Upotreba blockchain tehnologije za spremanje podataka

Petrović, Karlo

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:997981

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

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

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

UPOTREBA BLOCKCHAIN TEHNOLOGIJE ZA SPREMANJE PODATAKA

Diplomski rad

Karlo Petrović

Osijek, 2018.

Sadržaj

1. UVOD 1
2. BLOCKCHAIN 2
2.1. Kriptografske hash funkcije 2
2.1.1. Značajke kriptografskih hash funkcija 2
2.1.2. Hash funkcije u blockchainu 3
2.2. Stvaranje blokova 3
2.3. Spremanje podataka u blokove 8
2.3.1. Alternativni načini spremanja podataka u blockchain 9
3. UPOTREBA BLOCKCHAINA NA PRAKTIČNOM PRIMJERU 12
3.1. Prijedlog praktičnog primjera 12
3.1.1. Solidity 14
3.1.2. Remix IDE 15
3.1.3. Metamask 15
3.1.4. Ethereum test network 16
3.2. Programska realizacija 17
3.3. Izgled programa 23
4. ZAKLJUČAK 31
LITERATURA 32
SAŽETAK 33
ABSTRACT 34
ŽIVOTOPIS 35

1. UVOD

Termin blockchain u posljednje vrijeme, zahvaljujući porastu vrijednosti kriptovaluta, poglavito Bitcoina, postaje sve popularniji. Blockchain predstavlja lanac zapisa, tj. podatkovnih blokova, koji su povezani pomoću kriptografije. Svaki novi blok, nadovezuje se na prošli i njegova vrijednost ovisi o vrijednosti prošlog bloka. Jedna od glavnih prednosti blockchaina je eliminacija nezavisnog kontrolora, npr. banke, što blockchain čini nezavisnim *peer-to-peer* sustavom.

Zadatak ovog rada je detaljno objasniti funkcioniranje blockchain tehnologije te njegovu upotrebu pri spremanju podataka. Prvo će se objasniti kriptografske hash funkcije i njihova uloga u blockchain tehnologiji. Zatim će biti opisano kako se stvaraju blokovi i na koji način se spremaju podaci u blokove. Bit će objašnjene prednosti i nedostaci u odnosu na klasično spremanje podataka te će se na praktičnom primjeru pokazati upotreba tehnologije.

2. BLOCKCHAIN

Prvi blockchain opisala je nepoznata osoba koja se krije iza imena Satoshi Nakamoto u 2008. godini, kao osnova decentraliziranog sustava plaćanja pod nazivom Bitcoin. Opisan je kao „Elektronički sustav plaćanja koji se bazira na kriptografskom dokazu, umjesto na vjerovanju, dozvoljavajući dvjema voljnim stranama direktne transakcije bez potrebe za trećom, neovisnom stranom“ [1]. Riječ blockchain je izvedenica riječi „block“ i „chain“ odnosno lanac blokova. Prvi blok je izrudaren na početku 2009., čime je započeta nova revolucija u informacijskoj tehnologiji. Decentralizacija kao glavna odlika blockchaina, predstavlja uvod u Web 3.0, gdje će „Dapps“ odnosno decentralizirane aplikacije biti posvuda.

2.1. Kriptografske hash funkcije

Kriptografske hash funkcije, predstavljaju osnovu blockchain tehnologije, a to su funkcije koje pretvaraju proizvoljan niz znakova, u fiksni niz znakov**a** koji se naziva „hash“. Jedna od najvažnijih svojstava hash funkcija jest ta da mogu primiti bilo koji broj znakova (između 1 i beskonačno), npr „a“, „riječ“, „proizvoljna rečenica“ ili čak cijeli rječnik, a kao izlaz vraćaju fiksni broj znakova, npr. „CA978112CA1BBDCAFAC231B“.

2.1.1. Značajke kriptografskih hash funkcija

Da bi se kriptografska hash funkcija smatrala sigurnom, mora posjedovati određena svojstva. Prvo svojstvo je da hash funkcija mora biti deterministička. To znači da za svaki određeni ulaz, svaki put mora dati isti izlaz. Drugo svojstvo jest da kriptografska hash funkcija mora biti računalno efikasna, odnosno pretvaranje poruke u hash, ne bi trebala trajati relativno dugo. Treće svojstvo hash funkcija je to da moraju biti otporne na koliziju. To znači da je gotovo nemoguće da dva različita ulaza, daju isti hash. Riječ „gotovo“ zbog toga što je matematički nemoguće postići stopostotnu otpornost na koliziju. Razlog tome je što postoji beskonačan broj ulaza (zbog neograničene veličine ulazne poruke), a izlaz ima ograničeni broj kombinacija koje može prikazati zbog fiksne veličine hasha. „Gotovo nemoguće“ znači da je astronomski mala šansa da će se pronaći dva različita ulaza, koja daju isti izlaz. Četvrto svojstvo glasi: „svaka mala promjena na ulazu, čini veliku promjenu na izlazu“. To znači da će se hash od ulaza „a“ i ulaza „A“ međusobno znatno razlikovati te neće imati nikakve sličnosti kojima bi se moglo povezati da se radi o sličnim ulazima.

Tab. 2.1. Razlika hasha između velikog i malog slova

Tekst	**Sha-256 hash**
A	559AEAD08264D5795D3909718CDD05ABD49572E84FE55590EEF31A88A08FDFFD
a	CA978112CA1BBDCAFAC231B39A23DC4DA786EFF8147C4E72B9807785AFEE48BB

Posljednje svojstvo kriptografskih hash funkcija je to da je nemoguće saznati ulaz u kriptografsku hash funkciju na temelju izlaza, osim „brute force“ tehnikom, tako da se moguće poruke „hashiraju“, te se nakon toga uspoređuju s hashom koji se želi dekriptirati. Taj način je vrlo neefikasan ako postoji beskonačan broj mogućih rješenja. Neke od najpoznatijih hash funkcija su MD5, RIPEMD, SHA-1, SHA-2, SHA-3, itd.

2.1.2. Hash funkcije u blockchainu

Bitcoin blockchain koristi se SHA-2 kriptografskom hash funkcijom tj. varijacijom SHA-2 pod nazivom SHA-256, a Ethereum blockchain koristi SHA-3 kriptografsku hash funkciju, tj. varijaciju Keccak-256. Obje hash funkcije daju 256-bitni hash, koji u heksadecimalnom prikazu ima 64 znaka.

Kriptografske funkcije se koriste na više načina u blockchainu, neke od najvažnijih su kreiranje adresa iz javnog ključa, hashiranje transakcija u Merkle stablo, povezivanje blokova u blockchain te rudarenje.

2.2. Stvaranje blokova

Prilikom transakcije kriptovalute, primjerice BTC, od osobe A – osobi B, osoba A tvori transakciju. Transakcija se tvori tako da osoba A ispuni polja transakcije te je potpiše sa svojim privatnim ključem. Nakon toga osoba A prilaže potpisanu transakciju i svoj javni ključ pomoću kojeg se verificira da je osoba A stvarno izdala tu transakciju. Nakon što osoba A objavi transakciju blockchain mreži, svaki čvor će tvoriti svoj transaction pool (koji je približno isti kod svakog čvora), te će pri tom provjeriti određene uvjete, kako bi utvrdio je li transakcija validna. Transaction pool je skup svih nepotvrđenih transakcija. Neke od provjera su:

- Sintaksa transakcije i podatkovne strukture su točne
- Ulazne i izlazne vrijednosti imaju vrijednosti

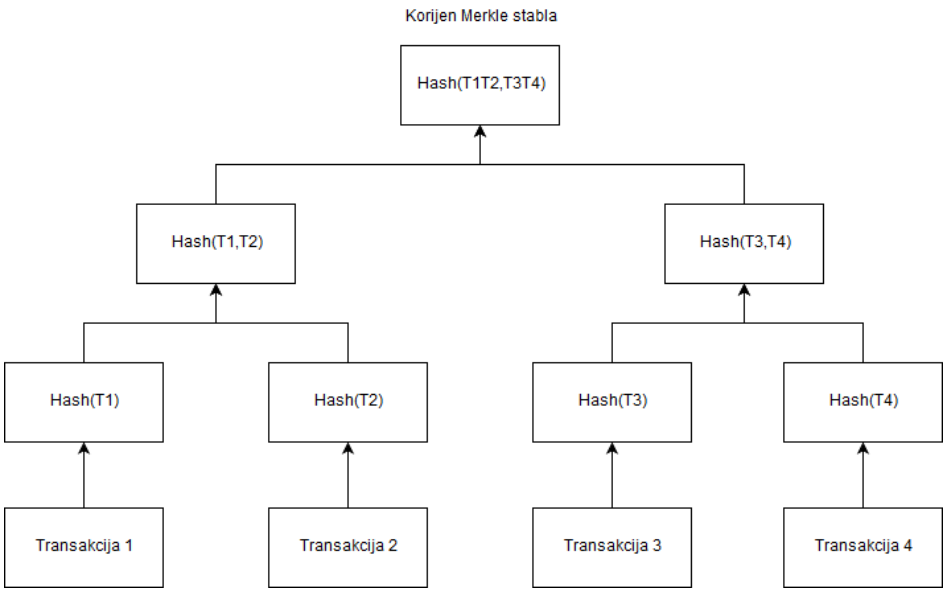
- Transakcija je manja od veličine bloka (1 MB)
- Vrijednosti moraju biti veće od 0 i manje od 21 milijuna
- Hash ulaznih vrijednosti ne smije biti 0
- Veličina transakcije je veća ili jednaka 100 bajtova
- Broj potpisa je manji od maksimalnog broja potpisa
- Spajajuće transakcije moraju postojati
- Ako je transakcija coinbase transakcija, mora imati 100 ili više potvrda
- Za svaki ulaz, mora postojati izlaz i ne smije biti potrošen
- Provjerava se je li svaka vrijednost ulaza u određenom rasponu
- Odbija se ako je ulazna vrijednost veća od izlazne vrijednosti
- Odbija se transakcija čija je vrijednost premala da bi ušla u prazan blok

Ove provjere se mogu promijeniti tijekom vremena zbog razvoja blockchaina. Nakon što je transakcija potvrđena, rudari (eng. *miners*) mogu je početi uvrštavati u moguće blokove te započeti rudariti. Rudarenje je proces zapisivanja transakcije u blockchain, tako da se prilaže dokaz o radu – „Proof of work“ ili dokaz o ulogu – „Proof of stake“. Proof of work je dokaz da je rudar utrošio određeni rad (u slučaju Bitcoina, to je rad CPU-a, a u slučaju Ethereuma, to je rad radne memorije grafičke kartice), tako da riješi određeni matematički zadatak. Također postoji Proof of stake koji se namjerava koristiti u Ethereum blockchainu u budućnosti. Kako se u proof of work svi rudari natječu koji će prije riješti zadatak i dodati novi blok u blockchain, tako u proof of stake nekolicina rudara dobiva zadatak dodati blok u blockchain. Ideja je da se to omogući jednom ili nekolicini rudara koji imaju najviše Ethera, jer se očekuje da im ne odgovara dodati nevažeće transakcije u blok, zbog toga što bi time vrijednost Ethera zbog toga mogla pasti (tako bi on najviše od svih izgubio), a i zbog toga što se dio njegovog Ethera uzima kao zalog, sve dok se ne dokaže da je blok važeći. Ako rudar u blockchain doda blok koji nije važeći, ostaje bez svog uloga Ethera. Trenutno je ovo samo ideja te se očekuje da će se u budućnosti doći do odgovarajućeg rješenja.

