Sustav praćenja opskrbnog lanca u automobilskoj industriji zasnovan na tehnologiji lanca blokova

Pejčić, Lovro

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

Sveučilišni diplomski studij automobilskog računarstva i komunikacija

Sustav praćenja opskrbnog lanca u automobilskoj industriji zasnovan na tehnologiji lanca blokova

Diplomski rad

Lovro Pejčić

Osijek, 2020.

Sadržaj

1. UVOD

Automobilska industrija, kao jedna od glavnih tehnoloških grana, prolazi kroz brze, česte i intenzivne promjene. Ova će industrija uskoro postati više od puke industrije vozila te će u skoroj budućnosti biti izrazito drugačija od današnje. Trenutna industrija je vrlo izložena određenim izazovima kao što su krivotvorenje, nedostatak transparentnosti te neučinkovito upravljanje resursima. Ovi izazovi potiču tvrtke da poboljšaju strategije i operacije unutar lanca opskrbe, jer se optimirani lanac opskrbe može pokazati ključnim elementom koji proizvođače automobila može izdvojiti od svojih konkurenata. Lanac blokova (eng. *blockchain*) je jedna od ključnih i inovativnih tehnologija koja mijenja način upravljanja podacima unutar opskrbnog lanca. Lanac blokova može sadržavati podatke bilo koje vrste, no u većini slučajeva sadrži financijske podatke i podatke svih transakcija u mreži. Kako opskrbni lanci s vremenom postaju sve složeniji, uključuju raznolike sudionike i uglavnom se oslanjaju na brojne vanjske posrednike, lanac blokova se pojavio kao snažni kandidat za upravljanje komunikacijom unutar ekosustava lanca opskrbe te razmjenama podataka i dokumenata.

Cilj ovog diplomskog rada je analizirati mogućnosti tehnologije lanca blokova s naglaskom na primjenjivost u potpori opskrbnog lanca u automobilskoj industriji. Također, jedan od ciljeva je i izraditi konceptualno programsko rješenje zasnovano na tehnologiji lanca blokova koje omogućuje praćenje opskrbnog lanca automobilske industrije te ukazati na prednosti i potencijal korištenja ovakve tehnologije unutar proizvodnog lanca.

U drugom je poglavlju analizirana tehnologija lanca blokova uključujući njene sastavnice kao što su konsenzus, konsenzusne metode, kriptografija unutar lanca blokova, struktura blokova i načini povezivanja blokova te pametni ugovori. U trećem poglavlju prikazano je trenutno stanje područja te prednosti koje lanac blokova može ponuditi opskrbnim lancima. U četvrtom poglavlju modeliran je sustav opskrbnog lanca automobilske industrije temeljen na lancu blokova pomoću kojeg je prikazan način funkcioniranja tog sustava te postupak njegove izrade. U petom poglavlju detaljno je prikazan postupak razvoja opskrbnog lanca temeljenog na lancu blokova pomoću pametnih ugovora, dok je u šestom poglavlju napravljena je analiza ostvarenog konceptualnog programskog rješenja kao i cijelog područja.

2. LANAC BLOKOVA

Prema [1], lanac blokova je zajednička, raspodijeljena knjiga koja olakšava proces evidentiranja transakcija i praćenja imovine u poslovnim mrežama. Imovina može biti materijalna ili nematerijalna. Materijalna je imovina primjerice kuća, automobil, novac ili zemlja, a nematerijalna intelektualno vlasništvo, poput autorskih prava, patenata ili brendova. Na mreži lanca blokova gotovo se sve može pratiti i razmjenjivati, smanjujući rizik i troškove za sve uključene sudionike. Glavna prednost lanca blokova je decentralizacija podataka, što se postiže glavnom knjigom (eng. *ledger*) koja je javna, sadrži sve podatke i u vlasništvu je svih korisnika, za razliku od centralizirane mreže gdje se podatci spremaju kod jednog poslužitelja te svi vode zasebne knjige. Tehnologija lanca blokova je kombinacija *peer-to-peer* (P2P) mreže i raspodijeljenog poslužitelja koji vremenskim žigom obilježava sve transakcije. Ukoliko nešto krene po zlu, u sustavima kao što su automobili, važno je znati porijeklo svake komponente unutar opskrbnog lanca; od proizvođača, datuma proizvodnje, pa čak i do programa za proizvodnju tih dijelova. Lanac blokova može sadržavati kompletne detalje o izvoru svake komponente, od same sirovine pa do gotovog proizvoda, koje su dostupne proizvođačima, odnosno sudionicima u opskrbnom lancu. Lanac blokova, koji čine blokovi transakcija međusobno povezani, otporan je na promjene podataka samih transakcija čime se sprječavaju prevare unutar sustava. Stanje mreže je sinkronizirano i dostupno svim sudionicima u mreži zbog čega slanje nepostojeće imovine nije moguće.

Kao što je prikazano u [2], problem dvostruke potrošnje može biti riješen bez posrednika, odnosno treće strane kojoj svi sudionici vjeruju. Svaka se transakcija, nakon što je odobrena, šalje različitim čvorovima u mreži koji ju zapisuju. Čvorovi kojima se takva transakcija šalje ovise o samoj vrsti mreže. Mreže mogu biti javne, privatne, konzorcijske ili hibridne. Javna mreža može raditi na dva načina. U prvom slučaju svi mogu zapisivati i čitati podatke, a u drugom slučaju svi mogu čitati podatke, ali ih samo ovjerene osobe mogu zapisivati. Privatna je mreža, za razliku od javne, restriktivna, odnosno mogu joj pristupati samo odabrani članovi koji posjeduju određeni *token*. Privatna je mreža vrlo pogodna za organizacije pri menadžmentu opskrbnog lanca, budući da je razina sigurnosti, ovlasti i dostupnosti u njihovim rukama. Konzorcijska je mreža, za razliku od privatne mreže, polu-decentralizirana, gdje sudjeluje više od jedne organizacije u upravljanju same mreže. Vrlo je povoljna za banke ili državne organizacije. Hibridna mreža kombinacija je javne i privatne mreže u kojoj se iskorištavaju odlike ovlasti privatne mreže, a sigurnost i transparentnost javne mreže. Neovisno o vrsti mreže koja se koristi, postizanje dogovora, odnosno konsenzus među sudionicima mreže mora se ostvariti, kako bi se potvrdila vjerodostojnost podataka.

2.1. Problem bizantskih generala

Kako bi shvatili način funkcioniranja lanca blokova i način na koji on postiže konsenzus, možemo koristiti metaforu [3] srednjovjekovne vojske koja je opkolila grad s velikim brojem divizija i generala koji ih predvode. Generali se trebaju dogovoriti o točnom trenutku kada će napasti grad, jer je grad u stanju obraniti se od pojedinačnih napada, ali ne i od kolektivnog. S obzirom da se generali ne mogu vidjeti međusobno, jedini način komunikacije između njih je putem glasnika, što predstavlja problem razmjene informacija. Kako bi generali izveli koordinirani napad moraju postići konsenzus. No, među generalima postoje dvije strane, oni odani ali i oni koje su neprijatelji potkupili. Kako bi odani generali uspjeli postići konsenzus, potrebna im je većina. Potreban im je mehanizam koji će im omogućiti da odluče koju će taktiku napada koristiti, a da pri tome potkupljeni generali pogrešnim informacijama ne uvjere odane generale da prihvate pogrešan plan. Oni odani ponašat će se u skladu s algoritmom, a potkupljeni će se ponašati po svojoj volji. Upravo problem bizantskih generala prikazuje probleme postizanja konsenzusa u raspodijeljenim sustavima u kojima postoje loš protok informacija i neprijatelji sustava. U ovakvom modelu generali predstavljaju sudionike raspodijeljene mreže temeljene na lancu blokova, a glasnici način komuniciranja. Odani generali predstavljaju one sudionike mreže koji žele da ona funkcionira na temelju točnih informacija. Potkupljeni generali predstavljaju one sudionike koji žele unijeti netočne informacije u sustav ili lažirati postojeće. Cilj odanih generala je odlučiti je li informacija unesena u sustav ispravna ili pogrešna. Njihov problem mogla bi premostiti tehnologija lanca blokova tako da bi isključila one potkupljene iz sustava prilikom postizanja konsenzusa.

2.2. Konsenzus

Tehnologija lanca blokova omogućava drastičnu promjenu načina na koji pojedinci i poduzeća razmjenjuju digitalnu imovinu i pouzdano prate vlasništvo nad njima bez prisustva središnjeg autoriteta. Konkretno, raspodijeljeni skup sudionika jamči dosljednost glavne knjige usprkos potencijalnim zlonamjernim sudionicima koji se ponašaju proizvoljno, a koje se naziva bizantskim defektom [3]. Ono što u sustavima lanca blokova čini temeljnu odrednicu za postizanje konsenzusa mreže, upravo je nedostatak povjerenja. Konsenzus se odnosi na ispravne procese decentraliziranog sustava koji donose odluku o određenom bloku transakcije na određenom indeksu lanca blokova. Kao što je rečeno u [4], ovakav proces sastoji se od tri faktora: sporazuma, valjanosti i prestanka. Sporazum podrazumijeva da se niti u jednom slučaju dva ispravna procesa nisu odlučila za različite blokove. Valjanost govori da je odlučeni blok onaj blok koji je predložen od strane jednog procesa, dok prestanak uvjetuje da svi ispravni procesi potvrđuju odluku.

Protokol za postizanje konsenzusa je potreban kako bi bilo zajamčeno da su blokovi u ispravnom redoslijedu te kako bi se spriječilo nadodavanje neispravnih transakcija na kraj lanca. Postoji nekoliko metoda za postizanje konsenzusa, primjerice *Proof of Work, Proof of Stake, Proof of Authority, Practical Byzantine Fault Tolerance* i *Proof of Elapsed Time.*

2.2.1. Proof of Work

Prema [5], P*roof of Work* (PoW) kao jedna od najkorištenijih metoda za postizanje konsenzusa, koristi princip rudarenja (eng. *mining*) kao proces postizanja konsenzusa. Tijekom tog procesa, čvorovi (eng. *nodes*) se međusobno natječu kako bi potvrdili točnost podataka sadržanih u bloku. Čvorovi rješavaju složene matematičke zadatke po principu pokušaja i pogreške u kojem pokušavaju izračunati *hash* novog bloka uzimajući u obzir *hash* prethodnog bloka, transakciju novog bloka i *nounce. Nounce* (*number only used once*) je slučajni cijeli broj, odnosno bilo koji cijeli broj koji se može napisati između 0 i 4 294 967 296. Kada čvor dobije *hash* rezultat s određenim brojem vodećih nula, koje su zadane od strane sustava, obavještava sve čvorove u mreži koji zapisuju taj blok u decentraliziranu i javnu glavnu knjigu, a čvor koji je prvi uspio riješiti problem biva nagrađen. Tijekom ovog procesa, u kojem se čvorovi natječu dok se cijela mreža ne složi, troši se mnogo računalne snage, energije i vremena, a kako se složenost matematičkih problema povećava, povećavaju se i troškovi, što potiče čvorove da ne pokušavaju prevariti sustav.

