Blok Chain tehnologije u transakcijskim sustavima

Forko, Tomislav

Master's thesis / Specijalistički diplomski stručni

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Algebra University College / Visoko učilište Algebra**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:225:160281

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**



Repository / Repozitorij:

Algebra Univerity College - Repository of Algebra Univerity College





VISOKO UČILIŠTE ALGEBRA

DIPLOMSKI RAD

Blok Chain tehnologije u transakcijskim sustavima

Tomislav Forko

Zagreb, travanj 2019.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada.“

U Zagrebu, datum.

Predgovor

Zahvaljujem mentoru predavaču Visoke škole Zlatanu Moriću na savjetima i prijedlozima koji su mi pomogli u usavršavanju ovdje priloženog rada, kao i na njegovoj prijateljskoj suradnji i otvorenosti.

Želio bih zahvaliti i ostalim predavačima Algebre na pružanju visoke razine znanja koje mi je bilo izrazito korisno u praktičnom dijelu ovog rada.

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

Ideja ovog rada je približiti i objasniti rad sustava za upravljanje transakcijama. Porastom količine podataka u poslovanju dolaze sve veći zahtjevi za brzom dostupnosti tih podataka. Dostupnost podataka danas je vrlo upitna čak i cloud rješenja koja se nude u pozadini su opet centralizirani serveri. U radu u nastavku opisan je sustav koji ima naglasak na decentralizaciju, skalabilnost, brzinu pristupa, sigurnost. Mnoge tvrtke ali i državne institucije teže rješenju koji omogućava globalnu izmjenu transakcijskih podataka, a opet da se za njih može jamčiti sigurnost . Ovaj rad opisuje jedan od sustava globalne razmjene transakcijskih podataka i naziva se Blok chain. Blok chain je sustav koji može obraditi velike količine transakcijskih podataka, a pritom zadržati jednaku brzinu obrade istih. Svojom inovativnom infrastrukturom pruža brzinu, skalabilnost te visoku dostupnost. Ovaj rad detaljno razrađuje funkcionalnosti i rad takvog sustava kao i samo programiranje jednog djela takvog sustava.

Programski dio rada pisan je u python programskom jeziku te se koriste razni gotovi python moduli. Za prikaz korisničkog sučelja koriste se html, css, javascript biblioteka axios. Za komunikaciju s aplikacijom koristi se HTTPS protokol.

Ključne riječi: Blok chain, visoka dostupnost, brzina podataka, skalabilnost, python, javascript, decentralizacija, html , HTTPS, sigurnost.

Abstract

The idea behind this paper is to get closer look and explain the operation of the transaction management system. Increasing the amount of data in the business comes with all the higher requirements for the fast availability of this data. Availability of data today is very questionable even the clouds solutions that are offered in the background are again centralized servers. The paper describes a system that focuses on decentralization, scalability, speed of access, security. Many companies or government institutions are in need for a global data transactional system that can guarantee security. This paper describes one of the global transaction management system and it is called blockchain. Blockchain is a system that can process large amounts of transaction data while maintaining the same processing speed. With its innovative infrastructure, it provides speed, scalability and high availability. This paper describes in detail the functionality and operation of such a system as well as the programming such a system.

The program part of the work is written in python programming language and various ready-made python modules are used. To display the user interface code is written by help of various languages: html, css, javascript axios library. An HTTPS protocol is used for communication with the application.

Keywords: Block Chain, High Availability, Data Rate, Scalability, Python, Javascript, Decentralization, html, HTTPS, security

Sadržaj

1. UVOD 1

2. BLOK CHAIN TEHNOLOGIJA 2

2.1. Blok chain Terminologija 3

2.1.1. Transakcije, blokovi i rudarenje 4

2.1.2. Algoritmi Blok chain-a 5

2.2. Digitalni novčanik 7

2.3. Burze kripto valutama 15

3. BLOK CHAIN RUDARENJE 17

3.1. Praktični prikaz rudarenja 19

3.2. Objašnjenje pojmova rudarenja 21

4. PRAKTIČNA IZVEDBA SOFTVERSKOG TRANSAKCIJSKOG RJEŠENJA 24

4.1. Pojašnjenje i komentiranje programskog koda 24

4.2. Algoritmi i decentralizacijski sustavi 32

4.2.1. Konsenzus 35

5. KRIPTOGRAFIJA I KRIPTO VALUTE 36

5.1. Sigurnost kripto valuta 36

5.2. Verifikacija i validacija 37

5.3. Hashing algoritmi 40

6. TRENUTNA ISTRAŽIVANJA I ANALIZE TEHNOLOGIJE 42

6.1. Primjena blok chain-a 42

6.2. Usporedbe različitih tehnoloških projekata nastalih u blok chain-u 44

6.3. Blok chain specijalizacija 45

6.4. Trenutni ekonomski statusi postojećih projekata 46
Zaključak 50
Popis kratica 51
Popis slika 52
Popis kôdova 53
Literatura 55
Prilog 57

1. UVOD

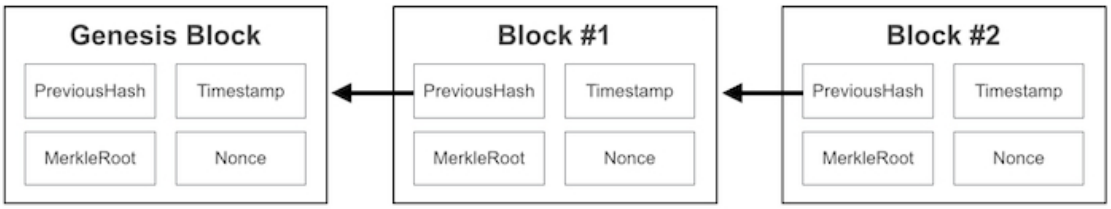
Tema ovog rada je opisati rad i funkcionalnosti blok chain sustava, a zatim pokazati njegovu praktičnu izvedbu u programskoj jeziku python. Rad je koncipiran na način da je u uvodnom dijelu rada objašnjena teorija blok chain rada i tehnički termini koje se kasnije koriste u programskom dijelu. Blok chain kao sustav dosta je kompleksan te su u ovom radu objašnjeni osnovni koncepti takvog sustava. Za demonstraciju rada sustava aplikacija je pisana da bude dostupna preko web preglednika odnosno sve radnje vezane uz blok chain pisan u ovom radu mogu se izvršiti kroz web sučelje. Blok chain je tehnologija koja je popularna unatrag desetak godina, a potječe iz Japana. Cilj ove tehnologije je tržištu ponuditi visoko dostupno, decentralizirano, brzo i efikasnom transakcijsko rješenje. Primjene ove tehnologije su razne, a opisane se u zadnjoj cjelini ovoga rada. Blok chain je sustav koji za dodavanje nove transakcije mora odraditi određene sigurnosne radnje i provjere kako bi sa sigurnošću mogao potvrditi transakciju. Sustav je dizajniran da ima višestruke sigurnosne provjere kako bi izbacio mogućnost manipulacije podataka od strane zlo namjernih korisnika. Kako bi dobili kompletan dojam u tehnologiji objašnjeno je i njegovo tržište odnosno burze digitalnih valuta, digitalni novčanici te razne statistike i procjene ove tehnologije i gdje bi mogla zauzeti svoje mjesto u tehničkoj primjeni. U radu je opisano više algoritama na temelju kojih može biti koncipiran blok chain sustav, a u praktičnom djelu rada se koristi samo jedan. Programski kod blok chain sustava napisanog u sklopu ovog rada objašnjen je kroz logičke cjeline rada. Počevši od grafičkog sučelja, objašnjenja glavnih funkcionalnosti te na samom kraju sigurnosnih provjera koji su bitne za sigurno slanje transakcija.

2. BLOK CHAIN TEHNOLOGIJA

Postoji više definicija blok chain-a ovisno s koje perspektive ga predstavljamo. Možemo ga gledati s poslovne perspektive te ga definirati u tom kontekstu, a kroz tehnološku perspektivu dobiva potpunu drugu definiciju. U ovom radu ću se dotaknuti obje definicije, a većinu rada ću posvetiti tehnološkoj definiciji blok chain-a. Gledajući s poslovne strane blok chain je platforma preko koje možemo razmjenjivati vrijednosti, kroz transakcije bez potrebe provjere treće strane, arbitratora ili tijelu kojem se vjeruje. Primjeri upotrebe blok chain tehnologije recimo mogu biti zamjena bankarskog sustava za razmjenu valuta bez potrebe provjere transakcije od strane banke. Možemo ga koristiti kao sustav koji sprema transakcije u kojima se nalaze podaci o ubranom voću kada je ono ubrano, kojim prijevoznim sredstvom je dostavljeno, kada je stiglo u trgovinu te tako pratiti povijest ubranog voća. Sve su to transakcije koje možemo smjestiti u blok chain sustav. Mogućnosti primjene ove tehnologije u raznim segmentima svakodnevnog života i stvari koje koristimo teško je nabrojati. Tehnološki gledano blok chain je u svojoj srži distribuirani mrežni lanac blokova, kriptografski siguran, ne promjenjive prirode u smislu izmjene podataka koji su jednom zapisani. Novi podaci odnosno transakcija koju nadodajemo u lanac blokova moraju biti potvrđeni od konsenzusa odnosno jednoglasne odluke svih servera(nodova) koji sudjeluju u blok cahin-u. Bok chain možemo gledati kao sloj distribuirane *peer to peer* mreže koji je dostupan preko Interneta. Više o tehnološkoj definiciji objašnjeno je u ostatku rada . Blok chain je osmišljen od osobe ili grupe ljudi koja se naziva Satoshi Nakamoto 2008 godine. Identitet osobe ili grupe ljudi koja se krije iza toga naziva je i dalje nepoznanica. Napravljen je da bi se koristio kao javni transakcijski sustav digitalne valute pod nazivom bitcoin. Bitcoin je prva digitalna valuta koja je riješila problem dvostruke potrošnje poznat pod nazivom *double-spending*. Ovo je zapravo problem koji se pojavljuje samo kod digitalnog novca, a misli se na zloupotrebu istog digitalnog tokena za kreiranje novog iznosa istog novca koji prije nije postojao što potencijalno vodi inflaciji. Za razliku od fizičkog novca digitalni novac se sastoji od digitalnog podatka koji može biti dupliciran, falsificiran. Blok chain koristi se kriptografskim tehnikama, algoritmima kako bi se upravo to spriječilo.

2.1. Blok chain Terminologija

Blok chain možemo gledati kao podatkovnu strukturu koja nosi naziv glavna knjiga (engl. *General ledger*) koja je u svojoj cjelini povezani lanac podataka koja umjesto uobičajenih pokazivača koristi hash pokazivače. Detaljnije o hashing funkcijama opisano je u poglavlju Hashing algoritmi, a za sada nam je dovoljno da je hash šifrirani rezultat fiksnih podataka koji osigurava njihovu nepromjenjivost i neporecivost(Andreas M.Antonopoulos, 2017 [2]). Blok chain se sastoji od niza blokova koji su međusobno povezani na način da svaki sljedeći blok koji se dodaje u sustav bude svjestan onog prethodnog. Ovakvu funkcionalnost mu upravo omogućuje hash vrijednost, po toj vrijednosti je svaki blok svjestan svoga prethodnika. Za svaki blok se generira jedinstvena hash vrijednost. Podaci blok chain-a se kao što i sam naziv govori spremaju u blokove. Blokovi sadrže skup podatka koji su zapravo jedna ili više transakcija. Osim transakcija blok može, a i najčešće sadrži identifikator, hash vrijednost prethodnog bloka, vremensku oznaku (engl. *Timestamp*), dokaz(engl. *Proof*) ili (engl. *Nonce*). Ovisno o poslovnom modelu blok može sadržavati i dodatne podatke. Slika 2.1 Redoslijed blokova prikazuje grafički prikaz blokova.



Slika 2.1 Redoslijed blokova

Osim transakcija svi ostali podaci u bloku se jednim nazivom nazivaju meta podaci (engl. *Metadata*) te ih pod tim nazivom susrećemo u službenoj literaturi. Meta podaci koji su nabrojani prethodno zapravo omogućuju i obavljaju funkcionalnosti u softverskom kodu blok chain-a. Dolazimo do gotovo najvažnije stavke blok chain sustava, a to su transakcije. Transakcije također ovisno poslovnom modelu i namjeni mogu sadržavati drugačije i dodatne vrijednosti. Ono što je najbitnije što gotovo svaka transakcija ima i od čega se sastoji su polja adresa pošiljatelja, adresa primatelja i vrijednost ili količina koja se šalje. Pod adresom se ne misli na adresu prebivališta već nešto što se u kriptografiji naziva javni ključ. Adresa je jedinstveni identifikator pošiljatelja ili primatelja , javni ključ koji se sastoji od niza znakova i brojki koji su jedinstveni. Javni ključ je javno dostupan i vidljiv od strane svih sudionika blok chain-a. Svaki korisnik može iz generirati više javnih ključeva na koji

mogu zaprimati transakcije. Javni ključ ujedno predstavlja i digitalni novčanik preko kojeg šaljemo i primamo transakcije. Postoji sigurnosni mehanizam koji nam kod kreacije digitalnog novčanika i zaprimanja javnog ključa ujedno generira i privatni ključ. Privatni ključ znamo samo mi osobno i trebamo ga držati u tajnosti. Privatni ključ služi za otključavanje digitalnog novčanika koji sadrži javni ključ.

2.1.1. Transakcije, blokovi i rudarenje

Izvorni ili prvi blok u blok chain-u se naziva *genessis* blok. *Genesis* blok je inicijaliziran od strane programera koji su pisali softver. Sadrži ručno popunjene podatke: adresu primatelja, indeks, potpis te ne sadrži vrijednost hasa prethodnog bloka jer je on prvi u nizu blok chain-a. Već sljedeći blok poprima hash izvornog *genessis* bloka. Glavna knjiga (engl. *General ledger*) je kao što smo već spomenuli lanac blokova. Ona se može koristiti za provjeru bilo koje napravljene transakcije između adresa, javnih ključeva primatelja i pošiljatelja. Moramo razlikovati transakcije koje su već odrađene i otvorene transakcije koje čekaju na obradu i zapisivanje u novi blok koji će se dodati na kraj blok chain-a. Proces obrade otvorenih transakcija nazivamo rudarenje(eng. *mining*). Rudari su zapravo računala koja pokreću komad softvera koristeći svoj GPU[1], CPU[2] kako bi odradile određene matematičke operacije nad otvorenim transakcijama i iz tih podataka iz generirale novi blok to jest hash tog bloka. Svaki proizvedeni hash bloka je unikatan. Ako se promjeni samo jedan znak u podacima bloka hash se kompletno mijenja. Time osiguravamo da podacima bloka nije manipulirano. U procesu rudarenja kreacije hasha osim transakcija uzima se i hash prethodnog bloka. Zbog toga što je hash svakog bloka napravljen pomoću hasha bloka prije njega, taj blok postaje digitalni oblik pečata. Ova funkcionalnost nam garantira da je svaki blok legitiman odnosno ništa u njemu nije mijenjano. Pokušaj krivotvorenja transakcije mijenjajući podatke u bloku koji je već sudionik blok chain-a mijenja hash tog bloka. Kada bi se ponovno provjeravao hash tog bloka on bi bio drugačiji od onog hasha koji je prethodno pohranjen u sljedeći blok do njega. Zapravo taj blok bi onda bio krivotvoren. Zbog toga što je hash svakog bloka korišten za stvaranje sljedećeg bloka u lancu, mijenjanje jednog bloka

[1] GPU - Graphics processing unit

[2] CPU - Central processing unit

bi izazvalo promjenu sljedećeg bloka. Što bi značilo da bi promjenom bilo kojeg bloka izazvali lančanu reakciju koja bi se protezala do kraja lanca.

