IZRADA KRIPTOVALUTE ZASNOVANE NA TEHNOLOGIJI BLOCKCHAIN

Zubić, Zvonimir

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split / Sveučilište u Splitu**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:228:668290

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-12-26**

Repository / Repozitorij:

Repository of University Department of Professional Studies

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijske tehnologije

ZVONIMIR ZUBIĆ

Z A V R Š N I R A D

IZRADA KRIPTOVALUTE ZASNOVANE NA TEHNOLOGIJI BLOCKCHAIN

Split, lipanj 2019.

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijske tehnologije

Predmet: Programski alati na UNIX računalima

Z A V R Š N I R A D

Kandidat: Zvonimir Zubić

Naslov rada: Izrada kriptovalute zasnovane na tehnologiji blockchain

Mentor: Nikola Grgić, viši predavač

Split, lipanj 2019.

Zahvala:

Želio bih se zahvaliti svom mentoru Nikoli Grgiću na uloženoj energiji, vremenu, strpljivosti i stručnoj pomoći te što mi je omogućio potrebnu opremu tijekom pisanja ovog završnog rada. Hvala na potpori i kontinuiranoj inspiraciji.

Zahvalio bih se i svim svojim prijateljima i kolegama koji su bili uz mene kroz sve uspone i padove. Bez njih bi sve bilo mnogo teže.

Na kraju, zahvalio bih se svojim najbližima na ljubavi i potpori koju su mi pružili kroz sve godine studiranja bez obzira na uspjeh.

Hvala vam svima od srca!

SADRŽAJ

SAŽETAK

Kroz ovaj rad precizno je objašnjeno kako izgraditi novu kriptovalutu po uzoru na već postojeću kriptovalutu. Opisane su tehnologije koje se koriste kako bi kriptovaluta funkcionirala. Pripremljeni su potrebni programi za izgradnju klijenta kriptovalute na sustavu Linux i objašnjena njegova svrha. Nakon toga je napravljena izmjena postojeće kriptovalute. Razlozi izmjene određenih parametara su detaljno objašnjeni. Navedeni su mogući problemi pri izradi i njhova rješenja. Na kraju je izgrađena kriptovaluta i podignuta mreža čvorova.

Ključne riječi: *blockchain*, kriptovaluta, Litecoin, Sosscoin

SUMMARY

Creating blockchain based cryptocurrency

Through this paper it's precisely explained how to build a new cryptocurrency based upon cryptocurrency that already exists. Technologies that are used for cryptocurrency to function are described. Programs that are needed to build cryptocurrency client on the Linux system are prepared and their purpose is explained. After that, the changes have been made on existing cryptocurrency. Reasons for modifying certain parameters are explained in detail. Possible problems with building a client are mentioned, and their solutions. In the end, cryptocurrency has been built and node network lifted.

Keywords: blockchain, cryptocurrency, Litecoin, Sosscoin

1. UVOD

Papirnati novac je prihvaćen u društvu kao sredstvo za plaćanje dobara i usluga, ali sve više se koriste aplikacije koje nude banke. Navedene aplikacije koriste mrežnu infrastrukturu koja je povezana s centralnim poslužiteljom koji prati i upravlja transakcijama. Iz toga je jasno da banke imaju uvid u sve transakcije. Bankarski sustav onemogućuje privatne transakcije i zbog toga su nastale kriptovalute. Zamišljene su kao digitalna gotovina i kao takve daju mogućnost privatnog obavljanja transakcija putem interneta. Budući da su za razliku od postojećih sustava koji nude banke kriptovalute većinom decentralizirane kontrola kriptovalute pripada zajednici. Većina korisnika interneta je danas upoznata s najpoznatijom kriptovalutom, Bitcoinom. No, to nije jedina kriptovaluta. Bitcoin je kriptovaluta otvorenog kôda zbog čega su na osnovi njega nastale mnoge druge kriptovalute koje se nazivaju *altcoin-i*.

Cilj ovog završnog rada je izradom nove kriptovalute objasniti način na koji kriptovalute funkcioniraju. Razumijevanjem rada kriptovalute se nastoji smanjiti nepovjerenje koje ulijevaju prosječnim korisnicima interneta.

Nakon uvoda, u drugom poglavlju opisane su tehnologije potrebne za rad kriptovalute. Razlozi zbog kojih je odabrana određena kriptovaluta i njen izvorni kod su navedeni unutar trećeg poglavlja. U četvrtom poglavlju su nabrojani paketi potrebni za izgradnju klijenta kriptovalute, te objašnjena njihova svrha. Poslije toga, u petom poglavlju opisuje se postupak izgradnje kriptovalute. Početak rada na vlastitoj valuti je opisan u šestom poglavlju, a nakon toga u zadnjem poglavlju je dan zaključak.

2. TEHNOLOGIJE

Unutar ovog poglavlja su opisane tehnologije korištene za rad kriptovalute. Izvorni kod odabrane kriptovalute je temeljen na Bitcoin-u. Zbog toga su najčešći programski jezici korišteni za rad kriptovalute programski jezik C i C++, s time da je C++ više zastupljen. Programski jezik Python, Linux ljuska i skripta Bash se koriste unutar ovog rada za izradu nove kriptovalute. I na kraju je opisana tehnologija *blockchain*, koja je najbitnija za nastanak kriptovaluta.

2.1. Programski jezik C

C je proceduralni programski jezik opće namjene koji je nastao 1972. godine od strane Dennisa Ritchieja. Iako veoma star ovaj programski jezik se koristi u mnogim aplikacijama kao što su operacijski sustavi iOS i Windows, videoigre te softveri koje se koriste u filmskoj industriji. Razlog njegovom višegodišnjem korištenju nije samo njegova fleksibilnost u korištenju, nego i njegova efikasnost. Programski jezik C je brz i stabilan i to je uočeno od strane programerske zajednice. Zbog toga je C utjecao na programerske jezike koji su nastali kasnije kao što su C++, C#, Java, PHP, Python i mnogi drugi. Zato ne začuđuje činjenica da je dio programskog koda za klijente pisan u programskom jeziku C.

2.2. Programski jezik C++

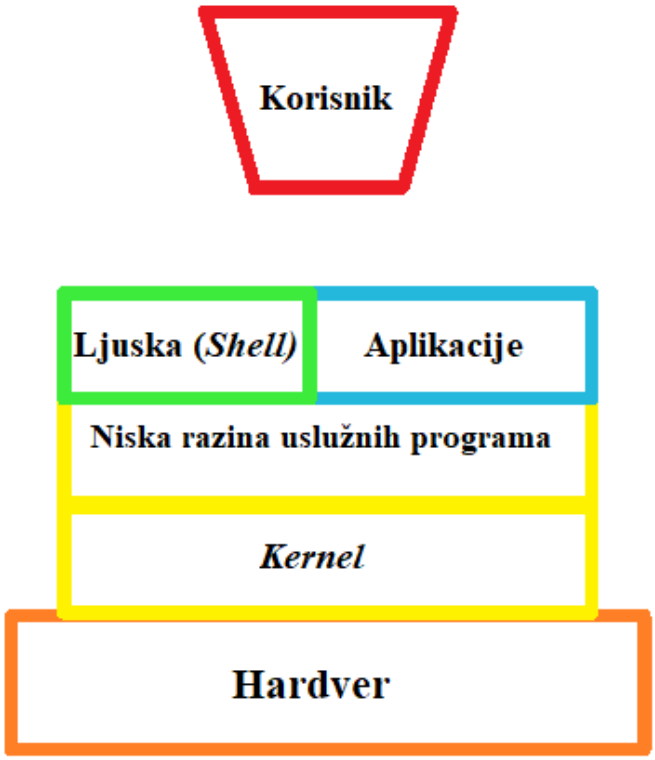
Iako je programski jezik C veoma zastupljen, ipak je kod kriptovaluta najzastupljeniji C++. Razvio ga je Bjarne Stroustrup davne 1979. godine, no standardiziran je tek 1998. godine. Prvotno je nazvan C withClasses, ali je kasnije je preimenovan u C++. Taj naziv je dobio jer je Stroustrup koristio C jezik za izgradnju svog jezika i zamišljen je kao poboljšana verzija programskog jezika C koja će imati mogućnost korištenja za objektno orijentirano programiranje. Zbog toga je C++ također veoma popularan jer je kao i C efikasan i upotrebljiv u širokom opsegu programa. Još uvijek se radi na razvoju programskog jezika C++. Zadnja nadogradnja standarda se dogodila 2017. godine, a iduća je planirana 2020. godine. Danas mnogi operacijski sustavi, pretraživači, sistemski *driveri* te video igre su pisani u C++-u. Iz tog razloga ga koriste i poznate kompanije, a za primjer se može uzeti Amazon, Facebook i Adobe.

2.3. Programski jezik Python

Za razliku od C++ koji je programski jezik srednje razine, Python je programski jezik visoke razine[1][2]. Nastao je 1990. godine od strane Guida van Rossuma, a objavljen je 1991. godine. Glavni motiv iza toga je izgradnja lakše čitljivog programskog kôda. Zanimljiv je način na koji je ovaj programski jezik dobio ime, a to je po televizijskoj seriji Monty Python's Flying Circus. Čitljiviji programski kôd je postignut korištenjem uvlačenja jer se umjesto posebnih znakova za početak i kraj programskog bloka koristi uvlačenje. Mana ovog programskog jezika je brzina izvršavanja programa napisanih u njemu. To je zato što je interpreterski jezik, a ne kompajlerski kao C i C++. Iako je to utjecalo na njegovu raširenost unutar programerske zajednice, ipak se broj njegovih korisnika kontinuirano povećava. I to ne samo zbog lako čitljivog kôda, nego i zbog njegovih mogućnosti. Naime, Python podržava više načina programiranja kao što je objektno orijentirano, funkcionalno i proceduralno programiranje. U izradi kriptovalute se koristi za generiranje novog *Genesis Block-a* i pridruženog *Merkle Root-a.*

2.4. Linux ljuska

Kroz ovaj će se završni rad koristiti operacijski sustav Linux, stoga je neizostavno poznavanje terminala i naredbi za rad u njemu. Terminal je bitan jer omogućuje da se putem tipkovnice daju naredbe operacijskom sustavu kako bi se izvršile. Točnije, terminal prosljeđuje naredbe softveru koji se naziva ljuska (eng. *shell*), a on prosljeđuje operacijskom sustavu. On to čini tako da otvori grafičko korisničko sučelje koje korisnik vidi, te preko kojega komunicira s ljuskom.



Slika 1: Linux terminal

2.5. Skripta Bash

Kod svakog se češćeg korištenja terminala korisnik susretne s određenim slijedom naredbi koje se ponavljaju. Kako bi se izbjeglo to ponavljanje, koriste se određene skripte, u ovom slučaju Bash skripta. Naredbe koje se moraju više puta upisivati skripta će izvršiti za korisnika. Iako uvelike olakšavaju rad, skripte nisu dobre ako se u njima nalazi veliki slijed naredbi. Razlog tomu je to što svaka pojedina naredba u osnovi poziv vanjskog programa [3]. Zato se skripte sporije izvršavaju od kompajliranih programa. Skriptnih jezika ima mnogo, a u ovom se završnom radu se koristi Bash zato što je to jedan od najpopularnijih skriptnih jezika za Unix operacijske sustave, a operacijski sustav koji se koristio je Linux distribucija Ubuntu[4].

2.6.Blockchain

Najvažnija tehnologija na kojoj se zasniva većina današnjih kriptovaluta je *blockchain*. Prijevod same riječi na hrvatski jezik je lanac blokova. Blokovi predstavljaju digitalne zapise koji su međusobno povezani kriptografskim algoritmima. Ti algoritmi osiguravaju autentičnost

i sigurnost informacija pomoću složenih matematičkih formula. Na ideju *blockchain-a* se došlo već 1991. godine [5], ali tek je 2008. godine uspješno dizajniran. Za njegov nastanak zaslužan je Satoshi Nakamoto, no još uvijek nije poznato odnosi li se ime na jednu osobu ili na grupu ljudi [6]. Zato je poznato da je svrha njegovog nastanka biti glavna komponenta kriptovalute *Bitcoin*. Preciznije rečeno, *blockchain* ima ulogu javne transakcijske knjige u čiji zapis može bilo tko imati uvid. Posebnost ove „knjige" je u tome što rješava problem duple potrošnje sredstava bez potrebe za centralnim poslužiteljom ili pouzdanim autoritetom. Način na koji to rješava leži u samoj njenoj strukturi. *Blockchain* se ne nalazi na jednom računalu, nego na više njih. Svaka osoba može preuzeti lanac, tako da ne postoji centralna vlast koja kontrolira informacije nego svaka osoba koja je preuzela lanac. Kad se doda novi blok u lancu taj lanac može biti prihvaćen jedino ako se čitava mreža usuglasi da je ispravan. Kako bi došlo do opće suglasnosti između korisnika mreže pravila po kojima se dolazi do suglasnosti su precizno određena.

Zajednica se usuglasi koristeći algoritam konsenzusa [7]. Taj se algoritam koristi za postizanje dogovora o vrijednosti podatka. Algoritam konsenzusa se koristi kod distribuiranih i decentraliziranih sustava kao što je *blockchain*. Osmišljeni su kako bi se postigla pouzdanost u mreži koja uključuje više nepouzdanih čvorova.

Bitcoin koristi algoritam konsenzusa pod nazivom „Dokaz o Radu" (engl. *Proof of Work*) [8]. Dokaz o radu je podatak kojega je zahtjevno financijski, energetski i vremenski proizvesti, ali je lako ostalim korisnicima mreže provjeriti ispravnost tog podatka. Pomoću tog algoritma se potvrđuju transakcije i proizvode novi blokovi. Blok je prihvaćen od strane mreže jedino ako korisnik koji je izrudario blok ima dokaz o radu kojim potvrđuje ispravnost svih podataka unutar bloka.

Sigurnost proizlazi, ne samo iz distribuiranosti, već i iz činjenice da lanac nije moguće mijenjati. Lancu je jedino moguće dodati novi blok. Pokušaj mijenjanja bloka unutar postojećeg lanca mijenja čitavi lanac jer je svaki blok povezan s prethodnim blokom s već spomenutim kriptografskim algoritmom. To znači da se zajednica ne bi usuglasila s izmijenjenim lancom i time se eliminira mogućnost izmjene lanca.

Kako se točno povezuju blokovi kod kriptovaluta? Blok unutar sebe ima ugrađen *hash* bloka prije njega, a nakon što dobije vlastiti *hash* taj se blok dodaje u lanac [9]. *Hash* je niz znakova čija je veličina fiksno određena. Dobije se pomoću kriptografskog *hash* algoritma koji koristi

složene matematičke algoritme kako bi, na osnovi ulaznih podataka (čija je veličina proizvoljna), vratio podatke točno određene veličine koje nazivamo *hash* [10]. Ulazni su podatci u slučaju *blockchain-a hash* i transakcije. Transakcije nisu obavezne, dok *hash* prethodnog bloka jest. Kriptografski *hash* algoritam jednosmjerni je algoritam, što znači da ne možemo dobiti ulazne podatke na osnovi *hasha*.

