DECENTRALIZIRANA APLIKACIJA ZA AUKCIJE NEZAMJENJIVIH TOKENA NA EUTHEREUM MREŽI

Ivančić, Bruno

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

Decentralizirana aplikacija za aukcije nezamjenjivih tokena na Ethereum mreži

Bruno Ivančić

Zagreb, veljača 2020.

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

U Zagrebu, datum.

Predgovor

Ovim putem zahvaljujem se svom mentoru i predavaču na Visokom učilištu Algebra Zlatanu Moriću na vrijednim savjetima i korisnim prijedlozima koji su obogatili konačnu formu ovog rada kao i na njegovom interesu te suradnji u pisanju samog rada.

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

Sažetak

Cilj ovog rada je otkriti i proširiti moje znanje o distribuiranim mrežama i teoriji aukcija izradom koncept aplikacije za aukcije na Ethereum mreži koristeći pametne ugovore. Nezamjenjivi tokeni koriste se za stvaranje digitalne povjerljivosti, a prvi put bili su implementirani koristeći Ethereum ERC721 token standard. NFT-Auction-dapp je aplikacija bazirana na Ethereumu koja omogućuje korisnicima da kreiraju i sudjeluju u aukcijama NFT tokena u stvarnom vremenu, na globalnoj i distribuiranoj mreži, zadržavajući garancije plaćanja i dostave. U radu ću se služiti znanjima koja sam stekao tijekom studija na Visokom učilištu Algebra i znanjima koja sam stekao samostalno kako bih izgradio aplikaciju, objavio ju na javnu Ethereum mrežu i testirao njezinu funkcionalnost.

Abstract

The goal of this thesis is to discover and expand my knowledge about distributed networks and auction theory by creating a proof of concept application for auctions on the Ethereum network using smart contracts. Non-fungible tokens are used to create verifiable digital scarcity and were first implemented using the Ethereum ERC721 token standard. NFT-Auction-dapp is an Ethereum based application that allows users to auction off and participate in NFT token auctions in real time on a global and distributed network, while maintaining payment and delivery guarantees. In this thesis I will use knowledge attained during my studies at Algebra University Collage as well as knowledge attained on my own to develop the application, publish it on a public Ethereum network and test it.

Sadržaj

1. Uvod

Od svoje koncepcije, ERC721 nezamjenjivi tokeni pridobili su prilično veliku količinu interesa od Ethereum zajednice zbog činjenice da mogu imati velik broj potencijalnih svrha, kao što su kolekcionarski predmeti (karte, avatari), i imovina (digitalne nekretnine, umjetnička djela). Ova vrsta tokena omogućava interoperabilnost tokena unutar Ethereum ekosustava kao i novi način kreiranja dokažljive nestašice digitalne imovine. Zbog interoperabilnosti sučelja ERC721 standarda, korisnici tokena mogu koristiti bilo koju od brojnih mjenjačnica na Ethereumu.

Decentralizirane mjenjačnice na Ethereumu omogućuju korisnicima *peer-to-peer* razmjenu kriptovaluta bez potrebe za centralnim autoritetom. Pošto su NFT tokeni međusobno nezamjenjivi, nije moguće koristiti iste mehanizme kao za razmjenu ostalih tokena. Zbog ove činjenice najlogičnija implementacija tržišta za ovu vrstu tokena je platforma za aukcije. Iako ERC20 tokeni još uvijek drže velik dio tržišta tokena, ERC721 tokeni ostvarili su veći rast u broju novih ugovora objavljenih na mreži u 2019. godini.

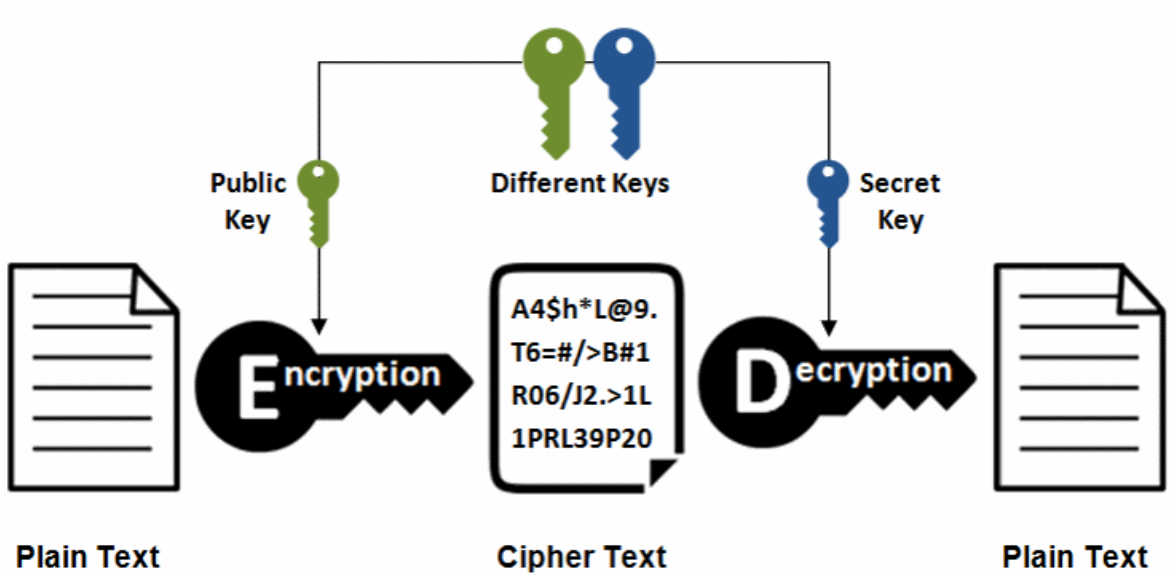
2. Osnove DLT tehnologije

DLT (*Distributed Ledger Technology*) tehnologija definirana je kao geografski distribuirana *peer-to-peer* mreža koja održava konsenzus o nekom sinkroniziranom setu digitalnih podataka. DLT mreže ne zahtijevaju centralnu administraciju podataka. *Ledger* ili glavna knjiga odnosi se na repliciranu bazu podataka podijeljenu između više sudionika mreže u kojoj svatko drži zasebnu kopiju te na temelju unosa mijenja stanje baze. Da bi promjena stanja bila uspješna, sudionici mreže moraju doći do konsenzusa oko novog stanja glavne knjige. DLT mreže dijele se na javne i privatne. Ovo određuje tko može vidjeti stanje zapisa glavne knjige i sudjelovati u procesu ostvarivanja konsenzusa. Javne mreže su generalno otvorene svima i bilo tko ih može koristiti i sudjelovati u ostvarivanju konsenzusa. Za razliku od javnih, privatne mreže su one u kojima samo određene stranke mogu sudjelovati.

2.1. Kriptografski principi i funkcije

2.1.1. Asimetrična kriptografija i digitalni potpisi

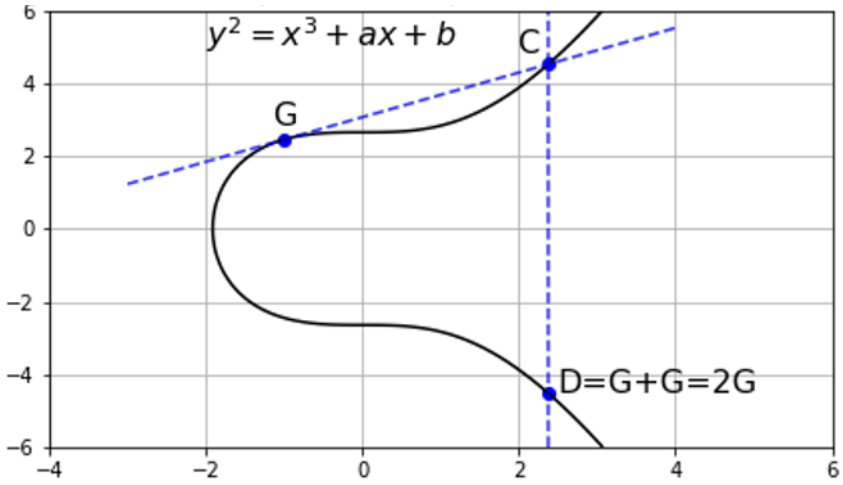
Asimetrična ili *public-key* kriptografija jedna je od osnovnih komponenta DLT tehnologije. Za razliku od simetrične kriptografije koja koristi samo jedan ključ za inkriptiranje i dekriptiranje podataka, asimetrična kriptografija nalaže korištenje para ključeva koji se sastoje od javnog (*public*) i privatnog (*private*) ključa. *Public* ključ predstavlja jedinstvenu javnu adresu na *block chainu*, dok *private* ključ služi za autoriziranje transakcija s te adrese. Asimetrična kriptografija zadovoljava uvjete izračunske neizvedivosti deriviranja privatnog ključa iz danog javnog ključa i mogućnost dokazivanja vlasništva odgovarajućeg privatnog ključa kroz potpis i bez odavanja informacija o ključu.



Slika 2-1 Asimetrična kriptografija

RSA je prvi i najčešće korišteni *public-key* kriptografski algoritam. U slučaju RSA enkripcije, algoritam sadrži takozvane *trapdoor* funkcije ili funkcije koje omogućuju lagano izvršavanje u jednom smjeru i teško izvršavanje u drugom smjeru. Sigurnost ovakvih algoritma oslanja se na činjenicu da enkripcija javnim ključem može biti poništena samo koristeći odgovarajući privatni ključ. Bitcoin i Ethereum kriptovalute koriste ECDSA (*Elliptic Curve Digital Signature Algorithm*) metodu za upravljanje digitalnim potpisima. ECDSA je nastao kao alternativa RSA algoritmu jer može pružiti jednaku razinu sigurnosti koristeći kraće ključeve, što omogućuje brže izračune.

Elliptic-curve kriptografija je pristup asimetričnoj kriptografiji koristeći strukturu eliptične krivulje simetrične oko x osi. Eliptična krivulja korištena za Bitcoin i Ethereum je secp256k1 krivulja dobivena jednadžbom $y^2 = x^3+7$. Generacija ključeva sastoji se od generiranja 256 bitne vrijednosti x koja predstavlja privatni ključ, te izračunavajući javnog ključa koristeći X= x*P formulu u kojoj P predstavlja predefiniranu točku na krivulji. Javni ključ X predstavljen je točkom na krivulji. Operacija multipliciranja točke na eliptičnoj krivulji jednosmjerna je operacija koja uključuje dodavanje vrijednosti točke na eliptičnoj krivulji više puta. Ova funkcija je jednosmjerna funkcija što znači da ju je lako izvesti u jednom smjeru, ali teže u obrnutom smjeru.



Slika 2-2 Multipliciranje točke na eliptičnoj krivulji

Digitalni potpisi su kriptografski potpisi koji služe za provjeru autentičnosti i integriteta digitalnih podataka. Generirani koristeći privatni ključ, digitalni potpisi omogućuju kontrolu pristupa sredstvima na *block chain* adresi. Potpisi su generirani od *hashirane* vrijednosti poruke i privatnog ključa. Matematički algoritam može biti korišten na potpisu da bi se otkrio da je originalno stvoren koristeći odgovarajući *hash* i privatni ključ, bez potrebe za otkrivanjem privatnog ključa.

2.1.2. Arhitektura mreže

Glavna razlika između javnih i privatnih DLT mreža je u razinama pristupa omogućenih sudionicima mreže. Javne *block chain* mreže otvorene su svim sudionicima te oni mogu sudjelovati potvrđujući blokove na mreži. Neke od najpoznatijih javnih *block chain* mreža su Bitcoin i Ethereum čiji kod je javno dostupan. Za razliku od javnih mreža, u privatnim mrežama samo autorizirane stranke mogu sudjelovati. Zbog manjeg broja validatora blokova, privatne *block chain* mreže često puno brže ostvaruju konsenzus što im omogućuje da ostvare veći broj transakcija po sekundi i veću razinu skalabilnosti. Kod *block chain* mreža s privolama, najčešće više organizacija osnuje konzorcijum za formiranje mreže. Pravila i dozvole dogovorene su međusobno tijekom stvaranja mreže.

2.1.2.1 **Node**

Node uređaji sadrže specijaliziran softver koji omogućuje računalu da komunicira s *block chain* mrežom. Neki od popularnih *node* softvera za Ethereum su *Geth* i *Pairity*. Neke implementacije *node* softvera omogućuju tim računalima da djeluju kao način komunikacije

s programima. Ovo je najčešće ostvareno koristeći RPC (*remote procedure call*) pozive. Generalno, softvere možemo podijeliti na 2 kategorije:

- *Full node* – implementacija *node* softvera u kojoj program verificira transakcije u *ledgeru* i objavljuje ih na mrežu. *Full node* sadrži cijelu kopiju *block chaina*.
- *Light node* – ne održavaju cijelu kopiju *block chaina* i ne verificiraju transakcije. *Light node* implementacije lakše je održavati jer zahtijevaju znatno manje komputacijskih resursa.

Ethereum-*as-a-service* usluge poput *Infura* daju developerima opciju spajanja aplikacija na Ethereum bez potrebe za konfiguriranjem vlastitog *nodea*. Ovo je ostvareno spajanjem na JSON-RPC *block chain endpoint* kroz dane API ključeve.

2.1.2.2 ***Block time* i *difficulty***

Block time definira vrijeme potrebno da bi se dodao blok na *block chain*. Bitcoin i Ethereum imaju definirano očekivano i prosječno vrijeme bloka. Pošto oba *block chaina* trenutno koriste PoW metodu ostvarivanja konsenzusa u mreži, očekivano vrijeme bloka postavljeno je kao konstanta te onemogućuje manipuliranje vremenom u *block chainu* s povećanjem ili smanjenjem količine PoW izračuna u mreži. Prosječno vrijeme generiranja bloka na Bitcoinu trenutno je 10 minuta, dok je na Ethereumu 15 sekundi. *Difficulty* metrika diktira brzinu kojom se dodaju novi blokovi. Ova metrika opisuje količinu PoW izračuna potrebnu da bi se pronašao *hash* odgovarajućeg bloka. U Bitcoin *block chainu difficulty* metrika promijenjena je svakih 2016 blokova ili 2 tjedna. Proporcionalno s vremenom devijacije pronalaska 2016 blokova i predefiniranih 2 tjedna, *difficulty* metrika se mijenja. U Ethereumu očekivan *block time* definiran je između 10 i 19 sekunda. Za razliku od Bitcoina, Ethereum sadrži koncept *difficulty* bombe kako bi developeri bili incentivizirani da naprave prelazak na PoS konsenzus mehanizam. *Difficulty* bomba odnosi se na povećanje *difficulty* metrike svakih 100 000 blokova što eksponencijalno povećava vrijeme otkrivanja novih blokova.

2.2. Ledger i transakcije

2.2.1. Metode konsenzusa

U tradicionalnim mrežama za razmjenu vrijednosti i informacija obično postoji posrednik koji osigurava mrežu i potvrđuje transakcije. Na primjer, banke jamče sigurnost imovine svojih klijenata te vrše centraliziranu kontrolu nad njihovim transakcijama i informacijama. Kako bi mogle eliminirati potrebu za posrednicima, *block chain* mreže moraju koristiti nekakav sustav za održavanje sigurnosti i valjanosti svih novih podataka na mreži.

Konsenzus mehanizam ključna je komponenta *block chaina* koja pruža način ostvarivanja dogovora oko trenutnog stanja mreže. Neki od *block chain* konsenzus mehanizma su PoW baziran na ASIC uređajima, PoW baziran na GPU izračunima, PoS i delegirani PoS. Svaki od ovih konsenzus mehanizma dolazi sa svojom filozofijom. Jedan od najpoznatijih konsenzus mehanizama je PoW korišten u Bitcoin mreži, te se bazira na filozofiji da je *block chain* s najvećom količinom kapitala uloženom u stvaranje komputacijske snage ispravan.

Byzantine Fault tolerance koncept odnosi se na problem u distribuiranim sustavima kada komponente sustava mogu zakazati te nije moguće imati potpunu informaciju o zakazanim komponentama. Ime je proizvedeno iz alegorije problema bizantskih generala koja opisuje situaciju u kojoj sudionici sustava moraju donijeti odluku o zajedničkoj strategiji kada su neki od sudionika nepouzdani. Greška u sustavu može se manifestirati s različitim simptomima različitim promatračima što otežava proces otkrivanja greške. BFT koncept u smislu distribuiranih sustava opisuje pouzdanost mreže izložene na ovakve uvjete.

