SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

ETHEREUM ALGORITAM

Diplomski rad

Ivan Kunsabo

Osijek, 2019.

SADRŽAJ

1. UVOD

Cilj ovog diplomskog rada je objasniti ethereum algoritam. Kako bi se ethereum algoritam pojasnio, prvo je potrebno objasniti što su kriptovalute i ethereum. Rad je podijeljen u tri cjeline: kriptovalute, ethereum i ethereum algoritam. U radu će biti objašnjeno što su kriptovalute i koje tehnologije koriste. Također, bit će objašnjeno što je Ethereum i princip rada Ethereuma. Zadnja cjelina odnosit će se na analizu ethereum algoritma. U njoj će biti govora o osnovama rada ethereum algoritma te nekim klasama ethereum mreže implementirane na ethereum-cpp klijentu te će biti napravljen uvod u programski jezik solidity.

1.1. Zadatak diplomskog rada

U ovom diplomskom radu potrebno je obraditi Ethereum algoritam. Potrebno je napisati postupak računanja Ethereuma kao i pojmove hash rate, blockchain i kriptovaluta. Nadalje, potrebno je objasniti zašto je ovaj algoritam pogodan za grafičke kartice. U radu je potrebno predočiti programsko rješenje i opisati njegove dijelove.

2. KRIPTOVALUTE

Kriptovalute su digitalni novac stvoren na internetu za internet. Kriptovaluta je digitalni novac u punom smislu riječi, odnosno kriptovalute su zapisi unutar programa koji je instaliran na više računala, koja su međusobno umrežena. Kad osoba uplati novac u banku i potom plaća karticom, ona zapravo rabi digitalni klasični novac, no to nije digitalni novac u punom smislu jer je stvoren i koristi se kao papirnati novac. Digitalni novac u klasičnoj banci ima drugačije karakteristike u odnosu na kriptovalute. Polaganjem novca u banku stvara se dužničko-vjerovnički odnos, odnosno banka postaje dužnik - ona je taj novac dužna vratiti na zahtjev vlasnika. Kod kriptovaluta nitko nikomu ništa ne duguje. Posjedovati jedinicu kriptovalute, može se opisati kao imati zrno zlata u ruci. Zlato može imati veću ili manju vrijednost, ali kao takvo ne predstavlja dug, isto je i s kriptovalutama.

Bitcoin je specifična pod-mreža na internetu i jedinica najpoznatije kriptovalute. To je sustav elektroničkoga plaćanja koji se zasniva na kriptografiji, odnosno šifriranju te odatle dolazi ime kriptovaluta. Šifriranje se koristi kako bi komunikacija između dviju osoba ostala privatna i nekompromitirana, iako u komunikacijskom kanalu postoje treće osobe koje tu komunikaciju mogu pratiti.

Ako tko želi sigurno poslati novac putem interneta, potrebna mu je pouzdana treća osoba, odnosno institucija u koju ima povjerenja da će novac prenijeti od A do B. U današnje vrijeme, te institucije su banke. Kod kriptovaluta, moguće je poslati novac bez posrednika i biti siguran da ga nitko putem neće ukrasti ili umanjiti iznos. Rješenje je u dijeljenom znanju u zajednici, u peer-to-peer mreži koja se temelji na open-source softveru, pri čemu je komunikacija vrlo snažno šifrirana.

Peer-to-peer mreža koncept je povezivanja računala bez središnje točke, odnosno bez centralnoga servera. Ondje svako računalo pronalazi i izravno komunicira s drugim računalima. Open-source softver je računalni program čiji je programski kod javno objavljen na internetu i svatko ga može i vidjeti i mijenjati. Temelji se na internetskoj zajednici, odnosno dobrovoljnoj suradnji većega broja ljudi, koji zajedno unaprjeđuju određeni softver na dobro svih korisnika. Cijela zajednica nadgleda prijedloge promjena i odobrava samo one koje smatra kvalitetnima.

Hash funkcija je funkcija koja za ulaz ima podatke proizvoljne veličine, a kao izlaz vraća podatke fiksne veličine. Izlaz funkcije naziva se još i hash vrijednost, dok se ulazni podatak

naziva poruka. Kriptografske hash funkcije nemaju inverz. Jedini način da se ulazni podatak dobije iz izlaznog podatka je pretraživanjem svih mogućih vrijednosti ulaza kako bi se vidjelo koji od ulaza odgovara izlazu koji posjedujemo.

Kriptografska hash funkcija koju kriptovalute koriste je SHA-256 (eng. Secure Hash Algorithm) te kada se povjeruje u neprobojnost šifre, a čak i ako se bude smatralo potrebno, kriptovalute se mogu prebaciti na još viši standard šifriranja, više nije nužno vjerovati osobi ili instituciji, dovoljno je povjerenje u sustav u cjelini.

Iako se kriptovaluta temelji na kriptografiji, samo šifriranje nije nikakva novost. To je poznati izum i postoje brojne aplikacije za sigurno komuniciranje putem interneta koje se temelje na korištenju javnoga i privatnoga ključa. Javni se ključ matematički izvodi, odnosno šifrira iz privatnoga ključa na način da je nemoguće iz javnoga ključa unazadnim postupkom doći do privatnoga ključa. Bitcoin mrežom ne šalju se šifrirane poruke za vojsku ili slično, nego se izvode transakcije. Također, transakcije kriptovalutama putem interneta temelje se na kriptografskom sustavu privatnoga i javnoga ključa.

Za transakciju kriptovalutama potrebno je da pošiljatelj i primatelj imaju bitcoin novčanike (eng. Wallet). Bitcoin novčanik je softver koji korisnik sam može besplatno instalirati na svoje računalo ili mobitel. Vlasnik bitcoin novčanika ima privatni ključ koji je samo njemu poznat i kojeg mora držati u tajnosti jer svaki posjednik privatnoga ključa može raspolagati kriptovalutama u njegovom novčaniku. Cijeli sustav je decentraliziran te u slučaju da netko izgubi privatni ključ ili isti bude ukraden, nema nikakve središnje institucije kojoj bi se korisnik mogao obratiti za zaštitu prava ili za povrat kriptovaluta.

Pošiljatelj kriptovalute šalje transakciju u mrežu i čeka na potvrdu transakcije. Bitcoin mrežu čine računala koja potvrđuju transakcije, a za nagradu plaćeni su kriptovalutama. Potvrđivanje transakcija naziva se i rudarenje (eng. Mining). Rudarenje je proces potvrđivanja i dodavanja novih transakcija u „glavnu knjigu“ koja se naziva blockchain.

2.1. Blockchain

Blockchain je rastuća lista digitalnih informacija koja je podijeljena između čvorova koji sudjeluju u sustavu. Ti zapisi nazivaju se blokovi te su oni povezani kriptografijom. Svaki blok sadrži hash funkciju prošlog bloka, vrijeme upisa i podatke tog zapisa. Po dizajnu, blockchain je otporan na promjene podataka. To je otvorena, distribuirana knjiga koja može učinkovito i na provjerljiv i trajan način evidentirati transakcije između dviju strana. Za upotrebu kao distribuirana knjiga, blockchain se koristi kod peer-to-peer mreža, koja se

kolektivno pridržava protokola za komunikaciju između čvora i provjere novih blokova. Jednom zabilježeni, podaci u bilo kojem bloku ne mogu se mijenjati retroaktivno bez promjene svih narednih blokova, što zahtijeva pristanak većine iz mreže. Iako zapisi o blockchainu nisu nepromjenjivi, blockchain se može smatrati sigurnim dizajnom i primjerom distribuiranog računalnog sustava s visokom tolerancijom na greške. Blockchain je u potpunosti decentraliziran, što znači da nema potrebe za središnjim autoritetom. Svaki novi zapis odvija se u gotovo realnom vremenu između mnoštva čvorova.

2.1.1. Struktura Bloka

Blockchain se sastoji od mnoštva blokova. Blok je struktura podataka u kojoj su zapisane digitalne informacije koje se dijele putem blockchaina. Blok se sastoji od:

- veličine bloka u bajtovima
- zaglavlja bloka u kojem se nalaze meta podaci o bloku
- broj zapisa koje blok sadrži
- zapisi koji su pohranjeni u bloku.

2.1.2. Zaglavlje bloka

U zaglavlju svakog bloka nalaze se meta podaci o bloku koji služe kao dodatne tehničke informacije o bloku i o povezivanju blokova u lanac. Struktura zaglavlja bloka sastoji se od:

1. Verzija
2. Hash prethodnog bloka
3. Korijen binarnog hash stabla
4. Vremenska oznaka
5. Težinska oznaka
6. Nonce.

Verzija se odnosi na verziju protokola u vrijeme nastajanja bloka, to je uglavnom specifično za Bitcoin mreže. Hash prethodnog bloka je referenca na prethodni blok u lancu, a korijen binarnog hash stabla je kriptografski hash koji sadrži informacije o svim zapisima u bloku. Vremenska oznaka je vrijeme kada je blok kreiran i uključen u blockchain, težinska oznaka je težina algoritma čije je rješenje potrebno za uključivanje bloka u blockchain, a nonce je broj pomoću kojeg je riješen algoritam za uključivanje bloka u blockchain.

2.1.3. Povezivanje blokova

Blokove u blockchainu možemo usporediti sa stranicama knjige. Na zaglavlju svake stranice se nalazi ime i broj poglavlja, a na dnu broj stranice. Zaglavlje jednog bloka u blockchain strukturi sadrži tehničke informacije o bloku, referencu na prethodni blok i hash svih podataka sadržanih u bloku dobiven uporabom funkcije hashiranja. Ukoliko bi netko istrgnuo sve stranice knjige lako bismo pomoću brojeva stranica zaključili u kojem ih redoslijedu trebamo čitati, isto tako možemo odrediti razmještaj blokova u lancu pomoću referenci na prethodni blok.

Uporabom hash funkcija umjesto vremenske oznake ili numeriranja dobivamo validaciju podataka. Svatko tko ima pristup podacima pojedinog bloka ili samo zaglavlju tog bloka može pomoću određene kriptografske funkcije odrediti hash tog bloka.

Blok koji je kasnije nastao pohranjuje hash bloka koji je neposredno prije njega uključen u lanac. Ako zapisan hash u blockchainu odgovara dobivenom hashu tada možemo biti sigurni da su podaci u bloku konzistentni. Ako netko pokuša napraviti izmjenu u podacima mora također mijenjati i sve hash-eve od tog trenutka nadalje. Time bi cijeli blockchain izgledao potpuno drugačije.

2.2. Novčanik

Novčanik (eng. Wallet) kriptovalute je aplikacija koja pohranjuje privatne i javne ključeve te omogućuje interakciju s raznim blockchainovima kako bi korisnici mogli slati i primati digitalne valute i nadzirati svoje trenutno stanje. Da bi se koristio Bitcoin ili bilo koja druga kriptovaluta, mora se imati digitalni novčanik.

Za razliku od tradicionalnih "džepnih" novčanika, digitalni novčanici ne pohranjuju valutu. Valute se ne pohranjuju ni na jednom mjestu niti postoje bilo gdje u bilo kojem fizičkom obliku. Sve što postoji jesu zapisi o transakcijama pohranjenim na blockchain mreži.

Kada osoba želi poslati bitcoine ili bilo koju drugu vrstu kriptovalute, ona otpisuje vlasništvo nad valutama koje su pripisane na adresi digitalnog novčanika. Privatni ključ pohranjen u novčaniku mora odgovarati javnoj adresi kojoj je valuta dodijeljena. Ako se javni i privatni ključevi podudaraju, suma u digitalnom novčaniku će se povećati, a pošiljatelju će se smanjiti u skladu s tim. Transakciju označavaju samo zapis transakcije na blockchainu i promjena stanja u digitalnom novčaniku.

Postoje razne vrste digitalnih novčanika:

- Desktop – novčanici koji su instalirani na računalu
- Online – novčanici koji su na oblaku podataka
- Mobile – novčanici koji su instalirani na mobitelu
- Hardware – novčanici instalirani na USB ili sličnom uređaju

Bez obzira koji se novčanik koristi, gubitak privatnog ključa dovesti će do gubitka kriptovalute. Ako se novčanik hakira, nema načina da se vrati izgubljena valuta ili se poništi transakcija. Razina sigurnosti ovisi o vrsti novčanika koji se koristi i o davatelju usluge. Online je rizičnije okruženje za održavanje novčanika u usporedbi s offline. Online novčanici izloženiji su napadima hakera jer su na internetskoj mreži. Offline novčanike ne može se hakirati jer jednostavno nisu povezani s internetom i za sigurnost se ne oslanjaju na treću stranu.

2.3. Digitalni potpis

Povjerenje je prosudba rizika između različitih strana, a u digitalnom svijetu određivanje povjerenja često se svodi na dokazivanje identiteta (autentifikacija) i dokazivanje dozvola (autorizacija). Odnosno, želi se odrediti: "Jesi li ti za koga kažeš da jesi?" i "Trebali biste biti u stanju raditi ono što pokušavate?".

