SANTERI TANI

TIEA1000: LOHKOKETJUTEKNOLOGIAT JA SOVELLUTUKSET

Sitaattilain turvaamana hyödynnetty materiaaleja seuraavilta tahoilta: JYU (Santeri Tani, Henrik Korhonen)

ÄLYSOPIMUSALUSTAN TOTEUTTAMINEN

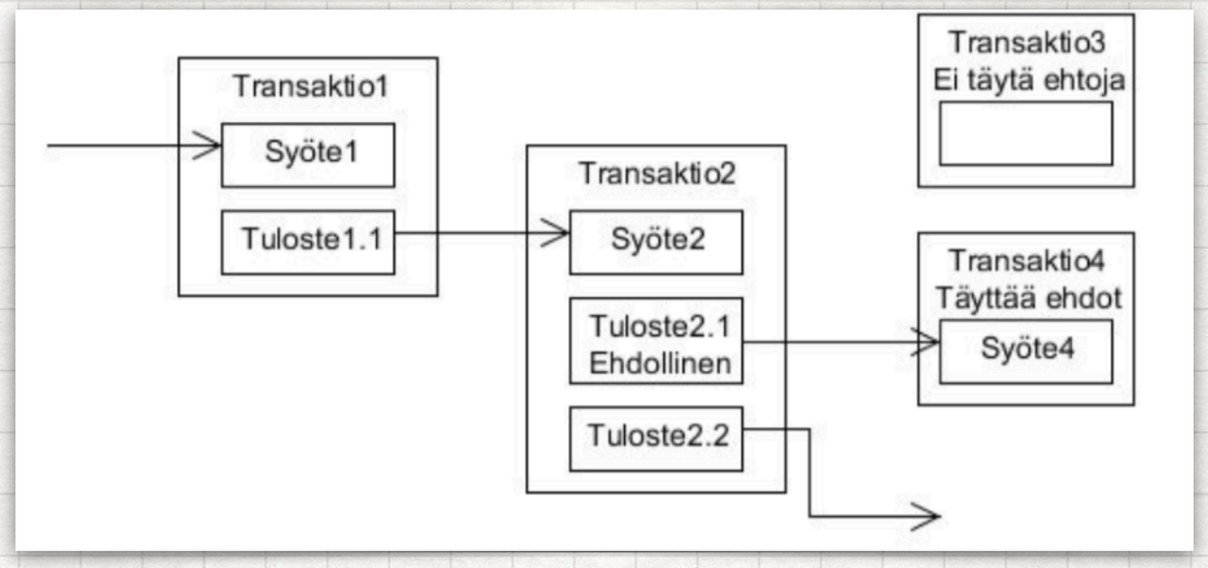
LUENTO 2

- Eri lohkoketjut mahdollistavat älysopimusalustan toteuttamisen eri tavoin
- Esitellään kolme tapaa:
  - Sopimusten sisällyttäminen transaktiotulosteisiin
  - Sopimusverkolle annettavien herätteiden tallentaminen
  - Sidosketjut

ÄLYSOPIMUSALUSTAN TOTEUTTAMINEN

SOPIMUKSET JA TRANSAKTIOTULOSTEET

- Kukin transaktio rakentuu edellisten päälle
- Näin ollen kukin transaktio ottaa syötteenä edellisten transaktioiden tulosteita, ja niiden tulosteet käytetään joidenkin seuraavien transaktioiden syötteinä
- Kukin tuloste voidaan hyväksyä syötteenä vain kerran
- Transaktiotulosteesta, jota ei ole vielä otettu syötteenä mihinkään toiseen transaktioon, käytetään lyhennettä UTXO (Unspent Transaction Output)
- Tulosteeseen voidaan liittää ehtoja, jotka sen syötteeksi ottavan transaktion täytyy toteuttaa
    - Yleensä nämä ehdot liittyvät transaktion vastaanottajan varmistamiseen, mutta niitä voidaan käyttää myös älysopimusten luomiseen



