Toteutusdokumentti

Projekti lyhyesti

Projektin tarkoituksena oli tuottaa implementaatio joka simuloisi oikeaa virtuaalivaluuttaa ja perustuisi alkuperäiseen Satoshi Nakamoton julkaisuun Bitcoin: A Peer-to-Peer Electronic Cash System. Projektissa rajattiin ulos alkuun varsinainen P2P toiminnallisuus, eli lohkoketjun jakaminen useiden noodien kesken ja keskityttiin simuloimaan lohkoketjun toimintaa paikallisesti yhdellä noodilla. Transaktioiden allekirjoitusta varten implementoitiin RSA-algoritmi ja muita pienempiä algoritmeja.

Projektin aikana kävi selväksi, että alkuperäinen julkaisu Bitcoinista ei ole riittävän yksityiskohtainen ja nykyinen Bitcoin implementaatio ei myöskään vastaa sitä kovinkaan hyvin. Pääasiallisena lähteenä onkin siis käytetty kirjaa Mastering Bitcoin, 2nd ed. jonka on kirjoittanut Andreas M. Antonopoulos sekä Bitcoinin varsinaista lähdekoodia. Jonkin verran vaikutteita on myös haettu Naivecoin projektista (https://github.com/conradoqg/naivecoin).

Ohjelman rakenne

Ohjelma rakentuu kolmen päämoduulin varaan:

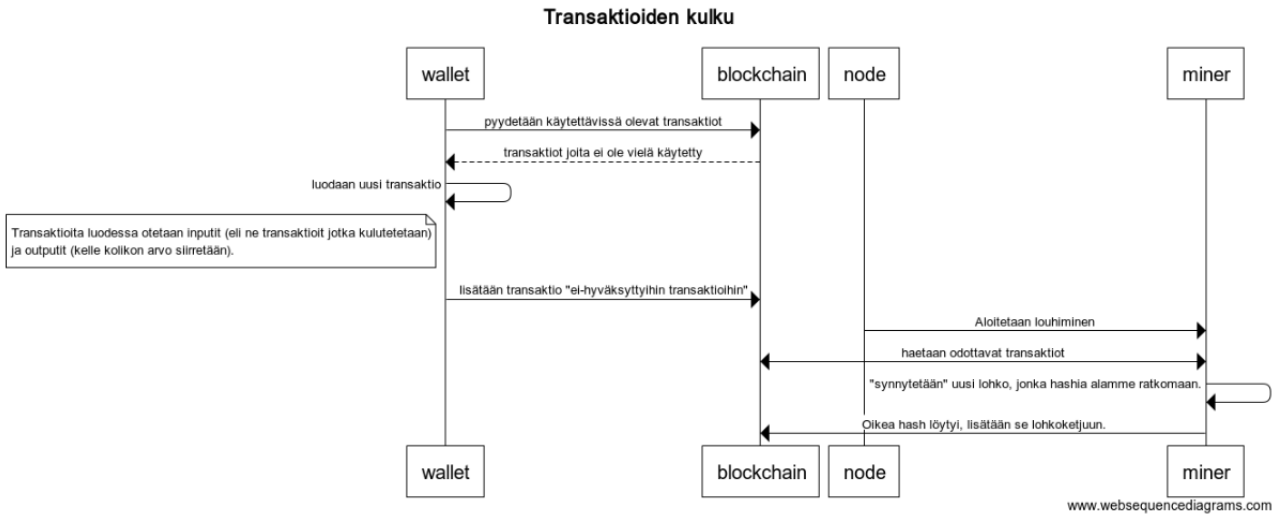
lib pitää sisällään lohkoketjujen, transaktioiden ja algoritmien implementaatiot

node pitää sisällään yksinkertaisen HTTP-serverin joka tarjoaa rajapinnat transaktioiden suorittamiseen, lompakon balanssin tarkistamiseen ja rahan siirtoon. Node tarjoaa myös ns. louhimisominaisuuden jolla lohkoketjuun voidaan lisätä lohkoja.

demoui on JavaScriptillä implementoitu *single page app* joka mahdollistaa lompakon hallinnan selaimessa. Siitä voi myös käynnistää mining prosessin.