2.2.2. Proof of Stake

Prema [6], P*roof of Stake* metoda postiže konsenzus zahtijevajući od korisnika ulaganje sredstava kako bi ostvarili mogućnost da budu izabrani za provjeru blokova transakcija. Kao i svakom konsenzusnom algoritmu korištenom u lancu blokova, cilj je postizanje raspodijeljenog konsenzusa, odnosno stvaranje sigurnog sustava u kojem će se čvorovi poticati na potvrđivanje transakcija drugih čvorova uz održavanje potpunog integriteta. U ovoj metodi, čvor koji će potvrditi blok je odabran polu-slučajnim postupkom koji sadrži dva koraka. U prvom koraku uzima se u obzir ulog čvora, s obzirom da svaki čvor koji potvrđuje transakcije mora posjedovati udio u mreži. Ulog u mreži podrazumijeva zaključavanje određenog dijela sredstava u virtualni sef koji služi kao jamstvo tog čvora. Što više čvor uloži, veća je šansa da bude izabran, ali ukoliko pokuša djelovati zlonamjerno ima i više sredstava za izgubiti. Nagrada čvorovima koji potvrde blok u ovakvim sustavima dolazi u obliku naknade za transakciju, za razliku od novo kreirane valute u P*roof of Work* sustavima koja se dodjeljuje tom čvoru. U prvom koraku ovog procesa nije puno slučajnosti, stoga je potrebno uključiti određeni stupanj „sreće“ kako bi se izbjegao scenarij u kojem najbogatiji čvorovi postaju još bogatiji. Zbog čega se u drugom koraku uključuje postupak

polu-slučajnog odabira koji ovisi o lancu blokova. Dvije najčešće metode su *Randomised Block Selection* i *Coin Age Selection*. U *Randomised Block Selection* metodi traže se čvorovi koji imaju kombinaciju najniže vrijednosti *hasha* i najvećeg uloga, dok se u *Coin Age Selection* metodi odabire čvor na temelju količine vremena u kojem su njegova sredstva zaključana u virtualni sef.

2.2.3. Proof of Authority

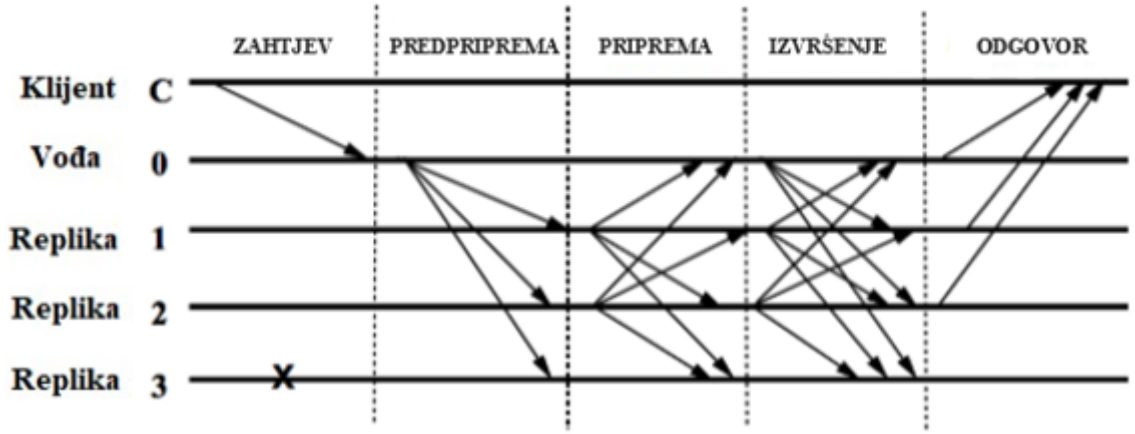
Kao što je opisano u [7], P*roof of Authority* je konsenzusni algoritam temeljen na reputaciji koji uvodi praktično i učinkovito rješenje za mreže lanca blokova, a posebno privatne. *Proof of Authority* algoritam iskorištava vrijednost identiteta, što znači da čvorovi koji potvrđuju blokove ne stavljaju ulog, već vlastiti ugled. Stoga su lanci blokova osigurani od strane validacijskih čvorova koji su proizvoljno odabrani kao pouzdani entiteti. Oslanja se na ograničeni broj čvorova koji potvrđuju blokove i to je ono što ga čini visoko skalabilnim sustavom. Blokove i transakcije ovjeravaju unaprijed odobreni sudionici, koji djeluju kao moderatori sustava. Ukoliko sudionik ne potvrdi blok prema odgovarajućim pravilima biva izbačen iz mreže te više ne može sudjelovati u procesu. Algoritam se može primijeniti u raznim scenarijima i smatra se značajnom opcijom za logističke aplikacije. Na primjer, kad je riječ o lancima opskrbe, smatra se učinkovitim i razumnim rješenjem. Omogućuje tvrtkama da zadrže svoju privatnost, dok iskorištavaju prednosti tehnologije lanca blokova. Microsoft Azure je još jedan primjer u kojem se provodi *Proof of Authority*. U nekoliko riječi, platforma Azure nudi rješenja za privatne mreže, sa sustavom koji ne zahtijeva izvornu valutu kao oblik nagrade.

2.2.4. Practical Byzantine Fault Tolerance

Practical Byzantine Fault Tolerance najčešća je metoda postizanja konsenzusa u privatnim sustavima lanaca blokova i jedno je od rješenja prethodno spomenutog problema bizantskih generala. Problem bizantskih generala se događa u raspodijeljenim računalnim sustavima gdje ne postoje informacije o tome koji su čvorovi zlonamjerni. Kako bi raspodijeljeni sustavi mogli funkcionirati, ispravni čvorovi moraju postići konsenzus unatoč prisutnosti zlonamjernih čvorova. Kao što je opisano u [8], algoritam funkcionira na sljedeći način: klijent šalje zahtjev za pozivanje na uslugu primarnom čvoru, nakon čega primarni čvor duplicira zahtjev te ga šalje pomoćnim čvorovima. Čvorovi izvršavaju zahtjev te šalju odgovor klijentu. Klijent čeka f[1] + 1 odgovor od različitih čvorova s istim rezultatom, što predstavlja rezultat operacije. Kao što je prikazano na slici 2.1, s obzirom da postoji vjerojatnost da određeni čvorovi šalju neispravne informacije, koristi se trofazni protokol za postizanja koncensusa, odnosno provjerava se svaka replika informacije.

[1] f – predstavlja maksimalan broj čvorova koji mogu biti zlonamjerni

Od čvorova se očekuje da su deterministički i da se svi pokreću se iz istog stanja. Konačni rezultat je da svi ispravni čvorovi postignu dogovor o redoslijedu zapisa. Primarni čvor se mijenja na principu *Round-robin* algoritma, a može se čak i zamijeniti ako je proteklo određeno vrijeme bez da je primarni čvor duplicirao zahtjeve pomoćnim čvorovima. Većina čvorova također može odlučiti je li primarni čvor neispravan i zamijeniti ga sljedećim čvorom u redu.



Slika 2.1. *PBFT validacija pomoću trofaznog procesa*

2.2.5. Proof of Elapsed Time

Proof of Elapsed T*ime* (PoET) je konsenzusni algoritam kojeg je razvila tvrtka Intel, a temeljen je na tehnologiji Software Guard Extension (SGX) te implementiran u *open source* Hyperledger Sawtooth *framework*. Kao što je opisano u [9], PoET je oblik P*roof of Work* konsenzusa koji eliminira rasipnu potrošnju energije za razliku od njegovog izvornog algoritma PoW-a. Tradicionalni PoW može se smatrati lutrijom gdje su šanse za pobjedu proporcionalne količini utrošenog računalnog rada. Pobjednik je nedeterministički, dok je vjerojatnost pobjede proporcionalna računalnom ulaganju, što znači da bilo koji čvor može prvi pronaći rješenje i objaviti blok. Ova nasumičnost pomaže spriječiti da jedna od strana sustavno kontrolira objavljivanje blokova. Slično tomu, PoET stvara lutrijski sustav za objavljivanje blokova. No, za razliku od PoW-a korištenje računalne snage je smanjeno tako što je svakom čvoru dodijeljeno nasumično vrijeme čekanja uzorkovano eksponencijalnom raspodjelom, a generirano kodom koji se izvodi unutar *trusted execution environment* (TEE). Svi validacijski čvorovi sastavljaju transakcije u blokove koje nakon isteka vremena čekanja emitiraju kroz mrežu, ukoliko do tada niti jedan čvor nije objavio svoj blok. Što znači da su blokovi svih čvorova samo kandidati za objavljivanje čije prvenstvo ovisi o najkraćem vremenu čekanja. Stoga se u slučaju objavljivanja dva ili više blokova istovremeno, favorizira onaj čije je vrijeme čekanja najniže.

Za rješavanje složenih matematičkih funkcija PoW algoritma ne postoje prečice, jer je njihovo rješavanje jedino moguće uporabom grube sile (eng. *brute force*) koja onemogućava lažiranje procesa rudarenja. Za razliku od PoW algoritma koji koristi grubu silu za zaštitu procesa rudarenja, PoET algoritam koristi SGX tehnologiju koja omogućuje TEE. TEE stvara sigurna područja glavnog procesora koja štite kod na razini aplikacije, čak i od procesa koji se izvode na višim razinama privilegije. Kriptografske potvrde proizvedene od TEE potvrđuju da je specifičan kod izvršen unutar TEE te da rezultat nije ugrožen. Svaki čvor koji sudjeluje u PoET algoritmu mora biti opremljen tim specijaliziranim sklopovljem. Nasumično se vrijeme čekanja kreira pomoću entropije unutar TEE i kriptografski potpisuje tako da ga mogu provjeriti svi čvorovi koji koriste PoET algoritam.

2.3. Kriptografija

Kriptografija je prema [10], znanost čuvanja informacija od potencijalnih neprijatelja, hakera ili javnosti. Pošiljatelj šifrira poruku s malim dijelom tajnog podatka (eng. *key*), a zatim šalje šifriranu poruku primatelju. Primatelj dešifrira šifriranu poruku pomoću tajnog podatka, odnosno ključa koji je jednak ili različit od ključa koji je koristio pošiljatelj, kako bi dobio originalnu poruku. U tom slučaju, potencijalni napadači ne mogu pročitati poslani podatak čak i ukoliko presretnu šifriranu poruku. Ukoliko se ključ, pomoću kojeg se podatak šifrirao, koristi i prilikom dešifriranja onda govorimo o kriptografiji tajnog ključa odnosno simetričnoj kriptografiji. Ukoliko je poruka šifrirana pomoću javnog ključa primatelja, a dešifrirana pomoću tajnog ključa primatelja onda govorimo o kriptografiji javnog ključa odnosno asimetričnoj kriptografiji. Usprkos svojoj brzini i efikasnosti nasuprot asimetrične kriptografije, simetrična kriptografija posjeduje nedostatke kao što su primjerice razmjena tajnih ključeva, održavanje tih ključeva u velikim sustavima te nemogućnost osiguravanja integriteta odnosno autentičnosti podataka. S obzirom da se asimetrična kriptografija uvelike koristi unutar lanca blokova, ona će biti detaljno razrađena u nastavku.