2.1.2. Algoritmi Blok chain-a

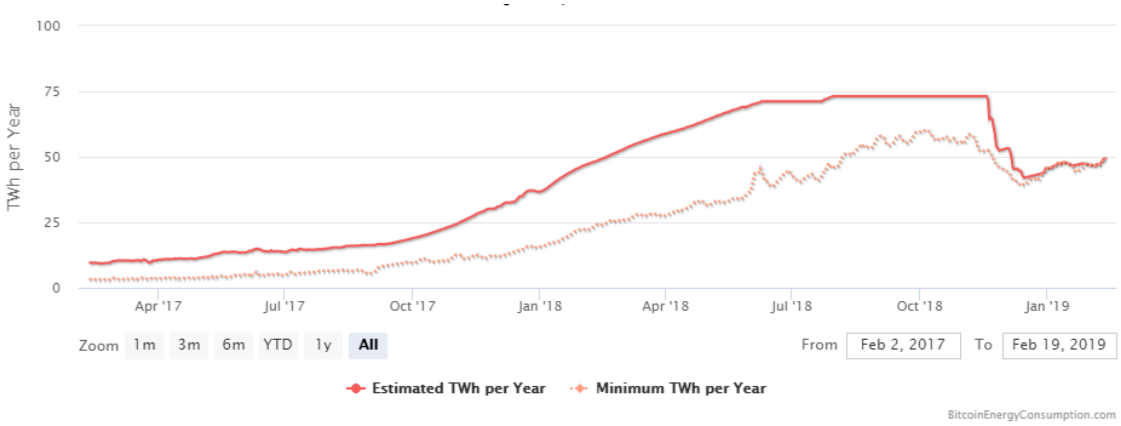
Jednom kada rudari stvore hash bloka i on je dodan u lanac blokova taj blok je zapečaćen. Kao nagradu za kreiranje hash vrijednosti njegovi rudari su nagrađeni određenom količinom novčića (engl. *coins*), digitalnom valutom. Lanac blokova tada je ažuriran po svim serverima (eng. *node*) koji su sudionici blok chain-a. Na taj način funkcionira obrada transakcija u blok chain-u. Pošto se hash iz podataka u bloku kreira dovoljno brzo to bi značilo da bi svi digitalni novčići bili iskopani vrlo brzo jer su računala brza u tom postupku. Zato se u blok chain-u namjerno otežava postupak obrade hasa za nove transakcije. Cijeli koncept obrade hasha nazivamo dokaz o radu (engl. *Proof of work*). Na njemu je bazirano funkcioniranje blok chain-a. Otežavanje izrade hasa radi se način da blok chain neće prihvatiti bilo kakav hash već traži da taj hash ima određenu strukturu. Uvjet blok chain-a može biti da hash mora imati određeni broj nula na početku. Što više nula na početku tražimo to je teže izračunati hash. Kako bi svaki put iz generirali drugačiji hash dok ne dobijemo onaj odgovarajući s određenim brojem nula na početku, moramo mijenjati neki podatak unutar transakcijskog bloka. Kao što je ranije spomenuto transakcije i ostale podatke nam nije dozvoljeno mijenjati unutar bloka, zato u bloku postoji podatak kojeg nazivamo dokaz (engl. *proof*) ili (engl. *nonce*) u različitoj literaturi. Dokaz (engl. *proof*) možemo vidjeti na slici Slika 2.1 Redoslijed blokova. Unutar bloka *nonce* nam je dozvoljeno mijenjati i pomoću njega dobivamo različite rezultate hasha. Kada je novi blok dodan u lanac blokova tada se ta informacija za sinkronizira po svim serverima, *nodovima*. Ova radnja u blok chain-u nosi naziv konsenzus. Postizanje konsenzusa će biti detaljnije objašnjeno kroz kod, odnosno programski, praktični dio rada. Trenutno je dovoljno znati da se konsenzus postiže međusobnim vjerovanjem *nodova* blok chain-a bez potrebe da se upliću treće strane. Algoritam dokaz o radu, skraćeno POW (engl. *Proof of work*) je korišten u programskog kodu blok chain-a koji je napisan u sklopu ovog rada. Takav algoritam se koristi kod najvećih i najpopularnijih blok chain-ova danas.

2.1.2.1 Dokaz o udjelu (eng. proof of stake)

U odnosu na prethodno spomenuti algoritam dokaz rada ovaj algoritam blok chain-a za postizanje konsenzusa koristi udio novčića od korisnika. To znači da ne postoji mehanizam obrade otvorenih transakcija, kreiranjem tražene hash vrijednosti. Već se validnost transakcija ostvaruje tako da korisnici koji imaju veći udio novčića mogu uložiti dio novčića u sustav na temelju kojih se odlučuje valjanost transakcija. Za taj proces su naravno nagrađeni kao u slučaju rudarenja kod POW algoritma. Takve korisnike blockchaina nazivamo rudarima. U POS algoritmu rudari novog bloka se nazivaju kovači (engl. *forger*). Kod izrade novog bloka kovač se odabire nasumice ovisno o dva parametra. Prvi parametar je koliko je taj korisnik uložio novčića , a drugi parametar koji se preispituje nosi naziv *random*. Što više korisnik novčića uloži to je veća vjerojatnost da bude izabran kao kovač novog bloka. Ako to gledamo na taj način najimućniji korisnici će biti uvijek odabrani za izradu novih blokova i shodno tome nagrađeni. Što bi značilo da najbogatiji korisnici postaju još bogatiji. Pošto se prvi parametar ne vodi logikom, uvodi se drugi parametar naziva *random*(hrv. Nasumičan odabir). Za parametar random postoji više metoda primjene, a najpoznatije su *Randomised Block Selection* i *Coin Age Selection.* Ključ drugog elementa je da omogući izradu bloka i korisnicima koji imaju manji ulog novčića u sustavu. Taj postupak polu slučajnog odabira kod metode *Randomised Block Selection* znači da se kovač (engl. *Forger*) (Max Thake, 2018 [8]) odabire na temelju kombinacije najmanje hash vrijednosti i najvećeg uloga. Kod *Coin Age Selection* metode kovač se odabire na temelju toga koliko je njegov ulog star, odnosno vremenski najduže postoji u blok chain-u.

2.1.2.2 Prednosti i mane različitih algoritama.

Najbitnija razlika između algoritama spomenutih u prošlim naslovima je da se kod POW algoritma broj novokreiranih novčića, tokena u sustavu stalno povećava. Iz tog razloga kod blok chain-a koji koristi POS algoritam količina tokena u sustavu definirana je odmah na početku lansiranja takvog blok chain projekta. Kako POW mehanizam za proces validacije novih transakcija koristi hardware, odnosno CPU,GPU zahtijeva i puno električne energije. Istraživanja su pokazala da je prosječna potrošnja energije u svijetu samo za rudarenje Bitcoina 49.39 TWh. Grafički prikaz potrošnje tokom prethodno dvije godine može vidjeti na slici Slika 2.2 Bitcoin potrošnja energije.



[3]

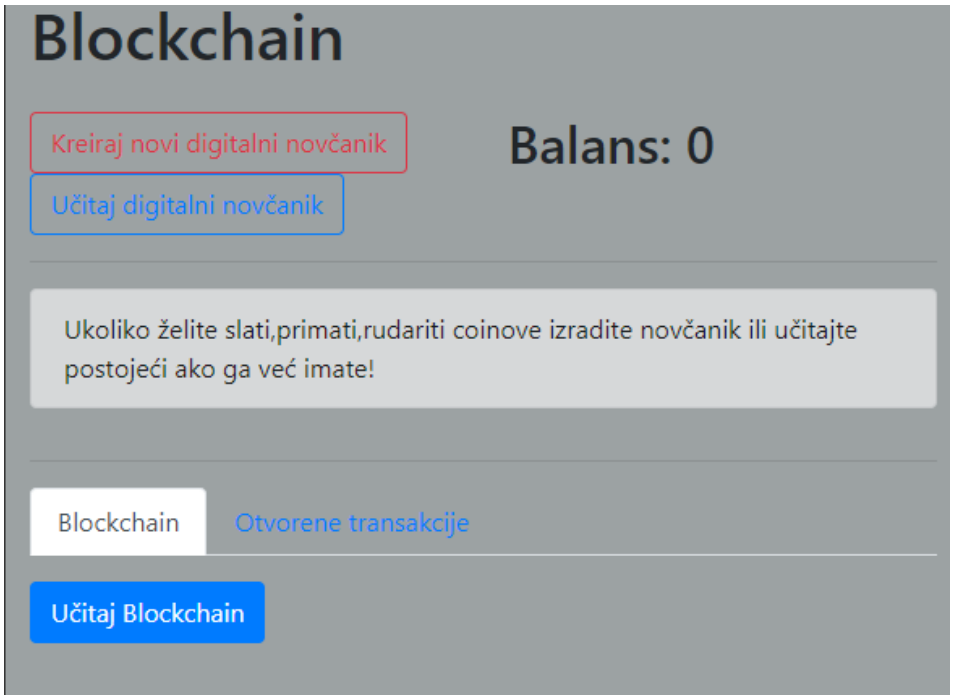
Slika 2.2 Bitcoin potrošnja energije

Prednosti POS konsenzus algoritma su što je energetski učinkovitiji. Sigurniji u smislu da napadači moraju imati vlastiti ulog u takvom blok chain-u da bi pokušali manipulirati transakcijskim podacima. Za usporedbu napadač kod POW konsenzus algoritma ne gubi hardver kao u slučaju novaca kod POS algoritma.

2.2. Digitalni novčanik

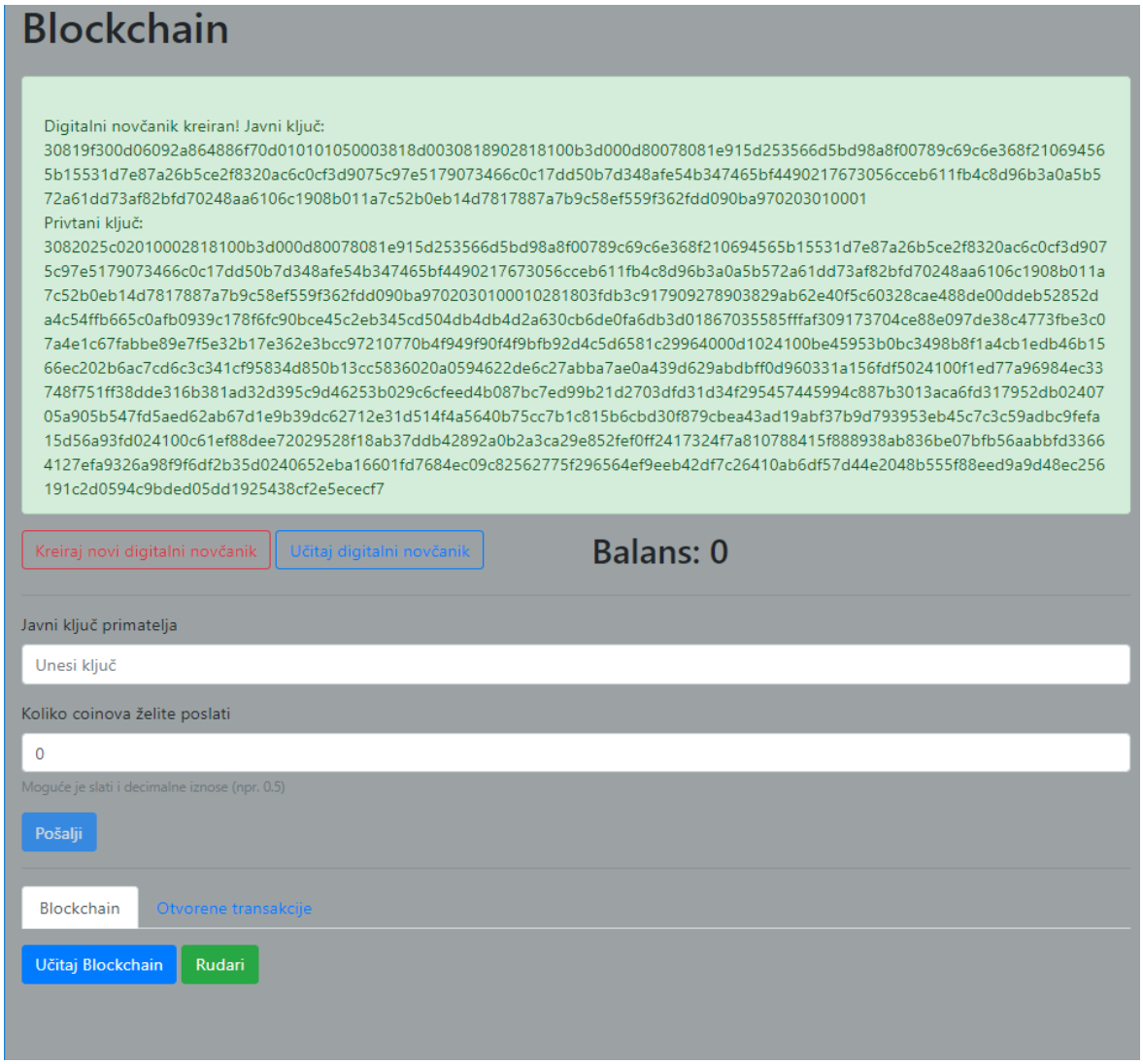
Kako bi slali transakcije potreban nam je digitalni novčanik, u sklopu ovog rada digitalni novčanik osim mogućnosti slanja transakcija ima i druge funkcije. U praksi digitalni novčanici služe samo za slanje, primanje, uvid u stanje tokena te povijest transakcija. Prikaz digitalnog novčanika u web pregledniku napisanom u ovom radu je malo drugačiji. On sadrži i dodatne opcije kako bi se mogle demonstrirati funkcije i rad cijelog blok chain-a. Opcije digitalnog novčanika programiranog u ovom radu opisane su u nastavku. Na sučelju prikazanom na slici Slika 2.3 Web sučelje digitalnog novčanika 1 imamo mogućnosti učitavanja postojećeg digitalnog novčanika ukoliko ga već imamo ili kreiranje novog. Možemo slati transakcije tako da upišemo javni ključ primatelja te upisati količinu i kliknuti na pošalji. Na vrhu sučelja prikazana je količina dostupnih tokena, a dnu ekrana imamo uvid u sanje cijelog blok chain-a te u stanje otvorenih transakcija koje čekaju na obradu.

[3] Slika 2.2 Bitcoin potrošnja energije– link izvora slike: https://digiconomist.net/bitcoin-energy-consumption, ožujak 2019.



Slika 2.3 Web sučelje digitalnog novčanika 1 „vlastiti rad autora“

Rudarenje otvorenih transakcija izvršava se kroz opciju rudari na dnu ekrana vidljivom na slici Slika 2.4 Web sučelje digitalnog novčanika 2. Kada učitamo ili kreiramo digitalni novčanik na vrhu ekrana prikazani su nam privatni i javni ključ.



Slika 2.4 Web sučelje digitalnog novčanika 2 „vlastiti rad autora“

Sve akcije koje izvršavamo u digitalnom novčaniku u web pregledniku zapravo su HTTP metode POST[4], GET[5]. Kada kliknemo na gumb u pregledniku u pozadini se odrađuje javascript koji poziva python flask API[6] zapisan u datoteci node.py koji tada dohvaća same klase i njihove funkcije kroz datoteke opisane u cjelini 4.1 Pojašnjenje i komentiranje programskog koda. Opis API-a ovog blok chaina slijedi u nastavku. Za izradu API-a koristi se gotov modul Flask. Pomoću njega definirane su rute, url [7] dostupan preko web

[4] POST – metoda http protokola koja se poziva za slanje podataka
[5] GET - metoda http protokola koja se poziva za dohvaćanje podataka
[6] API - Aplikacijsko programsko sučelje (engl. application programming interface, API)
[7] URL je akronim za eng. pojam Uniform Resource Locator, u prijevodu - ujednačeni ili usklađeni lokator sadržaja ili resursa

preglednika. Korisnička aplikacija sluša na 0.0.0.0 ip adresi[8] portu[9] 5000 kao što je definirano u kodu Kôd 2.1 Definiranje ip adrese i porta aplikacije. Ovo je ujedno i glavni (engl. Main) dio aplikacije koju pokrećemo naredbom `python node.py`.

```
app = Flask(__name__)
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Kôd 2.1 Definiranje ip adrese i porta aplikacije

Početna ruta / pokazuje digitalni novčanik na način da uzima statički file node.html u kojem je raspisano korisničko sučelje, koristi html metodu GET kod Kôd 2.2 Definiranje HTTP rute / za GET metodu.

```
@app.route('/', methods=['GET'])
def get_ui():
    return send_from_directory('ui', 'node.html')
```

Kôd 2.2 Definiranje HTTP rute / za GET metodu.

Ukoliko u korisničkom sučelju pritisnemo na „učitaj digitalni novčanik“ poziva se ruta `/wallet` koja pomoću html GET metode učitava postojeći novčanik ukoliko on postoji iz wallet.txt datoteke. Učitavanje wallet.txt datoteke objašnjeno je u cjelini 4.1. Ukoliko datoteka `wallet.py` postoji program učitava javni i privatni ključ u suprotnom vraća poruku `Loading keys failed` te HTTP kod `500`[10] vidljivo u kodu Kôd 2.3 Definiranje HTTP rute /wallet za GET metodu.