Proces rudarenja odnosno stvaranja blokova je sljedeći:

- Rudari uzimaju verificirane transakcije iz transaction poola kako bi od njih napravili potencijalni novi blok. O broju transakcija koje će se potencijalno dodati u blok, načinu prikupljanja (npr. prvo one s većom provizijom) i o ostalim kriterijima odlučuje rudar.

- Rudari stvaraju Merkle stablo iz transakcija iz prvog koraka, sve dok ne dobiju Merkle korijen. Merkle tree odnosno Merkle stablo je struktura podataka koja se koristi prilikom stvaranja blokova. Na dnu stabla (listovi) su sve transakcije koje su uključene u blok. One se hashiraju s nekim od hash funkcija te se nakon toga, dva dobivena hasha zajedno hashiraju u jedan, sve dok se ne dođe do jednog hasha – koji se zove Merkle korijen. Na temelju Merkle stabla, lako je utvrditi jesu li sve transakcije unutar stabla validne. Ako se bilo koji parametar transakcije promjeni, promijenit će se njegov hash, tako da korijenski hash neće odgovarati onome u zaglavlju bloka. Merkle stablo ne služi za pretraživanje, već služi kao dokaz da su određene transakcije sadržane unutar istoga.



Sl. 2.1. Merkle stablo

- Rudari kreiraju blok koji izgleda kao na sljedećoj slici

BLOK	
b	
4	MAGIC NUMBER
4	BLOCK SIZE
80	BLOCK HEADER
1-9	TX-COUNTER TRANSACTIONS

Sl. 2.2. Izgled bloka

Magic number su prva četiri bajta svakog bloka u blockchainu i svaki blok glavnoj mreži ima isti „čarobni broj“. Npr. u glavnom Bitcoin blockchainu – „0xD9B4BEF9“. Taj broj omogućuje razumjeti gdje određeni blok počinje, a gdje završava.

Block size je veličina bloka. Trenutna veličina Bitcoin bloka je ograničena na maksimalnih 1 MB, dok Ethereum blok nema ograničenja u smislu veličine bloka, već je ograničen s maksimalnom količinom „gas-a“ koji se može iskoristiti prilikom stvaranja bloka.

Block header može sadržavati 80 bajtova i opisan je u idućoj točki.

TX-Counter je broj transakcija sadržan u trenutnom bloku. Može sadržavati cijeli broj od 1 do 9 bajtova.

Nakon broja transakcija, u bloku su ispisane sve transakcije koje bi taj blok trebao uvrstiti u blockchain. Prva transakcija u popisu transakcija je tzv. „coinbase“ transakcija, odnosno transakcija kojom će rudar dobiti nagradu ako uspješno izvrši proof of work i priključi blok blockchainu. U Bitcoin blockchainu nagrada je trenutno 12.5 BTC, a u Ethereum blockchainu 3 ETH.

- Popunjavaju zaglavlje bloka kao na sljedećoj slici [2].

ZAGLAVLJE BLOKA	
b	
4	VERSION
32	PREV. BLOCK HEADER HASH
32	HASH MERKLE ROOT
4	TIME
4	BITS TARGET (DIFFICULTY)
4	NONCE

Sl. 2.3. Izgled zaglavlja bloka

Version u zaglavlju bloka označava trenutnu verziju bloka. Može imati do 4 bajta.

Prev. Block header hash označava hash zaglavlja prošlog bloka – tako nastaje blockchain – nizanjem blokova koje sadrže hash zaglavlja prošlog bloka. To onemogućuje promjenu bilo koje stavke u prošlim blokovima, jer promjenom bilo koje stavke, cijeli hash se drastično mijenja te se odmah može utvrditi gdje je došlo do promjene.

Hash Merkle root označava hash Merkle korijena, odnosno ukupnog hasha svih transakcija sadržanom u ovom bloku.

Time označava vremensku oznaku kada je blok stvoren.

Bits target odnosno težina (eng. *difficulty*) označava kolika je težina rudarenja novog bloka. U Bitcoin blockchainu vrijeme rudarenja jednog bloka trebalo bi trajati 10 minuta, a u Ethereum blockchainu 12 sekundi. Ako nakon određenog broja blokova, to vrijeme bude veće ili manje, „difficulty“ se prilagođava (povećava ili smanjuje) ovisno o tome je li se blokovi rudare prebrzo ili presporo.

Nonce je cijeli broj koji rudar traži kako bi uspješno dodao blok u blockchain.

- Rudari hashiraju zaglavlje bloka s odgovarajućom hash funkcijom. U Bitcoin blockchainu je to SHA-256 hash funkcija, a u Ethereum blockchainu je to Keccak-256. Prilikom hashiranja, potrebno je dobiti hash s određenim brojem početnih nula koji je određen „difficulty-jem“. Što je veći broj nula, to je teže pronaći odgovarajući hash. Hash s određenim brojem početnih nula se traži tako da se povećava „nonce“ sve dok se ne dobije broj manji od zadanog.
- Ako hash odgovara težini zadanoj od strane mreže, blok se dodaje u blockchain i objavljuje se ostalim rudarima, koji lako mogu provjeriti odgovara li „nonce“ hashu, a zatim zaustavljaju svoje rudarenje istoga bloka i prelaze na stvaranje idućeg bloka. Ako je iscrpljen sav raspon „noncea“ koji treba provjeriti, rudar mijenja jedan od elemenata zaglavlja blocka, poništava „nonce“ na 0 i ponavlja prošli korak. Time rudar ima dokaz o radu.
- Vrijeme koje je potrebno da rudar objavi cijeloj mreži svoje rješenje je oko 12 sekundi, a ako za to vrijeme još jedan rudar dođe do rješenja, nastaje tzv. „fork“, odnosno razdvajanje lanca blokova na dva ili više lanca. Taj problem će se ispraviti kroz sljedećih nekoliko blokova, jer će se ostali rudari prikloniti lancu koji ima najveću zbrojenu težinu, a transakcije koje su ostale u dijelu lanca koji nije prihvaćen, odlaze natrag u „transaction pool“ te će nakon

nekog vremena biti ponovno potvrđene. Ovim mehanizmom blockchain onemogućuje varanje, odosno duplo trošenje novca. Npr. osoba A pošalje osobi B 10 BTC kako bi nešto kupila. U isto vrijeme pošalje istih 10 BTC sebi te tako potroši istih 10 BTC dva puta. Moguće je da će neko vrijeme obje transakcije biti potvrđene, ali nakon nekog vremena, jedna transakcija će biti poništena. Zbog toga se osobi B preporučuje sačekati određeni broj potvrda transakcije (jedan blok – jedna potvrda) kako bi bila sigurna da ju osoba A nije prevarila. Što je veći broj potvrda, teže je osobi A natjecati se s ostatkom poštenih rudara, sve dok osoba A nema više od 51% rudara (jer je astronomski mala vjerojatnost da će moći izrudariti više blokova za redom). Siguran broj potvrda u Bitcoin blockchainu smatra se 6.

2.3. Spremanje podataka u blokove

Spremanje podataka u blockchain dolazi kao alternativa na klasične načine spremanja podataka, kao što su tradicionalne baze podataka. Tradicionalne baze podataka (npr. SQL i noSQL) rade na principu klijent – poslužitelj. Podaci se nalaze na centraliziranom poslužitelju, korisnik može dodavati i mijenjati podatke, a za kontrolu pristupa korisniku zadužena je treća strana. Ako je ta treća strana kompromitirana, podaci se mogu zlouporabiti, mijenjati ili čak i obrisati. S druge strane, blockchain se sastoji od mnoštva decentraliziranih čvorova (eng. *nodes*), koji sudjeluju u upravljanju mrežom. Svaki čvor potvrđuje dodavanje podataka u blockchain te se odluke donose na temelju opće suglasnosti, što zlouporabu čini neizglednom.

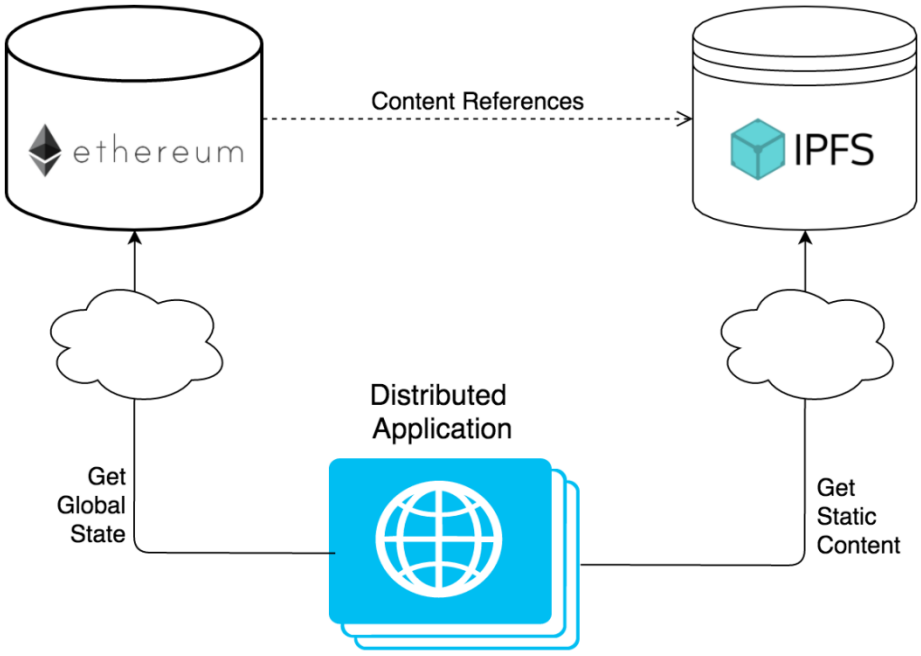
Spremanje podataka u blockchain zvuči primamljivo kada se u obzir uzme da je takav način spremanja siguran, robustan i pouzdan. Podaci koji se spreme unutar bloka su nepromjenljivi i ostaju tamo zauvijek. Podaci koji se spreme u blokove su „once-write, read only“ – što znači kada se jednom zapišu, ne mogu se više mijenjati odnosno mogu se samo čitati. Nepromjenljivost je velika prednost blockchaina koja mu daje visoku robusnost, ali je istovremeno nedostatak prilikom pohrane podataka. Npr. korisnik može promijeniti određene podatke, no svi prethodni podaci će zauvijek ostati u blockchainu te će ostati vidljivi svima. Nepromjenljivost rezultira još jednim nedostatkom - kapacitetom. Kada bi sve aplikacije zadržale svoje podatke u blockchainu, veličina blockchaina bi veoma brzo narasla u abnormalne količine podataka, koje jedan čvor ne bi mogao podnijeti. Zbog toga pohranjivanje podataka samo u blockchain nije dobar izbor za decentraliziranu aplikaciju bogatu podacima. Također u obzir treba uzeti ograničena veličina podataka koja se može spremiti u

