VERIFIKACIJA IZDANIH DIPLOMA NA BLOCKCHAINU

Perica, Marko

Undergraduate thesis / Završni rad

2019

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:724616

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-23**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Sveučilišni studij računalstva

VERIFIKACIJA IZDANIH DIPLOMA NA BLOCKCHAINU

Završni rad

Marko Perica

Osijek, 2019.

SADRŽAJ

1. UVOD ... 1

1.1. Zadatak završnog rada ... 1

2. TEHNOLOGIJA BLOCKCHAIN(LANAC BLOKOVA) ... 2

2.1. Hashing ... 2

2.2. Kriptografija javnog ključa ... 3

2.3. Peer-to-peer mreže ... 3

2.4. Merkle stabla ... 4

2.5. Postizanje konsenzusa ... 5

2.5.1. Problem bizantskih generala ... 6

2.5.2. Dokaz o radu(*proof-of-work*) ... 6

2.5.3. Dokaz o udjelu(*proof-of-stake*) ... 7

3. PREGLED KORIŠTENIH TEHNOLOGIJA I ALATA ... 9

3.1. IPFS (InterPlanetary File System) ... 9

3.2. Ethereum ... 10

3.2.1. Solidity ... 11

3.3. MetaMask ... 12

3.4. Ethereum Testnet ... 12

3.4.1. Javni testnet ... 13

Ropsten ... 13

Kovan ... 14

Rinkeby ... 14

4. PRIMJENA BLOCKCHAIN-A ZA VERIFIKACIJU DIPLOMA ... 15

4.1. Dizajn rješenja ... 15

4.2. Postavljanje diplome na IPFS ... 16

4.3. Postavljanje podataka na blockchain ... 19

4.4. Provjera valjanosti diplome ... 21

5. ZAKLJUČAK ... 23

SAŽETAK ... 26

ABSTRACT ... 27

ŽIVOTOPIS ... 28

1. UVOD

Tema ovog završnog rada je verifikacija diploma izdanih pomoću tehnologije blockchain-a. Neki obrazovni centri dopuštaju potvrdu vjerodostojnosti svojih diploma jednostavnim on-line upitom bez obzira na to tko zahtjeva tu informaciju. Drugi daju ovlaštenje nekoj trećoj strani (bilo svojevoljno ili zbog regulacija koje to zahtijevaju). U nekim slučajevima ne postoji alternativa izravnom kontaktiranju tajnika/-ice obrazovne institucije u svrhu potvrde vjerodostojnosti diplome. U međuvremenu krivotvorene diplome obrazovnih institucija su stvarnost i o slučajevima kada je to otkriveno nažalost često imamo priliku slušati ili čitati u medijima. Dio krivnje zbog toga leži i na nadležnim tijelima samih obrazovnih centara. Učestalost takvih događaja je dovoljna za pojavu tvrtki čija je zadaća otkrivanje takvih slučajeva. Svojim modelom suglasnosti blockchain tehnologija može stvoriti nepromjenjivu obrazovnu arhivu kojoj se može vjerovati u svrhu verifikacije diploma. U ovom završnom radu pružit ću uvid u teorijsku podlogu koja čini blockchain tehnologiju prikladnu za ovakav zadatak i implementirati jedno takvo rješenje koristeći Ethereum blockchain i IPFS - sustav za distribuirano pohranu hipermedijskih datoteka.

1.1. Zadatak završnog rada

U završnom radu treba objasniti probleme i predstaviti pregled dosadašnjih rješenja za verifikaciju izdanih diploma od strane ustanove. Također treba objasniti prednosti korištenja blokchaina i dati riješenje za verifikaciju diploma na jednom od javnih blockchainova.

2. TEHNOLOGIJA BLOCKCHAIN(LANAC BLOKOVA)

U ovom poglavlju će ukratko biti objašnjene neke od najvažnijih tehnologija i sustava na kojima počiva blockchain.

2.1. Hashing

Hashing je u suštini čin u kojem se koristi hashing funkcija kako bi dobili hashirani izlaz. Primjeri popularnih hashing funkcija su SHA265, MD5, Bcrypt i RIPEMD. Hashing funkcija na ulazu može imati argument bilo koje veličine(adresa elektronske pošte, lozinka, cijeli tekst *Zločina i kazne*...), dok je izlaz fiksne duljine.

Na primjer, hashiranjem rečenice sljedeće rečenice: „Očekuje se velik broj posjetitelja“ (bez točke) SHA-256 funkcijom dobivamo sljedeći rezultat:

Slika 2.1 *Izlaz za prvu vrijednost*

Hashiranjem tek neznatno preinačene poruke: „Očekuje se velik broj posjetitelja.“ (s točkom) dobivamo sljedeći rezultat:

Slika 2.2 *Izlaz za drugu vrijednost*

Hashiranje za danu ulaznu vrijednost lako je izračunati izlaznu vrijedost, ali je praktički nemoguće s izlaznom vrijednosti dobiti ulaznu vrijednost. Stoga se *hash* funkcije nazivaju i jednosmjerne funkcije. *Hashing* funkcije se ne mogu koristiti za šifriranje i dešifriranje (zbog njihove jednosmjerne prirode). Kriptografija i *hashiranje* su dvije različite stvari, ali mogu postojati primjene u kojima se mogu kombinirati(kao što je kriptografija javnog ključa). U teoriji je moguće da dva različita ulaza proizvedu jednak izlaz. Takav slučaj naziva se kolizija.

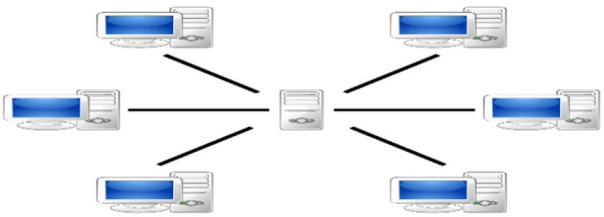
2.2. Kriptografija javnog ključa

Kriptografija javnog ključa, ili asimetrična kriptografija, je kriptografski sustav koji koristi par ključeva: javni ključevi koji se mogu distribuirati javno, i privatni ključevi koji su poznati samo vlasniku. Šifriranje i dešifriranje obavljaju se asimetričnim algoritmima koji su definirani tako da koriste par ključeva od koji se bilo koji može koristi za šifriranje. Ako se koristi jedan ključ iz para za šifriranje, onda se drugi ključ iz para koristi za dešifriranje. Obično se kriptiranje obavlja javnim ključevima, a dekriptiranje tajnima, tako poruku može dekriptirati samo vlasnik tajnog ključa. Osnovna sigurnosna pretpostavka je kriptografije javnog ključa je da je (praktički nemoguće) izvesti tajni privatni ključ iz poznatog javnog ključa. Neke od praktičnih primjena kriptografije javnog ključa su šifriranje poruka, digitalno potpisivanje i dogovaranje ključa koji se kasnije koristi u nekom simetričnom kriptosustavu.

Digitalni potpisi se za razliku od „običnih“ potpisa razlikuju po tome što se mijenja za različite poruke. Čak i mala preinaka poruke u potpunosti mijenja digitalni potpis ili formalno: produciranje potpisa uključuje funkciju koja ovisi i o samoj poruci i o privatnom ključu. Privatni ključ osigurava da samo vlasnik privatnog ključa producira određeni potpis, a činjenica što ovisi o poruci znači da nitko ne može jednostavno kopirati jedan od potpisa i krivotvoriti ga na nekoj drugoj poruci. Osim funkcije za digitalno potpisivanje mora postojati i funkcija koja potvrđuje valjanost potpisa i ovdje javni ključ postaje važan faktor, ova funkcija vraća *istinu* ili *laž* koja ukazuje na to je li potpis produciran privatnim ključem koji je povezan s javnim ključem.