```
@app.route('/wallet', methods=['GET'])
def load_keys():
    if wallet.load_keys():
        global blockchain
        blockchain = Blockchain(wallet.public_key)
        response = {
            'public_key': wallet.public_key,
            'private_key': wallet.private_key,
            'funds': blockchain.get_balance()
```

[8] IP adresa - (engl. Internet Protocol address) jest jedinstveni broj koji se dodjeljuje svakom uređaju u svrhu komunikacije preko mrežnog segmenta.
[9] Port - U računarskoj mreži port je softverski zadat kanal kojim komuniciraju aplikacije putem računarskih mreža
[10] 500 – http status kod koji predstavlja grešku servera

```
        }
        return jsonify(response), 201[11]
    else:
        response = {
            'message': 'Loading keys failed.'
        }
      return jsonify(response), 500
```

Kôd 2.3 Definiranje HTTP rute /wallet za GET metodu.

Ukoliko još nemamo digitalni novčanik kreiramo ga u korisničkom sučelju pritiskom na tipku „kreiraj digitalni novčanik“. U pozadini se odrađuje javascript koji poziva HTTP POST metodu kod Kôd 2.4 Definiranje HTTP rute /wallet za POST metodu. Zapravo se poziva funkcija `create_keys()` iz klase `wallet` također opisana u cjelini 4.1 koja generira ključeve. Zatim se izvršava jednak dio koda kao i kod *wallet GET* metode.

```
@app.route('/wallet', methods=['POST'])
def create_keys():
    wallet.create_keys()
    if wallet.save_keys():
        global blockchain
        blockchain = Blockchain(wallet.public_key)
        response = {
            'public_key': wallet.public_key,
            'private_key': wallet.private_key,
            'funds': blockchain.get_balance()
        }
        return jsonify(response), 201
    else:
        response = {
            'message': 'Saving keys failed.'
        }
        return jsonify(response), 500
```

Kôd 2.4 Definiranje HTTP rute /wallet za POST metodu.

Stanje svojih digitalnih valuta provjeravamo tako da upišemo *url* adresu `0.0.0.0:localhost/balance`. Izvršava se dio koda na slici koji pomoću HTTP GET metode sprema stanje u varijablu `balance` pozivajući funkciju `get_balance()` iz

[11] 201 – http kod koji označava uspješnu promjenu

`blockchian` klase te ovisno o rezultatu ispisuje poruke `'Fetched balance successfully'` te raspoloživo stanje u novčaniku ili `'Loading balance failed'` ukoliko ne postoji digitalni novčanik sa javnim ključem. Opisani proces vidljiv je u kodu Kôd 2.5 Definiranje HTTP rute /balance za GET metodu.

```
@app.route('/balance', methods=['GET'])
def get_balance():
    balance = blockchain.get_balance()
    if balance != None:
        response = {
            'message': 'Fetched balance successfully.',
            'funds': balance
        }
        return jsonify(response), 500
    else:
        response = {
            'message': 'Loading balance failed.',
            'wallet_set_up': wallet.public_key != None
        }
        return jsonify(response), 500
```

Kôd 2.5 Definiranje HTTP rute /balance za GET metodu

Jednom kad učitamo javni i privatni ključ iz digitalnog novčanika otvara na se novi izbornik u kojem možemo slati transakcije. Pritiskom na „pošalji“ zapravo se poziva http POST metoda kod Kôd 2.6 Definiranje HTTP rute `/transaction` za POST metodu. Programski kod prije slanja transakcije provjerava uvjete da li je digitalni novčanik učitan, ukoliko nije vraća http kod 400[12]. Zatim provjerava da li su svi podaci ispravno uneseni u za to predviđena polja. Ukoliko nisu isto odgovara http kodom 400. Nakon što poprimi ispravno popunjene podatke sprema ih u varijable, poziva se na funkciju `wallet.sign_transaction` kako bi kreirao potpis te transakcije. Navedena funkcija opisana je u cjelini 5.2. Verifikacija i validacija. Nakon toga poziva se funkcija `blokchain.add_transaction` opisana u cjelini 4.1.Pojašnjenje i komentiranje programskog koda koja posprema transakciju u otvorene transakcije i čeka na proces rudarenja.

```
@app.route('/transaction', methods=['POST'])
```

[12] 400 – http kod za pogrešan zahtjev

```
def add_transaction():
    if wallet.public_key == None:
        response ={
            'message': 'No wallet set up.'
        }
        return jsonify(response), 400
    values = request.get_json()
    if not values:
        response ={
            'message': 'No data found.'
        }
        return jsonify(response), 400
    required_fields = ['recipient', 'amount']
    if not all(field in values for field in required_fields):
        response ={
            'message': 'Required data is missing.'
        }
        return jsonify(response), 400
    recipient = values['recipient']
    amount = values['amount']
    signature = wallet.sign_transaction(wallet.public_key,
recipient, amount)
    success = blockchain.add_transaction(recipient,
wallet.public_key, signature, amount)
    if success:
        response ={
            'message': 'Successfuly added transaction.',
            'transaction': {
                'sender': wallet.public_key,
                'recipient': recipient,
                'amount': amount,
                'signature': signature
            },
            'funds': blockchain.get_balance()
        }
        return jsonify(response), 201
    else:
        response ={
            'message': 'Creating a transaction failed.'
        }
```

```
return jsonify(response), 500
```

Kôd 2.6 Definiranje HTTP rute /transaction za POST metodu.

Kao što je rečeno u početku ove cjeline digitalni novčanik u ovom blok chain-u osim standardnih opcija za slanje, primanje i uvid stanja digitalnih valuta ima i dodatne opcije: učitavanje cijelog blok chain-a i otvorenih transakcija te rudarenje. Blok chain se učitava pomoću programskog koda Kôd 2.7 Definiranje HTTP rute /chain za GET metodu. Pritiskom na '`učitaj blok chain`' u korisničkom sučelju poziva se HTTP GET metoda. U varijablu `chain_snapshot` dohvaćamo trenutne blok chain podatke. Radimo konverziju iz obične blok chain liste [13]u *dictionary*[14] kako bi pomoću json[15] formata mogli vratiti rezultat u web preglednik. Uz same podatke vraćamo i kod http kod 200 što znači da je dohvaćanje podataka uspješno.

```
app.route('/chain', methods=['GET'])
def get_chain():
    chain_snapshot = blockchain.chain
    dict_chain = [block.__dict__.copy() for block in
chain_snapshot]
    for dict_block in dict_chain:
        dict_block['transactions'] = [tx.__dict__ for tx in
dict_block['transactions']]
return jsonify(dict_chain), 200[16]
```

Kôd 2.7 Definiranje HTTP rute /chain za GET metodu.

Dohvaćanje otvorenih transakcija koje čekaju obradu dobijemo tako da u korisničkom sučelju pritisnemo na „otvorene transakcije“. Poziva se http GET metoda vidljiva u kodu Kôd 2.8 Definiranje HTTP rute /transaction za GET metodu , zatim funkcija `get_open_transactions` također radi konverziju otvorenih transakcija iz liste u *dictionary* kako bi ih mogla prikazati u json formatu.

```
@app.route('/transactions', methods=['GET'])
def get_open_transactions():
```

[13] Liste - su skupovi podataka varijabla koji predstavljaju jednu cjelinu. Elementi u listi odvojeni su zarezom
[14] Dictionary-kolekcija podataka u kojoj su podaci indeksirani, promjenjivi i nesortirani.
[15] JSON (engl. JavaScript Object Notation) tekstualni otvoreni standard dizajniran za čitljivu razmijenu podataka
[16] 200 – http kod koji predstavlja uspješno dohvaćanje podataka