2.2.1.1 ***Proof of Work*** **(PoW)**

Pojam „Dokaz posla“ *(Proof of Work)* bio je formaliziran 1999. godine u radu *Proofs of Work and Bread Pudding Protocols* od strane računalnih znanstvenika Ari Juelsa i Markusa Jakobssona. Jedan od ranih PoW sustava bio je *Hashcash* čija je svrha bila minimizirati neželjenu poštu i DoS napade. *Hashcash* ovo ostvaruje koristeći koncept parcijalnih kolizija *hash* vrijednosti zahtijevajući da uz svaku e-poštu bude priložen generiran *hash*. Za generiranje *hasha* koristi se e-pošta adresa primatelja, zahtijeva procesorsko vrijeme što dodaje mikro vrijednost svakoj poslanoj e-pošti, te čini slanje neželjnih poruka nepraktično u smislu potrebnog vremena za generiranje velikog broja *hash* vrijednosti za različite.

Block chain mreže koje koriste PoW za potvrđivanje transakcija i kreiranje novih blokova uvode koncept rudarenja (engl. *mining*). Svaki blok na *block chainu* ima svoj *hash* te se koristi kod izračunavanja *hasha* novog bloka dodavanjem sljedećeg bloka transakcija na tu *hash* vrijednost. Da bi blok bio prihvaćen na mreži, rudari moraju otkriti odgovarajuću *nonce* vrijednost što će rezultirati dodavanjem bloka na *block chain* i generiranjem nagrade za otkrivanje bloka.

S povećanjem broja rudara na mreži, vrijeme pronalaska bloka smanjuje se. Da bi blokovi bili konzistentno generirani na temelju *block time* vrijednosti, uvodi se *difficulty* koncept. Velika količina rudara u PoW *block chain* mrežama grupiraju se u *poolove*, kako bi mogli konzistentno generirati blokove i ostvarivati nagrade za otkrivanje bloka redovito, umjesto nasumično jednom svakih par godina. Osim pružanja sigurnosti mreži, PoW je također metoda za redistribuciju bogatstva sudionicima u mreži.

S obzirom na veliku količinu izračuna potrebnu kako bi se pronašli blokovi kroz rudarenje, tako se PoW često smatra rasipnim i neučinkovitim. Osim stoga, hardver koji se sve češće koristi za rudarenje su specijalizirani ASIC uređaji. Koncept *ASIC-resistance* nastao je kao odgovor na ovu pojavu sve veće količine specijaliziranih uređaja za rudarenje zbog činjenice da s pojavom ovakvog hardvera, *block chain* mreže teže centralizaciji zbog ovisnosti o malom broju proizvođača ovih uređaja. Ovo omogućuje otvoreni model distribucije tokena i smanjuje mogućnosti da jedan *node* na mreži kontrolira disproporcijalnom količinom izračuna.

ProgPoW (*Programmatic Proof-of-Work*) algoritam dizajniran je kako bi u procesu rudarenja koristio svaki dio komercijalno dostupnih GPU uređaja. ProgPoW djeluje tako da smanjuje razlike u performansama generiranja izračuna između ASIC i GPU uređaja te na ovaj način smanjuje isplativost ASIC uređaja.

2.2.1.2 ***Proof of Stake*** **(PoS)**

Dokaz o ulogu *(Proof of Stake)* konsenzus mehanizam uvodi koncept validiranja blokova umjesto rudarenja. Validatori ulažu kriptovalute kako bi mogli predlagati blokove. Iz perspektive algoritma konsenzus mehanizma PoS dijeli se na dva tipa:

1. *Chain-Based PoS* – tip konsenzus mehanizma u kojem algoritam nasumično odabere validatora koji tada generira novi blok

2. BFT PoS ili *Byzantine Fault Tolerant PoS* – algoritam nasumično određuje validatora koji predlaže novi blok te se pokreće proces glasanja gdje svaki validator glasa za sljedeći blok.

2.3. Ranjivosti

2.3.1. Napad 51 %

Napad 51 % je napad na konsenzus sloj *block chain* mreže u kojem napadač mora kontrolirati više od pola ukupnog *hashratea* na mreži. U ovom slučaju napadač ima mogućnost reorganiziranja transakcija u blokovima. Kontroliranje većine PoW snage na mreži također omogućuje napadaču sprječavanje otkrivanja novih blokova rudarima, monopolizirajući proces dodavanja blokova.

Ako napadač uspije stvoriti količinu PoW izračuna koja je veća od 50 % ukupnog izračuna na mreži može napraviti takozvani *double spend* napad što im omogućuje da naprave više transakcija s istim UTXO. Transakcija se smatra validnom kada je stavljena u blok na *block chainu*. Sa svakim novim dodatnim blokom, transakcija dobiva *block confirmation* i mogućnost *double spend* napada postaju manje.

U usporedbi s većim kriptovalutama kao što su Bitcoin i Ethereum, ostale kriptovalute imaju relativno malu količinu PoW snage koja osigurava njihov *block chain*. Nekoliko zapaženih primjera kriptovaluta koje su bile žrtve 51 % napada uključuju Ethereum Classic, Bitcoin Gold i ZenCash.

2.3.2. Arbitražni i *oracle* napadi

U radu *Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges* pokazani su potencijalni napadi na Ethereum mrežu koristeći aplikacijski sloj *block chaina*. Arbitražni sustavi na decentraliziranim mjenjačnicama na Ethereumu mogu sudjelovati u takozvanim *Priority Gas* aukcijama u kojima arbitražni botovi, povećavajući gas vrijednosti transakcija, mogu izvršiti narudžbe prije ostalih korisnika. U ovakvim situacijama botovi sudjeluju u jednoj vrsti aukcije natječući se čija transakcija će biti prva prihvaćena. Rad teoretizira napad na Ethereum koji se sastoji od rudara koji promatra nedavno iskorištene prilike za arbitražu na DEX, nakon čega rudar odbaci zadnjih par blokova te stvori *fork* radeći 51 % napad, reorganizira transakcije koje se

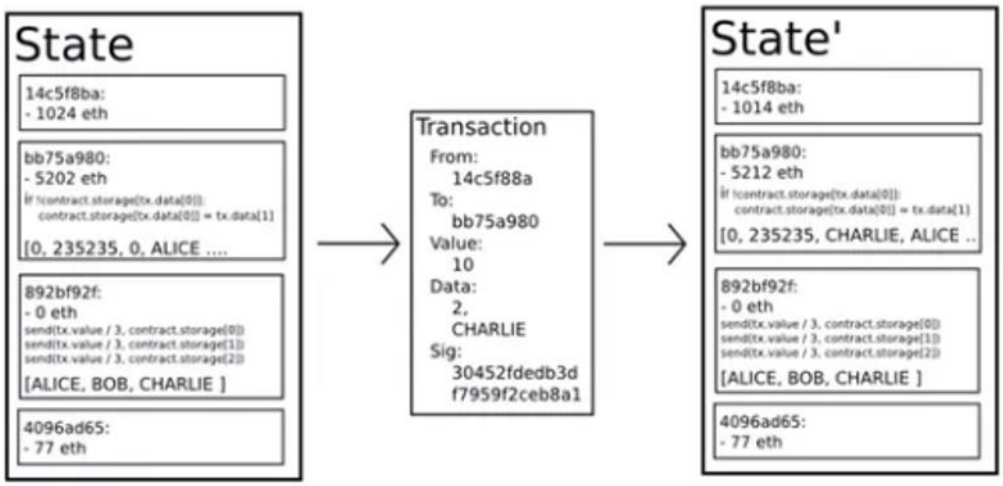
odnose na promatrane prilike za arbitražu kradući ta sredstva, te njima subvencionira daljnji napad na konsenzus sloj mreže kupovanjem više PoW snage. Ovaj primjer napada dokazuje da je moguće iskoristiti vrijednost aplikacijskog sloja Ethereum mreže za financiranje napada na konsenzus sloj.

S proširivanjem koncepta pametnog ugovora dodavanjem *oracle* mehanizma, površina napada na ugovore postaje veća zbog činjenice da sustav mora sadržavati *off-chain* komponentu. Synthetix, platforma za generiranje sintetičke imovine na Ethereumu, nedavno je pretrpjela napad na *oracle* mehanizam koji je zadužen za dostavljanje *off-chain* informacija pametnim ugovorima. Napadač je uspio manipulirati *oracle* mehanizmom koji je dostavljao cijenu južnokorejskog *wona* te na taj način ostvariti profit.

3. Ethereum

Ethereum je *block chain* s *peer-to-peer* mrežnim protokolom i Touring kompletnim programskim jezikom. Nodeovi na ETH mreži pokreću EVM (*Ethereum virtual machine*). Ugovori su opisani kao autonomni agenti unutra Ethereum izvršnog područja koji izvršavaju predefinirane funkcije ako je to zatraženo od njih kroz transakcije.

3.1. Ethereum kao sustav tranzicije stanja



Slika 3-1 Ethereum tranzicija stanja

Tijekom svog životnog vijeka, ugovori prolaze od početnog stanja, kroz nekoliko posrednih stanja do konačnog stanja. U svakom od stanja, ugovor se mora ponašati na drugačiji način te pružiti drugačije funkcionalnosti. *State machine* koncept odnosi se na stroj koji može unositi informacije i na temelju njih napraviti tranziciju u novo stanje.

Ledger kriptovalute Bitcoin može se smatrati kao sustav tranzicije stanja u kojem je stanje opisano vlasništvom svih postojećih Bitcoina ili nepotrošenih UTXO. Svaki UTXO ima odgovarajućeg vlasnika denominiranog adresom. Tranzicije stanja ostvarene su transakcijama koje sadrže jedan ili više ulaza i referenciraju postojeći UTXO. Transakcije sadrže kriptografski potpis kreiran s privatnim ključem asociranim s adresom vlasnika UTXO. Redoslijed transakcija ostvaren je u kombinaciji s konsenzus mehanizmom. Bitcoin *block chain* se obično smatra neprimjerenim za pametne ugovore zbog činjenice da skriptni

jezik nije Touring kompletan. Također, UTXO u Bitcoin mreži može imati samo 2 stanja – potrošen ili nepotrošen, što znači da ne postoji način za stvaranje kompleksnijih ugovora s više faza.

U Ethereum tehničkoj dokumentaciji, trenutno stanje *ledgera* opisano je kao apstrakcija koja nije enkodirana u blok. Stanje svakog bloka može biti ostvareno samo uzimajući u obzir sve transakcije u svim prijašnjim blokovima počevši od početnog ili *genesis* bloka. Na Ethereumu korisnici mogu stvarati pametne ugovore s proizvoljnim pravilima za upravljanje vlasništvom i funkcijama prelaska stanja.

3.2. Ethereum virutalni stroj

EVM je kvazi-Turing kompletan virtualni stroj koji služi kao izvršno okružje za pametne ugovore. Za razliku od standardnih računala koja su limitirana u izvršavanju prema količini memorije i vremenu, EVM limitira izvršavanje koda na temelju *gas* vrijednosti. *Gas* je jedinica koja koristi kao mjera cijene izvršavanja nekog koda. Izračuni na EVM-u održavaju se u *bytecode* jeziku te su potpuno deterministični, što znači da s danim ulaznim, uvijek stvaraju iste promjene stanja. EVM koristi odvojeno izvršno područje (*sandbox*) za odvajanje izvršavanja aplikacija od kritičnih resursa. EVM se razlikuje od von Neumannove arhitekture računala po tome što memoriju čuva u zasebnom virtualnom ROM-u kojem se može pristupiti samo kroz specijalizirane instrukcije. Pametni ugovori najčešće su pisani u jednom od viših programskih jezika koji EVM kompilira u *bytecode* prilikom izvršavanja.

Ewasm (Ethereum WebAssembly) je predložena zamjena za EVM prilikom Ethereum Serenity nadogradnje. *Wasm* ili *WebAssembly* je instrukcijski set razvijen od strane W3C *(World Wide Web Consortium)* konzorcija.

3.3. Računi

Ethereum račun je adresa od 20 bajtova koja sadrži stanje vlasništva Ethera. Računi su generirani kroz proces asimetrične kriptografije. Generiranjem heksadecimalne vrijednosti od 64 znamenke dobiva se privatni ključ. Koristeći operaciju multipliciranja točke na eliptičnoj krivulji dobiva se javni ključ. Računi u Ethereumu mogu se podijeliti na adrese računa s vanjskim vlasništvom *(externally owned*) i adrese ugovora *(contract account*). Računi s vanjskim vlasništvom su kontrolirani s privatnim ključevima u vlasništvu korisnika te ne sadrže nikakav kod asociran s adresom. Adrese ugovora kontrolira njihov kod. Slanjem

transakcije na račun ugovora, ugovor može izvršiti neku funkciju ili stvoriti podugovor. Neke decentralizirane aplikacije i novčanici koriste mnemoničke fraze ili JSON *keystore* datoteke za oporavak računa

3.4. Transakcije

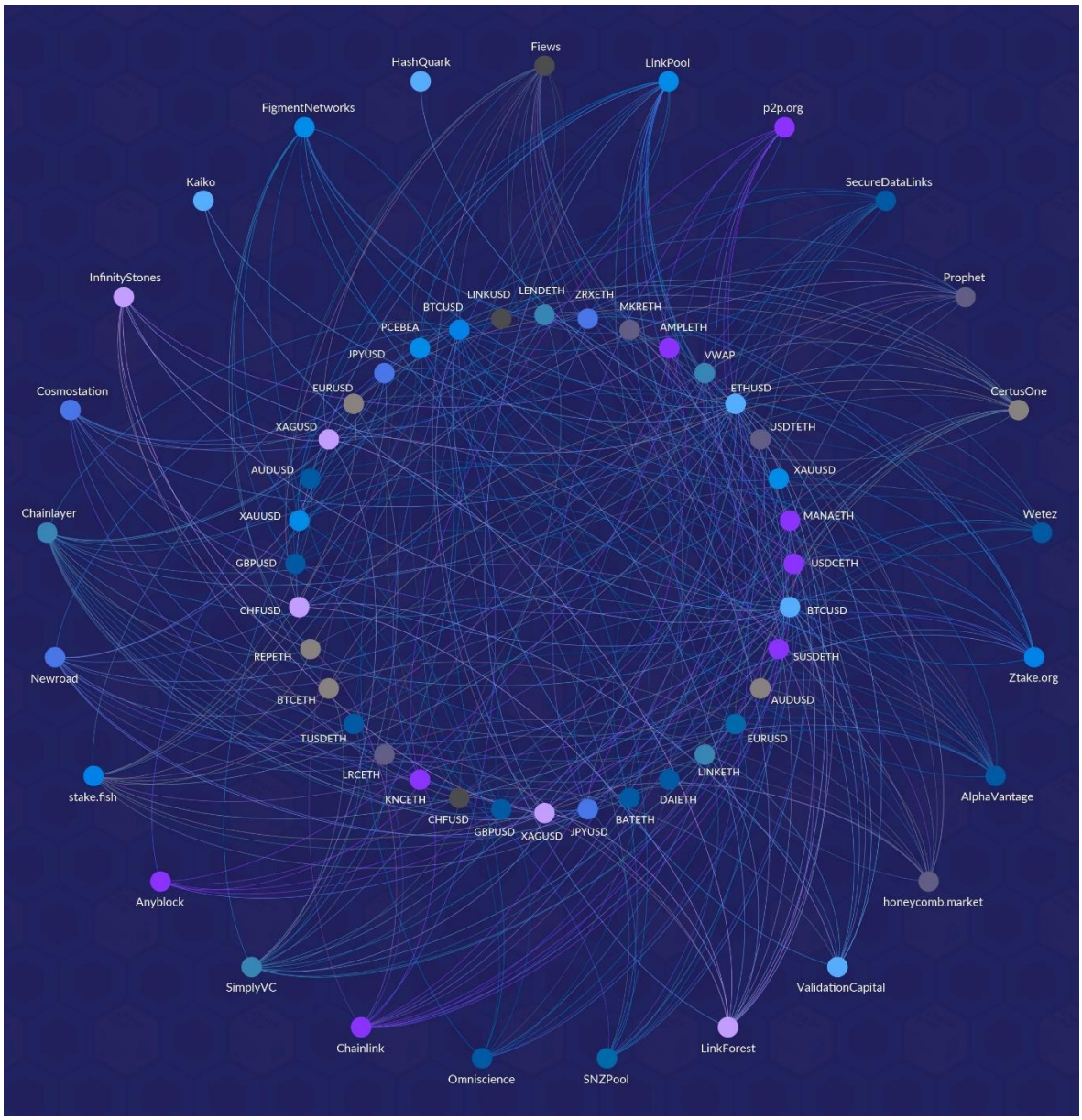
Transakcije u Ethereumu predstavljaju potpisane pakete informacija poslane iz računa u vanjskom vlasništvu. Pozivi poruka (*message calls*) proizvedeni su od ugovora. Za razliku od Bitcoin mreže, gdje sve transakcije zahtijevaju jednaku količinu izračuna na mreži, transakcije na Ethereumu mogu varirati u količini potrebnih proračuna. Transakcije se sastoje od adrese primatelja, količine za transakciju, potpisa pošiljatelja, *gas price* vrijednosti, *gas limit* vrijednosti i opcionalnog *data* polja. *Gas price* vrijednost definira maksimalan broj koraka u izračunu dozvoljenih transakciji, a *gas limit* vrijednost najveću količinu Ethera dozvoljenu transakciji. Cijena transakcije izračunava se umnožavanjem *gas price* i *gas limit* vrijednosti. S *gas* konceptom Ethereum mreža osigurava da svaki izračun na mreži ima svoju monetarnu vrijednost, što pomaže kod suzbijanja spam napada.

