Primjena pametnih ugovora

Vulić, Ante

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ante Vulić

Primjena pametnih ugovora

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ante Vulić

Matični broj: 44093/15-R

Studij: Informacijski sustavi

Primjena pametnih ugovora

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Neven Vrček

Varaždin, studeni 2020.

Ante Vulić

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovom radu objašnjena je tehnologija blockchain i kako je implementira Ethereum sustav koji omogućuje decentralizirani sustav transakcija te izvođenje pametnih ugovora. Navedena je upotreba pametnih ugovora u stvarnim aplikacijama i implementacije jednog takvog pametnog ugovora koji omogućuje decentralizirano i transparentno digitalno glasovanje.

Ključne riječi: blockchain, pametni ugovori, ethereum, solidity, digitalno glasovanje

Sadržaj

1. Uvod

Tema ovog rada je primjena pametnih ugovora, kako bi mogli razumjeti kada i kako primijeniti te što su pametni ugovori moramo razumjeti i sam mehanizam na kojem se ugovori pokreću. Sustav pod nazivom Bitcoin kreiran je 2009. godine te je on odgovor na dotadašnji monetarni sustav. Kako je Bitcoin sustav zamišljen za prijenos vlasništva tj. sustav plaćanja koji ima jako slabu podršku kreiranje aplikacije koje bi se izvršavale pomoću transakcije, stvoren je novi sustav tzv. Ethereum. Ethereum je globalno računalo pomoću kojeg se može na jednostavan način kreirati aplikacije koje će biti decentralizirane i transparentne. Upravo radi toga je opisana tehnologija Ethereum implementacije blockchain-a[1] i upotreba sustava u svrhe online glasovanja.

[1] Blockchain - tehnologija vođenja evidencije koja stoji iza Bitcoin mreže.

2. Metode i tehnike rada

Prilikom izrade pametnih ugovora koristili su se alati: Ethereum Ropsten, Metamask, Solidity i Remix. **Ethereum** je jedan od testnih Ethereum mreža koje možemo koristiti pored glavnog blockchaina za postavljanje i testiranje ugovora prije nego što stvarno postavimo ugovor na glavni blockchain. **Metamask** je digitalni kripto novčanik[2] u obliku browser ekstenzije. Koristimo ga za kreiranje transakcija za slanje ether-a te pozivanje na izvođenje pametnih ugovora. **Solidity** je jedan od vodećih programskih jezika za kreiranje pametnih ugovora za Ethereum sustav. **Remix** je preglednički oblik kompajlera za Solidity programski jezik za testiranje samih ugovora.

[2] Digitalni kripto novčanik – digitalni novčanici koji omogućuju brzo i sigurno pohranjivanje kripto valuta

3. Blockchain

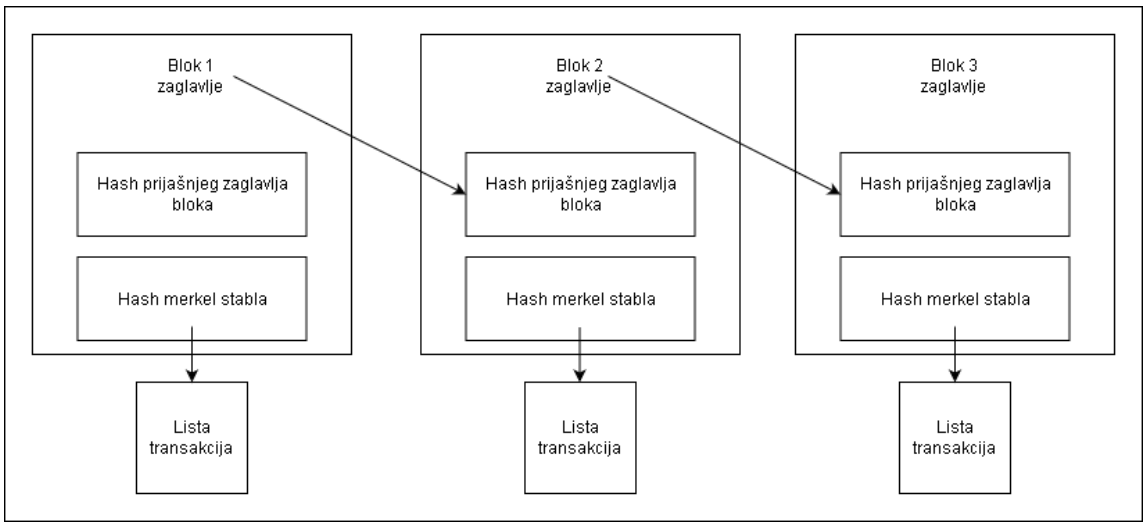
Blockchain ili tehnologija ulančanih blokova danas ima mnogo značenja ponajprije kada se spomene riječ blockchain odmah se pojavljuju kripto valute, međutim to nije uvijek slučaj jer kako postoji mnogo kripto valuta svaka od njih ne implementira blockchain tehnologiju u svoj sustav već koriste neki od drugih načina praćenja stanja transakcija.

Digitalni novac kao što je Bitcoin koji se još naziva kripto valuta koristi blockchain tehnologiju kako bi osigurao vjerodostojnost transakcija koje su nastale u mreži. To postiže pomoću zajedničkog zapisa svih transakcija koji je vidljiv svim sudionicima mreže kako bi se mogle transakcije upisati u zajednički zapis mora doći do sporazumne jednoglasne odluke, konsenzusa, svih članova kojima je dodijeljena mogućnost zapisa samih transakcija u mreži tzv. rudari (*eng. miners*).

3.1. Distribuirana glavna knjiga

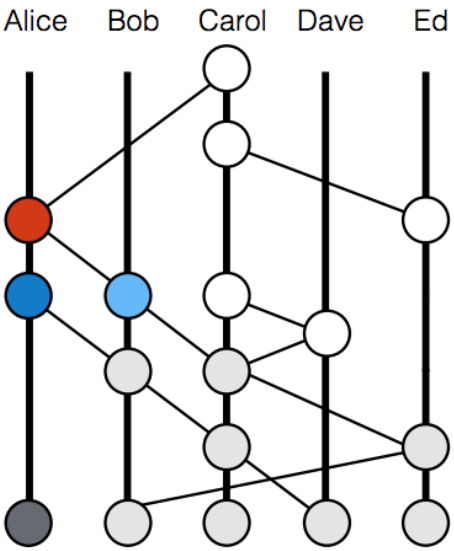
Distribuirana glavna knjiga (*eng. Distributed Public Ledger, kraće DLT*) je tehnologija repliciranog zajedničkog zapisa podataka koja bez centralne vlasti dolazi do sporazumne odluke o unosu podataka u zajednički zapis. Kako zapis postoji u *Peer-to-peer* (P2P) mreži potreban je mehanizam koji osigurava kopiranje zapisa među sudionicima mreže i vjerodostojnost samog zapisa. Tu ulogu ima konsenzus algoritam. Prema [1]

Blockchain je samo jedan tip tehnologije distribuirane glavne knjige koja kao zajednički zapis, kako i sama riječ kaže, koristi niz ulančanih blokova koji sadrže popis transakcije te su povezani, tj. svaki blok ima adresu sljedećeg bloka u mreži. Na slici ispod može se vidjeti pojednostavljena izvedba blockchain sustava. (Slika 1)



Slika 1 Pojednostavljeni blockchain (Izvor: vlastita izrada)

Kod ostalih DLT-ova možemo izdvojiti Hashgraph. Blockchain zapisuje novonastale transakcije grupirajući ih u blokove, kod hasgrapha je to drugačije, nove transakcije se zapisuju redoslijedno kako su nastale u strukturi hash grafa te se za razliku od blockchaina transakcije mogu zapisati paralelno s jednim vremenskim potpisom što omogućuje veću skalabilnost sustava. Konsenzus se postiže pomoću virtualnog glasanja (*eng. Virtual Voting*) i tehnologije tračanja (*eng. Gossip techniques*). Prema [2]



Slika 2 Hashgraph (Izvor: [2])

3.2. Konsenzus mehanizam

Konsenzus mehanizam omogućava sporazum svih sudionika sustava oko zapisa novog podatka u decentraliziranom jedinstvenom zapisu. U dinamičkim sustavima kao što je blockchain, transakcije se zapisuju u jedan zajednički zapis i taj se zapis koristi za izračun salda posjedovanja valuta za svakog korisnika, potrebno je osigurati zapis koji je vjerodostojan, nema kašnjenja, pošten i siguran te su sve zapisane transakcije odobrene od svih sudionika mreže. Sve to omogućuje konsenzus mehanizam gdje set pravila određuje tko sljedeći ima pravo zapisa podatka u zajednički zapis.

