment, develop pilots and POCs. So really the real case right now is we need to continue to create these use cases, create these POCs with companies who then actually start to transition to using what we call the mainnet or actually putting it live on the actual blockchain. The more we see that, the less riskier it will be for people in that industry, then other industries will step in. So you really have to identify whats the type of industries right now, where they're less risk averse and they'd be more interested to step in, obviously projects on the blockchain and businesses on blockchain are less risk averse because they're already utilizing the technology. So that's kind of the big explosion that we are seeing now, is that a lot of companies are developing products specifically on the blockchain so of course they're open to using

oracles and things like that. So as their success grows, I think it will be a matter of time.. it's hard for me to give you a date, but I would likely say within the next few years we'll start to see major legacy enterprise businesses step in and say "okay, this is interesting. I know this will save me money now, so I will use this one thing, one piece of my business on smart contracts". And when they see the value of that, they'll add another piece and another piece and another piece... but absolutely it will take time it won't happen overnight. But we're starting to see major enterprises at the very least, like JP Morgan, developing a blockchain, stuff like that. So we are seeing some of these things occurring, IBM, Google, all of these large companies now have a blockchain division and typically a head of blockchain. And so their job is to communicate back to their board and say "Hey, I can save us money right now, and the risk is low enough for us to be actually interested". So we're at that stage right now, where everyone sees there's value, but we have to be able to take that "I think that there's value" to be able to show that there is real value, and the risk has been reduced enough.