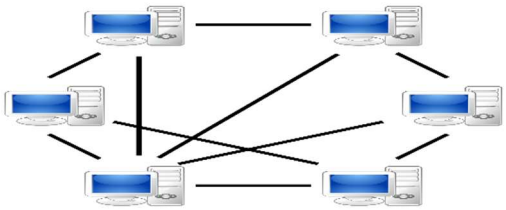
2.3. Peer-to-peer mreže

Pojam „peer-to peer“ označava na koji način se informacije unutar mreže razmjenjuju. Razlikuje se od modela klijent-poslužitelj u kojem klijenti komuniciraju s poslužiteljem koji je jedinstven.



Slika 2.3 *Prikaz client-server arhitekture*

Kod peer-to peer mreže klijenti komuniciraju međusobno, a poslužitelj više nije središte preko kojeg se vrši sva izmjena podataka, već je ostalim članovima mreže omogućena međusobna komunikacija i razmjena podataka.

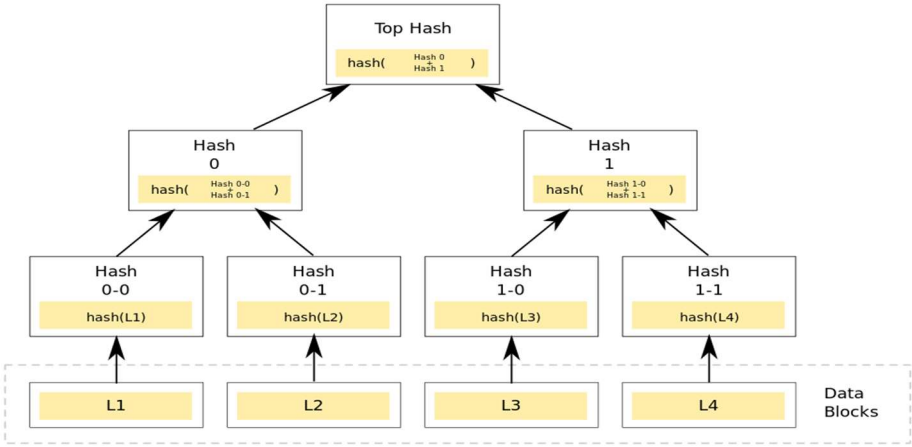


Slika 2.4 *Prikaz peer-to-peer arhitekture*

2.4. Merkle stabla

Merkle stabla su jedna od temeljnih sastavnica *blockchain*-a. Omogućuju učinkovitu i sigurnu verifikaciju velikih podatkovnih struktura, i u slučaju *blockchain*-a, potencijalno neograničene skupove podataka. Implementacija Merkle stabala u *blockchain*-ima višestruki učinak. Omogućuje im skaliranje dok u isto vrijeme pruža arhitekturu temeljenu na *hashe*-vima za odražavanje integriteta podataka i trivijalan način za verifikaciju integriteta podataka. *Blockchain* je u suštini povezana lista koja sadrži *hash* pokazivač koji pokazuje na prethodni blok, tako stvarajući lanac takvih blokova odakle dolazi i naziv blockchain (lanac blokova). Svaki blok je povezan sa svim drugim preko *hash* pokazivača(eng. *pointera*), koji je zapravo hash podataka unutar prethodnog bloka zajedno s adresama prethodnih blokova. Povezivanjem blokova u ovom obliku, svaki rezultirajući *hash* prethodnog bloka predstavlja cijelo stanje *blockchain*-a budući da su svi hashirani podaci prethodnog bloka *hashirani* u jedan *hash*. Iako je moguće koristiti kriptografske *hasheve* bez Merkle stabala, to je ekstremno neučinkovita i nije skalabilno.

Merkle stabla su u suštini stabla gdje je svaki čvor koji nije list zapravo *hash* čvorova koja su njegova djeca. Čvorovi listovi su najniža razina čvorova u stablu. Na vrhu stabla nalazi se *hash* cijelog stabla koji se naziva korijenski *hash*. Merkle stabla su podatkovna struktura koja može predstaviti n broj hasheva sa samo jednim hash-em. Struktura stabla dopušta učinkovito mapiranje proizvoljno velikih količina podataka i omogućuje jednostavnu identifikaciju gdje se promjene podataka događaju. Ovakav koncept omogućuje Merkle dokaze, s kojima se može verificirati da je *hashiranje* podataka konzistentno skroz do vrha i na ispravnom položaju bez da zapravo uzima u obzir cijeli skup *hasheva*. Umjesto toga, može se verificirati da je podatkovni blok konzistentan s korijenom *hasha* provjerom samo malog podskupa *hasheva* umjesto cijelog skupa podataka.



Slika 2.5 *Prikaz binarnog hash stabla*

Jedna od najvažnijih prednosti strukture Merkle stabla je mogućnost autentifikacije proizvoljno velikih skupova podataka sličnim *hashing* mehanizmom koji se koristi za verifikaciju mnogo manjih količina podataka. Stablo je korisno kod distribucije velikih skupova podataka u praktične manje dijelove gdje je barijera za verifikaciju integriteta podataka značajno smanjena u odnosu na ukupnu veličinu podataka.

Ethereum, trenutno drugi najpopularniji *blockchain* ima vlastiti način implementacije Merkle stabla. Budući da je Ethereum kao platforma *Turing potpuna*[1] i može poslužiti za izgradnju mnogo složenijih aplikacija, ona koristi složeniju verziju Merkle stabla zvanu Merkle Patricia Stabla. To su zapravo 3 odvojena Merkle stabla koja se koriste za tri vrste objekata.

2.5. Postizanje konsenzusa

Postizanje konsenzusa je grupni proces donošenja odluka u kojem članovi grupa iznose i na kraju slože oko odluke koja je u interesu cjeline. U jednostavnijim terminima, postizanje konsenzusa je dinamičan način postizanja dogovara u grupi. Metoda kojom se donosi konsenzus nazivaju se mehanizam konsenzusa(eng. *consensus mechanism*) Zbog toga što je *blockchain* decentralizirani *peer-to-peer* sustav potrebne su mu metode kojima postiže suglasnost ili konsenzusne metode.

Prije bitcoina koji je prvi blockchain, postojalo je više iteracija *peer-to-peer* decentraliziranih valutnih sustava koji nisu uspjeli jer nisu bili u mogućnosti pravilno adresirati najveći problem – problem bizantskih generala.

[1] *Turing potpunost* – mogućnost rješavanja svakog računskog problema, ali bez garancije u kojem vremenskom roku.

2.5.1. Problem bizantskih generala

Problem se svodi na dva generala koji opsjedaju grad sa suprotnih strana, i pokušavaju koordinirati napad. Ako general A pošalje poruku koja kaže „napadamo sutra u podne“, on ne može znati hoće li general B primiti točnu poruku, i potencijalno može marširati u smrt ako napadne bez drugog generala. Kada primi poruku, general B nema ideju je li poruka autentična ili je poslana od strane neprijatelja kako bi ga odvukli u zamku. Kako god, on će pretpostaviti autentičnost i poslati odgovor potvrđujući napad, ali bez znanja je li general A primio odgovor, može biti u strahu da će drugi general odlagati napad, što će rezultirati time da će general B biti onaj koji će sutra napadati sam u podne što znači sigurnu smrt. General A bi, naravno, mogao poslati poruku koja potvrđuje potvrdu Generala B, ali nikada neće sa sigurnošću znati je li ona došla do odredišta, ili čak je li poruka autentična. Ovo ga stavlja u isti položaj u kojem je bio General B. Ovaj problem se vraća od jednog do drugog generala do beskonačnosti, niti jedan od dva generala neće moći biti siguran je li njihova poruka došla do odredišta, još manje je li autentična. Postoji nekoliko načina na koje *blockchain* rješava ovaj problem, a neki od najčešće korištenih su: dokaz o radu (eng. *proof-of-work*), dokaz o udjelu(eng. *proof-of-stake*), izaslani dokaz o udjelu(eng. delegated *proof-of-stake*), dokaz auteriteta(eng. *proof of authority*) te Practical Byzantine Fault Tolerance(PBFT) koja je najzastupljenija u privatnim ili industrijskim blockchain-ovima. U ovom radu objasniti ćemo samo *proof-of-work* i *proof-of-stake*.