U slučaju blockchain tehnologije kriptografija privatnog ključa pruža snažan alat za vlasništvo koji ispunjava zahtjeve za provjeru autentičnosti. Posjedovanje privatnog ključa je vlasništvo. Također, osoba ne mora dijeliti više osobnih podataka nego što bi joj bilo potrebno za razmjenu te ih izložiti hakerima. Međutim, samo autentifikacija nije dovoljna. Zbog toga je potrebna i autorizacija jer se pomoću nje dobiva podatak ima li osoba dovoljno novca te se pokreće ispravna vrsta transakcija. Za rad autorizacije potrebna je distribuirana, peer-to-peer mreža kao početna točka. Distribuirana mreža smanjuje rizik od centralizirane korupcije ili neuspjeha.

Ova distribuirana mreža također mora biti posvećena evidenciji i sigurnosti transakcijske mreže. Autoriziranje transakcija rezultat je cijele mreže koja primjenjuje pravila po kojima je dizajnirana - blockchain protokol.

Autentifikacija i autorizacija pružena na ovaj način omogućuje interakcije u digitalnom svijetu bez oslanjanja na povjerenje treće strane.

2.4. Rudarenje

Rudarenje (eng. Mining) ima važnu ulogu u osiguravanju rada kriptovaluta. Mnogi misle da je jedina svrha rudarstva generiranje tokena na način za koji ne treba središnji izdavač.

Banke su obično zadužene za vođenje točne evidencije transakcija. One osiguravaju da novac ne bude stvaran iz zraka i da korisnici ne varaju te ne troše svoj novac više od jednom. Blockchain tehnologija uvodi potpuno novi način vođenja evidencije, takav gdje cijela mreža, a ne posrednik, provjerava transakcije i dodaje ih u javnu knjigu. Iako je cilj stvoriti sustav „bez povjerenja“, netko i dalje mora osigurati financijsku evidenciju osiguravajući tako da ga nitko ne vara. Rudarstvo je jedna od inovacija koja omogućava decentralizirano vođenje evidencije. Rudari su postigli dogovor o povijesti transakcija, istovremeno sprečavajući varanje. Posebno se to odnosi na dvostruko trošenje novca, problem koji još nije riješen u centraliziranim sustavima.

Dokaz rada (eng. Proof of Work) je protokol kojem je glavni cilj odvraćanje od cyber napada poput rasprostranjenog usporavanja usluge koje ima svrhu iscrpiti resurse računalnog sustava slanjem više lažnih zahtjeva. Koncept dokaz rada postojao je i prije bitcoina te on revolucionarno mijenja način transakcija koje su danas standardne. Siguran i distribuiran sustav dogovora donosi mogućnost primanja i davanja novca bez vjerovanja u usluge trećih strana. Kada se koriste tradicionalni načini plaćanja mora se postaviti povjerenje trećoj strani za postavljanje transakcije. Uz bitcoin i nekoliko drugih digitalnih valuta svi imaju primjerak knjige i sve transakcije od početka, tako da nitko ne mora vjerovati trećim stranama jer svatko može izravno provjeriti napisane podatke.

Dokaz rada je uvjet da se definira rudarenje, koji se treba izvesti kako bi se stvorila nova skupina sigurnih blokova na blockchainu. Princip rada jedne transakcije je:

- Transakcije su sjedinjene u blokove.
- Rudari potvrđuju da su transakcije unutar svakog bloka ispravne.
- Da bi to učinili, rudari trebaju riješiti matematičku zagonetku poznatu kao problem s dokazom rada.
- Nagrada se dodjeljuje prvom rudaru koji riješi problem.
- Ovjerene transakcije pohranjuju se u javni blockchain.

Matematička zagonetka koju rudari pokušavaju riješiti je dokaz rada. Transakcija mora biti dovoljno teška na strani podnositelja zahtjeva, a lagana za provjeriti na mreži. Svi rudari

mreže natječu se da bi bili prvi koji će pronaći rješenje za matematički problem koji se tiče trenutnog kandidiranog bloka. Problem se ne može riješiti na drugi način nego grubom silom, na način da zahtijeva veliki broj pokušaja. Kad rudar napokon nađe pravo rješenje, istovremeno ga najavljuje cijeloj mreži te primi nagradu u obliku kriptovalute u visini predviđenoj protokolom. Postupak rudarenja je operacija obrnutog hashiranja: ona određuje vrijednost nonce pa algoritam kriptografske hash funkcije nad podacima daje za rezultat blokove manje od određenog praga.

Ethereum mreža želi preći na novi sustav dogovora zvan dokaz udjela (eng. Proof of Stake). U dokazu udjela umjesto rudara postoje validatori. Validatori daju nešto svog ethera kao udio u ekosustavu. Nakon toga, validatori se klade na blokove za koje smatraju da će biti dodani na kraj blockchain lanca. Kada se doda blok, validatori dobivaju nagradu za blok razmjerno njihovom udjelu. Dokaz udjela daje za rezultat zeleniji i jeftiniji distribuirani oblik potvrda transakcija. Algoritam dokaz udjela mora biti što je moguće otporniji na vanjske napade jer bi mreža koja se temelji na algoritmu dokaza udjela bila mnogo jeftinija resursima za neželjeni napad, nego mreža temeljena na dokazu rada. Da bi se riješilo sigurnosno pitanje grupa programera stvorila je Casper protokol, dizajnirajući algoritam koji može koristiti skup nekih okolnosti pod kojima loš validator može izgubiti svoj depozit. Casper protokol može zahtijevati da validatori podnesu uloge za sudjelovanje te im njihovi ulozi mogu biti oduzeti ako protokol utvrdi da su postupali na neki način koji krši određeni skup pravila. Također, Casper protokol će odrediti iznos nagrade koju su primili validatori zahvaljujući kontroli nad sigurnosnim ulozima. Primjerice, ako jedan validator stvori "nevažeći" blok njegov sigurnosni ulog bit će izbrisan, kao i njegova privilegija da bude dio mreže. Casper-ov sigurnosni sustav zasnovan je na principu sličnom okladama. U sistemu sa sustavom dokaz udjela oklade su transakcije koje će, prema pravilima uloga, nagraditi svog validatora novčanom nagradom zajedno sa svakim blokom na kojeg se validator kladio.

2.5. Hashrate

Hashrate je mjera koja je povezana s algoritmom rudarenja na temelju dokaza rada koji se koristi u različitim kriptovalutama. Pri rudarstvu mora se pronaći određeni hash kreiran pomoću hash funkcije (npr. EthHash, SHA256). Da bi se uspješno izrudario blok, mora se puno puta pokrenuti zadana hash funkcija.

Također, hashrate se može opisati kao mjera za količinu energije koju mreža troši da bi bila kontinuirano funkcionalna. Pod stalnim funkcioniranjem podrazumijeva se koliko snage troši

hash za pronalaženje blokova u normalnom srednjem vremenu. Kod bitcoin mreže to je vrijeme od 10 minuta.

Da bi se blok uspješno rudario, rudar treba usmjeriti zaglavlje bloka na takav način da je manje ili jednako određenom cilju. Cilj je da SHA-256 hash zaglavlje bloka, koje mora biti 256-bitni niz, mora početi s određenim brojem nula. Broj nula na početku predstavlja težinu rudarenja.
Rudari dolaze do posebnog hash-a mijenjanjem malog dijela zaglavlja bloka, koji se naziva "nonce". Nonce vrijednost uvijek počinje s "0" i svaki put se povećava za dobivanje potrebnog hash-a. Određivanje nonce vrijednosti radi se na temelju pogotka i promašaja pa su šanse da se dobije poseban hash, koji započinje s puno nula, vrlo male. Zbog toga, rudar mora izvesti puno sličnih pokušaja mijenjanjem vrijednosti nonce.

U bitcoin mreži, hashrate koji rudar ima tijekom rudarenja na mreži, u odnosu na hasrate svih rudara u mreži, iznosi njegovu vjerojatnost da će uspješno doći do hasha i zaraditi nagradu svakih 10 minuta. Primjerice, ako cijela mreža ima snagu hashrate od 1000, a rudar ima snagu od 10, njegov postotak da će osvojiti nagradu je 1%.

3. ETHEREUM MREŽA

3.1. Ethereum

Prije stvaranja Ethereuma, blockchain aplikacije dizajnirane su za obavljanje vrlo ograničenog niza operacija. Primjerice, bitcoin i druge kriptovalute razvijene su isključivo da djeluju kao peer-to-peer digitalne valute.

Vitalik Buterin došao je na ideju kako unaprijediti bitcoin. Odnosno, omogućiti ne samo slučajeve financijske uporabe koje dopušta bitcoin, nego i puno veće stvari koje mogu služiti bilo kojoj svrsi koja bi se dala isprogramirati. U 2013. je objavio Bijelu knjigu (eng. White paper) koja opisuje platformu dizajniranu za bilo koju vrstu decentralizirane aplikacije. Ta platforma zvala se ethereum.

Slika 3.1. Logo ethereum mreže.

Na slici 3.1. vidi se kako je ethereum uobičajeno prikazan na internetu i u literaturi.

Ethereum je open-source program koji se temelji na blockchain tehnologiji te sadrži funkcionalnost pametnih ugovora (eng. Smart Contracts). Programerima omogućuje izradu blockchain aplikacija koje istodobno iskorištavaju visoku dostupnost Ethereum mreže, a pošto se koristi blockchain tehonologija, aplikacije su decentralizirane.

Serveri i oblaci podataka pomoću ethereuma bili bi zamijenjeni s tisućama takozvanih čvorova (eng. Nodes) koji se izvode na svim računalima u ethereum mreži te zajedno formiraju „Globalno računalo“ (eng. World Computer) koje bi decentraliziralo trenutni klijent–server model. Ideja je da se omogući da samo vlasnik podataka može raditi promjene nad svojim podacima. Pomoću ethereuma izbjegava se da vlasnik servera ili oblaka podataka može pristupiti privatnim podacima korisnika i raditi s njima bilo kakve radnje. U teoriji, ethereum je kombinacija laganog pristupa informacijama koji postoji u novom digitalnom dobu i kontrole nad informacijama koje su ljudi imali u prošlosti. Na primjer, svaki puta kada bi promijenili, dodali ili obrisali bilješku ili dokument, svaki čvor u mreži napravio bi promjene.

3.2. Ether

Ether je token čiji se blockchain generira ethereum platformom. Ether se može prenijeti između računa i koristiti za kompenzaciju sudionika rudarenja za izvedene transakcije. Ether opskrbljuje EVM. Služi za plaćanje goriva (eng. Gas), obračunske jedinice koja se koristi u transakcijama i za promjene u drugim stanjima. Upotrebljava se i za plaćanje računalnih resursa potrebnih za pokretanje aplikacije ili programa na ethereum mreži.

Kao kriptovaluta ether je naveden pod oznakom ETH, a simbol valute prikazan je grčkim znakom Xi (≡). Poput bitcoina, ether je sredstvo digitalnog donositelja, odnosno slično kao vrijednosni papir, poput obveznice, izdan u fizičkom obliku. Baš kao ni kod gotovine, nije potrebna treća strana da obrađuje ili odobrava transakciju.

Slika 3.2. Simbol i oznaka ethera.

Na slici 3.2. prikazan je simbol i oznaka digitalnog tokena ethereum mreže, ethera.

Gwei je manja novčana jedinica za ether, kripto „novčić“ koji se koristi u ethereum mreži.

1 Ether = 1.000.000.000 Gwei

Slično raznim kovanicama male vrijednosti, koje su u opticaju za stvarne valute poput penija ili hrvatskih lipa, nekoliko je novčića postalo popularno u svijetu kriptovalute kako bi se odnosili na različite frakcijske vrijednosti određenog tokena.