blockchain. Bitcoin blockchain ima maksimalnu dopuštenu veličinu bloka od 1 MB, a Ethereum blockchain ima ograničeni broj *gasa* koji se može upotrijebiti prilikom spremanja u blok. Još jedan nedostatak je visoka cijena spremanja podataka. Po današnjim podacima, kada je cijena 1 Ethera oko 200 američkih dolara, a cijena *gasa* 5.3 Gwei, dolazi se do izračuna da cijena spremanja 1 kilobajta u Ethereum blockchain košta oko 60 američkih centi, a ako se uzme u obzir da cijena Ethera na početku godine bila oko 1300 američkih dolara, a cijena *gasa* 3 Gwei, cijena spremanja 1 kilobajta u blockchain tada bi bila oko 2.43 dolara[3]. Vrlo brzo se dolazi do zaključka da u većini slučajeva, ovo je prevelika cijena za spremanje takvog malog seta podataka. U konačnici, najveća mana blockchaina, ne tiče se samo korisnika blockchaina, već cijelog svijeta, a to je negativan utjecaj na okoliš. Prema podacima s web stranice „digieconomist.net“, trenutna procijenjena godišnja potrošnja električne energije koju rudari troše je 73.12 TWh [4], što je veća potrošnja energije koju Austrija potroši cijelu godinu, a tolika potrošnja tvori 35 milijuna tona ugljikovog dioksida godišnje. Za usporedbu, Hrvatska godišnje potroši oko 17TWh.

2.3.1. Alternativni načini spremanja podataka u blockchain

Navedeni problemi pokušavaju se riješiti na različite načine. Budući da Bitcoin i Ethereum blockchain mreže nisu namijenjene niti pogodne za korištenje kao velike baze podataka, nastala su nova rješenja koja bi služila izričito za spremanje podataka, ali i ona imaju svoje prednosti i nedostatke. Neke od tih rješenja su peer-to-peer baze podataka, decentralizirano spremanje u oblaku (eng. *cloud*), distribuirane baze podataka, itd.

Peer-to-peer baze podataka postoje već godinama u različitim oblicima, temelje se na distribuiranim hash tablicama te BitTorrent protokolu. Distribuirane hash tablice omogućavaju hashiranje podataka, koji se zatim vežu uz odgovarajući ključ i spremaju na decentraliziranu distribuiranu mrežu. Uz pomoć tog ključa moguće je pronaći odgovarajući hash. Da bi se podaci dijelili, potrebno ih je postaviti na jedno ili više računala. Podaci će se preuzeti samo ako ih netko treba. Sadržaj je adresiran, pa je nemoguće krivotvoriti sadržaj po datoj adresi. Podaci se mogu preuzeti zahvaljujući BitTorrent protokolu. Služi samo statične datoteke, te se ne mogu izmijeniti niti ukloniti nakon što u postavljene. Negativna strana je to što se podaci ne mogu preuzeti ako računalo nije dostupno na mreži. Jedno od najpoznatijih implementacija ove tehnologije je IPFS (eng. *InterPlanetary File System*)[5]. Ideja IPFS-a je ta da se samo dobiveni hash spremi u blockchain, umjesto cijelog skupa podataka, koji je spremljen na distribuiranoj mreži.



Sl. 2.4. Način rada IPFS-a

Decentralizirano spremanje u oblaku uklanjaju ograničenja IPFS-a. Podaci su spremljeni u oblaku, kao npr. Google Drive. Razlika je u tome što se sadržaj nalazi na korisničkim računalima koja nude prostor na tvrdom disku za iznajmljivanje, a ne u podatkovnim centrima. Ova vrsta tehnologije je vrlo pouzdana, brza te ima ogroman kapacitet. Mogu se spremati samo statični podaci, a datoteke se nalaze na iznajmljenom hardveru te se plaća. Neki od primjera ove tehnologije su Storj i Ethereum Swarm. Storj se temelji na Ethereum blockchainu te se koristi svojim tokenom Storj. Podaci se dijele na više komada te se spremaju po cijeloj mreži. Razlika između prošlog primjera i ovog je to što u ovom primjeru samo vlasnik podataka zna njihovu lokaciju. Dijelovi podataka i njihove lokacije se mogu otkriti samo uz pomoć privatnog ključa. Vlasnici hardvera na kojem se podaci nalaze, moraju svakih sat vremena dokazati da još uvijek imaju vaše dijelove podataka, inače neće biti plaćeni Storj tokenom [6].

Ties DB još je jedno rješenje koje pokušava riješiti probleme spremanja podataka u blockchain. Ova platforma također funkcionira kao decentralizirana baza podataka, ali nudi rješenja koja ostale platforme nemaju. Otpornost na problem Bizantinskih generala je jedan od njih. Zatim nudi brisanje podataka, spremanje dinamičkih i strukturiranih podataka te pretraživanje[7]. Ties DB također

koristi set privatnih i javnih ključeva kako bi se osigurala nemogućnost neovlaštene promjene podataka. Svi podaci su javno vidljivi, a pretraživanje i izmjena podataka se naplaćuje.

BigChainDB projekt je koji tvrdi da rješava problem pohrane podataka i brzine transakcije[8]. Posjeduje najbolje karakteristike Bitcoin blockchaina i distribuiranih baza podataka, kao što su nepromjenjivost, decentraliziranost, brze transakcije, mogućnost upita te dozvola. BigChainDB građen je na klasteru RethinkDB. BigChainDB ga koristi za pohranu svih blokova i transakcija. Zbog toga omogućava veliku brzinu propusnosti podataka. Svi BigChainDB čvorovi povezani su s clusterom i imaju puni pristup za pisanje u bazu podataka. Postoji mogućnost spremanja podataka na javni i privatni blockchain.

3. UPOTREBA BLOCKCHAINA NA PRAKTIČNOM PRIMJERU

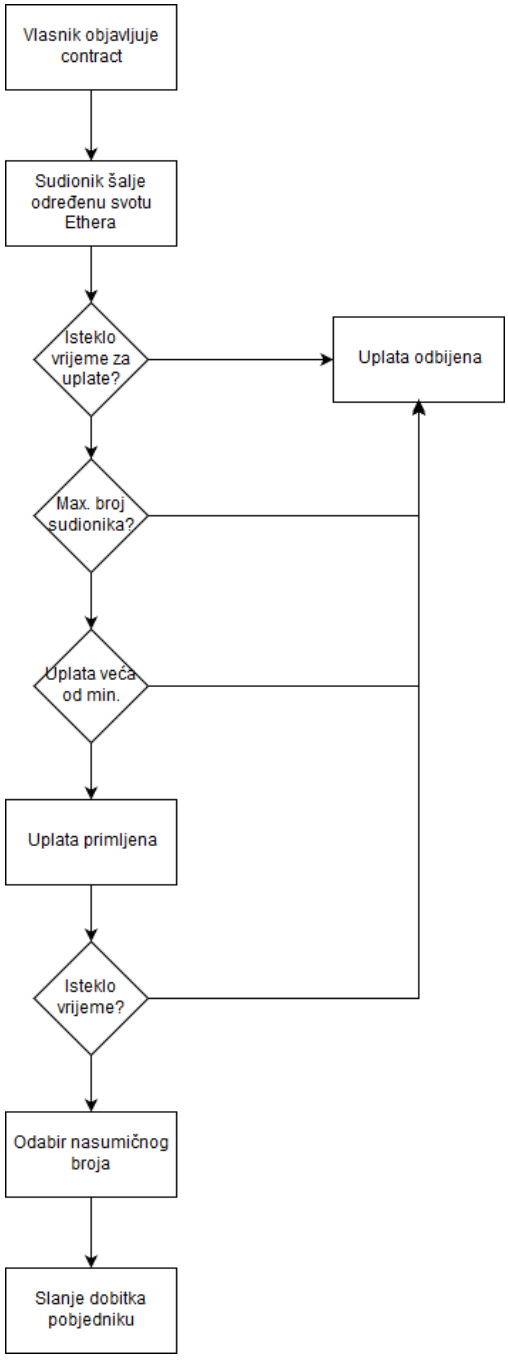
Na temelju tehnologije opisane u prošlom poglavlju, izrađuje se programsko rješenje kojim se prikazuje upotreba blockchaina na praktičnom primjeru.