2.5.2. Dokaz o radu(*proof-of-work*)

Ovaj protokol je u začetku osmišljen kako bi odvratio *DOS*(*Denial of service*)[2] napade tako što će od klijenta tražiti obavljanje određene količine rada, obično zadatak koji u određenoj količini opterećuje procesor. Koncept su osmislili Cynthia Dwork i Moni Naor. Termin „*proof-of-work*“ prvi puta je skovan 1999. godine u članku Markusa Jakobssona i Ari Juels.

Dokaz o radu je podatak koji zadovoljava određene uvijete, ali težak(skup, dugotrajan) za „proizvodnju“. Provjera zadovoljava li takav podatak zadane uvijete mora biti trivijalan. Proizvodnja dokaza o radu može biti nasumičan proces s niskom vjerojatnosti uspjeha, što iziskiva mnogo pokušaja i grešaka prije nego što se dođe do valjanog dokaza o radu. Jedna od primjena dokaza o radu je sprječavanje *spam*-a[3] kod elektronske pošte gdje se kod pošiljatelji traži dokaz o radu temeljen na sadržaju e-pošte(uključujući adresu primatelja) za svaki e-mail. Legitimni

[2] *DOS* – napad kod kojeg napadač nastoji učiniti računalni sustav ili mrežu nedostupnima tako da neprestano šalje suvišne zahtjeve u pokušaju prepterećenja funkcije
[3] *Spam* – neželjena elektronska pošta, često komercijalne

pošiljatelj neće imati nikakav problem u obavljanju rada koji je potreban za slanje, dok će *spammeri* imati problem kod generiranja zahtijevanih dokaza o radu jer oni iziskuju veliku količinu računalnih resursa.

Kod *blockchain*-a za prihvaćanje blokova potrebni su dokazi o radu koji su povezani s podacima koji se trebaju prihvatiti. Težina dobivanja tog dokaza o radu se kod bitcoina npr. Prilagođava na način koji ograničava količinu novih blokova koji se generiraju u mreži na jedan svakih 10 minuta. Vrlo mala vjerojatnosti uspješnog generiranja dokaza o radu čini nepredvidivim koje računalo u mreži će generirati sljedeći blok. Postupak pronalaska dokaza o radu još se naziva i „rudarenje"(eng. *mining*),

Promjena bloka(koja se može napraviti jedino stvaranjem novog bloka koji sadrži istog prethodnika) zahtijeva rekreaciju svih nasljednika i ponavljanje rada koji oni sadrže. Ovaj mehanizam štiti *blockchain* od remetilačkih utjecaja. Bitcoin koristi *hashcash* [1] dokaz o radu, dok Ethereum koristi *PoW* algoritam naziva „Ethash" koji se razvijen specifično za Ethereum. Razlog za razvoj nove PoW funkcije umjesto korištenja postojeće je adresiranje problema centralizacije kod kojeg mala skupina firmi koje proizvode hardware stekne nesrazmjerno veliku količinu računalne snage koja im omgoućava velik utjecaj na mrežu i omogućava manipulacije.

2.5.3. Dokaz o udjelu(*proof-of-stake*)

Dokaz o udjelu(PoS) je alternativni način validacije transakcija u bloku. Ovaj algoritam je stvoren kao konkurent PoW protokolu, ali cilj im je isti: osiguravanje decentralizacije moći i uključivanje transakcija u blokove i osiguranje slaganja oko ispravnog redoslijeda transakcija. Ideja da se svi natječu jedni protiv drugih u procesu rudarenja je rasipanje resursa. Umjesto toga, POS koristi proces biranja gdje se za validaciju sljedećeg bloka bira jedan čvor. Umjesto rudara, sada imamo validatore.

Validator ne rudari nove blokove, validator će" iskovati" novi blok. Bilo tko može postati validator. Ono što se mora napraviti da bi postao validator je polaganje određene količine žetona ili novčića koji se nazivaju udjel. Veličina udjela određuje šansu validatora da bude odabran za kovanje sljedećeg bloka. Korelacija u ovom procesu je linearna. Iako se na prvu to može činiti kao nepošteno jer sustav favorizira „bogate"(tj. one koji imaju veći udio), u stvarnost ovaj sustav je pošteniji od dokaza o radu jer „bogati" mogu koristiti moć ekonomije skale(cijene koje plaćaju za dodatnu opremu za rudarenje i struju ne raste linearno, nego što više od toga kupuju to će bolje cijene dobiti).

Kod PoS metode, kada je čvor odabran za validaciju sljedećeg bloka, on će provjeriti i osigurati da su sve transakcije unutar tog bloka valjane. Ako je sve kako bi trebalo biti čvor će validirati blok i dodati ga u lanac blokova. Nakon toga čvor dobije naknade koje su povezane s blokom. Razlog iz kojem možemo vjerovati validatorima je u tome što će isti izgubiti sav udio ako odobre neovlaštene transakcije. Dok god je količina udjela veća od nagrade koju prime od validacije, možemo biti sigurni da će ti čvorovi raditi svoj posao. Nitko neće raditi nešto što može proizvoditi gubitak novca.

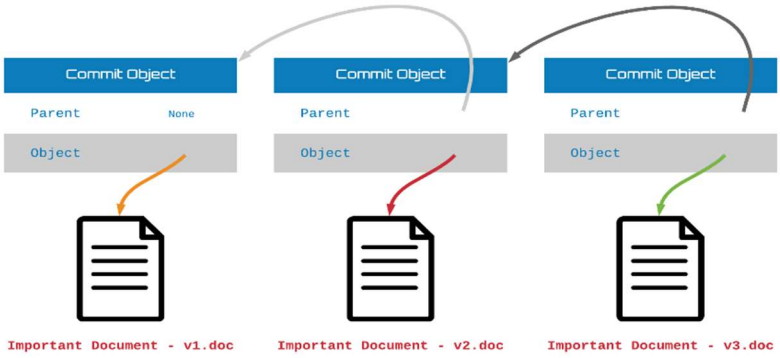
3. PREGLED KORIŠTENIH TEHNOLOGIJA I ALATA

U ovom poglavlju će ukratko biti objašnjene platforme, tehnologije i alati korišteni pri izradi praktičnog dijela završnog rada, tj. sustava za verifikaciju izdanih diploma na *blockchain*-u.

3.1. IPFS (InterPlanetary File System)

InterPlanetary File System(IPFS) je protokol i mreža dizajnirana za izradu sadržaja čija adresa se temelji na samom sadržaju (eng. *content addressable* ili skraćeno *CAS*) i metoda za pohranu i dijeljenje hipermedija u distribuiranom datotečnom sustavu. Sličan je torrentima po tome što dopušta korisnicima ne samo preuzimanje nego i smještanje sadržaja. Za razliku od centraliziranih poslužitelja IPFS se temelji na decentraliziranom sustavu korisnika od kojih svaki ima pohranjen dio cjelokupnih podataka, time stvarajući otporan sustav pohrane i dijeljenja podataka. Sustav nastoji povezati sva računala s istim sustavom datoteka.