3.5. *Off-chain* komponente

3.5.1. *Oracle*

U *block chain* terminologiji, *Oracle* mehanizam je digitalni agent koji uslužuje pametni ugovor da bi ugovor mogao pristupiti sustavima izvan *block chain* mreže. Ovo se ostvaruje povezivanjem s eksternim konektorima (API) i omogućuje unos ili izlaz podataka u pametnim ugovorima.

Chainlink je jedan od projekta koji pokušava olakšati developerima spajanje svojih pametnih ugovora na *off-chain* resurse. *Chainlink* ovo omogućuje koristeći decentraliziranu *Oracle* mrežu sačinjenu od velikog broja nezavisnih *nodeova*. Odabiranjem većeg broja izvora podataka (*nodeova*), povećava se šansa za dobivanjem točnog odgovora o nekoj vanjskoj informaciji. Prije slanja u pametni ugovor odgovori individualnih *nodeova* su agregirani. *Chainlink* mreža trenutno uslužuje informacije o cijenama 36 valutnih parova na Ethereum mreži.



Slika 3-2 *Chainlink* referentni ugovori i valutni parovi

3.5.2. Specifikacija za *off-chain* povjerljive izračune

TCF ili *Trusted Compute Specification* je specifikacija koja se odnosi na skalabilnost i privatnost *block chain* mreža kod izvođenja izračuna izvan *block chain* okružja. TCF ovo omogućava koristeći Intelovu implementaciju izoliranog izvršnog područja (*Trusted Execution Environment*) SGX, ili *Secure Guard Extensions*. Jedno od svojstva TEE okruženja je korištenje izoliranog okružja za *off-chain* izračune koje koristi set privatnih ključeva ugrađenih u čip od strane proizvođača, zapisanih na programibilnoj memoriji koja se ne može primijeniti. Pripadajući set javnih ključeva nalaze se kod proizvođača čipa i koriste se za kontrolu pristupa i potpisivanje autoriziranog softvera. Samo softver potpisan

od stranke koja sadrži javni par ključeva može se izvršiti u ovom okružju i dobiti pristup privatnim ključevima zapisanim na čipu. Jedna od implikacija ovakvog sustava je da nepotpisan softver ne može doći do privatnog ključa sadržanog u čipu bez interakcije sa strankom koja čuva drugi par ključeva. *Secure Guard Extensions* (SGX) predstavlja Intelovu implementaciju TEE i uvodi koncept *enclavea*, tj. privatnog dijela memorije čiji sadržaj je odvojen od ostalih procesa svih razina privilegija i operacijskog sustava. SGX atestacija osigurava da će se kod izvršiti na platformi sa odgovarajućim sposobnostima i samo s autoriziranim softverom.

3.6. Prelazak na PoS

Casper je *BFT PoS* implementacija koju će koristiti novi Ethereum Beacon *block chain* nakon takozvane Serenity nadogradnje. U Casper PoS sustavu validatori potvrđuju nove blokove stavljajući ulog uz svaki predloženi blok. Ulozi su potrebni da bi se izbjegao takozvani *Nothing at Stake* problem kod PoS sustava. *Nothing at Stake* odnosi se na situaciju kada se dogodi *fork* te govori da je validatorima u interesu nastaviti objavljivati blokove na oba *block chaina* jer nemaju nikakav ulog povezan s predlaganjem blokova na krivom *block chainu*. Na ovaj način validator se osigurava u slučaju ako bilo koji od *forkanih block chainova* postane najduži. Kod PoW sustava ovaj problem izbjegnut je činjenicom da rudari neće nastaviti rudariti manje profitabilan *block chain*. Postavljanje uloga sa svakim predloženim blokom omogućuje mehanizam kažnjavanja i osigurava da u ekonomskom interesu validatora budu iskreni.

Konsenzus mehanizam ključna je komponenta *block chaina* koja pruža način ostvarivanja dogovora oko trenutnog stanja mreže. Nakamoto Konsenzus algoritam bio je prvi *block chain* konsenzus algoritam stvoren i do danas pogoni Bitcoin mrežu. Ethereum PoS implementacija će donijeti napretke u usklađivanju incentivizacija sudionika mreže. Jedan od čestih problema u PoW konsenzus mehanizmima je da se interesi rudara i korisnika mreže ne podudaraju. Casper implementacija pokušava riješiti ovaj problem zahtijevajući da konsenzus agenti imaju direktnu investiciju u kriptovalutu.

4. Pametni ugovori

Izraz *smart contract* prvi puta bio je spomenut 1994. od strane američkog kriptografa i računalnog znanstvenika Nicka Szabe. Pametni ugovori definirani su kao automatizirani transakcijski protokoli koji mogu izvršavati uvjete ugovora i minimiziraju potrebu za posrednicima. Ovi ugovori predstavljaju skup predefiniranih pravila koja upravljaju ugovorom. Bitcoin je bio prvi javni *block chain* s podrškom za koncept pametnih ugovora. Jedan od primjera ugovora na Bitcoinu su takozvane M-od-N adrese ili *multi-signature* adrese. Za razliku od standardnih transakcija koje zahtijevaju samo jedan potpis privatnim ključem, *multi-signature* adrese zahtijevaju više potpisa s više privatnih ključeva kako bi se transakcija izvršila. Jedna od implementacija ovakvog tipa adresa su *escrow* ili uvjetni računi. Npr. 2-od-3 *escrow* adresa zahtijeva potpis minimalno 2 stranke da bi se transakcija izvršila.

4.1. Razvojni alati

4.1.1. Open-Zeppelin *library*

Open-Zeppelin je *open-source* zbirka ugovora za stvaranje decentraliziranih aplikacija na Ethereumu. Open-Zeppelin zbirka sastoji se od velikog broja ugovora koje developeri mogu koristiti prilikom izgradnje vlastitih ugovora i aplikacija u Solidity programskom jeziku. Koristeći Open-zeppelin ugovore aplikacija može dobiti neke garancije kvalitete iz perspektive revizije ugovora. Open-Zeppelin ERC721 ugovor definira minimalne zahtjeve sučelja za implementiranje ugovora koje može upravljati s ERC721 tokenima. SafeMath ugovor kontrolira aritmetičke operacije u ugovoru. U NFTAuction ugovoru SafeMath je korišten za objekte tipa uint256.

4.1.2. Truffle suite

Truffle je okruženje za razvijanje i testiranje pametnih ugovora na Ethereumu koristeći Solidity programski jezik. Truffle omogućuje kompilaciju i objavljivanje ugovora na javne ili privatne mreže te sadrži module za testiranje ugovora.

Ganache je jedan od alat Truffle suite grupe programa za razvijanje ugovora i decentraliziranih aplikacija. Omogućuje podizanje privatnog *block chain* okružja koje

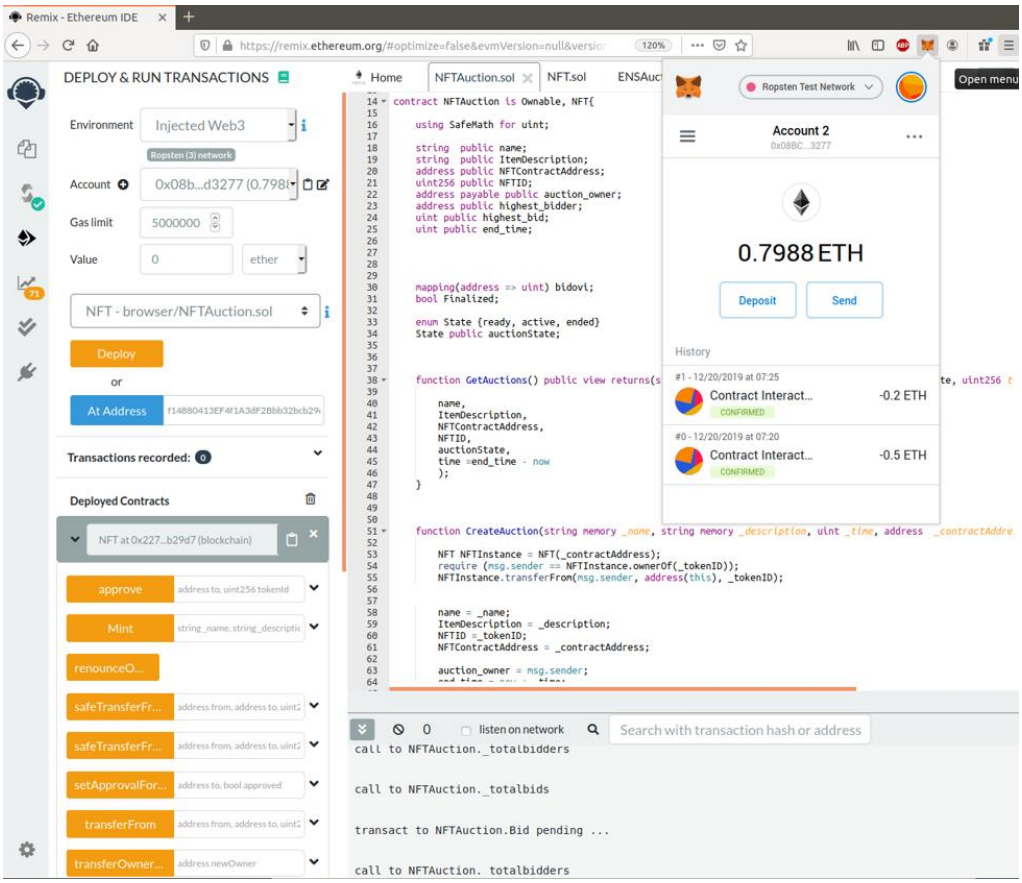
simulira javnu mrežu. Ganache pri inicijalizaciji postavlja 10 računa sa 100 simuliranih Ethera u svrhu testiranja te ne zahtijeva rudarenje već zapisuje sve transakcije koje zaprimi.

4.1.3. Remix IDE

Remix je Solidity kompajler i integrirano razvojno okružje bazirano na internetu. Remix simulira EVM te omogućuje developerima da testiraju koncepte aplikacija i pametnih ugovora bez spajanja na glavnu Ethereum mrežu. Postoje tri tipa okružja koja mogu biti korištena: *JavascriptVM*, *Injected provider* i *Web3 provider*. Dok *JavascriptVM* simulira izoliranu EVM direktno u *browseru*, *Injected provider* i *Web3 provider* opcije zahtijevaju eksterne alate za spajanje s mrežom.

Solidity je programski jezik sličan jezicima poput C++ i *javascript*. Solidity pokreće Ethereum virtualni stroj te podržava nasljeđivanje ugovora.

Metamask je *open source* dodatak za browser koji služi kao web3 *provider* za Ethereum tako da izlaže svoj API web stranicama i omogućuje interakciju s aplikacijama na Ethereumu bez potrebe za spajanje na Ethereum *node*. Web stranice koje podržavaju Metamask mogu slati poruke na API koje korisnik može potpisivati i objavljivati na mrežu.



Slika 4-1 Remix integrirano razvojno okružje

4.1.4. Testnet

Test mreže na Ethereumu služe da bi developeri pametnih ugovora mogli objaviti i testirati ugovore na mreži koja ima iste funkcionalnosti kao glavna Ethereum mreža bez potrebe za izvršavanjem transakcija sa stvarnom vrijednosti. Za Ethereum postoji više testnih mreža:

- Ropsten je *testnet* mreža koja najviše naliči na stvarnu Ethereum mrežu. Ropsten mreža koristi sličan PoW konsenzus algoritam kao glavna mreža i ima *block time* od 30 sekundi. U ovoj mreži Ether može biti rudaren.
- Rinkeby je *testnet* mreža koja koristi *Proof of Authority* konsenzus mehanizam. *Proof of Authority* zahtijeva da korisnici potvrde identitet prije nego mogu zatražiti Ether. Ovo se najčešće radi objavljivanjem objave na nekoj od društvenih mreža. Blokovi su generirani svakih 15 sekundi.
- Kovan također koristi *Proof of Authority* konsenzus mehanizam. Kovan je nastao kao projekt *block chain* infrastrukturne organizacije *Parity* kao odgovor na česte napade neželjenom poštom na Ropsten mrežu. Ether na Kovan mreži ne može se rudariti i *block time* je 4 sekunde.

5. Razvoj aplikacije

ERC je akronim za *Ethereum Request for Comments* i predstavlja set standarda za razvijanje tokena, registra domena i drugo. ERC standardi definiraju set standardnih sučelja i funkcionalnosti za različite vrste tokena kako bi bili kompatibilni s ostalim ugovorima na Ethereumu. Ovi standardi osiguravaju interoperabilnost između različitih aplikacija na mreži. Također, korištenje ERC standarda za kreiranje tokena ima prednosti u smislu korištenja već provjerenog koda. Velik broj aplikacija, kao što su digitalni novčanici za kriptovalute, podržavaju nekoliko najčešće korištenih ERC standarda kao što su ERC20 i ERC721. Tokeni na Ethereum mreži implementirani su kroz pametne ugovore i izvršavaju se na EVM.

5.1. ERC20

ERC20 je definiran kao tehnički standard za sučelje tokena koji opisuje set zajedničkih pravila kako bi token mogao biti interoperabilan sa ostalim ugovorim na Ethereum mreži. ERC20 standard često je korišten u kampanjama za financiranje kroz ICO (*Initial Coin Offering*) ugovore.

Pošto kriptovaluta Ether nije kompatibilna sa ERC20 specifikacijom, korisnici koji žele pretvoriti svoj Ether u ERC20 token mogu to napraviti slanjem Ether valute u pametni ugovor koji će vratiti nazad istu količinu WETH (*Wrapped ETH*) tokena koji je kompatibilan s ERC20 specifikacijom.

5.2. ERC 721

Za razliku od ERC20 standarda, ERC 721 opisuje implementaciju token ugovora u kojem su tokeni međusobno nezamjenjivi ili jedinstveni. ERC 721 mora održavati zapis vlasništva svakog individualnog tokena i adrese kojoj pripada. Također, ERC721 tokeni ne nose istu vrijednost. Ova vrsta tokena može biti vrlo korisna za umjetnička djela zbog svojstva dokažljive rijetkosti.

F1DeltaTime službena je Formula 1 igra koja koristi koncept nezamjenjivih tokena na Ethereum *block chainu*. Tijekom 2019. održao se niz nizozemskih aukcija na Ethereumu za

licencirane digitalne tokene koji predstavljaju F1 aute u kojem je jedan primjerak bio prodan za 415 Ethera.