2.3.1. Asimetrična kriptografija

Asimetrična kriptografija [11], poznata kao kriptografija javnim ključem, proces je koji koristi par povezanih ključeva: jedan javni ključ i jedan privatni ključ. Ona koristi javni ključ za šifriranje poruka, a privatni za dešifriranje i zaštitu od neovlaštenog pristupa ili uporabe. Javni ključ je kriptografski ključ koji svaka osoba može koristiti za šifriranje poruke, a dešifrirati ga može samo primatelj kojemu je poruka namijenjena svojim privatnim ključem. Privatni ključ, poznat i kao tajni ključ, dijeli se samo s inicijatorom ključa, a poruke se mogu šifrirati i dešifrirati na dva načina.

Prvi način je šifrirati poruku javnim ključem primatelja iz javnog imenika prije slanja, koju primatelj dešifrira svojim privatnim ključem. Dok je drugi način šifrirati poruku privatnim ključem pošiljatelja, koja se onda dešifrira korištenjem javnog ključa tog pošiljatelja, te na taj način pošiljatelj potvrđuje svoj identitet. Mnogi se protokoli oslanjaju na asimetričnu kriptografiju, uključujući transportni sloj sigurnosti (eng. *transport layer security*) i sloj sigurne komunikacije (eng. *secure socket layer*) koji omogućuju HTTPS (eng. *Hypertext Transfer Protocol Secure*). Postupak šifriranja koristi se i u programskoj podršci, kao što je preglednik (eng. *browser),* koji kreira i potvrđuje digitalni potpis, te uspostavlja sigurnu vezu preko nesigurne mreže poput Interneta. Povećana sigurnost podataka glavna je prednost asimetrične kriptografije. Najsigurniji je postupak šifriranja, jer korisnici nikada ne otkrivaju i ne dijele svoje privatne ključeve, čime se smanjuju šanse da potencijalni napadač otkrije privatni ključ korisnika tijekom prijenosa. Prednost asimetričnih algoritama je u tome što ih je teže „razbiti" nego simetrične, iako su u prosjeku deset tisuća puta sporiji od simetričnih. Jedan od prvih primjera asimetričnih algoritama je Diffie-Hellman razmjena tajnih ključeva, a jedan od najkorištenijih algoritama asimetrične kriptografije je RSA algoritam.

Diffie-Hellman

Kao što je rečeno, Diffie-Hellman algoritam jedan je od prvih primjera asimetričnih algoritama. Kreirali su ga kriptografi Whitfield Diffie i Martin Hellman koji u svom radu [12] govore da je za postizanje velikih i sigurnih telekomunikacijskih sustava nerealistično za očekivati da će korisnici, koji se ne poznaju, imati vrijeme potrebno za prenošenje ključeva preko sigurnih kanala. Također, govore, da je nerealistično očekivati da će svi ključevi biti unaprijed raspoređeni unutar sustava. Stoga su izradili sustav koji će usprkos korištenju isključivo javnih tehnika i kanala komunikacije biti siguran. Diffie-Hellman algoritam zasniva se na matematičkom principu komutativnosti potenciranja:

$$(x^y)^z = (x^z)^y \tag{2-1}$$

koja također vrijedi i kod kongruencije (modulo aritmetike):

$$(x^y \bmod p)^z = (x^z \bmod p)^y \tag{2-2}$$

odnosno:

$$\text{a}(x^y)^z \bmod p = (x^z)^y \bmod p \tag{2-3}$$

Proces razmjene ključeva između dvije strane (Alice i Bob) započinje nasumičnim odabirom cijelog broja q i broja α koji je primitivni korijen broja q. Ovi nasumično generirani brojevi se

nalaze u javnom kanalu te su stoga dostupni napadačima. U sljedećem koraku, Alice izabire cijeli broj a koji se nalazi unutar intervala [1, q -1] i taj broj čuva tajnim. Isto će to napraviti i Bob te generirati cijeli broj b. Nakon odabira tajnog broja, svaki od korisnika obavlja sljedeću operaciju te dobivenu vrijednost šalje drugom korisniku.

$$Alice\colon\ X = \alpha^{\,a}\ mod\ q \tag{2-4}$$

$$Bob\colon\ Y = \alpha^{\,b}\ mod\ q \tag{2-5}$$

Nakon primitka nove generirane vrijednosti, svaki od korisnika potencira tu vrijednost svojim tajnim ključem.

$$Alice\colon\ Y^{\,a} = (\alpha^{\,b})^{\,a}\ mod\ q = \alpha^{ab}\ mod\ q \tag{2-6}$$

$$Bob\colon\ X^{\,b} = (\alpha^{\,a})^{\,b}\ mod\ q = \ \alpha^{ab}\ mod\ q \tag{2-7}$$

Kao što je vidljivo na primjeru iznad, i Alice i Bob su zbog komutativnosti potenciranja dobili jednake vrijednosti, odnosno zajednički tajni ključ. Iako je napadač, prisluškivanjem javnog kanala saznao vrijednosti q, α, $\alpha^{\,a}\ mod\ q$, $\alpha^{\,b}\ mod\ q$, da bi saznao tajni ključ mora ili riješiti problem diskretnog logaritma, što je gotovo nemoguće, ili saznati tajne vrijednosti a i b. Jedan od problema Diffie-Hellman algoritama je taj što ne obavlja autentifikaciju korisnika. Taj problem može biti riješen implementacijom mehanizma za autentifikaciju, što podrazumijeva uporabu digitalnog potpisa i certificiranih javnih ključeva.

RSA algoritam

RSA algoritam, kao jedan od najkorištenijih algoritama asimetrične kriptografije, sastoji se od para javnog i privatnog ključa. Kao što je opisano u [13], sigurnost ovog algoritma temelji se na problemu faktorizacije velikih prirodnih brojeva. Otvoreni tekst (eng. *plaintext*) i šifrat cijeli su brojevi između nula i $n - 1$ za određenu vrijednost n, najčešće 1024 bita odnosno $n < 2^{1024}$. U kriptosustavu RSA par ključeva generira se tako da se slučajnim odabirom izaberu dva različita prosta broja p i q, pomoću kojih se izračunava vrijednost n. Iako je vrijednost n javna, brojevi p i q ostaju tajni kako bi se osigurala težina faktorizacije broja n.

$$n = p \cdot q \tag{2-8}$$

$$\varphi(n) = (p - 1)(q - 1) \tag{2-9}$$

Nakon toga se odabire cijeli broj d, tako da vrijedi:

$$d < \varphi(n) \tag{2-10}$$

$$(d, \varphi(n)) = 1 \tag{2-11}$$

U završnom koraku se računa vrijednost cijelog broja e pomoću vrijednosti p, q i d tako da bude recipročan broju d, $mod\ (p-1)\cdot(q-1)$.

$$e \cdot d \equiv 1\ (mod\ (p-1)\cdot(q-1)) \tag{2-12}$$

Pri tome se dobiva javni i privatni ključ:

$$PU = \{e, n\},\ PR = \{d, n\} \tag{2-13}$$

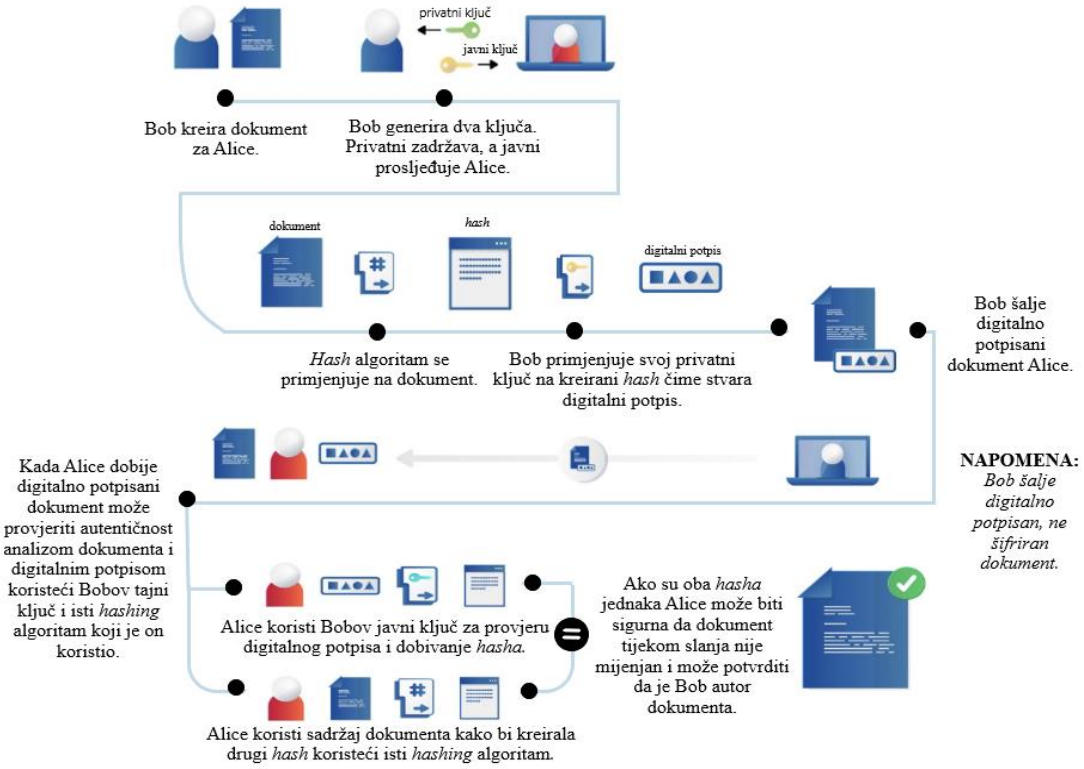
2.3.2. Hash

Hashing funkcija [14] jedna je od metoda kriptografije koja pretvara niz podataka proizvoljne dužine u kriptografski fiksni izlaz. Taj niz podataka može varirati od jednog znaka do niza znakova beskonačne dužine, no izlaz funkcije u svim slučajevima jednake je veličine. *Hashiranje je* jednosmjerna kriptografska funkcija, čijim dešifriranjem ne možemo doći do izvornih podataka. On se koristi unutar lanca blokova za povezivanje prethodnih blokova. Unutar prvog bloka u lancu (eng. *genesis block*), *hash* se računa pomoću transakcije tog bloka. Prilikom kreiranja drugog bloka, koristi se *hash* prethodnog bloka i transakcija tog bloka kao ulaz u funkciju. Na taj način kreira se lanac blokova u kojem *hash* novog bloka pokazuje na blok prije njega. Ovakav sustav osigurava permanentnost svih transakcija. Ukoliko se dogodi i najmanja promjena unutar određenog bloka, ona dovodi do promijene *hash* vrijednosti tog bloka kao i svih sljedećih blokova čime oni više nisu validni.