3.3. Ethereum blockchain

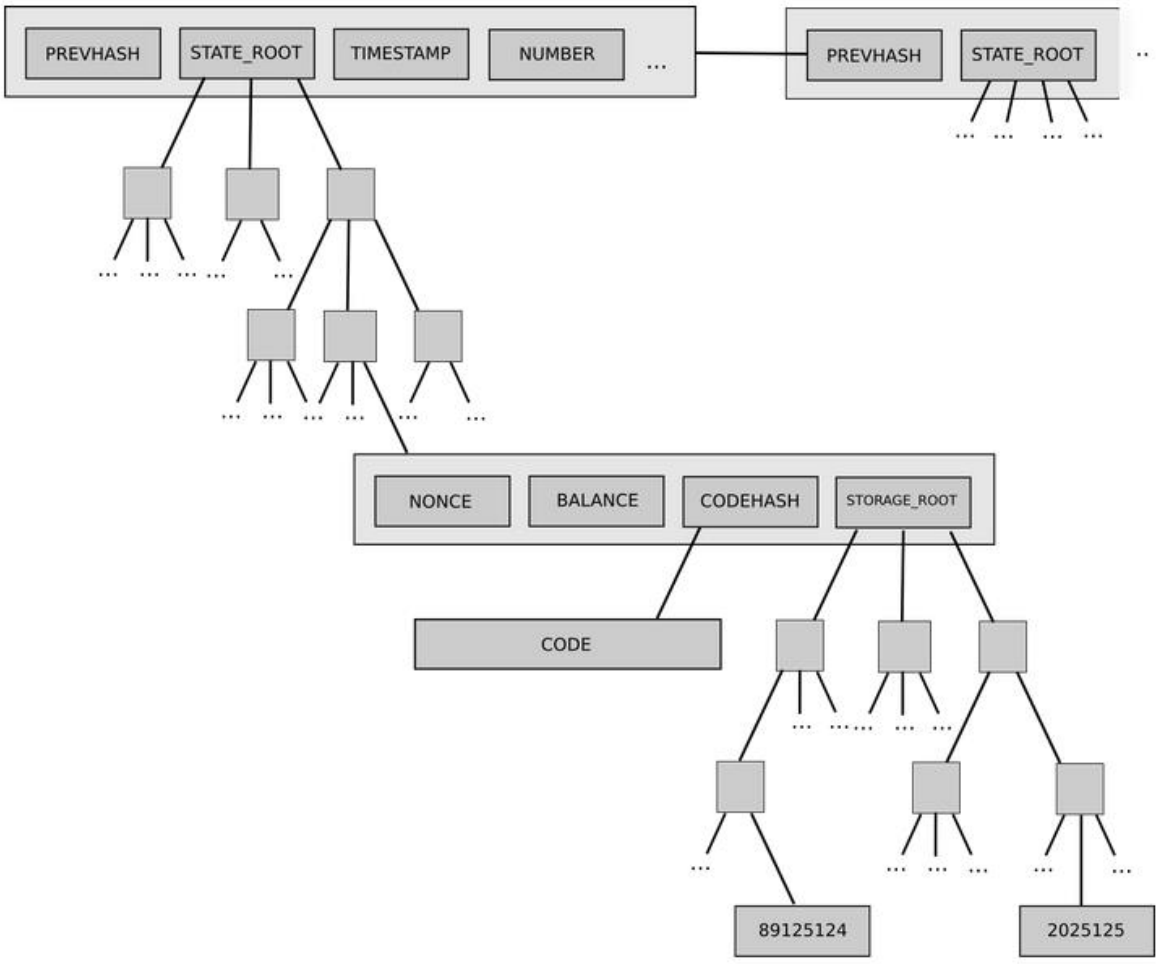
Ethereum blockchain je decentralizirana, masovno replicirana baza podataka u kojoj je pohranjeno trenutno stanje svih računa. Može se opisati kao blockchain s ugrađenim programskim jezikom ili kao globalno izveden virtualni stroj temeljen na protokolu. Dio protokola koji zapravo obrađuje unutarnje stanje i računanje naziva se Ethereumov virtualni stroj (EVM).

Slika 3.3. Temeljne tehnologije blockchaina.

Na slici 3.3. prikazane su tehnologije od kojih je napravljen ethereum blockchain. Sličan je kao i blockchaini kod ostalih kriptovaluta. Kriptografski je siguran, što znači da je stvaranje digitalne valute osigurano složenim matematičkim algoritmima koje je teško razbiti. Skoro da je nemoguće stvarati lažne transakcije, brisati transakcije ili slično. Ethereum blockchain je transakcijski singleton stroj što znači da postoji jedna kanonička instanca stroja odgovorna za sve transakcije stvorene u sustavu. Odnosno, postoji jedna globalna istina u koju svi vjeruju te blockhain ima podijeljena stanja sa cijelom mrežom što znači da je stanje pohranjeno na ovom računalu, kao i na svakom drugom, zajedničko i otvoreno za sve. Rezultat je sustav za digitalne interakcije kojemu nije potrebna treća strana. Način osiguravanja digitalnih povezivanja je implicitan, podupiran je jednostavno, ali je i robustan zbog arhitekture blockchain tehnologije.

Ethereum blockchain koristi bazu podataka za pohranu svih računa koja se naziva stablo Patricia. Stablo Patricia je u osnovi specijalizirana vrsta Merkleovog stabla koje djeluje kao generički ključ ili skladište vrijednosti. Kao i standardno Merkle stablo, i stablo Patricia ima "korijenski hash" koji se može koristiti za označavanje cijelog stabla, a sadržaj stabla ne može se mijenjati bez promjene hash korijena. Za svaki račun stablo pohranjuje četiri varijable koje sadrže: account_nonce, ether_balance, code_hash, storage_root, gdje je account_nonce broj transakcija poslanih s računa i on se čuva radi sprečavanja napada ponovnog ponavljanja, ether_balance je stanje računa, code_hash je hash koda ako je račun pametni ugovor, u suprotnom vrijednost su prazni dvostruki navodnici (""), a storage_root je korijen još jednog stabla Patricia koji pohranjuje podatke o pohrani.

Svake minute rudar proizvodi novi blok, a taj blok sadrži popis transakcija koje su se dogodile od posljednjeg bloka i korijenski hash stabla Patricia koji predstavlja novo stanje nakon primjene tih transakcija i davanja rudaru etherske nagrade za stvaranje bloka. Zbog načina na koji drvo Patricia funkcionira, ako se izvrši mala promjena, većina dijelova stabla bit će potpuno ista kao u posljednjem bloku. Prema tome, nema potrebe dvaput pohranjivati podatke jer će se čvorovi u novom stablu jednostavno moći usmjeriti na istu memorijsku adresu koja pohranjuje čvorove starog stabla na mjestima gdje su novo stablo i staro stablo potpuno ista stabla. Ako se tisuću komada podataka promijeni između bloka N i bloka N + 1, čak i ako je ukupna veličina stabla puno gigabajta, količina novih podataka koja treba biti spremljena u blok N + 1 je najviše nekoliko stotina kilobajta ili često znatno manja, posebno ako se unutar istog ugovora dogodi više promjena. Svaki blok sadrži hash prethodnog bloka te je to ono što čini da blok postavi lanac kao i pomoćne podatke poput broja bloka, vremenske oznake, adrese rudara i ograničenja količine goriva.



Slika 3.4. Grafički prikaz ethereum blockchaina.

Slika 3.4. prikazuje grafički izgled ethereum blockchaina. Blok se sastoji od varijabli vrijednosti prošlog hasha, od trenutnog stanja korijena, vremenske oznake i broja koji su povezani u lance. Trenutno stanje korijena širi se u obliku Patricia stabla.

3.4. Računi

Ethereum mreža se može smatrati velikim decentraliziranim računalom koje sadrži milijune objekata, nazvanih računi (eng. Accounts), koji imaju mogućnost održavanja blockchaina, izvršavanja koda i međusobnog razgovora. Postoje dvije vrste računa. Prvi je račun u vanjskom vlasništvu (eng. Externally Owned Account). Odnosno, račun za kojeg je potreban privatni ključ koji je povezan s EOA, taj račun ima mogućnost slanja ethera i poruka. Druga vrsta računa je ugovor. Račun koji ima vlastiti programski kod i upravljan je programskim kodom.

3.5. Pametni ugovori

Pametni ugovor (eng. Smart contract) samo je fraza koja se koristi za opisivanje računalnog koda koji može olakšati razmjenu novca, sadržaja, imovine, dionica ili bilo čega vrijednog. Kada se izvodi na blockchainu, pametni ugovor postaje poput ugrađenog računalnog programa koji se automatski izvršava kada su ispunjeni određeni uvjeti. Budući da pametni ugovori rade na blockchainu, oni se odvijaju točno onako kako su programirani bez ikakve mogućnosti promjene, zastoja, prijevara ili uplitanja trećih strana.

Iako sve blockchain tehnologije imaju mogućnost obrade koda, većina ih je strogo ograničena. Ethereum omogućava programerima da stvaraju sve što žele jer ne daje određen skup ograničenih operacija.

Prema zadanim postavkama, ethereum mreža je beživotna, odnosno ništa se ne događa i stanje svakog računa ostaje isto. Međutim, svaki korisnik može pokrenuti akciju slanjem transakcije s EOA računa i tako pokrenuti ethereum mrežu. Ako je odredište transakcije drugi EOA račun, transakcija može prenijeti neki ether, ali u suprotnom ne radi ništa. Ali ako je odredište pametni ugovor, tada se pametni ugovor aktivira i automatski pokreće svoj kod. Kod ima mogućnost čitanja i pisanja u vlastitu unutarnju pohranu podataka (baza podataka koja preslikava 32-bajtne ključeve u 32-bajtne vrijednosti), mogućnost čitanja pohrane podataka primljene poruke i slanja poruka drugim pametnim ugovorima, što pokreće njihovo izvršavanje. Nakon što se izvršenje zaustavi i sva podizvršenja pokrenuta porukom koju je poslao pametni ugovor zaustave, na način da se sve događa u determiniranom i sinkronom redoslijedu, podpoziv se potpuno dovršava prije nego što se nadređeni poziv nastavi. Daljnje

okruženje izvršenja se zaustavlja, sve dok se ne pokrene nova transakcija koja će ponovno pokrenuti kod pametnog ugovora.

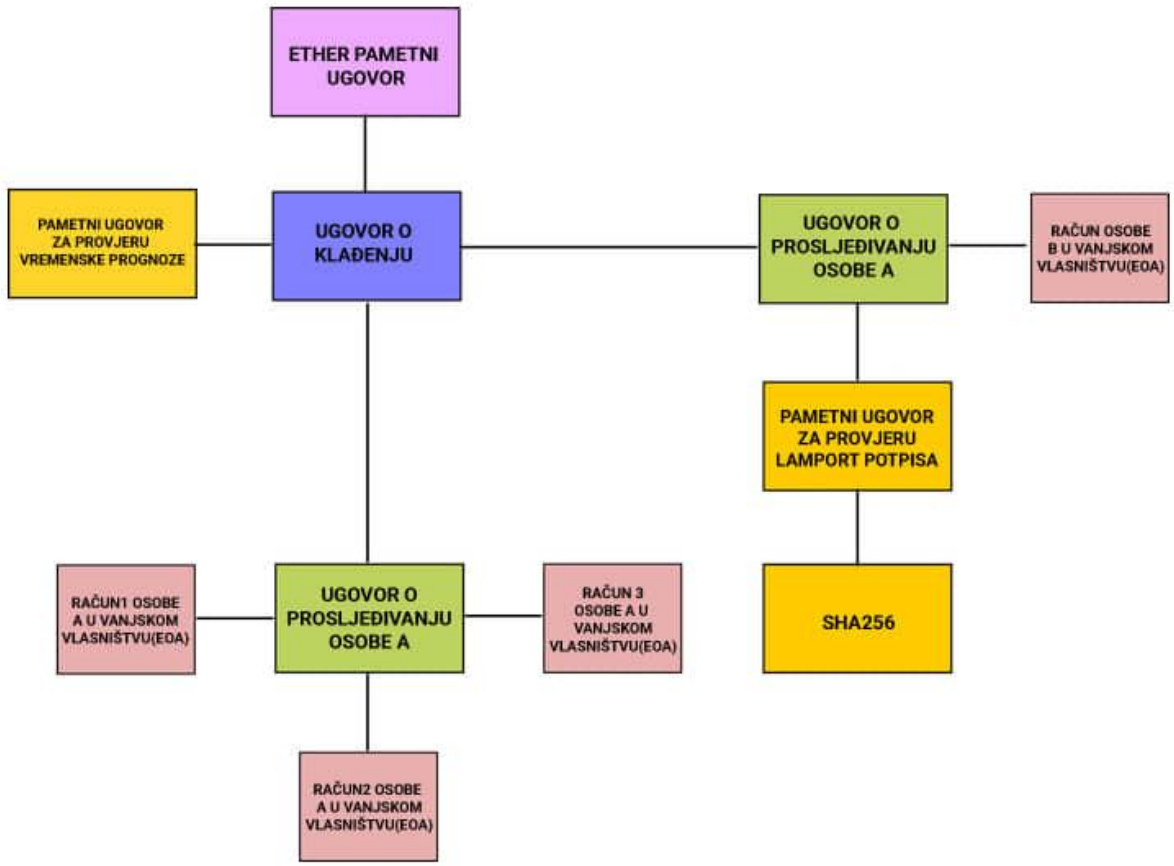
Ugovori uglavnom imaju četiri svrhe:

1. Zadržavanje podataka koji predstavljaju nešto što je korisno ili podatke za druge pametne ugovore ili za vanjski svijet. Primjer za to je ugovor koji simulira valutu, a drugi je ugovor koji bilježi članstvo u određenoj organizaciji.
2. Služe kao svojevrsni račun izvana u vlasništvu složenijih pravila pristupa. Kao takav, naziva se i "ugovor o prosljeđivanju" i obično uključuje jednostavno slanje dolaznih poruka na neko željeno odredište samo ako su ispunjeni određeni uvjeti. Na primjer, može postojati ugovor o prosljeđivanju koji čeka da dva od tri zasebna ključa potvrde određenu poruku prije nego što je ponovo pošalju. Složeniji ugovori o prosljeđivanju imaju različite uvjete temeljene na prirodi poslane poruke.
3. Upravljanje tekućim ugovorom ili odnosima između više korisnika. Primjeri toga uključuju financijski ugovor, depozit s nekim određenim nizom posrednika ili neku vrstu osiguranja. Također, može postojati otvoren ugovor s kojim jedna strana u bilo kojem trenutku ostavlja otvorenim za angažiranje bilo koje druge strane. Primjer za to je ugovor koji automatski plaća naknadu onome tko podnese valjano rješenje nekog matematičkog problema ili dokaže da pruža neki računalni resurs.
4. Pružanje funkcija drugim ugovorima te u osnovi služi kao softverska biblioteka.