Jedna od poznatijih implementacija ovog tipa ugovora je decentralizirana aplikacija CryptoKitties, igra na Ethereumu u kojoj svaki token predstavlja jedinstvenu virtualnu mačku, koje se mogu kupovati, prodavati i međusobno razmnožavati.

5.3. Praktična izvedba ERC721 token ugovora

NFT ugovor nasljeđuje Ownable ugovor iz OpenZeppelin zbirke ugovora što omogućuje postavljanje vlasništva ugovora tako da bi samo vlasnik ugovora mogao kreirati tokene.

```
pragma solidity 0.5.11;

import 'https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/ownership/Ownable.sol';
import 'https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol';
import 'https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/SafeMath.sol';



contract NFT is Ownable, ERC721  {

    using SafeMath for uint256;

    string public description;
    string public name;
    string public constant symbol = "UNFT";

    address NFTowner;

    mapping (uint256 => string) private metaData;
    mapping (uint256 => address)  NFTowners;


    function TokenMetadata (uint256 _tokenId) external view  returns (string memory) {
        string memory _tokenURL = metaData[_tokenId];
        return _tokenURL;
    }


    function Mint (string memory _name, string memory _description,address _to, uint256 _tokenId, string memory _tokenURL) onlyOwner public {
        require(_to != address(0));
        name = _name;
        description = _description;
        _mint(_to, _tokenId);

        metaData[_tokenId] = _tokenURL;

        NFTowners[_tokenId]= _to;

    }
```

5.4. Praktična izvedba ugovora za aukciju ERC721 tokena

Nft-auction-dapp je decentralizirana aplikacija na Ethereumu koja omogućuje stvaranje aukcija *on-chain* dobara. Svaka *on-chain* imovina koja podržava ERC721 sučelje može biti predmet aukcije. To mogu biti ENS imena, virtualni predmeti ili tokeni koji predstavljaju

off-chain imovinu kao na primjer ulaznice. Korisnici također imaju mogućnost generiranja vlastitih ERC721 kompatibilnih tokena kroz NFT.sol ugovor.

U NFT.sol ugovoru, token predstavlja objekt s atributima postavljenim kroz _setTokenURI metodu naslijeđenu iz ERC721Metadata ugovora. NFT ugovor sadrži Mint funkciju koja omogućava vlasniku ugovora da kreira tokene i preda vlasništvo nekoj adresi. Token služi kao digitalna reprezentacija imovine u vlasništvu korisnika. Prilikom kreiranja tokena potrebno je definirati naziv tokena, njegov opis, adresu na koju se token šalje, ID tokena i njegove meta-podatke. Svi ugovori pisani su za 0.5.11 verziju Solidity kompajlera.

Pošto NFTAuction.sol ugovor implementira standardno ERC721 sučelje, može se koristiti s bilo kojim tokenom implementiranim koristeći OpenZeppelin ili 0xcert zbirke ugovora. Ugovor nasljeđuje ERC721, Ownable i SafeMath ugovore iz OpenZeppelin zbirke ugovora.

```
14 contract NFTAuction is Ownable, NFT{
15
16     using SafeMath for uint;
17
18     string  public name;
19     string  public ItemDescription;
20     address public NFTContractAddress;
21     uint256 public NFTID;
22     address payable public auction_owner;
23     address public highest_bidder;
24     uint public highest_bid;
25     uint public end_time;
26     uint private totalbids;
```

Slika 5-1 Definiranje varijabli za aukciju

Ugovor definira varijable za spremanje podataka o adresi vlasnika aukcije (auction_owner) i adresi najvišeg ponuđača u aukciji (highest_bidder). NFTContractAddress varijabla služi za čuvanje adrese s koje je generiran NFT koji korisnik želi staviti na aukciju. End_time varijabla služi za izračunavanje kraja aukcije. SafeMath ugovor služi za sprječavanje greška u aritmetičkim operacijama u ugovoru i sprječava prelijevanje (*overflow*) uint operacija. Ovo omogućuje da se transakcija poništi i da se vrate sve promjene stanja u slučaju ako dođe do preljeva u izračunu.

```
30      mapping(address => uint) bidovi;
31      bool Finalized;
32
33      enum State {ready, active, ended}
34      State public auctionState;
35
36
```

Slika 5-2 Mapiranje adresa i ponuda

Mapiranje u Solidityu koristi se za strukturiranje vrijednosti. Na slici 6. vidljivo je mapiranje adrese na uint vrijednosti (bidovi). Ovo mapiranje je kasnije korisno u ugovoru jer stvara vezu između adresa i njihovih ponuda.

Za modeliranje različitih stanja u ugovoru koriste se enum funkcije. Enum stanja mogu se pretvoriti u cjelobrojne vrijednosti i predstavljaju korisnički definirane vrste podataka. U gornjem primjeru enum funkcija služi za enumeriranje stanja aukcije. Aukcija u svakom trenutku može imati samo jedno od 3 stanja: *ready*, *active* ili *ended*. Ova stanja služe da bi postojale restrikcije na neke radnje tijekom određenog stanja aukcije.

```
37  function GetAuctions() public view returns(string memory, string memory, address, uint256, uint256, uint256 time) { return (
38      name,
39      ItemDescription,
40      NFTContractAddress,
41      highest_bid,
42      NFTID,
43      time =end_time - now
44      );
45  }
```

Slika 5-3 GetAuctions funkcija

GetAuction funkcija služi sudionicima u aukciji da dobiju informacije o aukciji ako postoji. Funkcija ne zahtijeva nikakve ulazne argumente te vraća ime aukcije, opis aukcije, adresu ugovora koji upravlja s NFT tokenom, trenutnu najveću ponudu, jedinstveni ID tokena, trenutno stanje aukcije i broj sekunda do kraja aukcije. Funkcija je definirana kao *public* funkcija što omogućuje da bude zvana s bilo koje adrese.

```
53  function CreateAuction(string memory _name, string memory _description, uint _time, address  _contractAddress, uint256 _tokenID) public {
54
55      NFT NFTInstance = NFT(_contractAddress);
56      require (msg.sender == NFTInstance.ownerOf(_tokenID));
57      NFTInstance.transferFrom(msg.sender, address(this), _tokenID);
58
59      name = _name;
60      ItemDescription = _description;
61      NFTID =_tokenID;
62      NFTContractAddress = _contractAddress;
63
64      auction_owner = msg.sender;
65      auctionState = State.active;
66      end_time = now + _time;
67  }
```

Slika 5-4 CreateAuction funkcija

CreateAuction funkcija će izbaciti *revert* ako vlasnik tokena nije prethodno odobrio tu adresu. Odobravanje adrese da može upravljati s tokenom vrši se kroz approve() funkciju u NFT ugovoru. Funkcija kao ulaze zahtijeva adresu objavljenog NFT.sol ugovora, ID tokena, naziv i opis aukcije. Nakon instanciranja NFT ugovora funkcija provjerava je li adresa koja pokušava započeti aukciju stvarno vlasnik tokena, nakon čega će preuzeti vlasništvo tokena na svoju adresu koristeći transferFrom() funkciju. Ovakva implementacija depozit mehanizma daje sigurnost u izvršavanje bez potrebe da korisnici moraju sami slati token na adresu ugovora. Ako se transakcija uspješno izvrši, funkcija promjeni stanje aukcije u active i vrijeme trajanja aukcije izračunava se dodavanjem željenog vremena trajanja aukcije u sekundama na sadašnje vrijeme.

```
68      modifier IsNotOwner(){
69          require(msg.sender != auction_owner);
70          _;
71
72      }
73
74      modifier AuctionEnded(){
75          require(now > end_time);
76          _;
77
78      }
79
80      modifier AuctionActive(){
81          require(now < end_time);
82          _;
83
84      }
85
86      modifier IsNotHighestBidder(){
87          require (msg.sender != highest_bidder);
88          _;
89      }
90
```

Slika 5-5 Modifieri

Modifieri u Solidityu služe kao kontrola restrikcija funkcijama. Koristeći *modifere* mogu se postaviti uvjeti koji moraju biti zadovoljeni da bi se izvršila funkcija. IsNotOwner provjerava je li adresa koja instancira transakciju različita od vlasnika. AuctionEnded i AuctionActive modifieri korišteni su u Bid i Withdraw funkcijama.

```
92      function Bid() IsNotOwner AuctionActive public payable {
93
94          require (msg.value > 0);
95
96          uint256 newbid = bidovi[msg.sender].add (msg.value);
97          require( newbid > highest_bid);
98
99          if (highest_bid != 0) {
100
101             bidovi[highest_bidder] = bidovi[highest_bidder] + highest_bid;
102         }
103
104
105         highest_bid = newbid;
106         highest_bidder = msg.sender;
107         totalbids++;
108     }
```

Slika 5-6 Bid funkcija

Postavljanje ponuda na aukciji vrši se kroz Bid funkciju. Kroz IsNotOwner *modifier* postavljen je uvjet da samo adrese koje nisu vlasnici aukcije mogu postavljati ponude. Kroz newbid varijablu izračunava se trenutna ponuda zbrajajući prošlu i novu ponudu. Ponude mogu biti postavljene samo ako je nova ponuda (*newbid*) veća od trenutne highest_bid ponude.

```
110     function Withdraw() AuctionEnded IsNotHighestBidder public returns (bool) {
111
112         uint amount = bidovi[msg.sender];
113         if (amount != 0) {
114
115             bidovi[msg.sender] = 0;
116
117             if (!msg.sender.send(amount)) {
118
119                 bidovi[msg.sender] = amount;
120                 return false;
121             }
122         }
123
124         return true;
125     }
```

Slika 5-7 Withdraw funkcija

Withdraw funkcija može biti zvana samo nakon završetka aukcije da bi ponuđači koji nisu pobijedili na aukciji mogli dobiti natrag ponudu. Kod zvanja funkcije korišten je IsNotHighestBidder *modifier* koji postavlja restrikciju da adresa koja zove funkciju mora biti različita od adrese s koje dolazi najviša ponuda. U slučaju da transakcija ne uspije resetira se dugovana količina Ethera za korisnika *msg.sender*.

```
131     function EndAuction() public {
132
133         require (msg.sender == auction_owner || msg.sender == highest_bidder);
134         require(now > end_time);
135         require(!Finalized);
136
137         if(totalbids != 0){
138             NFT NFTInstance = NFT(NFTContractAddress);
139             NFTInstance.transferFrom(address(this), highest_bidder, NFTID);
140             auction_owner.transfer(highest_bid);
141         }
142         else if (totalbids == 0){
143             NFT NFTInstance = NFT(NFTContractAddress);
144             NFTInstance.transferFrom(address(this),auction_owner, NFTID);
145         }
146
147         auctionState = State.ended;
148         Finalized = true;
149
150     }
151 }
```

Slika 5-8 EndAuction funkcija

EndAuction funkcija može biti zvana samo nakon isteka perioda aukcije i samo od vlasnika aukcije i pobjednika aukcije. Nakon zvanja funkcije, ugovor šalje token na adresu pobjednika aukcije, a vlasniku aukcije šalje najvišu ponudu. Da bi korisnici bili osigurani od trajnog zaključavanja fondova u ugovoru, zvanje funkcije omogućeno je i pobjedniku aukcije. Ako nitko nije postavio ponudu za vrijeme trajanja aukcije ova funkcija vratit će vlasniku token.

5.5. Testiranje ugovora

U svrhu testiranja interoperabilnosti ugovora za aukcije sa standardnim implementacijama NFT tokena, u sljedećem primjeru korišten je ERC721 token dobiven registriranjem imena kroz ENS servis.

ENS ili Ethereum Name Service je usluga koja pruža mogućnost asociranja čitljivih domenskih imena na Ethereum adrese za njihovu identifikaciju. Adrese mogu predstavljati vanjske adrese ili adrese ugovora. Da bi se adrese prevele u čitljiva imena, ENS koristi rekurzivni JavaScript modul Namehash koji generira jedinstven *hash* za domensko ime. ENS omogućuje deriviranje pod-domena, npr. iz domene *alice.eth* moguće je proizvesti domenu *iam.alice.eth*. *Hashevi* domena dobiveni su koristeći keecak256 kriptografsku funkciju na čitljivo ime domene. Prilikom registriranja domene potrebno je napraviti zapis u Resolver ugovoru. Dostupne vršne domene su .eth na *mainnetu* i .eth i .test na ropsten *testnetu*. Trenutno, ENS *Root node* ili *node* koji sadrži glavni zapis je u vlasništvu *multisignature*

ugovora. Za razliku od ostalih *block chain* usluga koje pokušavaju stvoriti alternativu DNS sustavu, ENS pokušava stvoriti decentralizirani sustav za rezoluciju web3 resursa na Ethereumu.

Registriranje ENS imena radi se kroz *ENS Manager* aplikaciju. Registracija se radi tako da korisnik pošalje dvije transakcije, prvo transakciju kojom se obavezuje na registriranje imena, zatim transakciju u kojoj stvarno registrira ime. Kako bi se spriječile *front-running* situacije tako ove transakcije moraju biti vremenski odvojene 1 minutu. *Front-running* situacije nastaju kada napadač nadgleda transakcije kojima neki korisnik pokušava registrirati ime i slanjem transakcija za registriranje istog imena s većom gas vrijednosti uzima ime u svoje vlasništvo te odmah preprodaje korisniku ime za veću cijenu. Registriranje imena sa 7 ili više znakova košta 5 $ godišnje te tijekom ovog perioda vlasnik imena može stvarati pod-domene, dodavati zapise, prodati domenu, itd..

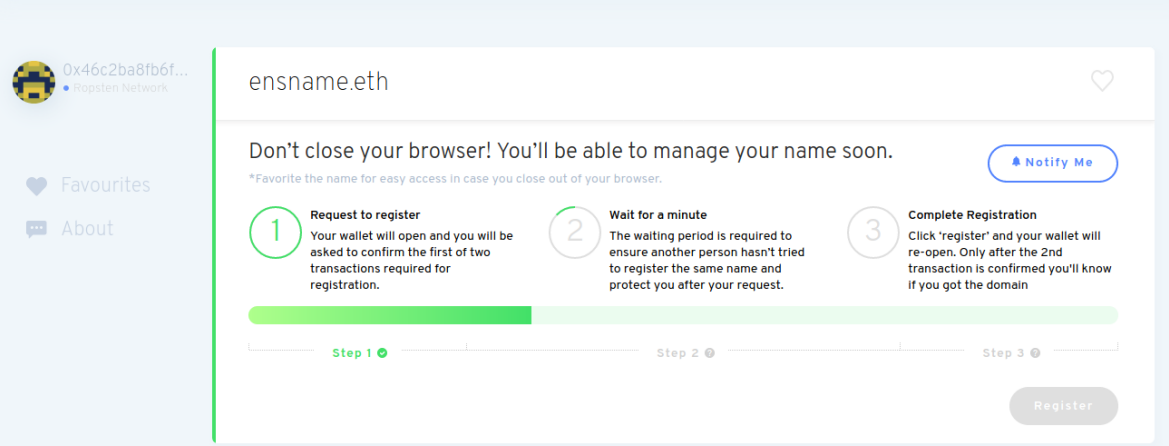
Iako predstavlja funkcionalnosti vrlo slične domenskom sustavu, ENS nije domenski sustav. Proces registriranja imena na ENS je automatiziran proces i ne garantira vlasništvo nekog *trademarka* ili nekog drugog oblika intelektualnog vlasništva. Registriranje ENS imena se ne bazira na nekoj centralnoj trećoj stranci, već zahtijeva proces zaključavanja fondova u pametni ugovor „Registrar“ na određeni vremenski period, minimalno godinu dana, nakon čega se ETH natrag isplaćuje korisniku i njegova prava na ime su oduzeta. Registrar ugovor sadrži standardno sučelje za upravljanje s ERC721 tokenima koji predstavljaju imena što znači da imena mogu biti upravljana na isti način kao bilo koji drugi NFT.