2.3.3. Digitalni potpis

Ključan čimbenik poslovnih aplikacija razmjena je poruka. Prema tome, primatelju je potrebno jamstvo da je poruka stigla od predviđenog pošiljatelja netaknuta, odnosno ne modificirana. Potpis osobe na određenom ugovoru jamstvo je koje veže sporazum. Postoje ručno pisani potpisi te oni digitalni. Ručno pisani potpisi svoju valjanost jamče rukopisom, budući da svaka osoba ima jedinstveni rukopis, poput otiska prsta. Znanost koja se bavi proučavanjem rukopisa naziva se grafologija, a grafolozi lako mogu otkriti razliku između valjanog potpisa i onog krivotvorenog. Slično tome, digitalni potpis je kriptografska tehnika koja povezuje osobu s digitalnim podacima,

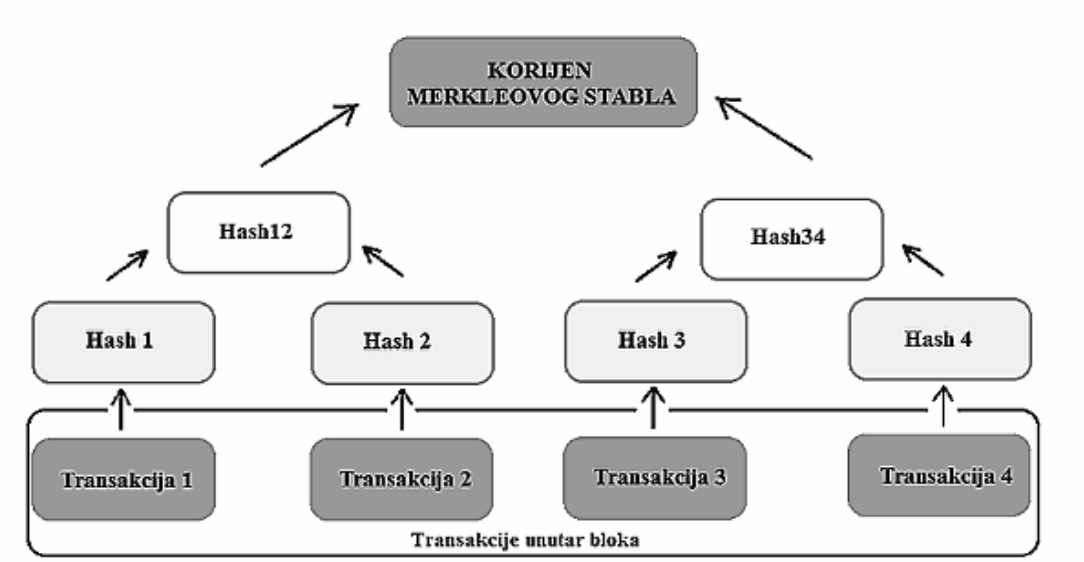
odnosno osigurava da je poruku kreirala određena osoba, a da je sama poruka netaknuta i neizmijenjena tijekom prijenosa. Digitalni potpis jedan je od glavnih segmenata sigurnosti i integriteta podataka koji je zabilježen na lancu blokova. On je standardni dio protokola lanca blokova, koji se uglavnom koristi za otkrivanje i sprječavanje bilo kakve promijene podataka neovlaštenih osoba, osiguravanje transakcija i blokova transakcija, prijenos informacija i upravljanje ugovorima. Digitalni potpisi koriste asimetričnu kriptografiju, što znači da se informacije mogu podijeliti s bilo kojom željenom osobom, uporabom javnog ključa. Unutar lanca blokova, svaka transakcija se digitalno potpisuje privatnim ključem vlasnika, što osigurava vlasniku da samo on može rukovati podatcima, poput transakcije novca s računa. Digitalni potpis ključna je prednost spremanja i prijenosa informacija u lancu blokova. Prije svega, on jamči integritet. Ukoliko dođe do promjene podataka koji se šalju, a zaštićeni su digitalnim potpisom, on postaje nevažeći. Digitalni potpis ne štiti samo podatke, već osigurava i integritet identiteta pojedinca koji ih šalje. Kao što je vidljivo na slici 2.2 po uzoru na [15], Bob je primijenio *hashing* algoritam za određeni dokument kako bi dobio izlaz fiksne duljine odnosno *hash*. Zatim, koristeći svoj privatni ključ stvara digitalni potpis. Alice nakon primitka potpisanog dokumenta, može potvrditi identitet Boba pomoću njegovoj javnog ključa i istog *hashing* algoritma kojeg je koristio.



Slika 2.2. *Prikaz korištenja digitalnog potpisa*

2.3.4. Merkleovo stablo

Merkleovo stablo (binarno *hash* stablo) jedan je od temelja tehnologije lanca blokova. Kao što je opisano u [16], ova struktura omogućuje učinkovitu i sigurnu provjeru sadržaja u velikom skupu podataka. Također, struktura pomaže u provjeri dosljednosti i sadržaja podataka. Merkleovo stablo sažima sve transakcije u bloku tako što stvara digitalnu presliku cijelog skupa transakcija i na taj način omogućuje korisniku provjeru transakcije. Nastaje višestrukim *hashiranjem* parova čvora dok ne ostane samo jedan *hash* odnosno korijen Merkleovog stabla. Svaka prethodna razina sadrži dvostruko više čvorova od sljedeće. Čvorovi se grade iz *hasheva* pojedinačnih transakcija od dolje prema gore, kao što je vidljivo na slici 2.3.



Slika 2.3. *Prikaz Merkleovog stabla*

Kao što se vidi na slici 2.3, postoje četiri transakcije u bloku. Svaka od njih je *hashirana* te pohranjena u listove stabla. Nakon čega se tvore roditeljski čvorovi koji nastaju konkateniranjem Hasha 1 i Hasha 2, čime se dobiva Hash12, te odvojenim konkateniranjem Hasha 3 i Hasha 4, čime se dobiva Hash34. Na kraju se dva dobivena *hasha*, Hash12 i Hash34, konkateniraju, čime se stvara korijen Merkleovog stabla. Svaki list stabla je *hash* transakcijskih podataka, a svaki čvor koji nije list je *hash* njegovih prethodnih *hasheva*. Merkle stabla su binarna i stoga zahtijevaju ravnomjeran broj listova odnosno 2^n čvorova. U slučaju da je broj transakcija neparan, odnosno različit od 2^n, posljednji *hash* bit će dupliciran jednom kako bi se stvorio parni broj čvorova lista. Korijen Merklovog stabla sažima sve podatke povezane transakcijama te se pohranjuje u zaglavlje bloka i služi za održavanje integriteta podataka. Ukoliko se redoslijed transakcija promijeni ili bilo koji detalj u bilo kojoj od transakcija, mijenjat će se i Merkleov korijen. Upotreba stabla Merkle omogućava brzu i efikasnu pretragu određenih transakcija. Merkleovo stablo razlikuje se od *hash*

liste po tome što se Merkleovim stablom može preuzeti jedina grana te odmah provjeriti njezin integritet, čak i ukoliko ostatak stabla još nije dostupan. Ova metoda vrlo je korisna, jer omogućava da se datoteke podijele u male blokove podataka. Na taj način pruža mogućnost ponovnog preuzimanja manjih blokova ukoliko se originalna inačica ošteti. Merkleovo stablo može značajno smanjiti količinu podataka koji se moraju održavati u svrhu verifikacije, a na taj način se razdvaja validacija podataka od samih podataka. Merkleova stabla imaju tri glavne prednosti:

- Omogućuju dokazivanje integriteta i valjanosti podataka
- Za dokazivanje zahtijevaju malu količinu informacija
- Zahtijevaju vrlo malo memorije

Sposobnost dokazivanja da je glavna knjiga cjelovita i dosljedna ključna je za tehnologiju lanca blokova. Merkleova stabla pomažu u provjeri sadrže li nove inačice zapisnika sve prethodne, jesu li svi podaci zabilježeni i jesu li prikazani kronološkim redoslijedom. Dokaz da je glavna knjiga dosljedna zahtijeva prikaz podataka u kojemu nijedan prethodni zapis nije dodan, izmijenjen ili neovlašteno uređen. Merkleovo stablo koristi rudarima (eng. *miners*) jednako kao i korisnicima na lancu blokova. Rudar može izračunati *hasheve* postepeno za vrijeme primanja transakcija od čvorova, a korisnik može pojedinačno verificirati dijelove blokova i provjeravati pojedinačne transakcije pomoću *hasheva* drugih grana stabla. Merkleova stabla snažan su i neophodan alat za rudare i korisnike na lancu blokova. Izuzetno su robusni i nalaze se u središtu nekoliko *peer-to-peer* mreža poput *BitTorrent, Git, Bitcoin* i *Ethereum.*

2.4. Struktura bloka

Blok je struktura podataka koja objedinjuje transakcije radi uključivanja u lanac blokova. Kao što je prikazano u tablici 2.1, blok se sastoji od magičnog broja koji služi za određivanje početka i kraja određenog bloka. Svaki blok u istoj mreži sadrži isti magični broj, primjerice magični broj Bitcoin lanca blokova prikazan je u tablici 2.1. Jedna od sastavnica strukture bloka je i veličina bloka, trenutna veličina bloka u Bitcoin mreži je 1 MB, dok u Ethereum mreži nema ograničenja veličine bloka već nju određuje maksimalna količina „gasa“. „Gas“ je jedinica koja mjeri količinu računalne snage koja je potrebna za izvršavanje određene operacije. Također, blok sadrži zapise koji predstavljaju sadržaj bloka te brojač zapisa. On sadrži broj transakcija u trenutnom bloku, a može biti predstavljen cijelim brojem veličine od 1 do 9 bajtova. Zaglavlje bloka je detaljnije objašnjeno u nastavku.

Tablica 2.1. Struktura bloka

Naziv	Opis	Veličina
Magični broj	0xD9B4BEF9	4 bajta
Veličina bloka	Veličina bloka u bajtovima	4 bajta
Zaglavlje bloka	Meta-podaci o bloku	80 bajtova
Brojač zapisa	Koliko zapisa sadrži blok	1-9 bajtova
Zapis	Zapisi pohranjeni u bloku	Varijabilno

2.4.1. Zaglavlje bloka

Kao što je opisano u [17], zaglavlje bloka, veličine 80 bajtova, služi za dobivanje dodatnih informacija o bloku te za povezivanje blokova u lanac. Inačica predstavlja trenutnu verziju korištenog protokola. *Hash* prethodnog bloka predstavlja vrijednost dobivenu *hashiranjem* prethodnog bloka u lancu, što predstavlja jedinstveni identifikator svakog bloka. Upravo ta referenca na prethodni blok sprječava promijene podataka u već kreiranim blokovima. Korijen Merkle stabla predstavlja *hash* svih transakcija unutar bloka, a vremenska oznaka predstavlja vrijeme kreiranja bloka. Težinska oznaka predstavlja vrijeme koje je potrebno za rudarenje određenog bloka, odnosno težinu rudarenja bloka. *Nonce* je cijeli broj koji rudari koriste kako bi dodali novi blok u lanac. Struktura zaglavlja prikazana je u tablici 2.2.