- UTXO:ja käyttävät älysopimukset ovat mahdollisia muun muassa Bitcoinin lohkoketjuun liitettyinä
- Ohjelmointikieli, jolla Bitcoinin transaktioiden skriptit kirjoitetaan, ei ole toteutettu Turing-täydellisenä
    - Pyritty tietoisesti pitämään mahdolliset toiminnot mahdollisimman rajattuina
- Muissa lohkoketjuissa taikka sidosketjuissa samalla tavoin käytettävä kieli voisi toki olla Turing-täydellinen

Transaktiotulosteisiin perustuvien älysopimuksien käyttöä rajoittaa muukin, kuin ohjelmointikielen mahdollinen suppeus.

- Koska sopimus on sidottu kiinni tiettyyn UTXO:hon ja sen arvoon, on vaikeaa ja raskasta luoda sopimuksia, joiden arvo voi vaihdella
    - Käytännössä tällaisen sopimuksen laatiminen vaatisi useita sopimuksia, joista vain valitut toteutetaan.
- Transaktiotulosteilla on vain kaksi sisäistä tilaa: käytetty (spent) ja käyttämätön (unspent). Tästä johtuen vaiheittaisten tai porrastettujen sopimusten toteuttaminen on varsin raskasta

ÄLYSOPIMUSALUSTAN TOTEUTTAMINEN

SOPIMUSVERKOT JA HERÄTTEET

- Ethereum käyttää älysopimusten mahdollistamiseen tileihin ja niiden välisiin herätteisiin perustuvaa mallia
- Sopimusten koodi ajetaan hajautetulla virtuaalikoneella (Ethereum Virtual Machine, EVM)
- Käyttäjät tai muut tahot, joilla on yksityinen tiivisteavain voivat luoda tilin. Lisäksi kullakin sopimuksella on oma tili
- Tileihin sisältyy seuraavat:
    1. tilin käytössä olevan kryptovaluutta ja sen määrä,
    2. transaktioiden ainutkertaisuuden varmistava nonce-numero,
    3. sopimuskoodi ja
    4. sopimuksen käyttämä tallennustila

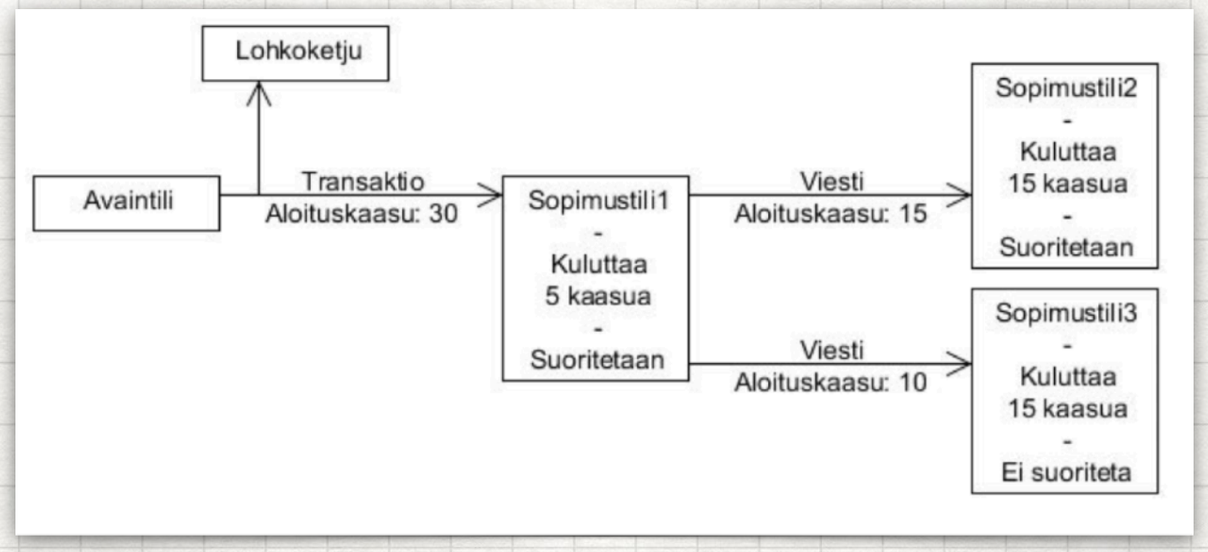
- Tilien välillä kulkee herätteitä, joita on myös kahdenlaisia
  - Avaintilit lähettävät sopimustileille transaktioita
  - Sopimustilit lähettävät toisilleen viestejä
- Kun avaintili lähettää transaktion, tallennetaan kyseinen transaktio myös lohkoketjuun. Vastaanottava sopimustili ajaa saadun herätteen koodinsa läpi, tekee itseensä tarvittavat muutokset ja mikäli tarpeellista lähettää viestejä muille sopimustileille, jotka jatkavat saadun tiedon käsittelyä