Datoteke unutar IPFS sustava su pohranjene u obliku IPFS objekata i ti objekti mogu pohraniti datoteke veličine do 256kb. Također, mogu sadržavati poveznice na druge IPFS objekte. Datoteke veće od 256kb kao što su slike ili videozapisi se dijele u više IPFS objekata od kojih je svaki velik 256kb nakon čega će sustav stvoriti prazan objekt koji povezuje sve dijelove datoteke. Ovakva arhitektura nam omogućava korištenje IPFS-a kao datotečnog sustava. IPFS također podržava verzioniranje datoteka. IPFS sustav stvara *commit* objekte od kojih svaki ima poveznicu na svoju verziju datoteke i na prethodnu(osim prvog *commit*-a). IPFS se može promatrati kao jedinstveni BitTorrent *roj* [4](eng. *swarm*) gdje se objekti razmjenjuju unutar jednog Git repozitorija.



Slika 3.1 *Prikaz verzioniranja u IPS sustavu*

Razmotrimo što se događa kada dodamo datoteku na IPFS:

- Svaka datoteka i svi blokovi unutar nje dobijaju jedinstveni kriptografski *hash*.

[4] *Roj* – skup svih korisnika koji dijele torrent, uključujući one koji preuzimaju i one koji dijele datoteku

- IPFS uklanja duplikate na mreži i vodi računa o inačici svake datoteke.
- Svaki čvor u mreži poharajuje samo onaj sadržaj koji ga zanima, i uz to neke informacije koje pomažu kod identificiranja što tko pohranjuje.
- Kada tražimo datoteku, mreži šaljemo upit da pronađe čvorove koje pohranjuju sadržaj koji se nalazi iza jedinstvenog *hash*-a.
- Svaka datoteka može biti pronađena po njenom imenu koji je čitljiv za ljude koristeći decentralizirani sustav imenovanja koji se naziva **IPNS**[5].

3.2. Ethereum

Ethereum je javna, distribuirana, računalna platforma i operacijski sustav *otvorenog koda* [6] temeljena na *blockchain*-u koja kao jednu od značajki ima funkcionalnost *pametnih ugovora*[7]. Podržava izmijenjenu inačicu *Nakamoto konsensusa* [8]. Ether je „žeton" čiji je *blockchain* generiran od strane Ethereum platforme. Ether se može transferirati između računa i koristi za kompenzaciju *rudara*[9]. Ethereum pruža decentralizirani *virtualni stroj*[10], Ethereum Virtual Machine(EVM), koji može pokretati skripte koristeći mrežu javnih čvorova. Skup instrukcija koje izvodi EVM može izvoditi je Turing potpun. Spam u mreži se sprječava interim sustavom cijena transakcija koji se naziva „Gas".

Ethereum je predložen 2013. godine od strane Vitalika Buterina, istraživača kriptovaluta i programera. Razvoj je financiran *on-line* gurpnim financiranjem koje je odražano između srpnja i kolovoza 2014. Sustav je pušten u pogon 30. srpnja 2015. sa 72 milijuna unaprijed izrudarenih žetona, što u trenutku pisanja ovog teksta čini udjel od oko 68% ukupne zalihe u opticaju.

Ether, za razliku od npr. Bitcoin-a nema limit, ali fiksni omjer proizvodnje. U trenutku pisanja ovog teksta u prosjeku svakih 14 do 15 sekundi izrudari se prosječno 2 Ethera. Za kasnije je planiran prijelaz na hibridni sustav koji kombinira *proof-of-stake* i *proof-of work.*

---

[5] IPNS (Inter-Planetary Name System) – sustav za izradu i ažuriranje promjenjivih poveznica na IPFS sadržaj
[6] *Otvoreni kod* – softver čiji je izvorni kod dostupan javnosti na uvid, korištene, izmjene i daljnje raspačavanje
[7] *Pametni ugovor* – unaprijed definirani set pravila koja se pišu u obliku računalog koda i čini jedan softverski program koji se pokreće na određenoj *blockchain* platformi.
[8] *Nakamoto konsesnsus* – skup pravila koja upravljaju mehanizmom konsensusa
[9] *Rudari* - čvorovi koji dodaju zapise transakcija u blokove pri tome obavljajući potrebne izračune
[10] *Virtualni stroj* – Emulirani računalni sustav

3.2.1. Solidity

Pametni ugovori za Ethereum se pišu u jeziku zvanom Solidity. Solidity je *statički tipkan*[11] programski jezik dizajniran za razvoj pametnih ugovora koji se vrte na EVM-u. Solidity se kompajlira u *bytekod* [12]koji se izvodi na EVM-u. Sa Solidiy-em programeri su u mogućnosti pisati aplikacije koje implementiraju poslovnu logiku koja se može samostalno provoditi i ugrađena je u pametne ugovore, koji bilježe neporecive i mjerodavne zapise transakcija. Pisanje pametnih ugovora u programskim jezicima kao što je Solidity se smatra jednostavnim za one koji već imaju programerskog iskustva.

Temelji se na ECMAScript sintaksi kako bi se približio već postojećim web developerima, ali za razliku od EXMAScripta, kao što je već navedeno, Solidity ima statičko tipkanje i može vratiti promjenjivi broj tipova. U usporedbi s ostalim dotadašnjim jezicima za EVM, kao što su Serpent i Mutan, Solidity sadrži nekoliko važnih razlika. Ugovori podržavaju nasljeđivanje, uključujući višestruko nasljeđivanje.

Primjer Solidity programa:

```
1   pragma solidity 0.5.8;
2
3   contract Coin {
4
5   address public minter;
6   uint public totalCoins;
7
8   event LogCoinsMinted(address deliveredTo, uint amount);
9   event LogCoinsSent(address sentTo, uint amount);
10
11  mapping (address => uint) balances;
12  constructor (uint initialCoins) public {
13  minter = msg.sender;
14  totalCoins = initialCoins;
15  balances[minter] = initialCoins;
16  }
17
18
19  function mint(address owner, uint amount) public {
20  if (msg.sender != minter) return;
21  balances[owner] += amount;
22  totalCoins += amount;
23  emit (LogCoinsMinted(owner, amount);
24  }
25
26  function send(address receiver, uint amount) public {
27  if (balances[msg.sender] < amount) return;
28  balances[msg.sender] -= amount;
29  balances[receiver] += amount;
30  emit(LogCoinsSent(receiver, amount));
31  }
32
33  function queryBalance(address addr) public view returns (uint balance) {
34  return balances[addr];
35  }
36
37  function killCoin() public returns (bool status) {
38  if (msg.sender != minter) revert("");
39  selfdestruct(minter);
40  }
41  }
```

Slika 3.2 *Prikaz jednostavnog Solidity programa*

[11] *Statički tipkan* – provjera tipova se vrši prilikom kompilacije
[12] *Bytekod* – vrsta seta instrukcija dizajnirana za učinkovito izvođenje od strane interpretera

U primjeru se nalazi jednostavni ugovor napisan u Solidity-u za novi žeton ili novčić. Ovaj žeton se može koristiti za praćenje bilo kojeg digitalnog dobra koji ima vrijednost.

U ovom ugovoru, stvorili smo digitalno dobro zvano „Coins“ koje je inicijalizirano s početnom zalihom. Ovdje će izvršitelj postati kovač/admin. Napisali smo i funkciju koja šalje novčiće na adresu i funkciju za upit o bilanci adrese. Također smo napisali funkciju koja kreira nove novčiće koristeći funkciju *mint*.

Ovdje je očito kako Solidity slijedi sintaksu JavaScripta, stoga je sintaksa slična osim na par mjesta.