Proof of work (POW) je jedan od popularnijih mehanizama za način dolaska do konsenzusa kojeg koristi Bitcoin. Sudionici koji imaju pravo zapisa novih transakcija u

zajednički zapis nazivaju se rudari (*eng.miners*) te je sam postupak poznat kao rudarenje (*eng.mining*). Prilikom kreiranja zahtjeva za transakcijom rudari uzimaju novokreirane transakcije još neodobrene te kreiraju jedan blok, zatim slijedi pogađanje heksadecimalnog broja do određene veličine što iziskuje velik broj kalkulacija, a samim time i sigurnost mreže. Rudar koji prvi pogodi broj ima pravo zapisa, kreiranog bloka transakcija, u zajednički zapis svih transakcija. Prema [3]

Proof of Stake (POS) mehanizam je koji je alternativa za POW, samim time što je energetski učinkovitiji te u teoriji može omogućiti više transakcija po sekundi nameće se kao glavni konsenzus mehanizam za novije kripto valute te i sam Ethereum. Još možemo istaknuti i ***Delegated Proof of Stake***, ***Proof of History***, ***Proof of Authority***, ***Proof of Burn*** kao neke od konsenzus mehanizama. Prema [3]

4. Kripto valute

U ovoj cjelini opisuje se prva implementacija virtualne kripto valute pomoću blockchaina, zvan Bitcoin. Budući da je tema rada pametni ugovori, u sljedećim poglavljima prikazan je detaljniji opis kako radi Ethereum i kako se izvode pametni ugovori na Ethereum virtulanoj mašini (*eng. Ethereum Virtual Machine,* kraće EVM*).*

4.1. Bitcoin

Potaknuto krizom izazvanom pohlepom te samim fraktalnim rezervama bankarskog sustava, 2008. godine pojavio se rad [4] kreatora pod pseudonimom Satoshi Nakamoto koji opisuje i ujedinjuje postojeće metode cyberpunk zajednice [5] kako bi kreirao Bitcoin, prva uspješna implementacija decentralizirane kripto valute. Prema [6]

Bitcoin kao sustav, lanac je digitalnih zapisa baziranih na kriptografskom dokazu koji osigurava vjerodostojnost svakog zapisa, tj. slanja valute drugom sudioniku mreže. Svakom korisniku mreže kreira se jedinstveni javni ključ (*eng.Public key*) iz kojeg se generira adresa Bitcoin novčanika i jedinstveni privatni ključ (*eng.Private key)*. Prilikom slanja Bitcoin-a nekom drugom korisniku potpisuje se javnim ključem transakcija i predaje se mreži te se transakcija provjerava i zapisuje u glavni zapis transakcija koji je u ovom slučaju blockchain zapis. Kako bi se svakom zapisu, tj. transakciji u zapisu moglo vjerovati da je jedinstvena, tj. nije prethodno iskorištena za neku drugu transakciju, tzv. problem dvostrukog trošenja (*eng.Double spending problme*), sustav za konsenzus mehanizam koristi tzv. dokaz o radu (*eng.POW*). Prema [4]

4.1.1. Limitacije

UTXO (*eng.Unspedn Transaction Output*) što predstavlja valutu za slanje u Bitcoin mreži, može biti u vlasništvu ne samo javnog ključa, već i više složene skripte (programskog jezika temeljenog na stogu). Navedeni jezik ne podržava sve moguće operacije Turing računala[3], nema pristup podacima iz blokova blockchain-a što sprječava Bitcoin u kreiranju složeniji pametnih ugovora te su iz tog razloga kreirani drugi sustavi kao što je Ethereum. Prema [7]

4.2. Ethereum

Ethereum, zamišljen je kao opće namjenski programibilni globalni blockchain te nadogradnja postojećeg sustava Bitcoin, kako bi omogućio kreiranje i izvršavanje složenijih operacija, a ne samo praćenje i prijenos vlasništva u obliku digitalne valute. Budući da može izvršavati određeni kod, Ethereum možemo opisati kao globalno decentralizirano računalo koje izvršava računalni kod zvan pametni ugovori. Koristi blockchain tehnologiju kako bi pratio zapis promjene sistemskog stanja gdje se upotrebljava, tzv. „Ether“, interna digitalna valuta Ethereum bockchin-a s kojim mjeri i ograničava nepotrebno trošenje resursa u sustavu te samog sustava. [Prema 8]

Kao i mnogi ostali blockchain sustavi Ethereum ima slične dijelove koji su implementirani na drugačiji način kako bi se koristili u svrhu izvršavanja pametnih ugovora:

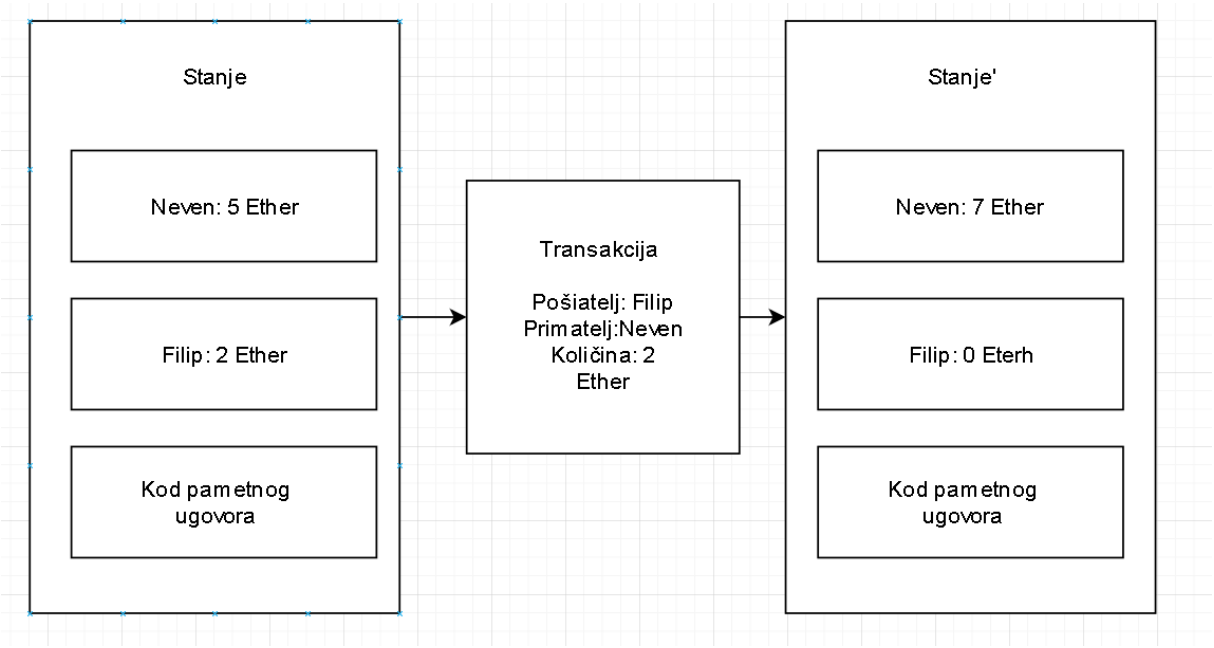
- Peer to peer mreža računala
- Byzentinov tolerantni konsenzus algoritam za sinkronizaciju transakcija
- Internu digitalnu valutu Ether
- Kriptiranje (potpisivanje transakcija)
- Distribuirani zapis svih transakcija

4.2.1.Ethereum globalno stanje

Kao globalno računalo Ethereum prati zapis ne samo svih transakcija i vlasništva nad valutom, već i zapis svih podataka spremljenih u Ethereum globalno stanje. Sustav može zapisati računalni kod i podatke te se koristi blockchain kako bi pratio promjene stanja zapisa i po potrebi učitavao određeni kod ugovora u svoju virtualnu mašinu te izvodio funkcionalnosti koda. Stanje je globalno distribuirano i svako njegovo mijenjanje je podvrgnuto pravilima konsenzusa do kojeg dolazi pomoću POW algoritma. U Ethereum sustavu svjetsko stanje