Tablica 2.2. Struktura zaglavlja bloka

Naziv	Opis	Veličina
Inačica	Inačica protokola u vrijeme nastajanja bloka	4 bajta
Hash prethodnog bloka	Referenca prethodnog bloka u lancu	32 bajta
Korijen Merkle stabla	Korijen binarnog stabla koje sadrži sve informacije transakcija u bloku	32 bajta
Vremenska oznaka	Vrijeme kada je blok kreiran	4 bajta
Težinska oznaka	Težina rudarenja bloka	4 bajta
Nonce	Broj koji pomaže rudarima	4 bajta

2.4.2. Povezivanje blokova

Rudari *hashiraju* zaglavlje bloka prilikom kojeg trebaju dobiti određen početni broj nula, koji zadaje težinska oznaka. Kao što je opisano u [18], za proces pronalaska odgovarajućeg *hasha* koristi se *nounce* koji se povećava sve dok rudar ne izračuna odgovarajuću vrijednost. Ukoliko dobiveni *hash* odgovara težinskoj oznaci, blok se dodaje na kraj lanca te objavljuje ostalim

rudarima koji mogu provjeriti odgovara li *nounce* dobivenom *hashu*. Na taj način, rudar stvara dokaz o radu. S obzirom da je potrebno 12 sekundi da rudar objavi svoje rješenje ostatku mreže, za to vrijeme drugi rudar može doći do rješenja te ga objaviti čime se stvara „fork“ odnosno razdvajanje lanca. Razdvajanje lanca će se riješiti tako što će se većina rudara prikloniti jednom od lanaca. Transakcije iz lanca koje nisu prihvaćene, vraćaju se natrag u „transaction pool“ te čekaju ponovnu potvrdu.

2.5. Pametni ugovori

Prema [19], pametni ugovor je digitalni transakcijski protokol koji izvršava uvjete ugovora. Opći ciljevi su zadovoljiti zajedničke ugovorne uvjete (poput plaćanja, povjerljivosti, pa čak i izvršenja), minimizirati slučajne i zlonamjerne iznimke, te isključiti potrebu za pouzdanim posrednikom. Ekonomski ciljevi uključuju smanjenje gubitaka, arbitražnih troškova, troškova izvršenja i drugih troškova transakcija, a isključuju prevaru. Pametni ugovori su digitalni ugovori koji omogućuju automatsko izvršavanje i samo provođenje nakon što su prethodno definirani uvjeti zadovoljeni. Samim time, pametni ugovori su sigurnija inačica stvarnih ugovora, jer su implementirani unutar lanca blokova koji ih štiti. Uz veću sigurnost, oni također eliminiraju potrebu za posrednicima, poput banaka i odvjetnika, što je omogućeno svojstvom nepromjenjivosti podataka unutar lanca blokova te ujedno ne zahtjeva povjerenje između strana koje komuniciraju. Pametni ugovor, kao i sam lanac blokova, uzima u obzir stanje glavne knjige te može pokrenuti događaj kad je to potrebno. Primjerice, ako je novac uplaćen, pametni ugovor može pokrenuti isporuku određenog proizvoda, no ukoliko nije ispunjen taj uvjet, pametni ugovor može pokrenuti određene financijske penale ili druge kazne. Postoje četiri koraka pri kreiranju pametnog ugovora:

- Postavljanje pravila razmjene dobara između dviju strana
- Zapisivanje ugovora unutar lanca blokova
- *Trigger event*[2]
- Provjera aktivnosti i rezultata definiranih ugovorom

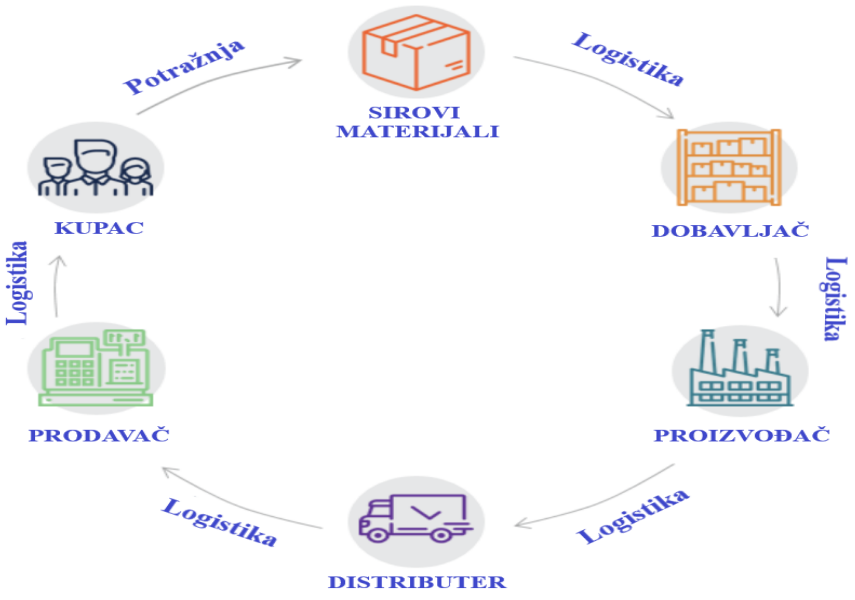
Uz pametne ugovore, često se koristi sustav *oracles.* On omogućava pametnim ugovorima prikupljanje informacija iz „stvarnog svijeta“. Postoji nekoliko oblika sustava *oracles,* primjerice *oracle* programske podrške i sklopovski *oracle. Oracle* programske podrške može prikupljati podatke od treće strane koristeći njihov API, poput oscilacije cijene goriva na tržištu. Sklopovski *oracle* može prikupljati podatke koristeći RFID čitač kako bi izmjerio temperaturu određenog područja. Tehnologija lanca blokova zajedno s pametnim ugovorima i *oracles* sustavima može

[2] *Trigger event* – događaj koji potiče izvršavanje ugovora prema dogovorenim pravilima.

pomoći, i u velikoj mjeri riješiti, primjerice probleme u trgovanju. Unatoč tehnološkom razvitku u mnogim područjima financijskih usluga, trgovina je i dalje uglavnom ručni proces koji se temelji na papiru i uključuje više sudionika iz različitih nadležnosti širom svijeta, sklonih ljudskim pogreškama i kašnjenjima unutar lanca opskrbe. Upravo se taj problem može riješiti korištenjem spomenutih sustava, jer decentralizirana knjiga bolje prati robu tijekom postupaka u kojem se roba otprema, skladišti i dostavlja. Može pratiti skladišna mjesta te provjeravati drži li se roba na odgovarajućoj temperaturi i slično. Omogućava svim strankama jednak pristup evidenciji transakcija, olakšava bržu verifikaciju i autentičnost proizvoda. Time se smanjuje vrijeme financijskih transakcija, kao i samog prijevoza. Potencijalna korist pametnih ugovora može biti ogromna, pogotovo ukoliko postoji mreža proizvođača, prijevoznika dobara, luka, carinskih vlasti te uvoznika u kojoj svi sudjeluju u stvarnom vremenu. Ipak, prema [20], postoje još uvijek potencijalna pitanja koja treba riješiti prije nego što pametni ugovori postignu raširenu upotrebu. Jedan od problema je skalabilnost, neisplativo je očekivati da će svaki čvor obraditi svaku transakciju ako se broj ugovora i korisnika drastično poveća. Drugi problem je ispravnost koda. Korisnici pametnih ugovora, kao i inženjeri koji su ih kreirali, moraju biti sigurni da ugovor izvršava svoju stvarnu namjenu. Također, postoji i pitanje odnosa između digitalnog pametnog ugovora i pravne strane, odnosno kako se na raspodijeljenom sustavu mogu kreirati pravno obvezujući ugovori u potencijalno više pravnih nadležnosti.

3. OPSKRBNI LANAC

Kao što je opisano u [21], opskrbni lanac je sustav proizvodnje i isporuke proizvoda ili usluge, od početne isporuke sirovina do konačne isporuke proizvoda ili usluge krajnjim korisnicima. Lanac opskrbe sadrži sve aspekte proizvodnog procesa, uključujući aktivnosti svakog stadija, odnosno informacija koje se prenose, prirodnih resursa koji se pretvaraju u korisne materijale, ljudskih resursa i drugih komponenata koje ulaze u konačan proizvod ili uslugu. Oblikovanje lanca opskrbe jedan je od ključnih koraka u provođenju vanjske analize u procesu strateškog planiranja. Važnost jasno postavljenog opskrbnog lanca je u tome što pomaže tvrtki da definira vlastito tržište i postavi dugoročni cilj tvrtke. Razvijajući strategije na korporativnoj razini, tvrtka često mora donositi odluke o tome hoće li raditi jednu djelatnost ili ulaziti u druge srodne ili nepovezane industrije. Svaka faza opskrbnog lanca u osnovi je različita industrija, primjerice, izvlačenje sirovina i proizvodnja. Lanac opskrbe omogućuje tvrtki razumjeti ostale sudionike uključene u svaki od stadija proizvodnje, pruža uvid u atraktivnost ili konkurentnost industrija u koje bi tvrtka poželjela ući u budućnosti. Opskrbni lanac započinje vađenjem i dobavljanjem sirovina. Sirovine se zatim logistikom dostavljaju dobavljaču, koji djeluje kao veletrgovac. Materijali se šalju do proizvođača, koji ih rafiniraju i prerađuju u gotov proizvod. Nakon toga proizvod odlazi kod distributera, koji ga dalje isporučuje prodavaču. Prodavač prodaje potrošačima proizvod u trgovini. U trenutku kada potrošač kupi proizvod, on završava ciklus. Prilikom čega se potražnja povećava te pokreće ponovnu proizvodnju, a samim time ciklus se nastavlja, kao što je prikazano na slici 3.1 napravljenoj po uzoru na [22].



Slika 3.1. *Prikaz opskrbnog lanca*

3.1. Prednosti opskrbnog lanca temeljenog na tehnologiji lanca blokova

Prema [23], zbog velikog broja sudionika koji su uključeni u velike opskrbne lance globalnih razmjera, odnosi među njima su vrlo složeni. Upravljanje opskrbnim lancem nosi mnogo problema utemeljenih na nepovjerenju, što je glavni uzrok implementacije lanca blokova. Prije svega, nedostatak povjerenja među sudionicima opskrbnog lanca zahtjeva da svaka uključena strana u procesu posjeduje točne podatke o razmjeni dobara među drugim stranama. Nažalost, to se ne događa u trenutnim lancima opskrbe gdje su teretni kvarovi, ljudska pogreška ili namjerne prevare česta pojava. Nadalje, vrlo je važno da takvi sustavi učinkovito prikupljaju valjane podatke, sigurno ih bilježe i pruže točne podatke različitim sustavima. Neuspjesi u takvim procesima se događaju često i rezultat toga je da globalni sustav opskrbnih lanaca ovisi o svojim krhkim vezama koje usporavaju, mijenjaju ili sprečavaju protok informacija. Osim toga, nedostatak visokog standarda i cjelovite integracije dovodi do neusklađenosti informacija među sudionicima. Svaka stranka mora prikupiti i potvrditi valjanost podataka kako bi se osigurala cjelovitost i učinkovitost takvog sustava čime se smanjuju performanse. Zbog složene arhitekture sustava, praćenje proizvoda i nadzor će dodatno oslabiti performanse i učinkovitost sustava.