Ugovori međusobno djeluju kroz aktivnost koja se naizmjenično naziva transakcija ili slanje poruka. Poruka je objekt koji sadrži određenu količinu ethera, niz bajtova podataka bilo koje veličine, adrese pošiljatelja i primatelja. Kad ugovor primi poruku ima mogućnost vratiti neke podatke koje izvorni pošiljatelj poruke može odmah upotrijebiti. Na ovaj je način slanje poruke točno poput pozivanja neke funkcije.

Kao primjer, uzmimo situaciju u kojoj se osoba A i osoba B klade na 10 ethera da temperatura u Zagrebu u sljedećoj godini neće preći 35 ° C. No, osoba A je vrlo sigurnosno oprezna i kao svoj primarni račun koristi ugovor o prosljeđivanju koji šalje samo poruke s odobrenjem dva od tri privatna ključa. Osoba B je paranoična u pogledu kvantne kriptografije, pa koristi ugovor o prosljeđivanju koji prosljeđuje samo poruke koje su potpisane Lamport znakovima na temelju SHA256, što nije podržano u ethereumu izravno. Sam ugovor o klađenju mora dohvatiti podatke o vremenu u Zagrebu iz nekog pametnog ugovora, a također treba

razgovarati s ether ugovorom kad zapravo treba proslijediti ether ili osobi A ili osobi B na njihove ugovore o prosljeđivanju. Na slici 3.5. prikazani su odnosi između računa.



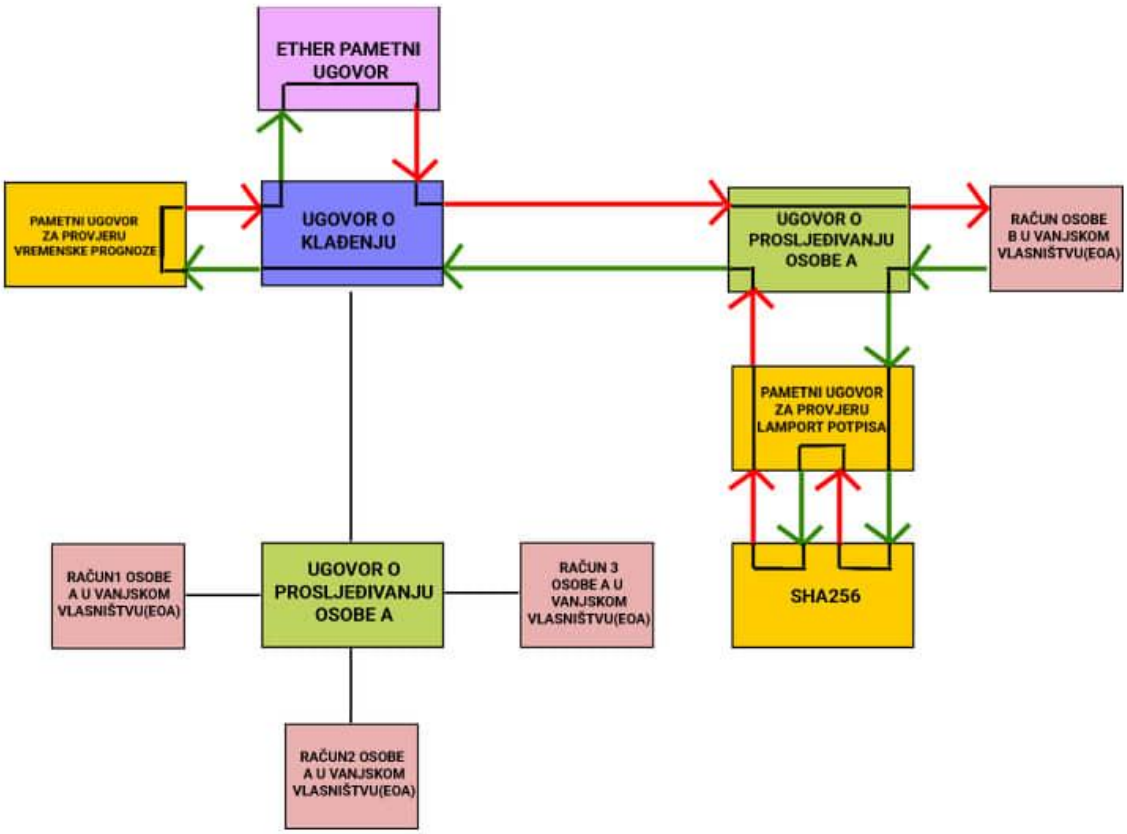
Slika 3.5. Odnosi između računa.

Kada osoba B želi dovršiti okladu, događaju se sljedeći koraci:

1. Poslana je transakcija, što izaziva poruku s EOA od osobe B na ugovor o prosljeđivanju osobe B.
2. Ugovor o prosljeđivanju od osobe B šalje hash poruke i Lamport potpis na pametni ugovor koji funkcionira kao biblioteka za provjeru potpisa Lamport.
3. Pametni ugovor za provjeru potpisa vidi da osoba B želi Lamport potpis temeljen na SHA256 pa poziva SHA256 biblioteku onoliko puta koliko je potrebno da provjeri potpis.
4. Nakon što pametni ugovor za provjeru potpisa Lamport vrati 1, što znači da je potpis provjeren, pametni ugovor osobe B šalje poruku ugovoru o klađenju koji predstavlja okladu.

5. Ugovor o okladi provjerava ugovor koji pruža temperaturu u Zagrebu da vidi koja je temperatura.
6. Ugovor o okladi vidi da odgovor na poruke pokazuje da je temperatura iznad 35 ° C pa šalje poruku ether ugovoru da prebaci ether sa svog računa na ugovor o prosljeđivanju osobe B.

Svagdje gdje se ether spominje, ether je "pohranjen" kao unos u bazi podataka ether ugovora, riječ "račun" u kontekstu koraka 6 znači da postoje podaci u varijabli data u ether ugovoru gdje su ti podaci ključ adrese za adresu oklade i vrijednosti za njegovo stanje. Nakon primitka poruke, ether ugovor smanjuje vrijednost za neki iznos i povećava vrijednost unosa koji odgovara adresi ugovora o prosljeđivanju osobe B. Te korake u obliku dijagrama možemo vidjeti na slici 3.6.



Slika 3.6. Dijagram poruka u slučaju da osoba B želi dovršiti okladu.

3.6. EVM (Ethereum Virtual Machine)

Ethereum Virtual Machine (EVM) je cjelovit softver koji radi na ethereum mreži. Omogućuje bilo kome tko je na ethereum mreži pokretanje bilo kojeg programa, bez obzira na programski jezik.

Tablica 3.1. EVM implementacije – njihovo ime i programski jezik u kojem su napisane.

Ethereum Virtual network implementacije		
EVM implementacije	Ime	Programski jezik
Rade na mreži	go - ethereum	go
	parity	rust
	cpp-ethereum	c++
	pyethereum	python
	py-EVM	python
	ethereumJ	Java
Ostale implementacije	SputnikVM	rust
	ruby-ethereum	ruby
	ethereumjs	javaScript

Tablica 3.1. prikazuje neke od najpopularnijih EVM implementacija. Odnosno, klijenti koji su napravljeni na temelju ethereum protokola.

EVM čini postupak stvaranja blockchain aplikacija mnogo lakšim i učinkovitijim jer se ne mora izgraditi potpuno originalan blockchain za svaku novu aplikaciju, nego se sve aplikacije mogu vrtiti na jednoj platformi. Omogućuje čvorovima ethereuma da zapravo pohranjuju i obrađuju podatke u zamjenu za plaćanje, reagirajući na događaje u stvarnom svijetu.

Prednosti koje donosi ethereum platforma su:

- Nepromjenjivost - treća strana ne može izvršiti nikakve promjene u podacima.
- Zaštita protiv korupcije i neovlaštenosti - aplikacije se temelje na mreži formiranoj oko određenog protokola, što onemogućuje cenzuru.
- Sigurnost – nema središnje točke kvara i podaci su osigurani pomoću kriptografije te su samim time aplikacije dobro zaštićene od hakerskih napada i lažnih aktivnosti.
- „Zero downtime“- aplikacije se nikad ne isključuju i nikad se ne mogu isključiti.

Sve te prednosti su omogućene zbog korištenja blockchain tehnologije.

Tablica 3.2. Popis nekih od debuggera i analizatora koda.

Debuggeri i analizatori koda		
	IME	OPIS
Debuggeri	REMIX	IDE koji ima EVM kod debugger
	debug_traceTransaction method	Poučni debugger koji pruža informacije iz mreže, omogućen u go-ethereum implementaciji
	Ethereum function signature database	Baza podataka za dešifriranje, primjerice 0x165ffd10 u ponovno pokretanje (32 - bytes, 32 - bytes)
	moesif web3 debugging	Snima Web3 transakcije iz decentraliziranih aplikacija uživo i dekodira ih sastavljenim ABI za uklanjanje pogrešaka i nadgledanje.
Analizatori koda	Echidna	analizator koda koji uzima i solidity ulaz
	MAIAN	Automatski alat koji otkriva manje ranjivosti pomoću prvog dubinskog pretraživanja simboličkog izvršenja više poziva
	Mythril	Alat za istraživanje blok-lanca koji indeksira sve ugovore na mreži, koji sadrži rastavljač, detektor ABI funkcije i analizator kontrolnog protoka
	porosity	Obrnuti inženjerski alat, dissasembler, detektor ABI funkcije i dekompajler koji ističe ranjivosti

Usprkos tome što donose brojne prednosti, decentralizirane aplikacije nisu besprijekorne. Tablica 3.2. prikazuje neke od debuggera i analizatore koda za EVM mreže radi lakšeg programiranja i uklanjanja grešaka. Budući da kod pametnih ugovora pišu ljudi, pametni ugovori su podjednako dobri kao i ljudi koji ih pišu. Greške u kodovima ili previdi mogu dovesti do nenamjernih štetnih radnji. Ako se pogreška u kodu iskoristi u negativne svrhe, ne postoji učinkovit način na koji se može zaustaviti napad ili iskorištavanje osim dobivanja mrežnog dogovora i prepisivanja temeljnog koda, a to ide suprotno osnovi blockchain tehnologije koja bi trebala biti nepromjenjiva.

Tablica 3.3. Programski jezici koji se kompajliraju u EVM.

Programski jezici koji se kompajliraju u EVM	
IME	OPIS
Solidity	Najpopularniji programski jezik za ethereumove pametne ugovore
Vyper	Programski jezik sa provjerom "overflowa", koristi numeričke jedinice, ali bez beskonačnih petlji
Flint	Programski jezik sa nekoliko sigurnosnih značajki, primjerice: tip podatka sa ograničenim setom operacija

U tablici 3.3. vide se novi programski jezici koji su stvoreni kako bi radili u EVM mreži. Iako se čini da su aplikacije moguće, nejasno je koji će se programi zaista pokazati korisnim i sigurnim te hoće li ikada biti jednako prikladni za upotrebu kao i aplikacije koje danas koristimo.

3.6.1. State machine

State machine je pametni ugovor koji služi kao registar imena. Izračunavanje u EVM-u vrši se korištenjem stack-based bytecode jezika koji je poput križanja između Bitcoin Script-a i tradicionalnog assemblerskog programiranja i Lispa zbog rekurzivne funkcionalnosti slanja

poruka. Svrha ovog konkretnog ugovora je služiti kao registar gdje bilo tko može poslati poruku koja sadrži 64 bajta podataka, 32 bajta za ključ i 32 bajta za vrijednost. Ugovor provjerava je li ključ već registriran u pohrani, a ako još nije, onda ugovor registrira vrijednost na tom ključu.