”Kolikon” liikkuminen lohkoketjussa

DummyCoin mahdollistaa arvon siirtämisen kahden osoitteen välillä. Osoitteet ovat RSA-algoritmilla luotuja julkisia avaimia.



Sekvenssikaavio kuvaa implementaation perustoiminnallisuuden, kun arvoa siirretään kahden osoitteen välillä.

Algoritmien analyysi

Tässä analyysissä keskitytään pääasiassa RSA:han liittyviin algoritmeihin. Tässä RSA on implementoitu perustuen *modular exponentation* algoritmiin. Tämän perusteella algoritmin suoritusaika on seuraava: oletetaan, että julkinen avain (e,n), salainen avain (d, n) toteuttavat $\lg e = O(1)$, $\lg d \leq \beta$ *ja* $\lg n \leq \beta$. Tällöin transaktion allekirjoituksen varmentaminen julkisella avaimella vaatii O(1) modulaarista kertolaskua ja käyttää $O(\beta^2)$ bittioperaatiota. Allekirjoitus salaisella avaimella vaatii $O(\beta^3)$ bittioperaatiota. [CLRS]

Avaimien luomiseen käytetään Miller-Rabinin alkulukutestiä. Se toimii niin, että pyydämme satunnaislukugeneraattorilta isoja lukuja ja tarkistamme, että onko kyseessä alkuluku. Se voidaan siis löytää joko ensimmäisellä, tai vasta kymmenennellä kerralla. Usein luku tulee kuitenkin yllättävän nopeasti.

Testattu seuraavalla koodilla, eli n=1000 ja arvottu luku on 256 bittiä pitkä. RNG on Pythonin random moduulin getrandbits jolla saamme juuri oikean pituisen luvun. Käytämme s:n arvona 40:tä jonka pitäisi tuottaa meille hyvin isolla varmuudella oikea alkuluku.

```
tries = []
for i in range(1000):
    j = 0

    while not is_prime(getrandbits(256), s=40):
        j+=1

    tries.append(j)

print "Avg Tries: ", (sum(tries) / float(len(tries)))
```

Tulos on, että kokeilu joudutaan tekemään keskimäärin n. 175 kertaa. Eli osuimme siis keskimäärin n. 1/175 todennäköisyydellä. Jos tutkitaan todennäköisyyttä osua alkulukuun, niin alkulukulauseeseen perustuen todennäköisyys osua 256 bittiseen alkulukuun on $1/\lg N = 1/\lg 2^{256}$, eli:

$$\frac{1}{175} \approx \frac{1}{\lg(2^{256})} \approx 0.0057.$$

Implementaatio näyttäisi tällä perusteella toimivan oikein.

Puutteet ja parannusehdotukset

Tällä hetkellä ohjelmisto ei ole oikeasti hajautettu, sen tekeminen ei kuitenkaan olisi kovin monimutkaista, ainoastaan työlästä testaamisen takia. Tästä syystä se myös jätettiin pois alun perin. Lisäksi olisin halunnut implementoida jonkinlaisen turvallisen randomin, mutta siihen ei aika riittänyt. Blockchainin havainnollistaminen UI:ssa olisi myös mielenkiintoinen lisä. Itse valuuttaimplementaatioon liittyen joitain tarkistuksia puuttuu louhimisosiosta. En ehtinyt niitä lisätä ja koska sinällään ne tehdään kyllä kertaalleen eri vaiheessa. Jos tästä tekisi hajautetun, pitäisi koodia hieman refaktoroida niin, että transaktioiden verifiointia siirretään louhimisen puolelle.

Todellisuudessa Bitcoinissa ei käytetä RSA:ta. RSA on valittu tähän puhtaasti siksi, että se sopi paremmin projektin aikavaatimuksiin.

Lähteet

[CLRS] Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (2009). Introduction to algorithms s. 962 Cambridge, Mass. : New York: MIT Press ; McGraw-Hill.