Opskrbni lanci mogu iskorištavati brojne prednosti tehnologije lanca blokova kako bi povećali kvalitetu i smanjili broj nedostataka. Prema [24], lanac blokova može doprinijeti transparentnosti i preglednosti koje će u velikoj mjeri smanjiti kršenje uvjeta, ljudske pogreške i prevare. Uvjeti proizvoda mogu biti kontrolirani i prijavljeni glavnoj knjizi ukoliko dođe do ljudske pogreške ili bilo kakve štete unutar lanca opskrbe. Namjerna prevara će biti otkrivena i prijavljena na lanac blokova kao transakcija, dok će poruka biti emitirana unutar mreže kako bi obavijestila ostale strane o sudioniku koji je pokušao izvesti prevaru. Uz to, lanac blokova će osigurati kontinuitet informacija kroz svoja svojstva nepromjenljivosti i nepobitnosti. Sigurno dijeljenje podataka između različitih sudionika koji sudjeluju u globalnom opskrbnom lancu bit će nužno kako bi se zajamčilo praćenje proizvoda i smanjili rizici od pogrešaka ili prevara. Dijeljenje podataka lako će se provesti unutar javne ili privatne mreže lanca blokova koja osigurava nepromjenjivost podataka i međusobnu komunikaciju između strana koje si ne vjeruju.

Prema [25], vrlo velik broj sudionika sudjeluje u modernim lancima opskrbe, globalnih razmjera, zajedno s ogromnim protokom novostvorenih i vremenski osjetljivih informacija. U modernim opskrbnim lancima loše se upravlja tim podacima, jer u sustavu ne postoje zajedničke baze podataka. Lanac blokova može značajno doprinijeti skalabilnosti pružanjem umrežene i decentralizirane baze podataka kojom bi se sve stranke lanca opskrbe služile. Na taj način, eliminira se jedna točka kvara (eng. *single point of failure*), dok su istovremeno svi podaci

opskrbnog lanca zabilježeni u glavnoj knjizi, dijeljenoj među svim sudionicima mreže. Kao što je već rečeno, transportne i ljudske pogreške te namjerne prevare česte su u lancima opskrbe. Stoga je upravo sigurnost jedan od najvažnijih čimbenika svakog lanca opskrbe. Budući da su podaci koji cirkuliraju opskrbnim lancima u različitim oblicima te trebaju zadovoljiti različite potrebe njihova kontrola tijeka, osiguranje nepromjenjivosti i transparentnost između transakcija i podataka je sama po sebi zahtjevnija. Na primjer, zlonamjerna stranka koja sudjeluje u lancu opskrbe može krivotvoriti podatke o fakturi i promijeniti plaćene ili dospjele vrijednosti. Dakle, presudno je uključiti mehanizam koji će osigurati nepromjenljivost i povjerljivost transakcija u opskrbnom lancu. Upravo tehnologija lanca blokova može pružiti rješenje spomenutih sigurnosnih problema unutar lanaca opskrbe i jamčiti kontrolu integriteta i transparentnosti proizvoda te njihovih sadržaja. Ono što omogućava lancu blokova osiguravanje nepromjenjivosti podataka upravo je konsenzus. Njegovo je glavno svojstvo postići dogovor o svim poslanim transakcijskim informacijama između čvorova mreže. Transakcijske informacije mogu biti vremenske oznake, adrese pošiljatelja i primatelja, količina uplate, oznake ili elektronički pečati koji prate robu i slično.

Tijekom životnog ciklusa proizvoda unutar opskrbnog lanca njegove već spomenute mane također mogu uzrokovati i smanjenje performansi sustava. Koristeći lanac blokova povećavaju se performanse sustava isključujući linearni rast mogućih pogrešaka zbog toga što se većina aktivnosti može predstaviti kao transakcija zapisana u glavnu knjigu. Nakon što se aktivira pametni ugovor može u nekoliko minuta, neovisno o geografskoj lokaciji sudionika, automatski isplatiti određenim strankama točan iznos, što će biti prikazano u konceptualnom programskom rješenju koje slijedi u poglavlju broj 4. Također, unutar klasičnog opskrbnog lanca osiguravanje valjanosti, integriteta i povjerljivosti podataka dugotrajan je proces. On se može skratiti upravo implementacijom tehnologije lanca blokova koja sama po sebi sadržava svojstva valjanosti, integriteta i povjerljivosti. Još jedna od prednosti opskrbnih lanaca temeljenih na tehnologiji lanca blokova osiguravanje je privatnosti korisnika. Javni lanci blokova svojim korisnicima nude mogućnost pseudonima, što znači da je svaki korisnik u mogućnosti komunicirati s glavnom knjigom kroz novostvorenu adresu bez otkrivanja svog stvarnog identiteta. Privatni lanci blokova, kao i konzorcijski, pružaju mogućnost da se stranke pridružuju anonimno, ali unaprijed moraju biti autorizirane pomoću izvan mrežnog sustava opskrbnog lanca. Na taj su način pravi identiteti korisnika sačuvani u tajnosti od ostalih sudionika mreže, dok je s druge strane osiguran njihov legitimitet.

Osim anonimnosti koju opskrbni lanac temeljen na lancu blokova pruža svojim korisnicima, on im također pruža i geografsku neovisnost. Budući da se raspodijeljena mreža može dijeliti putem

Interneta, bilo koja stranka lanaca opskrbe može doprinijeti životnom ciklusu proizvoda s bilo kojeg mjesta na svijetu. Dakle, mreža opskrbnog lanca poboljšana tehnologijom lanca blokova može biti još veća na globalnoj razini od one tradicionalne, s obzirom da sudionici i tvrtke iz udaljenih i nepristupačnih dijelova svijeta također mogu sudjelovati u lancu opskrbe te nuditi svoje usluge ili kupovati proizvode [26]. Najvažnija stavka s obzirom na fleksibilnost lokacije, koju lanac blokova nudi opskrbnom lancu, upravo je vremenska učinkovitost. U slučaju tradicionalnih novčanih transakcija između klijenata na različitim dijelovima svijeta mobilnost isplate se smanjuje te može potrajati i mjesecima, ovisno o zakonima i međusobnim odnosima zemalja. Za razliku od njih, transakcija kriptovaluta putem globalne mreže lanca blokova, njena potvrda te završetak plaćanja, svodi se na minute. Na taj se način značajno smanjuje i trošak čime lanac blokova pruža ekonomsko rješenje za opskrbni lanac. Uz to, cijeli radni obim opskrbnog lanaca može biti znatno brži od tradicionalnog budući da se većina aktivnosti može predstaviti kao transakcija. Primjerice, cijela mreža lanca blokova u nekoliko minuta može saznati da je iskopavanje sirovina završeno, jer će završetak iskopavanja biti zapisan kao transakcija u glavnoj knjizi, a samim time korisnici mreže dobivaju informaciju da je sljedeći korak u povojima.

3.2. Trenutni doseg tehnologije lanca blokova u opskrbnom lancu

Većina je istraživačkih projekata lanca blokova još uvijek usmjerena na samu tehnologiju i aplikacije u financijskoj industriji, dok se interes za iskorištavanje tehnologije lanca blokova povećava i u proizvodnji. U nastavku je prikazana uloga tehnologije lanca blokova za Industriju 4.0 koja će omogućiti nove proizvodne procese. Osobito se razvija interes za primjenu lanca blokova u upravljanju i reviziji lanca opskrbe kojeg trenutno istražuje nekoliko *startup,* ali i velikih tvrtki. Primjeri takvih tvrtki i njihova podjela prikazana je u nastavku.

3.2.1. Tehnološke tvrtke

BlockVerify startup tvrtka osnovana 2015. godine [27]. BlockVerify tvrtka nudi tehnologiju lanaca blokova s mogućim rješenjima protiv krivotvorenih dijamanata, lijekova, elektronike i luksuznih predmeta. Jedno od rješenja je upravo mehanizam kojim tehnologija potvrđuje je li proizvod krivotvoren. On utvrđuje je li proizvod preusmjeren od izvornog odredišta te prati lažne transakcije. Također, može pronaći i locirati ukradene proizvode. BlockVerify platforma radi na globalnoj, digitalnoj glavnoj knjizi koja daje mogućnost praćenja proizvoda kroz cijeli lanac opskrbe. Svaki je proizvod označen s BlockVerfiy oznakom te verificiran tijekom cijelog opskrbnog lanca. Kako je platforma izgrađena na hibridnom tehničkom modelu korištenjem privatnog lanca blokova istodobno s bitcoin lancem blokova, omogućava transparentnost

opskrbnog lanaca u mjeri u kojoj tvrtka klijent zahtijeva. BlockVerify koristi autentifikaciju na razini potrošača i prati promjene u vlasništvu nad proizvodom. Kada potrošač kupuje proizvod može provjeriti je li proizvod originalan, na isti način i trgovci koji preprodaju određeni proizvod mogu svojim mobilnim uređajima verificirati proizvod i utvrditi njegovo porijeklo i originalnost. Svaki proizvod ima trajno zabilježenu povijest na lancu blokova čime se osigurava autentičnost svakog proizvoda u svrhu transparentnosti i revizije. BlockVerify testira rješenja u pilot projektima sa švicarskom farmaceutskom tvrtkom i kozmetičkom tvrtkom kojoj je sjedište u Londonu [28].

Eximchain je startup tvrtka osnovana 2016. godine koja kreira rješenja za financijske opskrbne lance na restriktivnom hibridnom lancu blokova pomoću pametnih ugovora. Lanac blokova za male i srednje kupce te male i srednje dobavljače djeluje kao alat za optimizaciju opskrbnog lanca te pomaže u ostvarivanju pristupa povoljnim izvorima kapitala za rast poslovanja. Pametni ugovori omogućuju malim i srednjim poduzećima da brzo implementiraju prilagođena rješenja za svoj financijski opskrbni lanac. Rješenje se temelji na Ethereumu lancu blokova, konsenzusnom protokolu i modelu upravljanja na temelju kvadratnog glasanja kako bi se osiguralo praktično, vremenski ograničeno jamstvo sigurnosti za njihov hibridni model mreže. Eximchain omogućuje tvrtkama da se učinkovitije i sigurnije povežu, vrše transakcije i dijele informacije. Kao što je rečeno u [29], koristeći tehnologiju lanca blokova, Eximchain uklanja tradicionalne barijere u opskrbnom lancu i integrira velike i male aktere transparentnu i sigurnu globalnu mrežu.