Proces registriranja imena do nedavno izvršavao se tako da je korisnik koji je želio registrirati domenu otvorio javnu aukciju za to ime. Narednih 2 dana, tijekom perioda postavljanja ponuda, druge adrese mogle su sudjelovati u aukciji postavljajući ponude. Postavljanje ponuda u ovakvoj aukciji vrši se slanjem određene količine Ethera i *hash* vrijednosti dobivene od imena aukcije, vrijednosti ponude, adrese s koje dolazi ponuda i *salt* vrijednosti. Korisnici moraju u transakciju uključiti količinu Ethera veću ili jednaku ponudi. Na ovaj način neki korisnici šalju velike količine Ethera kako bi sakrili svoje stvarne ponude. Nakon postavljanja ponuda, pokreće se period otkrivanja ponuda. Ovaj period traje 3 dana i korisnici moraju ručno otkriti ponude u aukciji u kojoj su sudjelovali. Korisnik koji je postavio najvišu ponudu pobjeđuje u aukciji i plaća količinu Ethera kolika je u drugoj najvišoj ponudi u aukciji. Ova vrsta aukcije zove se *Vickery* ili *second price sealed-bid* aukcija. Prednost korištenja ovakve vrste aukcije leži u činjenici da, za vrijeme perioda

postavljanja ponuda, korisnici ne mogu vidjeti tuđe ponude, što incentivizira sudionike da postavljaju ponude na temelju vlastite percepcije vrijednosti predmeta na aukciji i otežava strankama koliziju u smislu namjernog održavanja niske cijene postavljajući veće ponude samo od ponuđača koji nisu dio grupe.

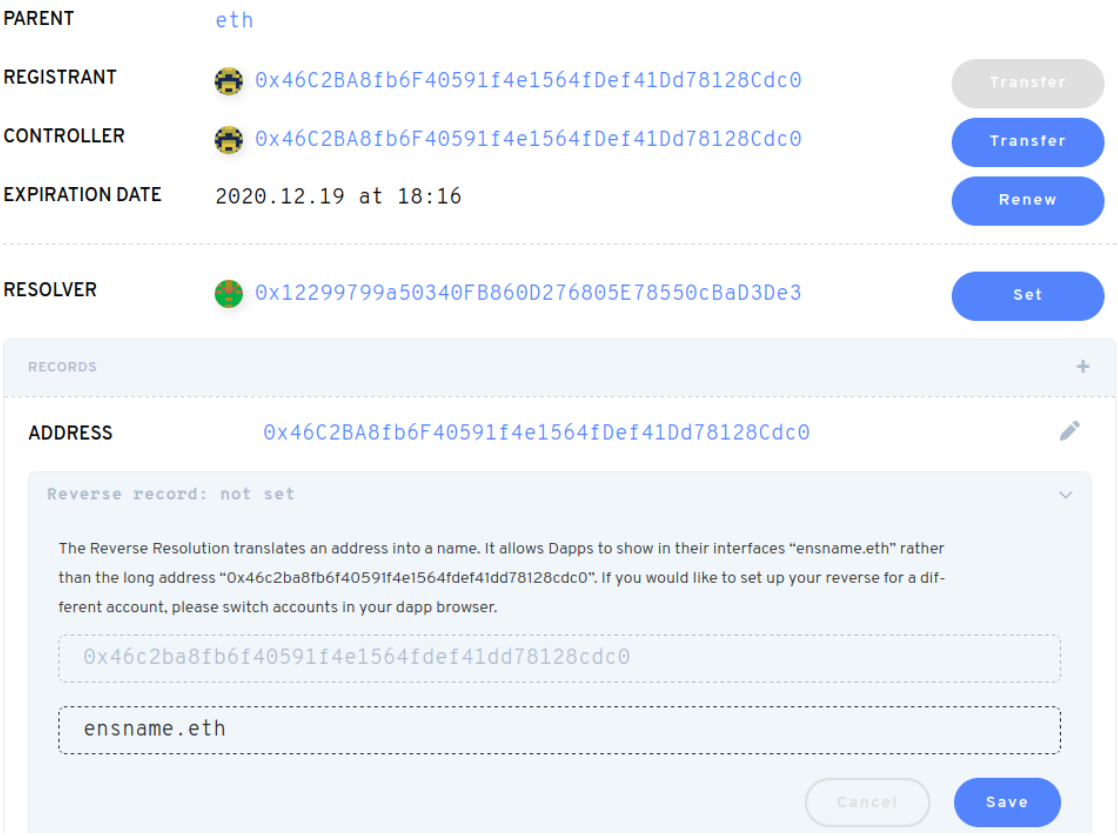
Zbog činjenice da bez odgovarajućeg otkrivanja ponude, EVM ne može znati kolika je ponuda, sudionici aukcije koji nisu otkrili svoje ponude tijekom *reveal* faze aukcije izgubili su ponude i nisu ih mogli izvaditi iz ugovora. Ovakav sustav anonimizacije ponuda u Ethereumu zove se *commit and reveal* sustav i omogućuje aukcije u kojima su ponude anonimne do izvršenja ugovora.

5.5.1. Registriranje ensname.eth imena



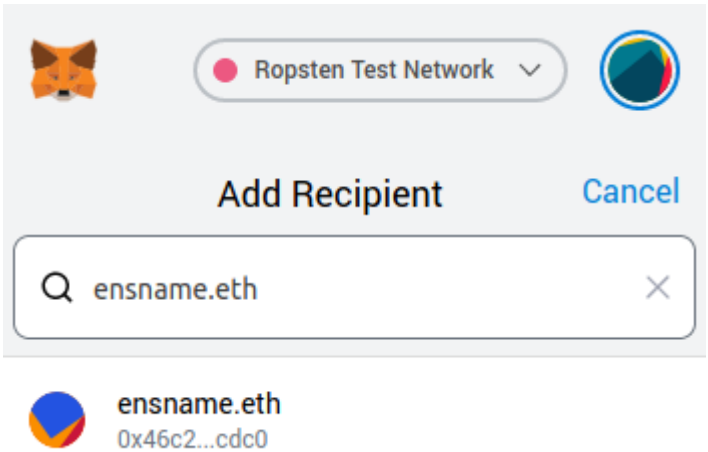
Slika 5-9 Registriranje ensname.eth imena

Permanent registrar je pametni ugovor koji upravlja alokacijom imena u .eth TLD (*Top Level domeni*). Služi kao zamjena za *auction registrar* ugovor koji je koristio *Vickery* tip aukcije za alokaciju imena. Ovaj novi način registriranja imena koristi FCFS (*first-come-first-serve*) metodu dodjele imena registrantima.



Slika 5-10 Postavljanje reverznog zapisa

Registrant adresa je adresa računa koji registrira ime. *Registrant* račun može upravljati s vlasništvom imena i postavljati *controller* adresu. *Resolver* adresa upravlja translacijom imena u adrese i najčešće je postavljena na adresu javnog *resolvera*. Records polje omogućava zapisivanje tekstualnih zapisa ili adresa.

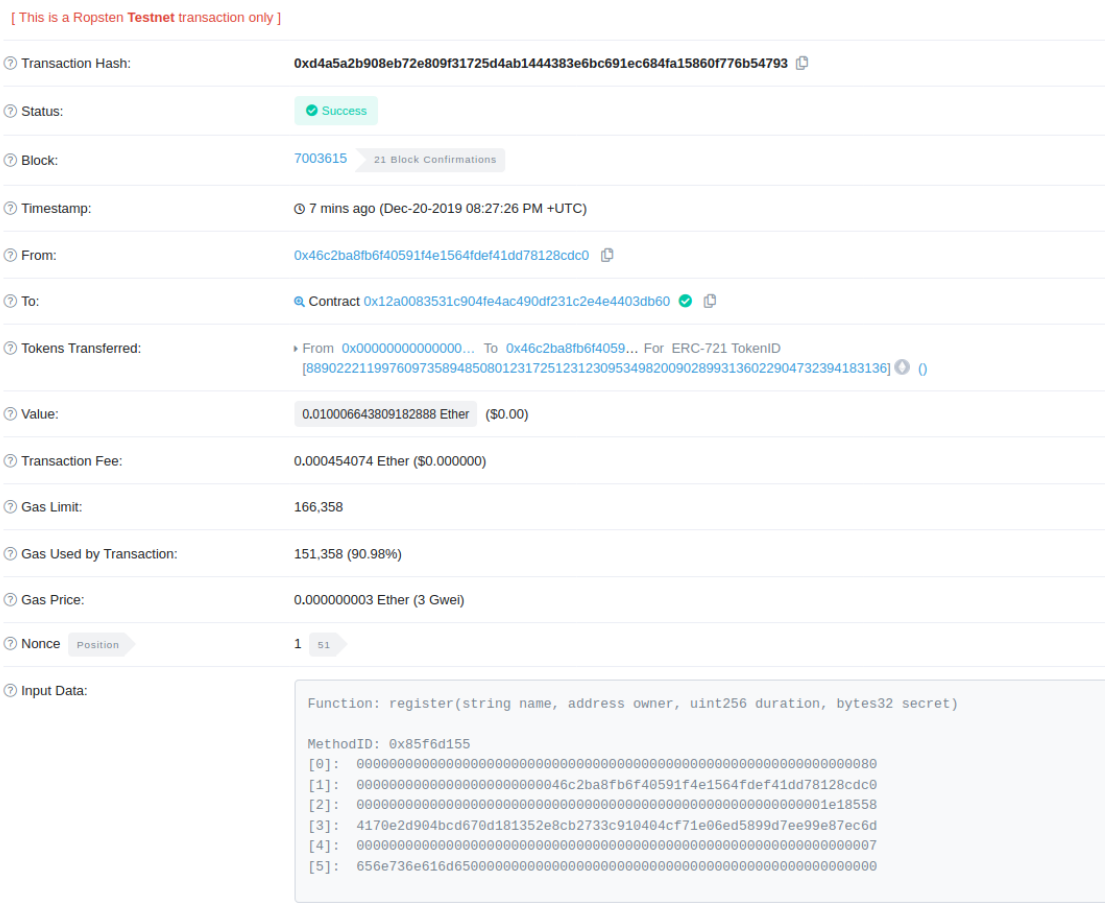


Slika 5-11 Metamask

Na slici 5-11 moguće je vidjeti da aplikacije kao što je Metamask mogu prepoznati adrese asocirane s ENS imenima.

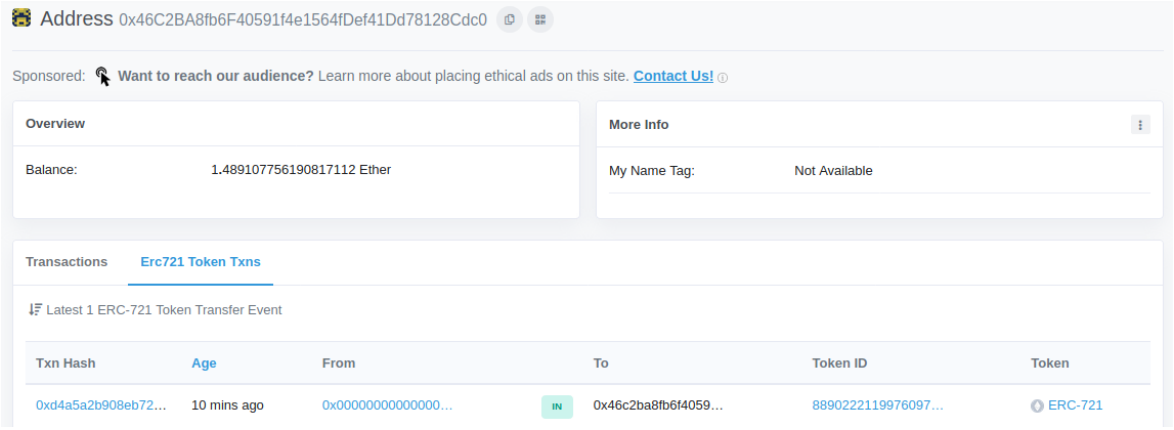
5.5.2. Aukcija ENS NFT tokena

Imena u ENS-u kontrolira *Registrar* ugovor. Vlasništvo imena predstavljeno je kroz ERC721 kompatibilan NFT token, što znači da mogu biti upravljana koristeći standardna ERC721 sučelja ili kroz podržana tržišta kao npr. *OpenSea*. Nakon registracije imena, NFT token je poslan na adresu specificiranu kao vlasnik tog imena.



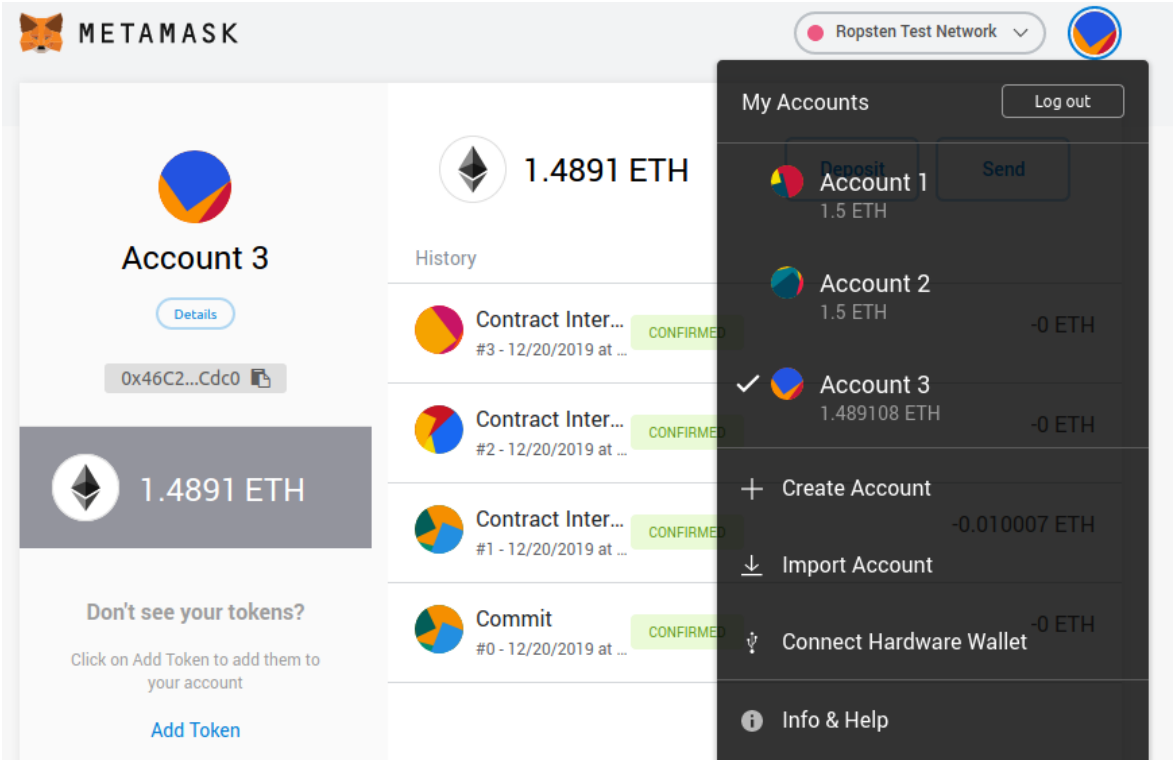
Slika 5-12 Transakcija koja generira NFT

Na slikama 5-12 i 5-13 vidljiva je transakcija iz prijašnje slike generiranja NFT tokena. Vlasniku adrese generiran je token zvanjem *register* funkcije *RegistrarController* ugovora. Tokenu je dodan jedinstven ID. Na slici 5-13 vidljiv je ERC-721 token koji predstavlja vlasništvo ENS imena. Koristeći ENSNFTAuction ugovor vlasnik tokena može stvoriti aukciju.



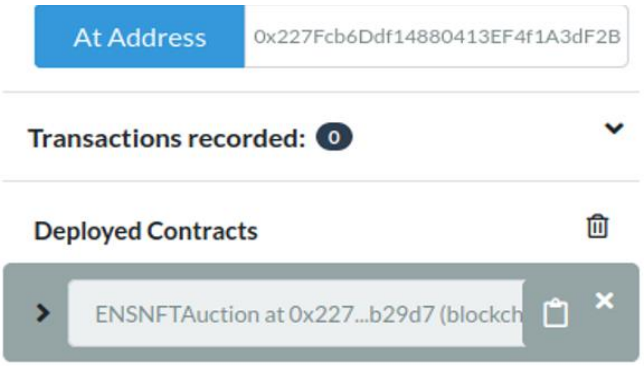
Slika 5-13 ENS token

Testiranje koncepta ugovora za aukcije NFT tokena rađeno je kroz Remix integrirano razvojno okružje. Na Ropsten *testnet* mreži, 3 adrese unesene su u Metamask i svakoj dodano je 1.5 Ropsten Ethera za testiranje funkcija postavljanja ponuda i izvršavanja ugovora.