On merkillepantavaa, että tämän tyyppiseen lohkoketjuun ei tallenneta jokaista kertaa, kun tilien välillä siirtyy arvoa. Lohkoketjuun tallentuvat vain heräteaallon liikkeelle laittavat transaktiot ja niiden keskinäinen järjestys.

ESIMERKKI: ETHEREUM

SOPIMUSVERKOT JA HERÄTTEET

- Ethereum käyttää sopimusten laskennan rajoittamiseen niin kutsuttua kaasumaksua (gas)
- Avaintilien lähettämiin transaktioihin sisältyy sen käsittelyyn max käytettävän kaasun määrä sekä yhden kaasuyksikön hinta
- Transaktion ja sitä seuraavien viestien käsittelyssä käytetty laskennallinen askel tai sykli kuluttaa yhden tai useamman yksikön kaasua
- Viesteihin sisältyy niille itselleen ja niitä seuraaville viesteille osoitettu kaasun määrä
- Laskenta päättyy, kun se saadaan valmiiksi tai kun kaikki kaasu on käytetty
- Transaktion tekijä asettaa sopimuksen käytettävissä olevan kaasumäärän
- Kaasu vähentää turhaan käytettyjä laskentaresursseja, estää päättymättömät silmukat, suojaa järjestelmää palvelunestohyökkäyksiltä ja vähentää kompleksisuutta



ÄLYSOPIMUSALUSTAN TOTEUTTAMINEN

ÄLYSOPIMUKSET SIDOSVERKOSSA

- Jos älysopimusten toteuttaminen lohkoketjun x avulla on hankalaa, voidaan kyseiseen ketjuun sidostaa toinen, älysopimuksille parempana alustana toimiva lohkoketju
- Tätä kutsutaan kaksisuuntaiseksi sidokseksi (2-way peg)
- Kaksi ketjua sidostetaan toisiinsa varmistamalla, että niiden arvo suhteessa toisiinsa pysyy aina samana ja tarjoamalla tapa siirtää arvoa ketjujen välillä.

ÄLYSOPIMUSALUSTAN TOTEUTTAMINEN

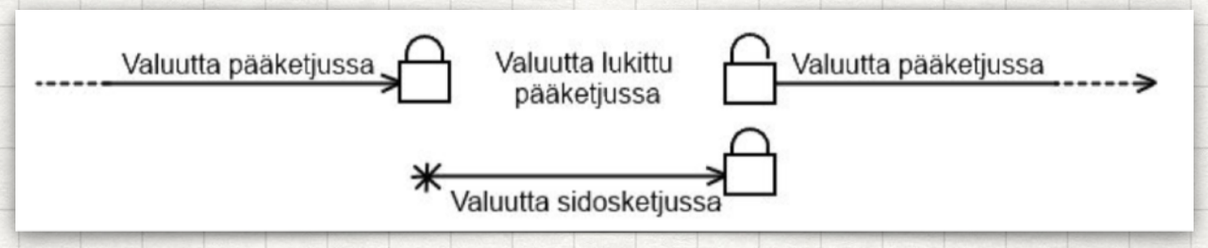
ÄLYSOPIMUKSET SIDOSVERKOSSA

- Käyttäjä siirtää pääketjussa olevaa valuuttaa lukittuun kohteeseen, josta sen saa ulos vain todistamalla että hän on tehnyt vastaavanlaisen siirron sidosketjussa