Modum je startup tvrtka osnovana 2016. godine s eksperimentalnom aplikacijom temeljenom na lancu blokova za decentralizirane poslužitelje na Sveučilištu u Zürichu. Kao što je rečeno u [30], u farmaceutskoj industriji pojačana je regulativa koja zahtijeva nove standarde logistike, stoga im Modum danas nudi rješenja temeljena na lancu blokova. Tehnologija je razvijena za medicinske proizvode i proizvode osjetljive na temperature što tvrtkama pomaže da se lako usklade s regulativama unutar industrije. Za daljnju integraciju i proširenje svoje ponude usluga Modum dodatno surađuje i s drugim tvrtkama koje se bave poljem lanca blokova poput SAP-a [31].

Prema [32], VeChain tvrtku osnovao je Sunny Lu, bivši CIO (*Chief information officer*) Louis Vuittona za Kinu, 2015. godine. Tvrtka je nastala kao podružnica tvrtke Bitse, jedne od najvećih kineskih tvrtki koja se bavi lancem blokova. VeChain je u početku implementiran na Ethereum lancu blokova, ali 2018. godine počeo je rabiti vlastiti lanac blokova čime je stvoren VeChain Thor. Vechain Thor je projekt lanca blokova kojem je u cilju poboljšati neefikasnost i netransparentnost trenutnog lanca opskrbe. Kao što je rečeno u [33], VeChain Thor lanac blokova u teoriji može obraditi 10 000 transakcija u sekundi. On omogućuje tvrtkama i kupcima da prate robu na svakom koraku, od proizvodnje, tranzita pa do isporuke. Na taj su način sve uključene

strane sigurne gdje se proizvod proizvodi, kako se njime rukuje tijekom otpreme, kada je stigao kod potrošača te gdje je stigao. Proizvođači koriste pametne čipove koji rade na sličan način kao NFC i RFID tehnologija, a mogu koristiti i IoT (eng. Internet of things) aplikaciju za proizvode kojima je bitno pratiti temperaturu okoline. Nakon što se proizvod skenira, informacije se upisuju u glavnu knjigu Vechain Thora kojem mogu odmah pristupiti sve uključene strane, čime se jamči sigurnost podataka.

3.2.2. Prehrambena industrija

IBM je pionir tehnologije lanca blokova koja kontinuirano pokušava poboljšati upravljanje lancima opskrbe. Walmart, za razliku od IBM-a, maloprodajni je gigant u Sjedinjenim Američkim Državama. Upravo te dvije tvrtke od listopada 2016. godine zajedno rade na pilot projektu praćenja prehrambenih proizvoda kroz opskrbni lanac temeljen na lancu blokova. Kao što je Frank Yiannas, potpredsjednik službe za sigurnost hrane Walmarta, rekao u [34], već u početcima razvoja opskrbnog lanca temeljenog na lancu blokova pronalaženje podrijetla prehrambenog artikala trajalo je 2,2 sekunde, dok je isti postupak primjenom starih metoda trajao gotovo sedam dana. Ovakva će učinkovitost pomoći smanjiti vrijeme odziva u slučaju zagađenja proizvoda kao i njegov opoziv. Projekt koristi IBM platformu lanca blokova razvijenu kroz suradnju s Hyperledger *open source* zajednicom uključujući najnoviju Hyperledger Fabric inačicu, koju podržava Linux Foundation. Lanac blokova sadrži potpunu povijest prehrambenih proizvoda kroz cijeli lanac opskrbe koji osigurava praćenje hrane „od farme do stola“. Kao što je rečeno u [35], lanac blokova stvara povjerenje potrebno za rješavanje osnovnih problema vidljivosti podataka, optimizacije procesa i upravljanje potražnjom u bilo kojem opskrbnom lancu. Privilegije ove tehnologije prepoznali su i giganti kao što su Dole, Driscoll's, Golden State Foods, Kroger, McCormick and Company, McLane Company, Nestlé, Tyson Foods i Unilever te su se već od kolovoza 2017. godine udružili s IBM-om kako bi pomogli u daljnjem prepoznavanju prioriteta novih područja u kojima lanac blokova može koristiti prehrambenom ekosustavu i pozitivno utjecati na globalno nadziranje hrane.

3.2.3. Prekooceanska prijevoznička industrija

Maersk, danska prekooceanska prijevoznička tvrtka, još jedan je od giganta koji je prepoznao prednosti tehnologije lanca blokova. Iako se eko-sustav oceanskog prijevoza kontinuirano mijenja i razvija, većina procesa zahtjeva ručno ispunjavanje papirologije. Prema Maersku jednostavna pošiljka između kontinenata zahtijeva marke i odobrenja više od 30 ljudi i organizacija koje se sastoje od otpremnika, carinskih vlasti, agenata, prijevoznika i špeditera te proces obuhvaća više

od 200 razmjena informacija među tim sudionicima. Stoga su se IBM i Maersk udružili te kreirali projekt pod nazivom "Cross-Border Supply Chain" kako bi riješili složene probleme tog područja pomoću tehnologije lanca blokova. Partnerstvo je namijenjeno razvoju rješenja koje će digitalizirati i upravljati otpremom kontejnera širom svijeta, što također omogućava razmjenu dobara u stvarnom vremenu pomoću transakcija unutar opskrbnog lanca. Kao što je rečeno u [36], lanac blokova je kreiran tako da pruža transparentnu zajedničku mrežu s visokom razinom sigurnosti u kojoj će svaki sudionik moći vidjeti napredak robe kroz opskrbni lanac zajedno sa statusom carinskih dokumenata, računa i drugih podataka. Platforma uz sve to nudi i suradnju te razmjenu dokumenata u stvarnom vremenu.

CargoX je tvrtka osnovana u kolovozu 2017. godine, a glavni cilj im je bio postati pionirima tehnologije lanca blokova u globalnoj brodskoj industriji. Njihov prvi proizvod "Smart Bill of Lading" (SBL), temeljen na decentraliziranoj aplikaciji (dApp), pokrenut je 2018. godine. Kao što je rečeno u [37], tvrtka se usmjerila na razvoj i uporabu otpremnica unutar lanca blokova za globalnu brodsku industriju. Otpremnica[3] (eng. *bill of landing*) je dokument koji predstavlja ugovor o otpremi i dokazuje vlasništvo nad teretom. SBL projekt digitalizira sve papirnate otpremnice pomoću lanca blokova čime se postiže poboljšanje globalne trgovine s obzirom na brzinu, sigurnost i transparentnost. Ova tehnika omogućuje skraćenje roka isporuke zbog toga što je izvozniku otpremnica odmah dostupna, a u roku od deset minuta vlasništvo se prenosi na zakonitog vlasnika robe bez potrebe za tradicionalnim dostavljačima. Ova tehnika prijenosa kroz lanac blokova onemogućuje slanje dokumenata na pogrešne adrese te smanjuje novčana izdavanja. SBL je ekvivalentan tradicionalnoj otpremnici i uključuje sve njegove značajke. Prednost SBL-a pred tradicionalnim otpremnicama je ta što nudi dodatnu pogodnost za svoje korisnike, nema potrebe za ispisom, slanjem, pohranjivanjem i arhiviranjem, jer aplikacija koja se koristi za SBL sama po sebi uključuje te stavke. Svaka se transakcija SBL-a prati i sprema u Ethereum lanac blokova koji pomaže uključene strane obavijestiti o procesu otpreme te statusu proizvoda. Budući da je otpremnica jedan od najvažnijih dokumenata globalne trgovine sigurnost ovog dokumenta je od velike važnosti. Kroz lanac blokova dokument se prvo šifrira te sigurno sprema na mrežu. S obzirom da se koristi Ethereum privatni lanac blokova, dokument je dostupan samo određenim korisnicima, u ovom slučaju trgovcima, čime se sprječavaju nezakonite radnje trećih strana i mogućnost gubitka ili krađe dokumenta. Korisnici platformi mogu pristupiti putem različitih uređaja kao što su računala, telefoni ili tableti. Kako bi se mogli registrirati, prijaviti i spremiti dokumente korisnici trebaju kupiti takozvani „trezor" koji je sličan uređaju za autentifikaciju pri

[3] Otpremnica – izraz korišten zbog nedostatka hrvatskog termina za izraz *bill of landing*

internetskom bankarstvu. Trezor se koristi kao zamjena za korisnikov otisak prsta ili privatni ključ pomoću kojeg je prijava i prijenos SBL-a moguć. Svaki korisnik ima jedinstveni otisak prsta i adresu putem trezora bez kojih nije moguć pristup računu. Nakon registracije na platformu korisnici mogu postaviti otpremnice u PDF formatu te ih poslati putem dApp-a do željenog primatelja. Kod slanja otpremnice dokument se prenosi putem lanca blokova nakon čega se automatski sprema u arhivu. Dokumente obično treba prvo ispisati, potpisati, skenirati, a zatim poslati primatelju. SBL skraćuje postupak obrade dokumenta, jer su dokumenti potpisani putem tehnologije lanca blokova koja jamči sigurnost i brzu isporuku do primatelja. Trenutno CargoX nudi samo pohranu i prijenos SBL-a putem dApp-a, međutim namjerava ponuditi daljnju integraciju značajki kao što su pametni ugovori i osiguranje. Štoviše, CargoX planira izravno integrirati dApp u postojeće tehnologije pomoću API-ja, a prednost daje većim tvrtkama s postojećim SAP sustavima.

3.2.4. Automobilska industrija

Prema [38], automobilski lanac opskrbe temeljen na lancu blokova može se implementirati kao logistički servis za proizvodnju i kao distribucijski servis za trgovce i uvoznike. U logističkom servisu prilikom proizvodnje glavni prioriteti su točna dostava auto dijelova i održavanje optimalne količine zaliha koji su omogućeni učinkovitom koordinacijom između logistike, prijevozničkih tvrtki i dobavljača. Proces se koordinacije može uvelike poboljšati korištenjem IoT integriranog lanca blokova. Sustav će koristiti pametne IoT senzore i pametne uređaje koji mogu pratiti primjerice lokaciju dijelova, njihovu količinu te druge korisne informacije u stvarnom vremenu. Ovakvo poboljšanje sustava dovodi do brojnih prednosti koje će utjecati na proizvodnju kao što je poboljšanje protoka materijala, informacija i robe te planiranje rasporeda proizvodnje. S druge strane prednosti koje utječu na dobavljače smanjenje su neispravnih narudžbi, troškova skladištenja te optimizacija količine zaliha. Također, u distribucijskim servisima za trgovce i uvoznike glavni je prioritet vrijeme isporuke automobila koje će se ostvariti učinkovitijom koordinacijom logistike i prijevoznika. Ovakav sustav proizvodnom pogonu poboljšava pravovremenu logistiku, kontrolu inventara te smanjuje broj oštećenih vozila, dok distributerima smanjuje skladišne troškove te skraćuje vrijeme isporuke od narudžbe do gotovog vozila.