3.3. MetaMask

Očito je da Ethereum ekosustav ima puno tehničkih prednosti koje može pružiti, ali većina ljudi ih smatraju podosta zbunjujućima i teško je shvatiti kako stvari rade. Čak i pristup većini decentraliziranih aplikacija može predstavljati popriličan izazov jer zahtjeva pokretanje cijelog Ethereum čvora na računalu, na običnu osobu to je previše. Ovdje MetaMask ulazi u igru jer adresira većinu takvih problema. Ovaj priključak omogućuje bilo kome tko koristi neki od popularnijih preglednika(Google Crome, Mozilla Firefox, Brave) pristup decentraliziranim aplikacijama izravno iz preglednika. Samo ta činjenica je velik napredak, uzimajući u obzir da je to za novake do sada bio veliki problem. Kompliciran proces čini decentralizirane aplikacije automatski manje privlačnima većini ljudi, što svakako nije dobra stvar.

MetaMask ide i korak dalje, mada, ovaj protokol otklanja potrebu za pokretanjem punog Ethereum čvora. MetaMask donosi veliki napredak za Ethereum ekosustav. Osim jednostavnije pristupa decentraliziranim aplikacija, MetaMask također pruža sigurni trezor za identitet, koji dopušta bilo kome upravljanje različitim identitetima na različitima web stranicama. Štoviše, ovi različiti identiteti se mogu koristiti za potpisivanje transakcija na *blockchain*-u. Sve je dostupno kroz pristupačno korisničko sučelje, koje dodatno snižava barijeru za ulaz.

3.4. Ethereum Testnet

Kada razvijamo programa ze EVM, njihovo pokretanje i korištenje plaćamo „gasom“. Ova trošak nekada može biti faktor koji sprječava tako nešto u vremenima kada je mreža opterećena i može biti financijski opasan – program koji je postavljen na mrežu i sadrži bug je program za koji je zauvijek moguća zlouporaba jer su sve promjene na Ethereum *blockchain*-u trajne i ne mogu se opozvati.

Testneti su gotovo identične kopije Ethereum *blockchain*-a osim činjenice da je njihov Ether bezvrijedan(kao i programi koji se vrte na tim testnetima). Postoji nekoliko vrsta testneta.

3.4.1. Javni testnet

Javni testeneti su dostupni svima, spojeni su na internet. Svatko im može pristupiti u bilo koje doba, čak i iz popularnih sučelja za novčanike kao što su *MyEtherWallet* ili već spomenuti MetaMask. Postoji nekoliko javnih *testnet*-a, a to su:

1) Ropsten
2) Rinkeby
3) Kovan

Svi navedeni testneti su dostupni kroz MetaMask.

Ropsten

Ropsten je pokrenut u studenom 2016. godine. Njegov Ether se može rudariti isto kao onaj na *Mainnet*-u. Od sva tri *testnet*-a, Ropsten najviše liči na trenutni *Mainnet*. Njegovi rezultati liče na *Mainnet* rezultate jer njegov mehanizam konsenzusa PoW, stoga je i simulacija potvrde transakcije najrealnija.

Na Ropseten mreži, Ether se može rudariti ili zatražiti kroz Ropsten slavinu(eng. *faucet*) – web stranicu koja postoji samo za podjelu besplatnog testnog Ethera.

Zato što se Ether može rudariti na Ropsten-u, susceptibilan je na *spam* napade – val beskorisnih transakcija koji zakrčuju mrežu. Jedan takav napad dogodio se u veljači 2017 kada su napadači rudarili enormne količine Ethera i neprestano slali prevelike transakcije u mrežu. Ethereum-ova veličina bloka je dizajnirana tako da bude prilagodljiva i da raste zajedno sa potražnjom. Uspjeli su napumpati maksimalnu veličinu blokova na nekoliko bilijuna gasa(s do tada uobičajenih 4 miljiuna). Nakon povećanja blokova na tu razinu, počeli su vellike i zahtjevne transakcije time blokirajući rad svih drugih sudionika u mreži.

Ropsten transakcije se mogu pregledavati na *Etherscanu*[13].

[13] *Etherscan* – stranica na kojoj je moguće pregledavati transakcije na Ethereum *blockchain*-ovima.

Kovan

Kovan je pokrenut u ožujku 2017. od strane Parity tima nakon što je Ropsten napadnut. Umjesto rudarenja s *proof-of-work* algoritmim, Kovan koristi PoA(*proof-of-authority*) koji pojednostavljeno znači da su određeni čvorovi autorizirani za proizvodnju novih blokova i potvrdu transakcija, i isključivo ti čvorovi mogu obavljati taj posao. Kovan se može nabaviti jedino zahtijevanjem od takvog čvora putem za to namijenjenog čvora, ili dobivanjem od adrese koja ga već posjeduje. Zbog toga što radi na PoA mehanizmu, Kovan se blago razlikuje od *Mainneta* i stoga se ne može smatrati točnom simulacijom. Unatoč tome, sjajan je za javno testiranje, pouzdan i siguran, i imun na *spam* napade. Kovan je također podržan od strane Etherscan-a.

Rinkeby

Rinkeby je pokrenut u travnju 2017. godine, posjeduje neke prednosti Kovana s dvije manje preinake. Također je podržan na Etherscan-u i ether se može dobiti na zahtjev od autoriziranih *faucet*-a. Kod implementacije našeg rješenja koristi ćemo Rinkeby *testnet*.

4. PRIMJENA BLOCKCHAIN-A ZA VERIFIKACIJU DIPLOMA

Za potrebe ovoga završnog rada dizajnirati ćemo samo MVP[14], odnosno rješenje koje zadovoljava tek osnovne funkcije. Kod takvog prozvoda imati ćemo samo dvije uloge:

- Osoblje fakulteta/obrazovne ustanove
- Studenti i javnost

Zadaće osoblja fakulteta/obrazovne ustanove:

- Priprema digitalne kopije ili skenirane diplome
- Skupljanje podataka studenta: ime i prezime, smjer, datum itd.
- Stavljanje traženih informacija u sustav za verifikaciju diploma

Studenti i javnost:

- Provjera valjanost diplome

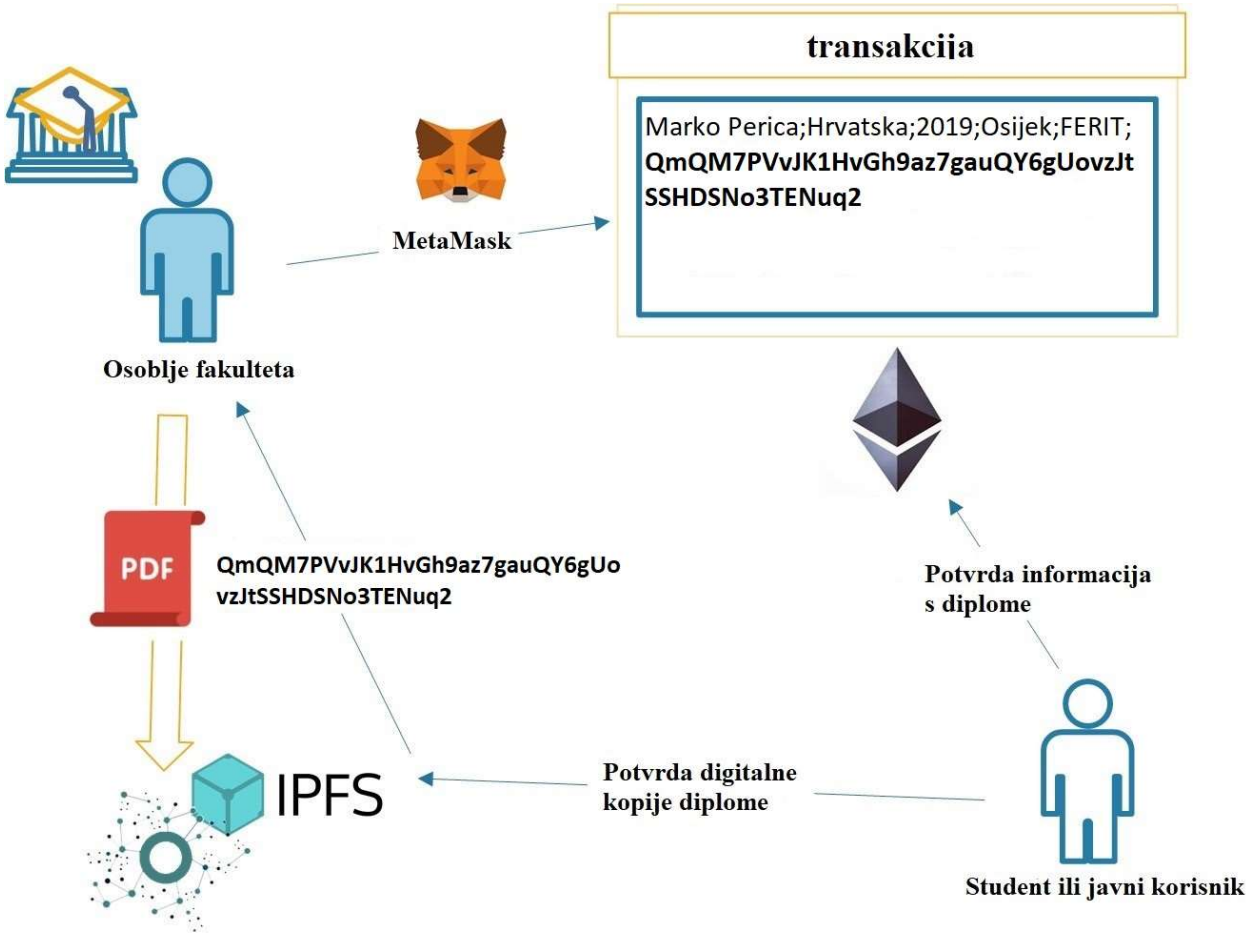
Zahtjevi na sustav za verifikaciju diploma:

- Samo obrazovna ustanova može dodavati informacije o diplomama
- Podržava bilo koji format diplome
- Pruža stalan i javan pristup

4.1. Dizajn rješenja

1. IPFS se koristi za pohranu datoteka diplome. Svaka jedinstvena datoteka (uključujući i diplomu) posjeduje svoj jedinstveni *hash*.
2. Osoblje obrazovne ustanove unosi informacije o studentu i jedinstveni *hash* diplome se uzima s IPFS-a i unosi u *data* polje Ethereum transakcije.
3. Za svaku obrazovnu ustanovu objavljuje se jedinstvena javna adresa u svrhu jamčenja autoriteta.

MVP – Minimum viable product



Slika 4.1 *Prikaz dizajna rješenja*

4.2. Postavljanje diplome na IPFS

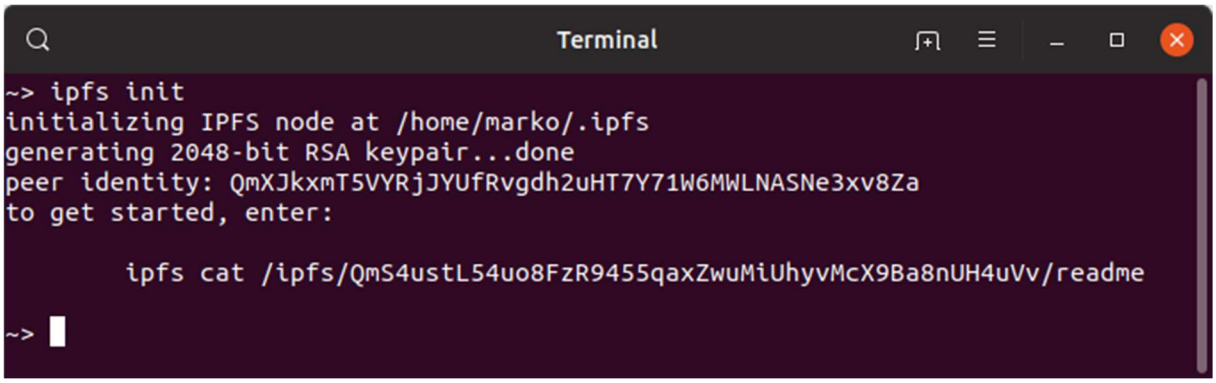
Za potrebe ovog primjera napravili smo testnu diplomu koja će nam poslužiti samo za demonstraciju. Korišteni su resursi sa stranice Freepik.com[15]

[15] Freepik.com – repozitorij besplatnih resursa uz atribuciju

Slika 4.2 *Testna diploma*

Ethereum *blockchain* ne podržava pohranu medijskih datoteka. Za to koristimo IPFS, dok na Ethereum *blockchain* postavljamo samo negov *hash*. Za potrebe završnog rada postaviti ćemo lokalni IPFS na kojemu će se nalaziti datoteke.

IPFS pohranjuje sve postavke i interne podatke u direktorij zvan *repozitorij*. Prije prve upotrebe IPFS-a moramo inicijalizirati repozitorij naredbom ipfs init.



Slika 4.3 *Inicijalizacija repozitorija*

Ovom naredbom stvorili smo lokalni repozitorij za trenutnog korisnika. Također je i generiran kriptografski par ključeva koji omogućuje IPFS čvoru kriptografsko potpisivanje sadržaja i poruka koje stvorimo. Također, dobili smo natuknicu kako započeti s radom. ipfs cat će pročitati sadržaji koji odgovora putu koji smo unijeli. Ako sadržaj nije dostupan lokalno, IPFS će ga pokušati pronaći *na peer-to-peer* mreži.

Pokretnimo ipfs cat komandu zajedno s putem koji smo dobili u init poruci. Dobijemo ovako nešto:

```
~> ipfs cat /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv/readme
Hello and Welcome to IPFS!

If you're seeing this, you have successfully installed
IPFS and are now interfacing with the ipfs merkledag!

 -------------------------------------------------------
| Warning:                                              |
|   This is alpha software. Use at your own discretion! |
|   Much is missing or lacking polish. There are bugs.  |
|   Not yet secure. Read the security notes for more.   |
 -------------------------------------------------------

Check out some of the other files in this directory:

  ./about
  ./help
  ./quick-start     <-- usage examples
  ./readme          <-- this file
  ./security-notes
~>
```

Slika 4.4 *Output ipfs cat komande*

To je znak da je IPFS uspješno instaliran i možemo komunicirati s IPFS-ovim merkledag-om[16].

Pokretanjem naredbe ipfs daemon u zasebnom terminalu naš lokalni čvor se pridružuje javnoj mreži.

```
Downloads> ipfs daemon
Initializing daemon...
go-ipfs version: 0.4.22-
Repo version: 7
System version: amd64/linux
Golang version: go1.12.7
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/192.168.5.3/tcp/4001
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /p2p-circuit
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/192.168.5.3/tcp/4001
Swarm announcing /ip6/::1/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
WebUI: http://127.0.0.1:5001/webui
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

Slika 4.5 *Pridruživanje čvora javnoj mreži*

Dodavanje datoteke vršimo unosom naredbe ipfs add <ime datoteke>.

[16] merkledag – struktura slična merkle stablu koja za razliku od merkle stabla ne mora biti strogo balansirana i njeni ne-lisni čvorovi mogu sadržavati podatke