- Siirrot todistetaan käyttämällä SPV-todistusta (Simplified Payment Verification)

- Niin kauan, kuin valuutta pääketjussa on lukittuna saa käyttäjä käyttöönsä sidosketjussa vastaavan määrän valuuttaa

- Mikäli ketjut ovat sidoksissa useisiin toisiin ketjuihin on sidosketjun valuuttaan merkitty, mistä se on alunperin kotoisin



- Arvon siirto ketjujen välillä vaatii luottamusta louhijoihin, jotka antavat valuutan käytettäväksi
- Täytyy varmistaa, että pää- ja sidosketjujen valuutat pysyvät saman arvoisina, tai muuten varmistaa valuuttojen arvojen vastaavuus
- Sidosketjuilla voi olla hyvinkin erilaisia funktioita:
  - Esimerkki: sidosketju yhdistää hallintopapereita hallinnoivan lohkoketjun, sekä valuuttalohkoketjun
    - Suorittamalla valuuttalohkoketjussa transaktion, käyttäjälle myönnetään pääsy tiettyyn hallintopaperilohkoketjuun talletettuun sisältöön
  - Esimerkki: kommunikaation mahdollistaminen ei-vapaiden ja suljettujen lohkoketjujen sekä vapaiden ja avoimien lohkoketjujen välillä

ÄLYSOPIMUSTEN OHJELMOINTI

KIELET

- Älysopimusalustat sallivat usein sopimusten kirjoittamisen usealla eri ohjelmointikielellä
- Ethereum ja Rootstock: Solidity, Serpent (2.0) ja Lisp-Like-Language (LLL). Solidity muistuttaa vahvasti Javaa, Serpent Pythonia ja LLL Lispiä. Ne ovat kuitenkin muistuttamiaan kieliä erikoistuneempia nimenomaan älysopimusten luomiseen
- Corda: Java
- Lisk: Javascript
- Hyperledger: Go

MAHDOLLISESSA ETEHEREUM-HARJOITUSTYÖSSÄ KÄYTETÄÄN JOTAKIN ÄLYSOPIMUSOHJELMOINNILLE SPESIFIÄ OHJELMOINTIKIELTÄ

ÄLYSOPIMUSTEN OHJELMOINTI

KOODIESIMERKKI

- Vierellä oleva koodiesimerkki on Solidityn dokumentaatiosta (Ethereum 2017)
- Sillä luodaan hyvin yksinkertainen kryptovaluutta
- Ohjelmointi 1 & 2 käyneet ymmärtänevät sisällön hyvin
- ERC20-standardi

```
pragma solidity ^0.4.0;

contract Coin {
    // The keyword "public" makes those variables
    // readable from outside.
    address public minter;
    mapping (address => uint) public balances;

    // Events allow light clients to react on
    // changes efficiently.
    event Sent(address from, address to, uint amount);

    // This is the constructor whose code is
    // run only when the contract is created.
    function Coin() {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) {
        if (msg.sender != minter) return;
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) {
        if (balances[msg.sender] < amount) return;
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        Sent(msg.sender, receiver, amount);
    }
}
```

ÄLYSOPIMUKSET

TURVALLISUUS JA AUDITOINTI

- Lohkoketjujen muuttumattoman luonteen vuoksi älysopimusten turvallisuus ja auditointi on äärimmäisen tärkeää
- Esimerkiksi, kun koodi on lähetetty EVM:lle suoritettavaksi Ethereumin pääverkkoon, päätöstä ei voida perua
- Jos koodissa on bugi, mahdollinen hyökkääjä saattaa voida esimerkiksi varastaa kaikki sopimuksen kautta siirrettävät rahat itselleen
- Ks. Edellisen luennon esimerkki: The DAO