```
    transactions = blockchain.get_open_transactions()
    dict_transactions = [tx.__dict__ for tx in transactions]
    return jsonify(dict_transactions), 200
```

Kôd 2.8 Definiranje HTTP rute /transaction za GET metodu

2.3. Burze kripto valutama

Burze i mjenjačnice su mjesta većinom platforme gdje korisnici mogu kupovati, prodavati i trgovati kripto valutama. Kao i s tradicionalnim valutama moguće je trgovati bitcoinom, ethereumom, litecoinom i raznim ostalim kripto valutama dostupnim na tržištu. To nam omogućuju kripto burze i mjenjačnice. Na burzi kripto valuta možemo zamijeniti pravi novac za digitalne valute odnosno tokene i obratno. Neki od naj popularnijih platforma, burza kripto valuta su Binance, Poloniex, Bitfinex, Kraken, Coindesk, Gdax. Koliko takve burze zarađuju dnevno nije poznato niti je to javno dostupna informacija. Nagađanja i procjene raznih analitičkih kuća te popularne stranice za poslovne vijesti bloomberg[17] koja donosi članak dostupan na linku[18] objavljuju da je najpopularnija kripto platforma Binance. Binance na dnevnoj razini ostvari prihod od 3.4 milijuna američkih dolara. Sve burze kripto valutama podliježu zakonima svoje države. U nekim zemljama trgovanje kripto valutama je ilegalno dok druge koriste priliku pa kripto valute smatraju dobrima i naplaćuju porez. Primjer države koja je poduprla i prijateljski se okrenula ovoj tehnologiji je Singapur. Karakteristike transakcija na kripto burzama su sljedeće:

- Nepovratne ili ne poništive - Nakon što potvrdimo transakciju za slanje akcija slanja ne može biti poništena.
- Anonimne – Niti su transakcije niti računi povezani sa stvarnim identitetom osobe, transakcije se primaju u digitalne novčanike te se samo vidi javna adresa pošiljatelja. Moguće je pratiti povijest transakcije no nije vidljiv uvid u stvarni identitet pošiljatelja.
- Brze i globalne – Transakcije se odrađuju gotovo odmah, potvrđuju se u nekoliko sekunda ovisno o tehnologiji u pozadini. Pod globalnom karakteristikom misli se da

[17] www.bloomberg.com – popularna novinarska kuća ,web stranica koja donosi vijesti u poslovnom svijetu
[18]https://www.bloomberg.com/news/articles/2018-03-05/crypto-exchanges-raking-in-billions-emerge-as-kings-of-coins

transakciju možemo poslati bilo gdje, bilo kad. Blok chain tehnologija nije ovisna o fizičkoj lokaciji.

- Sigurne – Sredstva su zaštićena u obliku javnog ključa i privatnog ključa pomoću kriptografije.
- Ne iziskuju dozvolu – Ne trebamo tražiti dozvolu državnih institucija da bi slali transakcije. Preuzmemo softver ili preko web preglednika ovisno o aplikaciji izradimo digitalni novčanik i šaljemo sredstva.

Rano financiranje projekata je osim na kripto burzama moguće i kroz takozvani ICO (engl. *initial coin offering*). ICO ima dosta sličnosti s financiranjem tradicionalnih *startup* projekata. U početnu ranu faza blok chain projekta može se uložiti tako da kupujemo digitalne valute za vrijeme trajanja ICO-a. Sredinom ljeta 25. srpnja, američki SEC (U.S. Securities and Exchange Commission), glavni regulator američkih financijskih tržišta (burze vrijednosnica), objavio je priopćenje o novom financijskom instrumentu (*initial coin offering*) karakterističnom za okruženje u kojem je novčana jedinica neka digitalna valuta. ICO bi mogli usporediti sa IPO (engl. *initial public offering*), javnom ponudom dionica. Riječ ICO se puno koristi i u marketingu pa su tako nastale sve popularnije marketinške konferencije koje se održavaju u ime novih blok chain projekata. Takve konferencije su prilika za okupljanje programera, inženjera te raspravljanju o tehnološkom napretku financija zvanim još fintech[19]. Ključno je za ICO da se tim putem skupljaju sredstva za projekte i tvrtke dostupna putem kupnje digitalnim valutama.

[19]Finetech- vrsta primjene digitalne tehnologije u financijama

3. BLOK CHAIN RUDARENJE

Što je rudarenje i POW algoritam već je objašnjeno u prethodnoj cjelini 2.1 Blok chain Terminologija no kako ono izgleda u praksi slijedi u nastavku. Rudariti možemo na dva načina samostalno ili u mining pool-u[20]. Cilj obje metode je da budemo nagrađeni iznosom tokena čije transakcije obrađujemo kako bi ostvarili dobit. Rudarenje kroz mining pool je pristup u kojem zajedničkom procesorskom snagom svih sudionika, klijenata pridonosimo generiranju novih blokova. Nakon što je blok iz generiran nagrađeni smo određenim iznosom tokena ovisno o pridonesenoj procesorskoj snazi u procesu izrade bloka. Mining polovi su stoga način da rudari tokena udruže svoje resurse i podjele zajedničku hardversku moć kako bi rješavali izračunavanje hasheva brže od ostalih. Mining polovi koordiniraju računala odnosno takozvane rudare (engl. workers) kako bi se hash brže izračunao. Dogovor u mining polu je da se nagrada dijeli s obzirom na doprinos hardverske moći. Na taj način dobijemo samo djelić nagrade na primjer 0.001 Bitcoina, ali je mogućnost nagrade puno veća nego da pokušamo rudariti samostalno gdje je šansa dobitka mala. Mining pool u koordinaciji rudara (engl. workers) svrhu ravnopravne raspodjele nagrade prolazi kroz sljedeće četiri osnovne funkcije:

- Uzima stanje izračunatih hasheva članova pool-a.
- Traži da li postoji uspješno odrađen blok
- Snima stanje koliko je određeni sudionik pridonio u izračunavanju hasha
- Proporcionalno dodjeljuje blok nagrade rudarima

Kako bi se mogao izračunati uloženi rad svih rudara te kako bi na temelju toga svi sudionici mogli biti nagrađeni uvodi se komponenta zvana share. Kako ne možemo nagraditi samo onog rudara koji je pronašao odgovarajući hash za blok jer bi to bio ekvivalent samostalnome rudarenju uvodi se share. Mining poolovi uvode share-ove kako bi izračunali uloženi rad rudara (engl. *worker*). *Share*[21] je jedinica za mjerenje uloženog napora u rudarenju dobitnog bloka to jest odgovarajućeg hasha. Jednom kad se pronađe pravi hash tada taj blok ulazi u stvarnu mrežu blok chain-a, a rudari su nagrađeni na temelju odrađenih share-ova. *Share* se

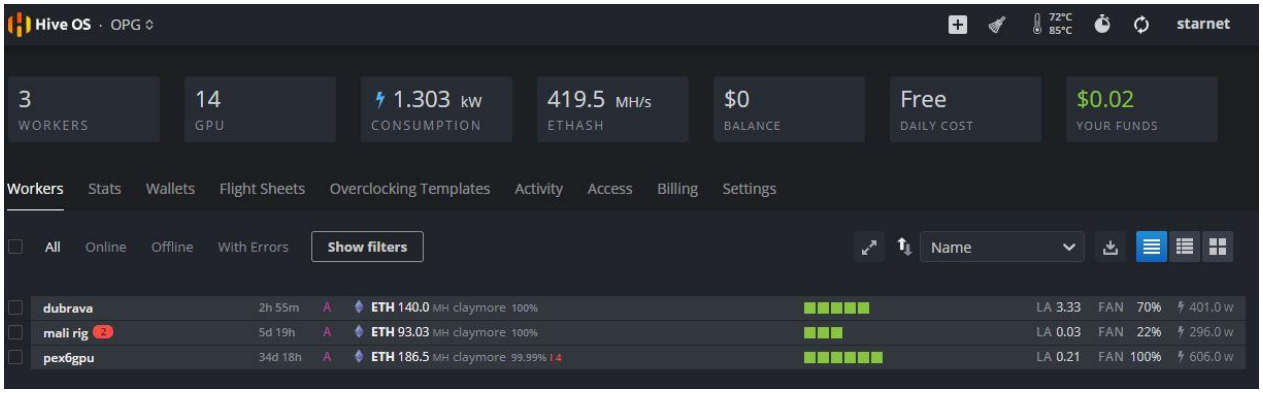
[20] Mining pool- udruživanje resursa od strane rudara koji zajedničkom procesorskom snagom sudjeluju u obradi tekućih transakcija da bi kreirali novi blok.
[21] Share - jedinica za mjerenje uloženog napora u rudarenju dobitnog bloka

izračunava kao doprinijeta količina posla proporcionalna ukupnoj količini posla koja je bila potrebna da se izračuna pravi hash. Pojednostavljeno recimo da je za iz rudariti novi blok potrebno 1000MH[22], ako smo mi doprinijeli rudarenju bloka sa 25MH na 2 sekunde uložili smo ukupno 50 MH znači da dio naše nagrade iznosi (50 / 1,000 == 0.05 == 5%).

Modela isplate rudara ima više , a najpoznatiji su:

- PPS (engl. Pay Per Share)
- SMPPS (engl. Shared Maximum Pay Per Share)
- PPLNS (engl. Pay Per Last N Shares)

Popularni operacijski sustavi za rudarenje etherenum tokena i ostalih su hiveos, ethos. Oni nam omogućuju alate za monitoring i upravljanje hardverom koji sudjeluje u rudarenju. Skupine hardverskih komponenti koje sudjeluju u rudarenju popularno se nazivaju *mining* farme. Kako bi se olakšalo njihovo upravljanje u svrhu praćenja potrošnje energije, temperature komponenti, performansi nastaju operacijski sustavi namijenjeni samo za tu svrhu. Slika 3.1 Prikaz početne stranice Hiveos operativnog sistema prikazuje pregled raspoloživog hardvera.

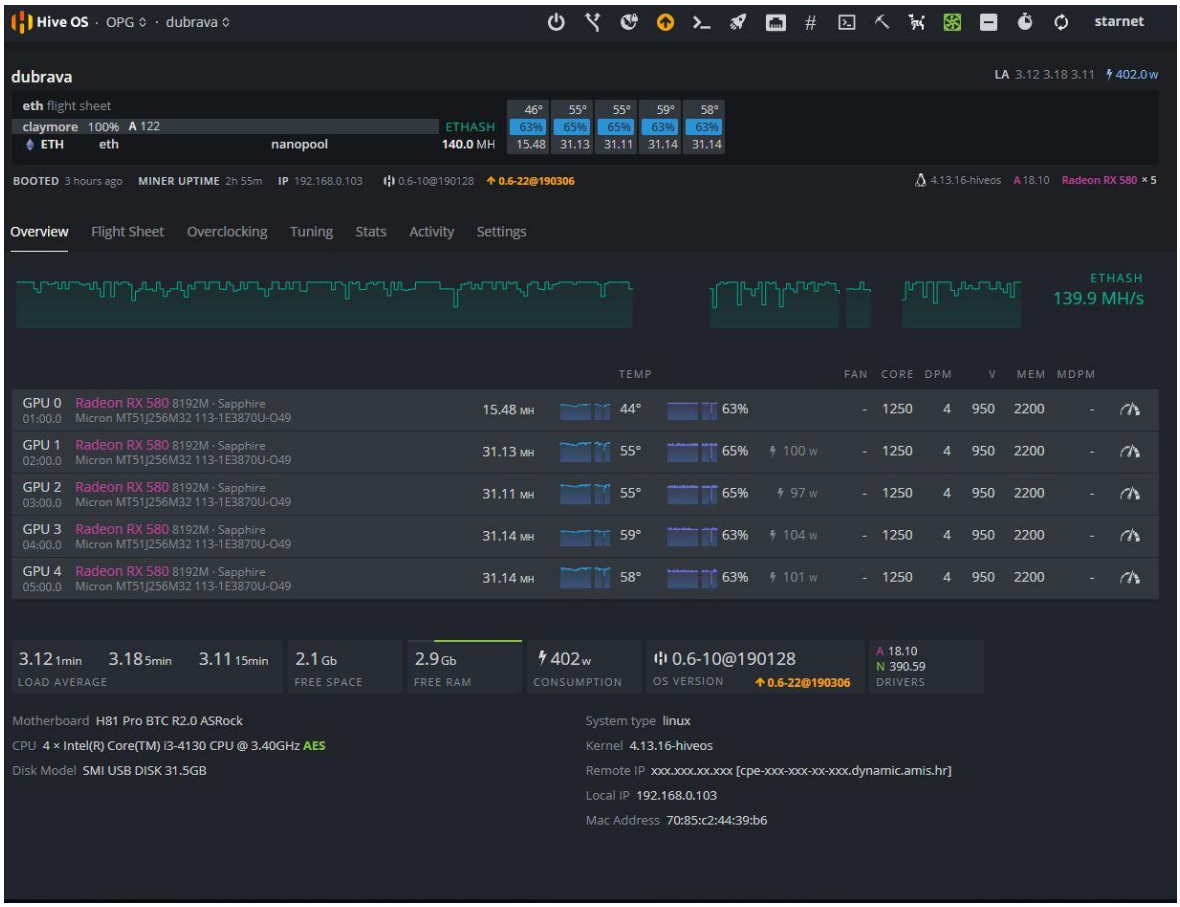


Slika 3.1 Prikaz početne stranice Hiveos operativnog sistema[23]

Koliko detaljno možemo pratiti opterećenje pojedine komponente prikazuje Slika 3.2 Prikaz opterećenja grafičkih kartica u hiveos sustavu. na kojoj možemo vidjeti hashing snagu pojedine grafičke kartice, njezinu minimalnu i maksimalnu temperaturu rada te parametre kojima upravljamo brzinu hashing operacija. Programiranje u svrhu olakšavanja procesa

[22] MG - MH/s (ili megahasha po sekundi) ili pak GH/s (gigahasha po sekundi). Jedinica kojom se mjeri sposobnost učinkovitosti rudarenja . Što je hash bolji i veći, to korisnik može učinkovitije i brže izračunati hash bloka.
[23] Slika je "vlastiti rad autora" uzeta iz Hive OS aplikacije.

rudarenja postalo je vrlo popularno pa tako da poznatoj programerskoj stranici github možemo pronaći razni softver koji nam pripomaže u samom procesu. Primjer je open-callisto-pool [24] projekt koji je nastao kao otvoreni projekt koji nudi instalaciju vlastitog mining pool-a.



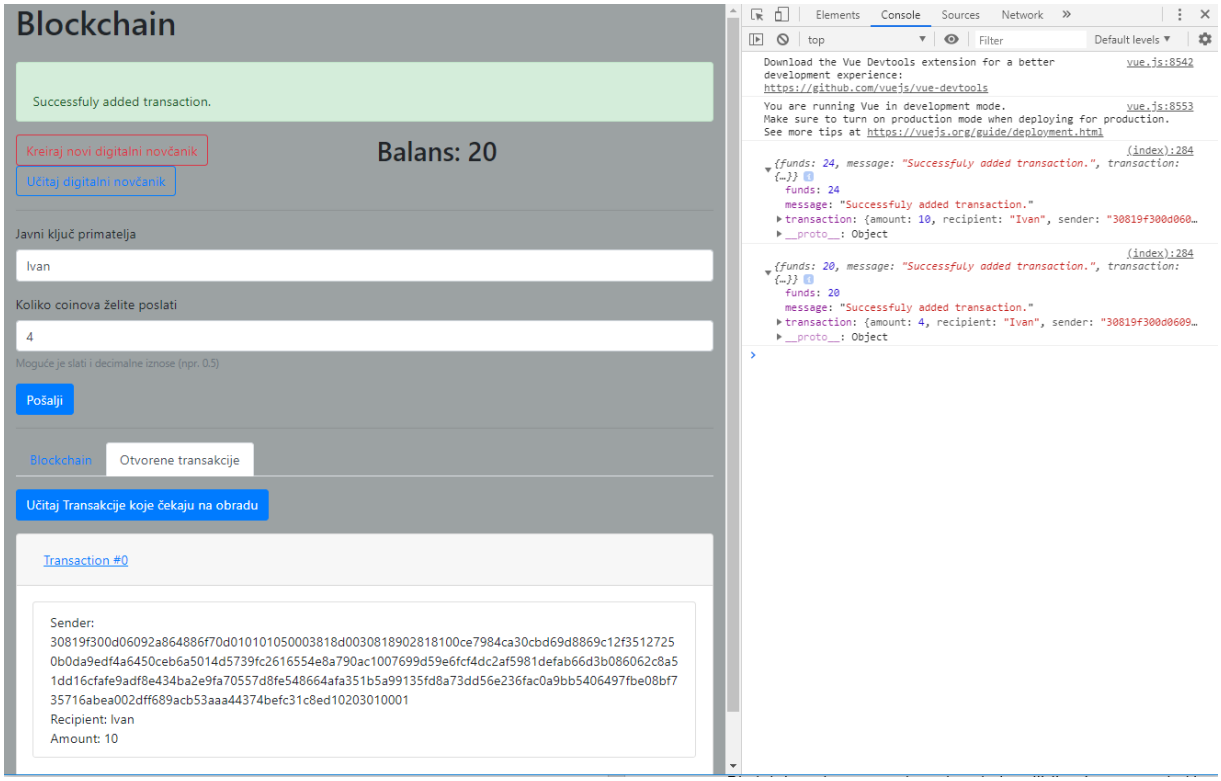
Slika 3.2 Prikaz opterećenja grafičkih kartica u hiveos sustavu.[25]

3.1. Praktični prikaz rudarenja

U praktičnom djelu ovog rada rudarenje je zamišljeno na način da je korisniku dostupno korisničko sučelje kroz koje možemo izvršiti radnje nad otvorenim transakcijama te za taj proces biti nagrađeni. Nagrada za obradu otvorenih transakcija je 10 jedinica odnosno tokena. Na slici Slika 3.3 Slanje transakcija putem digitalnog novčanika. na dnu ekrana imamo opcije rudari i otvorene transakcije. Poslane su dvije transakcije iznosa 10 i 4 prema

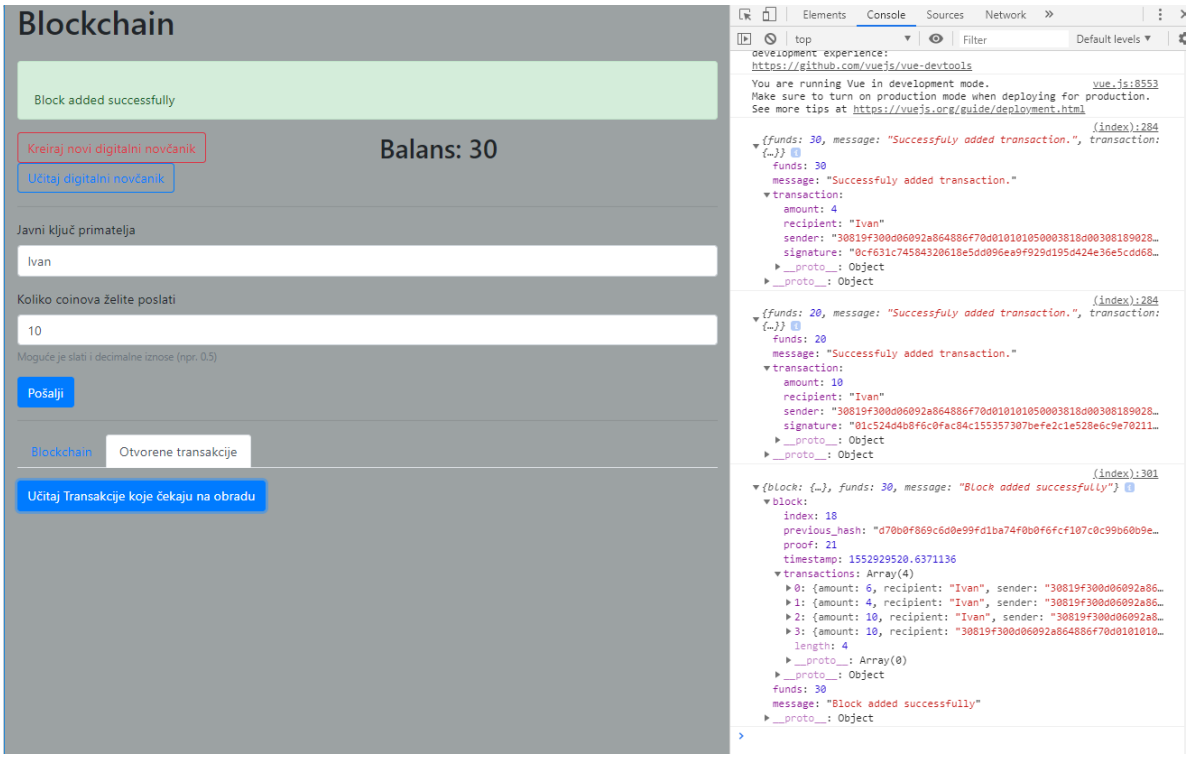
[24] EthPool – blok chain mining pool projekt – link izvora: https://github.com/ethpool-update-project/open-callisto-pool
[25] Slika je "vlastiti rad autora" uzeta iz Hive OS aplikacije.

korisniku Ivan te u popisu otvorenih transakcija vidimo da još nisu obrađene što znači da još nisu zapisane u blok chain mrežu.



Slika 3.3 Slanje transakcija putem digitalnog novčanika. “vlastiti rad autora”

Kako bi zapis bi trajno zapisan u blok chain potrebno je izračunati hash za sljedeći blok. Ukoliko kliknemo na gumb rudari na dnu ekrana za otvorene transakcije bit će izračunata hash vrijednost novoga bloka te će one biti spremljene u novo kreirani blok. Navedeni postupak vidljiv je na slici .Kao što možemo vidjeti nagrađeni smo iznosom od 10 tokena te je kreiran novi blok s indexom 18 koji nastao na temelju parametra: index:18, previous_hash: "d70b0f869c6d0e99fd1ba74f0b0f6fcf107c0c99b60b9e69ec11d1fa6caa7169", proof: 21, timestamp: 1552929520.6371136, transactions. Nakon kreacije novog bloka popis otvorenih transakcija ponovno je prazan.



Slika 3.4 Prikaz dodavanja novog bloka. “vlastiti rad autora”

3.2. Objašnjenje pojmova rudarenja

Programski kod koji obavlja funkciju rudarenja u prethodnoj cjelini 3.1 opisan je u nastavku. Klikom na gumb rudari zapravo pozivamo http POST metodu kojom zapisujemo u blok chain kod Kôd 3.1 Definiranje HTTP rute /mine za POST metodu. Zapravo u varijablu `blok` spremamo rezultat koji smo dobili obavljanjem funkcije `mine_block` iz `blockchain` klase slika . Ovisno o rezultatu dobivenom iz `mine_block` funkcije vraća HTTP kod 201[26] blok je uspješno dodan ili kod 500[27] dodavanje bloka je neuspješno.

```
@app.route('/mine', methods=['POST'])
def mine():
    block = blockchain.mine_block()
    if block != None:
        dict_block = block.__dict__.copy()
```

[26] HTTP 201 - zahtjev klijenta je uspješno zaprimljen i kreiran je novi resurs.
[27] HTTP 500 - zahtjev nije proveden zbog greške na serveru.

```
            dict_block['transactions'] = [tx.__dict__ for tx in
 dict_block['transactions']]
            response = {
                'message': 'Block added successfully',
                'block': dict_block,
                'funds': blockchain.get_balance()
            }
            return jsonify(response), 201
        else:
            response = {
                'message': 'Adding a block failed',
                'wallet_set_up': wallet.public_key != None
            }
            return jsonify(response), 500
```

Kôd 3.1 Definiranje HTTP rute /mine za POST metodu.

Funkcija `def mine_block` kao ulazni parametar uzima cijeli blok chain, u varijablu `last_block` posprema zadnji blok u blok chainu te zatim u varijablu `hashed_block` posprema izračunatu hash vrijednost tog bloka. Funkcija `hash_block` opisana je u poglavlju 5.4 Hasing algoritmi. Zatim se kroz funkciju `proof_of_work()` opisanoj u cjelini 4.2. Algoritmi i decentralizacijski sustavi dobiva dokaz (engl. Proof). U varijablu `copied_transactions` unosimo sve trenutno otvorene transakcije te provjeravamo autentičnost potpisa za sve otvorene transakcije. Ukoliko je provjera prošla u redu u listu otvorenih transakcija dodajemo i nagradnu transakciju koja je zapravo isplata za rudarenje. Nakon toga kreiramo novi blok sa vrijednostima: hash prethodnog bloka, otvorene transakcije + nagradna transakcija, dokaz (engl. proof) te broj koji se izračunava ugrađenom funkcijom `len(self.chain)` koja zapravo označava id[28] bloka. Dodajemo blok u lanac blokova , ispraznimo otvorene transakcije te pospremimo stanje blok chain-a sa novim blokom u datoteku `bockchain.txt`. Funkcija je prikazana u kodu Kôd 3.2 Funkcija za kreaciju blokova.

```
def mine_block(self):
        last_block = self.chain[-1]
        hashed_block = hash_block(last_block)
```

[28] ID – (engl. identification number) jedinstveni je broj koji služi za identifikaciju.