Nakon toga se mora potvrditi ispravnost transakcije, a provjera je zadatak mreže računala. Ako je provjera prošla informacije o transakciji kao što su iznos te digitalni potpis pošiljatelja i primatelja se spremaju unutar bloka. Svi blokovi u lancu su javno dostupni, što znači da bilo tko ima pristup informacijama kao što su vrijeme dodavanja bloka ili visina lanca na kojem je dodan.

3. ODABIR IZVORNOG KÔDA KAO OSNOVE ZA IZGRADNJU VALUTE

U vrijeme kad je ovaj rad pisan na tržištu je dostupno preko 2000 kriptovaluta [11]. Većina tih kriptovaluta dijeli sličnosti s valutama koje su nastale prije njih. No, funkcionalnost kriptovalute ne garantira njen uspjeh na tržištu. To ovisi o mnogo drugih čimbenika koji neće biti pokriveni unutar ovog završnog rada jer zalaze u domenu ekonomije.

Poznata kriptovaluta Litecoin je također nastala na osnovi druge kriptovalute, Bitcoina. Kako bi se lakše prikazala razlika između tih valuta, glavne će razlike biti prikazane tablicom:

Tablica 1: Usporedba Litecoin-a i Bitcoin-a

	Litecoin	**Bitcoin**
Stvorena	7. 11. 2011.	3. 1. 2009.
Stvorio	Charlie Lee	Satoshi Nakamoto
Nagrada po bloku	25 LTC	12.5 BTC
Prosječno vrijeme između blokova	2.5 minute	10 minuta
Ograničenje opskrbe	84 milijuna	21 milijun
Vrijednost jedinice u USD	113 USD	7944 USD
Vrijednost tržišta u USD	7 milijardi USD	140 milijardi USD

Brojke navedene u tablici odgovaraju stanju na tržištu 7. 6. 2019. godine.

Za nastanak Litecoina je zaslužan Charlie Lee, bivši zaposlenik kripto mjenjačnice Coinbase i Google-a. Charlie Lee nije bio zadovoljan načinom na koji Bitcoin radi stoga je odlučio napraviti novu kriptovalutu 2011. godine [11].

Informacije o transakcijama se spremaju u blokove koji se dodaju u lanac, a povezuju se s pripadajućim *hash-om*. Za generiranje *hash-a* potrebno je riješiti kompleksnu matematičku jednadžbu, a da bi se riješila jednadžba potrebna je računalna snaga. Stoga se za svaku uspješno riješenu jednadžbu za nagradu dobiju nove jedinice te valute. U slučaju Bitcoina trenutna nagrada je 12.5 *bitcoin-a,* dok je kod Litecoina 25 *litecoin-a*. Ovaj proces se naziva rudarenje. Rudarenje Bitcoina zahtjeva snažno računalo i mnogo električne energije, stoga je veoma financijski zahtjevno. Prosječno vrijeme potrebno za obradu transakcija i dodavanje novog bloka u lanac je 10 minuta kad je u pitanju Bitcoin. S druge strane, Litecoinu je potrebno 2.5 minute što je 4 puta brže nego u slučaju Bitcoina [12]. Razlog tomu je razlika u vrijednosti

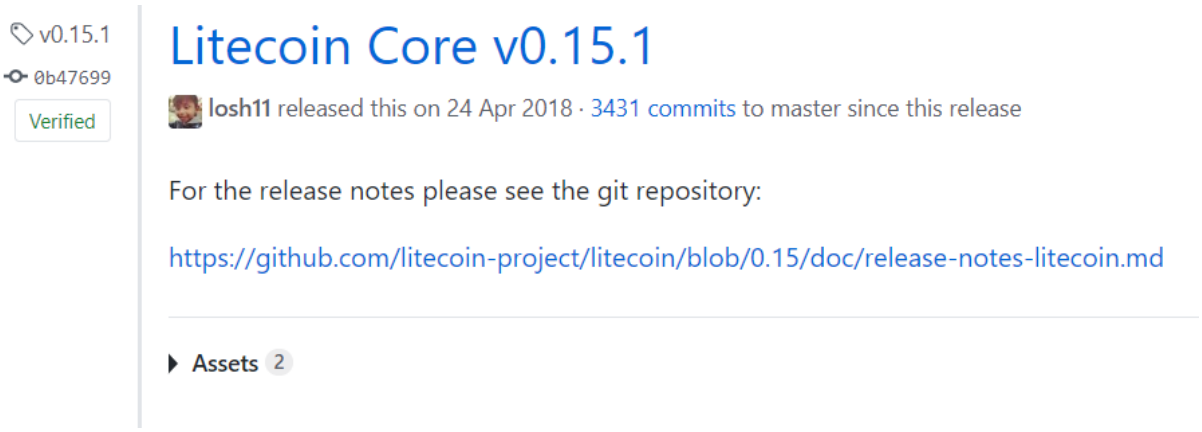
varijable koja određuje ciljano vrijeme rudarenja jednog bloka. Osim toga, još jedna njihova razlika su različiti kriptografski algoritmi. Bitcoin koristi algoritam SHA-256 koji se koristi dugi niz godina, dok Litecoin koristi algoritam posebno dizajniran za njega pod nazivom Scrypt.

Smatra se da je SHA-256 kompleksniji algoritam, ali zato omogućuje više paralelnog procesiranja informacijama. Te činjenice, brzina rudarenja i vrijednost samih *bitcoin-a*, dovele su do toga da sve više rudara koristi specijalizirane uređaje za rudarenje. Takav se uređaj naziva ASIC (*Application-Specific Integrated Circuits*), što bi na hrvatskom značilo: Integrirani sklopovi specifični za aplikaciju. Ovi su uređaji zamijenili korištenje glavnog procesora računala i grafičkih kartica u svrhu rudarenja. No, veoma su skupi za prosječnog korisnika računala.

Ovdje dolazi Scrypt, algoritam koji otežava korištenje specijaliziranog hardvera kao što je ASIC. Jedan od razloga nastanka Litecoina je taj da se svi korisnici valute mogu priključiti rudarenju. To je i ostvareno jer zbog svoje brzine i načina na koji se kôd izvršava, većina *litecoin-a* je i danas izrudarena pomoću računala prosječnih korisnika [14].

I to je razlog zbog čega se u nastavku rada koristi Litecoin kao predložak za novu kriptovalutu. Cilj je da se budući korisnici kriptovalute mogu aktivno uključiti u rudarenje novih jedinica.

Izvorni kôd Litecoina je moguće preuzeti na internetskoj stranici Github, na poveznici https://github.com/litecoin-project/litecoin/releases. Na navedenoj poveznici moguće je pronaći najnoviju i sve prethodne verzije Litecoina. U nastavku će se koristiti verzija Litecoin Core v0.15.1.



Slika 2: Preuzimanje izvornog koda programa Litecoin Core

4. POTREBNI PAKETI ZA IZGRADNJU SOFTVERSKOG KLIJENTA

Kako bi korisnik bio siguran da ima sve potrebne pakete softverskog klijenta za izgradnju vlastite kriptovalute i njeno korištenje prvo je potrebno izgraditi klijenta za kriptovalutu za koju je siguran da je funkcionalna. To je u ovom slučaju Litecoin. Upute za izgradnju klijenta ovisno o operacijskom sustavu mogu se pronaći unutar poddirektorija `doc`. Koristit će se operacijski sustav Ubuntu, a upute za njega se nalaze unutar datoteke `build-unix.md`. Osim popisa programa koji su potrebni za izgradnju klijenta, unutar datoteke se nalaze i upute za instalaciju tih programa.

4.1. Potrebni paketi

U nastavku teksta će biti nabrojani minimalni paketi za pokretanje procesa izgradnje klijenta.

4.1.1. `Libssl`

Ovaj se paket koristi za sigurnu komunikaciju putem interneta, a dio je implementacije OpenSSL projekta SLL i TLS kriptografskih protokola. Sadrži datoteke, zaglavlja, upute i razvojne biblioteke za `libssl` i `libcrypto` [15].

4.1.2. `Libboost`

`Libboost` paket se koristi kao biblioteka za dretve, strukture podataka i ostale slične podatke. To su besplatne C++ izvorišne biblioteke koje funkcioniraju s C++ standardnim bibliotekama [16].

```
node2@node2-VirtualBox:~$ sudo apt-get install libboost-system-dev libboost-filesystem-d
ev libboost-chrono-dev libboost-program-options-dev libboost-test-dev libboost-thread-de
v
Setting up libboost-chrono1.65-dev:amd64 (1.65.1+dfsg-0ubuntu5) ...
Setting up libboost-test-dev:amd64 (1.65.1.0ubuntu1) ...
Setting up libboost-system1.65-dev:amd64 (1.65.1+dfsg-0ubuntu5) ...
Setting up libboost-program-options-dev:amd64 (1.65.1.0ubuntu1) ...
Setting up libboost-date-time1.65-dev:amd64 (1.65.1+dfsg-0ubuntu5) ...
Setting up libboost-chrono-dev:amd64 (1.65.1.0ubuntu1) ...
Setting up libboost-system-dev:amd64 (1.65.1.0ubuntu1) ...
Setting up libboost-filesystem1.65-dev:amd64 (1.65.1+dfsg-0ubuntu5) ...
Setting up libboost-thread1.65-dev:amd64 (1.65.1+dfsg-0ubuntu5) ...
Setting up libboost-thread-dev:amd64 (1.65.1.0ubuntu1) ...
Setting up libboost-filesystem-dev:amd64 (1.65.1.0ubuntu1) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
```

Ispis 1: `Libboost` naredba za instalaciju

4.1.3. `Libevent`

Aplikacijsko programsko sučelje libevent daje mogućnost izvršavanja funkcije povratnog poziva. Ta funkcija se pozove kada se dogodi određeni događaj u deskriptoru datoteke ili nakon što je isteklo vrijeme. Osim toga, `libevent` podržava povratne pozive zbog

redovitih vremenskih ograničenja ili signala. Kod kriptovalute se koristi za asinkrono umrežavanje neovisno o OS-u [17].

U nastavku je prikazan postupak instalacije navedenih paketa.

```
node2@node2-VirtualBox:~$ sudo apt-get install build-essential libtool autotools-dev aut
omake pkg-config libssl-dev libevent-dev bsdmainutils
```

```
node2@node2-VirtualBox:~$ sudo apt-get install build-essential libtool autotools-dev aut
omake pkg-config libssl-dev libevent-dev bsdmainutils
[sudo] password for node2:
Reading package lists... Done
Building dependency tree
Reading state information... Done
bsdmainutils is already the newest version (11.1.2ubuntu1).
The following additional packages will be installed:
  autoconf cpp cpp-7 dpkg-dev fakeroot g++ g++-7 gcc gcc-7 gcc-7-base gcc-8-base
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan4
  libatomic1 libc-dev-bin libc6-dev libcc1-0 libcilkrts5 libevent-core-2.1-6
  libevent-extra-2.1-6 libevent-openssl-2.1-6 libevent-pthreads-2.1-6 libfakeroot
  libgcc-7-dev libgcc1 libgomp1 libitm1 liblsan0 libltdl-dev libmpx2 libquadmath0
  libsigsegv2 libssl-doc libstdc++-7-dev libstdc++6 libtsan0 libubsan0 linux-libc-dev
  m4 make manpages-dev
Suggested packages:
  autoconf-archive gnu-standards autoconf-doc cpp-doc gcc-7-locales debian-keyring
  g++-multilib g++-7-multilib gcc-7-doc libstdc++6-7-dbg gcc-multilib flex bison
  gcc-doc gcc-7-multilib libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg
  libasan4-dbg liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrts5-dbg libmpx2-dbg
  libquadmath0-dbg glibc-doc libtool-doc libstdc++-7-doc gfortran | fortran95-compiler
  gcj-jdk m4-doc make-doc
The following NEW packages will be installed:
  autoconf automake autotools-dev build-essential dpkg-dev fakeroot g++ g++-7 gcc
  gcc-7 libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl
  libasan4 libatomic1 libc-dev-bin libc6-dev libcilkrts5 libevent-core-2.1-6
  libevent-dev libevent-extra-2.1-6 libevent-openssl-2.1-6 libevent-pthreads-2.1-6
  libfakeroot libgcc-7-dev libitm1 liblsan0 libltdl-dev libmpx2 libquadmath0
  libsigsegv2 libssl-dev libssl-doc libstdc++-7-dev libtool libtsan0 libubsan0
  linux-libc-dev m4 make manpages-dev pkg-config
The following packages will be upgraded:
  cpp cpp-7 gcc-7-base gcc-8-base libcc1-0 libgcc1 libgomp1 libstdc++6
8 upgraded, 42 newly installed, 0 to remove and 214 not upgraded.
Need to get 38,7 MB of archives.
After this operation, 137 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Ispis 2: `Libssl` i `libevent` instalacija

Prije nego se izvrši zadana naredba, korisnik je upitan želi li nastaviti s instalacijom jer paketi zauzimaju dodatni prostor na disku od 137 MB. Potrebno je odgovoriti potvrdno te ako je naredba uspješno izvršena dobije se sljedeći ispis.

```
Setting up libevent-dev (2.1.8-stable-4build1) ...
Setting up libalgorithm-diff-xs-perl (0.04-5) ...
Setting up automake (1:1.15.1-3ubuntu2) ...
update-alternatives: using /usr/bin/automake-1.15 to provide /usr/bin/automake (automake
) in auto mode
Setting up cpp (4:7.4.0-1ubuntu2.2) ...
Setting up gcc-7 (7.4.0-1ubuntu1~18.04) ...
Setting up g++-7 (7.4.0-1ubuntu1~18.04) ...
Setting up gcc (4:7.4.0-1ubuntu2.2) ...
Setting up g++ (4:7.4.0-1ubuntu2.2) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up libtool (2.4.6-2) ...
Setting up build-essential (12.4ubuntu1) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
```

Ispis 3: Završetak `libssl` i `libevent` instalacije

4.2. Izborni paketi

Prethodni su paketi dovoljni za pokretanje klijenta, ali to nije dovoljno za izradu kriptovalute jer je cilj pokrenuti u potpunosti funkcionalnu kriptovalutu sa svim svojim mogućnostima. Stoga su potrebni dodatni paketi.

4.2.1. `Miniupnpc`

Za UPnP podršku potrebno je instalirati program `miniupnpc`. Cilj miniUPnP projekta je besplatno softversko rješenje za podršku „Internet Gateway Device“ dijela UPnP protokola [18]. `Miniupnpc` je u trenutku pisanja ovog rada najmanja moguća biblioteka za tu podršku.

```
node2@node2-VirtualBox:~$ sudo apt-get install libminiupnpc-dev
```

Ispis 4: Naredba za instalaciju `miniupnpc-a`

4.2.2. `Libdb4.8`

Digitalni novčanik je jedna od glavnih odlika kriptovaluta, a da bi ga korisnik koristio kod većine kriptovaluta je potrebna Berkeley baza podataka. Berkeley baza podataka sadrži softverske biblioteke koje pružaju ugrađenu bazu podataka visokih performansi za podatke o ključu/vrijednosti [19]. Koristi se `libdb4.8` jer pruža sve potrebne podatke.