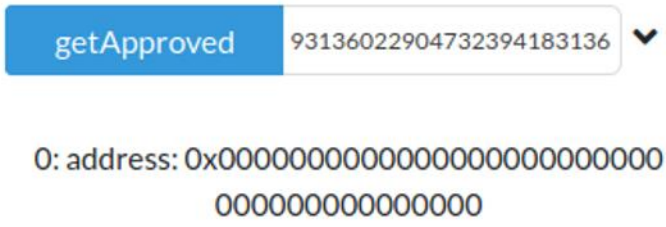
Slika 5-14 Stanja računa prije aukcije

Instanciranje *Registrar* ugovora omogućava zvanje *approve* funkcije tog ugovora kako bi korisnik odobrio adresu ugovora aukcije. Ovo omogućuje ugovoru za aukcije da uzme vlasništvo nad tokenom tijekom procesa aukcije.



Slika 5-15 Instanciranje *Registrar* ugovora

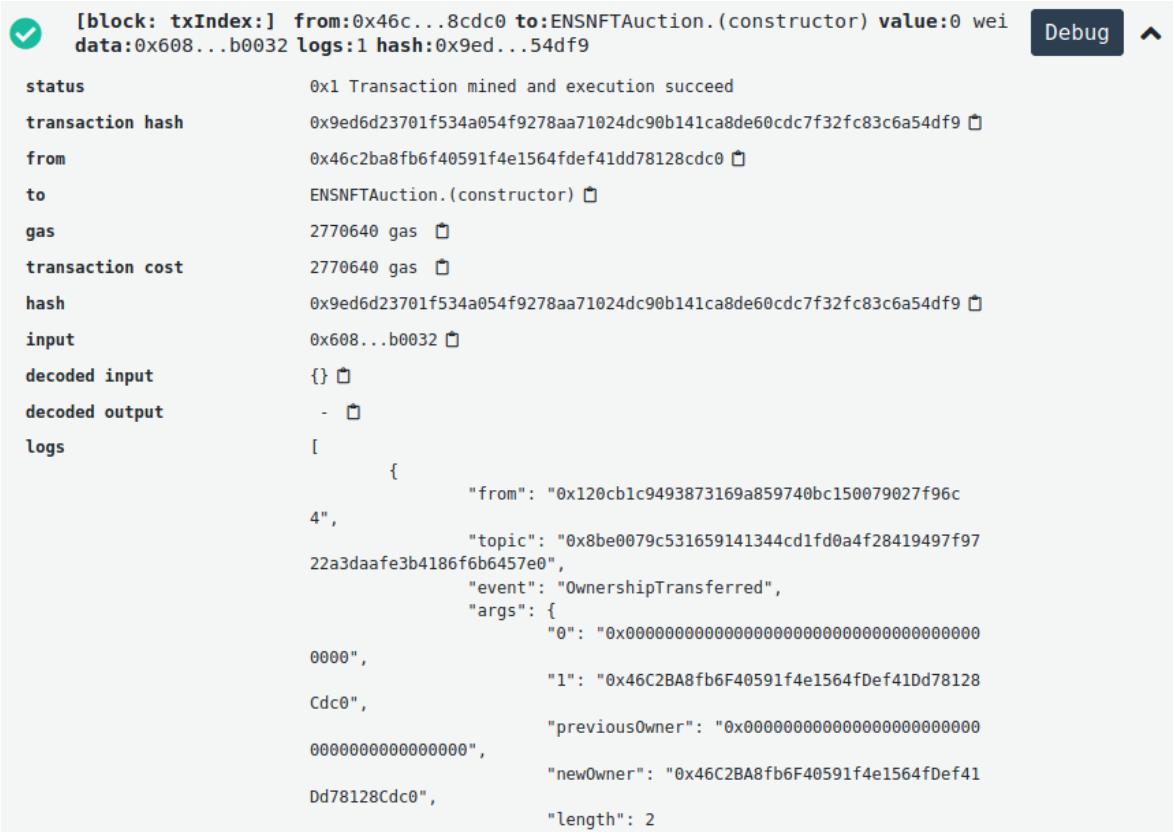
Zvanjem *OwnerOf* i *getApproved* funkcija s ID-om generiranog tokena *Registrar* ugovora moguće je vidjeti adrese vlasnika tokena i odobrenih adresa.



Slika 5-16 *getApproved* funkcija

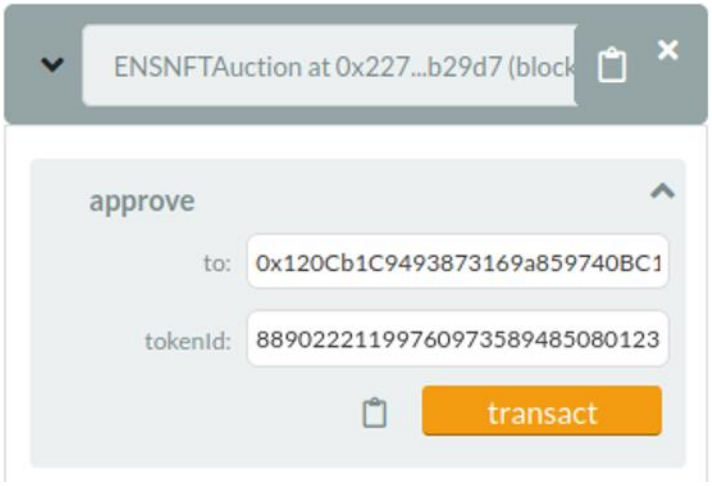
Slika 5-17 *ownerOf* funkcija

Na slici 5-18 vidljiva je transakcija koja generira ugovor za aukcije.



Slika 5-18 Transakcija generiranje ugovora

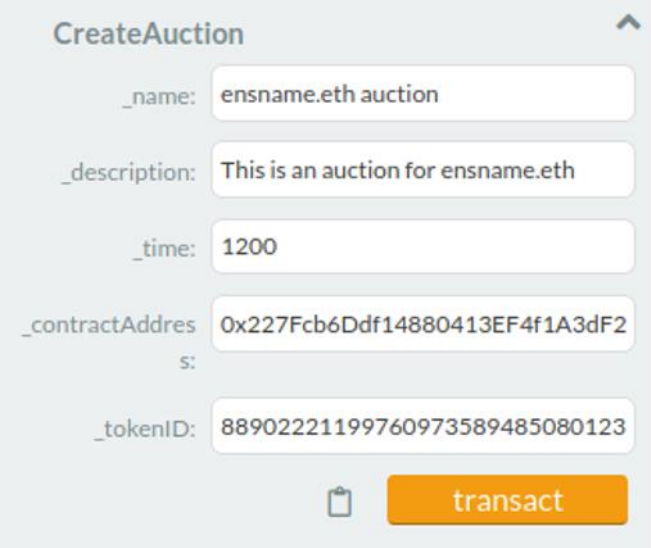
Odobravanje aukcije da upravlja tokenom radi se zvanjem *approve* funkcije od vlasnika tokena koja sadrži adresu ugovora za aukcije i ID odgovarajućeg tokena. Ovu transakciju nije moguće inicirati s ENSNFTAuction ugovora pošto samo vlasnik tokena može odobravati adrese. Transakcija mora biti inicirana direktno prema *Registrar* ugovoru.



Slika 5-19 Potvrđivanje ugovora za aukcije da upravlja s tokenom

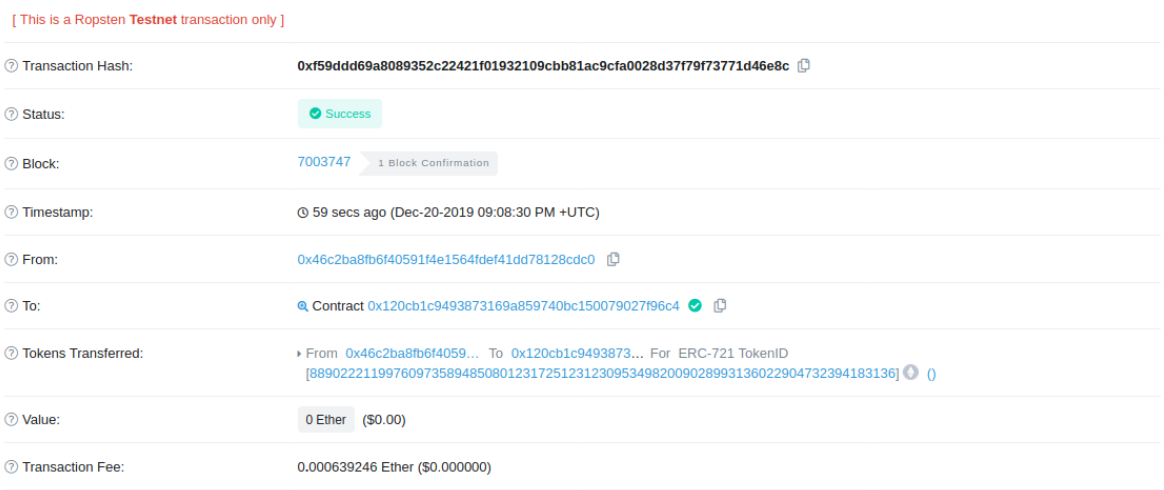
Nakon što je korisnik odobrio ugovor, može kreirati aukciju tokena. Iako je vidljivost CreateAuction funkcije definirana kao *public*, kreiranje aukcije uspjet će samo adresama

koje su vlasnici potvrđenih tokena. Prilikom kreiranja aukcije potrebno je navesti naziv aukcije, opis predmeta aukcije, vrijeme, adresu ugovora s kojeg je kreiran token i njegov ID. Uneseno vrijeme predstavlja koliko sekunda će aukcija trajati.



Slika 5-20 Kreiranje aukcije

Prilikom Kreiranja aukcije ugovor uzima vlasništvo nad tokenom i pokreće aukciju. Na slici 5-21 vidljiva je transakcija kojom ugovor preuzima vlasništvo tokena.

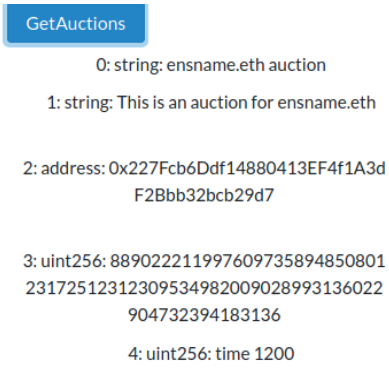


Slika 5-21 Ugovor uzima vlasništvo nad tokenom i započinje aukciju

Zvanjem ownerOf funkcije moguće je vidjeti da je vlasništvo tokena uspješno preneseno na adresu ugovora aukcije. Token će ostati na adresi ugovora sve dok aukcija ne završi.

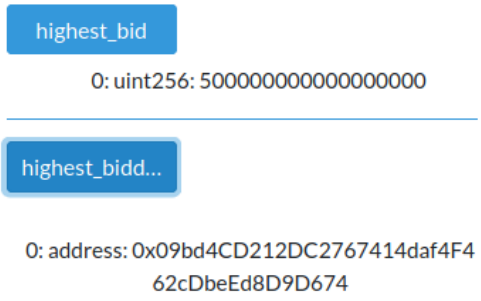
Slika 5-22 Preuzimanje vlasništva nad tokenom

Zvanjem GetAuctions funkcije potencijalni ponuđači mogu dobiti informacije o aukciji.



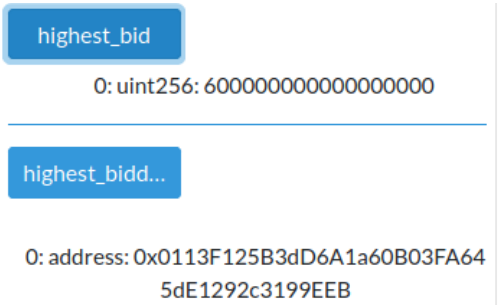
Slika 5-23 *GetAuctions* funkcija

Zvanjem Bid funkcije moguće je postaviti ponudu na aukciji. Na slici 5-24 adresa 0x09bd4CD212DC2767414daf4F462cDbeEd8D9D674 postavlja ponudu od 0.5 Ethera.



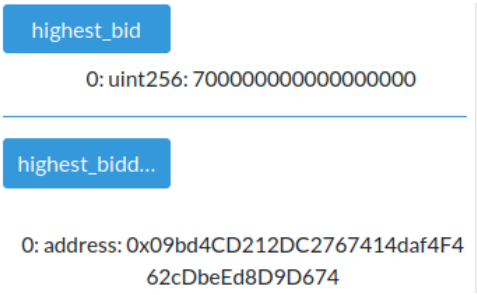
Slika 5-24 Prva ponuda

Račun 0x0113F125B3dD6A1a60B03FA645dE1292c3199EEB postavlja ponudu od 0.6 Ethera na aukciji. Nakon izvršavanja transakcije moguće je vidjeti da su se promijenile vrijednosti u highest_bidder i highest_bid varijablama da bi reflektirale novu najvišu ponudu.



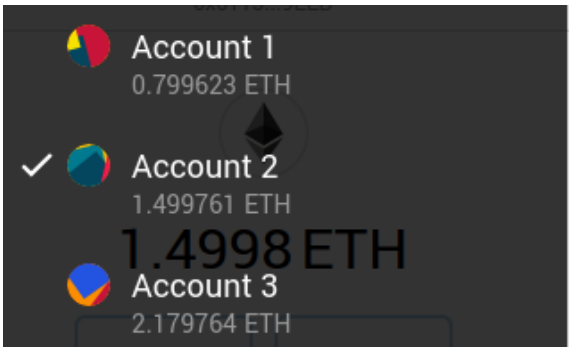
Slika 5-25 Druga ponuda

Ako korisnik koji je već postavio ponudu želi povećati ponudu može to napraviti ponovnim zvanjem Bid funkcije. Korisnik će moći postaviti novu ponudu samo ako je zbroj svih njegovih ponuda veći od trenutne najveće ponude. Na slici 5-26 korisnik koji je već postavio ponudu povećava ponudu slanjem još 0.2 Ethera u ugovor.



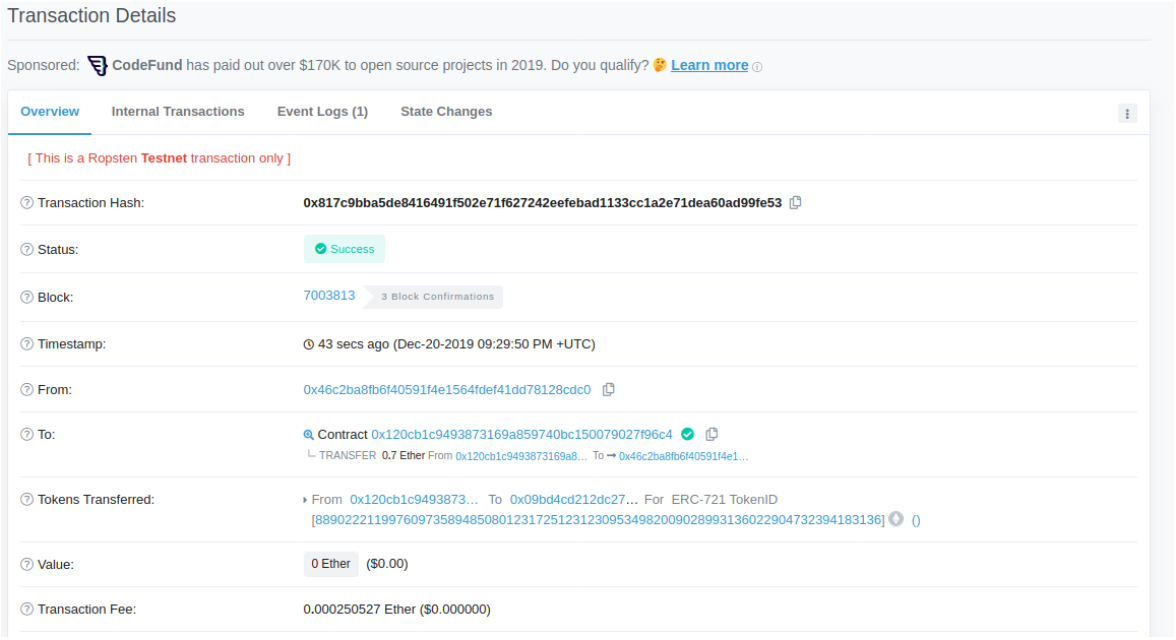
Slika 5-26 Promjena najveće ponude

Nakon isteka vremena vlasnik aukcije zove EndAuction funkciju koja isplaćuje najvišu ponudu vlasniku aukcije i šalje token odgovarajućoj adresi. Da bi se minimizirala mogućnost pojave situacija u kojima bi sredstva ostala zaključana u ugovoru, zvanje EndAuction funkcije dozvoljeno je vlasniku aukcije i pobjedniku aukcije. Sudionici aukcije koji nisu pobijedili mogu dobiti povrat fondova zvanjem Withdraw funkcije. Na slici 5-27 vidljivo je da su stanja računa prikladno promijenjena, s prvog računa oduzeto je 0.7 Ethera i dodano na treći račun, dok drugi račun nije izgubio svoj Ether.