Za vrijeme izvođenja održava se beskonačno proširivi niz bajtova nazvanih memorija (eng. Memory), programski brojač koji upućuje na trenutnu instrukciju i niz od 32-bajtnih vrijednosti. Na početku izvršenja memorija i stack su prazni. Program u EVM-u je redoslijed naredbi. Primjer jednog dijela programa:

```
PUSH1 0 CALLDATALOAD SLOAD NOT PUSH1 10 JUMPI STOP JUMPDEST PUSH1 32
CALLDATALOAD PUSH1 0 CALLDATALOAD SSTORE
```

Postoji high-level jezik posebno dizajniran za pisanje ugovora, poznat kao Solidity, da bi olakšao pisanje ugovora. Postoje i drugi jezici, ovisno o preferencijama korisnika. Svaki kod koji se piše na tim jezicima sastavlja se u EVM-u i za stvaranje ugovora šalje transakciju koja sadrži EVM bytecode.

Postoje dvije vrste transakcija: transakcija slanja i transakcija koja stvara ugovor. Transakcija slanja je standardna transakcija koja sadrži adresu za primanje, etherski iznos, podatkovne bajtove, neke druge parametre i potpis privatnog ključa koji je povezan s računom pošiljatelja. Transakcija koja stvara ugovor izgleda kao standardna transakcija, osim što je adresa za primanje prazna. Kada ugovor koji stvara transakciju dođe do blockchaina, podatkovni bytecode u transakciji tumači se kao EVM kod, a vrijednost vraćena izvršenjem EVM-a uzima se kao kod novog ugovora.

3.6.2. Gorivo

Jedan važan aspekt načina na koji EVM radi je taj da svaku pojedinačnu operaciju koja se izvodi unutar EVM-a istovremeno izvršava svaki puni čvor. Ovo je nužna komponenta protokolnog modela ethereum mreže i ima prednost što svaki ugovor o EVM-u može pozvati bilo koji drugi ugovor po gotovo besplatnim troškovima, ali ima i nedostatak što su računski koraci na EVM-u vrlo skupi. Prihvatljiva upotreba EVM-a uključuje pokretanje poslovne logike, „ako je to onda to“, i provjeru potpisa i drugih kriptografskih objekata. Na gornjoj granici su aplikacije koje provjeravaju dijelove drugih blockchaina (npr. decentralizirana razmjena etera u bitcoin). Neprihvatljiva upotreba uključuje korištenje EVM-a kao sustava za pohranu datoteka, e-pošte ili tekstualnih poruka, bilo kakve veze s grafičkim sučeljima i

aplikacijama koje su najprikladnije za programiranje u oblaku poput genetskih algoritama, analize grafikona ili strojnog učenja.

Kako bi se spriječili namjerni napadi i zlouporabe, ethereum protokol naplaćuje naknadu po računarskom koraku. Naknada je tržišna, iako je u praksi obvezna. Veliko ograničenje broja operacija koje mogu biti sadržane u bloku prisiljava čak i rudare koji mogu priuštiti uključivanje transakcija približno i bez troškova da bi naplatili naknadu razmjernu trošku transakcije na cijeloj mreži.

Način na koji gorivo (eng. Gas) funkcionira je da svaka transakcija mora zajedno sa ostalim podacima sadržavati i vrijednosti GASPRICE i STARTGAS. STARTGAS je iznos goriva koji transakcija dodjeljuje sama, a GASPRICE je naknada koju transakcija plaća po jedinici goriva. Kad je poslana transakcija, prvo što se napravi tijekom procjene je oduzimanje STARTGAS * GASPRICE plus vrijednosti transakcije iz računa slanja. GASPRICE postavlja pošiljatelj transakcije, ali rudari će vjerojatno odbiti obraditi transakcije čiji je GASPRICE prenizak.

Gorivo se može otprilike smatrati brojem računskih koraka i nešto je što postoji tijekom izvršavanja transakcije, ali ne i izvan nje. Kad započne izvršavanje transakcije, preostalo gorivo se postavlja na STARTGAS - 21000 - 68 * TXDATALEN, gdje je TXDATALEN broj bajtova u transakcijskim podacima, primjerice nula bajtova naplaćuje se samo 4 goriva zbog veće stisljivosti dugih nizova nula bajtova. U svakom računatom koraku oduzima se određena količina goriva. Ako količina goriva padne na nulu, tada se izvršenje poništava, ali transakcija je i dalje valjana i pošiljatelj još uvijek mora platiti za gorivo. Ako izvršenje transakcije završi s preostalim N > = 0, račun za slanje vraća se u obliku N * GASPRICE.

Za vrijeme izvršavanja ugovora, kada ugovor šalje poruku, sam taj poziv dolazi s ograničenjem goriva, a podizvršenje djeluje na isti način, može nestati goriva i vratiti poruku ili se uspješno izvršiti i vratiti vrijednost. Ako u podizvršenju ponestane goriva, nadređeno izvršenje se nastavlja. Potpuno je sigurno da ugovor poziva drugi ugovor ako se u podizvršenju postavi ograničenje maksimalnog goriva. Ako u podizvršenju ostaje nešto goriva, tada se to gorivo vraća u nadređeno izvršenje za nastavak upotrebe.

3.7. Ethereum rudarenje

Riječ rudarstvo potječe iz konteksta zlatne analogije za kriptovalute. Zlato ili plemeniti metali su rijetki, pa tako i digitalni tokeni, a jedini način za povećanje ukupnog volumena mreže jest rudarenjem. To je primjereno u mjeri u kojoj i je, odnosno da je i u ethereum mreži jedini

način pokretanja putem rudarstva. Međutim, za razliku od ovih primjera, rudarstvo je također način osiguranja mreže stvaranjem, provjerom, objavljivanjem i propagiranjem blokova u blockchainu. Slika 3.7. ilustrativno pokazuje što se dobiva i što radi ethereum rudarenje.

Slika 3.7. Ilustrativni prikaz ethereum rudarenja.

Ethereum mreža, kao i sve tehnologije blockchaina, koristi poticajni model sigurnosti. Protokol se temelji na odabiru bloka s najvećom ukupnom težinom. Rudari proizvode blokove za koje drugi provjere valjanost. Između ostalih dobro utvrđenih kriterija, blok je valjan samo ako sadrži dokaz o radu (PoW) određene težine. Upotrijebljeni algoritam dokaza rada zove se Ethash koji uključuje pronalaženje nonce vrijednosti ulaza u algoritam tako da je rezultat ispod određenog praga, ovisno o težini. Poanta u PoW algoritmima je da ne postoji bolja strategija za pronalaženje nonce vrijednosti od nabrajanja mogućnosti, dok je provjera rješenja neznatna i jeftina. Ako izlazi imaju jednoliku raspodjelu, tada možemo jamčiti da u prosjeku vrijeme potrebno za pronalazak nonce vrijednosti ovisi o pragu težine, omogućavajući kontrolu vremena pronalaska novog bloka samo manipuliranjem težine.

Težina se dinamički prilagođava tako da u prosjeku jedan blok proizvede cijela mreža svakih 12 sekundi. S time se naglašava sinkronizacija stanja sustava i osigurava da je nemoguće održavanje račvanja (kako bi se omogućilo dvostruko trošenje) ili prepisivanje povijesti, osim ako haker ne posjeduje više od polovice rudarske snage mreže (tzv. Napad od 51%).

Svaki čvor koji sudjeluje u mreži može biti rudar i njihov očekivani prihod od rudarstva bit će izravno proporcionalan njihovoj (relativnoj) rudarskoj snazi ili hashrate-u, tj. broju pokušaja u sekundi normaliziranom ukupnim hashrate-om mreže.

Ethash PoW je težak za memoriju, što ga čini uglavnom ASIC otpornim. To znači da izračunavanje PoW-a zahtijeva odabir podskupova fiksnog resursa ovisno o nonce vrijednosti i zaglavlju bloka. Taj resurs (nekoliko podataka o veličini gigabajta) naziva se DAG (eng. directed acyclic graph). DAG je potpuno različit svakih 30000 blokova (prozor od 100 sati, koji se naziva epoha) i treba vremena da se generira. Budući da DAG ovisi samo o visini bloka, može se ponovno izgenerirati, ali ako ne, klijent mora pričekati kraj ovog procesa da bi

se stvorio blok. Sve dok klijenti ne predbilježe DAG-ove prije vremena, mreža može doživjeti veliko blokadno kašnjenje na svakom prijelazu epohe. DAG se ne treba generirati za provjeru PoW-a koji u osnovi omogućuje provjeru i s nižim CPU-om i s malo memorije. Kao poseban slučaj, kada se pokrene čvor ispočetka, rudarstvo će započeti tek kada je DAG izgrađen za trenutnu epohu.

Uspješni PoW rudar rudarenog bloka uobičajeno dobiva nagradu za statički blok za "pobjednički" blok, koja se sastoji od točno 3.0 Ethera. Svo potrošeno gorivo unutar bloka, odnosno svo gorivo potrošeno izvršavanjem svih transakcija u bloku, koji je predao pobjednički rudar, nadoknađuje pošiljatelj. Navedeni trošak energije pripisuje se računu rudara u sklopu protokola mreže. Dodatna nagrada za uključivanje rođaka (eng. Uncles) kao dijela bloka, nagrada je u obliku dodatnih 1/32 po uključenom partneru. Rođaci su ustaljeni blokovi, tj. s roditeljima koji su preci (maksimalno 6 blokova natrag) uključenog bloka. Vrijedni partneri nagrađuju se kako bi se neutralizirao učinak zaostajanja mreže na raspodjelu nagrada za rudarstvo i na taj način povećala sigurnost. Rođaci uključeni u blok formiran od strane uspješnog PoW rudara dobivaju 7/8 nagrade za statički blok, što je 2.625 ehtera. Maksimalno dva rođaka dopuštena su po bloku.

3.7.1. Ethash

Ethash se temelji na pružanju velikog, prolaznog, nasumično generiranog skupa podataka koji tvori DAG i pokušaja da se riješi određeno ograničenje na njega, dijelom određeno kroz blok-zaglavljivanje bloka.

Iskopavanje ethash-a uključuje dohvaćanje slučajnih podataka iz skupa podataka (DAG), izračunavanje nekih nasumično odabranih transakcija iz bilo kojeg bloka te zatim vraćanje hash-a rezultata. To znači da će rudar trebati pohraniti cijeli DAG kako bi mogao dohvatiti slučajne podatke i izračunati nasumično odabrane transakcije. Kao rezultat toga, rudarenje temeljeno na dokazu rada na ethereumu može se okarakterizirati kao memorijski tvrd ili memorijski ovisno. Memorijski tvrd odnosi se na situaciju u kojoj je vrijeme potrebno za dovršavanje određenog računalnog problema prvenstveno odlučeno količinom memorije koja je potrebna za pohranu podataka. To znači da će velika većina napora rudara biti potrošena na čitanje DAG-a, umjesto na računanje podataka preuzetih iz njega. Zbog toga je ethereum rudarenje otporno na ASIC čipove jer veliki zahtjev za memorijom znači da rudari velikih razmjera dobivaju malo koristi od pakiranja terabajta memorije u svoje uređaje. To je zato što manji rudari mogu postići isti efekt jednostavnom kupnjom velike količine uređaja za

zadržavanje memorije jer su energetski zadržavanja memorije puno manja u odnosu na ASIC čipove i za bilo koji fizički uređaj koji može pohraniti memoriju.

3.7.2. Grafičke kartice

Rudarenje bitcoina više nije isplativo na potrošačkim grafičkim karticama jer su ljudi izgradili sofisticiranu prilagođenu rudarsku opremu koja se temelji na ASIC-u i koja je daleko učinkovitija. Ethereum ima algoritam rudarenja koji se temelji na memoriji i otporan je na ASIC optimizaciju. To znači da je rudarstvo Ethereuma i dalje praktično s potrošačkom grafičkom karticom ako ima više od dva gigabajta memorije. Obične grafičke kartice nemaju dovoljno memorije, ali naprednije imaju.

Rudarstvo na grafičkoj kartici je proces rudarstva kriptovalutama pomoću grafičkih procesora (GPU-ova). Da bi to bilo moguće, koristi se snažna grafička kartica u kućnom računalu ili u posebno sastavljenoj farmi s nekoliko uređaja u jednom sustavu. GPU se koristi za ovaj proces zbog toga što su dizajnirane za obradu velikih količina podataka izvođenjem operacija istog tipa, kao što je slučaj s obradom videa. Slično se događa i kod rudarenja kriptovalutama.

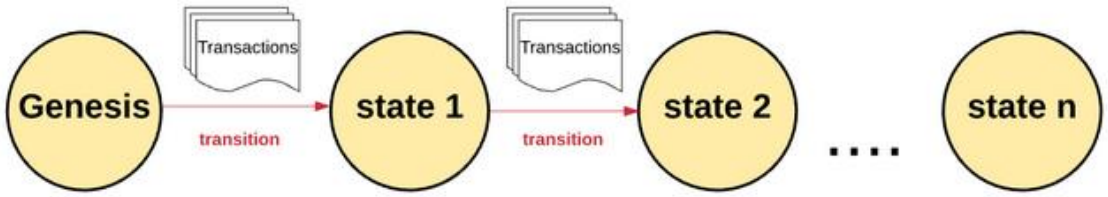
Svaka od arhitektura ima svoje prednosti. CPU izvršava sekvencijalne zadatke puno bolje. Za velike količine obrađenih informacija GPU ima prednost. Glavni uvjet je da zadatak mora biti paralelan.