[3] Turing računala -provjera inteligencije računala ili programske potpore

sastoji se od mapiranja adresa s objektima računa. Svaki račun ima svoju adresu te se izvedbom transakcija prelazi iz trenutnog stanja u sljedeće. Za prebacivanje iz jednog stanja u drugo brine se EVM. Svaki čvor u Peer-to-peer mreži sadrži kopiju EVM-a i potvrđuje izvršavanje pametnih ugovora dok Ethereum blockchain zapisuje proces transakcija i podataka obrađenih pametnim ugovorima. [Prema 8, 9]



Slika 3 Prijelaz stanja (Izvor: Prema[2])

4.2.2.Ether valuta

Kako bi se koristile funkcionalnosti Ethereum sustava koristi se interna valuta **"ether"**, ovdje postoji razlika, Ethereum nije valuta, nego sustav, a ether je sama digitalna valuta koja se šalje u transakcijama. Ether kao i ostale fiat valute mogu se denominirati (Slika 4) na manje jedinice, najmanja moguća jedinica zvana wei, gdje je 1 ether jednak 10^18 wei. U sustavu je uvijek valuta predstavljena kao *unsigned integer* denominacije u wei-u, gdje ako želimo poslati 1 ether, u transakciji je zabilježeno slanje 1 000 000 000 000 000 000 wei kao vrijednost, međutim pomoću viših programskih jezika pri programiranju pametnih ugovora postoje ugrađene konstante za jednostavnije baratanje konverzijom između denominacija i pisanju urednijeg koda. U transakcijama će se spominjati parametar **gas** kao sredstvo plaćanja troška (Slika 5) izvršavanja usluga sustava kao što je pozivanje funkcija pametnih ugovora, gas nije isto što i ether nego je jedinica za sebe te se radi konverzija između gas-a i ether-a prilikom isplate rudarima za obavljeni posao i pružene resurse. Korisnik stavlja cijenu koju je željan platiti za gas i njegovu količinu u svaku transakciju. Gas ima više uloga u sustavu: služi kao ublaživač nagle promjene cijene ether-a i kao nagrada rudarima za sudjelovanje u mreži te

kao osigurač od DoS napad sustava[4] do kojeg može doći generiranjem neograničenog broja transakcija. [Prema 8]

Value (in wei)	Exponent	Common name	SI name
1	1	wei	Wei
1,000	10^3	Babbage	Kilowei or femtoether
1,000,000	10^6	Lovelace	Megawei or picoether
1,000,000,000	10^9	Shannon	Gigawei or nanoether
1,000,000,000,000	10^{12}	Szabo	Microether or micro
1,000,000,000,000,000	10^{15}	Finney	Milliether or milli
1,000,000,000,000,000,000	*10^{18}*	*Ether*	*Ether*
1,000,000,000,000,000,000,000	10^{21}	Grand	Kiloether
1,000,000,000,000,000,000,000,000	10^{24}		Megaether

Slika 4 Denominacija Ether-a (Izvor: [8])

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for an EXTCODESIZE operation.
$G_{extcodehash}$	400	Amount of gas to pay for an EXTCODEHASH operation.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
$G_{txcreate}$	32000	Paid by all contract-creating transactions after the *Homestead* transition.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
$G_{logdata}$	8	Paid for each byte in a LOG operation's data.
$G_{logtopic}$	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.
$G_{quaddivisor}$	20	The quadratic coefficient of the input sizes of the exponentiation-over-modulo precompiled contract.

Slika 5 Lista troškova (Izvor: [9])

[4] Dos napad sustava – eng. Denial-off-service attack je napad na računalo u kojem napadač želi resurse ili servise tog računala učiniti nedostupnim za njihove korisnike.

4.2.3.Račun

U Ethereum sustavu postoje dvije vrste računa: **vanjski račun** (*eng.Externally Owned Account, kraće EOA*) kojim se upravlja pomoću ključeva te **račun pametnog ugovor** koji nema privatni ključ nego je kontroliran kodom napisanim u samom ugovoru. EVM ne razlikuje ta dva računa, a oba računa imaju saldo u wie-u koji se mogu mijenjati slanjem i primanjem ether-a te adresu. Svaka adresa generira se iz privatnog ključa korisnika tako što se generira privatni ključ generatorom slučajnih brojeva te se kriptira pomoću eliptičke krivulje kako bi se dobio javni ključ te hashiranjem javnog ključa Keccak algoritmom dolazimo do adrese računa.

Primjer adrese i privatnog ključa:

- Adresa: 0xe668c9020db045da0cbd4de9581ca8d92713cc0e
- Privatni ključ: 6cdb26d0fdae958f97e12bd5as4d5a4sd54as5a6e9ff093084c637bca5943e2

EOA je kontroliran privatnim ključem kojeg možemo kreirati pomoću digitalnog novčanika npr. *MetaMask*, koji nema kod te može kreirati potpisane transakcije za prijenos ether-a ili pokretanje izvršavanja funkcija pametnih ugovora. **Račun ugovora** sadrži kod pametnog ugovora koji se izvršava u EVM na primljenu transakciju koja sadrži informacije o pozivu funkcija ugovora, time mijenja stanje sustava, piše i zapisuje podatak u internu memoriju EVM-a te može slati i primati ether ili kreirati nove pametne ugovore. Kako račun ugovora nema privatni ključ nije u mogućnosti inicirati transakciju, nego to može samo EOA te sam ugovor reagira na primljenu transakciju pozivanjem funkcija. U tablici (Tablica 1) možemo vidjeti strukturu računa. [Prema 8]

balance	Saldo u ether-u
nonce	Broj transakcija poslanih za EOA ili broj kreiranih ugovora za račun ugovora
codeHash	Hash za EVM kod
storageRoot	256 bit hash od korijenskog čvora Merkel stabla za stanje računa

Tablica 1 Struktura računa (Izvor: [9])

4.2.4.Transakcije i poruke

Transakcije i poruke služe za kreiranje novih ugovora, pozivanje funkcija u pametnim ugovorima te slanje ethera vanjskim računima korisnika ili ugovorima. Kako bi korisnik mogao koristiti usluge Ethereum sustava mora slati valutu ether. To može pomoću **transakcije** koje se inicijaliziraju preko korisničke aplikacije novčanika. Transakcija je objekt koji se šalje u sustav s namjerom dodjele ethera drugom sudioniku mreže, pametnom ugovoru ili pozivanjem funkcija pametnih ugovora te time mijenja interna stanja i zapisuje se u blockchain, pošiljalac potpisuje poruku u kojoj se nalaze podaci o primatelju koji može biti vanjski račun ili pametni ugovor te se ta poruka emitira u sustavu i uvrštava u skup transakcija koje čekaju svoj red validacije i zapis u blockcahain. **Poruka** je slična kao i transakcija samo što je generira pametni ugovor, a ne vanjski račun. Poruka se generira tako što trenutni pametni ugovor pozove računalni kod (*opcode*) *CALL*, koji kreira poruku i emitira je u sustav. U tablici niže (Tablica 2) možemo vidjeti strukturu poruke i transakcije. [Prema 7]

Potrebno je spomenuti poseban slučaj transakcije koja kreira novi ugovor na blockchain-u. Transakcija koja kreira ugovor šalje se na posebnu adresu koja sadrži samo nule, polje primatelja u transakciji sadrži adresu 0x0. Ova adresa nije vanjski račun niti adresa pametnog ugovora, ne može potrošiti ether iz transakcije, a služi EVM-u kao točka za kreiranje novih ugovora. Međutim, ako pošaljemo praznu transakciju bez sadržaja koji je trebao biti kod pametnog ugovora tada dolazi do poništenja ether-a poslanog s tom transakcijom, u suprotnom će se kreirati ugovor i imati početno stanje salda ether-a koliko je bilo poslano u samoj transakciji.[Prema 8]