```
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:bitcoin/bitcoin
sudo apt-get update
sudo apt-get install libdb4.8-dev libdb4.8++-dev
```

Ispis 5: Naredbe za instalaciju paketa `libdb4.8`

4.2.3. Qt

Qt je razvojni okvir pomoću kojega se mogu razviti aplikacije za različita okruženja i sustave. Najčešće se koristi za izgradnju grafičkog korisničkog okruženja, a u tu svrhu će se koristiti i u ovom završnom radu.

4.2.4. `Protobuf`

Valute s kojima se korisnik svakodnevno susreće najčešće se koriste za plaćanje dobara i usluga. Ta se mogućnost isto očekuje i od kriptovalute. `Protobuf` paket će se koristiti kako bi se omogućilo plaćanje unutar grafičkog korisničkog sučelja. U njemu se nalazi format razmjene podataka koji se koristi za protokol plaćanja.

```
node2@node2-VirtualBox:~$ sudo apt-get install libqt5gui5 libqt5core5a libqt5dbus5 qttoo
ls5-dev qttools5-dev-tools libprotobuf-dev protobuf-compiler
```

Ispis 6: Naredba za instalaciju Qt razvojnog okvira i `protobuf` paketa

5. IZGRADNJA KLIJENTA LITECOIN CORE

Prvi korak u izgradnji klijenta Litecoin Core-a je preuzimanje izvornog kôda. Ako su instalirani svi potrebni paketi navedeni u prošlom poglavlju može se prijeći na samu izgradnju. `Autogen.sh` je prva skripta koju je potrebno pokrenuti, a nakon skriptu `configure`. Sljedeći korak je pokretanje `make` naredbe. Naposlijetku, potrebno je pokrenuti naredbu `make install` za korištenje digitalnog novčanika i grafičkog korisničkog sučelja.

Nakon što se preuzme izvorni kôd kriptovalute i instaliraju svi potrebni paketi za izgradnju može se prijeći na izgradnju kôda. Preuzeti kôd je arhiviran, zato je najprije potrebno raspakirati datoteku prije početka rada. Kad je to izvršeno, korisnik se treba pozicionirati unutar raspakiranog direktorija i izvršiti sljedeće naredbe:

```
node2@node2-VirtualBox:~/litecoin-0.15.1$ bash
node2@node2-VirtualBox:~/litecoin-0.15.1$ ./autogen.sh
```

Ispis 7: Naredba za pokretanje skripte `autogen.sh`

`Autogen.sh` je skripta koja se koristi za generiranje `configure` skripte i svih potrebnih datoteka za nju. Zbog toga se instaliraju prethodni paketi, kao što je `autoconf`, pomoću kojega se generira configure skripta iz `configure.ac` datoteke. Isto tako se kreira `Makefile.in` iz `Makefile.am` datoteke koristeći `automake` paket. To je glavni razlog zbog kojega se treba instalirati `autotools`.

Postoji mogućnost da se pri izvršavanju skripte dobije pogreška „*command not found*“. U tom je slučaju potrebno izmijeniti dozvole u datotečnom sustavu `autogen.sh` datoteke. Treba omogućiti datoteci da se izvršava kao program. To je moguće sljedećom naredbom:

```
node2@node2-VirtualBox:~/sosscoin$ chmod 777 autogen.sh
```

Ispis 8: Naredba za promjenu dozvola nad skriptom

Kad se ponovno pokrene naredba treba se dobiti sljedeći ispis:

```
node2@node2-VirtualBox:~/litecoin-0.15.1$ sudo ./autogen.sh
libtoolize: copying file 'build-aux/m4/lt~obsolete.m4'
configure.ac:78: installing 'build-aux/compile'
configure.ac:28: installing 'build-aux/config.guess'
configure.ac:28: installing 'build-aux/config.sub'
configure.ac:38: installing 'build-aux/install-sh'
configure.ac:38: installing 'build-aux/missing'
Makefile.am:12: warning: user variable 'GZIP_ENV' defined here ...
/usr/share/automake-1.15/am/distdir.am: ... overrides Automake variable 'GZIP_ENV' defin
ed here
src/Makefile.am: installing 'build-aux/depcomp'
src/Makefile.am:500: warning: user target '.mm.o' defined here ...
/usr/share/automake-1.15/am/depend2.am: ... overrides Automake target '.mm.o' defined he
re
parallel-tests: installing 'build-aux/test-driver'
```

Ispis 9: Pokretanje skripte `autogen.sh`

Skripta koja ovo izvršava se može vidjeti na sljedećoj slici:

```
1   #!/bin/sh
2   # Copyright (c) 2013-2016 The Bitcoin Core developers
3   # Distributed under the MIT software license, see the accompanying
4   # file COPYING or http://www.opensource.org/licenses/mit-license.php.
5
6   set -e
7   srcdir="$(dirname $0)"
8   cd "$srcdir"
9   if [ -z ${LIBTOOLIZE} ] && GLIBTOOLIZE="`which glibtoolize 2>/dev/null`"; then
10    LIBTOOLIZE="${GLIBTOOLIZE}"
11    export LIBTOOLIZE
12  fi
13  which autoreconf >/dev/null || \
14    (echo "configuration failed, please install autoconf first" && exit 1)
15  autoreconf --install --force --warnings=all
```

Ispis 10: Skripta `autogen.sh`

Ako je sve uspješno izvršeno, u idućem se koraku može pokrenuti `configure` skripta koja generira `Makefile` datoteku i ostale datoteke koje su potrebne za izgradnju klijenta kriptovalute. Ovdje se koristi `Makefile.in` kao uzorak za generiranje datoteke `Makefile`. Pri tome se koriste alati koji su unaprijed instalirani na Ubuntu operacijskom sustavu.

```
node2@node2-VirtualBox:~/litecoin-0.15.1$ ./configure
```

Ispis 11: Pokretanje skripte `configure`

```
configure: Building ECDSA pubkey recovery module: yes
configure: Using jni: no
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating Makefile
config.status: creating libsecp256k1.pc
config.status: creating src/libsecp256k1-config.h
config.status: executing depfiles commands
config.status: executing libtool commands
Fixing libtool for -rpath problems.

Options used to compile and link:
  with wallet   = yes
  with gui / qt = yes
    qt version  = 5
    with qr     = yes
  with zmq      = yes
  with test     = yes
  with bench    = yes
  with upnp     = yes
  debug enabled = no
  werror        = no

  target os     = linux
  build os      =

  CC            = gcc
  CFLAGS        = -g -O2
  CPPFLAGS      =  -DHAVE_BUILD_INFO -D__STDC_FORMAT_MACROS
  CXX           = g++ -std=c++11
  CXXFLAGS      = -g -O2 -Wall -Wextra -Wformat -Wvla -Wformat-security -Wno-unused-para
meter -Wno-implicit-fallthrough
  LDFLAGS       =
  ARFLAGS       = cr
```

Ispis 12: Ispis skripte `configure`

Nakon što se uspješno izvršila configure skripta pomoću naredbe *make* će se pripremiti softver za instalaciju valute.

```
node2@node2-VirtualBox:~/litecoin-0.15.1$ make
```

Ispis 13: Naredba make

Pri prvom se pokušaju pokretanja može se dogoditi da je zabranjen pristup skripti `genbuild.sh`. No, kao i u slučaju `autogen.sh` skripte samo je potrebno omogućiti skripti da se izvrši kao program.

```
config.status: univalue-config.h is unchanged
config.status: executing depfiles commands
config.status: executing libtool commands
  CXX      lib/libunivalue_la-univalue.lo
  CXX      lib/libunivalue_la-univalue_read.lo
  CXX      lib/libunivalue_la-univalue_write.lo
  CXXLD    libunivalue.la
ar: `u' modifier ignored since `D' is the default (see `U')
make[3]: Leaving directory '/home/node2/litecoin-0.15.1/src/univalue'
  CXX      support/libbitcoin_util_a-lockedpool.o
  CXX      libbitcoin_util_a-chainparamsbase.o
 bin/bash: ../share/genbuild.sh: Permission denied
Makefile:9836: recipe for target 'obj/build.h' failed
make[2]: *** [obj/build.h] Error 126
make[2]: Leaving directory '/home/node2/litecoin-0.15.1/src'
Makefile:9324: recipe for target 'all-recursive' failed
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory '/home/node2/litecoin-0.15.1/src'
Makefile:747: recipe for target 'all-recursive' failed
make: *** [all-recursive] Error 1
```

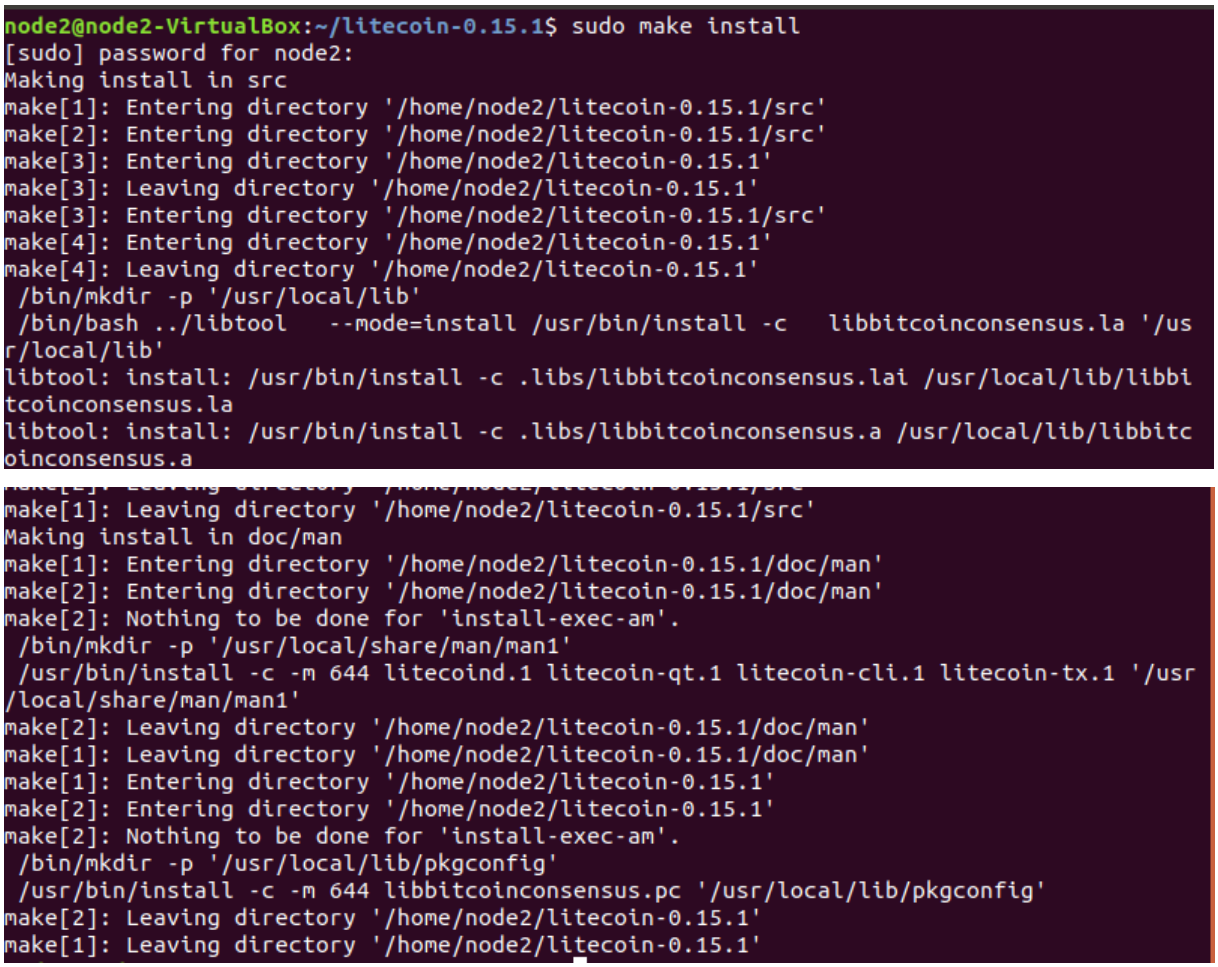
Ispis 14: Pogreška tijekom izvođenja naredba make

Kada se otkloni taj problem naredba se treba uspješno izvršiti. Tada se dobije sljedeći ispis:

```
  CXX      qt/test/qt_test_test_litecoin_qt-paymentservertests.o
  CXX      qt/test/qt_test_test_litecoin_qt-wallettests.o
qt/test/wallettests.cpp:107:6: warning: ‘void {anonymous}::BumpFee(TransactionView&, con
st uint256&, bool, std::__cxx11::string, bool)’ defined but not used [-Wunused-function]
 void BumpFee(TransactionView& view, const uint256& txid, bool expectDisabled, std::stri
ng expectError, bool cancel)
      ^~~~~~~
  CXX      wallet/test/qt_test_test_litecoin_qt-wallet_test_fixture.o
  GEN      qt/test/moc_compattests.cpp
  CXX      qt/test/qt_test_test_litecoin_qt-moc_compattests.o
  GEN      qt/test/moc_rpcnestedtests.cpp
  CXX      qt/test/qt_test_test_litecoin_qt-moc_rpcnestedtests.o
  GEN      qt/test/moc_uritests.cpp
  CXX      qt/test/qt_test_test_litecoin_qt-moc_uritests.o
  GEN      qt/test/moc_paymentservertests.cpp
  CXX      qt/test/qt_test_test_litecoin_qt-moc_paymentservertests.o
  GEN      qt/test/moc_wallettests.cpp
  CXX      qt/test/qt_test_test_litecoin_qt-moc_wallettests.o
  CXXLD    qt/test/test_litecoin-qt
  CXX      test/test_test_litecoin_fuzzy-test_bitcoin_fuzzy.o
  CXXLD    test/test_litecoin_fuzzy
make[2]: Leaving directory '/home/node2/litecoin-0.15.1/src'
make[1]: Leaving directory '/home/node2/litecoin-0.15.1/src'
Making all in doc/man
make[1]: Entering directory '/home/node2/litecoin-0.15.1/doc/man'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/node2/litecoin-0.15.1/doc/man'
make[1]: Entering directory '/home/node2/litecoin-0.15.1'
make[1]: Nothing to be done for 'all-am'.
make[1]: Leaving directory '/home/node2/litecoin-0.15.1'
```

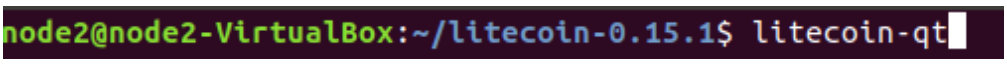
Ispis 15: Uspješno izvršena naredba make

Upozorenja se mogu zanemariti, to je očekivano ponašanje. Sad kad se pripremio softver za instaliranje može se instalirati kriptovaluta naredbom *make install* da bi se mogao koristiti digitalni novčanik i grafičko korisničko sučelje.

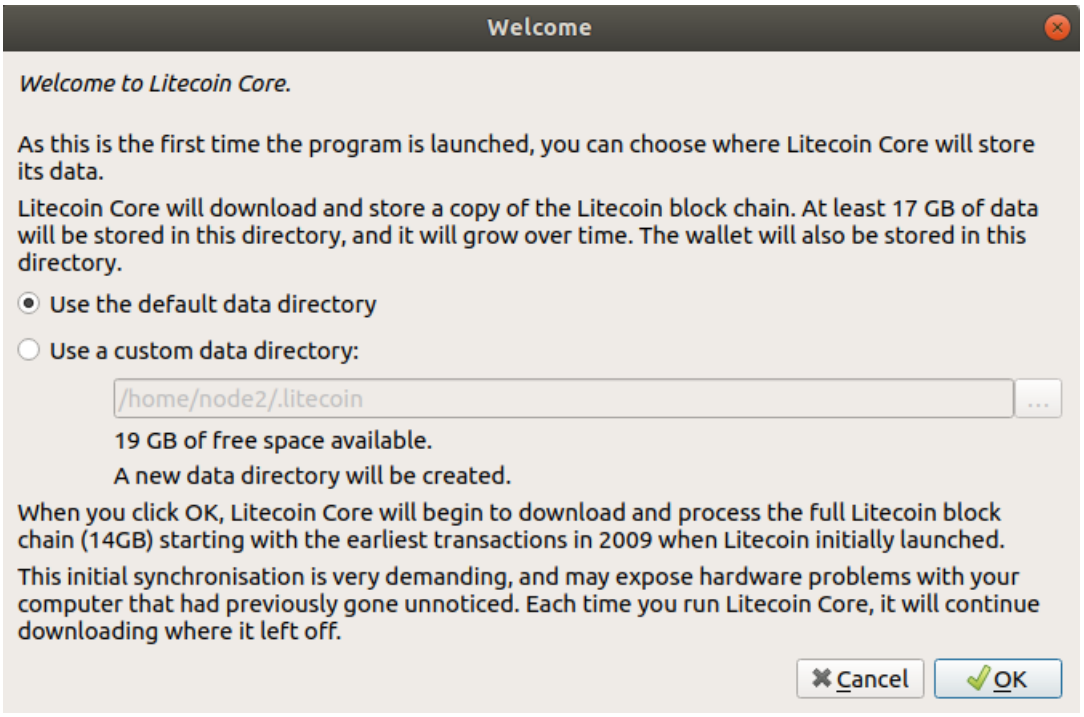


Ispis 16: Naredba make install i njen ispis

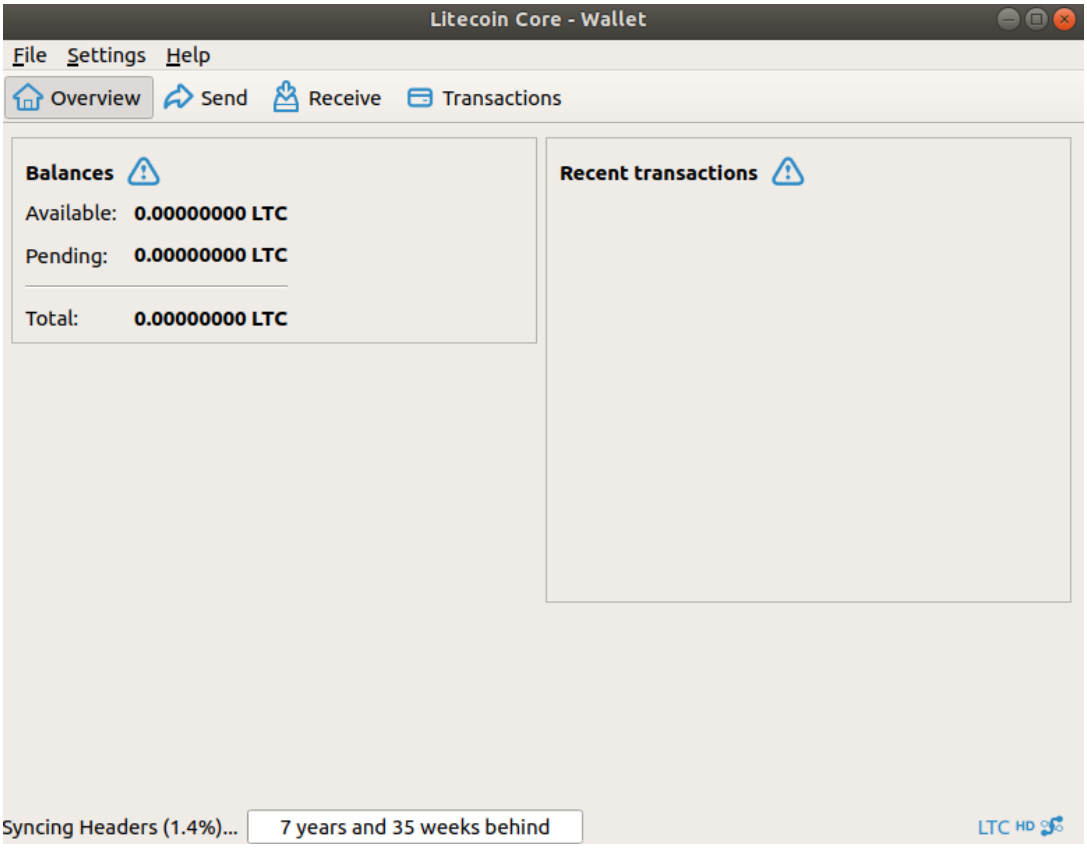
Kako bi korisnik bio siguran da se softver uspješno instalirao, treba pokrenuti grafičko korisničko sučelje kriptovalute naredbom *litecoin-qt*.



Ispis 17: Pokretanje novčanika naredba *litecoin-qt*



Ispis 18: Uspješno pokrenut novčanik za Litecoin



Ispis 19: Uspješno korištenje novčanika za Litecoin

Kada se klijent uspješno pokrene, može se vidjeti da je potrebno preuzeti čitavi lanac kako bi se sinkroniziralo s ostatkom mreže. To znači da se uspješno izgradio klijent i da su instalirani svi potrebni paketi za rad na vlastitoj kriptovaluti.

6. IZGRADNJA KRIPTOVALUTE SOSSCOIN

U ovom poglavlju je opisan postupak izrade kriptovalute Sosscoin. Početni dio se odnosi na preimenovanje postojećih naziva koji se nalaze u preuzetom izvornom kodu. U idućem dijelu je opisan način izmjene parametara unutar koda. Zatim je pomoću skripte generiran izvorni blok i pripadajući *merkle root*. Nadalje, nastavljena je izmjena parametara unutar koda. Ispravnost tih parametara je provjerena izgradnjom klijenta kriptovalute i pokretanjem digitalnog novčanika. Nakon toga su izrađene pripadajuće grafike i zamijenjene s postojećima te ponovno izgrađen klijent kriptovalute. Potom je stvorena konfiguracijska datoteka. Sljedeće je bilo spajanje čvorova. Naposlijetku je započeto rudarenje i testirane su funkcionalnosti valute.

6.1. Izmjena naziva odabrane valute

Naziv kriptovalute koji će se koristiti unutar ovog završnog rada je Sosscoin, a skraćenica OSS. Izmjena postojećih naziva ne predstavlja nikakav zakonski problem zato što se Litecoin distribuira pod licencom MIT/X11. Zbog toga se Litecoin redovito spaja s promjenama od Bitcoina, a nakon spajanja se naprave izmjene naziva datoteka [20]. Naredbe kojima se unutar terminala mogu učiniti izmjene nad svim datotekama su sljedeće:

```
find ./ -type f -readable -writable -exec sed -i "s/Litecoin/Sosscoin/g" {} \;
find ./ -type f -readable -writable -exec sed -i "s/LiteCoin/SossCoin/g" {} \;
find ./ -type f -readable -writable -exec sed -i "s/LTC/OSS/g" {} \;
find ./ -type f -readable -writable -exec sed -i "s/litecoin/sosscoin/g" {} \;
find ./ -type f -readable -writable -exec sed -i "s/litecoind/sosscoind/g" {} \;
```

Ispis 20: Naredbe za izmjenu naziva

Postoji mogućnost da sve datoteke unatoč naredbama ipak ne izmjene naziv, što može rezultirati problemom pri izgradnji klijenta kriptovalute. Zato je potrebno pojedinačno izmijeniti nazive tim datotekama. Lista datoteka koje možda ne promijene naziv unatoč naredbi se nalazi u tablici.

Tablica 2: Datoteke kojima nije izmjenjen naziv

Izvorna datoteka	Izmjenjena datoteka
litecoind.1	sosscoind.1
litecoin-qt.1	sosscoin-qt.1
litecoin-cli.1	sosscoin-cli.1
litecoin-tx.1	sosscoin-tx.1
litecoin_splash.png	sosscoin_splash.png

Postoje i nazivi za jedinice koje su manje od jednog LTC-a. To je također potrebno izmijeniti. Za naziv *lites* je odabran pripadajući naziv ocjene. Naredba kojom se to izvršilo se može vidjeti na slici.

```
find ./ -type f -readable -writable -exec sed -i "s/lites/ocjene/g" {} \;
```

Ispis 21: Naredbe za izmjenu naziva *lites-a*

Jedan Sosscoin sadrži 1000 ocjena. Nakon izmjene svih naziva potrebno je ponovo izgraditi klijenta kriptovalute kako bi korisnik bio siguran da nije došlo do nikakvih pogreški tijekom izmjene.

6.2. Izmjena parametara

Idući će koraci sadržavati izmjenu parametara prema kojima će se izgraditi novi *blockchain* prema drugačijim parametrima nego u slučaju Litecoin-a za novonastalu kriptovalutu.

6.2.1. Promjena parametra `pchMessagestart`

Prvi korak je izmjena niza `pchMessagestart` koji ima četiri člana. Najčešće se nazivaju „magičnim“ brojevima jer se pomoću njih identificiraju poruke klijenta unutar mreže kao da pripadaju određenom protokolu. Vrijednosti brojeva unutar niza mogu biti od 0 do 255 i jedino je bitno da su jedinstvene u odnosu na vrijednosti ostalih kriptovaluta. Bitno je jer ako se koristi ista vrijednost kao kod neke druge kriptovalute klijenti će imati problem pri raspoznavanju. Sosscoin za identifikaciju unutar mreže koristi sljedeće brojeve:

```
/**
 * The message start string is designed to be unlikely to occur in normal data.
 * The characters are rarely used upper ASCII, not valid as UTF-8, and produce
 * a large 32-bit integer with any alignment.
 */
pchMessageStart[0] = 0xd1;
pchMessageStart[1] = 0xa0;
pchMessageStart[2] = 0xf5;
pchMessageStart[3] = 0xbd;
```

Ispis 22: Izmjena parametra `pchMessagestart`

6.2.2. Nadogradnja parametra `base58Prefixes`

Parametar `base58Prefixes` se koristi za prefikse privatnih, javnih i adresa koje mogu samo primati valutu, ali ne i trošiti. Vrijednost s prefiksom varijable `base58Prefixes`

`(PUBKEY_ADDRESS)` se sastoji od javnih ključeva koji predstavljaju mogućnost trošenja. `Base58Prefixes (SECRET_KEY)` su prefiksi za privatni ključ koji klijent koristi kako bi potrošio valutu. Varijabla `base58Prefixes (SCRIPT_ADDRESS)` je povezana s adresama koje su *hashevi* skripti. Način na koji klijent zna koji od dva zadana bajta troši skriptu za primjenu je traženje razlike između početnih bajtova. Preostali su prefiksi `base58Prefixes (EXT_PUBLIC_KEY)` i `base58Prefixes (EXT_SECRET_KEY)`, prefiksi od četiri bajta za takozvane "skrivene adrese". Ovi prefiksi omogućuju adrese koje imaju mogućnost generiranja novih ključeva koji ne mogu trošiti, ali mogu primati isplate. Kod ove dvije varijable je bitno da imaju isti prvi bajt, ali se trebaju razlikovati u ostala tri bajta.

```
base58Prefixes[PUBKEY_ADDRESS] = std::vector<unsigned char>(1,83);
base58Prefixes[SCRIPT_ADDRESS] = std::vector<unsigned char>(1,5);
base58Prefixes[SCRIPT_ADDRESS2] = std::vector<unsigned char>(1,50);
base58Prefixes[SECRET_KEY] =     std::vector<unsigned char>(1,83);
base58Prefixes[EXT_PUBLIC_KEY] = {0xff, 0x88, 0xB2, 0x1E};
base58Prefixes[EXT_SECRET_KEY] = {0xff, 0x88, 0xAD, 0xE4};
```

Ispis 23: Izmjena parametra `base58Prefixes`

6.3. Stvaranje *genesis blocka* i povezanog *merkle roota*

Za stvaranje novog *blockchain-a* potreban je izvorni blok (eng. *genesis block*). Osim spomenutog izvornog bloka potreban je i *merkle root*. *Merkle root* je *hash* koji sadrži sve *hash-eve* svih transakcija koje su dio blokova u *blockchainu*.

No, kôd za rudarenje izvornog bloka se ne nalazi više unutar kôda Litecoin-a. Razlog tomu je što se to izbrisalo iz kôda kod novijih verzija Litecoin-a nakon što je izrudaren izvorni blok.

Postoji više načina za kreiranje novog izvornog bloka. Ovdje će se koristiti besplatno rješenje korisnika lhartikk s internetske stranice GitHub, koji je izradio skriptu pod nazivom „GenesisH0“ u programskom jeziku Python.

6.3.1. Korištenje skripte za generiranje bloka

Skriptu je moguće preuzeti s internetske stranice Github, a moguće je i pronaći unutar direktorija Sosscoin-a u poddirektoriju `GenesisH0-master`.

Za instalaciju paketa potrebnih za pokretanje skripte treba instalirati `Python3`.

```
node2@node2-VirtualBox:~$ sudo apt install python3-pip
```

Ispis 24: Naredba za instaliranje `Python3` paketa

Ako se korisnik susretne s problemom pri instalaciji paketa može pokušati s `Python2` paketom.

```
node2@node2-VirtualBox:~$ sudo apt install python-pip
```

Ispis 25: Naredba za instaliranje `Python2` paketa

Nakon uspješne instalacije potrebnih programa mogu se instalirati potrebni paketi.