Slika 5-27 Stanja računa nakon aukcije

Slika 5-28 pokazuje transakciju iniciranu od ugovora u kojem se vlasništvo tokena premješta na pobjedničku adresu aukcije. Transakcija također isplaćuje Ether vlasniku aukcije.



Slika 5-28 Transakcija dobivena zvanjem EndAuction funkcije

5.5.3. Testiranje kroz Truffle

Truffle omogućuje automatizirano testiranje ugovora. Testiranje se radi na lokalnoj mreži kreiranjem JavaScript test datoteke. *Block chain* za testiranje generiran je kroz Ganache alat. Test datoteka za NFT i NFTAuction ugovore nalazi se u test direktoriju pod nazivom auctiontest.js. Konfiguracijska datoteka za Truffle zove se truffle-config.js. Konfiguriranjem network parametra kao na slici 5-29, Truffle će koristiti Ganache *block chain* na lokalnoj

mreži. *Provider* sekcija u datoteci može biti konfigurirana da koristi Infura servis za objavljivanje ugovora.

```
development: {
  host: "127.0.0.1",     // Localhost (default: none)
  port: 7545,            // Standard Ethereum port (default: none)
  network_id: "*",       // Any network (default: none)
 },
```

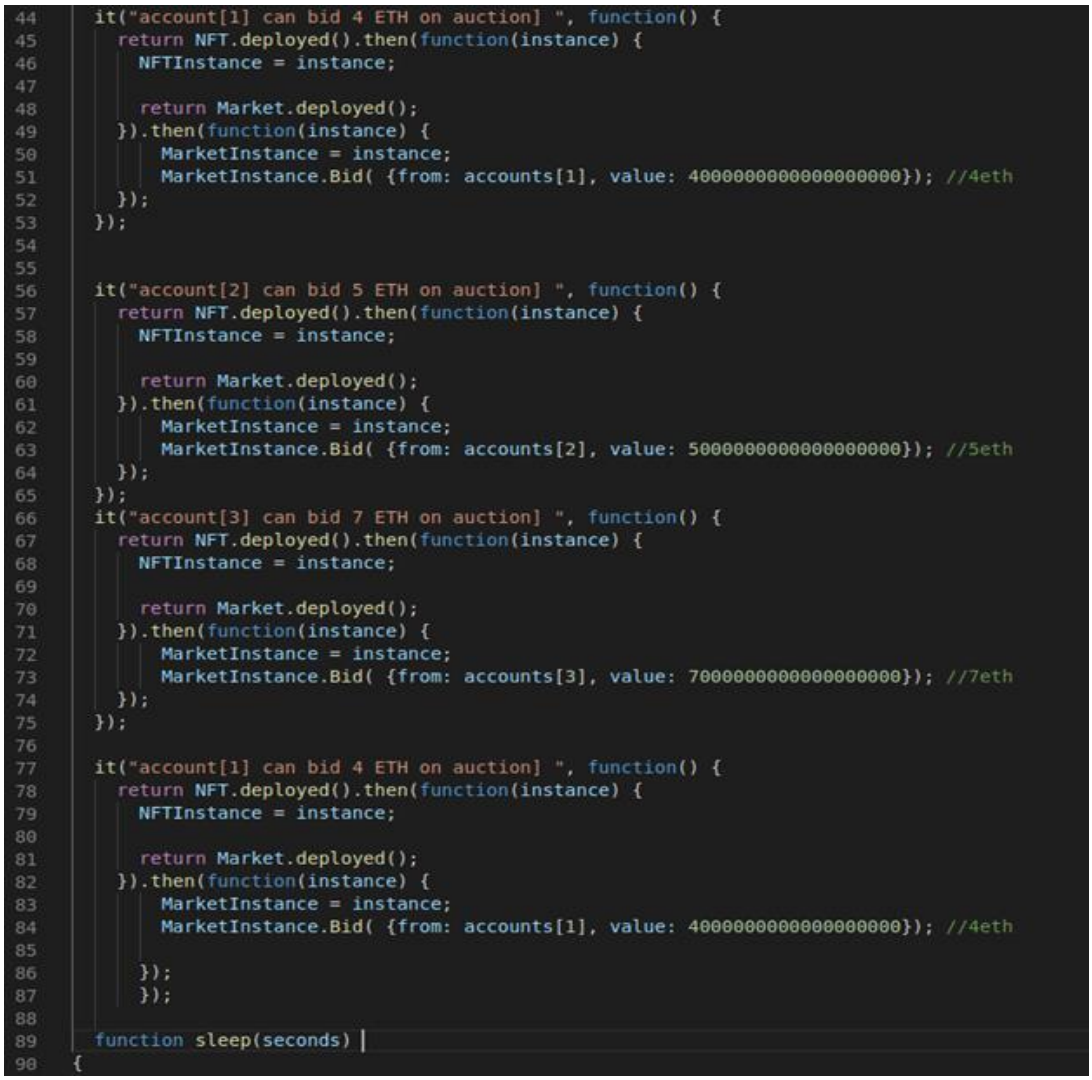
Slika 5-29 Konfiguriranje truffle-config.js

Testiranje NFT i NFTAuction ugovora sastoji se od objavljivanja token ugovora, kreiranja novog tokena, odobravanja Auction ugovora, uzimanja vlasništva tokena od adrese vlasnika aukcije, te procesa postavljanja ponuda i zaključivanja aukcije. Na slici 5-30 testirane su Mint i Approve funkcije NFT ugovora i CreateAuction funkcija NFTAuction ugovora. Nakon zvanja CreateAuction funkcije test provjerava je li vlasništvo tokena promijenjeno zvanjem ownerOf funkcije NFT ugovora koja mora vratiti adresu NFTAuction ugovora. Trajanje aukcije postavljeno je na 3 sekunde.

```
it("mint NFT tokenId 1 to account[0]", function() {
      return NFT.deployed().then(function(instance) {
        NFTInstance = instance;

        return NFTInstance.Mint('My Crypto Collectible','description', accounts[0], 1, 'metadata', {from: accounts[0]});
      })
});


it("account[0] can approve Auctioncontract ", function() {
  return NFT.deployed().then(function(instance) {
  NFTInstance = instance;

  return Market.deployed();
  }).then(function(instance) {
    MarketInstance = instance;
    return NFTInstance.approve(MarketInstance.address, 1, {from: accounts[0]});
    });
});


it("AUCTION takes ownership of account 0 NFT ID 1 and starts auction ", function() {
  return NFT.deployed().then(function(instance) {
  NFTInstance = instance;
  return Market.deployed();
  }).then(function(instance) {
    MarketInstance = instance;
    return MarketInstance.CreateAuction('MY NFT','my nft description',3, NFTInstance.address, 1 ,{from: accounts[0]});
  }).then(function(res) {
    return NFTInstance.ownerOf(1);
  }).then(function(account) {
    assert.equal(account, MarketInstance.address, "The NFT is not owned by AUCTION contract.");
  });
});
```

Slika 5-30 auctiontest.js datoteka – Kreiranje tokena i pokretanje aukcije

Testiranje postavljanja ponuda na aukciji odrađeno je postavljanjem 4 ponude. Prvi račun postavlja ponudu od 4 Ethera, drugi račun od 5 Ethera, a treći račun 7 Ethera. Da bi se testirala mogućnost povećavanja inicijalne ponude, prvi račun postavlja još jednu ponudu slanjem još 4 Ethera.



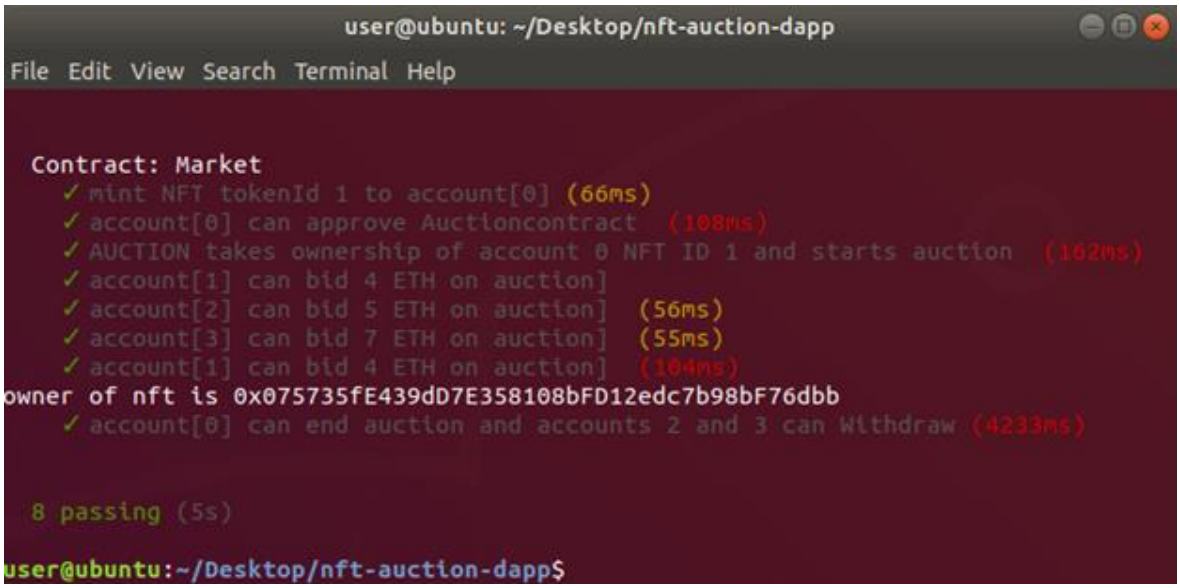
```
44    it("account[1] can bid 4 ETH on auction] ", function() {
45      return NFT.deployed().then(function(instance) {
46        NFTInstance = instance;
47
48        return Market.deployed();
49      }).then(function(instance) {
50          MarketInstance = instance;
51          MarketInstance.Bid( {from: accounts[1], value: 4000000000000000000}); //4eth
52      });
53    });
54
55
56    it("account[2] can bid 5 ETH on auction] ", function() {
57      return NFT.deployed().then(function(instance) {
58        NFTInstance = instance;
59
60        return Market.deployed();
61      }).then(function(instance) {
62          MarketInstance = instance;
63          MarketInstance.Bid( {from: accounts[2], value: 5000000000000000000}); //5eth
64      });
65    });
66    it("account[3] can bid 7 ETH on auction] ", function() {
67      return NFT.deployed().then(function(instance) {
68        NFTInstance = instance;
69
70        return Market.deployed();
71      }).then(function(instance) {
72          MarketInstance = instance;
73          MarketInstance.Bid( {from: accounts[3], value: 7000000000000000000}); //7eth
74      });
75    });
76
77    it("account[1] can bid 4 ETH on auction] ", function() {
78      return NFT.deployed().then(function(instance) {
79        NFTInstance = instance;
80
81        return Market.deployed();
82      }).then(function(instance) {
83          MarketInstance = instance;
84          MarketInstance.Bid( {from: accounts[1], value: 4000000000000000000}); //4eth
85
86        });
87        });
88
89    function sleep(seconds)
90  {
```

Slika 5-31 auctiontest.js datoteka – Testiranje postavljanja ponuda

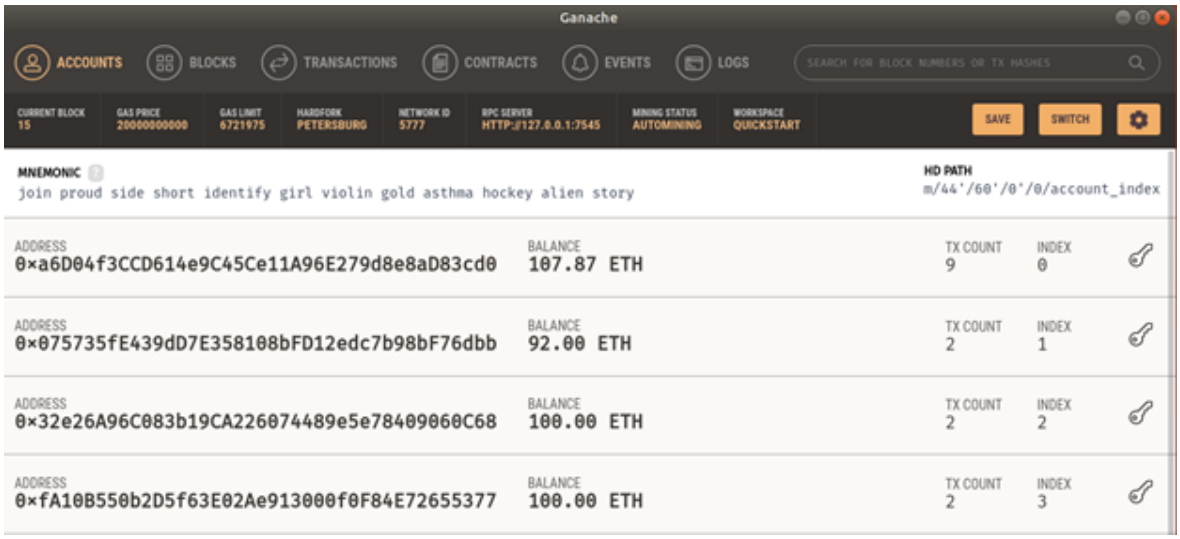
Testiranje EndAuction i Withdraw funkcija odrađuje se nakon zvanja Sleep funkcije na 4 sekunde. EndAuction funkciju zove adresa vlasnika aukcije te rezultira slanjem tokena na adresu najviše ponude te slanjem najviše ponude vlasniku aukcije. Nakon završetka aukcije, računi 2 i 3 mogu zvati Withdraw funkciju koja im vraća ponude postavljene tijekom aukcije. Zvanjem ownerOf funkcije moguće je potvrditi je li se stvarno promijenilo vlasništvo tokena. Radi jednostavnosti kod testiranja, adresa vlasnika ispisuje biti će ispisana i bacit će grešku u slučaju ako vlasništvo tokena nije promijenjeno na odgovarajuću adresu.

Zvanjem Truffle test komande ugovori će biti migrirani na Ganache te će se pokrenut niz akcija specificiranih u auctiontest.js datoteci. Provjeravanjem stanja računa u Ganache alatu moguće je potvrditi je li se aukcija izvršila na pravilan način. Iz slika 5-32 i 5-33 vidljivo je da su svi testovi uspješno izvršeni te da se vlasništvo tokena promijenilo kao i stanja računa. Račun vlasnika aukcije s indeks brojem 0 dobio je 8 Ethera koliko je bila najviša ponuda,

dok su računi pod indeks brojem 2 i 3 mogli uspješno zvati Withdraw funkciju te dobiti povrat uloga. Uspoređivanjem adrese vlasnika tokena nakon završetka aukcije, dane prilikom izvršavanja testa i adrese koja je konfigurirana da postavi najveću ponudu moguće je potvrditi da odgovarajuća adresa posjeduje token.