```
        proof = self.proof_of_work()
        reward_transaction = Transaction('MINING',
      self.hosting_node, '', MINING_REWARD)
        copied_transactions = self.__open_transactions[:]
        for tx in copied_transactions:
            if not Wallet.verify_transaction(tx):
                return None
        copied_transactions.append(reward_transaction)
        block = Block(len(self.chain), hashed_block,
copied_transactions, proof)
        self.chain.append(block)
        self.__open_transactions = []
        self.save_data()
        return block
```

Kôd 3.2 Funkcija za kreaciju blokova

4. PRAKTIČNA IZVEDBA SOFTVERSKOG TRANSAKCIJSKOG RJEŠENJA

Praktični dio rada je podijeljen u više cjelina te je baziran isključivo na programskom rješenju koje je opisano kroz naslove koji slijede u nastavku. Programski kod je pisan u objektno orijentiranom obliku s ciljem pojednostavljenja dijelova koda. Za izvedbu programskog koda odlučio sam se na python[29] radi jednostavnosti, preglednosti, a i iskustva s navedenim programskom jeziku u svakodnevnom poslovnom okruženju. Za glavni dio projekta, sveukupnu logiku i mehanizme koristi se python verzije 3.7. Http, css, java-script te python-flask koriste se za korisničko sučelje dostupno preko internet browsera http protokolom. Za mehanizme pohrane i čitanja podataka o blok chain-u koristi se json. Uz sam python koristim se i modulima koje python pruža kao što su python-flask za API (rest api) koji služi za interakciju s korisničkim unosima preko grafičko sučelja dostupnog http-om. Python moduli openssl te pycrypto korsite se uglavnom za kriptografiju, sigurnost blockchaina, pa tako recimo iz modula pycrypto koristim se hashing algoritmima SHA256 za potpisivanje blokova i transakcija. Modul pycrypto u sebi ima ugrađene funkcionalnosti za generiranje privatnog i javnog ključa koja su izrazito važni kod digitalnih novčanika kao adrese pošiljatelja i primatelja.

4.1. Pojašnjenje i komentiranje programskog koda

Blockchain python programska struktura koda podijeljena je u programske klase. Glavna klasa u ovom radu naziva se blockchain.py i služi za primarne operacije nad blok chain-om a to su:

- Dodavanje novih transakcija u blockchain sustav
- Rudarenje novih blokova, dohvaćanje otvorenih transakcija
- Dohvaćanje balansa
- Pospremanje i učitavanje blockchain podataka
- Dodavanje, brisanje, učitavanje bok chain nodova u sustav

[29] Python- programski jezik opće namjene

Blockchain.py programski dio koda za navedene operacije koristi sljedeće klase bez koji ne bi mogao funkcionirati.

`Block` je klasa koja definira osnovne karakteristike bloka, a sadrži indeks, prethodnu hash vrijednost, vremensku oznaku, transakcije i dokaz koji je zapravo ranije spomenuti *nonce* koji mijenjamo u svrhu drugačijeg ishoda hasha odnosno dok ne udovoljimo POW algoritmu da bi kreirali novi blok nad otvorenim transakcijama. Klasa je prikazana u kodu Kôd 4.1 Definiranje klase Block.

```
class Block:
      def __init__(self, index, previous_hash, transactions, proof,
      time=time()):
            self.index = index
            self.previous_hash = previous_hash
            self.timestamp = time
            self.transactions = transactions
            self.proof = proof
```

Kôd 4.1 Definiranje klase Block

Prethodni isječak koda 4.1 definira klasu naziva `block` koja sadrži funkciju u kojoj su kreirani ranije navedeni objekti. `Transaction` je klasa koja sadrži objekte vezane uz transakcije blockchaina. U njoj se definira pošiljatelj, primatelj, iznos koji šaljemo te popis koji je generiran pomoću `def_sign_transaction` funkcije . Kod vezan uz generaciju potpisa objašnjen je u dijelu 5.2 Verifikacija i validacija blockchaina. Definicija klase slijedi u prikazu koda Kôd 4.2 Definiranje klase Transaction

```
class Transaction:
      def __init__(self, sender, recipient, signature, amount):
            self.sender = sender
            self.recipient = recipient
            self.amount = amount
            self.signature = signature
```

Kôd 4.2 Definiranje klase Transaction

Klasa `Verification` sadrži funkcije koje se odnose na sigurnost blockchain sustava a sadrži funkcije:

- `valid_proof`

- `verify_chain`
- `verify_transaction`
- `verify_transactions`

Funkcije i njihove namjena su detaljno objašnjenje u cjelini 5.2 Verifikacija i validacija. Klasa `wallet` definira funkcije vezane za digitalni novčanik, a sadrži funkcionalnosti za izradu privatnih i javnih ključeva, pospremanje i učitavanje ključeva iz datoteke. U klasi wallet definiran je konstruktor[30] `def__init__(self)` u kojem su definirane dvije važne varijable `private_key`, `public_key`. Funkcija `create_keys(self)` generira ključeve i dodjeljuje ih prethodno spomenutim varijablama pomoću funkcije `generate_keys` koja je također dio `wallet` klase. Kada se ključevi iz generiraju otvara se datoteka wallet.txt u statusu za pisanje te se zapisuju prvo javni ključ zatim u novom redu privatni ključ. Ukoliko datoteka prethodno ne bi postojala dobili bi grešku `'Saving wallet failed'`. Opisana klasa prikazana je u prikazu koda Kôd 4.3 Definiranje klase Wallet.

```
class Wallet:
    def __init__(self):
        self.private_key = None
        self.public_key = None

    def create_keys(self):
        private_key, public_key = self.generate_keys()
        self.private_key = private_key
        self.public_key = public_key
        try:
            with open('wallet.txt', mode='w') as f:
                f.write(public_key)
                f.write('\n')
                f.write(private_key)
        except (IOError, IndexError):
            print('Saving wallet failed')
```

Kôd 4.3 Definiranje klase Wallet

[30] Konstruktor -Većina klasa sadrži metodu pod nazivom __init__. Donje crte pokazuju da je to specijalna vrsta metode. Ova metoda se zove konstruktor , i automatski se poziva kada netko kreira novi objekt iz vaše klase. Konstruktor se obično koristi za dodjelu početnih vrijednosti varijablama iz klase

Sličnu zadaću obavlja `load_keys` funkcija otvara datoteku wallet.py u statusu za čitanje te definiramo da se datoteka čita redak po redak kako bi pročitali javni i privatni ključ te ih dodijelili varijablama `public_key` te `private_key`.

```
def load_keys(self):
    try:
        with open('wallet.txt', mode='r') as f:
            keys = f.readlines()
            public_key = keys[0][:-1]
            private_key = keys[1]
            self.public_key = public_key
            self.private_key = private_key
        return True
    except (IOError, IndexError):
        print('Loading wallet failed')
        return False
```

Kôd 4.4 Funkcija za učitavanje ključeva

Ključevi osim što se spremaju u datoteku wallet.txt prilikom kreiranja, mogu se pospremiti i u slučaju kada se već koriste u blok chain-u pomoću funkcije `def save_keys` koja provjera da li ključevi već postoje ukoliko varijable nisu prazne radi istu funkcionalnost kao i kod funkcije `create_keys`.

```
def save_keys(self):
    if self.public_key != None and self.private_key
!= None:
        try:
            with open('wallet.txt', mode='w') as f:
                f.write(self.public_key)
                f.write('\n')
                f.write(self.private_key)
            return True
        except (IOError, IndexError):
            print('Saving wallet failed')
            return False
```

Kôd 4.5 Funkcija za pospremanje ključeva.

Pomoću python modula pycrypto generiramo privatni i javni ključ. Ključevi su usko vezani stoga se kreiraju zajedno. Privatni ključ je dio javnog ključa no to nije moguće zaključiti iz samog javnog ključa. Pošto su ključevi generirani u binarnom formatu, želimo ih konvertirati u string odnosno niz znakova kako bi se mogli koristiti u blok chain-u, pospremati u datoteke i slično.

```
    def generate_keys(self):
        private_key = RSA.generate(1024,
Crypto.Random.new().read)
        public_key = private_key.publickey()
        return
(binascii.hexlify(private_key.export_key(format='DER'))
.decode('ascii'),
(binascii.hexlify(public_key.export_key(format='DER')).
decode('ascii')))
```

Kôd 4.6 Funkcija za generiranje ključeva.

Najvažnija klasa u kojoj definiramo naš blok chain, a ujedno i objedinjuje sve ostale klase naziva se blockchain. Ona sadrži funkcionalnosti učitavanje te pospremanje podataka u blok chain. Dohvaćanja raspoloživog stanja, transakcija. Dodavanje novih transakcija u blok chain te rudarenje koje je objašnjeno u cjelini 3.2. Objašnjenje pojmova rudarenja. Definirana je klasa naziva Blok chain sa varijablama u konstruktoru. Varijable koje u početku imaju dvije doljnje crtice python interpretira kao privatne da bi naglasili da su informacije koje te varijable sadrže povjerljive, odnosno izmjene u takvim varijablama treba raditi s oprezom. Definirana je varijabla genesis_block u kojoj je definiran prvi blok blok chain-a. U početku .py datoteke napravljen je ulaz ostalih klasa te modula pošto kao što je spomenutu u početku ova klasa koristi većinu funkcija iz ostalih klasa jer je pisana kao glavni dio koda u ovom radu. Iznad same klase definirana je i varijabla `MINING_REWARD` koja zapravo predstavlja nagradu za obradu otvorenih transakcija.

```
import functools
import hashlib as hl
import json
import pickle
```

```
from block import Block
from transaction import Transaction
from verification import Verification
from hash_util import hash_block
from wallet import Wallet

MINING_REWARD = 10

class Blockchain:
    def __init__(self, hosting_node_id):
        genesis_block = Block(0, '', [], 100, 0)
        self.chain = [genesis_block]
        self.__open_transactions = []
        self.hosting_node = hosting_node_id
        self.__peer_nodes = set()
        self.load_data()
```

Kôd 4.7 Definiranje klase Blockchain

Funkcija `load_data` radi učitavanje blok chain-a iz blockchain.txt datoteke pozivom ugrađene json.loads funkcionalnosti iz json modula.. Otvara navedenu datoteku u statusu za čitanje te ovisno o sadržaju datoteke popunjava listu podataka definiranu u varijabli `self.chain`. Pošto se sadržaj blok chain-a sprema u json formatu baš kao i kod učitavanja ključeva digitalnog novčanika potrebno je napraviti konverziju standardne liste podatka u sortirani *dictionary* koji u ovom kodu spremamo u varijablu updated_blockchain. Zatim varijabli self.chain pridijelimo vrijednosti updated_blockchain varijable. Jednaku proceduru radimo i s otvorenim transakcijama. Razlog ovog postupka je to što nove transakcije pospremamo kao nesortirane liste odnosno *dictionare*, a redoslijed učitavanja podataka u blok chain je važan.

```
def load_data(self):
        try:
            with open('blockchain.txt', mode='r') as f:
                file_content = f.readlines()
                blockchain =
json.loads(file_content[0][:-1])
                updated_blockchain = []
                for block in blockchain:
```

```
                    converted_tx =
[Transaction(tx['sender'], tx['recipient'],
tx['signature'], tx['amount']) for tx in
block['transactions']]
                    updated_block =
Block(block['index'], block['previous_hash'],
converted_tx, block['proof'], block['timestamp'])

updated_blockchain.append(updated_block)
                self.chain = updated_blockchain
                open_transactions =
json.loads(file_content[1][:-1])
                updated_transactions = []
      for tx in open_transactions:
                    updated_transaction =
Transaction(tx['sender'], tx['recipient'],
tx['signature'], tx['amount'])

updated_transactions.append(updated_transaction)
                self.__open_transactions =
updated_transactions
      except (IOError, IndexError):
         pass
```

Kôd 4.8 Funkcija za učitavanje blok chain podataka

Sličan postupak prolazi i `save_data` funkcija. Ona služi za spremanje podataka u datoteku i to radi također pomoću json standarda. U kodu datoteka se otvara u statusu za pisanje. Također se radi konverzija podataka blokova i transakcija u sortirane *dictionare* kako bi podaci poprimili *key : value* strukturu.

```
    def save_data(self):
        try:
            with open('blockchain.txt', mode='w') as f:
                saveable_chain =  [block.__dict__ for
block in [Block(block_el.index, block_el.previous_hash,
[tx.__dict__ for tx in block_el.transactions],
block_el.proof, block_el.timestamp) for block_el in
self.chain]]
                f.write(json.dumps(saveable_chain))
```

```
                    f.write('\n')
                    saveable_tx =  [tx.__dict__ for tx in
self.__open_transactions]
                    f.write(json.dumps(saveable_tx))
                    f.write('\n')

f.write(json.dumps(list(self.__peer_nodes)))
            except IOError:
                print('File se nije uspio pospremiti')
```

Kôd 4.9 Funkcija za pospremanje blok chain podataka

Funkcija `get_balance` služi za dohvaćanje stanja tokena, a ujedno i kao provjera kod slanja novih transakcija kao mehanizam zaštite da ne bismo mogli poslati više tokena nego što ih imamo. Kao rezultat vraća stanje tokena kojim raspolažemo.

```
  def get_balance(self):
            if self.hosting_node == None:
                return None
            participant = self.hosting_node
            tx_sender = [[tx.amount for tx in
block.transactions if tx.sender == participant] for
block in self.chain]
            open_tx_sender = [tx.amount for tx in
self.__open_transactions if tx.sender == participant]
            tx_sender.append(open_tx_sender)
            amount_sent = functools.reduce(lambda tx_sum,
tx_amt: tx_sum + sum(tx_amt) if len(tx_amt) > 0 else
tx_sum + 0, tx_sender, 0)
            tx_recipient = [[tx.amount for tx in
block.transactions if tx.recipient == participant] for
block in self.chain]
            amount_received = functools.reduce(lambda
tx_sum, tx_amt: tx_sum + sum(tx_amt) if len(tx_amt) > 0
else tx_sum + 0, tx_recipient, 0)
            return amount_received - amount_sent
```

Kôd 4.10 Funkcija za učitavanje stanja tokena i provjere iznosa kod slanja transakcija

Funkcija `add_transaction` služi za dodavanje novih transakcija u otvorene transakcije koje tada čekaju na obradu. Prije dodavanja nove transakcije prolazi kroz provjeru funkcije `verify_transaction` definiranoj u Verification klasi koja je objašnjena u cjelini 5.2.Verifikacija i validacija. Ukoliko je provjera potpisa uspješna transakcija se dodaje u

listu otvorenih transakcija. Nakon toga poziva se funkcija `save_data()` koja posprema ažurirane otvorene transakcije u blockchain.txt datoteku.

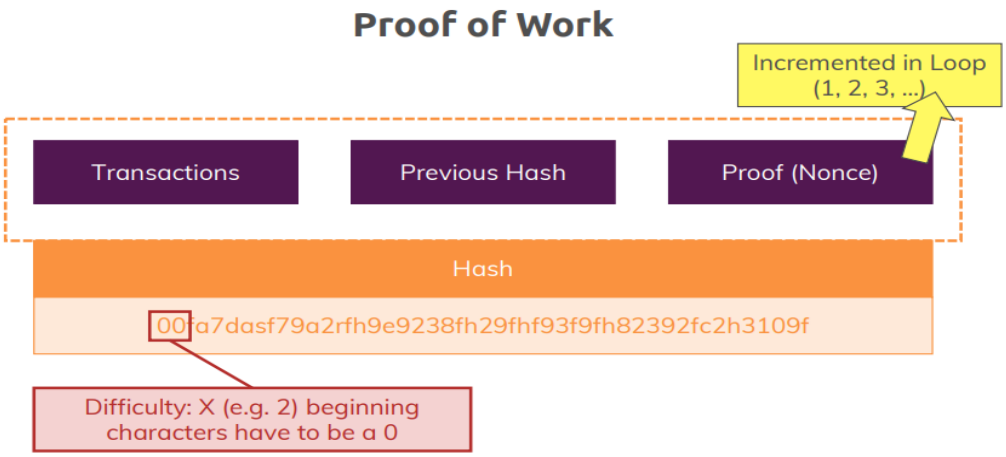
```
def add_transaction(self, recipient, sender, signature,
amount=1.0):
        if self.hosting_node == None:
            return False
        transaction = Transaction(sender, recipient,
signature, amount)
        verifier = Verification()
        if verifier.verify_transaction(transaction,
self.get_balance):