```
node2@node2-VirtualBox:~$ sudo pip install scrypt construct==2.5.2
The directory '/home/node2/.cache/pip/http' or its parent directory is not owned by the current user and the cache has been disabled. Please check the permissions and owner of t
hat directory. If executing pip with sudo, you may want sudo's -H flag.
The directory '/home/node2/.cache/pip' or its parent directory is not owned by the current user and caching wheels has been disabled. check the permissions and owner of that dir
ectory. If executing pip with sudo, you may want sudo's -H flag.
Collecting scrypt
  Downloading https://files.pythonhosted.org/packages/80/3d/141eb80e754b86f6c25a2ffaf6c3af3acdb65a3e3700829a05ab0c5d965d/scrypt-0.8.13.tar.gz (53kB)
    100% |████████████████████████████████| 61kB 348kB/s
Collecting construct==2.5.2
  Downloading https://files.pythonhosted.org/packages/14/d7/f94299cc597f74c558907aa76c635966b029d5c58d8cda7ecc4035fb8ab4/construct-2.5.2-py2.py3-none-any.whl (72kB)
    100% |████████████████████████████████| 81kB 553kB/s
Requirement already satisfied: six in /usr/lib/python2.7/dist-packages (from construct==2.5.2)
Installing collected packages: scrypt, construct
  Running setup.py install for scrypt ... done
Successfully installed construct-2.5.2 scrypt-0.8.13
```

Ispis 26: Naredba za instaliranje paketa potrebnih za GenesisH0 skriptu

Za pokretanje skripte potrebno se pozicionirati unutar direktorija u kojem se skripta nalazi. Kako bi se dobio željeni izvorni blok potrebne su određene varijable za njegovu izgradnju. Skripti se proslijede ulazni parametri kao što je vrsta algoritma koji će se koristiti za *hash-iranje*, komad teksta kao dokaz da je *blockchain* nastao na navedeni datum, javni ključ, vremenska oznaka i *nonce*.

U ovom se slučaju koristi algoritam scrypt, ali ako korisnik odluči raditi svoju kriptovalutu na osnovi kriptovalute koja koristi sha256 (kao na primjer Bitcoin), samo je potrebno za taj parametar postaviti sha256.

Tekstualni parametar može biti bilo kakav tekst, ali se najčešće koristi tekst koji dokazuje da novonastala kriptovaluta nije postojala prije navedenog datuma. Zbog toga se koriste članci iz novina kako bi ostali korisnici mogli potvrditi vjerodostojnost izjave. Kriptovaluta nastala unutar ovog završnog rada ima akademsku svrhu, stoga nije korišten naslov iz novina nego jednostavno „This is Sosscoin made for college 4/4/2019“.

Idući parametar je javni ključ. Ovdje je korišten javni ključ od Litecoina i to ne predstavlja problem, ali ako korisnik želi može koristiti vlastiti javni ključ. Razlog tomu je beskorisnost pripadajućeg privatnog ključa jer se ne mogu koristiti Sosscoin-ovi koji su povezani s izvornim blokom. On je dodatni parametar za generiranje novog izvornog bloka i to je njegova jedina svrha.

Za vremensku oznaku je najbolje iskoristiti trenutno vrijeme koje se može pronaći na internetskoj stranici unixtimestamp.com. Taj broj predstavlja vrijeme proteklo u sekundama od 1. siječnja 1970. godine. U slučaju Sosscoin-a to je 1554486171.

Zadnji parametar *nonce* predstavlja proizvoljni broj koji se mijenja za vrijeme rudarenja (sve dok se ne pronađe odgovarajući *hash*). Nije ga moguće izračunati prije pokretanja ove skripte. Zato se to polje može ostaviti prazno ili korisnik može pokušati pogoditi *nonce*. Ako je broj blizu traženog broja skratit će vrijeme potrebno za pronalazak *nonce-a* i *hash-a* izvornog bloka.

```
zvonimir@zvonimir-VirtualBox:~/sosscoin/GenesisH0-master$ python genesis.py -a scrypt -z "This is Sosscoin made for college 4/4/2019" -p "040184710fa689ad5023690c80f3a49c8f13f8d
45b8c857fbcbc8bc4a8e4d3eb4b10f4d4604fa08dce601aaf0f470216fe1b51850b4acf21b179c45070ac7b03a9" -t 1554486171  -n 2084524493
04ffff001d01042a54686973206973205367f7373636f696e206d61646520666f7220636f6c6c65676520342f342f32303139
algorithm: scrypt
merkle hash: e002f1d03a8ebeecb37dc61b19d6f5cfca2ba5945c063a23cb699d2fe7d530b0
pszTimestamp: This is Sosscoin made for college 4/4/2019
pubkey: 040184710fa689ad5023690c80f3a49c8f13f8d45b8c857fbcbc8bc4a8e4d3eb4b10f4d4604fa08dce601aaf0f470216fe1b51850b4acf21b179c45070ac7b03a9
time: 1554486171
bits: 0x1e0ffff0
Searching for genesis hash..
```

Ispis 27: Naredba za pokretanje GenesisH0 skripte

Proces kreiranja izvornog bloka i pronalaska *nonce-a* može dugo potrajati jer računalo pokušava pronaći *nonce* za ulazne parametre koji će ispuniti uvjete izgradnje bloka. Kad se skripta uspješno izvrši dobije se sljedeći ispis:

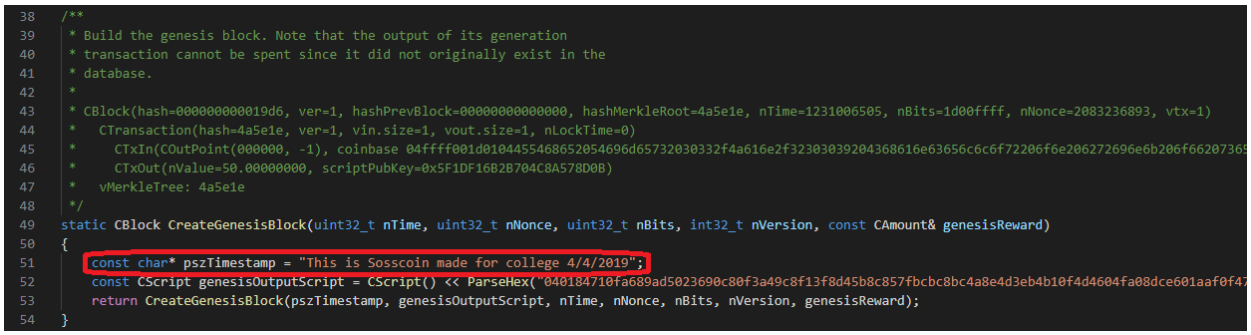
```
genesis hash found!
nonce: 2084591612
genesis hash: 192dac065d700135529286cfdab423894ee530b5f11adc957ba738786d34d9b2
zvonimir@zvonimir-VirtualBox:~/sosscoin/GenesisH0-master$
```

Ispis 28: Uspješno pronađen *hash* izvornog bloka

Ispravnost dobivenog *nonce-a* i *hash-a* izvornog bloka može se provjeriti tako da se ponovo pokrene skripta. Ovaj put s poznatim *nonce-om*. U slučaju da su svi parametri ispravni, skripta mora dati rezultat u roku nekoliko sekundi. U rezultatu treba biti *hash* kojega smo prethodno dobili. Sve dobivene rezultate i ulazne parametre je potrebno sačuvati jer su potrebni u sljedećim koracima.

6.3.2. Izmjena parametara unutar kôda

Nakon pronalaska *hash-a* izvornog bloka i *nonce-a* može se prijeći na izmjene unutar samog kôda. Izmjene će u ovom koraku samo utjecati na datoteku `chainparams.cpp` koja se nalazi unutar direktorija `src`. Prva promjena se odnosi na vremensku oznaku, a nalazi se na 51. liniji kôda. Tu se treba staviti vrijednost varijable `pszTimestamp` na vrijednost koja je korištena pri pokretanju GenesisH0 skripte. Ako je korisnik promijenio javni ključ onda je potrebno napraviti izmjenu u liniji 52.



Ispis 29: Izmjena vremenske oznake

Iduće izmjene će se napraviti unutar `CmainParams` klase. Na liniji 121 gdje se poziva funkcija *`CreateGenesisBlock()`* potrebno je izmjeniti prva dva parametra. Prvi parametar je trenutačno vrijeme koje se koristilo kao ulazni parametar kod pokretanja skripte, a drugi parametar je dobiveni *nonce*. U ovom području je moguće izmjeniti blok nagradu ako korisnik to želi, odnosi se na zadnji parametar koji se šalje funkciji.

```
121         genesis = CreateGenesisBlock(1554486171, 2084591612, 0x1e0ffff0, 1, 50 * COIN);
```

Ispis 30: Izmjena poziva funkcije `CreateGenesisBlock()`

Provjera *hash-a* od izvornog bloka i *merkle root-a* treba biti postavljena na *hash* koji smo dobili koristeći prethodno navedenu skriptu.

```
123         assert(consensus.hashGenesisBlock == uint256S("0x192dac065d700135529286cfdab423894ee530b5f11adc957ba738786d34d9b2"));
124         assert(genesis.hashMerkleRoot == uint256S("0xe002f1d03a8ebeecb37dc61b19d6f5cfca2ba5945c063a23cb699d2fe7d530b0"));
```

Ispis 31: Izmjena parametara `hashGenesisBlock` i `hashMerkleRoot`

Unutar varijable `checkpointData` treba izbrisati sve postojeće kontrolne točke i dodati novu kontrolnu točku za blok 0. Vrijednost te kontrolne točke treba biti *hash* izvornog bloka.

```
147         checkpointData = (CCheckpointData) {
148             {
149                 {  0, uint256S("0x192dac065d700135529286cfdab423894ee530b5f11adc957ba738786d34d9b2")},
150             }
151         };
```

Ispis 32: Izmjena parametra `checkpointData`

Varijabli `chainTxData` treba postaviti UNIX vremensku oznaku na onu koja je korištena pri pokretanju skripte. Ostali parametri trebaju biti postavljeni na 0 jer je u pitanju nova kriptovaluta koja još nije pokrenuta.

```
153         chainTxData = ChainTxData{
154             // Data as of block 59c9b9d3fec105bdc716d84caa7579503d5b05b73618d0bf2d5fa639f780a011 (height 1353397).
155             1554486171, // * UNIX timestamp of last known number of transactions
156             0,  // * total number of transactions between genesis and that timestamp
157                 //   (the tx=... number in the SetBestChain debug.log lines)
158             0.00     // * estimated number of transactions per second after that timestamp
159         };
160     }
```

Ispis 33: Izmjena parametra `chainTxData`

6.4. Izmjena varijabli `nPowTargetTimespan` i `nPowTargetTimeSpacing`

Osim izmjena koje utječu na izvorni blok i *merkle root*, unutar datoteke `chainparams.cpp` je moguće napraviti izmjene koje utječu na vrijeme potrebno za stvaranje novih blokova. Varijabla `nPowTargetTimespacing` označava ciljano vrijeme za rudarenje jednog bloka. Za Sosscoin je to vrijeme postavljeno na 30 sekundi kako bi se lakše demonstriralo rudarenje blokova, dok je kod najpoznatije kriptovalute Bitcoin to vrijeme 10 minuta. S razlogom se naziva ciljano vrijeme jer vrijeme za rudarenje jednog bloka ovisi o računalnoj snazi mreže računala koja rudare. Ako su računala veoma snažna to se vrijeme skraćuje, ako su preslaba to se vrijeme produžuje. Zbog toga je potrebno mijenjati težinu rudarenja blokova, a intervali u kojima se to radi su određeni varijablom `nPowTargetTimespan`. Kod Sosscoin-a je vrijeme prilagođavanja težine rudarenja jednog bloka postavljeno na 5 minuta. Time se pokušava zadržati traženo ciljano vrijeme rudarenja jednog bloka od 30 sekundi. U slučaju Bitcoina vrijeme kad se radi prilagođavanje je 2 tjedna, a kod Litecoina 3.5 dana.

```
83          consensus.nPowTargetTimespan = 10 * 30; // 5 minutes
84          consensus.nPowTargetSpacing = 1 * 30; // 30 seconds
```

Ispis 34: Izmjena parametara `nPowTargetTimespacing` i `nPowTargetTimespan`

6.5. Brisanje dnsseed-a i seednode-ova

Korisnik treba izbrisati čvorove koji su zapisani unutar kôda. *Seed node* je čvor koji ima mnogo veza s ostalim čvorovima i duže vrijeme radi bez prestanka. Njegova je svrha predaja popisa ostalih čvorova za povezivanje čvorovima koji prvi puta pristupaju mreži. Čvorovi koji se spajaju na *seed node* se samo spoje i preuzmu popis ostalih čvorova. Zatim se spajaju s čvorovima koji su unutar popisa i s njima ostaju povezani. *Dnsseed node* predaje listu *seed node-ova*, a ako nije dostupan koriste se *seed node-ovi* koji su zapisani unutar kôda.

Zbog toga je potrebno izbrisati čvorove. U suprotnom, klijent bi se pokušao spojiti s čvorovima koji koriste Litecoin što bi rezultiralo pogreškama i nemogućnošću rada. Promjene je ponovo potrebno napraviti unutar `chainparams.cpp` datoteke. Sosscoin ima zakomentiran taj dio kôda jer nema još dovoljno veliku mrežu da bi imao fiksne čvorove i da bi korisnik koji želi napraviti vlastitu kriptovalutu znao gdje treba u budućnosti dodati vlastite čvorove.

```
126         // Note that of those with the service bits flag, most only support a subset of possible options
127         /*vSeeds.emplace_back("dnsseed.sossco.in", true);
128         vSeeds.emplace_back("seed-a.sosscoin.loshan.co.uk", true);
129         vSeeds.emplace_back("dnsseed.thrasher.io", true);
130         vSeeds.emplace_back("dnsseed.sosscointools.com", true);
131         vSeeds.emplace_back("dnsseed.sosscoinpool.org", true);
132         vSeeds.emplace_back("dnsseed.koin-project.com", false);*/
```

Ispis 35: Brisanje dnsseed-a i seednode-ova

Osim unutar `chainparams.cpp` datoteke promjene je potrebno napraviti i unutar datoteke `chainparamsseeds.h`. Unutar statičke varijable `pnSeed6_main[]` potrebno je izbrisati sve postojeće fiksirane čvorove. Sosscoin i ovdje ima samo zakomentirani kôd kako bi se budući korisnici lakše snalazili unutar kôda.

```
3   /**
4    * List of fixed seed nodes for the sosscoin network
5    * AUTOGENERATED by contrib/seeds/generate-seeds.py
6    *
7    * Each line contains a 16-byte IPv6 address and a port.
8    * IPv4 as well as onion addresses are wrapped inside a IPv6 address accordingly.
9    */
10  static SeedSpec6 pnSeed6_main[] = {
11  /*
12      {{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0x05,0x13,0xab,0xad}, 10333},
13      {{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0x05,0x27,0x40,0x07}, 9555},
14      {{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0x05,0x27,0xae,0x74}, 9555},
15      {{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0x05,0x2d,0x45,0x0d}, 9555},
16      {{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0x05,0x46,0x3c,0x3b}, 9555},
17      {{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0x05,0x53,0x80,0xc7}, 9555},
```

Ispis 36: Brisanje dnsseed-a i seednode-ova unutar datoteke `chainparamsseeds.h`

6.6. Postavljanje minimum chain work-a

Minimalni rad lanca predstavlja očekivani sveukupni broj *hash-eva* potrebnih za izgradnju postojećeg lanca. To je zapisano u heksadecimalnom obliku i nalazi se u datoteci *`chainparams.cpp`*. Prije je najduži lanac predstavljao točan lanac što je omogućilo jednostavne napade na mrežu, stoga se to generalno ne koristi. Korisnik koji započinje potpuno novi lanac treba postaviti ovu vrijednost na 0, što je i učinjeno u slučaju Sosscoin-a.