4. ETHEREUM ALGORITAM

U ovom poglavlju dan je opis rada ethereum algoritma i nekih od klasa EVM-a iz cpp-ethereum implementacije napisanog u C++ programskom jeziku. Također, prikazan je primjer programiranja pametnog ugovora u solidity programskom jeziku.

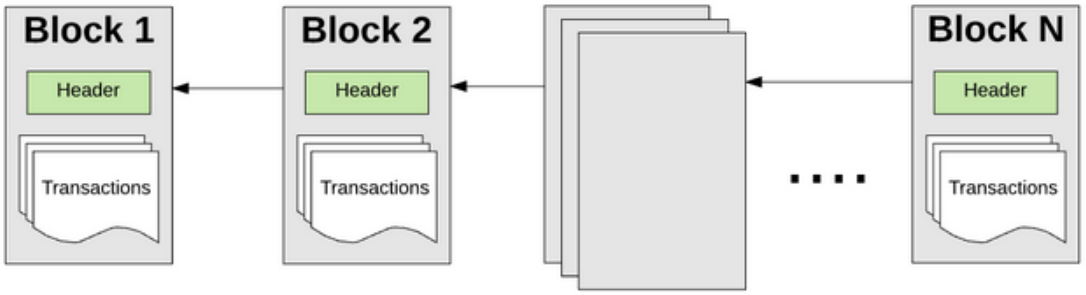
4.1. Opis rada ethereum algoritma

U ethereumovoj mreži započinjemo s "genesis stateom". To je analogno praznoj pločici, prije nego što se na mreži dogodi bilo koji posao. Kad se transakcije izvrše, genesis state prelazi u neko konačno stanje. U bilo kojem trenutku ovo konačno stanje predstavlja trenutni ethereum state.



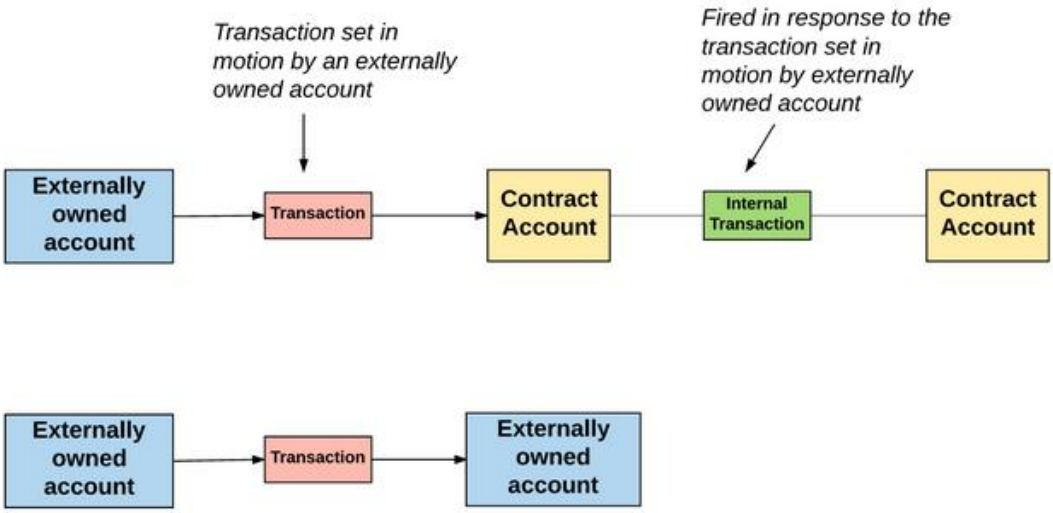
Slika 4.1. Ethereum lanac, počinje sa genesis stanjem.

Ethereum state ima milijune transakcija. Te se transakcije grupiraju u „blokove.“ Blok sadrži niz transakcija, a svaki se blok veže zajedno sa svojim prethodnim blokom.



Slika 4.2. Povezivanje blokova.

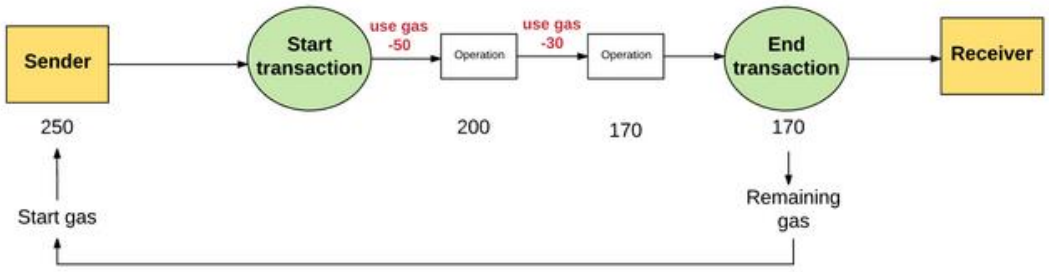
Da bi se prouzročio prijelaz iz jednog stanja u drugo, transakcija mora biti valjana. Da bi se transakcija smatrala valjanom, mora proći postupak provjere validacije poznat kao rudarstvo.



Slika 4.3. Prikaz rada poruka i transakcija.

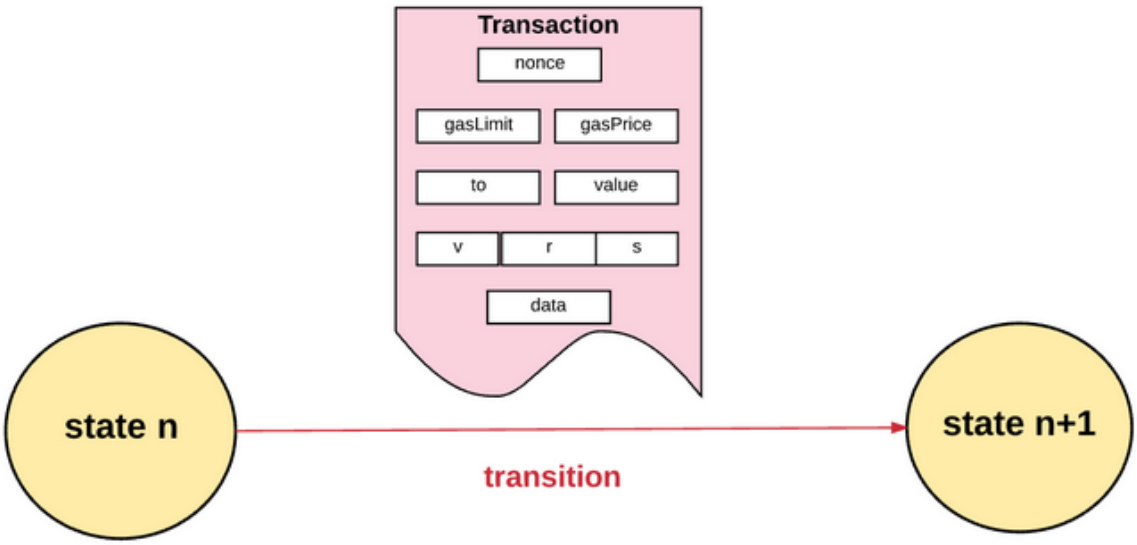
Račun u vanjskom vlasništvu može slati poruke na druge račune u vanjskom vlasništvu ili na druge račune ugovora stvaranjem i potpisivanjem transakcije pomoću privatnog ključa. Poruka između dva računa u vanjskom vlasništvu je zapravo prijenos vrijednosti tokena, ali poruka s računa u vanjskom vlasništvu na račun ugovora aktivira kod ugovora, omogućavajući mu obavljanje različitih radnji. Kao što možemo vidjeti na slici 4.3. za razliku od računa u vanjskom vlasništvu, pametni ugovori ne mogu samostalno pokrenuti nove transakcije iz početka, ali mogu započeti transakcije s drugim ugovorima ili krajnjim računom kao odgovor na druge primljene transakcije.

Svako rudarenje koje mora biti odrađeno kao rezultat transakcije u ethereum mreži plaća naknadu za gorivo. Uz svaku transakciju postavlja se ograničenje goriva i cijena goriva. Ograničenje i cijena goriva predstavlja maksimalni iznos koji je pošiljatelj spreman platiti za izvršenje transakcije.



Slika 4.4. Prikaz praćenja goriva prilikom jedne transakcije.

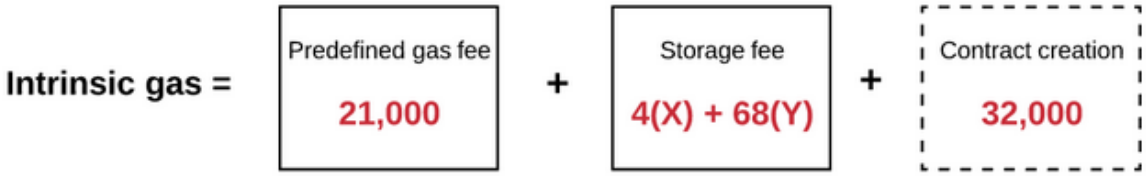
Sav novac koji je pošiljatelj potrošio na gorivo šalje se na adresu rudara. Budući da rudari troše napore na pokretanje računanja i potvrđivanje transakcija, rudari dobivaju naknadu za gorivo kao nagradu. Što više plati, rudari će imati veću vrijednost nad tom transakcijom. Što znači da će je i prije odabrati.



Slika 4.5. Prikaz pokretanja jedne transakcije.

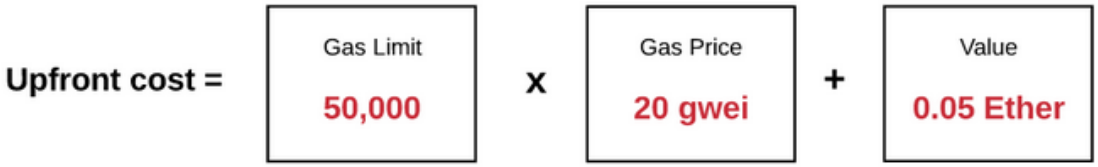
Ono što se dogodi prilikom izvršavanja jedne transakcije prikazano je na slici 4.5., sve transakcije moraju ispunjavati početni skup zahtjeva da bi bile izvršene. To uključuje:

- Transakcija mora biti pravilno oblikovana RLP. „RLP" označava „Recursive Length Prefix" i format podataka koji se koristi za kodiranje ugniježđenih nizova binarnih podataka. RLP je format koji ethereum koristi za serijalizaciju objekata.
- Valjani potpis transakcije.
- Valjani potpis nonce vrijednosti. Nonce vrijednost računa je broj transakcija poslanih s tog računa. Da bi bila valjana, transakcijski nonce mora biti jednaka vrijednosti nonce na pošiljateljevom računu.
- Limit goriva transakcije mora biti jednak ili veći od svojstvenog goriva koji se koristi u transakciji. Svojstveno gorivo uključuje: unaprijed definirani trošak goriva za izvršenje transakcije, naknadu za gorivo za podatke poslane transakcijom, ako je transakcija ugovor koji stvara ugovor plaća se dodatno gorivo.



Slika 4.6. Izračun svojstvenog goriva.

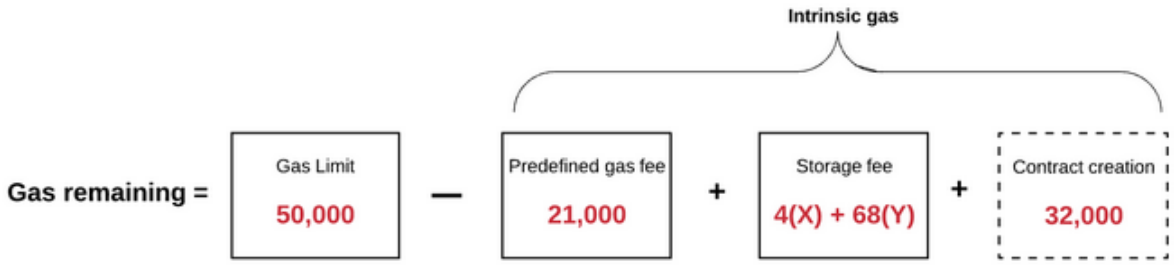
Stanje na računu pošiljatelja mora imati dovoljno ethera da unaprijed pokrije troškove goriva koje pošiljatelj mora platiti. Izračun za gornji trošak goriva sa slike 4.6. je jednostavan: ograničenje goriva transakcije množi se s cijenom goriva transakcije kako bi se odredio maksimalni trošak goriva. Potom se maksimalni trošak dodaje ukupnoj vrijednosti koja se s pošiljatelja prenosi na primatelja.



Slika 4.7. Izračun za maksimalne moguće troškove goriva.