```
Downloads> ls
'diploma example.png'
Downloads> ipfs add 'diploma example.png'
added QmQM7PVvJK1HvGh9az7gauQY6gUovzJtSSHDSNo3TENuq2 diploma example.png
 478.35 KiB / 478.35 KiB [=========================================] 100.00%
Downloads>
```

Slika 4.6 *Dodavanje datoteke*

Za dohvaćanje datoteke s IPFS-s možemo koristiti *curl* (*command line tool and library*)[17]. Kod korištenja opcije -o navodimo ime datoteke pod kojim će biti spremljen naš URL.

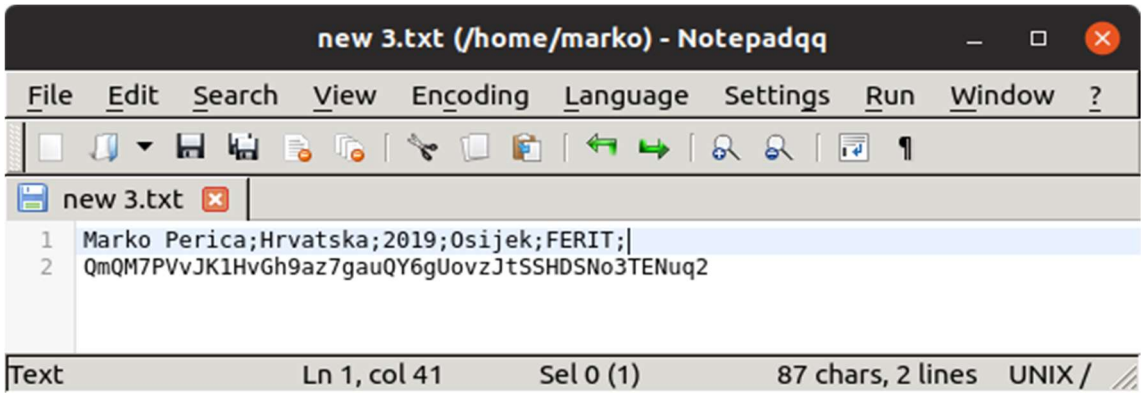
```
Downloads> ls
'diploma example.png'
Downloads> curl -o diploma.png "http://127.0.0.1:8080/QmQM7PVvJK1HvGh9az7gauQY6gUovzJtSSHDSNo3TENuq2"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    19  100    19    0     0   9500      0 --:--:-- --:--:-- --:--:--  9500
Downloads> ls
'diploma example.png'   diploma.png
Downloads>
```

Slika 4.7 *Preuzimanje korištenjem curl-a*

4.3. Postavljanje podataka na blockchain

Sada trebamo napraviti zapis naše diplome koji će biti postavljen na *blockchain* i uključivati će *hash* diplome. Za to možemo koristiti bilo koji uređivač teksta.

[17] curl – besplati alat koji korist URL sintaksu za prijenos podataka s i na servere.

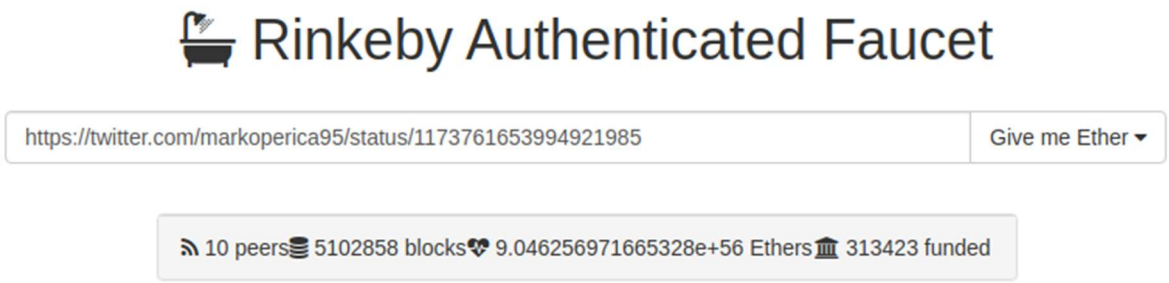


Slika 4.8 *Zapis koji postavljamo na blockchain*

Ethereum polje podataka podržava samo heksadecimalni format. Za pretvorbu možemo koristiti jedan od brojnih alata koji se mogu pronaći na Internetu. Za naš zapis dobijamo sljedeći znakovni niz koji ćemo postaviti na Ethereum blockchain:

4d61726b6f205065726963613b4872766174736b613b323031393b4f73696a656b3b4645524954
3ba516d514d375056764a4b314876476839617a376761755159366755 6f767a4a7453534844534
e6f3354454e757132

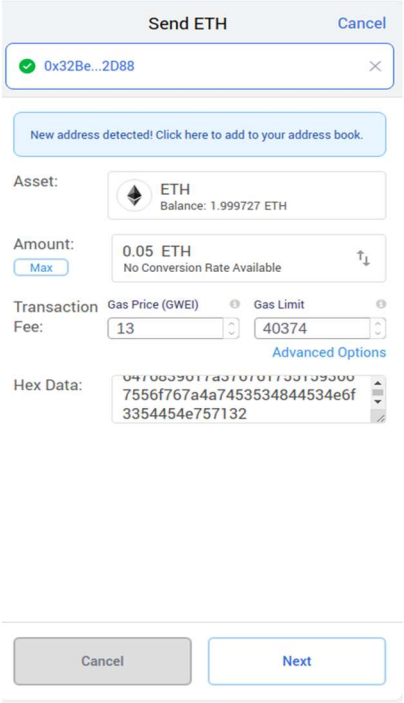
Za taj zadatak koristimo MetaMask kriptovalutni novčanik koji se može besplatno preuzeti i instalirati na *https://metamask.io.* Za potrebe ovog rješenja koristiti ćemo Rinkeby testnet, stoga nam treba i nešto Ethereum žetona koje možemo dobiti na jednom od autoriziranih *faucet-a (https://faucet.rinkeby.io).* Za dobivanje Ethereum žetona na Rinkeby testnet-u na jednoj od podržanih društvenih mreža objaviti javan post koji sadrži Etherum adresu na koju će žetoni biti poslani – ovo je uvjet zbog sprječavanja iscrpljivanja svih dostupnih sredstava ili akumulacije dovoljno Ethera koji bi bili dovoljni za izvođenje *spam* napada.



Slika 4.9 *Rinkeby faucet*

Nakon što smo zatražili i zaprimili naš Ether možemo pokrenuti Ethereum transakciju. U MetaMask novčaniku odabiremo Rinkeby Test Network. U trenutku pisanja ovog teksta polje za

unos transakcijskih podataka je maknuto jer je zbunjivalo i bilo višak za veliki broj korisnika, ali može se uključiti u postavkama. Pod adresu primatelja(Recipient Address) možemo staviti bilo što, u budućnosti to može biti npr. adresa studenta, pod iznos(Amount) možemo staviti nula ili nešto drugo(pod uvjetom da imamo dovoljno sredstva) i na kraju pod Transaction Data(transakcijski podaci) stavljamo naš heksadecimalni znakovni niz.



Slika 4.10 *Popunjavanje polja za transakciju*

4.4. Provjera valjanosti diplome

Otvorimo poveznicu na obavljenu transakciju. Provjerimo neke od najvažnijih parametara transakcije.

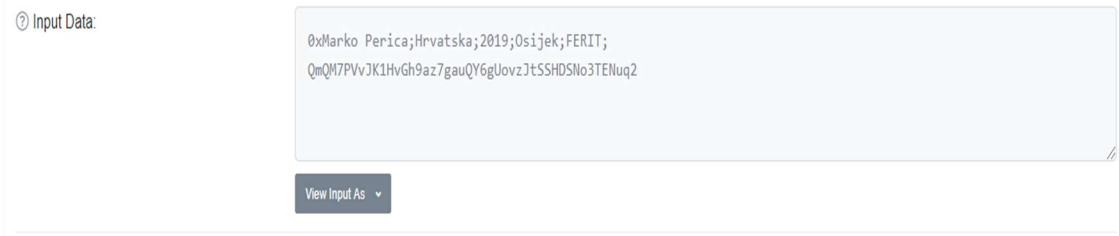
- Status transakcije je „Success“(*hrv*. uspješna)

Slika 4.11 *Status transakcije*

- Polje „From“ (*hrv*. od) je popunjeno našom adresom.

Slika 4.12 *Adresa pošiljatelja*

- Ako pogledamo Input Data u UTF-8 formatu vidimo naš izvorni znakovni niz.



Slika 4.13 *Data polje u utf-8 formatu*

Možemo pogledati diplomu na https://ipfs.io/ipfs/{hash_datoteke} i usporediti je s našom izvornom diplomom koju smo učitali.

5. ZAKLJUČAK

Lanac blokova je domišljato rješenje za problem koji nam je do njegovog otkriće djelovao nerješivo, a to je siguran prijenos informacija između dvije strane bez posredovanja treće strane. Tehnologija lanaca blokova velikim dijelom duguje svoju popularnost kriptovalutama koje su ipak samo mali djelić svih potencijalnih primjena ove tehnologije i jedan od tih ostalih potencijalnih primjena(ili već i stvarnih jer se već na nekim mjestima koriste) je verifikacija diploma.

Nakon pregleda tehnologija i alata u ovom završnome radu demonstrirali smo upotrebu istih za rješavanja problema verifikacije diploma. Prikazano je rješenje koje zadovoljava osnovne funkcije upotrebom IPFS sustava za distribuirani web i Ethereuma. Iako je demonstrirano rješenje poprilično rudimentarno ono može poslužiti kao temelj za sustave i aplikativna rješenja koja bi proces pohrane i verifikacije diploma napravila lakšim i pristupačnijim širokim masama.

LITERATURA

[1] »What is Hashcash?,« 24 6 2019. [Mrežno]. Available: https://www.bitcoinmining.com/what-is-hashcash/.

[2] »Public key cryptography,« 20 lipanj 2019. [Mrežno]. Available: https://en.wikipedia.org/wiki/Public-key_cryptography.

[3] B. Curran, "What is a Merkle Tree? Beginner's Guide to this Blockchain Component," 22 Lipanj 2019. [Online]. Available: https://blockonomi.com/merkle-tree/.