```
196        // The best chain should have at least this much work.
197        consensus.nMinimumChainWork = uint256S("0x0000000000000000000000000000000000000000000000000000000000000000");
```

Ispis 37: Minimalni rad lanca

6.7. Izmjena zadanog porta

Čvorovi komuniciraju koristeći P2P (isti s istim eng. *peer to peer*) način umrežavanja. To čine preko točno određenog porta. Na primjer, Litecoin koristi port 9333 pa ako korisnik želi smanjiti neželjeni promet prema čvorovima najbolje je koristiti jedinstvenu vrijednost. Jedino je bitno da broj nije manji od 1024 jer nitko neće moći pokrenuti čvor osim *root-a*.. Naredba koja je korištena za izmjenu porta koji koristi Sosscoin je sljedeća:

```
find ./ -type f -readable -writable -exec sed -i "s/9333/9555/g" {} \;
```

Ispis 38: Naredba za izmjenu broja porta

Tablica 3: Datoteke u kojima je promijenjen broj porta

Datoteke
tc.sh
generate-seeds.py
tor.md
chainparams.cpp
net.cpp
proxy_test.py

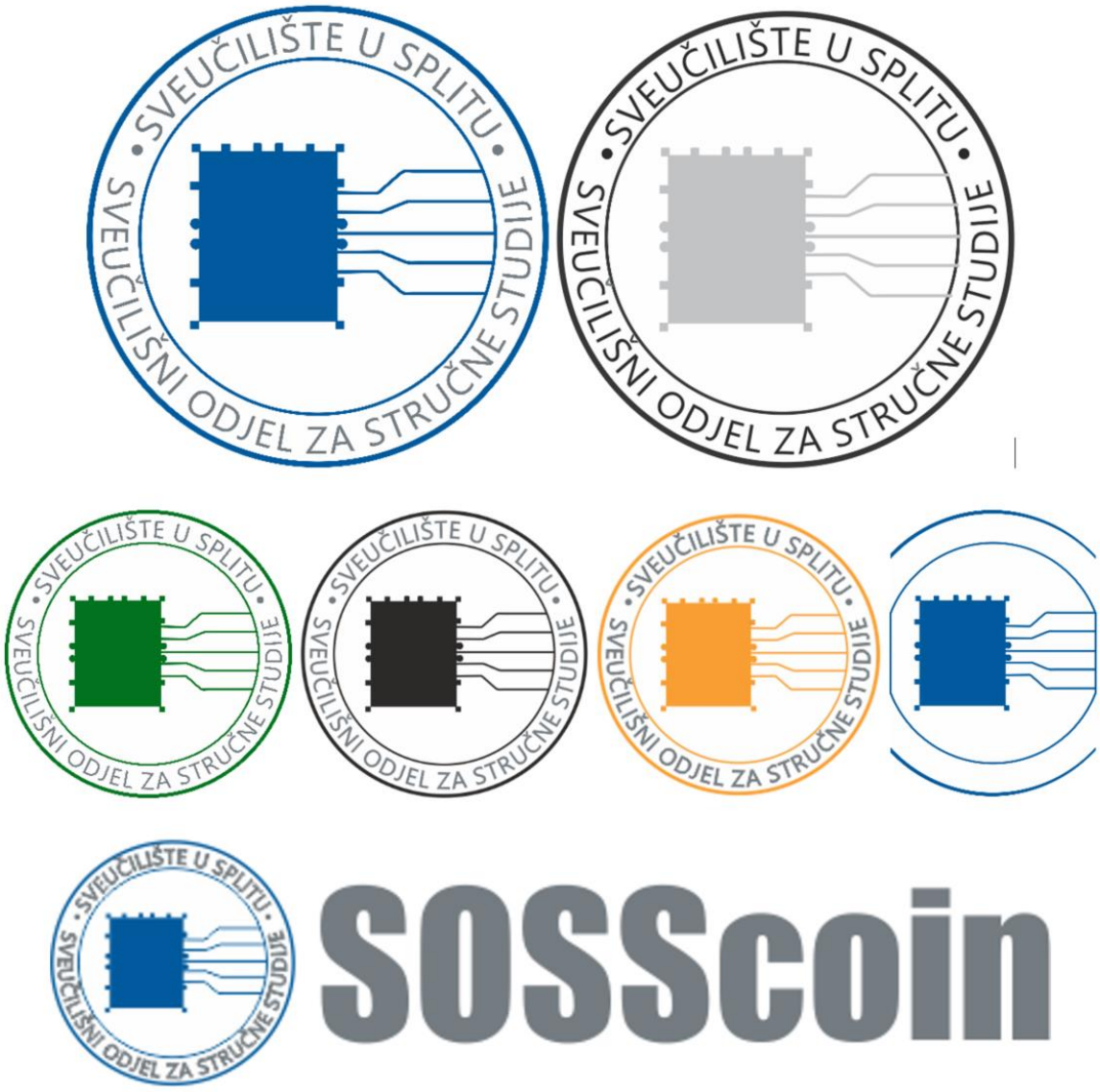
6.8. Izgradnja novčanika

Nakon što je napravljena većina izmjena unutar kôda treba provjeriti jesu li promjene ispravne. Najbolji način za to provjeriti je izgradnja klijenta kriptovalute, ako je sve ispravno klijent će se uspješno izgraditi. Naredbe za izgradnju su ostale nepromijenjene.

Sljedeća provjera je pokretanje novčanika. Još uvijek nisu napravljene sve potrebne izmjene za potpuno funkcionalni novčanik, ali potrebno je pokrenuti novčanik kako bi se generirao skriveni direktorij u kojemu se nalaze informacije o mreži i lancu.

6.8.1. Izmjena loga i pripadajućih grafika

Većina današnjih kriptovaluta se razlikuje ne samo u tehničkom, nego i u vizualnom aspektu. Za Sosscoin je korišten program CorelDRAW jer omogućuje rad s vektorskom grafikom, zbog čega je puno lakše skalirati željene grafike. Korišten je logo od Sveučilišta u Splitu Sveučilišnog odjela za stručne studije.



Slika 3: Logo i grafike Sosscoin-a

Putanju do datoteka koje je potrebno izmijeniti moguće je pronaći unutar tablice.

Tablica 4: Putanje do grafika

Putanja	Datoteka
`src/qt/res/src/`	`bitcoin.svg`

src/qt/res/icons/	faithcoin_splash.png
src/qt/res/icons/	bitcoin_testnet.ico
src/qt/res/icons/	bitcoin.png
src/qt/res/icons/	bitcoin.ico
src/qt/res/icons/	about.png
share/pixmaps/	nsis-wizard.bmp
share/pixmaps/	nsis-header.bmp
share/pixmaps/	bitcoin64.png
share/pixmaps/	bitcoin32.png
share/pixmaps/	bitcoin256.png
share/pixmaps/	bitcoin16.png
share/pixmaps/	bitcoin128.png
share/pixmaps/	bitcoin.ico
doc/	bitcoin_logo_doxygen.png
src/qt/res/icons/	bitcoin.ico
src/qt/res/icons/	bitcoin_testnet.ico

6.9. Stvaranje konfiguracijske datoteke

Za klijente koji će koristiti kriptovalutu potrebno je generirati konfiguracijsku datoteku. Pomoću nje klijenti će znati gdje će emitirati transakcije, primati *blockchain* ažuriranja i slati *hash-eve*.

Konfiguracijska datoteka se može pronaći u skrivenom direktoriju .(`ime kriptovalute`), u ovom slučaju `.sosscoin` koji se nalazi unutar *home-a*, a ako se ne nalazi u skrivenom direktoriju potrebno je izgenerirati datoteku. Ime datoteke je `(ime kriptovalute).conf`, u ovom slučaju `sosscoin.conf`. Vrijednosti `addnode` se koriste za spajanje klijenta na postojeće čvorove na sljedeći način:

```
30    addnode=192.168.1.114:9555
31    addnode=192.168.1.115:9555
```

Ispis 39: Dodavanje čvorova

Ako korisnik želi lokalno rudariti vlastitu kriptovalutu treba izmijeniti dodatne parametre `rpcuser` i `rpcpassword`. U datoteci su napisani komentari od strane zajednice koji olakšavaju pronalazak potrebnih parametara kako bi se klijent prilagodio korisnikovim

zahtjevima. Cijela datoteka se može vidjeti na idućoj slici, a parametri koje je potrebno izmijeniti za lokalno rudarenje su zaokruženi.

```
# sosscoin.conf configuration file. Lines beginning with # are comments.
# Network-related settings:
# Run on the test network instead of the real sosscoin network.
#testnet=0
# Connect via a socks4 proxy
#proxy=127.0.0.1:9050
################################################################
##            Quick Primer on addnode vs connect              ##
##  Let's say for instance you use addnode=4.2.2.4            ##
##  addnode will connect you to and tell you about the        ##
##    nodes connected to 4.2.2.4.  In addition it will tell   ##
##    the other nodes connected to it that you exist so       ##
##    they can connect to you.                                ##
##  connect will not do the above when you 'connect' to it.   ##
##    It will *only* connect you to 4.2.2.4 and no one else.  ##
##                                                            ##
##  So if you're behind a firewall, or have other problems    ##
##  finding nodes, add some using 'addnode'.                  ##
##                                                            ##
##  If you want to stay private, use 'connect' to only        ##
##  connect to "trusted" nodes.                               ##
##                                                            ##
##  If you run multiple nodes on a LAN, there's no need for   ##
##  all of them to open lots of connections.  Instead         ##
##  'connect' them all to one node that is port forwarded     ##
##  and has lots of connections.                              ##
##       Thanks goes to [Noodle] on Freenode.                 ##
################################################################
# Use as many addnode= settings as you like to connect to specific peers
addnode=192.168.1.114:9555
addnode=192.168.1.115:9555
# ... or use as many connect= settings as you like to connect ONLY
# to specific peers:
#connect=localhost:9555
# Do not use Internet Relay Chat (irc.lfnet.org #sosscoin channel) to
# find other peers.
#noirc=0
# Maximum number of inbound+outbound connections.
#maxconnections=
# JSON-RPC options (for controlling a running sosscoin/sosscoind process)
# server=1 tells sosscoin-QT to accept JSON-RPC commands.
server=1
# You must set rpcuser and rpcpassword to secure the JSON-RPC api
rpcuser=zvonimir
rpcpassword=123
# How many seconds sosscoin will wait for a complete RPC HTTP request.
# after the HTTP connection is established.
rpctimeout=30
# By default, only RPC connections from localhost are allowed.  Specify
# as many rpcallowip= settings as you like to allow connections from
# other hosts (and you may use * as a wildcard character):
#rpcallowip=10.1.1.34
#rpcallowip=192.168.*.*
#rpcallowip=1.2.3.4/255.255.255.0
rpcallowip=127.0.0.1
```

Ispis 40: Konfiguracijska datoteka prvi dio