3.1. Prijedlog praktičnog primjera

Cilj je izraditi Ethereum pametni ugovor (eng. *smart contract*) koji funkcionira na principu vrlo sličnome loto igri na sreću. Osnovna ideja ovog pametnog ugovora je omogućiti ograničenom broju korisnika uplatu određene svote ETH (Ether) kriptovalute na taj pametni ugovor, nakon čega ulaze u izbor za osvajanje cjelokupne svote Ethera. Pravila koja su određena ovim pametnim ugovorom su:

- Ograničeni broj uplata za loto igru
- Uplate za loto igru traju ograničeno vrijeme
- Odabiranje sretnog dobitnika određuje se izvlačenjem nasumičnog broja
- Izvlačenje nasumičnog broja se ne može pokrenuti prije nego li istekne vrijeme za uplatu
- Isplata dobitniku se ne može izvršiti prije isteka vremena i izvlačenja nasumičnog broja
- Jedan korisnik može imati više uplata

Vlasnik pametnog ugovora je osoba koja ga implementira, tj. objavljuje na blockchain mrežu. Prilikom objave vlasnik mora odabrati koliko će maksimalno uplata moći biti uplaćeno, koliko će loto igra trajati te iznos minimalne uplate u ETH kriptovaluti. Nakon što vlasnik objavi pametni ugovor na blockchainu, korisnici imaju propisano vrijeme za prijenos sredstava sa svog virtualnog novčanika za kriptovalute (eng. *wallet*), na adresu pametnog ugovora. Iznos sredstava ne može biti manji ili jednak minimalnom iznosu koji je vlasnik propisao. Kada istekne vrijeme za uplate, bilo koja osoba može pokrenuti postupak odabira nasumičnog broja koji će odabrati pobjednika. Nakon što je nasumični broj izvučen, također bilo koja osoba može pokrenuti isplatu pobjedniku.



Sl. 3.1. – Dijagram tijeka programa

Programsko rješenje razvija se koristeći Solidity programski jezik te Remix integrirano razvojno okruženje, a kao virtualni novčanik za kriptovalute koristit će se dodatak za pretraživač pod imenom Metamask. Budući da je za funkcioniranje Ethereum pametnog ugovora potrebna Ether kriptovaluta,

testiranje u fazi programiranja bilo bi poprilično skupo. Zbog toga se koristi Ethereum test network, tj. testna mreža na kojoj se Ether može dobiti besplatno.

3.1.1. Solidity

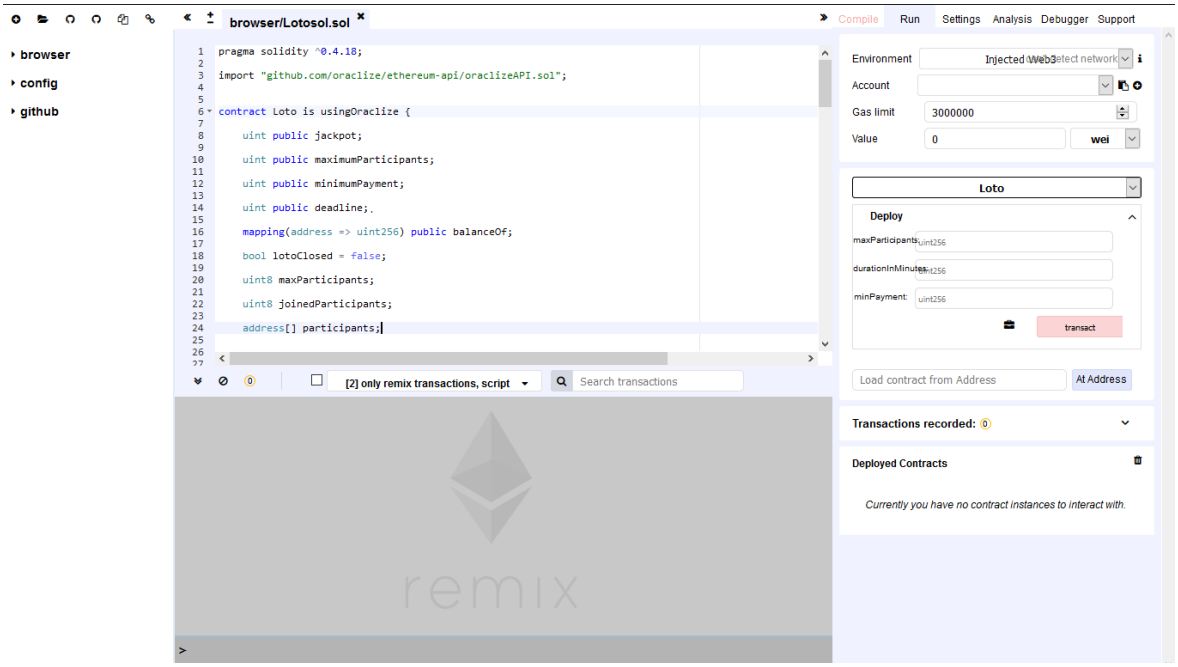
Solidity je programski jezik namijenjen programiranju pametnih ugovora. Nastao je pod utjecanjem drugih programskih jezika, kao što su C++, Python te najviše JavaScript-a, kako bi bio što poznatiji web programerima[9]. Napravljen je kako bi funkcionirao s EVM (eng. *Ethereum virtual machine*). EVM je virtualni stroj na kojemu se pokreće kod Ethereum pametnog ugovora. Ethereum virtual machine radi samo s byte-kodom, koji se dobiva korištenjem ABI-a. ABI (eng. *application binary interface*) služi za kodiranje i dekodiranje Solidity-a u byte-kod i suprotno.



Sl. 3.2. Solidity logo

3.1.2. Remix IDE

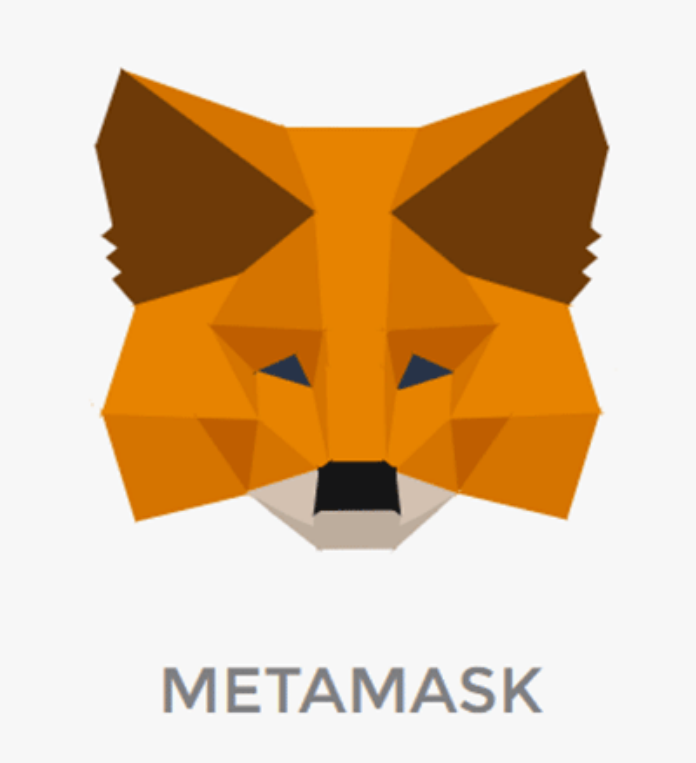
Remix IDE (eng. *integrated development environment*) je integrirano razvojno okruženje koje omogućuje programiranje, testiranje, debugiranje i implementaciju Solidity pametnih ugovora direktno iz internet pretraživača [10].



Sl. 3.3. Izgled razvojnog okruženja Remix

3.1.3. Metamask

Metamask je virtualni novčanik za Ether ili ERC20 tokene, koji je integriran u internet pretraživač. Služi kao posrednik između internet pretraživača i Ethereum blockchaina [11]. Pogodan je za programiranje pametnih ugovora zbog toga što omogućava rad na Ethereum Testnet-u tj. testnoj mreži Ethereuma.



Sl. 3.4. Metamask logo

3.1.4. Ethereum test network

Ethereum test network je testna mreža koja simulira Ethereum blockchain i namijenjena je za testiranje i razvijanje pametnih ugovora, prije nego li uđu u pravi Ethereum blockhain. Pogodnosti Ethereum testne mreže je to što kriptovaluta Ether nema vrijednost (tzv. *faux-ether*) i može se dobiti besplatno. To omogućuje programerima testiranje bez korištenja pravog Ethera koji ima stvarnu vrijednost. Testna mreža koja se koristi u ovom primjeru jest Ropsten, koja je najsličnija pravom Ethereum blockchainu zbog *proof of work* odnosno mogućnosti rudarenja.

3.2. Programska realizacija

Programiranje ovog primjera odvijat će se na Ropsten testnoj mreži. Zbog toga je potrebno konfigurirati Metamask odabirom Ropsten testne mreže, umjesto glavne Ethereum mreže.

Budući da bi prilikom programiranja i testiranja bilo preskupo koristiti pravi Ether, koristi se Ether faucet (slavina). Ona omogućava preuzimanje besplatnog Ethera, koji se može koristiti samo na Ropsten testnoj mreži.



Sl. 3.5. Dobivanje besplatnog Ethera

Nakon preuzimanja besplatnog Ethera, u Remix razvojnom okruženju potrebno je napraviti datoteku s nazivom našeg pametnog ugovora, nakon kojega slijedi ekstenzija „.sol“, predstavljajući Solidity programski jezik..

Na početku koda navodi se verzija Solidity programskog jezika, koja je u ovom slučaju 0.4.18. Zatim je potrebno uvesti „OraclizeAPI.sol“ pametni ugovor, čime nasljeđujemo njegove funkcije. Ovaj pametni ugovor će se koristiti kao posrednik u izvlačenju dobitnika.

```
1.  pragma solidity ^0.4.18;
2.
3.  import "github.com/oraclize/ethereum-api/oraclizeAPI.sol";
```

Programski kod 3.1. Solidity verzija i uključivanje drugog ugovora

Pametni ugovor u Solidity-ju je kao klasa u objektno-orijentiranim programskim jezicima, mogu nasljeđivati druge ugovore te koriste različite elemente kao što su npr. funkcije „functions“ i događaji „events“. U ovom slučaju, ugovor se zove Loto, te nasljeđuje „usingOraclize“ ugovor iz

„OraclizeAPI.sol“ datoteke. Nakon toga slijedi deklariranje varijabli koje će se koristiti u daljnjem kodu, gdje prvi dio označava tip varijable („uint“; „bool“, „address“, itd.), a drugi dio označava ime varijable (npr. „jackpot“, „participants“). Sve „public“ varijable automatski dobivaju getter metodu. Varijabla „address[] participants“ koristi se za bilježenje adresa sudionika koji su izvršili uplatu. Tip podatka ove varijable je adresa te je ona niz. U nju se bilježi redni broj uplate počevši od nule te adresa sudionika koji je poslao uplatu.

```
1.  contract Loto is usingOraclize {
2.
3.      uint public jackpot;
4.
5.      uint public maximumParticipants;
6.
7.      uint public minimumPayment;
8.
9.      uint public deadline;
10.
11.     mapping(address => uint256) public balanceOf;
12.
13.     bool lotoClosed = false;
14.
15.     uint8 maxParticipants;
16.
17.     uint8 joinedParticipants;
18.
19.     address[] participants;
```