self.__open_transactions.append(transaction)
            self.save_data()
            return True
        return False
```

Kôd 4.11 Funkcija za dodavanje transakcija

4.2. Algoritmi i decentralizacijski sustavi

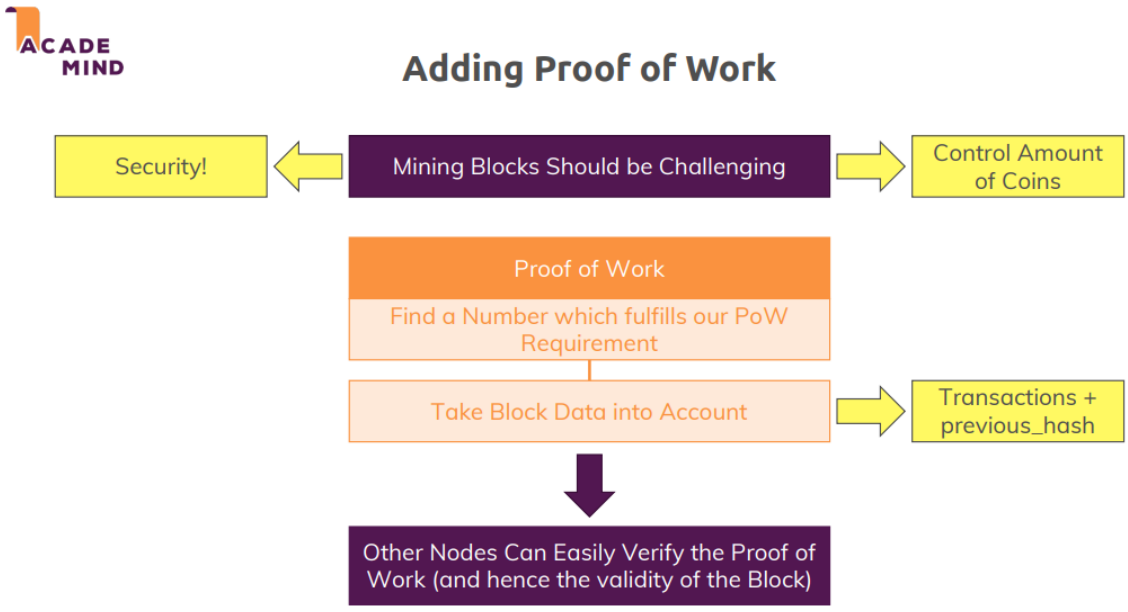
Kako je objašnjeno u cjelini 2 kod ovog blok chain-a je baziran na POW algoritmu. Provjera da li blok dva ima jedan ima jednak hash kao blok jedan koji bi dobili tako da preračunamo hash vrijednost bloka dva nije dovoljna. Ovo provjeru radimo kroz funkciju `verify_chain` u poglavlju 5.2. Netko bi jednostavno mogao manipulirati transakcijama spremljenim u bloku jedan i zatim promijeniti blok dva da sadrži novokreirani hash iz bloka jedan i tako za do kraja cijelog lanca blokova. Zaštitu od ovakve radnje nam upravo pruža POW algoritam. Imati jednake hasheve jednostavno nije dovoljno. Zato POW broj kojeg smo ranije spomenuli kao dokaz (engl. *proof*) ili (engl. *nonce*) je pogađan za svaki blok. Navedenu proceduru opisuje Slika 4.1 POW algoritam.



Slika 4.1 POW algoritam[31]

POW je samo broj, ali to je broj koji sa ostalim ulaznim podacima generira različiti rezultat hasha dok ne poprimi hash koji smo definirali u uvjetu odnosno funkciji `valid_proof` u cjelini 5.2. Taj hash nije isti hash koji uspoređujemo između dva bloka. Ovo je hash koji se koristi kod kreacije novih blokova, a najčešće je generiran iz podataka transakcija koje će biti spremljene u novi blok, hasha prethodnog bloka i nonce broja koji se mijenja dok ne dobijemo odgovarajući hash. Ako smo u uvjetu definirali da taj hash mora početi s određenim brojem nula ovisno koliko nula tražimo to će računalima duže trebati za kreaciju novog bloka. Za primjer za Bitcoin valutu svaki novi blok je dodan u prosjeku od 15 minuta. Ostali nodovi u blok cahin-u mogu jednostavno provjeriti novo dodane blokove i dozvoliti ili odbiti validaciju takvog bloka. Kada bi zlonamjerni nodovi pokušali promijeniti postojeće transakcije nekog bloka, hash ostalih blokova više ne bi bio valjan. Zato bi takvi nodovi trebali promijeniti i sve blokove poslije manipuliranog bloka za što bi im trebalo jako puno vremena jer se štitimo da hash mora sadržavati dvije vodeće nule. Ovo je ne moguće provesti jer ne postoje dovoljno brza super računala koja bi ovo stigla odraditi pošto se novi blokovi stalno na dodaju. Prethodni tekst opisuje sljedeća Slika 4.2 Logički prikaz POW procesa.

[31] Izvor slike Udemy, https://www.udemy.com/learn-python/, ožujak 2019



Slika 4.2 Logički prikaz POW procesa[32]

U programskom kodu dokaz (engl. proof) dobivamo kroz funkciju `proof_of_work` koja povećava dokaz za vrijednost jedan sve dok se ne dobije hash vrijednost koja je određena funkcijom `valid_proof` u klasi `Verification`. Funkcija `valid_proof` objašnjena je u cjelini 5.2.Verifikacija i validacija.

```
def proof_of_work(self):
        last_block = self.chain[-1]
        last_hash = hash_block(last_block)
        proof = 0
        verifier = Verification()
        while not
verifier.valid_proof(self.__open_transactions, last_hash,
proof):
            proof +=1
        return proof
```

Kôd 4.12 Funkcija za generiranje hash vrijednosti

[32] Izvor slike Udemy, https://www.udemy.com/learn-python/, ožujak 2019

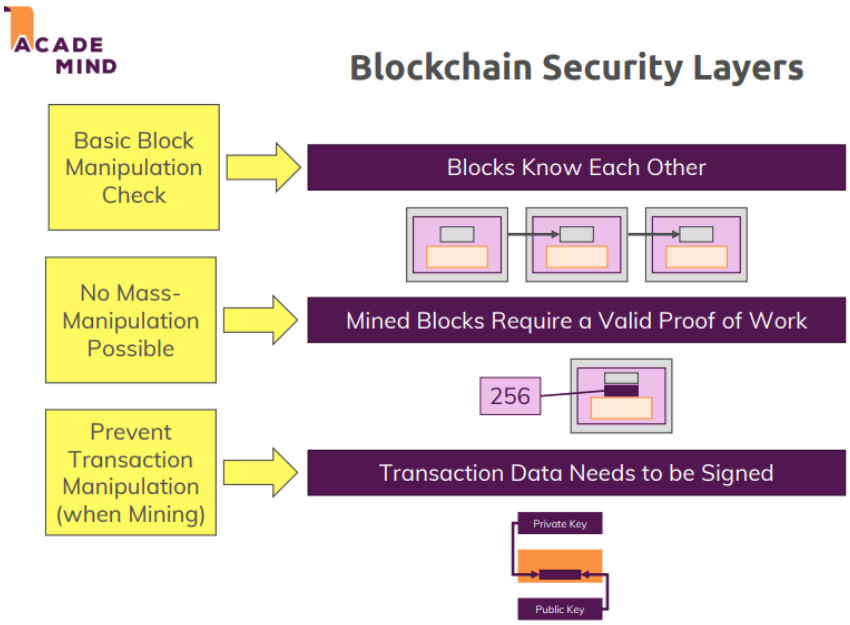
4.2.1. Konsenzus

Kada kažemo da je blok chain sustav decentraliziran mislimo na to da da nema centralnu točku kontrole. Samim time što nema centralnu točku upravljanja sustav je pošteniji i sigurniji. Blok chain to postiže takozvanim konsenzus protokolom koji je distribuiran po svim nodovima blok chaina. Konsenzus se bavi validacijom i pohranom transakcija jer je blok chain-u kao sustavu od iznimne važnosti da se transakcije obrađuju precizno i pravedno. Nadalje za transakcije koje su jednom upisane u blok chain možemo sa sigurnošću tvrditi da su pravovaljane radi činjenice da je za manipulaciju istih potrebno promijeniti višestruke kopije istih podataka koji osigurava konsenzus algoritam. Sa strane podatkovne pohrane možemo reći da su kopije istih podataka distribuirane duljem mreže te samim time ne postoji potreba za centralni podatkovnim sustavom. Postavlja se problem kako različiti nodovi razmjenjuju blokove podataka. Recimo da je na nodu1 kreiran novi blok sa zapisima o transakcijama, zatim je nodu2 kreiran također novi blok. Oba noda sadrže jednoliko veliki lanac blokova podataka. Nakon toga je node3 također kreirao novi blok ali on ne sadrži kompletan lanac blokova jer još nije sinkroniziran sa ostalim nodovima. Postavlja se pitanje koji će blok blok chain sustav uzeti kao validan i pospremiti u sustav. Odgovor je da se blok kreiran na nodu3 uopće neće uzeti u obzir jer on ne sadrži konzistentno stanje blok chain-a, dok će se između noda1 i noda2 odlučiti na onaj blok kojim ima većinu mreže. Ukoliko je node1 uspio poslati informaciju o novom bloku na 51% nodova njegov blok će biti dodan u lanac blokova blok chain sustava. Konflikti u blok chain-u su česti te da bi sustav držali ažurnim a podatke o blokovima između nodova sinkroniziranim koristi se upravo konsenzus algoritam.

5. KRIPTOGRAFIJA I KRIPTO VALUTE

5.1. Sigurnost kripto valuta

Da bi osigurali integritet i ne promjenjivost podataka blok chain se koristi različitim mehanizmima zaštite. Sigurnosni slojevi koji su implementirani u programski kod ovog rada sprečavaju manipulaciju podacima. Blok chain štitimo na razini blokova tako da provjeravamo prethodni hash blokova ukoliko je on različit od onoga u bloku ispred njega smatramo da je u blok chain-u došlo do korupcije podataka odnosno ne željenih manipulacija. Osim provjere ne promjenjivosti podataka blokova kod dodavanja novih blokova tražimo određenih hash da bi se taj novi blok uspješno dodao u lanac blokova POW mehanizam. Navedene zaštite opisuje Slika 5.1 Sigurnosni slojevi blok chain-a .

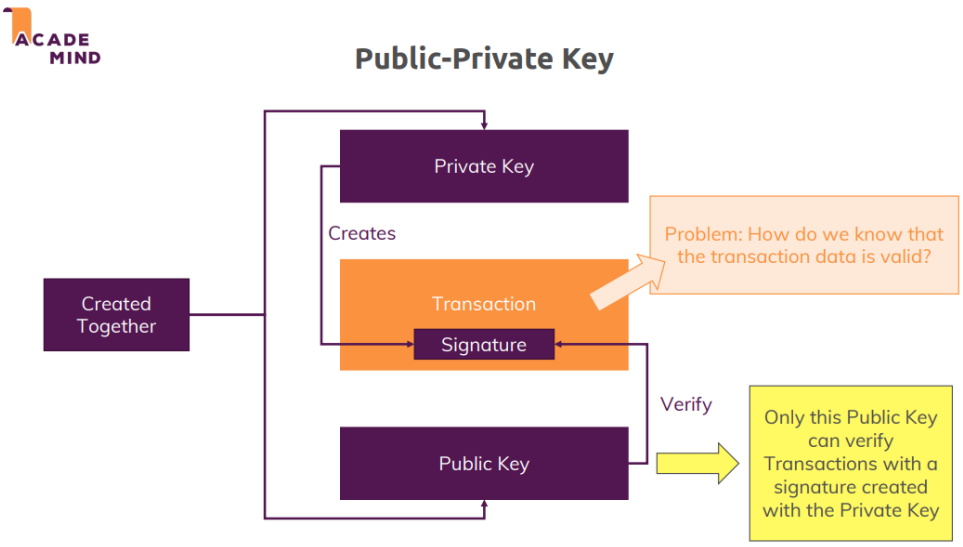


Slika 5.1 Sigurnosni slojevi blok chain-a[33]

Postoji još jedno mjesto gdje bi moglo doći do manipulacije podataka, a to su otvorene transakcije. Kako možemo vjerovati onome što je zapisano u transakcije koje nisu još dio blok chain-a i da li su ti podaci validni. Kao sloj zaštite protiv takvih manipulacija podataka koristi se mehanizam potpisivanja transakcija. Svaku transakciju potpisujemo privatni

[33] Izvor slike Udemy, https://www.udemy.com/learn-python/, ožujak 2019

ključem tako da stvaramo popis nad podacima koliko čega šaljemo i kome šaljemo. Potpis se može verificirati samo pomoću javnog ključa koji je kreiran u paru s privatnim ključem. Ukoliko bi se podacima probalo manipulirati potpis više ne bi bio valjan te samim time transakcija se ne bi uzela u obzir kod postupka rudarenja. Ukoliko bi napadač pokušao promijeniti i potpis nakon što je promijenio podatke u transakciji to bi mogao napraviti samo vlastitim privatnim ključem jer je informacija privatnog ključa pošiljatelja uvijek tajna. Samim time verifikacija potpisa putem javnom ključa bi bila neispravna jer je taj javni ključ vezan uz privatni ključ pošiljatelja koji napadač ne posjeduje. Navedeni proces opisuje Slika 5.2 Provjera potpisa transakcije .



Slika 5.2 Provjera potpisa transakcije[34]

5.2. Verifikacija i validacija

Cilj kvalitetnog blockchaina je da bude siguran i eliminira sve zlonamjerne radnje i pokušaje manipulacije podacima. Blockchain u ovom radu napisan je na način da sadrži različite korake validacije i provjere prilikom obavljana operacija slanja, rudarenja, dohvaćanja transakcija. Dio koda vezan uz verifikaciju i validaciju podataka većinom je smješten u datoteci verification.py i hash_util.py. Sigurnosne provjere vezane uz provjeru transakcija također su dijelom pisane u wallet.py datoteci jer se tiču samog digitalnog novčanika koji

[34] Izvor slike Udemy, https://www.udemy.com/learn-python/, ožujak 2019

radi s transakcijama. Kako algoritme i mehanizme za validaciju, verifikaciju podataka teško možemo kvalitetno napisati sami jer uključuju i dobro poznavanje matematike koristimo gotove python module koji nam ih pružaju.

Funkcija `valid_proof` služi za validaciju POW mehanizma. Provjerava da li dobivena hash vrijednost odgovara zadanom algoritmu u kojem tražimo da hash vrijednost počinje s dvije vodeće nule. Ovo nije isti hash koji se koristi kao prethodni hash bloka već je ovo hash koji zahtijevamo prilikom rudarenja, obrade otvorenih transakcija kako bi stvorili novi blok. Kako što je opisano u poglavlju 2.1.2 Algoritmi Blok chain-a ovim algoritmom nemjereno otežavamo kreaciju novih blokova kako bi spriječili zlonamjerne radnje i brzo iskopavanje svih novčića.

```
def valid_proof(self, transactions, last_hash, proof):
      guess = (str([tx.to_ordered_dict() for tx in transactions]) +
      str(last_hash) + str(proof)).encode()
      guess_hash = hash_string_256(guess)
      return guess_hash[0:2] == '00'
```

Kôd 5.1 Funkcija za provjeru hash vrijednosti.

Validacija lanca blokova vrši se kroz funkciju def `verify_chain` prikazanoj u kodu Kôd 5.2 Funkcija za verifikaciju lanca blokova. U *for* petlji prolazimo kroz cijeli blok chain te u slučaju da je indeks bloka jednak 0 taj blok preskačemo jer je to zapravo izvorni blok (engl. *genessis block*) koji nema svog prethodnika. Nastavljamo provjeravati iduće blokove u lancu te za svaki blok provjeravamo prethodnu hash vrijednost ukoliko je ona različita od vrijednosti u trenutnom bloku vraća se rezultat `false` što znači da je proces provjere blok chain-a negativan, odnosno da je u blok chain-u došlo do korupcije podataka.

```
def verify_chain(self, blockchain):
for (index, block) in enumerate(blockchain):
      if index == 0:
            continue
      if block.previous_hash != hash_block(blockchain[index1]):
            return False
      if not self.valid_proof(block.transactions[:-1],
      block.previous_hash, block.proof):
            print('Proof of work is invalid')
            return False
```

```
            return True
```

Kôd 5.2 Funkcija za verifikaciju lanca blokova

Provjeravamo pojedinu transakciju ovisno o stanju tokena kojima raspolažemo. Funkcija def verify_transaction prikazana u kodu je mehanizam zaštite koji nam ne dozvoljava da pošaljemo više nego što imamo u digitalnom novčaniku. Uz raspoloživost tokena provjerava i da li je popis transakcije validan.