Ako transakcija ispunjava sve navedene uvjete za valjanost koji su dani na slici 4.7., odnosno ako korisnik ima dovoljno ethera da pokrije maksimalne moguće troškove, prelazi se na sljedeći korak.

Prvi korak je da oduzimamo uplaćeni trošak izvršenja iz salda pošiljatelja i povećamo nonce vrijednost pošiljateljevog računa za 1 da bi se prikazala trenutna transakcija. U ovom trenutku možemo izračunati preostalo gorivo kao ukupno ograničenje goriva za transakciju umanjeno za uporabljeno unutarnje gorivo, formulu možemo vidjeti na sljedećoj slici 4.8.



Slika 4.8. Izračun preostalog goriva.

Zatim započinje izvršenje transakcije. Tijekom izvršavanja transakcije, ethereum mreža prati substate. Substate je način zapisivanja podataka prikupljenih tijekom transakcije koji će biti potrebni odmah nakon što se transakcija završi. Ono konkretno sadrži:

- Skup samouništenja: skup računa (ako postoje) koji će se odbaciti nakon završetka transakcije.
- Log serija: arhivirane i indeksirane kontrolne točke izvršenja koda virtualnog stroja.
- Saldo povrata novca: iznos koji treba biti vraćen na račun pošiljatelja nakon transakcije.

Nakon što su obrađeni svi koraci potrebni za transakciju i pod pretpostavkom da ne postoji nevažeće stanje, stanje se finalizira određivanjem količine neiskorištenog goriva koje će se vratiti pošiljatelju. Pored neiskorištenog goriva, pošiljatelju se vraća i dodatak iz „salda povrata“. Nakon što je pošiljatelju vraćen novac događa se sljedeće:

- ether za gorivo daje se rudaru
- gorivo koje se koristi transakcijom dodaje se brojaču bloka goriva (koji prati ukupno gorivo potrošeno u svim transakcijama u bloku i koristan je pri provjeri bloka)
- brišu se svi računi u skupu samouništenja (ako ih ima)

Preostalo je novo stanje i skup logova stvorenih transakcijom.

Da bismo stvorili novi račun pametnog ugovora, prvo deklariramo adresu novog računa pomoću posebne formule. Potom inicijaliziramo novi račun:

- Postavljanje nonce vrijednosti na nulu
- Ako je pošiljatelj poslao neki iznos ethera kao vrijednosti s transakcijom, postavljanje stanja računa na tu vrijednost
- Oduzimanje dodane vrijednosti na računu novog računa od stanja računa pošiljatelja
- Postavljanje spremnika na prazno
- Postavljanje hash koda ugovora kao hash praznog niza

Izvršenje transakcijske poruke slično je stvaranju ugovora, ali s nekim razlikama. Izvršenje transakcijske poruke ne uključuje nijedan početni kod jer se ne stvaraju novi računi. Međutim, može sadržavati ulazne podatke ako ih je dostavio pošiljatelj transakcije. Jednom

izvršene, transakcijske poruke također sadrže dodatnu komponentu koja sadrži izlazne podatke, a koja se koristi ako je potrebno daljnje izvršavanje.

Kao što je slučaj sa stvaranjem ugovora, ako izvršenje transakcijske poruke prestane jer ponestane goriva ili zato što je transakcija nevaljana (npr. Stack overflow, nevažeće odredište ili nevaljana uputa), nijedno korišteno gorivo ne vraća se izvornom pozivaču. Umjesto toga, troši se svo preostalo neiskorišteno gorivo, a stanje se vraća na točku neposredno prije prijenosa stanja računa.

U nastavku biti će kratko opisani neki od dijelova koda iz cpp-ethereum implementacije ethereuma.

4.2. Block.h & Block.cpp

Klasa Block predstavlja blok unutar transakcije i sadrži sljedeće atribute:

```
State m_state;
Transactions m_transactions;
TransactionReceipts m_receipts;
h256Hash m_transactionSet;
State m_precommit;
BlockHeader m_previousBlock;
BlockHeader m_currentBlock;
bytes m_currentBytes;
bool m_committedToSeal = false;
bytes m_currentTxs;
bytes m_currentUncles;
Address m_author;
SealEngineFace* m_sealEngine = nullptr;
```

Slika 4.9. Atributi klase Block.

Opis slike 4.9. koja prikazuje atribute klase Block:

- m_state – stanje mreže
- m_transactions – trenutna lista transakcija koje su uključene u trenutnom stanju mreže
- m_receipts – lista potvrda transakcije
- m_transactionSet – skup hasheva korištenih unutar stanja mreže
- m_precommit – stanje mreže u trenutku odmah prije dobivanja nagrade
- m_previousBlock – informacije prethodnog bloka
- m_currentBlock – informacije trenutnog bloka
- m_currentBytes – veličina bloka u byte-ovima
- m_currentTxs – enkodirani blok transakcija

- m_currentUncles – enkodirani blok rođaka
- m_author – adresa računa (adresa gdje će ići novac)
- m_sealEngine – seal engine

```
Block(u256 const& _accountStartNonce): m_state(_accountStartNonce,
OverlayDB(), BaseState::Empty), m_precommit(_accountStartNonce) {}
```

Slika 4.10. Zadani konstruktor klase Block.

Funkcija koja je zadani konstruktor unutar klase Block prikazana je na slici 4.10. Pomoću njega stvara se blok s praznom bazom podataka koja je popunjena od genesis bloka. Kao prvi parametar konstruktoru predaje se varijabla _accountStartNonce. Kao drugi parametar predaje se varijabla klase m_state i kao treći parametar predaje se varijabla klase m_precommit.

```
Block(BlockChain const& _bc, OverlayDB const& _db, h256 const& _root,
Address const& _author = Address());
```

Slika 4.11. Osnovni konstruktor klase Block.

Konstruktor funkcija koja stvara osnovni blok iz baze podataka prikazana je na slici 4.11. Kao parametre prima varijable BlockChain, OverlayDB, h256 i Address. Vratit će InvalidRoot grešku ako se predani korijen ne nalazi u bazi podataka. Postoje još nekoliko različitih konstruktora, a ovi prikazani su osnovni.

4.3. Account.h & Account.cpp

Atributi Account klase prikazani su na slici 4.12.

```
bool m_isAlive = false;
bool m_isUnchanged = false;
bool m_hasNewCode = false;
u256 m_nonce;
u256 m_balance = 0;
h256 m_storageRoot = EmptyTrie;
h256 m_codeHash = EmptySHA3;
u256 m_version = 0;
mutable std::unordered_map<u256, u256> m_storageOverlay;
mutable std::unordered_map<u256, u256> m_storageOriginal;
bytes m_codeCache;
static const h256 c_contractConceptionCodeHash;
```

Slika 4.12. Atributi klase Account.

Opis atributa klase account:

- m_isAlive – zastavica ako račun postoji, ako je lažan, predstavlja obrisani račun
- m_isUnchanged – zastavica će biti istinita ako na računu nisu rađene nikakve promjene
- m_hasNewCode – zastavica će biti istinita ako je novi kod ispisan u račun
- m_nonce – nonce vrijednost računa
- m-balance – trenutno stanje računa
- m_storageRoot – osnovni storage root, ako se koristi m_storageOverlay on ima prednost
- m_codeHash - hash kod
- m_version – verzija računa
- m_storageOverlay – ovisi o stanju storageRoota u trenutnom stanju bloka u mreži
- m_storageOriginal – cache vrijednost od ne mijenjanih spremljenih podataka
- m_codeCache – kod korišten na računu
- c_contractConceptionCodeHash – vrijednost za m_codeHash kada je ovaj račun izvršen

```
Account(u256 _nonce, u256 _balance, Changedness _c = Changed): m_isAlive(true),
m_isUnchanged(_c == Unchanged), m_nonce(_nonce), m_balance(_balance) {}
```

Slika 4.13. Konstruktor klase Account za stvaranje računa.

Konstruktor funkcija koja stvara novi aktivni račun, koristi se i za stvaranje EOA-računa i za pametne ugovore kod kojih kod još nije poznat.

```
Account(u256 const& _nonce, u256 const& _balance, h256 const& _contractRoot,
    h256 const& _codeHash, u256 const& _version, Changedness _c)
  : m_isAlive(true),
    m_isUnchanged(_c == Unchanged),
    m_nonce(_nonce),
    m_balance(_balance),
    m_storageRoot(_contractRoot),
    m_codeHash(_codeHash),
    m_version(_version)
{assert(_contractRoot);}
```

Slika 4.14. Konstruktor klase Account za stvaranje nestandardnih pametnih ugovora.

Eksplicitni konstruktor koji se koristi za stvaranje pametnih ugovora koji nisu uobičajeni.

4.4. Transaction.h & Transaction.cpp

Klasa transaction nasljeđuje klasu TransactionBase.

```
class Transaction: public TransactionBase
{
public:
    Transaction() {}

    Transaction(TransactionSkeleton const& _ts,
    Secret const& _s = Secret()): TransactionBase(_ts, _s) {}

    Transaction(u256 const& _value, u256 const& _gasPrice, u256 const& _gas,
    Address const& _dest, bytes const& _data, u256 const& _nonce, Secret const& _secret):
        TransactionBase(_value, _gasPrice, _gas, _dest, _data, _nonce, _secret) {}

    Transaction(u256 const& _value, u256 const& _gasPrice, u256 const& _gas,
    bytes const& _data, u256 const& _nonce, Secret const& _secret):
        TransactionBase(_value, _gasPrice, _gas, _data, _nonce, _secret) {}

    Transaction(u256 const& _value, u256 const& _gasPrice, u256 const& _gas,
    Address const& _dest, bytes const& _data, u256 const& _nonce = Invalid256):
        TransactionBase(_value, _gasPrice, _gas, _dest, _data, _nonce) {}

    Transaction(u256 const& _value, u256 const& _gasPrice, u256 const& _gas,
    bytes const& _data, u256 const& _nonce = Invalid256):
        TransactionBase(_value, _gasPrice, _gas, _data, _nonce) {}

    explicit Transaction(bytesConstRef _rlp, CheckTransaction _checkSig);

    explicit Transaction(bytes const& _rlp, CheckTransaction _checkSig):
        Transaction(&_rlp, _checkSig) {}
};
```

Slika 4.15. Konstruktori klase Transaction.

Na slici 4.15. su prikazani konstruktori klase Transaction.

- Prvi konstruktor stvara praznu transakciju.
- Drugi konstruktor stvara transakciju iz kalupa transakcije te daje na opciju korištenje tajne.
- Treći konstruktor stvara potpisanu transakciju za stvaranje poruke.
- Četvrti konstruktor stvara potpisanu transakciju za stvaranje pametnog ugovora.
- Peti konstruktor stvara nepotpisanu transakciju za stvaranje poruke.
- Šesti konstruktor stvara nepotpisanu transakciju za stvaranje pametnog ugovora.
- Sedmi i osmi konstruktor stvaraju transakciju na temelju danog rlp enkodiranog podatka.

Četvrti i peti konstruktor koriste nonce vrijednost te se koriste češće od petog i šestog konstruktora gdje je nonce vrijednost stavljena kao Invalid256 vrijednost.

Struktura u koju budu zapisani rezultati izvršavanja transakcije prikazana je na sljedećoj slici 4.16.

```
struct ExecutionResult
{
    u256 gasUsed = 0;
    TransactionException excepted = TransactionException::Unknown;
    Address newAddress;
    bytes output;
    CodeDeposit codeDeposit = CodeDeposit::None;
    u256 gasRefunded = 0;
    unsigned depositSize = 0;
    u256 gasForDeposit;
};
```

Slika 4.16. Struktura ExecutionResult.

Iz strukture se može iščitati koliko je goriva potrošeno, vraćeno i koliko je još goriva ostalo za depozit koda. Vidi se depozit veličina i kod od depozita te još neke dodatne informacije o izvršenoj transakciji.

4.5. Solidity

Solidity je high-level jezik čija je sintaksa slična onome u JavaScript-u i dizajniran je radi lakšeg programiranja pametnih ugovora koji se kompajliraju za EVM.

Solidity se može početi koristiti u web pregledniku, bez potrebe za preuzimanjem i instaliranjem. Ova aplikacija podržava samo kompilaciju, ako se želi pokrenuti kod ili ga ubaciti u blockchain, mora se koristiti klijent poput cpp-ethereuma.

Trenutno implementirani (osnovni) tipovi su vrste bool, int (cijeli brojevi) i stringovi s fiksnom duljinom. Cijeli brojevi mogu biti signed i unsigned što omogućuje različite širine bita (int8 / uint8 do int256 / uint256 u koracima od 8 bita, pri čemu je uint / int isto što i uint256 / int256) i adrese (do 160 bita).

Usporedbe (<=,! =, ==, itd.) uvijek daju logične znakove koji se mogu kombinirati pomoću &&, || i !.

Broj može imati sufiks wei, finney, szabo ili ethera za pretvaranje između poddominacija ethera, pri čemu se brojevi etherskih valuta bez postfix pretpostavljaju kao "wei", npr. 2 etera == 2000 finney ocjenjuje se istinitim.