Programski kod 3.2. Deklaracije varijabli

Mapping (mapiranje) predstavlja strukturu sličnoj hash tablicama tako da oboje imaju parove ključ – vrijednost, ali razlika je u tome što mapping na početku programa inicijalizira sve ključeve koji su određenog tipa. Vrijednosti se mogu dodavati, ali ključevi ne. Pomoću ovoga se provjerava je li određena adresa zapisana.

```
1.  mapping (address => bool) participantsMapping;
```

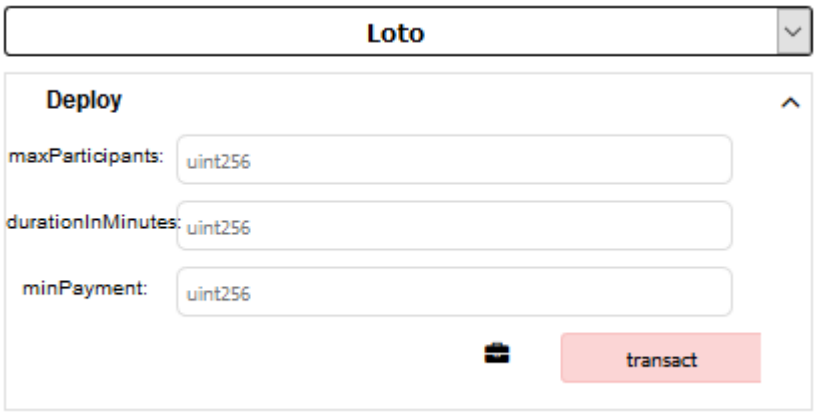
Programski kod 3.3. Mapiranje

Metoda Loto je konstruktor i naziv mora biti isti kao i naziv ugovora. Nakon što je ugovor implementiran, ove postavke se više ne mogu mijenjati. Ovaj konstruktor prima vrijednosti „maxParticipants“, koja označava maksimalan broj uplata, „durationInMinutes“ koja označava trajanje loto igre te „minPayment“ koja označava minimalni iznos koju sudionik može uplatiti. Te

vrijednosti određuje vlasnik ugovora tijekom objavljivanja na blockchain. Slika 3.3. Pokazuje kako to izgleda na Remix razvojnom sučelju. Unutar Loto metode, nalazi se „require()“ funkcija koja provjerava zadani uvjet. Ako je uvjet zadovoljen, kod u nastavku se izvršava, a ako uvjet nije zadovoljen, dolazi do pogreške i preostali *gas* se vraća pošiljatelju. Deadline varijabla predstavlja zbroj vremena trenutka objavljivanja ugovora i broj minuta koje je vlasnik ugovora odredio, tj. vrijeme kada prestaju uplate.

```
1.  function Loto(
2.
3.      uint maxParticipants,
4.
5.      uint durationInMinutes,
6.
7.      uint minPayment
8.
9.  ) public {
10.     require(!participantsMapping[msg.sender]);
11.
12.     maximumParticipants = maxParticipants;
13.
14.     minimumPayment = minPayment;
15.
16.     deadline = now + durationInMinutes * 1 minutes;
17.
18. }
```

Programski kod 3.4. Konstruktor



Sl. 3.6. Izgled unosa zahtjeva vlasnika ugovora

Funkcije bez imena su funkcije koje se pozivaju svaki put kada bilo tko pošalje sredstva na adresu ugovora. Payable znači da ovaj ugovor, putem ove metode može primati Ether. Unutar ove metode se događa sljedeće. Potrebno je ispuniti 4 uvjeta: prvi uvjet je da loto ne smije biti završen, drugi uvjet je da broj uplata mora biti manji od maksimalnog broja uplata, treću uvjet je da uplata mora biti veća od minimalne uplate te četvrti uvjet je da uplata ne može biti uplaćena nakon isteka vremena. Nakon toga se adresa sudionika dodaje u mapping uz vrijednost true, kako bi se kasnije moglo potvrditi da je ta adresa poslala sredstva. Također se broj sudionika povećava za jedan te se primljena vrijednost dodaje u ukupni dobitak – „jackpot“. Nakon toga se s „emitom“ događaj zapisuje u blockchain.

```
1.  function () payable public {
2.
3.      require(!lotoClosed);
4.
5.      require(joinedParticipants < maximumParticipants);
6.
7.      require(minimumPayment < msg.value);
8.
9.      require(deadline > now);
10.
11.     participants.push(msg.sender);
12.
13.     participantsMapping[msg.sender] = true;
14.
15.     joinedParticipants ++;
16.
17.     uint amount = msg.value;
18.
19.     balanceOf[msg.sender] += amount;
20.
21.     jackpot += amount;
22.
23.     emit FundTransfer(msg.sender, amount, true);
24.
25. }
```

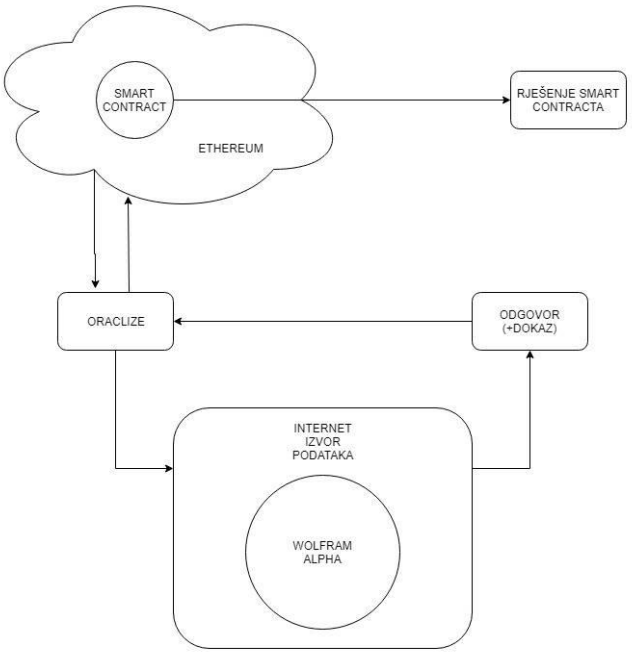
Programski kod 3.5. Payable funkcija

Funkcija „drawWiner“ je najbitnija funkcija u cijelom ugovoru. Ona služi za slanje upita za odabir nasumičnog broja. Kod u Solidty-ju je deterministički, što znači da bi trebao dati iste rezultate na svakom čvoru (node) na kojem je pokrenut, koliko god puta je pokrenut. Zbog toga je nemoguće dobiti nasumičan broj u ugovoru koji bi prilikom svakog pokretanja bio isti. Ne može se dobiti putem određene formule jer što god je u ugovoru, dostupno je svima na mreži, što može dovesti do varanja ili manipuliranja ugovorom. Jedan od mogućih rješenja je dobivanje nasumičnih brojeva iz

blockchaina, kao npr. „block.timestamp“ koji predstavlja vrijeme kada je trenutni blok izrudaren. To rješenje nije dobro, jer ne treba vjerovati rudarima, zbog toga što mogu manipulirati tim podacima, kao npr. jednostavno neuključivanjem određene transakcije u trenutni blok. U ovom primjeru, korišten je servis Oraclize kao posrednik između pametnog ugovora i vanjskog web API-a. Oraclize je servis pomoću kojega ugovor posredno šalje upit izvan blockchaina, na web, a nakon što dobije odgovor, od npr Wolfram Alphe, Oraclize ga prosljeđuje natrag na ugovor. Jedna od pogodnosti Oraclize-a je to što daje dokaz da je taj upit došao sa točno određene stranice. U pametnom ugovoru, putem funkcije „drawWinner“, bilo koji korisnik može, nakon isteka vremena za uplate, pokrenuti postupak traženja nasumičnog broja. Naš ugovor šalje upit Oraclize-u, u kojem je definirano odakle želi upit – u ovom slučaju Wolfram Alpha, te koji je upit – „nasumični cijeli broj između 0 i broj sudionika, umanjen za jedan“. Oraclize se naplaćuje po upitu, stoga pozivatelj funkcije mora platiti transakciju, u ovom slučaju ETH u protuvrijednosti 3 američka centa. Nakon što je Oraclize dobio nasumični broj od Wolfram Alphe, šalje ga u „__callback“ funkciju. U toj funkciji se provjerava je li taj rezultat stvarno dobiven od Oraclize-a, a ne od neke druge, zlonamjerne stranice, te ako je uvjet zadovoljen, nasumični broj se upisuje u „result“ varijablu, a loto se zatvara.

```
1.  function drawWinner() payable {
2.      require(deadline < now);
3.      oraclizeID = oraclize_query("WolframAlpha", strConcat("random integer from 0 to ", uin
    t2str(joinedParticipants-1)));
4.  }
5.
6.  function __callback(bytes32 _oraclizeID, string _result) {
7.      if(msg.sender != oraclize_cbAddress()) throw;
8.      result = _result;
9.     lotoClosed = true;
10. }
```

Programski kod 3.6. Funkcija za izvlačenje nasumičnog broja



Sl. 3.7. Funkcionirajne Oraclize servisa

Nakon što je nasumični broj dobiven, bilo tko može poslati zahtjev za slanje dobitka na pobjednikovu adresu, pozivanjem funkcije „sendToWinner“. Unutar te funkcije se provjerava je li isteklo vrijeme za uplate, te je li loto zatvoren. Nakon što su ti uvjeti zadovoljeni, dobitak se šalje na dobitnikovu adresu tako da se uz pomoć izvučenog broja adresa pobjednika izvlači iz niza. Npr. ako je izvučeni broj 3, pobjednik je korisnik koji je četvrti poslao uplatu.

```
1.  function sendToWinner()  {
2.      require(deadline < now);
3.      require(lotoClosed);
4.      participants[parseInt(result)].transfer(jackpot);
5.  }
```

Programski kod 3.7. Funkcija za slanje dobitka dobitniku

Sljedeće funkcije korištene su kao pomoć prilikom programiranja ovog pametnog ugovora, ali su ostavljene kako bi korisnici imali uvid u određene varijable. Sve funkcije čitaju određenu vrijednost