```
def verify_transaction(self, transaction, get_balance, check_funds=True):
      if check_funds:
            sender_balance = get_balance()
            return sender_balance >= transaction.amount and
            Wallet.verify_transaction(transaction)
      else:
            return Wallet.verify_transaction(transaction)
```

Kôd 5.3 Funkcija za provjeru raspoloživog stanja tokena

Funkcija def verify_transactions služi za provjeru otvorenih transakcija.

```
def verify_transactions(self, open_transactions, get_balance):
      return all([self.verify_transaction(tx, get_balance, False) for tx
      in open_transactions])
```

Kôd 5.4 Funkcija za provjeru otvorenih transakcija

Kod operacije kreiranja nove transakcija potrebne su nam provjere kako bi dokazali da te transakcije nisu lažirane. Kako se ove radnje tiču samih transakcija, a transakcije su vezane uz naš digitalni novčanik metode, funkcije potpisivanja i verificiranja transakcija smještene su u `wallet.py` datoteku. Radi potrebe različitih algoritma za enkripciju i dekripciju koriste se gotovi python moduli hashlib, pycryptodome. Primjer koda Kôd 5.5 Funkcija za potpisivanje transakcija pokazuje definiranje funkcije `sign_transaction`. Funkcija `sign_transaction` poprima objekte pošiljatelj, primatelj i iznos. Definiramo varijablu `signer` te u nju spremamo privatni ključ u binarnom obliku koji smo proslijedili `PKCS1_v1_5` algoritmu radi enkripcije. Kako bismo kreirali potpis osim privatnog ključa u varijablu `h` spremamo i podatke nad kojima kreiramo hash vrijednost. U varijabli `singature` kreiramo potpis nad gore spomenutim vrijednostima te kao rezultat vraćamo taj potpis kao niz znakova (engl. string).

```
def sign_transaction(self, sender, recipient, amount):
```

```
signer=PKCS1_v1_5.new(RSA.import_key(binascii.unhexlify(self.privat
e_key)))
h = SHA256.new((str(sender) + str(recipient) +
str(amount)).encode('utf8'))
signature = signer.sign(h)
return binascii.hexlify(signature).decode('ascii')
```

Kôd 5.5 Funkcija za potpisivanje transakcija

Kako bismo verificirali transakcije dodajemo novu metodu slika 5.1, funkciju pomoću koje provjeravamo da li je potpis validan. U metodi dohvaćamo kompletan transakcija(engl. transaction) objekt jer se tamo nalaze svi potrebni podaci nad kojima moramo obaviti validaciju. Kreirana je varijabla `public_key` u kojoj dohvaćamo javni ključ pošiljatelja u binarnom obliku. Zatim u varijablu `verifier` spremamo taj javni ključ pomoću `PKCS1_v1_5` algoritma. Nakon toga u varijabli `h` kreiramo hash od vrijednosti pošiljatelja, primatelja i iznosa. Kao rezultat vraćamo provjerenu vrijednost hasha i potpisa provjerenih ugrađenom `verify` funkcijom koja provjerava autentičnost potpisa.

```
@staticmethod
      def verify_transaction(transaction):
            public_key =
            RSA.import_key(binascii.unhexlify(transaction.sender))
            verifier = PKCS1_v1_5.new(public_key)

            h = SHA256.new((str(transaction.sender) +
            str(transaction.recipient) +
            str(transaction.amount)).encode('utf8'))
            return verifier.verify(h,
            binascii.unhexlify(transaction.signature))
```

Kôd 5.2 Definiranje funkcije za provjeru transakcije

5.3. Hashing algoritmi

Kao što se može primijetiti riječ hash se koristi izrazito puno u blockchain svijetu, no što je to hash?. Važno je shvatiti što je to točno hash i zašto ga koristimo. Najjednostavnije rečeno hash možemo zamisliti kao dugačak niz znakova, točnije 64 znaka ako je u pitanju SHA256 algoritam. Nad hash vrijednosti nije moguće napraviti obrnuti inženjering (engl. Reverse engineering). Hash je kreiran od nekih ulaznih podataka, a ti isti podaci će uvijek dati identičnu hash vrijednost u svakom ponovnom preračunavanju hasa za te ulazne podatke.

Ulazne podatke ne možemo nikako saznati osim ako nemamo mapu ulaza i odgovarajućih hasheva. Naravno to nije nešto što možemo lako kreirati , brute forcing takve mape bi zahtijevao bilijune mogućih kombinacija ovisno o hashng algoritmu koji je primijenjen. Hash bloka sadrži 64 znaka što je kraće od bloka koji ima desetke transakcija. Dužina hasha se ne povećava ili smanjuje ovisno o ulaznim podacima pa je efikasnije spremati hash vrijednosti umjesto nasumičnih nizova znakova dobivenih od svih podataka u bloku. Hash se u blok cahin-u ne koristi da bi se sakrili pravi podaci, zapravo poanta blockchaina je da svi podaci budu javno dostupni. U blok chain kodu priloženom u ovom radu koriti se ugrađeni modul hashlib za generiranje hash vrijednosti Kôd 5.6 Funkcija za generiranje hash vrijednosti nad zadanom nizu znakova. Najjednostavnija funkcija u cijelom kodu je funkcija hash_string_256 koja za argument uzima bilo koji niz znakova (engl. string) i vraća hash vrijednost napravljenu nad tim *stringom* pomoću SHA256 algoritma.

```
import hashlib as hl
def hash_string_256(string):
      return hl.sha256(string).hexdigest()
```

Kôd 5.6 Funkcija za generiranje hash vrijednosti nad zadanom nizu znakova.

Funkcija hash_block vidljiva u kodu `Kôd 5.7 Funkcija za generiranje hash vrijednosti bloka` nad podacima bloka napravi novi hash i vrati string vrijednost bloka u json formatu. Za argument prima cijeli blok.