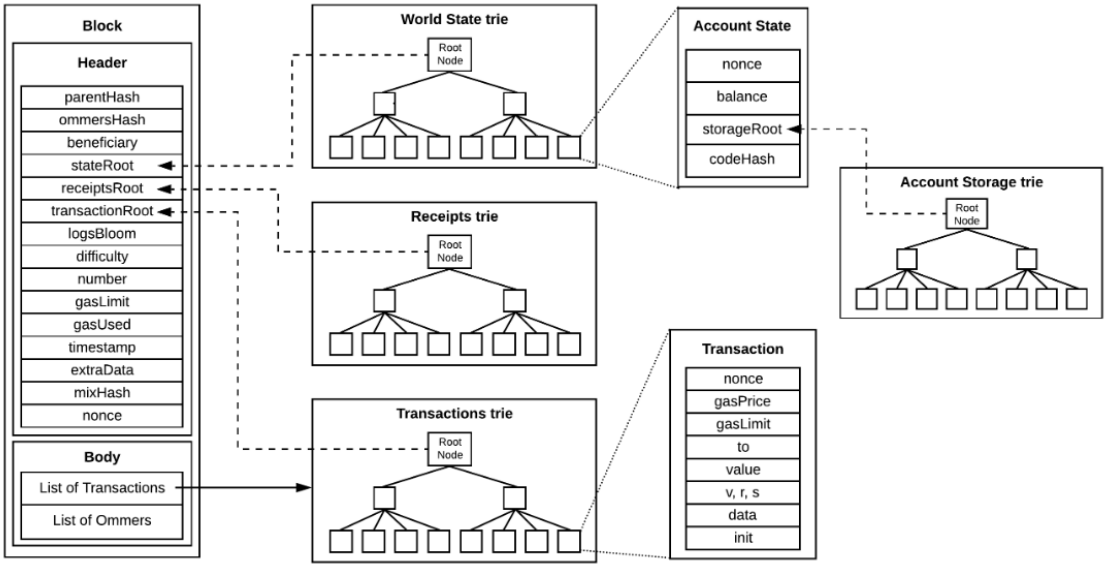
Nonce	Broj poslanih transakcija računa
To	Adresa primatelja
Value	Količina ether-a poslana u transakciji
gasPrice	Cijena za gas u wei-u koje je poslanik voljan platit
gasLimit/startGas	Maksimalna količina gas-a koju je poslanik voljan kupiti za ovu transakciju
Data	Podaci prema kojemu će se pokrenuti određena funkcija u pametnom ugovoru ili *bytecode* novog pametnog ugovora koji se generira
v, r ,s	Polja za ECDSA digitalni potpis

Tablica 2 Struktura transakcije (Izvor: [9])

4.2.5. Blok

Kreiranjem transakcija pomoću novčanika ili pametnih ugovora stvara se objekt transakcije te se sprema u grupu s drugim transakcijama koje nisu sprovedene i čekaju minere kako bi kreirali novi blok, zapisali transakcije u taj blok te povezali trenutni blok s prijašnjim i time izmijenili trenutno stanje blockchain-a.

Svaki blok sadrži zaglavlje, validne transakcije te zadnje stanje nakon što su se transakcije sprovele. Zaglavlje sadrži broj bloka, hash zapis prijašnjeg bloka i informacije za konsenzus mehanizam. Kako bi se moglo u jedan blok zapisati sve transakcije koristi se struktura Merket stabla [10]. Na slici ispod možemo vidjeti strukturu bloka. (Slika 6)



Slika 6 Struktura bloka (Izvor: [2])

5. Ethereum pametni ugovori

Naziv pametni ugovori označava nepromjenjivi kompjuterski kod koji se izvršava kao autonomni agent u kontekstu EVM-a. Ugovor nije nešto što dvije strane moraju zadovoljiti ili ispuniti uvjete već je to prihvaćeni izraz za dio sustava koji pokrene isti dio koda te prema potrebi mijenja interni saldo i ostale varijable kada mu se pošalje poruka. [Prema 7]

Objekt računa pametnih ugovora opisane u poglavlju 4.2.1.1 sadrži kod koji kad se jednom spremi na blockchain nije u mogućnosti promijeniti, sadrži spremljene podatke vezane uz račun i saldo računa kojeg može mijenjati. Kako bi izvršili određenu funkcionalnost pametnog ugovora moramo napraviti transakciju na adresu samog pametnog ugovora iz našeg novčanika ili drugog pametnog ugovora, transakcija mora sadržavati naziv funkcije koja se poziva u ugovoru i parametre koji se šalju samoj funkciji, ako je potrebno šaljemo i ether ako postoji uvjet izvršavanja funkcija ugovora. Isto tako ugovor može slati ether drugim ugovorima te s time pozivati njihove funkcije. Izvršavanje kod-a odvija se na svim čvorovima u mreži pomoću EVM-a, podaci koji se kreiraju prilikom izvršavanja funkcija ugovora se spremaju u internu memoriju svakog računa te se pomoću blokova u blokchainu memorirani zauvijek u strukturi glave bloka, podaci su spremljeni u obliku ključ-vrijednost.

5.1. Izvođenje koda

Kod u Ethereum pametnim ugovorima napisan je u niskom, stog temeljenom bajt kod jeziku tzv. EVM kod. Operacije imaju pristup do tri vrste memorije za spremanje podataka:

- **Stogu** prema principu *last-in-first-out* (kraće LIFO)
- **Kratkoročna memorija** (eng. *memory*) neograničeni prostor za pohranu podataka
- **Memorija ugovora** (eng. *storage*) za dugoročnu pohranu podataka, za razliku od stoga i kratkoročne memorije ova vrsta memorije se neće resetirati nakon izvršavanja pametnog ugovora nego je spremljena u stanju računa.

Kod može pristupiti poljima: vrijednost, pozivatelj, podaci od primljene transakcije i zaglavljima blokova blockchain-a. Kada EVM izvršava transakciju u prvom koraku dodijeljen je gas iz transakcije iz polja limit. Za svaku operaciju koja se izvede smanji se količina gas-a koja je bila dodijeljena kako EVM prolazi kroz kod iz pametnog ugovora. Prije nego što se krenu izvoditi operacije EVM provjeri jeli moguće izvesti, jeli ima dovoljno gas-a za izvedbu, u slučaju da nema dovoljno gas-a za nastavak izvedbi operacija, transakcija se prekida te se podigne zastavica za nestanak gas-a. Ako EVM dođe do kraja koda izvršavanja pametnog ugovora bez da ostane bez gas-a trenutno stanje se mijenja i zapisuje u blockchain. Gas se koristi kako bi platio rudarima za dodijeljene resurse u izvršavanje pametnog ugovora, provizija mineru = cijena gas-a * količina gas-a, ostatak gas-a iz transakcije koji je ostao se vraća pozivatelju kao višak. Kod pametnog ugovora se nikad ne pokreće bez vanjskog poziva. Vrijedi spomenuti da se pametni ugovori ne izvršavaju u paraleli, već svaki ugovor ima svoje vrijeme izvršavanja te čeka ako je potrebno na izvršavanje drugih ugovora.

Transakcije se izvršavaju u potpunosti i mijenjaju globalno stanje jedino ako nema grešaka u procesu te se sve potrebne funkcionalnosti izvrše, ako dođe do greške transakcija

se poništava te se sve izmjene stanja koje su se dogodile također poništavaju, tj. vraćaju na stare vrijednosti. Poništena transakcija se također zapisuje u blockchain te se gas potrošen na transakciju oduzima s glavnog računa. Kako je već spomenuto, ugovori se ne mogu ažurirati već se moraju „obrisat“ pomoću *opcode* funkcije *SELFDESTRUCT* točnije odvojiti kod i interni sadržaj memorije od adrese ostavljajući prazan račun pametnog ugovora. Ako se izvrši transakcija na ovakvu vrstu ugovoru i dalje se može izvesti kod, ali će sav ether poslan u takav ugovor biti izgubljen, stoga je potrebno prije same pohrane ugovora na blockchain implementirati logiku koja će onemogućiti pozivanje funkcija ugovora pomoću zastavica. Prema[7,8]

5.2. Solidty

Ethereum podržava različite programske jezike za pisanje pametnih ugovora. Najpopularniji među njima je Solidity, viši programski jezik sa sintaksom sličnoj JavaScript jeziku. Slično kao i klasa u objektnom programiranju u Solidity-ju se koristi ključna riječ *contract. Contract* može sadržavati polja, funkcije, strukture i nasljeđivati druge ugovore. U daljem dijelu biti će opisani važni dijelovi Solidity-a.