```
# Listen for RPC connections on this TCP port:
#rpcport=9555
#listen=0
#daemon=1
# You can use sosscoin or sosscoind to send commands to sosscoin/sosscoind
# running on another host using this option:
#rpcconnect=192.168.2.29
# Use Secure Sockets Layer (also known as TLS or HTTPS) to communicate
# with sosscoin -server or sosscoind
#rpcssl=1
# OpenSSL settings used when rpcssl=1
#rpcsslciphers=TLSv1+HIGH:!SSLv2:!aNULL:!eNULL:!AH:!3DES:@STRENGTH
#rpcsslcertificatechainfile=server.cert
#rpcsslprivatekeyfile=server.pem
# Miscellaneous options
# Set gen=1 to attempt to generate sosscoins
gen=1
# Use SSE instructions to try to generate sosscoins faster.
4way=1
# Pre-generate this many public/private key pairs, so wallet backups will be valid for
# both prior transactions and several dozen future transactions.
#keypool=100
# Pay an optional transaction fee every time you send sosscoins.  Transactions with fees
# are more likely than free transactions to be included in generated blocks, so may
# be validated sooner.
paytxfee=0.001
# Allow direct connections for the 'pay via IP address' feature.
#allowreceivebyip=1
# User interface options
# Start sosscoin minimized
#min=1
# Minimize to the system tray
#minimizetotray=1
```

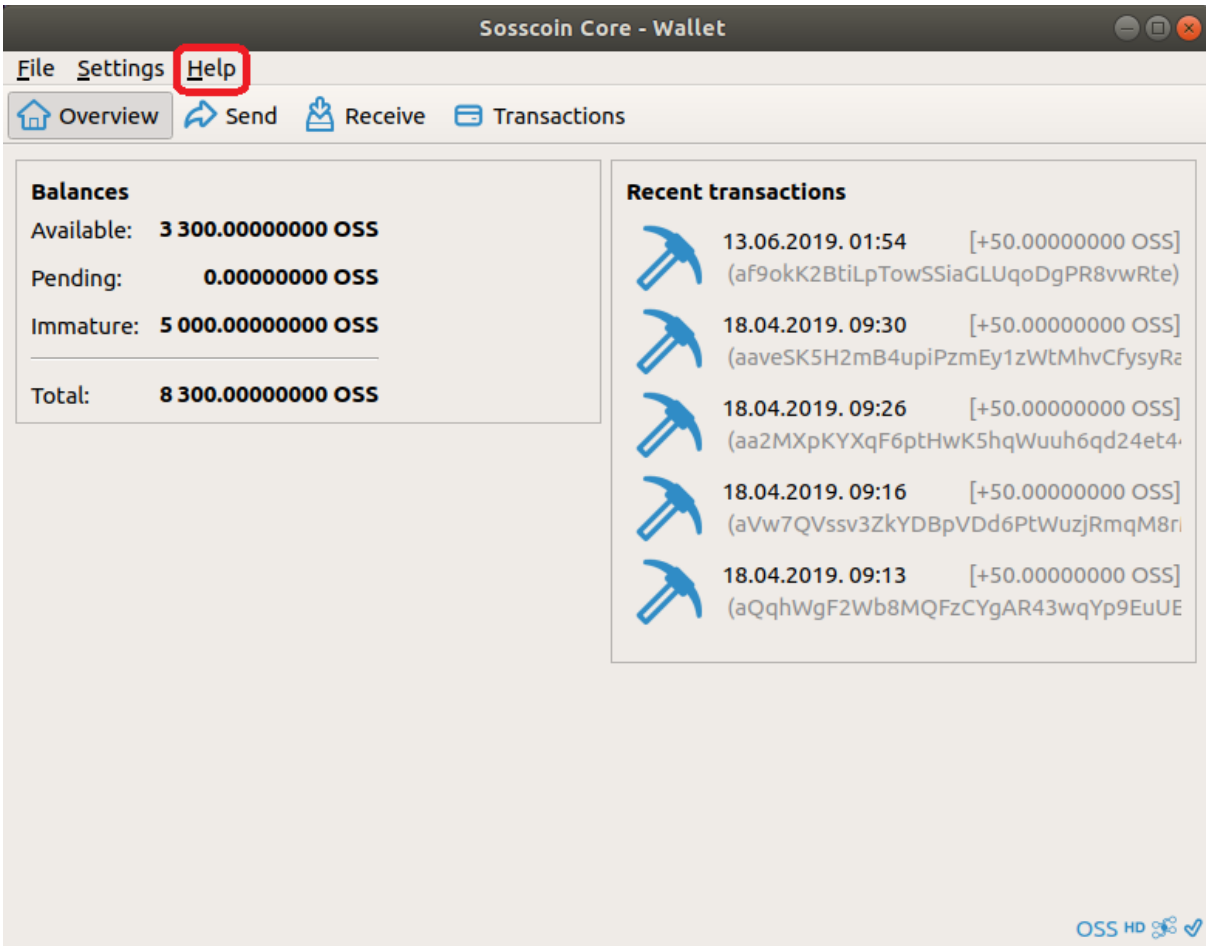
Ispis 41: Konfiguracijska datoteka drugi dio

6.10. Spajanje dvaju čvorova

Svi potrebni koraci za rad kriptovalute su završeni. Preostalo je distribuirati kôd i testirati novčanik i mrežu. Izvorni kôd kriptovalute je potrebno prebaciti na drugo računalo izgraditi ponovo klijenta kriptovalute i unutar konfiguracijske datoteke dodati poznate čvorove. Za to je potrebno otvoriti portove (u ovom slučaju 9555) za izlazni i ulazni promet. Također, potrebno je potvrditi da su čvorovi dostupni. To se može vidjeti unutar prozora za *debug-iranje*. Do prozora za *debug-iranje* se može doći pokretanjem digitalnog novčanika te ulaska u sekciju *Help*. Pri pokretanju digitalnog novčanika prikažu se dva upozorenja zbog promjene ikona koja se mogu zanemariti jer ne utječu na rad kriptovalute.

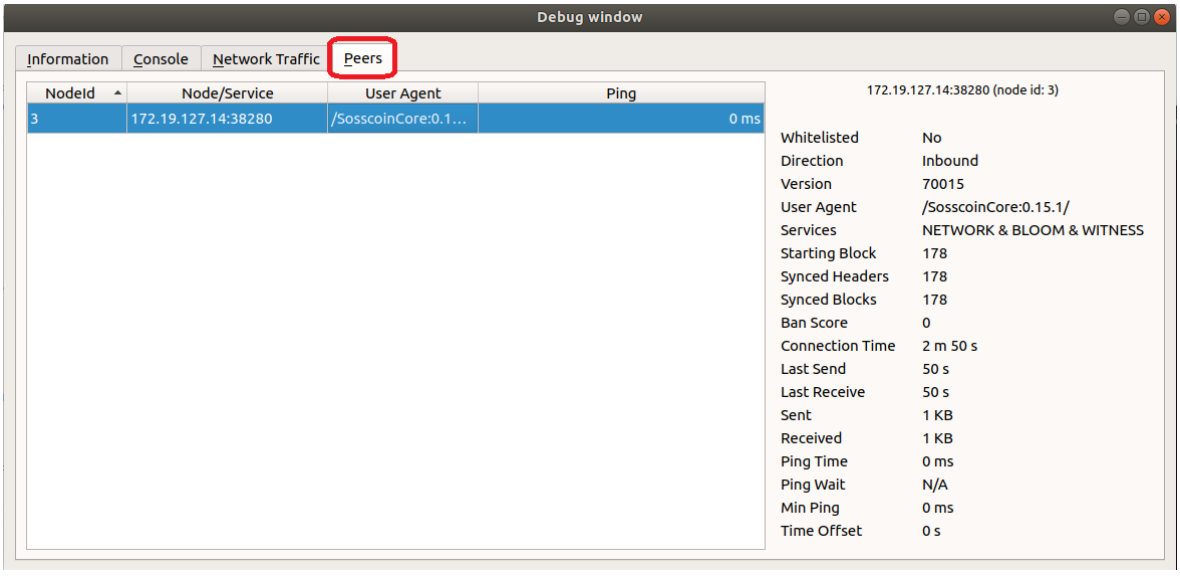
```
node2@node2-VirtualBox:~/sosscoin$ sosscoin-qt
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: cHRM chunk does not match sRGB
```

Ispis 42: Pokretanje digitalnog novčanika za Sosscoin



Slika 4: Digitalni novčanik za Sosscoin

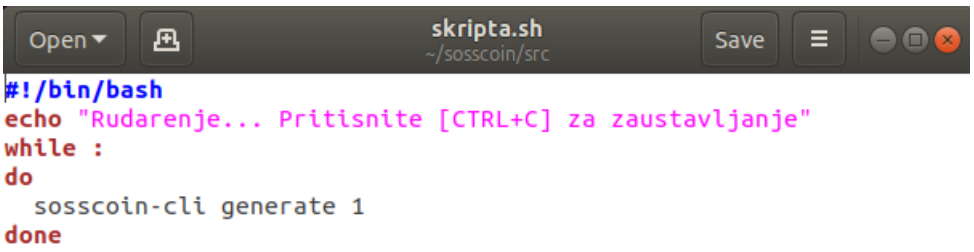
Unutar prozora za *debug-iranje* moguće je pronaći sekciju *Peers*. U toj sekciji korisnik može provjeriti je li uspješno spojen na čvorove koje je dodao unutar konfiguracijske datoteke.



Slika 5: Provjera povezanosti čvorova

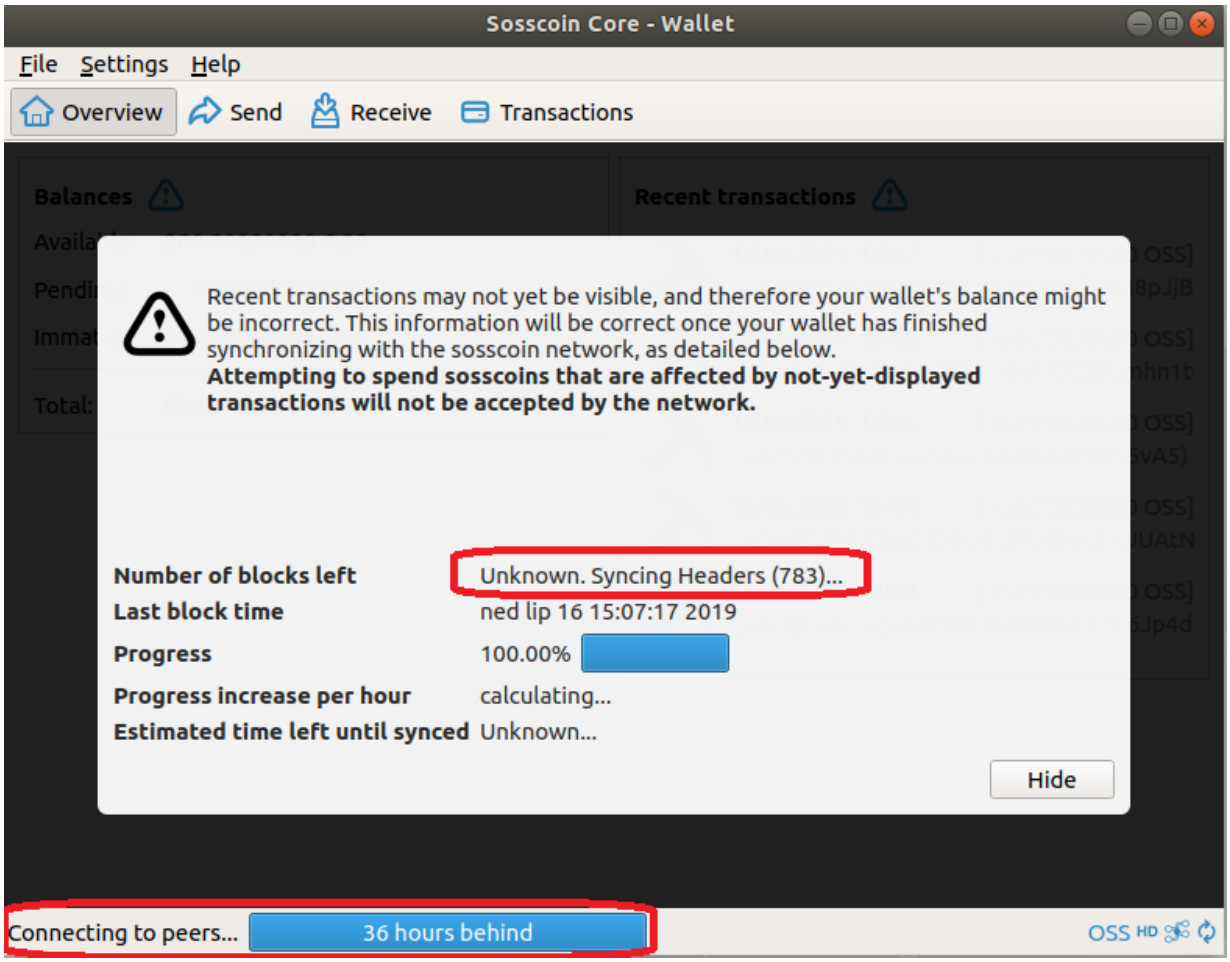
6.11. Rudarenje blokova

Ako su se uspješno spojili čvorovi, to je predzadnji korak do novog funkcionalnog *blockchain-a*. Budući da je lanac nov, ne bi trebao biti problem rudariti blokove na osobnom računalu. Za početak se treba pokrenuti vlastiti digitalni novčanik. Nakon toga u drugom prozoru se treba pokrenuti naredba sosscoin-cli generate 1. Ovom naredbom računalo pokušava generirati nove blokove, ako to uspješno izvrši nove jedinice Sosscoin-a su pridodane u novčanik. Kako bi se izbjeglo često pisanje naredbe, napisana je skripta koja pokreću tu istu naredbu za korisnika. Može se pronaći i pokrenuti unutar direktorija `src` pod nazivom `skripta.sh`.



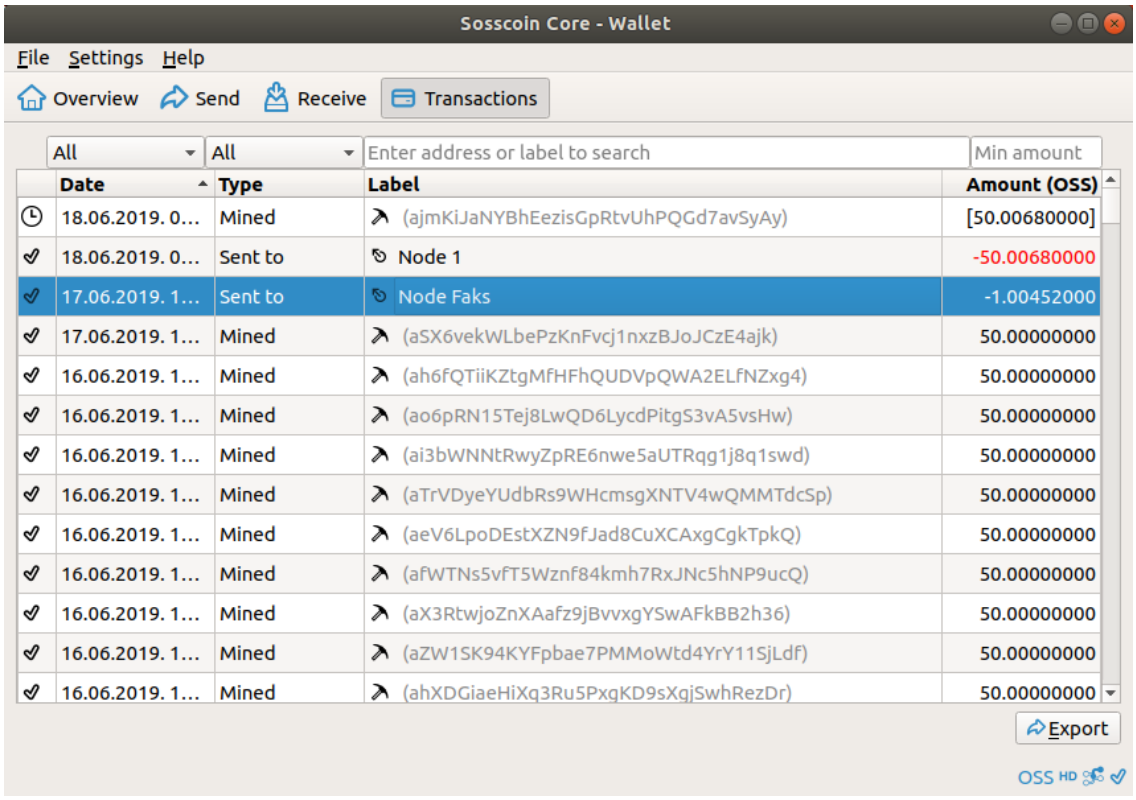
Ispis 43: Skripta za rudarenje

Kad se izrudari dovoljno blokova moguće je slati jedinice *sosscoin-a* drugim korisnicima. Sve funkcionalnosti se nalaze unutar digitalnog novčanika. Za slanje je jedino potrebno znati adresu na koju se šalje.

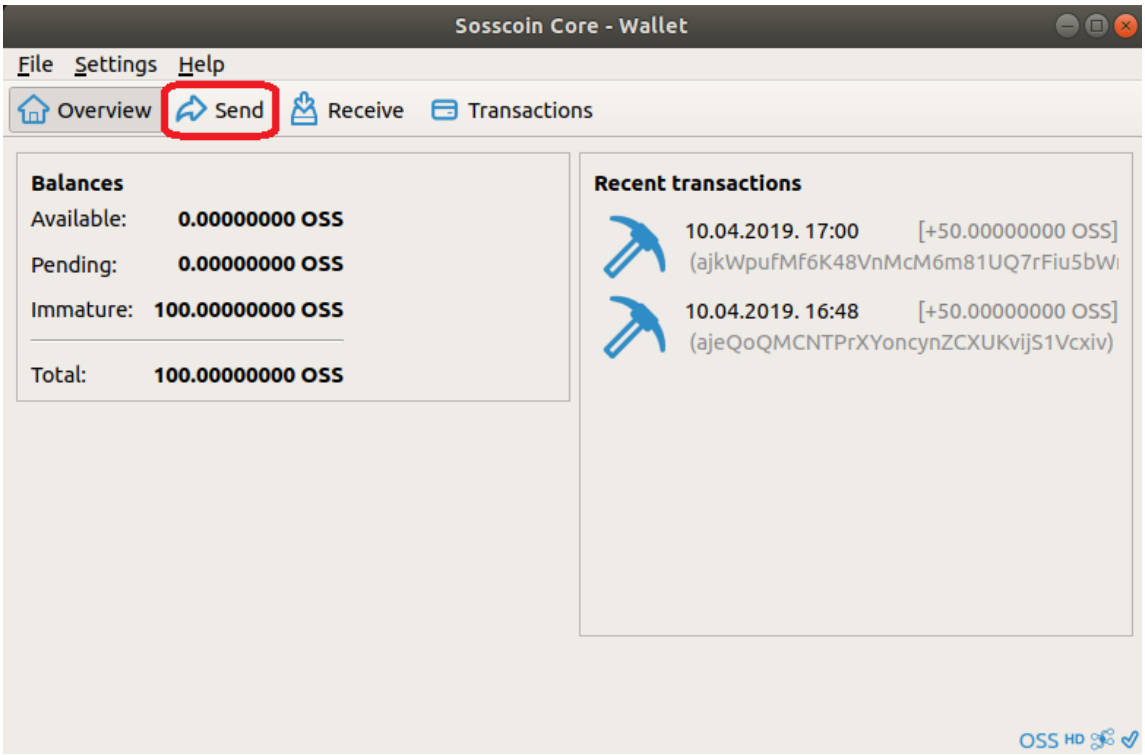


Slika 6: Sinkroniziranje čvorova

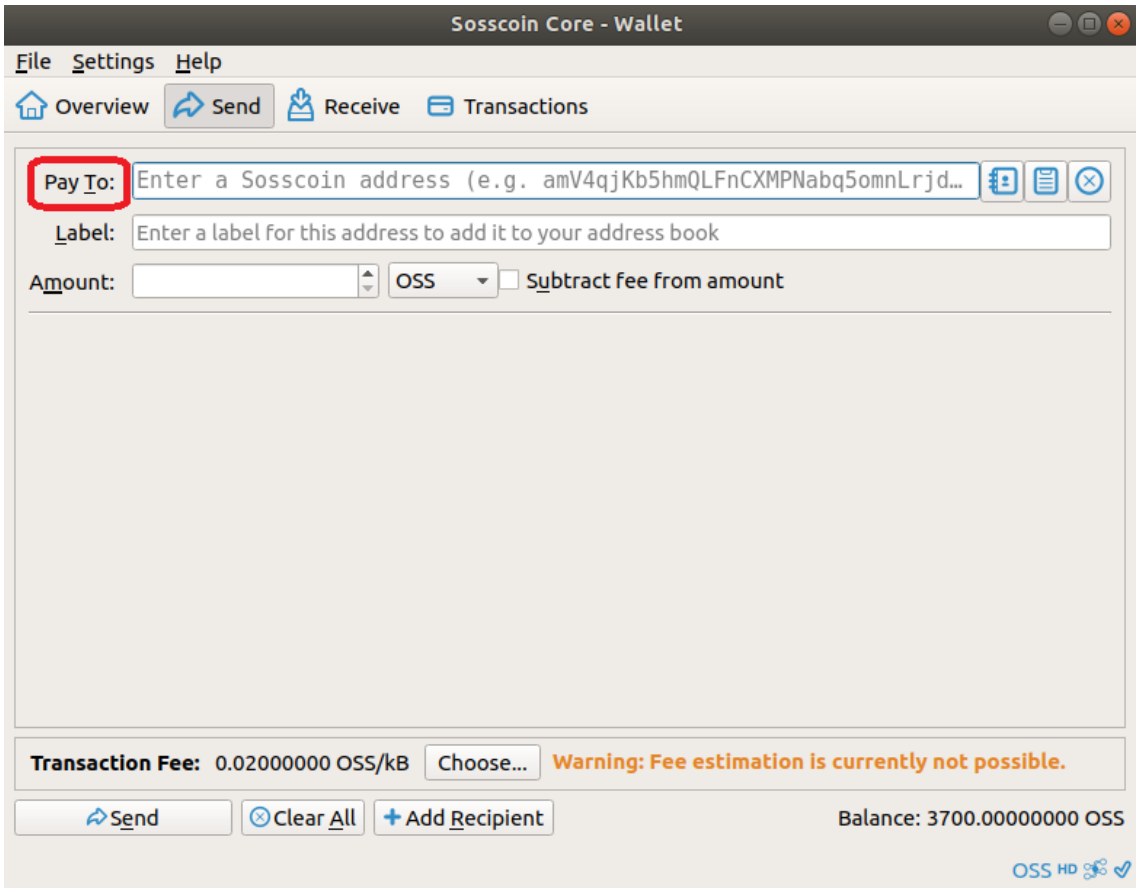
Slika 7: Znak za uspješnu sinkronizaciju čvorova



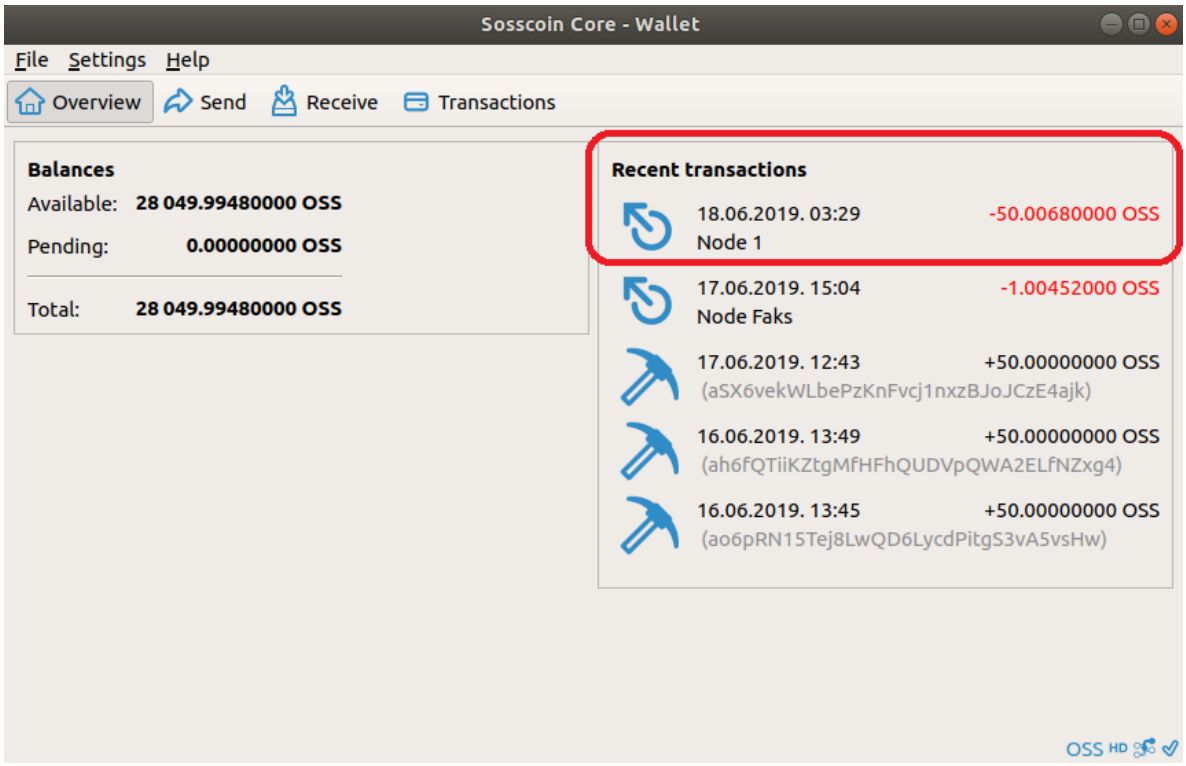
Slika 8: Lista prethodni transakcija povezanih s digitalnim novčanikom



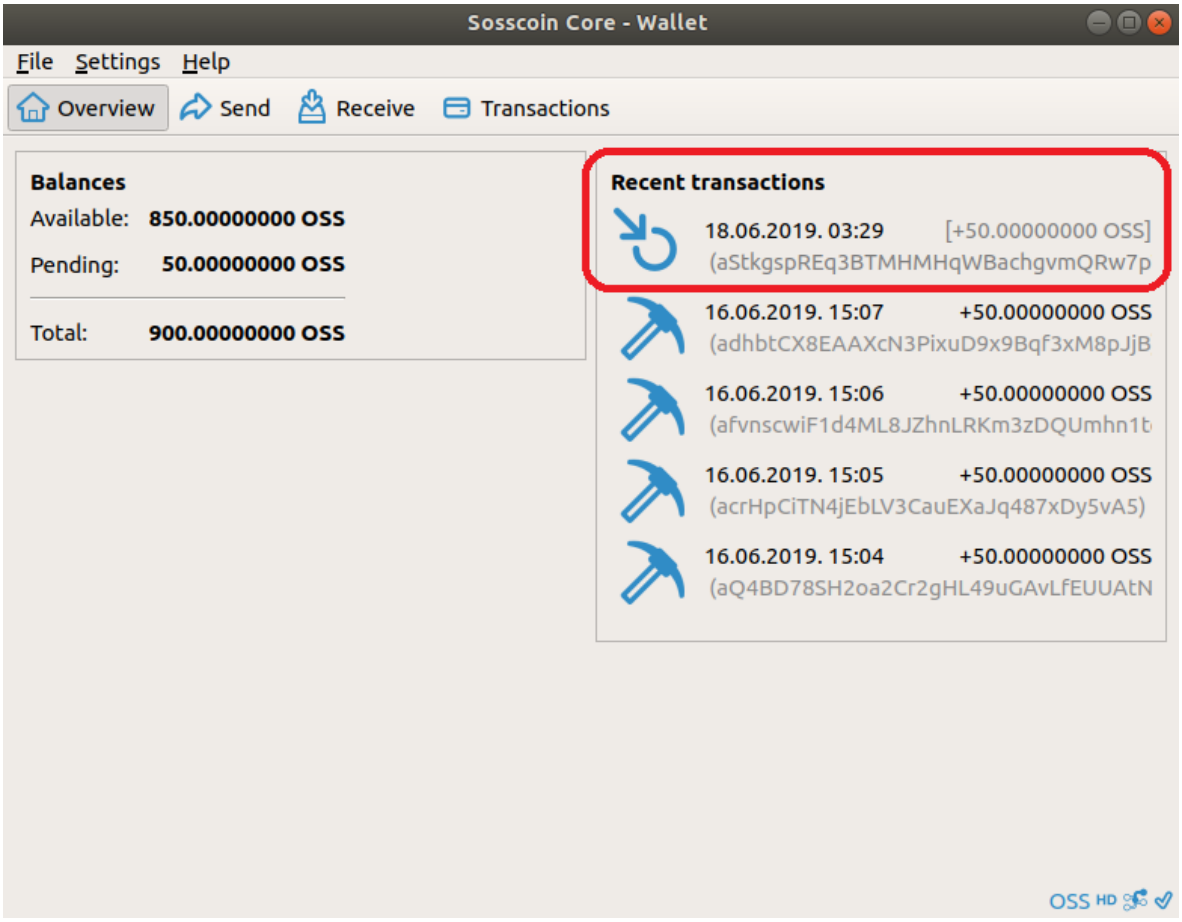
Slika 9: Slanje Sosscoin-a



Slika 10: Odabir primatelja



Slika 11: Poslane jedinice Sosscoin-a



Slika 12: Primljene jedinice Sosscoin-a

7. ZAKLJUČAK