Ethereumin toinen perustaja Vitalik Buterin listaa seuraavia ohjelmointivirheitä esiin suurimmista älysopimusten ohjelmointivirheistä johtuvista tappiotapauksista (Buterin 2016):

1. Väärän muuttujan tai funktion kutsuminen

2. Ei-julkiseksi tarkoitettu data julkisesti tarkasteltavaa (peleissä huijaaminen, yksityisyysrikokset)

3. Valuutansiirto epäonnistuu liian pienen kaasurajan vuoksi

- Ongelmat suurimmassa osassa tapauksia hyvin yksinkertaisia, mutta mikäli koodia ei auditoida riittävän tarkasti on sopimus täysin arvoton

- Ongelmat voivat myös olla monimutkaisempia, ja Buterin nostaa esille myös rajatapauksia, joissa ongelmat ovat peliteoreettisia rajatapauksia, joiden luokittelu bugiksi ylipäätään on kyseenalaista (The DAO)

- Älysopimusten yksinkertainen auditointi ja validointi on hyvin pitkälti pyritty automatisoimaan, ja Internetistä löytyykin jo useita palveluja joissa voi tarkistuttaa koodinsa

ÄLYSOPIMUKSET

ÄLYSOPIMUSAUDITOINNIN ONGELMIA

- Inhimillisyyden ongelma:
  - Varkaus ja tappio ovat pohjimmiltaan sosiaalisia rakennelmia, ja älysopimuksien yhteydessä lahjoituksen ja varkauden eron määrittelee ihmisen arvomaailma
  - **Älysopimus toimii pilkuntarkasti niin kuin se on ohjelmoitu, ja jos joku pystyy nostamaan älysopimuksesta kaikki siellä liikkuvat rahat, hän on tehnyt niin älysopimuksen sääntöjen puitteissa**

- Älysopimuksissa tulee ottaa huomioon reiluuden konsepti—kaikki älysopimuksen toiminta on sen ohjelmoinnin puitteissa sallittua

- Suuria valuuttamääriä liikuttelevissa älysopimuksissa ei todennäköisesti koskaan tule riittämään formaali ja koneistettu tarkistuttaminen ja varmentaminen, sillä reiluus ei ole matemaattisesti todistettavissa oleva konsepti, vaan ihmisillekin äärimmäisen monimutkainen sosiaalinen rakennelma, jonka yksiselitteinen määrittäminen ei liene koskaan joka tilanteessa mahdollista

- Varmistamisessa, tarkistuttamisessa ja auditoinnissa tulisi ihanteellisessa tilanteessa aina olla läsnä inhimillinen elementti — joku, joka tarkastelee sopimuksen sallimaa toimintaa, ja arvioi odottamattomienkin toimintojen merkitystä eettiseltä näkökannalta

Koska älysopimuksissa “koodi on laki,“ Daniel Zakrisson 2017 ehdottaa seuraavaa rakenteellista linjausta älysopimusten ohjelmointiin:

1. Auditoidun lähdekoodin käyttö ja kierrättäminen (esimerkiksi ERC20 - tokenistandardi)

2. Tokeneiden liikkeellepano ja hallinnointi tulisi tapahtua eri sopimuksista käsin

3. Sopimuksissa tulisi olla hätäkatkaisu yllättävien tilanteiden varalta (The DAO)

4. Tulisi implementoida funktio jolla voidaan siirtää sopimuksen omistajuus välittömästi

(Nämä ominaisuudet soveltuvat suurimpaan osaan yleisimmistä käytössä olevista älysopimuksista, mutta jotkin spesifit käyttötapaukset saattavat vaatia esimerkiksi kahden viimeisen kohdan sivuuttamisen)

LOPPU

— Kysymyksiä?