Nadalje, sufiksi sekundi, minuta, sati, dana, tjedana i godina mogu se koristiti za pretvaranje između jedinica vremena u kojima su sekunde osnovna jedinica, a jedinice se naivno pretvaraju, tj. godina je uvijek točno 365 dana.

Većina upravljačkih struktura iz C / JavaScript-a dostupne su u Solidityju, osim za switch i goto. Dakle, postoji: if, else, while, for, break, continue, return. Ne postoji pretvorba tipa iz neboolean u boolean tipove kao što je to u C i JavaScript, primjer: if(1) {...} nije valjan u Solidity programskom jeziku. Pozivanje funkcija isto je kao u JavaScript jeziku.

Svojstva bloka i transakcije

- `block.coinbase` (`address`) – trenutna adresa rudara bloka
- `block.difficulty` (`uint`) – težina bloka
- `block.gaslimit` (`uint`) – limit za gorivo
- `block.number` (`uint`) – broj bloka
- `block.blockhash` (`function(uint) returns (bytes32)`) – hash bloka
- `block.timestamp` (`uint`) – vremenska oznaka bloka
- `msg.data` (`bytes`) – cijeli calldata
- `msg.gas` (`uint`) – preostalo gorivo
- `msg.sender` (`address`) – pošiljatelj poruke
- `msg.value` (`uint`) – broj wei-a poslano sa porukom
- `now` (`uint`) – trenutna vremenska oznaka bloka, alias za block.timestamp
- `tx.gasprice` (`uint`) – cijena goriva za transakciju
- `tx.origin` (`address`) – pošiljatelj transakcije

Kriptografske funkcije

- `sha3(...) returns (bytes32)` - izračunava SHA3 hash argumenata
- `sha256(...) returns (bytes32)` - izračunava SHA256 hash argumente
- `ripemd160(...) returns (bytes20)` - izračunava RIPEMD od 256 argumenata
- `ecrecover(bytes32, byte, bytes32, bytes32) returns (address)` - oporavi javni ključ iz potpisa eliptičke krivulje

Primjer pametnog ugovora koji radi na način da podijeli zaposlenicima njihove plaće. Plaće se dijele tako da svaki zaposlenik dobije jednaku svotu novca. Organizator posla stvori pametni

ugovor i ubaci u njega nešto ethera, a zaposlenici moraju pozvati ugovor kako bi im se plaća isplatila.

```
pragma solidity >=0.4.22 <0.6.0;
contract SplitIt {
    address[] employees = [0x8142C51DB7e1807e64b11e54983F5b46FBB425dB,
                           0xe0bba18dD678dCC9A130bD950d29891D7dEc0514];
    uint totalReceivedMoney = 0;
    mapping (address => uint) withdrawnAmounts;

    function SplitIt() payable {
        updateTotalReceivedMoney();
    }

    function () payable {
        updateTotalReceivedMoney();
    }

    function updateTotalReceivedMoney() internal {
        totalReceivedMoney = totalReceivedMoney + msg.value;
    }

    modifier canWithdraw() {
        bool contains = false;
        for(uint i = 0; i < employees.length; i++) {
            if(employees[i] == msg.sender) {
                contains = true;
                // break
            }
        }
        require(contains);
        _;
    }

    function withdraw() canWithdraw {
        uint amountAllocated = totalReceivedMoney / employees.length;
        uint amountWithdrawn = withdrawnAmounts[msg.sender];
        uint amount = amountAllocated - amountWithdrawn;
        withdrawnAmounts[msg.sender] = amountWithdrawn + amount;
        if(amount > 0) {
            msg.sender.transfer(amount);
        }
    }
}
```

Slika 4.17. Implementacija pametnog ugovora za dijeljenje novca zaposlenicima.

Na slici 4.17. možemo vidjeti kako je to implementirano. Na početku smo stvorili tri globalne varijable. Prva je lista tipa address gdje se spremaju adrese računa zaposlenika. Samo adrese koje su na toj listi mogu pristupiti ugovoru. Druga je tipa uint i predstavlja ukupni primljeni novac u ugovoru. Treća je mapa pomoću koje pratimo koliko je svaki zaposlenik povukao novca iz ugovora, kako se ne bi dogodilo da dva puta dobije isplatu.

Sljedeća funkcija je konstruktor funkcija koja se izvodi kada se ugovor stvori.

```
function SplitIt() payable {
	updateTotalReceivedMoney();
}
```

Ključna riječ payable omogućava da se ether pridoda u ugovor jer po zadanim postavkama pametni ugovori ne prihvaćaju ether jer ugovori nemaju privatni ključ i postoji mogućnost da ether zauvijek ostane unutar ugovora.

Funkcija koja se poziva kada netko želi ubaciti novac u ugovor, koristi se pomoćna funkcija pomoću koje uvijek ažuriramo stanje ugovora.

```
function () payable {
	updateTotalReceivedMoney();
}
```

Pomoćna funkcija koja zbraja sav ukupni primljeni novac. Pomoćna funkcija ima ključnu riječ internal te je pomoću nje vidljiva samo unutar ugovora.

```
function updateTotalReceivedMoney() internal {
	totalReceivedMoney = totalReceivedMoney + msg.value;
}
```

Sljedeće što trebamo napraviti je da omogućimo pristup samo računima zaposlenika, kako se ne bi dogodilo da bilo tko na mreži napravi transakciju na ugovor i dobije novac. To je omogućeno sljedećim programskim kodom:

```
modifier canWithdraw() {
	bool contains = false;
	for(uint i = 0; i < employees.length; i++) {
		if(employees[i] == msg.sender) {
		contains = true;
		// break
		}
	}
```

```
        require(contains);

        _;
}
```

Provjerava se je li adresa računa koji zahtjeva transakciju u listi adresa. Ako je vraća istinu, ako nije vraća laž. Break naredba je zakomentirana kako bi svi zaposlenici koji traže transakciju jednako platili gorivo za transakciju. Ako break nije zakomentiran, prvi zaposlenik u listi platit će gorivo manje jer će se for petlja manje vrtiti i samim time će se brže izvršiti kod ugovora.

Zadnja funkcija kojom radimo isplatu zaposlenicima omogućena je sljedećim programskim kodom:

```
function withdraw() canWithdraw {

        uint amountAllocated = totalReceivedMoney / employees.length;

        uint amountWithdrawn = withdrawnAmounts[msg.sender];

        uint amount = amountAllocated - amountWithdrawn;

        withdrawnAmounts[msg.sender] = amountWithdrawn + amount;

        if(amount > 0) {

                msg.sender.transfer(amount);

        }
}
```

Funkcija withdraw nasljeđuje funkciju canWithdraw te će se izvršiti samo ako je rezultat funkcije canWithdraw istinit. Postupak koji radi funkcija za isplaćivanje je da podijeli čitav novac u ugovoru sa brojem zaposlenika. Zatim provjerava koliko je već novaca zaposlenik koji traži transakciju dobio. Izračunava se količina novca koju bi trebao dobiti zaposlenik na način da se oduzima novac predviđen za tog zaposlenika od onoga što je već dobio. Na kraju se još dodaje ukupnoj svoti novca kako bi se pratilo koliko je koji zaposlenik dobio. Ako je količina novca veća od 0, odnosno ako ima nešto da se pošalje na račun šalje se pomoću naredbe transfer, a u suprotnom se ne dogodi ništa.

5. ZAKLJUČAK

U ovom diplomskom radu obrađeno je što su kriptovalute i Ethereum te princip rada Ethereum algoritma pomoću pametnih ugovora. Kriptovalute su novi oblik digitalnog novca koji funkcionira i nastaje na temelju složenih kriptografskih algoritama na računalnim mrežama. Koriste se jednako kao i plaćanje karticom ili internet bankarstvom. To su promjene zapisa na računima, bez fizičke razmjene novca. Ethereum je open-source program, zasnovan na blockchain tehnologiji, koja dozvoljava drugim programerima da grade i razvijaju decentralizirane aplikacije. Osnovni principi Ethereum algoritma su rad na blockchainu i omogućavanje računima transakcije i poruke unutar Ethereum mreže. Svaka transakcija i poruka mora proći validaciju zbog koje postoji rudarenje. Za svaku transakciju plaća se gorivo koje je nagrada rudarima za validiranje transakcije. Aplikacije ili pametni ugovori su računalni kod koji može olakšati razmjenu novca ili bilo čega vrijednog. Pametni ugovori su pouzdani, što znači da će se, nakon što budu "učitani" u Ethereum, uvijek izvoditi prema programu. Oni mogu kontrolirati digitalnu imovinu radi stvaranja novih vrsta financijskih aplikacija. Mogu se decentralizirati, što znači da ih ne kontrolira niti jedan entitet ili osoba te su samim time zaštićeni od krađa i nelegalnih radnji. Ethereum se tek počeo razvijati i on tek dolazi, a pošto je programabilan otvara vrata raznim aplikacijama koje se mogu decentralizirati i koristiti pogodnosti koje ethereum mreža pruža.

LITERATURA

[1] Kriptovalute, srpanj 2019.
https://www.kriptovaluta.hr/bitcoin/princip-rada-kriptovaluta/

[2] Blockchain, srpanj 2019.
https://repozitorij.pmf.unizg.hr/islandora/object/pmf%3A779/datastream/PDF/view

[3] Wallet, kolovoz 2019.
https://blockgeeks.com/guides/cryptocurrency-wallet-guide/

[4] Digitalni potpis, kolovoz 2019.
https://www.coindesk.com/information/what-is-blockchain-technology

[5] Ethereum, kolovoz 2019.
https://github.com/ethereum/wiki/wiki/Mining

[6] Hashrate, kolovoz 2019.
https://coinsutra.com/hash-rate-or-hash-power/

[8] Ethereum, kolovoz 2019.
https://www.coindesk.com/information/what-is-ethereum

[8] Ethereum, kolovoz 2019.
https://en.wikipedia.org/wiki/Ethereum

[9] Što je ether, ethereum i kriptovalute, kolovoz 2019.
https://www.coindesk.com/information/what-is-ether-ethereum-cryptocurrency/

[10] Upute za razvoj Ethereuma, kolovoz 2019.
https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial

[11] Ethereum blockchain, kolovoz 2019.
https://blog.ethereum.org/2015/06/26/state-tree-pruning/

[12] Aleth, kolovoz 2019.
https://github.com/ethereum/aleth

[13] Ethereum virtualna mašina, kolovoz 2019.
https://blockgeeks.com/guides/ethereum/#The_Ethereum_Virtual_Machine

[14] Solidity, kolovoz 2019.
https://blockgeeks.com/guides/solidity/

[15] Bitcoin, kolovoz 2019.
https://coincentral.com/ethereum-mining-vs-bitcoin-mining-which-is-more-profitable/

SAŽETAK

Naslov: Ethereum algoritam

Kriptovalute su novi oblik digitalnog novca koji funkcionira i nastaje na temelju složenih kriptografskih algoritama na računalnim mrežama. Ethereum je open-source program, zasnovan na blockchain tehnologiji, koja dozvoljava drugim programerima da grade i razvijaju decentralizirane aplikacije. Diplomski rad podijeljen je u tri glavna dijela. U prvom dijelu objašnjeni su osnovni pojmovi kako bi se razumjelo što su kriptovalute. U drugom dijelu rada objašnjeno je što je Ethereum, a u trećem dijelu opisani su osnovni principi rada i dijelovi koda Ethereum algoritma i programiranje u solidity programskom jeziku.

Ključne riječi: Kriptovalute, Blockchain, Ethereum, računi, rudarenje, hash rate, pametni ugovori

ABSTRACT

Title: Ethereum algorithm

Cryptocurrencies are a new form of digital money that works and is generated by complex cryptographic algorithms on computer networks. Ethereum is a blockchain-based open-source program that allows other developers to build and develop decentralized applications. Graduate paper is divided into three main parts. The first part explains everything basic to understand what cryptocurrencies are. The second part explains everything you need to understand about Ethereum, and the third part describes the basic principles of operation and code sections of the Ethereum algorithm and programming in a solidity programming language.

Key words: Cryptocurrency, blockchain, Ethereum, accounts, mining, hash rate, smart contract

ŽIVOTOPIS

Ivan Kunsabo rođen je u Osijeku 4. lipnja 1994. godine u kojemu živi od tada. 2001. godine kreće u Osnovnu školu Frana Krste Frankopana u Osijeku. 2009. godine završava osnovno školovanje i kreće u Prirodoslovnu i Matematičku gimnaziju u Osijeku, koju završava 2013. godine. Iste godine upisuje Elektrotehnički fakultet u Osijeku, preddiplomski studij, smjer Računarstvo. 2016. godine završava preddiplomski studij, te upisuje diplomski studij, smjer procesno Računarstvo koji završava 2019. godine.

PRILOZI

Prilog na CD-u.