Slika 5-32 Izvršavanje testa



Slika 5-33 Promjena stanja računa u Ganache alatu

6. Primjena pametnih ugovora

6.1. Decentralizirane aplikacije

Decentralizirane aplikacije su *open-source* aplikacije koje se izvršavaju na distribuiranim računalnim sustavima kao što je Ethereum. Funkcionalnost ovih aplikacija ostvarena je kroz pametne ugovore, što znači da korisnici mogu sami provjeriti kod ugovora i biti sigurni u uslugu koju aplikacija pruža. Za razliku od tradicionalnih aplikacija pokretanih s centraliziranih sustava, decentralizirane aplikacije nude prednosti u smislu troškova održavanja i velike razine dostupnosti. Za interakciju s decentraliziranim aplikacijama korisnici ne moraju stvarati račune već mogu koristiti postojeće račune koje kontrolira njihov par ključeva.

Jedna vrsta decentraliziranih aplikacija na Ethereumu su DEX ili decentralizirane mjenjačnice. Ovaj tip mjenjačnica omogućuje korisnicima da trguju s kriptovalutama i tokenima bez potrebe za posrednicima za održavanje stanja ili kontroliranje korisničkih sredstva te omogućuju korisnicima privatnost zbog činjenice da se od korisnika zahtijeva otkrivanje samo svoje javne adrese, za razliku od centraliziranih mjenjačnica sa skrbništvom sredstva, koje često zahtijevaju velike količine osobnih informacija kako bi bile u skladu sa zakonima država iz kojih nude usluge. Zbog činjenice da ne postoji ni jedna centralna točka u DEX sustavu, ove sustave je hakerima puno teže napasti te nude bolje garancije dostupnosti. Neke od poznatijih DEX mjenjačnica su Kyber network, EtherDelta, Uniswap i 0x.

6.2. *Internet of Things*

IoT je relativno nov i brzo rastući koncept povezivanja velike količine raznih svakodnevnih uređaja na Internet. Broj *IoT* uređaja raste velikom brzinom te napredak u tehnologijama kao što su AI i *block chain* mogu imati znatan utjecaj na razvoj *IoT* ekosustava. Istraživačka i savjetodavna kompanija Gartner procjenjuje da će 2020. godine 5.8 milijardi *IoT* krajnjih točaka biti spojeno na Internet. Najveći korisnici *IoT* uređaja trenutno su kompanije distributeri električne energije i gradski podsustavi koji uključuju napajanje električnom energijom, vodoopskrbu, upravljanje prometom i gradskim uslugama. Gartner također

predviđa rast komercijalnih *IoT* tržišta kao što su automatizacija zgrada koristeći pametne rasvjetne uređaje kao i automotivne industrije.

Opseg informacija povezanih sa *IoT* uređajima može biti velik i predstavljati veliki problem kod upravljanja pravima pristupa osjetljivim informacijama. Jedan od najvećih problema postojećih *IoT* sustava je njihova sigurnosna arhitektura s centraliziranim klijent-poslužitelj modelom kojim upravlja jedna stranka što ga čini osjetljivim na jednu točku neuspjeha. *Block chain* sustavi mogu pružiti rješenje ovog problema decentraliziranjem procesa odlučivanja kroz konsenzus mehanizam na zajedničku mrežu uređaja.

6.3. Decentralizirane financije

DeFi predstavlja ekosustav decentraliziranih aplikacija na javnim *block chain* mrežama kojima je cilj rekreiranje tradicionalnih financijskih instrumenta kao što su kreditiranje i zaduživanje. Javni *block chainovi* s programibilnim pametnim ugovorima najčešće su korišteni za izradu ovakvih aplikacija i omogućuju *peer-to-peer* tržište za financijske usluge.

Za razliku od koncepta otvorenih financija u kojima razni *startupovi* kroz usluge bazirane na API-jima nude korisnicima bankarske usluge, De-Fi usluge funkcioniraju na transparentan način te mogu biti otporne na cenzuru i ne zahtijevaju usluge skrbništva sredstva. Zbog činjenice da se veliki dio aplikacijskog *stacka* „*open-finance*“ aplikacija bazira na strogo reguliranim bankarskim sustavima, ove vrste aplikacija često zahtijevaju stroge KYC/AML procedure za korisnike, te su geografski ograničene.

Projekti koji spadaju pod definiciju decentraliziranih financija zahtijevaju sljedeće atribute:

- otpornost na cenzuru – skrbništvo nad imovinom i njihova razmjena ne smije biti ograničena od nekolicine sudionika zaduženih za upravljanje mrežom
- pseudo anonimnost – aplikacije ne zahtijevaju od korisnika KYC/AML procedure i koriste standarde web 3.0 tehnologija za autentifikaciju i transakcije
- transparentnost i minimizacija povjerenja – skrbništvo nad imovinom radi se kroz *open-source* pametne ugovore.

6.3.1. MakerDAO

MakerDAO je decentralizirana organizacija i ekosustav izgrađen s pametnim ugovorima na Ethereumu. DAI je kriptovaluta koja održava konstantnu vrijednost tako da 1 DAI token stalno drži vrijednost 1 $ USD.Za razliku od drugih takozvanih *stablecoin* kriptovaluta koje održavaju vrijednosti koristeći skrbničke usluge treće stranke, vrijednost DAI tokena bazirana je na Etheru zaključanom u pametnom ugovoru.

Rizik kod *stablecoin* projekta koji se oslanjaju na treću stranku za čuvanje financijskih sredstava potrebnih za održavanje 1:1 omjera s tokenom u cirkulaciji leži u činjenici da ta stranka može iz bilo kojeg razloga biti spriječena ili odbiti pristup sredstvima što fundamentalno stavlja ove projekte u rizik. Godine 2019. poznata mjenjačnica kriptovaluta Bitfinex pod istim vlasništvom kao stablecoin projekt Tether, našla se u sličnoj situaciji kada je pravnički ured u New Yorku optužio Bitfinex zbog korištenja 850 $ milijuna fondova iz Tether rezerva kako bi se zataškali gubitci sredstva. Crypto Capital, organizacija zadužena za procesiranje fiat transakcija za korisnike Bitfinex mjenjačnice, odbila je procesirati transakcije jer su im sredstva bila blokirana od regulatornih agencija.

Maker je trenutno najuspješniji De-Fi projekt sa više od 2 % Ethera u cirkulaciji zaključano u pametnom ugovoru. Koristeći Maker korisnici mogu zaključati Ether u ugovor i stvoriti CDP (*collateral debt position*) ili kolaterizirani zajam, što znači da mogu dobiti određena sredstva u obliku DAI kriptovalute na temelju pologa. Polog služi kao kolateralna imovina u slučaju ako zajam nije vraćen.

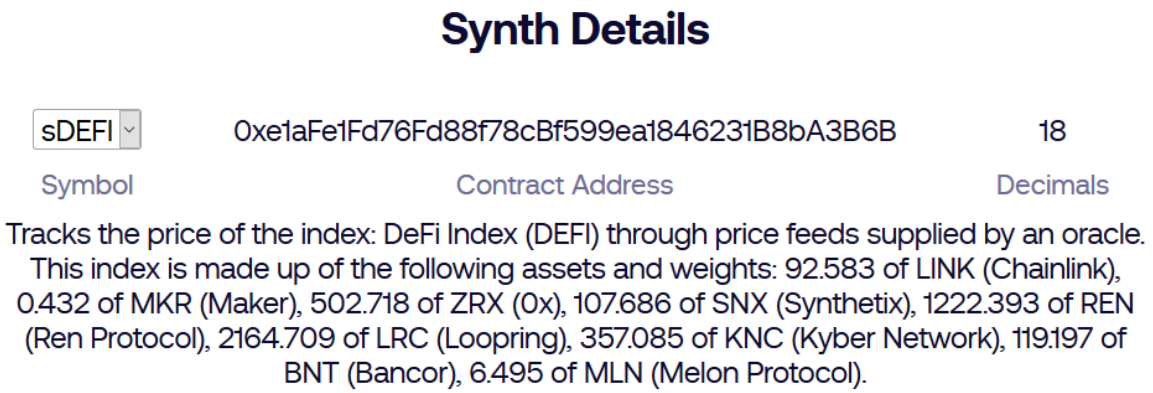
6.3.2. *Compound Finance*

Compound Finance je DeFi protokol dizajniran da služi kao transparentno tržište za ostvarivanje zajmova i zarađivanje kamata kroz pametne ugovore na Ethereumu. *Compound* trenutno podržava 5 tržišta (BAT, DAI, WETH, ZRX, REP). Kamatne stope izračunavaju se na temelju likvidnosti za svako od tržišta u stvarnom vremenu. Na ovaj način korisnici mogu ostvariti zajmove bez posrednika.

6.3.3. Synthetix

Synthetix je platforma za stvaranje sintetičkih *on-chain* sredstva. Ova sintetička sredstva predstavljaju ERC20 tokeni i omogućuju korisnicima izlaganje tržištima bez potrebe za posjedovanjem tog sredstva. SNX token koristi se kao kolateralna vrijednost za sintetička

sredstva kako bi korisnici mogli stvarati svoje sintetičke tokene. Synthetix koristi Chainlink *oracle* mrežu za decentralizirano dobavljanje *off-chain* informacija o cijenama *forex* valuta.



Slika 6-1 Synthetix sDEFI

6.4. Upravljanje identitetom

Upravljanje identitetom je metoda korištena za određivanje identiteta individualnih korisnika neke aplikacije ili usluge. Koncept pametnog identiteta u *block chain* terminologiji odnosi se na jedinstveni ID koji korisnik može sam stvoriti i upravljati s njim te se oslanja na koncept da vlasnik identiteta ima potpunu kontrolu i autonomiju nad svojim identitetom i njegovim atributima. Tradicionalni alati za upravljanje identitetom oslanjaju se na centralizirane usluge kod kojih su treće stranke zadužene za upravljanje identitetima korisnika.

Sa sazrijevanjem *block chain* tehnologija pojavljuje se potreba za korištenjem *Zero-Knowledge Proof* metoda autentifikacije za upravljanje digitalnim identitetima. ZKP sustavi omogućuju dokazivanje da neka stranka posjeduje određenu informaciju bez odavanja ikakvih podataka koje bi mogle otkriti tu informaciju. *Block chain* sustavi koji podržavaju ZKP protokol mogu omogućiti korisnicima da potvrde svoj identitet bez odavanja ikakvih specifičnih informacija o identitetu.

Zaključak

Umjesto da vjeruju ugovornoj stranci ili se oslanjaju na treću stranku za ručnu verifikaciju ugovornih obaveza, pametni ugovori pretvaraju uvjete ugovora u kod koji se izvršava na temelju unosa podataka. Za velike organizacije, pametni ugovori daju jedno od najboljih rješenja za automatizaciju ugovornih procesa poslovanja. Istraživačka i savjetodavna organizacija Gartner procjenjuje da će pametni ugovori postati popularniji i početi utjecati na globalna tržišta do 2022. godine uz pretpostavku da će 25 % organizacija koristiti pametne ugovore u poslovanju.

Popis kratica

ATM *Asynchronous Transfer Mode*
ISDN *Integrated Services Digital Network*
EIP *Ethereum Improvement Proposal*
EEA *Ethereum Enterprise Alliance*
API *Application Programming Interface*
PoW *Proof of Work*
PoS *Proof of Stake*
DPoS *Delegated Proof of Stake*
ProgPoW *Programmatic Proof of Work*
EVM *Ethereum Virtual Machine*
WETH *Wrapped Ether*
ASIC *Application Specific Integrated Circuit*
DOS *Denial of Service*
TCF *Trusted Compute Specification*
TEE *Trusted Execution Environment*
FCFS *First Come First Serve*
DNS *Domain Name Service*
ENS *Ethereum Name Service*
ERC *Ethereum Request for Comments*
TLD *Top Level Domain*
DE-FI *Decentralized Finance*
DAO *Decentralized Autonomous Organisation*
CDP *Collaterlized Debt Position*
ZKP *Zero Knowledge Proof*
GPU *Graphics Processing Uint*
BFT *Byzantine* Fault Tolerance
UTXO *Unspent Transaction Output*
VM *Virtual Machine*
ICO *Initial Coin Offering*
AI *Artificial Intelligence*
KYC *Know Your Customer*
AML *Anti Money Laundering*

Popis slika

Literatura

Svaki autor piše popis literature na kraju rada. Popis literature se piše stilom literatura.

[1] VITALIK BUTERIN, https://github.com/ethereum/wiki/wiki/white-paper, svibanj 2019.

[2] ANDREAS M. ANTONOPOULOS, GAVIN WOOD, https://github.com/ethereumbook/ethereumbook, veljača 2020.

[3] STEVE ELLIS, ARI JUELS, SERGEY NAZAROV, https://link.smartcontract.com/whitepaper, rujan 2017.

[4] NICK SULLIVAN, https://blog.cloudflare.com/ecdsa-the-digital-signature-algorithm-of-a-better-internet/, listopad 2014.

[5] SVETLIN NAKOV, https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages, svibanj 2019.

[6] PHILIP DAIAN, ARI JUELS, STEVEN GOLDFEDER, https://arxiv.org/pdf/1904.05234.pdf, travanj 2019.

[7] ALIBABA CLOUD, https://medium.com/datadriveninvestor/from-distributed-consensus-algorithms-to-the-blockchain-consensus-mechanism-75ee036abb65, listopad 2019.

[8] PARIKSHIT HOODA, https://www.geeksforgeeks.org/proof-of-work-pow-consensus/

[9] EDCHAIN, https://medium.com/@EdChain/pow-vs-pos-a-comparison-of-two-blockchain-consensus-algorithms-f3effdae55f5, lipanj 2018.

[10] CAMMILA RUSSO, https://thedefiant.substack.com/p/a-faulty-oracle-will-be-behind-the, listopad 2019.

[11] JAKE FRANKENFIELD, https://www.investopedia.com/terms/1/51-attack.asp, svibanj 2019.

[12] STACKEXCHANGE,https://ethereum.stackexchange.com/questions/58535/understanding-the-state-transition-function, veljača 2020

[13] VAIBHAV SAINI https://hackernoon.com/getting-deep-into-evm-how-ethereum-works-backstage-ac7efa1f0015, kolovoz 2018.

[14] Colin Schwarz, https://medium.com/chainsafe-systems/ethereum-2-0-a-complete-guide-casper-and-the-beacon-chain-be95129fc6c1, kolovoz 2019.

[15] OPENZEPPELIN, https://docs.openzeppelin.com/openzeppelin/, veljača 2020.

[16] TRUFFLE SUITE, https://www.trufflesuite.com/docs, veljača 2020.

[17] SANJAY BAKSHI, YEVGENIY (EUGENE) YARMOSH, LEI ZHANG, https://entethalliance.org/wp-content/uploads/2019/10/EEA_Off-Chain_Trusted_Compute_Specification_v1.1.pdf, listopad 2019.

[18] DEFI PUSLE, https://defipulse.com/defi-list/, veljača 2020

[19] MAKER TEAM, https://makerdao.com/whitepaper/ , prosinac 2017.

[20] ROBERT LESHNER, GEOFFREY HAYES, https://compound.finance/documents/Compound.Whitepaper.pdf, veljača 2019.

[21] SYNTHETIX, https://www.synthetix.io/uploads/synthetix_litepaper.pdf, prosinac 2019.

[22] JOEL JOHN, LAWRENCE LUNDY-BRYAN, https://outlierventures.io/wp-content/uploads/2019/06/Mapping-Decentralised-Finance-DeFi-report.pdf, veljača 2020.

[23] FRANGIO, https://forum.openzeppelin.com/t/how-to-implement-erc20-supply-mechanisms/226, siječanj 2019.

[24] ARTEM SHUROV, DANIIL MALEVANNIY, OLEG IAKUSHKIN, AND VLADIMIR KORKHOV, Blockchain Network Threats: The Case of PoW and Ethereum

ALGEBRA
VISOKO
UČILIŠTE

NASLOV ZAVRŠNOG RADA

Pristupnik: Hrvoje Horvat, JMBAG

Mentor: Zlatan Morić