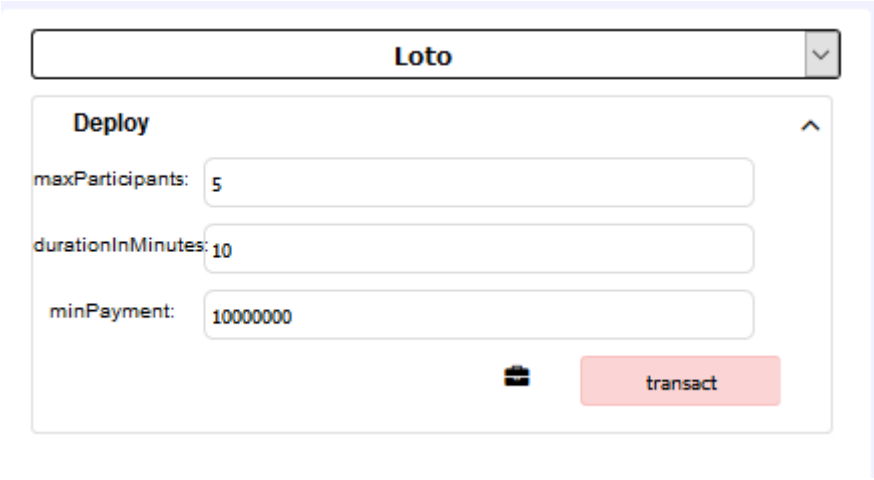
varijable iz blockchaina, npr „result“ odnosno izvučeni nasumični broj, „participants“ adrese korisnika koji su uplatili loto, „joinedParticipants“ broj igrača koji sudjeluju.

```
1.
2.  bytes32 public oraclizeID;
3.
4.  function getParticipants() constant returns (address[]) {
5.      return participants;
6.  }
7.
8.  function showRandomNumber() constant returns (uint) {
9.      return parseInt(result);
10. }
11.
12. function showParticipantNumber() constant returns (uint) {
13.     return joinedParticipants;
14. }
```

Programski kod 3.8. Funkcije za provjeravanje variajbli

3.3. Izgled programa

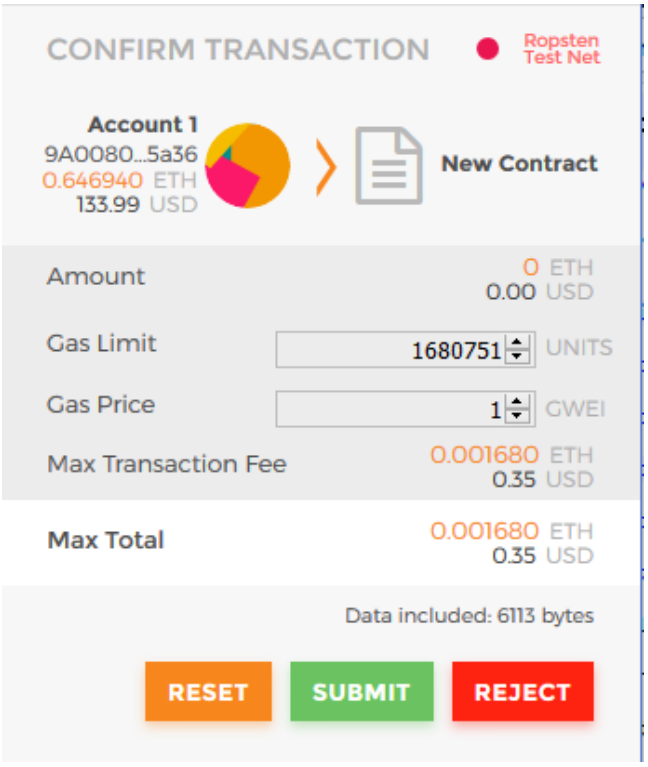
Nakon što se kod kompajlira, vlasnik pametnog ugovora unosi podatke na temelju kojih će se odvijati loto igra. U ovom slučaju je stavljen maksimalan broj uplata 5, trajanje igre 10 minuta te maksimalna uplata 10000000 wei-a.



Sl. 3.8. Unos uvjeta loto igre

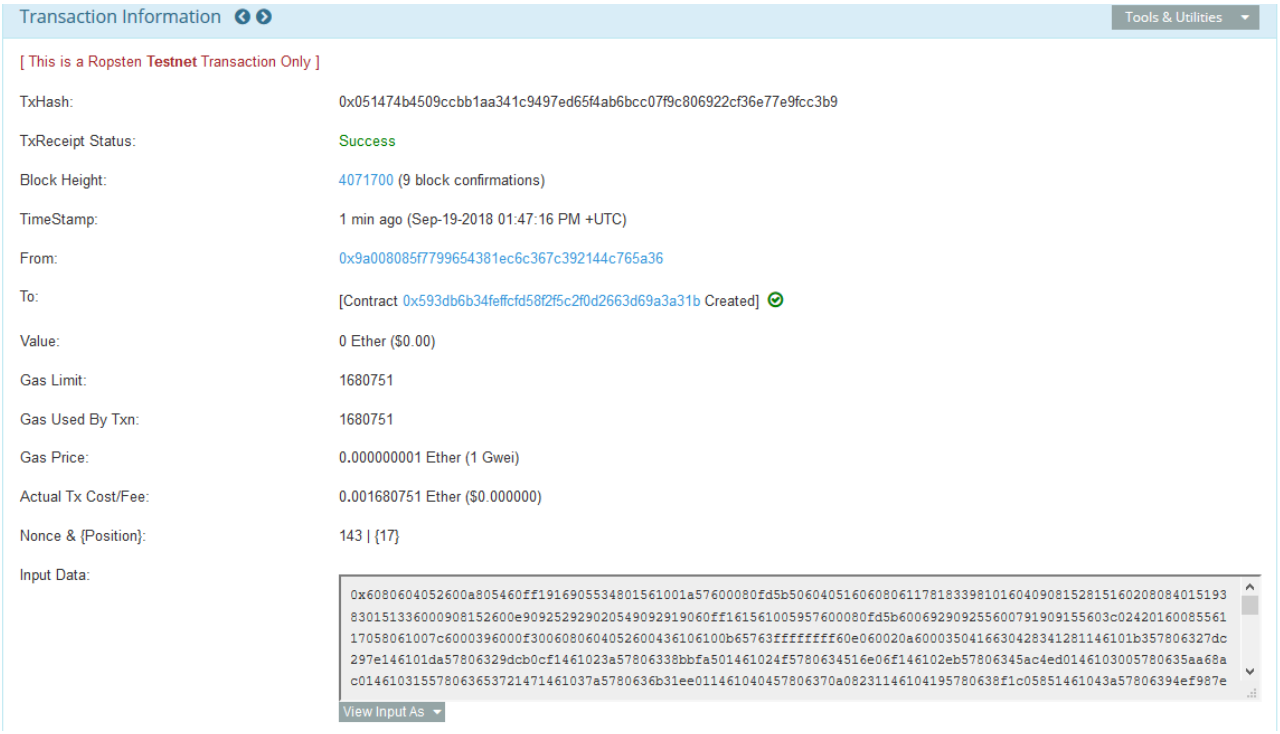
Klikom na „transact“, dobiva se Metamask prozor pomoću kojega se uplaćuje transakcija kojom se pametni ugovor objavljuje na blockchain. Metamask je izračunao moguću cijenu *gasa* koji će biti

potreban da bi se transakcija uspješno izvršila. U ovom konkretnom slučaju, ukupna cijena *gasa* je 1680751.



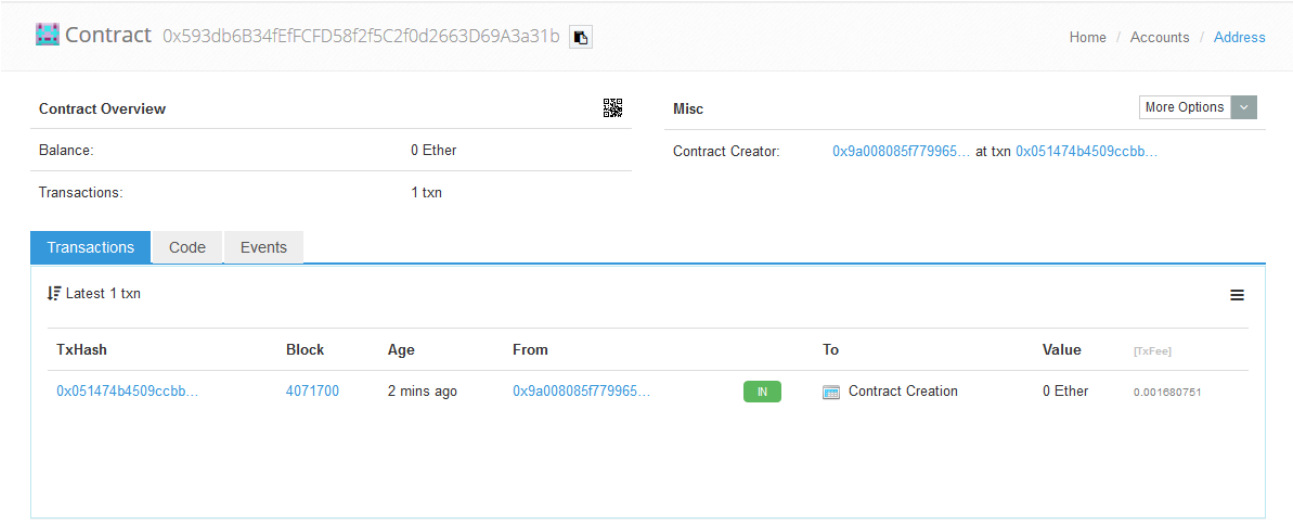
Sl. 3.9. Slanje transakcije pomoću Metamaska

Nakon što je transakcija uspješno dodana u blockchain, ona postaje vidljiva na stranici „etherscan.io“.



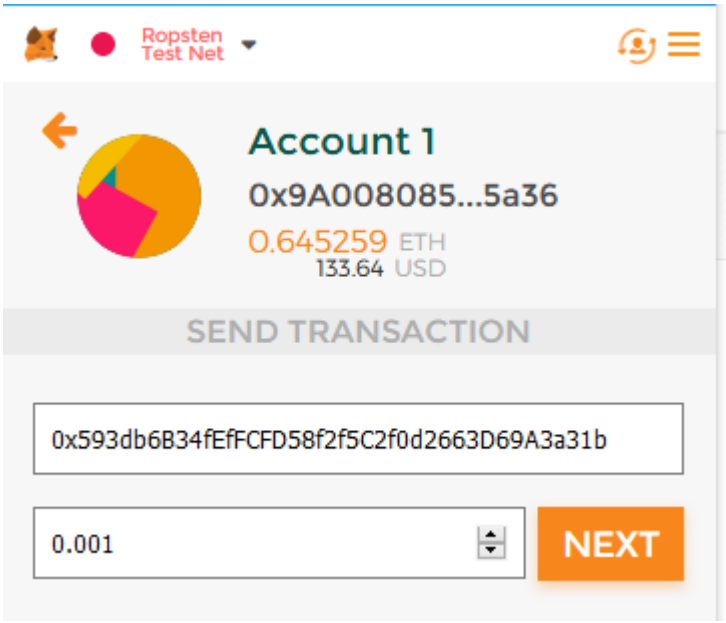
Sl. 3.10. Izgled transakcije na etherscan.io

Objavljeni pametni ugovor također je vidljiv na istoj stranici. Prikazane su informacije o adresi ugovora na koji se mogu slati sredstva za sudjelovanje, trenutni balans Ethera, vlasnik ugovora, transakcije vezane za ugovor itd.