5.2.1. Varijable

Varijable sadrže podatke koje se obrađuju za vrijeme izvršavanja pametnog ugovora te su spremljene u stanju računa ili kao kratkoročna memorija. Prije korištenja varijabli moramo ih inicijalizirati s odgovarajućim tipom podataka, Solidity nam za tu mogućnost pruža niz različiti tipova podataka od kojih su neke opisane u tablici ispod (Tablica 3). Prema[11]

Boolean	Vrijednosti *true* ili *false*, podržava logičke operacije !, &&, ||, ==, !=
Integer	Cijeli brojevi (*int*) ili samo pozitivni (*uint*)
Address	Predstavlja 20 bajtnu Ethereum adresu.
Struct	Korisnički definiran tip podataka
Arrays (fixed/ dynamic)	Polje s podacima istog tipa koje može biti fiksne duljine ili se duljina mijenja tokom izvođenja programa.

Mapping	Hash kolekcija mapiranja u obliku ključ => vrijednost, gdje ključevi moraju biti istog tipa te i sve vrijednosti istog tipa

Tablica 3 Tipovi podataka(Izvor: Prema[11])

5.2.2. Globalne varijable i funkcije

Prilikom izvršavanja pametni ugovor ima pristup jednom dijelu globalnih objekata i funkcija. U daljnjem dijelu će biti opisani neki od objekata i funkcija koje Solidity pruža pristup iz pametnih ugovora.

Kontekst poruke je sadržan u **msg** objektu te sadrži atribute:

- **msg.sender**: predstavlja adresu novčanika koja je potpisala transakciju kod pozivanja ugovora, ako je ugovor pozvan direktno od EOA inače je adresa ugovora.
- **msg.value**: količina ether-a poslana u pozivu funkcije.
- **msg.data:** podaci iz polja *data* iz transakcije koja poziva ugovor.
- **msg.sig:** prva 3 bajta polja dana koja označavaju pozvanu funkciju.

Kontekst transakcije je **tx** objekt koji omogućava pristup informacijama transakciji:

- **tx.gasprice:** cijena gas-a u transakciji.
- **tx.origin:** adresa od EOA koji je kreirao transakciju.

Blok kontekst je sadržaj u **block** objektu te sadrži informacije o trenutnom bloku:

- **block.number:** broj trenutnog bloka.
- **block.timestamp:** vrijeme zapisano u trenutnom bloku.

Address objekt sadrži informacije o adresi ili ether-u koji su proslijeđeni pametnom ugovoru:

- **address.balance**: saldo računa s adresom (address) u wei-u.
- **address.call({value})():** opcode CALL funkcija, kreira poziv funkcije ugovora s proslijeđenim podacima

Neke ugrađene funkcije Solidity-a:

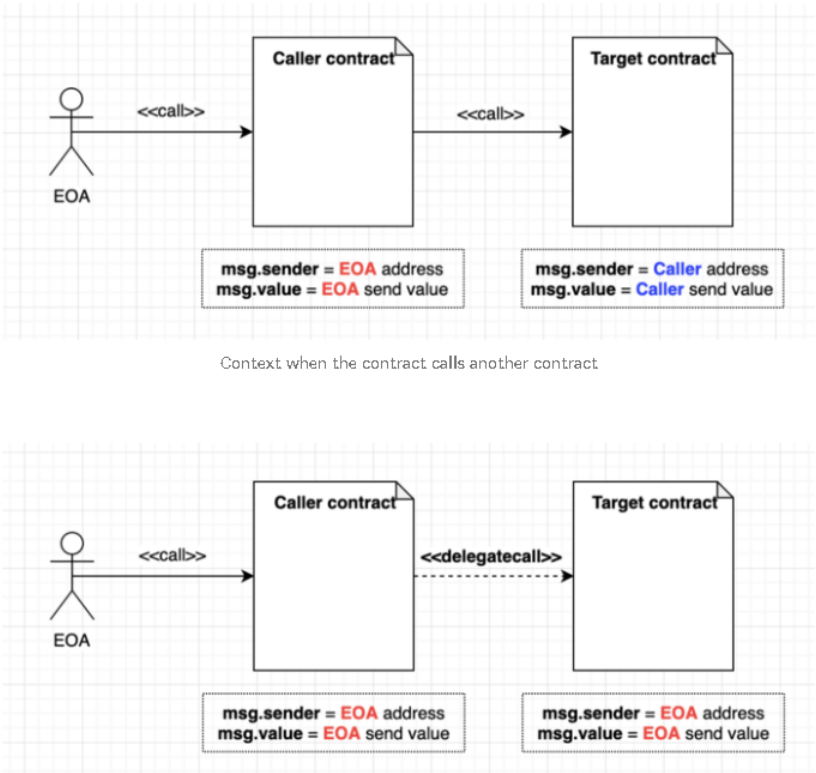
- **keccak256, sha256, sha3, ripemd160**: funkcije za kreiranje hash-a
- **selfdestrunct(*recipient_address*):** funkcija briše trenutni ugovor i ostatak ethera šalje na adresu „*recipient_address*“.

5.2.3. Funkcije

Funkcije su dio pametnog ugovora koje mogu biti pozvane ili delegirane. Zvanje funkcije je poziv pomoću *opcode* CALL, delegirano zvanje je poziv funkcije pomoću opcode DELEGATECALL. Prilikom pozivanja funkcija pametnih ugovora kreira se poruka te sadrži ime i argumente za funkciju koju se poziva, također može sadržavati i određenu količinu ether-a koja će biti dodijeljena saldu računa u kojem se nalazi funkcija i gas-a koji je potreban kao provizija rudarima za dodjelu resurse prilikom izvršavanja. **Delegat poziv** je poziv funkcije kada jedan pametni ugovor poziva funkcije drugog pametnog ugovora, ali kontekst pri izvođenju tih funkcija je od računa pozivatelja te se stanje računa pozivatelja mijenja. Na sljedećoj slici prikazan je delegat poziv (Slika 7).

Pored klasičnih funkcija postoje i funkcije tipa:

- *View*: funkcija ne mijenja stanje računa.
- *Pure*: funkcija obećava da ne čita ili mijenja stanje.



Slika 7 Delegat poziv (Izvor: [13])

5.2.3.1. Modifikator funkcija

Modifikatori funkcija se koriste u slučajevima kada nama je potrebna provjera uvjeta za određene funkcije, kao što je uvjet da pozivatelj funkcije bude korisnik s posebnom adresom ili količina ether-a poslana u pametni ugovor kako bi se funkcija mogla izvesti. Ime

modifiaktora se navodi u potpisu funkcije. Uvedeni su kako bi pisanje koda bilo lakše, odvajanjem logike funkcija od uvjeta koji se moraju zadovoljiti prije samog izvršavanja.

5.2.3.2. Rezervna funkcija

Kada se pametni ugovori pohranjuju na blockchain prvo ih je potrebno svesti na strojni jezik tzv. *bytcode* koji je poznat EVM te kako bi mogao izvoditi funkcije ugovora, svakoj funkciji se kreira potpis prema imenu i argumentima koje prima i prema tom potpisu koji se šalje u poruci EVM zna pronaći funkciju na blockhain te ju pozvati. U slučaju kada je funkcija pozvana, a ne nalazi se u pametnom ugovoru kojemu je upućen poziv, tada se izvodi rezervna (*eng.fallback*) funkcija. Funkcija nema argumenata i ne vraća vrijednosti. U slučaju slanja transakcije s ether-om ugovoru bez sadržaja o izvedbi funkcije EVM će pozvati *fallback* funkciju kako bi odredio daljnji tok izvođenja ukoliko *fallback* funkcija nije definirana poslani ether će biti dodijeljen saldu računa ugovora.

5.2.4. Događaji

Događaji (*eng. Event*) su sučelje koje omogućava korištenje EVM zapisnik zapisa (eng.*log)*. Log se može upotrijebiti za opisivanje događaja unutar pametnog ugovora, poput prijenosa tokena ili promjene vlasništva. Te uz pomoć drugih biblioteka kao što je web3 [14] možemo slušati promjene stanja ugovora bez da smo u direktnoj interakciji sa samim ugovorom. Prema [15]

5.2.5. Vidljivost varijabli i funkcija

Postoji četiri različite vrste vidljivost za varijable i funkcije u Solidity-u:

- **public**: javne funkcije mogu biti pozvane interno od ugovora u kojem se nalaze ili eksterne klase generiranjem poruke.
- **private**: privatne funkcije i varijable su vidljive samo ugovoru u kojem se nalaze. Važno je znati da iako je varijabla ili funkcija privatna to ne znači da je nije moguće dohvatiti s blockchain-a.
- **external**: eksterne funkcije se ponašaju kao javne, razlika je u tome što se eksterne funkcije pozivaju isključivo pomoću ključne riječi „this“.
- **internal**: interne funkcije i varijable se ponašaju kao privatne jedino što se interne ne pozivaju preko ključne riječi „this“.