[4] »Peer-to-peer,« 22 lipanj 2019. [Mrežno]. Available: https://en.wikipedia.org/wiki/Peer-to-peer.

[5] V. Miletić, »Kriptografija javnog ključa alatom GnuPG,« 20 lipanj 2019. [Mrežno]. Available: https://lab.miletic.net/hr/nastava/materijali/gnupg-kriptografija-javnog-kljuca/.

[6] T. Sundaramoorthy, »Hashing and Public Key Cryptography for Beginners,« 19 lipanj 2019. [Mrežno]. Available: https://medium.com/@thyagsundaramoorthy/hashing-and-public-key-cryptography-for-beginners-292aaf14efae.

[7] "InterPlanetary File System," 23 lipanj 2019. [Online]. Available: https://en.wikipedia.org/wiki/InterPlanetary_File_System.

[8] S. E. -. Savjee, »IPFS: Interplanetary file storage!,« 23 lipanj 2019. [Mrežno]. Available: https://www.youtube.com/watch?v=5Uj6uR3fp-U.

[9] A. Rosic, »Basic Primer: Blockchain Consensus Protocol,« 23 lipanj 2019. [Mrežno]. Available: https://blockgeeks.com/guides/blockchain-consensus/.

[10] B. Mining, »What is Proof of Work,« [Mrežno]. Available: https://www.bitcoinmining.com/what-is-proof-of-work/. [Pokušaj pristupa 24 lipanj 2019].

[11] »Ethereum,« [Mrežno]. Available: https://en.wikipedia.org/wiki/Ethereum. [Pokušaj pristupa 23 lipanj 2019].

[12] »Wikipedia,« [Mrežno]. Available: https://en.wikipedia.org/wiki/Denial-of-service_attack. [Pokušaj pristupa 23 lipanj 2019].

[13] »What is Proof of Stake? | Delegated Proof of Stake,« [Mrežno]. Available: https://achainofblocks.com/2018/08/24/proof-of-stake-delegated-proof-of-stake-proof-of-work/. [Pokušaj pristupa 24 6 2019].

[14] »Learn Solidity: Basics of Solidity By Example,« [Mrežno]. Available: https://www.toshblocks.com/solidity/basics-solidity-example/. [Pokušaj pristupa 25 lipanj 2019].

[15] B. Škvorc, »Što je to Ethereum Testnet i kako se koristi?,« bitfalls.com, 31 svibanj 2018. [Mrežno]. Available: https://bitfalls.com/hr/2018/05/31/what-is-an-ethereum-testnet-and-how-is-it-used/. [Pokušaj pristupa 25 06 2019].

[16] A. Ryzhenko, »Hackernoon,« 4 August 2018. [Mrežno]. Available: https://hackernoon.com/develop-blockchain-trusted-diploma-verification-system-in-15-minutes-step-by-step-instruction-fdcf37a244ab. [Pokušaj pristupa 12 September 2019].

SAŽETAK

U ovom završnom radu najprije je prikazana teorijska analiza tehnologija na kojima počiva lanac blokova. Teorijski se obrađuju hashing, kriptografija javnog ključa, peer-to-peer mreže, Merkle stabla i neke od konsenzusnih metoda. Zatim je ukratko predstavljena najpopularnija javna decentralizirana platforma za stvaranje tzv. „pametnih ugovora“ – Ethereum i jezik na kojemu se ti ugovori pišu – Solidity. Zbog činjenice da je lanac blokova nepromjenjiv i troška transakcija na Eteheurum-ovom Mainnet-u za testno izvođenje napisanih programa preporučuje se korištenje nekog od dostupnih test-neta koji su predstavljeni ovdje. U teoriji, ne postoji granica veličinu bloka u Ethereum-u, ali tehnologija lanca blokova nije namijenjena za pohranu većih datoteka jer je to jako skupo. Međutim, blockchain se može koristiti za pohranu hasheva podataka koji se nalaze negdje drugdje – u našem slučaju to je IPFS. Korištenjem IPFS-a možemo osigurati dostupnost podataka bez visokog troška što je demonstrirano u ovom završnom radu. Sinergijom ovih dvaju tehnologija i alata koji njima barataju osigurano je relativno jednostavno i jeftino rješenje za verifikaciju diploma koje se kasnije može nadograditi.

Ključne riječi: distribuirani, hash,konsenzus, kriptovalute, lanac blokova, pametni ugovor

ABSTRACT

In this paper firstly we introduced and theoretically analyzed what forms a basis for blockchain technology. We elaborated on hashing, public-key cryptography, peer-to-peer networks, Merkle trees and some of the consensus methods. Subsequently, the most popular decentralized platform for „smart contracts“ – Ethereum and the language in which they are written is introduced. Because of the fact that blockchains are immutable and the cost of transactions on Ethereum's Mainnet it is recommended to use one of the testnets who are also introduced. In theory, there is no limit for a block size in Ethereum, but blockchain technology as it is is not meant to serve as a storage place for large files because that would be very expensive. However, blockchain can be used to store hashes of data that is stored somewhere else – in our case that is IPFS. By using IPFS we can ensure the availability of data without high costs which is demonstrated in this paper. A synergy of these two technologies and tools that work with them we made a relatively simple and cheap solution for verification of diplomas that can later be built upon.

Keywords: blockchain, consensus, cryptocurrencies, distributed, hash, smart contract

ŽIVOTOPIS

Marko Perica rođen je 13. svibnja. 2019. u Vinkovcima. U Starim Mikanovcima završava osnovnu školu „Stjepan Cvrković“ te 2010. upisuje Tehničku školu Nikole Tesle u Vukovaru. 2014. godine ostvaruje upis na Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek.

Marko Perica