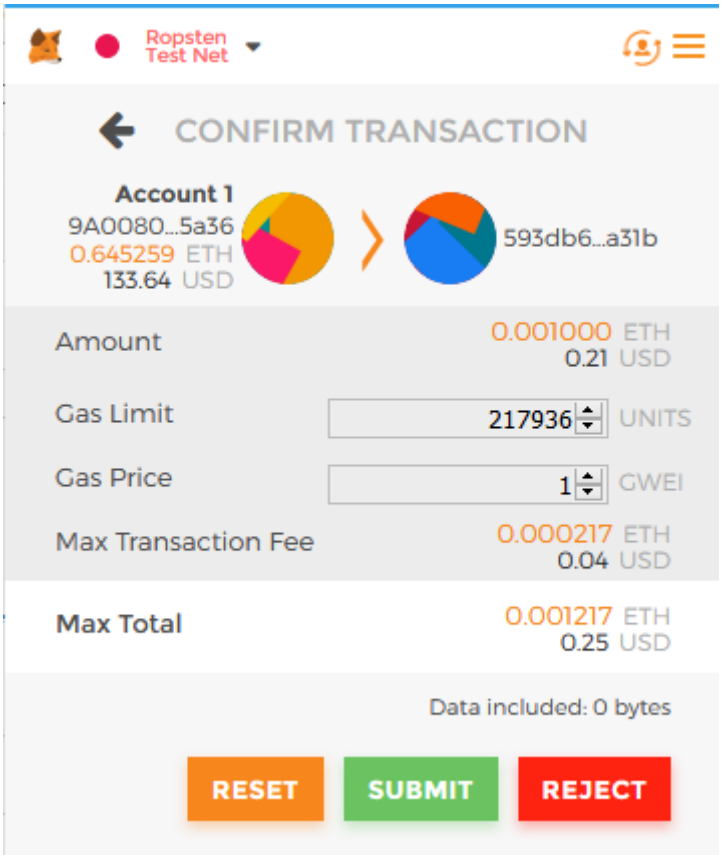


Sl. 3.11. Informacije o pametnom ugovoru na etherscan.io

Nakon što je adresa ugovora postala dostupna, sudionici loto igre mogu uplaćivati sredstva putem Metamaska. Na ovom primjeru, prva uplata je bila 0.001 Ethera. *Gas* iskorišten za ovu transakciju je 217936 i pribraja se ukupnoj uplati koju sudionik mora poslati.

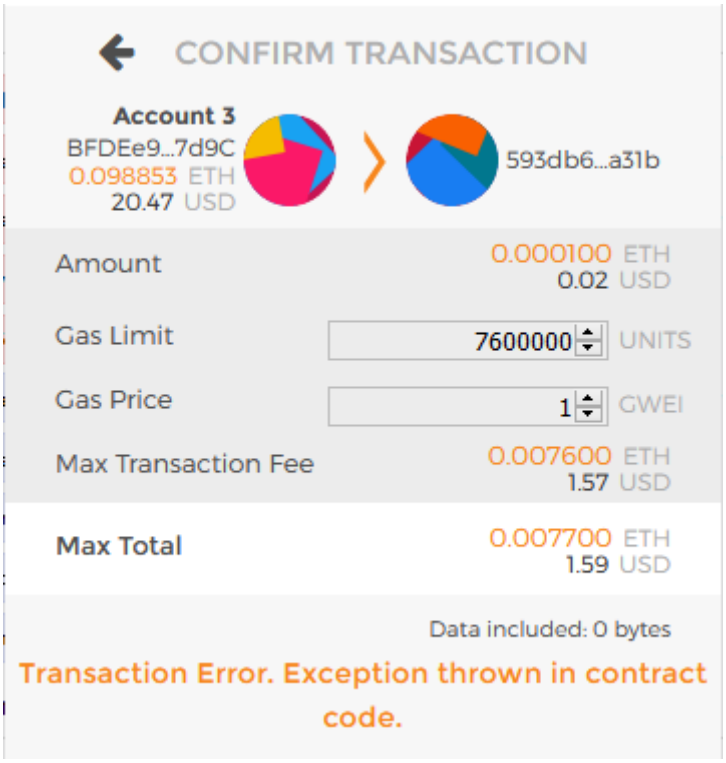


Sl. 3.12. Unos adrese ugovora i željene uplate Ethera



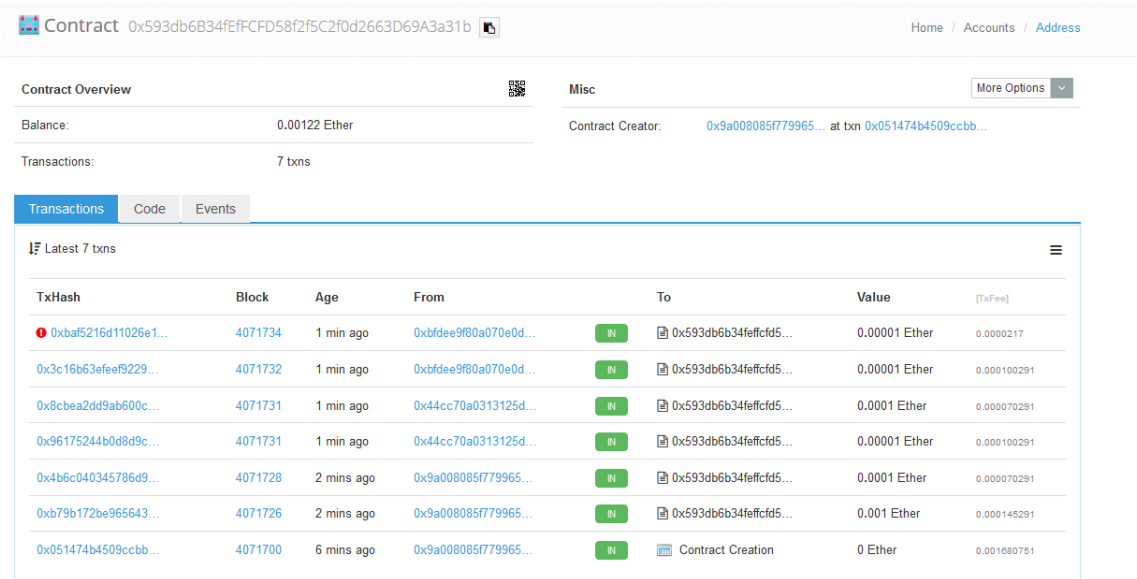
Sl. 3.13. Uplata na adresu ugovora pomoću Metamaska

U slučaju da sudionik želi uplatiti manje Ethera nego li je vlasnik propisao ili je isteklo vrijeme uplata ili je broj sudionika dosegao maksimum, ugovor odbacuje nove uplate.



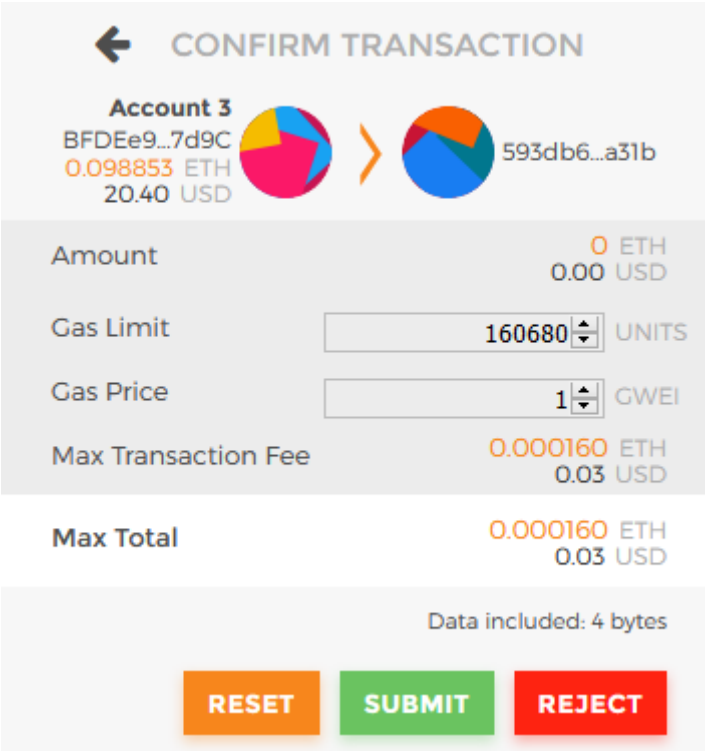
Sl. 3.14. Pametni ugovor odbacuje uplatu

Nakon što je uplata odbačena od strane pametnog ugovora, poslana sredstva se vraćaju sudioniku, a neuspješna transakcija je vidljiva na Etherscanu.