6. Primjena pametnih ugovora

Neke implementacije pametnih ugovora na Ethereum sustavu:

Decentralizirane financije (*eng.* Decentralized Finance, *kraće **DEFI***)**.** Mehanizmi tradicionalnih financija prenose se na blockchain radi njegove decentralizacije, otpornosti na cenzuru kojeg može bilo tko koristiti u svijetu bez obzira na stanje ili lokaciju na kojoj živi. Neki od mehanizama:

- Sintetički vrijednosni papiri
- Decentralizirane mjenjačnice
- Posudba valute
- Štednje
- Derivati

Glasovanje. Prilikom tradicionalnog demokratskog glasovanja i dalje je problem centraliziranog načina obrade glasova kod koje se treba vjerovati trećoj stranci u ispravnosti brojenja glasova kako nitko drugi nije u mogućnosti to provjeriti. Stoga je blockchain jedan od najboljih rješenja zbog svoje transparentnosti.

Igre na sreću. Igre na sreću mogu biti napravljene u korist kuće koja ih kreira bez da je sreća u pitanju te tako prevari svoje klijente. Pametnim ugovorima se postiže kreiranje igara čija logika može biti vidljiva svima te time postiže transparentnost sustava, igre u slučaju dobitka za korisnika moraju biti isplaćene jer se nagrade dodjeljuju pomoću pametnih ugovora za zadovoljene uvjete igre.

Lanac opskrbe. Lanac opskrbe je tok proizvodnje dobara počevši od sirovine do gotovog proizvoda. Pomoću pametnih ugovora i *Internet of Things (*kraće*.IoT)*[5] senzora može se kreirati zapis kako se proizvod kretao kroz opskrbni lanac i gdje se nalazi u određenom trenutku. Za gotovi proizvod s polica kupac može potvrditi od kuda je došao te isto tako sam dobavljač može vidjeti gdje se određeni proizvod nalazi i jeli došlo do zastoja.

Digitalni identitet. Danas različite institucije drže različite podatke o korisniku, a kako bi se svi ti podatci sakupili u jednu datoteku, trebali bi nositi ogromnu hrpu papira, referenci i primjeraka. Prilično je nezgodno, posebno ako se mora proći provjera identiteta. Pametni ugovori rješavaju ovaj problem i omogućavaju da se svi podaci o jednoj osobi drže na jednom mjestu. Kad god bi vam se nešto dogodilo, registrirali biste promjene na blockchainu kako bi vaš identitet bio cjelovit. Zbog toga bi potvrda o korisniku (eng. *Know your customer,* kraće

[5] IoT- sustav međusobno povezanih računalnih uređaja, mehaničkih i digitalnih strojeva, predmeta ili ljudi s ciljem prijenosa podataka putem mreže bez ikakve interakcije.

KYC*)* postala trenutna. I na vašu privatnost neće utjecati jer vi odlučujete koje ćete podatke otkriti. [Prema 12]

Valutni sustav. Valutni sustav kreiran na Ethereum sustavu može biti upotrijebljen kako bi se kreirala valuta koja predstavlja entitet iz određene domene kao što su dionice ili zlato te sam dolar. Zasebni tokeni koji predstavljaju kupone ili tokene za događaje koji se ne vežu za postojeće fiat valute nego su zaseban podsustav te imaju svoju intrinzičnu vrijednost ili služe kao poticaj za korištenje tog sustava**.**

Decentralizirane autonomne organizacije (eng.Decentralized Autonomous Organizations, kraće DAO)**.** Opći pojam "decentralizirane autonomne organizacije" je virtualni entitet. Članovi bi zajedno odlučivali o tome kako organizacija treba rasporediti svoja sredstva. Metode raspodjele sredstava DAO-a mogu se kretati u rasponu od plaće, do još egzotičnijih mehanizama poput interne valute za nagrađivanje rada. To u osnovi ponavlja pravne karakteristike tradicionalne tvrtke ili neprofitne organizacije, ali koristeći samo kriptografsku tehnologiju blockchain-a za provođenje zakona. Dosad su se mnogi razgovori oko DAO-a odvijali oko "kapitalističkog" modela "decentralizirane autonomne korporacije" (DAC) s dioničarima koji primaju dividendu i trgujućim dionicama; alternativa, možda opisana kao "decentralizirana autonomna zajednica", imala bi za sve članove jednak udio u odlučivanju i zahtijeva 67% postojećih članova da pristanu na dodavanje ili uklanjanje člana. Uvjet da jedna osoba može imati samo jedno članstvo tada je potrebno kolektivno provoditi u zajednici. Prema [7]

6.1. Implementacija digitalnog glasovanja

U ovom poglavlju opisuje se primjena pametnih ugovora za digitalno glasovanje te sama implementacija glasovanja u Solidity programskom jeziku. Cijeli izvorni kod pametnog ugovora nalazi se u prilogu 1.

6.1.1. Slučaji upotrebe

Opisani su sljedeći slučajevi upotrebe:

- Postavljanje ugovora na blockchain testnu mrežu
- Kreiranje prvog prijedloga s opcijama
- Dodavanje glasača koji imaju prava glasa
- Početka glasovanja
- Glasovanje
- Završetak glasovanja i odabir pobjednika

6.1.1.1. Postavljanje ugovora

Prvi korak nakon uspješno napisanog koda za pametni ugovor i testiranih svih funkcija pomoću kompajlera kao što je Remix, je postavljanje samog ugovora na blockchain. Kako postoji mogućnost za greškama uvijek je bolje prvo postaviti ugovor na testnu mrežu u našem slučaju Ropsten, a ne direktno na samu glavnu mrežu Ethereuma. Pomoću Remixa kopiramo *bytecode* od pametnog ugovora u izborniku *Run & Deploy* gdje odabiremo *enviroment Injected Web3* koje označava korištenje MetMask i Ropsten *testneta*. Kako bi poslali *bytecode* na *testnet* odabiremo *Deploy* opciju (Slika 8) i u polja postavljamo parametre koji će se poslati ugovoru nakon njegovog kreiranja. Početni podaci su:

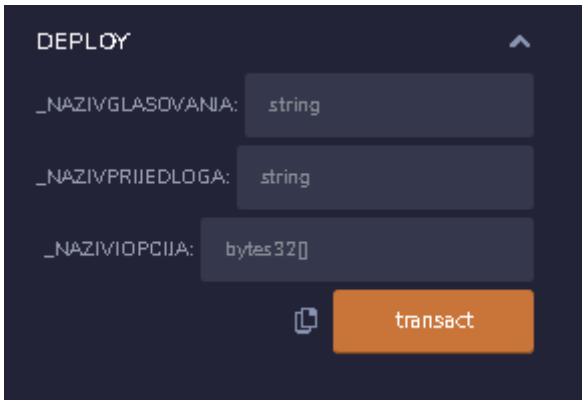
- Naziv glasovanja: tip podatka je tekstualni
- Naziv prijedloga: tip podatka je tekstualni
- Opcije prijedloga: tip podatka je polje bytes32 koje predstavlja tekstualni tip pretvoren u strojne riječ veličine bytes32

Nakon kreiranja transakcije (Slika 9) te postavljanja ugovora na blockchain, poslat će se vrijednosti iz transakcije u konstruktor ugovora i kreirat će se prvi prijedlog sa svojim opcijama za glasovanje pozivom funkcije „kreirajPrijedloge“ (Slika 10), kod konstruktora vidljivo u izvornom kodu:

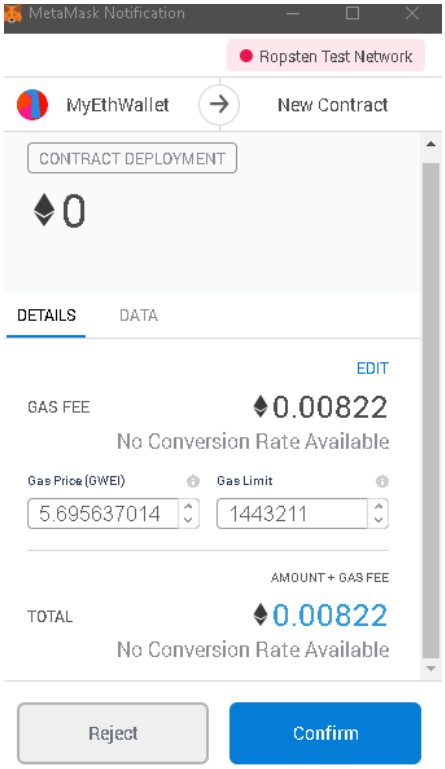
```
contract Glasovanje {
      address public vlasnikGlasovanja;
      string public nazivGlasovanja;

      constructor(
            string memory _nazivGlasovanja,
            string memory _nazivPrijedloga,
            bytes32[] memory _naziviOpcija
      ) public {
             vlasnikGlasovanja = msg.sender;
             nazivGlasovanja = _nazivGlasovanja;
             kreirajPrijedlog(_nazivPrijedloga, _naziviOpcija);
      }
```

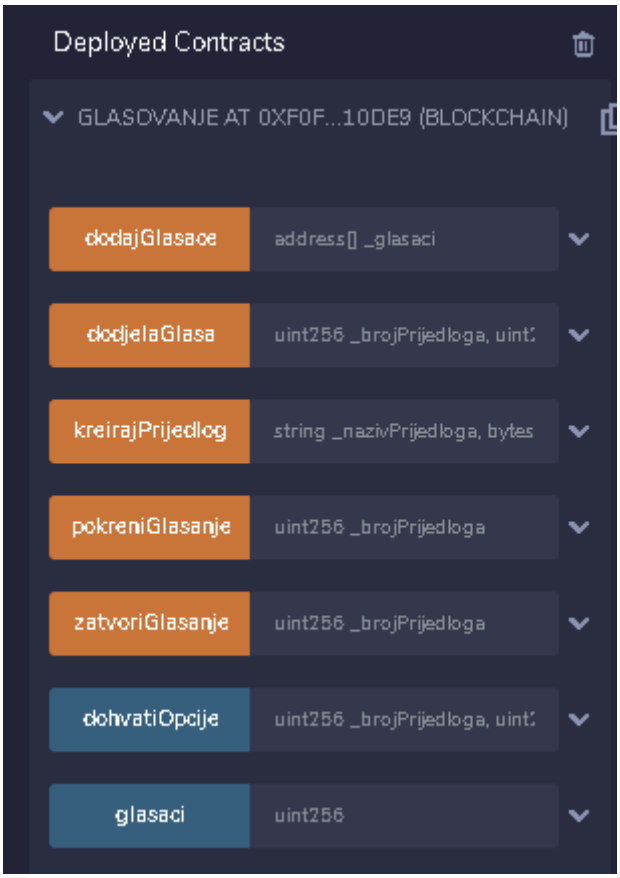
Trošak izvođenja ove operacije je 1443211 wei (0.00822 ETH) te u trenutku pisanja rada cijena troška u Američkim dolarima je otprilike iznosila 61 $.



Slika 8 *Deploy* opcija u Remix-u (Izvor: vlastita izrada)



Slika 9 Transakcija zapisa ugovor na *blockchain* (Izvor: vlastita izrada)



Slika 10 Funkcije pametnog ugovora (Izvor: vlastita izrada)

6.1.1.2. Kreiranje novog prijedloga

Vlasnik ugovora ima opciju kreiranja novih prijedloga s opcijama za glasovanje. Prilikom poziva funkcije „kreirajPrijedlog“ provjerava se adresa računa koji šalje zahtjev za pokretanje te se u memoriji kreiraju nove opcije i prijedlog.

Trošak izvođenja ove operacije je 235699 wei (0.0294 ETH) te u trenutku pisanja rada cijena troška u Američkim dolarima je otprilike iznosila 10$.

```
struct Opcija{
   uint id;
   bytes32 naziv;
   uint brojGlasova;
}
uint brojOpcijaPoPrijedlogu;

struct Prijedlog {
      uint id;
      string naziv;
```

```
        Stanje stanje;
        mapping(uint => Opcija) opcije;
        mapping(address => Glas) glasovi;
        uint idPobjednika;
        uint brojOpcija;
}
mapping(uint => Prijedlog) public prijedlozi;
uint brojPrijedloga;

function kreirajPrijedlog(
        string memory _nazivPrijedloga,
        bytes32[] memory _naziviOpcija
) public provjeraVlasnika{
        Prijedlog memory p = Prijedlog({
                id : brojPrijedloga++,
                naziv : _nazivPrijedloga,
                stanje : Stanje.Kreiran,
                idPobjednika : 0,
                brojOpcija : 0
        });
        brojOpcijaPoPrijedlogu = 0;
        for(uint i = 0; i < _naziviOpcija.length; i++){
                Opcija memory o = Opcija({
                        id : ++brojOpcijaPoPrijedlogu,
                        naziv : _naziviOpcija[i],
                        brojGlasova : 0
                });
                prijedlozi[brojPrijedloga].opcije[brojOpcijaPoPrijedlogu] = o;
        }
        p.brojOpcija = brojOpcijaPoPrijedlogu;
        prijedlozi[brojPrijedloga] = p;}
```

6.1.1.3. Dodjela glasača

Račun koji postavi ugovor na blockchain je ujedno vlasnik tog ugovora i jedino on može kreirati nove prijedloge i dodavati osobe koje imaju prava glasanja. Kako bi onemogućili pristup drugim sudionicima mreže u potpis funkcije „dodajGlasace“ stavljamo testiranje uvjeta pomoću modifikatora funkcije „provjeraVlasnika“. Funkcija „dodajGlasace“ dobiva polje adresa računa koji imaju pravo glasati na kreiranim prijedlozima te se te adrese spremaju u varijablu, glasaći koja je tip podataka hash tablica.

Trošak izvođenja ove operacije za 5 glasaća je 130780 *wei* (0.01674 ETH) te u trenutku pisanja rada cijena troška u Američkim dolarima je otprilike iznosila 6 $.

```
modifier provjeraVlasnika(){
      require(msg.sender == vlasnikGlasovanja);
      _;
}

mapping(uint => address) public glasaci;

event glasaciDodani();
function dodajGlasace(address[] memory _glasaci) public provjeraVlasnika{
      for(uint i = 0; i < _glasaci.length; i++){
            glasaci[i] = _glasaci[i];
      }
      emit glasaciDodani();
}
```

6.1.1.4. Početak glasovanja

Za početak glasovanja kreator ugovora mora pozvati funkciju „pokreniGlasovanje“ koja prima parametar „_brojPrijedloga“ za prijedlog koji će se omogućiti glasovanje. Prije nego što se omogući pokretanje glasovanja prijedlog mora biti u stanju „Kreiran“ tj. prijedlog mora biti kreiran i spremljen u memoriju stanja ugovora.

Trošak izvođenja ove operacije je 45367 wei (0.005807 ETH) te u trenutku pisanja rada cijena troška u Američkim dolarima je otprilike iznosila 2 $.

```
enum Stanje{ Kreiran, Pocetak, Zavrsetak }

modifier provjeraStanja(uint idPrijedolga, Stanje _stanje){
      require(prijedlozi[idPrijedolga].stanje == _stanje);
      _;
}

event glasovanjePokrenuto(uint idPrijedlog);
function pokreniGlasanje(
      uint _brojPrijedloga
) public provjeraStanja(_brojPrijedloga, Stanje.Kreiran) provjeraVlasnika{
      prijedlozi[_brojPrijedloga].stanje = Stanje.Pocetak;
      emit glasovanjePokrenuto(_brojPrijedloga);
}
```

6.1.1.5. Dodjela glasa opciji

Kako bi korisnik mogao dodijeliti svoj glas nekoj opciji iz prijedloga, mora pozvati funkciju „dodjelaGlasa“ točnije mora kreirati transakciju koja poziva navedenu funkciju. Funkcija provjerava je li adresa računa koji poziva funkciju zapisana u varijabli glasovi prijedloga, točnije je li korisnik već zabilježen da je dao svoj glas nekoj opciji te u slučaju da je, prekida se izvođenje i sve promjene stanja se vraćaju na staro.