U ovom završnom radu je opisan postupak izrade kriptovalute zasnovane na tehnologiji *blockchain*. Svrha izrade nove kriptovalute je upoznavanje korisnika s radom kriptovalute. Ne samo kako bi korisnik mogao napraviti svoju kriptovalutu, nego da može koristiti kriptovalutu s boljim razumijevanjem mehanizama koji omogućuju njen rad. Također, cilj je da se korisnici upoznaju s mogućnostima tehnologije *blockchain*. Ta tehnologija se može primjeniti u mnogim područjima svakodnevnog života. Jedan od primjera su zapisi vlasnika imovine. U lancu bi bile javno sadržane sve potrebne informacije tako da bi uvijek bilo jasno tko je trenutni vlasnik određene imovine te na koji način je postao vlasnik. Osim toga, ako bi obrt koristio *blockchain* za svoje proizvode, kupci tih proizvoda bi znali točno odakle dolaze proizvodi. Znali bi je li svojom kupnjom financiraju nešto što ne žele financirati. Izbori su još jedan od primjera gdje bi sigurnost i nemogućnost izmjene *blockchaina* moglo biti iskorišteno. To bi omogućilo pravednije i sigurnije izbore. To su samo neki od primjera. Tehnologija *blockchain* ima zaista velik opseg primjene.

Sosscoin je funkcionalna valuta, ali za njen uspjeh je potrebno još truda. Mjesta za napredak ima, u vidu izmjene unutar klasa `TestNet` i `RegTest` kako bi odgovarali ostatku koda i popraviti nekoliko *unit* testova. Treba postaviti nekoliko čvorova koji će biti uvijek uključeni kako bi se budući korisnici mogli na njih spojiti i preuzeti trenutni lanac. Ti čvorovi bi se također trebali zapisati unutar koda. Iako Sosscoin koristi *Scrypt*, za koji manje korisnika ima specijalizirane uređaje za rudarenje nego za Bitcoin ipak treba biti oprezan. Ako se unutar mreže koja ima veoma malo računala pojave takvi korisnici, mogu veoma podignuti težinu rudarenja bloka cijeloj zajednici. Sljedeće što je potrebno je *block explorer* i *mining pool*. Naposlijetku, kad se riješe svi ti problemi i pronađe dovoljno zajednica koja bi podržala Sosscoin može se krenuti na njegovo lansiranje u javnost. Dovoljno velika zainteresiranost bi otvorili mogućnost dolaska Sosscoin-a na listu razmjene. Nakon toga se može krenuti raditi na dodatnim mogućnostima kao što su mobilni novčanici. Naravno, za sve ovo je potrebna kontinuirana podrška.

Izvorni kod klijenta Sosscoin-a moguće je preuzeti na poveznici: https://github.com/zuka2512/Sosscoin

8. LITERATURA

1. Programiz, *Learn C++ Programming* https://www.programiz.com/cpp-programming (6.6.2019.)
2. GeeksforGeeks, *Python Programming Languange* https://www.geeksforgeeks.org/python-programming-language/ (6.6.2019.)
3. Rouse, M., *Script* https://whatis.techtarget.com/definition/script (6.6.2019.)
4. Ubuntu, https://ubuntu.com/ (6.6.2019.)
5. Beyer, S. *Blockchain Before Bitcoin: A History* https://blocktelegraph.io/blockchain-before-bitcoin-history/ (6.6.2019.)
6. Goyal, S. *The History of Blockchain Technology: Must Know Timeline* https://101blockchains.com/history-of-blockchain-timeline/ (7.6.2019.)
7. Rouse, M., *Consensus algorithm* https://whatis.techtarget.com/definition/consensus-algorithm (7.6.2019.)
8. Lisk, *Proof of Work* https://lisk.io/academy/blockchain-basics/how-does-blockchain-work/proof-of-work (7.6.2019.)
9. Crypto Beginners, *Blockchain explained in 1000 words* https://cryptobeginners.info/blog/what-is-blockchain/ (7.6.2019.)
10. Faife, C. *Bitcoin Hash Functions Explained* https://www.coindesk.com/bitcoin-hash-functions-explained (7.6.2019.)
11. CoinMarketCap https://coinmarketcap.com/all/views/all/ (7.6.2019.)
12. Agar, J. *Litecoin (LTC) vs. Bitcoin (BTC) – What's the Difference?* https://tokenmantra.com/litecoin-ltc-vs-bitcoin-btc-whats-the-difference/ (7.6.2019.)
13. Litecoin https://litecoin.org/ (7.6.2019.)
14. Fernando, *J. Bitcoin vs. Litecoin: What's the Difference?* https://www.investopedia.com/articles/investing/042015/bitcoin-vs-litecoin-whats-difference.asp (7.6.2019.)
15. Packages Debian, *Package: libssl-dev (1.1.1c-1 and others)* https://packages.debian.org/sid/libssl-dev (8.6.2019.)
16. Boost, *Welcome to boost.org!* https://www.boost.org/ (8.6.2019.)
17. Libevent, *libevent – an event notification library* https://libevent.org/ (8.6.2019.)
18. Free Software Directoy, *Miniupnpc* https://directory.fsf.org/wiki/Miniupnpc (8.6.2019.)

19. Oracle, *Oracle Berkeley DB* https://www.oracle.com/database/technologies/related/berkeleydb.html (8.6.2019.)
20. Dantoni, J., *Knowing the developers: an analysis of Litecoin Core* https://www.theblockcrypto.com/2019/06/13/knowing-the-developers-an-analysis-of-litecoin-core/ (9.6.2019.)