```
def hash_block(block):
      hashable_block = block.__dict__.copy()
      hashable_block['transactions'] = [tx.to_ordered_dict() for tx
      in hashable_block['transactions']]
      return hash_string_256(json.dumps(hashable_block,
      sort_keys=True).encode())
```

Kôd 5.7 Funkcija za generiranje hash vrijednosti bloka

6. TRENUTNA ISTRAŽIVANJA I ANALIZE TEHNOLOGIJE

6.1. Primjena blok chain-a

Blok chain osim što bi se mogao primijeniti kao zamjena za transakcijski sustav opisan u ovom radu koristi se za mnoge druge svrhe koje ćemo opisati u nastavku. Najpopularnije kripto valute odnosno blok chain-ovi koje one predstavlja su platformski blok chain-ovi. Neki od popularnijih platformskih blok cahin-ova su Ethereum, EOS, NEO. Nazivamo ih platformskim blok chain-ovima iz razloga jer se koriste kao što i samo ime kaže kao platforma za aplikacije odnosno decentralizirane aplikacije koje se pišu na takvim platformama. Sve popularnije decentralizirane aplikacije uzimaju pažnju programerima, poslodavcima u IT industriji iz razloga je su visoko dostupne, decentralizirane odnosno nismo ih potrebni pokretati na vlastitoj serverskoj opremi. Da bi se olakšalo kreatorima takvih aplikacija izumljene su platforme bazirane na blockcahin tehnologiji. Ono što one pružaju su gotove biblioteke (engl. Libary), kosturi (engl. Franework) za izradu aplikacija. Pokušava se olakšati programerima u pisanju decentraliziranih aplikacija da sve imaju na jednom mjestu i da se ne moraju brinuti oko temelja svojih aplikacija kao što su sigurnost, visoko dostupnost, skalabilnost. Nekoliko potencijalnih primjena blok chain- a slijedi u nastavku (Digital Information World , Anastasia Stefanuk [13]).[35]

- Obrada plaćanja i prijenos novca - za ovaj model možemo reći da ga dijelom opisuje sam ovaj rad. Mislim se na model blockchaina koji zamijenio stvarna kartična plaćanja.
- Praćenja lanaca opskrbe resursima – pošto blok chain može spremati velik broj transakcija, informacija i vidjeti njihovu povijest. Tvrtke bi mogle raznim analitičkim postupcima zaključiti gdje se njihove materijali, resursi, proizvodi zadržavaju najdulje u procesu nabave i dostave te sukladno tome promijeniti način poslovanja kako bi udovoljile krajnjim korisnicima.
- Digitalni identitet- Koliko god zvučalo apsurdno mnoge država diljem svijeta

35https://www.digitalinformationworld.com/2018/05/top-5-countries-embracing-blockchain.html,ožujak2019.

pogotovo one slabijeg ekonomskog statusa još uvijek nisu riješile pitanje digitalnog identiteta. Ideja je da bi se osobni podaci mogli čuvati u blok chain-u, te bi se na taj način omogućio ljudima pristup financijskim uslugama, pokretanje vlastitog biznisa.

- Digitalno glasanje – Recimo da država primjeni digitalno glasanje bazirano na blok chain- u. Lažni glasovi, promjena jednom unesenih podataka teško bi mogle manifestirati.
- Trgovanje dionicama – U nekom trenutku, blok chain bi se mogao natjecati ili zamijeniti postojeće platforme za trgovanje dionicama za kupnju ili prodaju dionica. Budući da blok chain mreže tako brzo provjeravaju i rješavaju transakcije, to bi moglo eliminirati vrijeme višekratnog čekanja koje investitori susreću prilikom prodaje zaliha i tražeći pristup svojim sredstvima u svrhu ponovnog ulaganja ili povlačenja.
- Vođenje medicinske dokumentacije- Dobra vijest je da se medicinski sektor već godinama udaljava od papira u svrhu vođenja evidencije. Međutim, blok chain nudi još veću sigurnost i praktičnost. Osim pohranjivanja podataka o pacijentima, pacijent koji posjeduje ključ za pristup tim digitalnim zapisima, imao bi kontrolu nad time tko će dobiti pristup tim podacima. To bi bilo sredstvo za jačanje HIPAA[36] zakona koji su osmišljeni kako bi zaštitili privatnost pacijenata.

Kao što se vidi iz priloženoga mogućnosti ovakve tehnologije su brojne. Zapravo još nismo ni svjesni gdje bi sve ovu tehnologiju mogli primijeniti, no gdje su danas države u toj priči i kako su prihvatile blok chain. Sjedinjene Države Amerike postale su utočište za burze dionicama baziranih na blok chain-u[37]. Mreža za provedbu financijskih zločina FinCEN[38] 2013 godine utvrdila je da je potrebno pratiti i regulirati kartične i gotovinske transakcije vezane uz trgovanje kripto vrijednostima. Slični projekti nastali su iz potrebe regulacije prometa kripto valutama da bi spriječili pranje novaca, sponzoriranja terorizma i drugih zločina. Što se tiče digitalnih valuta Bitcoin i ostale kripto valute tretiraju se kao vlasništvo, što znači da se nakon ostvarene kapitalne dobiti nastale prodajom valuta plaća porez. ICO [39]je također pod regulativom SAD-a, što znači da postoji razvijen postupak kontrole nastao

[36] HIPAA - Zakon o prenosivosti i odgovornosti zdravstvenog osiguranja (engl. (Health Insurance Portability and Accountability Act)
[37]https://www.digitalinformationworld.com/2018/05/top-5-countries-embracing-blockchain.html,ožujak2019.
[38] FinCEN - Mreža za provedbu financijskih zločina
[39] ICO - financijski instrument sličan startapu odnosno ranom financiranju projekta (engl. initial coin offering)

od komisije za sigurnost i razmjenu SEC i[40] SAFT-a[41] kojim se kontroliraju rane investicije novih projekata. Što bi zapravo značilo da moramo obrazložiti s koliko ćemo tokena plasirati blok cahin projekt na tržište te koliko brzo će se stvarati novi tokeni ako je blok chain POW tipa. Kanada isto kao i SAD pruža niz bankomata diljem zemlje gdje možemo razmijeniti bitcoin i ostale valute za novac i obrnuto. U velikim trgovinama dozvoljeno je plaćanje kripto valutama. Potiču se i održavaju programi za obrazovanje ljudi u svrhu razumijevanja blok chain arhitekture i njegove primjene.

6.2. Usporedbe različitih tehnoloških projekata nastalih u blok chain-u

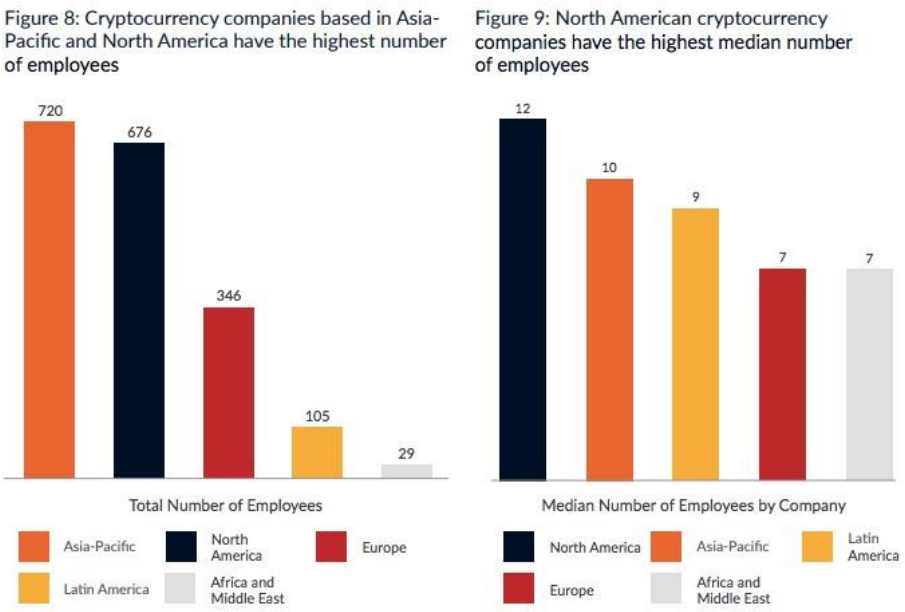
Bitcoin je nastao kao prvi blok chain protokol. Karakteristike ovog blok chain protokola su da je javno dostupan u smislu da bilo tko može biti sudionik. U praktičnom djelu rada korištene se iste tehnološke komponente koje su na sličan način implementirane u bitcoin, a to su: POW algoritam, konsenzus, digitalnim potpisi, izračunavanje kriptografskih hash vrijednosti, privatni i javni ključevi. Svaki server (engl. Node) sadrži kompletnu informacijsku strukturu što ga čini decentraliziranim. Svaki novi blok u prosjeku je iz generiran svakih deset do petnaest minuta. U odnosu na bitcoin koji samo omogućava kripto plaćanja preko decentralizirane mreže ethereum je planiran i napravljen za postignuće većih ciljeva. Programiran je kao platfroma koja se naziva EVM[42] (engl. Ethereum Virtual Machine) koja omogućava izgradnju decentraliziranih aplikacija. Broj etherenum aplikacija prošao je broj tisuću krajem 2018. godine. Aplikacije se grade pomoću takozvanih pametnih ugovora (engl. Smart contracts) koji služe za izvršavanje programskih funkcija kao što su: preuzimanje kontrole nad entitetom na temelju određenih uvjeta i prijenos kripto tokena na temelju ispunjavanja potrebnih uvjeta. Ethereum je napisan u vlastitom jeziku naziva solidity koji je nastao inspiracijom postojećih jezika c++, java, python.

[40] SEC - Komisija za sigurnost i razmjenu
[41] SAFT – (engl. Simple Agreement for Future Tokens)
[42] EVM - Ethereum Virtual Machine, komponenta ethereum blok chaina na kojoj se grade decentralizirane aplikacije (2).

6.3. Blok chain specijalizacija

Razvojem i napretkom tehnologije potražnja za blok chain inženjerima porasla je za 400% [43]od kraja 2017 godine. Potražnju je uzrokovalo jako i dinamično tržište blok chain projekata. Nove ideje i želje poslodavaca uzrokovale su znatno povećanje oglasa za posao u ovoj IT branši. Prema CNBC[44]-u te statistici koju provodi stranica hired.com (Hired, 2018 [15]) prosječna plaća blok chain inženjera kreće se između 150,000 do 175,000 dolara godišnje. Isto tako napominju da potražnja od strane većih tržišnih giganta Facebook, Amazon , IBM ,Microsoft već postoji. Takvim oglasima za posao najčešće odgovaraju i IT stručnjaci sljedećih struka: sistem inženjer, *solutions architect* te *backend* inženjer. Znanja koja se očekuje su poznavanje mrežnih protokola , baze podataka, kriptografija, programiranje, big data. Sve su popularniji online tečajevi koji pružaju blok chain certifikaciju. Popularna stranica Linkedin objavila je izvještaj potražnje za blok chain inženjerima te ih smjestila na prvo mjesto za 2018 godinu. Potražnja ove branše prema njima je narasla je 33 puta (Economic Graph Team, 2018 [16]). Isto tako ubrzo nakon toga izašao je i izvještaj od Cambridge-a centra za alternativne financije na Sveučilištu Cambridge u čijem se grafu nalazio tada trenutan broj stalnih blok chain zaposlenika Slika 6.1 Dijagram stalnih radnih mjesta u blok chain industriji 2018.

[43]https://cointelegraph.com/news/demand-for-blockchain-engineers-has-grown-400-since-end-of-2017 report-says, ožujak 2019.
44 CNBC-e američki televizijski kanal za novosti u poslovnom svijetu.

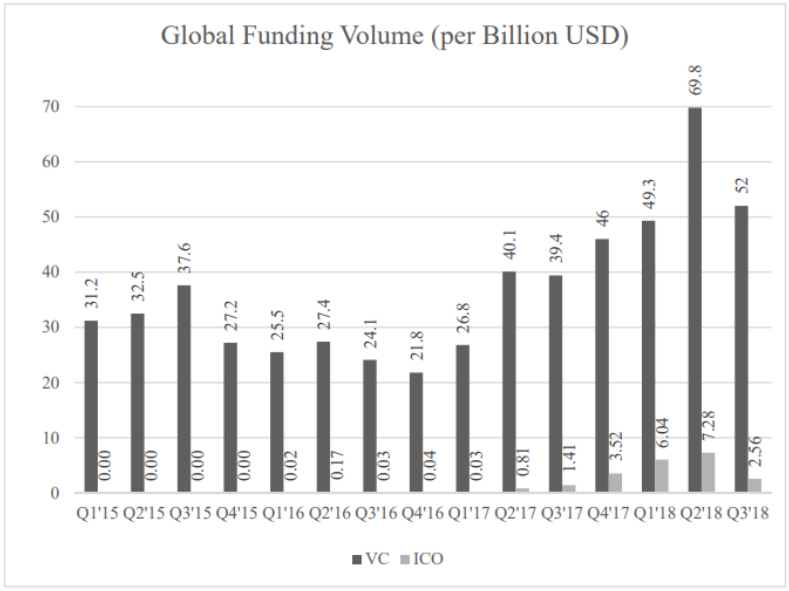


Slika 6.1 Dijagram stalnih radnih mjesta u blok chain industriji 2018.[45]

6.4. Trenutni ekonomski statusi postojećih projekata

Novi blok chain projekti financirani su kroz financijski model opisan u poglavlju 2.3 Burze kripto valutama ICO (engl. Initial token offerings) poznat još i pod nazivom prodaja tokena. Metoda ovakvog tipa prikupljanja sredstava ubrzo je postala i najpopularnija metoda za nove *startup* projekte. Problem financiranja kroz rizične kapitale [46](engl. Venture capital) bio je u tome da se nije mogla procijeniti stvarna vrijednost takvih projekata isto tako bilo je i za poslovne anđele. Rast kripto ekonomije donosi obećanja i opasnosti od mogućih ne uspjelih projekata te je iz tog razloga ICO potisnuo prethodna dva modela ulaganja. Za prvi kvartalu2018. godine coindesk[47] je objavio izvještaj da je ukupno bilo dvjesto dvije prodaje tokena čijim je rezultatom podignuto preko 6.3 biliona [48]dolara (18). Sljedeći graf Slika 6.2 Razlika ostvarenih sredstava putem VC i ICO prikazuje razliku prikupljenih sredstava za sve tehnološke projekte kroz modele ulaganja rizični kapitali VC te ostvarene prodajom tokena ICO.

[45] Izvor slike https://www.coindesk.com/research/state-blockchain-q3-2018, ožujak 2019.
[46] Rizični kapitali [46](engl. Venture capital) - pojedinci, tvrtke ili fondovi koji ulažu u pojedina poduzeća kako bi im pomogli u razvoju.
[47] Coindesk – najveća stranica za kripto Novosti
48 https://www.coindesk.com/research/state-blockchain-q1-2018, ožujak 2019.



[49]

Slika 6.2 Razlika ostvarenih sredstava putem VC i ICO[50]

Prema web stranici bitcoinmarketjournal.com [51]kao jedne od uglednijih stranica od strane blok chain investitora najuspješniji ROI[52] odrazio se kroz sljedeće prodaje tokena ICO-e slika 5. Izračuni su uzeti u kolovozu 2018. godine kada je objavljen i sam članak[53](Alex Lielacher, 2018 [12]).

[49] Lin Lin, Dominika Nestarcova,„Venture Capital in the Rise of Crypto Economy“,“NUS Law Working Paper 2019/003.
[50] Izvor slike https://www.coindesk.com/research/state-blockchain-q1-2018, ožujak 2019.
[51] https://www.bitcoinmarketjournal.com/ ožujak 2019- popularna web stranica kripto investitora
[52] ROI (engl. Return on Investment) povrat od uloženog, formula putem koje se prati zarada ostvarena od svake investicije, izražena u postocima.
[53] https://www.bitcoinmarketjournal.com/biggest-icos-roi/ ožujak 2019.

Token	ROI
NXT	598054%
IOTA	226013%
Ethereum	147877%
Neo	103133%
Spectrecoin	51328%
Stratis	38642%
Ark	11969%
Storj	6889%
Lisk	6414%
Augur	4917%

Slika 6.3 ROI od ulaganja u prodaji tokena ICO.[54]

Podaci u nastavku Slika 6.3 ROI od ulaganja u prodaji tokena ICO. izračunati su u vrijeme pisanje ovog rada. Za izračun ROI-a koristio sam se sljedećom formulom Slika 6.4 ROI formula za izračun povrata uloženog.

$$ROI = \frac{\text{Zarada} - \text{Uloženo}}{\text{Uloženo}}$$

Slika 6.4 ROI formula za izračun povrata uloženog

Postoci ROI-a su izrazito veliki, a da su istiniti dokazuje popularni ethereum blok chain projekt čiju su tokeni takozvani ether skraćeno ETH[55] imali vlastiti ICO odnosno prodaju tokena u ljetu 2014. godine. Ukupno je prodano 11,9 miliona tokena te je skupljeno 16 miliona američkih dolara, cijena jednog ether tokena tada je iznosila 0.311 američkih dolara. Najviša cijena jednog ehereum tokena iznosila je 1431.77 američkih dolara u siječnju 2018. godine. U vrijeme pisanja ovog rada jedan ether token vrijedi 136.11 američkih dolara. Iza toga zašto je ethereum bio toliko uspješan stoji revolucijalna ideja da se napravi platfroma koja pruža sve uvjete za izradu decentraliziranih aplikacija. Shodno uspješnom masovnom

[54] Izvor slike https://www.bitcoinmarketjournal.com/biggest-icos-roi/ ožujak 2019.
[55] ETH- Ether, kriptovaluta, token ethereum blok chaina

usvajanju tehnologije postotak povrata investicije je postao veći. U vremenu od godine dana kada su izračunati povrati investicija i trenutnog mjerenja koje je napravljeno u ovom radu možemo vidjeti da se je ROI znatno smanjio. Zašto je tome tako je razlog što je tržište kripto valuta jako promjenjivo i nestabilno. Puno su veće oscilacije cijena tokena od primjerice stvarnih tržišnih dionica ili zlata. U trgovanje kripto valutama uključeni su brokeri i profesionalci pa su bilo kakva osobna ulaganja dosta rizična. Projekti koji su ostvarili najveći ROI prikazan su na slici Slika 6.5 ROI od ulaganja u prodaji tokena ICO 2019.

ICO	Početna cijena	cijena 6.3.2019	ROI
NXT	0.0000168$	0.024311$	144608%
Ethereum	0.311$	136.11$	13548%
IOTA	0.001$	0.277971$	27697%
NEO	0.032$	8.82$	27462%

Slika 6.5 ROI od ulaganja u prodaji tokena ICO 2019. „vlastiti rad autora“

Zaključak

U različitim zemljama vlada ima drugačiji pristup u odnošenju prema blok chain tehnologiji. Postoje zemlje koje prihvaćaju tehnologiju i koriste njene prednosti i mogućnosti te one koje istu još uvijek zabranjuju. Unatrag godinu dana razvilo se na stotine različitih blok chain projekata različitih namjena. Najviše pozornosti privukli su takozvani platformski blok chain-ovi koji pružaju jednostavno razvijanje aplikacija uz riješeno pitanje skalabilnosti, redundancije podataka te dostupnosti. Unatrag par godina financiranje novih projekata putem digitalnih valuta odnosno ICO tipu ulaganja itekako se isplatilo za tvrtke koje su se odlučile na takav način prikupljanja sredstava. Blok chain sustavi pružaju kompleksnu i sigurnu infrastrukturu koja se mogla primijeniti za visoko rizične transakcije, projekte državnih institucija, globalne projekte. Osobno smatram da je kvalitetno programiran blok chain sustav dovoljno siguran da bih mu povjerio upravljanje bankovnim transakcijama i transakcijama između banaka. Tržište zahtijeva sve veći broj blok chain programera i stručnjaka što su pokazale statistike raznih analitičkih kuća opisane u zadnjoj cjelini ovoga rada. Trenutno na tržištu ne postoji slična tehnologija koja bi mogla konkurirati blok chain-u. Često dolazi do zabune da je cloud slična tehnologija no zapravo sadrži samo neke slične karakteristike kao što su visoko dostupnost, redundancija, skalabilnost. Decentralizacija i logika putem konsenzus algoritma te višestruke kopije istih podataka osiguravaju nam zaštitu od havarije i gubitka podataka. U konačnici smatram da je ovaj tip tehnologije izuzetno koristan te bi svakako ubrzo mogao postati i dio našeg svakodnevnog života.

Popis kratica

HTTP	*Hypertext Transfer Protocol*
GPU	*Graphics processing unit*
CPU	*Central processing unit*
TCP	*Transmission Control Protocol*
VM	*Virtual Machine*
IP	*IP Internet Protocol*
IT	*Information Technology*
ICO	*Initial token offerings*
VC	*Venture capital*
ROI	*Return on Investment*
POW	*Proof of Work*
POS	*Proof of Stake*
EVM	*Ethereum Virtual Machine*
SAFT	*Simple Agreement for Future Tokens*
HIPAA	*Health Insurance Portability and Accountability Act*
API	*Application programming interface*
URL	*Uniform Resource Locator*
PPS	*Pay Per Share*
SMPPS	*Shared Maximum Pay Per Share*
PPLNS	*Pay Per Last N Shares*

Popis slika

Slika 2.1 Redoslijed blokova 3

Slika 2.2 Bitcoin potrošnja energije 7

Slika 2.3 Web sučelje digitalnog novčanika 1 „vlastiti rad autora“ 8

Slika 2.4 Web sučelje digitalnog novčanika 2 „vlastiti rad autora“ 9

Slika 3.1 Prikaz početne stranice Hiveos operativnog sistema 18

Slika 3.2 Prikaz opterećenja grafičkih kartica u hiveos sustavu..... 19

Slika 3.3 Slanje transakcija putem digitalnog novčanika. “vlastiti rad autora” 20

Slika 3.4 Prikaz dodavanja novog bloka. “vlastiti rad autora”..... 21

Slika 4.1 POW algoritam..... 33

Slika 4.2 Logički prikaz POW procesa 34

Slika 5.1 Sigurnosni slojevi blok chain-a 36

Slika 5.2 Provjera potpisa transakcije 37

Slika 6.1 Dijagram stalnih radnih mjesta u blok chain industriji 2018. 46

Slika 6.2 Razlika ostvarenih sredstava putem VC i ICO..... 47

Slika 6.3 ROI od ulaganja u prodaji tokena ICO..... 48

Slika 6.4 ROI formula za izračun povrata uloženog 48

Slika 6.5 ROI od ulaganja u prodaji tokena ICO 2019. „vlastiti rad autora“ 49

Popis kôdova

Kôd 2.1 Definiranje ip adrese i porta aplikacije 10
Kôd 2.2 Definiranje HTTP rute / za GET metodu. 10
Kôd 2.3 Definiranje HTTP rute /wallet za GET metodu. 11
Kôd 2.4 Definiranje HTTP rute /wallet za POST metodu. 11
Kôd 2.5 Definiranje HTTP rute /balance za GET metodu 12
Kôd 2.6 Definiranje HTTP rute /transaction za POST metodu. 14
Kôd 2.7 Definiranje HTTP rute /chain za GET metodu. 14
Kôd 2.8 Definiranje HTTP rute /transaction za GET metodu 15
Kôd 3.1 Definiranje HTTP rute /mine za POST metodu. 22
Kôd 3.2 Funkcija za kreaciju blokova 23
Kôd 4.1 Definiranje klase Block 25
Kôd 4.2 Definiranje klase Transaction 25
Kôd 4.3 Definiranje klase Wallet 26
Kôd 4.4 Funkcija za učitavanje ključeva 27
Kôd 4.5 Funkcija za pospremanje ključeva. 27
Kôd 4.6 Funkcija za generiranje ključeva. 28
Kôd 4.7 Definiranje klase Blockchain 29
Kôd 4.8 Funkcija za učitavanje blok chain podataka 30
Kôd 4.9 Funkcija za pospremanje blok chain podataka 31
Kôd 4.10 Funkcija za učitavanje stanja tokena i provjere iznosa kod slanja transakcija 31
Kôd 4.11 Funkcija za dodavanje transakcija 32
Kôd 4.12 Funkcija za generiranje hash vrijednosti 34
Kôd 5.1 Funkcija za provjeru hash vrijednosti. 38
Kôd 5.2 Funkcija za verifikaciju lanca blokova 39

Kôd 5.3 Funkcija za provjeru raspoloživog stanja tokena..... 39

Kôd 5.4 Funkcija za provjeru otvorenih transakcija..... 39

Kôd 5.5 Funkcija za potpisivanje transakcija..... 40

Kôd 5.6 Funkcija za generiranje hash vrijednosti nad zadanom nizu znakova. 41

Kôd 5.7 Funkcija za generiranje hash vrijednosti bloka 41

Literatura

[1] Coindesk, https://www.coindesk.com/research/state-of-blockchains-q3-2018, ožujak 2019.

[2] Andreas M.Antonopoulos; Mastering Bitcoin Programming the Open Blockchain 2nd Edition ; (2017); http://oreilly.com/safari

[3] Don Tapscott; Alex Tapscott; Blockchain Revolution; (2016); ISBN 9781101980132

[4] Irman Bashir Mastering Bitcoin; (2017); ISBN 978-1-78712-544-5.

[5] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, Huaimin Wang ; An Overview of Blockchain Technology

[6] Architecture, Consensus, and Future Trends (2017)IEEE 6th International Congress on Big Dana

[7] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, Charalampos Papamanthou; The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts; (2016) IEEE Symposium on Security and Privacy

[8] Max Thake, https://medium.com/nakamo-to/what-is-proof-of-stake-pos-479a04581f3a, siječanj 2019.

[9] Jake Frankenfield, https://www.investopedia.com/terms/i/initial-coin-offering-ico.asp, listopad 2018.

[10] Bernard Marr, https://www.forbes.com/sites/bernardmarr/2018/05/14/30-real-examples-of-blockchain-technology-in-practice/#55641ba9740d, ožujak 2019.

[11] Marie Huillet, https://cointelegraph.com/news/demand-for-blockchain-engineers-has-grown-400-since-end-of-2017-report-says, listopad 2018.

[12] Alex Lielacher https://www.bitcoinmarketjournal.com/biggest-icos-roi/, siječanj 2019.

[13] Digital Information World , Anastasia Stefanuk, https://www.digitalinformationworld.com/2018/05/top-5-countries-embracing-blockchain.html, ožujak 2019.

[14] Željko Ivanković, https://www.bug.hr/kriptovalute/sto-je-ico-initial-coin-offering-1127, listopad 2018.

[15] Hired, https://hired.com/state-of-salaries-2018, veljača 2019.

[16] Economic Graph Team, https://economicgraph.linkedin.com/en-us/research/linkedin-2018-emerging-jobs-report, veljača 2019.

[17] Coindesk, https://www.coindesk.com/research/state-blockchain-q1-2018, ožujak 2019.

[18] Udemy, https://www.udemy.com/learn-python/, ožujak 2019

Prilog

Na obrani završnog rada bit će priložena prezentacija s opisom izvedbe praktičnog dijela rada.



Algebra
visoka škola za
primijenjeno računarstvo

Blok Chain tehnologije u transakcijskim sustavima

Pristupnik: Tomislav Forko 0321003317

Mentor: Predavač Visoke škole Zlatan Morić