Trošak izvođenja ove operacije je 86922 wei (0.00049 ETH) te u trenutku pisanja rada cijena troška u Američkim dolarima je otprilike iznosila 3 $.

```
struct Glas {
      address adresaGlasaca;
      uint idOpcije;
}
event korisnikGlasao(address voter);
function dodjelaGlasa(
      uint _brojPrijedloga,
      uint _brojOpcija
) public provjeraStanja(_brojPrijedloga, Stanje.Pocetak){
      if(_brojOpcija == 0)
            revert("Opcija 0 se ne koristi");
      if(prijedlozi[_brojPrijedloga].glasovi[msg.sender].idOpcije == 0){
            prijedlozi[_brojPrijedloga].glasovi[msg.sender] = Glas({
                  adresaGlasaca : msg.sender,
                  idOpcije : _brojOpcija
            });
            prijedlozi[_brojPrijedloga].opcije[_brojOpcija].brojGlasova++;

            emit korisnikGlasao(msg.sender);
      }else{
            revert("Krosnik je vec glasao");
      }
    }
```

6.1.1.6. Završetak glasovanja

Kako bi odredili pobjednika u određenom prijedlogu vlasnik ugovora mora pozvati funkciju „zatvoriGlasovanje“ i kao argument poslati broj prijedloga kojeg želi zatvoriti. Funkcija prolazi kroz sve opcije prijedloga, ali traži opciju koja ima najveću vrijednost u varijabli „brojGlasova“ u slučaju da su dvije opcije s istim brojem glasova, opcije u varijablama „max“ i „drugi“, prekida se izvođenje i šalje poruka da nema pobjednika.

Trošak izvođenja ove operacije je 61322 wei (0.008033 ETH) te u trenutku pisanja rada cijena troška u Američkim dolarima je otprilike iznosila 3 $.

```
event glasovanjeZatvoreno(uint idPrijedlog, uint pobjednik);
function zatvoriGlasanje(
      uint _brojPrijedloga
) public provjeraStanja(_brojPrijedloga, Stanje.Pocetak) provjeraVlasnika{
      prijedlozi[_brojPrijedloga].stanje = Stanje.Zavrsetak;
      Opcija memory max;
      for(uint i = 1; i <= prijedlozi[_brojPrijedloga].brojOpcija; i++){

            if(prijedlozi[_brojPrijedloga].opcije[i].brojGlasova
            > max.brojGlasova){
                  max = prijedlozi[_brojPrijedloga].opcije[i];
            }
      }
      prijedlozi[_brojPrijedloga].idPobjednika = max.id;
      emit glasovanjeZatvoreno(
            _brojPrijedloga,
            prijedlozi[_brojPrijedloga].idPobjednika
      );
}
```

6.1.1.7. Dohvaćanje naziva pobjednika

Funkcija „pobjednik“ za poslani argument „_brojPrijedloga“ vraća naziv opcije koja ima najveći broj glasova. Naziv je tip podataka bytes32 koji se treba pretvoriti u *string* kako bi bio čitljiv ljudima. Kako je funkcija tipa *view* poziv takve funkcije neće biti naplaćen jer funkcija čita vrijednosti iz stanja.

```
function pobjednik(
      uint _brojPrijedloga
) public view provjeraStanja(
      _brojPrijedloga, Stanje.Zavrsetak
) returns(bytes32 nazivOpcije){
        return
      prijedlozi[_brojPrijedloga].opcije[prijedlozi[_brojPrijedloga].idPobj
      ednika].naziv;
}
```

6.1.1.8. Dohvaćanje podataka opcije

„dohvatiOpciju“ pomoćna funkcija tipa *view* pomoću koje možemo zatražiti detalje o određenoj opciji prijedloga. Funkcija je korisna kod kreiranja korisničkog sučelja kako bi ispisali podatke o opciji prilikom same dodjele glasa od strane korisnika.

```
function dohvatiOpciju(
      uint _brojPrijedloga,
      uint _brojOpcije
) public view returns(uint id, uint brojGlasova, bytes32 nazivOpcije){
      Opcija memory opcija;
      opcija = prijedlozi[_brojPrijedloga].opcije[_brojOpcije];
      return (opcija.id, opcija.brojGlasova, opcija.naziv);
}
```

7. Zaključak

Pojavom tehnologije blockchain otvorile su se brojne opcije poboljšanja dosadašnjih financijskih mehanizama pa tako i kreiranje novih. Ethereum je omogućio svima jednostavno kreiranje takvih mehanizama u obliku pametnih ugovora. Sa svojom otvorenosti i decentraliziranosti te najvećom zajednicom programera nameće se kao najbolja mreža za kreiranje decentraliziranih aplikacija kao što je digitalno glasovanje. Baš zbog svoje otvorenosti uvijek treba biti pažljiv prilikom komunikacije s pametnim ugovorima jer ne samo što bilo tko može kreirati ugovor s logičkim pogreškama nego i samih zlonamjernih pametnih ugovora koji pokušavaju prevariti svoje korisnike i uzet im novac.

Popis literature

[1] „*Distributed ledger*“ (2020) u Wikipedia*, the Free Encyclopedia*. Dostupno: https://en.wikipedia.org/wiki/Distributed_ledger [pristupano 25.07.2020]

[2] Datadriveninvestor (2020). *What are the different types of DLT how they wor . Dostupno:* https://www.datadriveninvestor.com/2019/02/14/what-are-the-different-types-of-dlts-how-they-work/# [pristupano 26.07.20]

[3] „*Consensus mechanisa cryptocurrency.“* (2020) Investopedia. Dostupno: https://www.investopedia.com/terms/c/consensus-mechanism-cryptocurrency.asp [pristupano 30.7.2020]

[4] Sathosi Nakamoto. „*Bitcoin: A Peer-To-Peer Electronic Cash System*“, 2020. [Na internetu]. Dostupno: https://bitcoin.org/bitcoin.pdf [pristupano 01.07.2020]

[5] „*Cypherpunk*“ (2020) u Wikipedia*, the Free Encyclopedia*. Dostupno: https://en.wikipedia.org/wiki/Cypherpunk [pristupano 01.08.2020.]

[6] „*History of bitcoin*“ (2020) u Wikipedia*, the Free Encyclopedia*. Dostupno: https://en.wikipedia.org/wiki/History_of_bitcoin [pristupano 01.08.2020.]

[7] Vitalki Buterin. „*Ethereum whitepaper*“, 2020. [Na internetu] Dostupno: https://ethereum.org/en/whitepaper/ [pristupano 01.07.2020]

[8] Andreas M. Antonopoulos, Gavin Wood. *Mastering Ethereum*, O'Reilly Media, Inc. 2018

[9] Gavin Wood. „*Ethereum: A secure decentralised generalised transaction ledger*.“, 2020. [Na internetu] Dostupno: https://ethereum.github.io/yellowpaper/paper.pdf [pristupano 15.08.2020]

[10] „*Modified Merkel patricia trie how ethereum saves a state*“ (2020). Autor [Na internetu] Dostupno: https://medium.com/codechain/modified-merkle-patricia-trie-how-ethereum-saves-a-state-e6d7555078dd [pristupano 25.08.2020]

[11] https://solidity.readthedocs.io/en/v0.7.0/types.html#types [Pristupano 20.08.2020]

[12] „Smart contrats 10 use cases bussines“, 2020. [Na internetu]. Dostupno: https://ambisafe.com/blog/smart-contracts-10-use-cases-business/ [Pristupano 20.08.2020]

[13] „Delegatecall calling another contract in solidity“, 2020. [Na internetu]. Dostupno: https://medium.com/coinmonks/delegatecall-calling-another-contract-function-in-solidity-b579f804178c [Pristupano 20.08.2020]

[14] Ethereum. Web3.js (2020) [Na internetu]. Dostupno: https://web3js.readthedocs.io/en/v1.2.11/ [Pristupano 20.08.2020]

[15] „Understanding event logs on the ethereum blockchain“, 2020. [Na internetu]. Dostupno: https://medium.com/mycrypto/understanding-event-logs-on-the-ethereum-blockchain-f4ae7ba50378 [Pristupano 20.08.2020]

Popis slika

Popis tablica

Prilozi 1

Glasovanje.sol