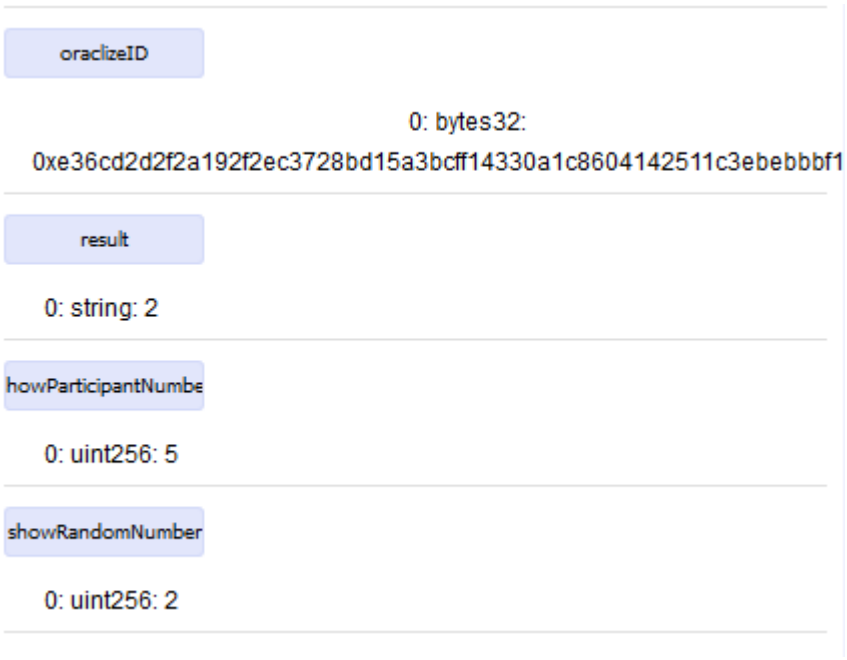
Sl. 3.15. Ugovor nakon završenih uplata, s jednom odbijenom transakcijom

Nakon što je isteklo vrijeme uplata, bilo koji sudionik može pozvati funkciju za izvlačenje nasumičnog broja, kojim bi se odabrao pobjednik. Pozivatelj funkcije mora platiti 3 centa jer je to cijena koju je Oraclize servis prepisao.



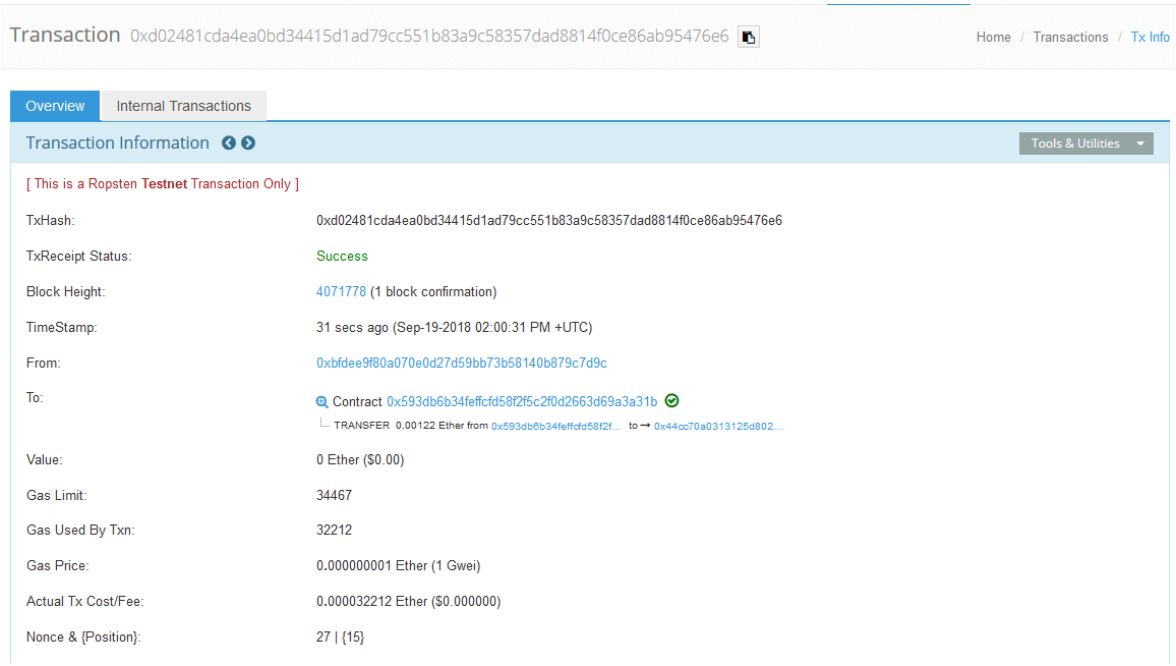
Sl. 3.16. Transakcija za izvlačenje pobjednika putem Oraclize servisa

U pametnom ugovoru se također mogu pozvati funkcije koje čitaju iz blockchaina (čitanje je besplatno) te se mogu vidjeti informacije kao što je OraclizeID – ID pomoću kojega oraclize poziva „__callback“ funkciju, broj sudionika(showParticipantNumber), izvučeni broj (result), itd.



Sl. 3.17. Funkcije za čitanje iz blockchaina

Nakon slanja svih sredstava sretnom dobitniku, transakcija je vidljiva na Etherscanu. Vidljivo je da je poslano 0.00122 Ethera s adrese pametnog ugovora, na adresu dobitnika.



Sl. 3.18. Transakcija dobitka s adrese pametnog ugovora na adresu dobitnika

4. ZAKLJUČAK

Premda je nastao prije više od 10 godina, u posljednje vrijeme riječ blockchain se može čuti gotovo svakodnevno. Razlog tome su njegova revolucionarna svojstva i primjene, koje predvode internet u novu eru, Web 3.0. Najvažnije svojstvo od svih – decentralizacija, potaklo je reevaluaciju starih i razvoj novih ideja. Svjedoci smo nastajanju decentraliziranih aplikacija čiji je temelj upravo blockchain. Decentralizirane aplikacije još su u ranoj fazi, ali postoji sve veća i veća potreba za spremanjem podatka na blockchain.

U ovom diplomskom radu objašnjeno je funkcioniranje blockchaina te nastajanje blokova. Na praktičnom primjeru uspješno je prikazano rješenje spremanja i čitanja podataka u i iz Ethereum blockchaina. Također su objašnjene razlike između spremanja podataka u klasične baze podataka i spremanja podataka u blockchain, a uz to su nabrojane nove solucije koje već postoje ili su u nastanku. Trenutno postoji nekolicina ideja kojima bi se poboljšalo spremanje podataka u blockchain, ali svaka ideja ima svoje prednosti i mane te je na korisniku da odluči koja mu platforma najviše odgovara.

Blockchain nezaustavljivo raste, u 2015. godini tržište blockchaina je vrijedilo 315,9 milijuna, a pretpostavlja se da će do 2024. vrijediti 20 milijardi američkih dolara, s godišnjim rastom od 58.7% [12].

Uz sve pozitivne primjere blockchaina, postoji i velika negativna strana, a to je velik utrošak energije na rudarenje, odnosno spremanje podataka. Kako blockchain konstantno napreduje, ovaj problem se namjerava riješiti u bliskoj budućnosti, prelaskom s *proof of work* načinom rudarenja, na *proof of stake* način, gdje će manje ljudi sudjelovati u rudarenju, što će se pozitivno odraziti na okoliš.

Blockchain je još relativno nova tehnologija, koja je daleko od savršenstva, ali konstantno se radi na njenom poboljšavanju te je nedvojbeno kako će blockchain imati veliku ulogu u budućnosti informacijske tehnologije.

LITERATURA

[1] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, https://bitcoin.org/bitcoin.pdf, pristupljeno: 20.9.2018.

[2] Block, https://en.bitcoin.it/wiki/Block, pristupljeno: 20.9.2018.

[3] Ether gas station, https://ethgasstation.info/, pristupljeno: 20.9.2018.

[4] Bitcoin energy consumption index, https://digiconomist.net/bitcoin-energy-consumption, pristupljeno: 20.9.2018.

[5] InterPlanetary File System, https://ipfs.io/#how, pristupljeno: 20.9.2018.

[6] Storj, https://storj.io/, pristupljeno: 20.9.2018.

[7] TiesDB, https://tiesdb.com/database, pristupljeno: 20.9.2018.

[8] BigChain DB, https://docs.bigchaindb.com/en/latest/, pristupljeno: 20.9.2018.

[9] Solidity, https://solidity.readthedocs.io/en/v0.4.25/, pristupljeno: 20.9.2018.

[10] Remix IDE, https://remix.ethereum.org/, pristupljeno: 20.9.2018.

[11] Metamask, https://metamask.io/, pristupljeno: 20.9.2018.

[12] Data Shows the Growth in Value of Blockchain Markets, https://www.prnewswire.com/news-releases/data-shows-the-growth-in-value-of-blockchain-markets-676985293.html, pristupljeno: 20.9.2018.

SAŽETAK

U ovom diplomskom radu obrađena je tema „Upotreba blockchain tehnologije za spremanje podataka“. Opisane su osnove blockchain protokola i kriptografskih hash tablica te njihova primjena u blockchainu. Objašnjen je način spremanja podataka u blokove, prednosti i nedostatci u odnosu na klasično spremanje podataka te alternativna rješenja za spremanje podataka u blokove. Zatim je na praktičnom primjeru, u Solidity programskom jeziku, izrađen pametni ugovor koji funkcionira kao loto igra. Izrađeni pametni ugovor omogućuje određenom broju sudionika uplatu određene svote Ether kriptovalute čime ulaze u izbor za osvajanje cjelokupnog dobitka. Dobitnik se odabire izvlačenjem nasumičnog broja, nakon čega se iznos cjelokupnih uplata šalje na njegov račun.

Ključne riječi: blockchain, blok, kriptografske hash funkcije, podaci, pametni ugovor, Solidity

ABSTRACT

Title: Using blockchain technology for storing data

Topic of this graduate thesis is „Using blockchain technology for storing data“. The basics of blockchain protocol and cryptographic hash tables are described, along with their application in blockchain. The way in which data is stored in blocks, advantages and disadvantages compared to traditional data storage and alternatives of storing data in blocks are explained. Subsequently, a smart contract is created in Solidity programming language as a practical example. Smart contract which was created, functions as a lottery game. It enables a certain number of participants to enter the game by paying a certain amount of Ether cryptocurrency. The winner is selected by drawing a random number, after which the total amount of payments is sent to its account.

Keywords: blockchain, block, data, cryptographic hash functions, smart contract, Solidity

ŽIVOTOPIS

Karlo Petrović, rođen je u Osijeku 3. Veljače 1995. godine. Pohađao je osnovnu školu Dobriša Cesarić u Osijeku, nakon koje upisuje I. Gimnaziju Osijek. 2013. godine maturira te upisuje preddiplomski studij računarstva na tadašnjem Elektrotehničkom fakultetu. Nakon uspješnog završetka preddiplomskog studija računarstva, dobiva zvanje prvostupnik (baccalaureus) inženjer računarstva (univ. bacc. ing. comp.). Svoje obrazovanje iste godine nastavlja na diplomskom studiju računarstva na fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer Računalno inženjerstvo. Trenutno je na praksi u tvrtki Barrage te je u procesu završetka studija, nakon kojega će steći zvanje magistar inženjer računarstva (mag. ing. comp.).

_____

Karlo Petrović