Mercedes, jedan on najvećih proizvođača automobila na svijetu, potaknut inicijativom startupa Autobahn čiji je cilj identifikacija automobila nove generacije, udružio se s tvrtkom Circulor kako bi implementirali lanac blokova za praćenje emisije plinova i količine recikliranog materijala. Također kao što je rečeno u [39], prikupljene podatke ovog pilot projekta namjeravaju upotrijebiti za razvoj nove linije „carbon neutral“ automobila. Projekt pod nazivom Ambition2039,

planira smanjiti oslobađanje ugljikovog dioksida prilikom izrade i korištenja automobila, a u konačnici ukloniti ugljikov dioksid iz atmosfere. Početni je fokus projekta praćenje zalihe kobalta iz razloga što su etika njegovog vađenja i njegovo podrijetlo diskutabilni. Kobalt kao ključan mineral za proizvodnju litij-ionskih baterija presudan je za izradu električnih automobila. Većina tog minerala potječe iz Demokratske Republike Kongo u kojoj su uvjeti za rad tijekom vađenja kobalta kritizirani kao nemoralni. Kao što je rečeno u [40], UN je 2017. godine procijenio da je na svjetskoj razini 168 milijuna djece sudjelovalo u eksploatacijskim radovima, od kojih 40 tisuća sudjeluje u iskopavanju kobalta u Kongu. Ovakvi su podaci potaknuli tvrtke na razvoj opskrbnih lanaca temeljenih na lancu blokova, jer im je važno znati odakle dolaze njihovi materijali. Uporabom lanca blokova mogu pratiti emisiju plinova i proizvodni ciklus tih materijala te bilježiti količinu recikliranog materijala korištenog u lancu opskrbe. Upravo tako će tvrtka Mercedes imati mogućnost provjeriti krše li partneri zahtjeve održivosti, naročito po pitanju ljudskih prava.

Također, BMW grupacija, u suradnji s 10 različitih dobavljača, najavila je pokretanje opskrbnog lanca temeljenog na lancu blokova nazvanog PartChain. Razvoj PartChaina najavljen je za 2020. godinu, a koristi tehnologiju lanca blokova kako bi pružilo transparentnost i omogućio praćenje komponenti i sirovina korištenih za proizvodnju automobila. Prema [41], automobilski su lanci opskrbe složeni ekosustavi koji uključuju brojne sudionike u različitim fazama proizvodnje. Sudionici se mogu vrlo brzo promijeniti ovisno o potrebama određenog dijela industrije. Potrebe uključuju dostupnost sirovina, potražnju automobila u različitim regijama te opskrbu određenim komponentama željenog proizvođača. Velik je broj partnera u opskrbnom lancu obično upravljao vlastitim podacima što je rezultiralo nastankom mreža u kojima je teško pratiti podrijetlo ili put isporuke komponenata. Kako se lanac blokova pokazao izuzetno korisnim pri upravljanju lanca opskrbe brojni proizvođači u automobilskoj industriji počeli su iskorištavati prednosti te tehnologije. Primjeri uključuju Ford Motor i IBM koji koriste lanac blokova za praćenje kobalta koji se koristi za izradu baterija električnih automobila, dok Volkswagen pomoću Minespider lanca blokova prati slične sirovine. Mobility Open Blockchain Initiative konzorcij je koji uključuje BMW, a obavlja testiranje međunarodnog sustava identifikacije vozila koji se temelji na lancu blokova, dok ostali proizvođači automobila kao što su Hyundai Motor Group koriste tehnologiju za praćenje rabljenih automobila.

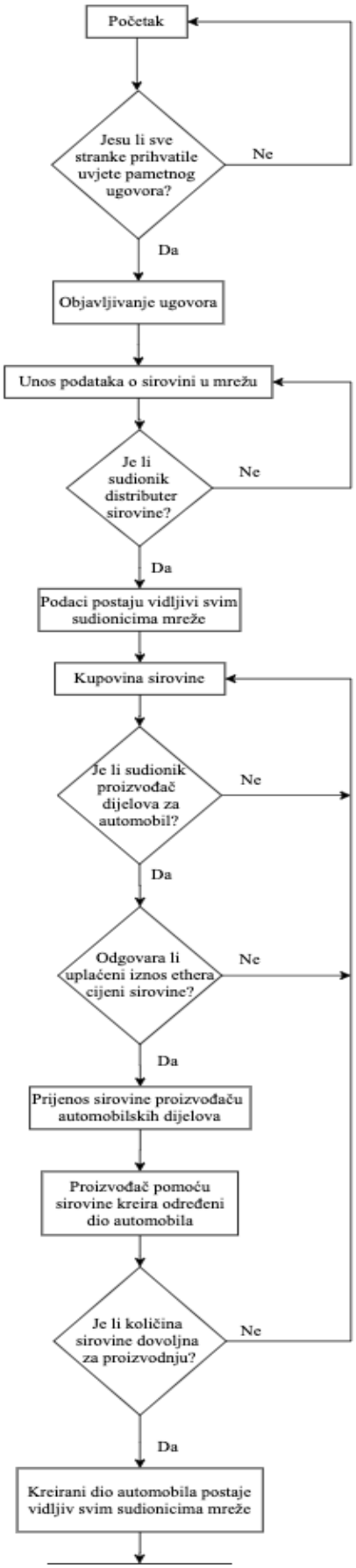
Po uzoru na prikazane primjere u sljedećem će poglavlju biti prikazano modeliranje opskrbnog lanca automobilske industrije temeljenog na tehnologiji lanca blokova. Cilj je razviti sustav koji će unaprijediti proizvodni ciklus automobila te omogućiti bolji protok materijala, informacija i robe između sudionika mreže.

4. MODELIRANJE SUSTAVA OPSKRBNOG LANCA TEMELJENOG NA LANCU BLOKOVA

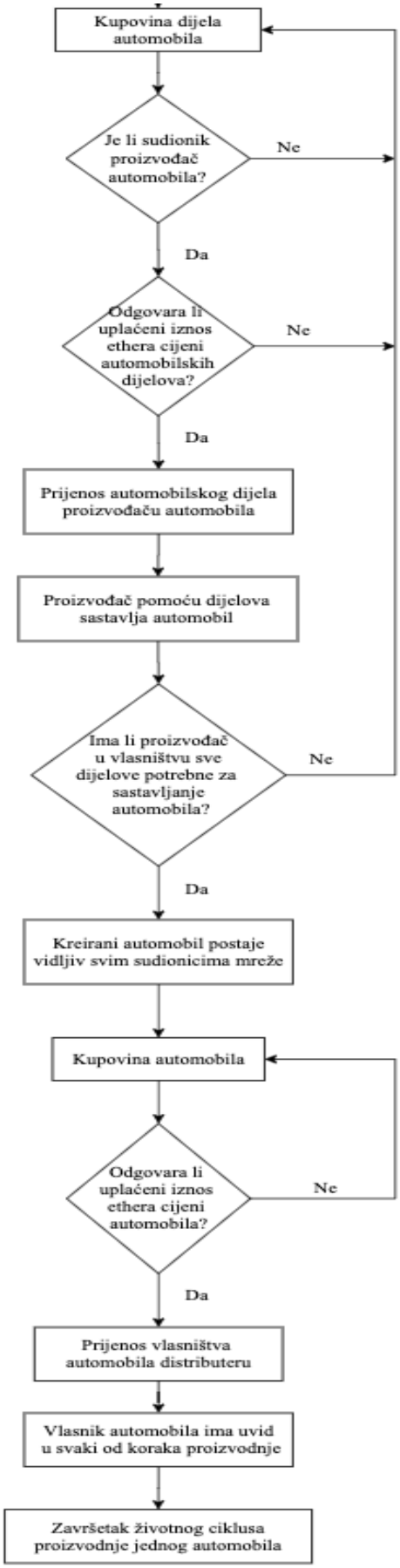
U ovom poglavlju će na temelju prethodno spomenutih prednosti tehnologije lanca blokova i nedostataka tradicionalnih opskrbnih lanaca biti kreiran model sustava. Model će uključivati sve osobine, informacije, sudionike i tehnologije potrebne za rad. Kreirani model prikazivat će životni ciklus automobila unutar sustava opskrbnog lanca automobilske industrije temeljenog na tehnologiji lanca blokova.

4.1. Prikaz sustava opskrbnog lanca temeljenog na lancu blokova

Kao što je rečeno u prethodnom poglavlju, nedostatak povjerenja, transparentnosti i točnosti podataka glavni su izazovi postojećih opskrbnih lanaca. Upravo implementacija tehnologije lanca blokova može riješiti te probleme i na taj način stvoriti sigurnu i transparentnu mrežu između svih uključenih strana. Stoga je cilj izraditi sustav opskrbnog lanca u automobilskoj industriji koji će pratiti cjeloviti ciklus izrade automobila. Opskrbni lanac započinje s distributerom sirovine, a završava s vlasnikom automobila. Također, opskrbni lanac će sadržavati i sudionike kao što su proizvođač automobilskih dijelova, proizvođač automobila te distributer automobila. Sustav će biti temeljen na pametnim ugovorima kojima je zadatak isključiti potrebu za pouzdanom trećom stranom, osigurati izvršavanje ugovornih uvjeta (plaćanje, povjerljivost) te smanjiti mogućnost prevare. Također, pametni ugovori mogu pokrenuti automatsku isporuku robe i prijenos vlasništva nakon uplate novca. Dijagram toka takvog sustava prikazan je na slikama 4.1 i 4.2. Kao što je i vidljivo na slici 4.1 prvi korak prije pokretanja opskrbnog lanca samo je ugovaranje pravila i uvjeta između uključenih sudionika. Nakon postizanja dogovora između svih strana sustav počinje s radom.



Slika 4.1. *Dijagrama toka od objavljivanja ugovora do kreiranja automobilskih dijelova*



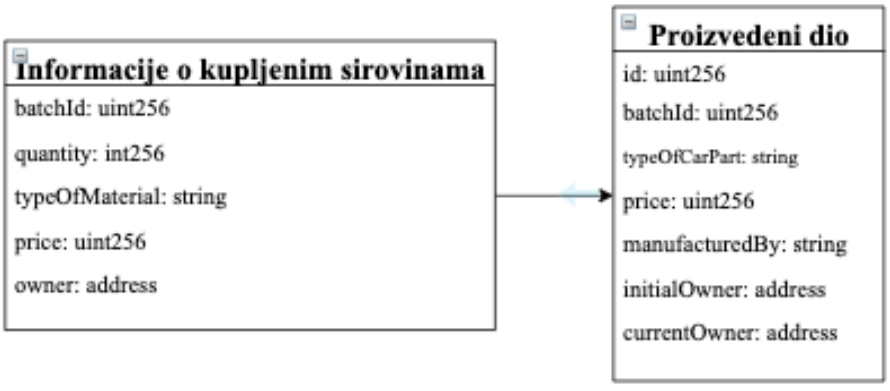
Slika 4.2. *Nastavak dijagrama toka do konačne kupovine automobila*

4.2. Prikaz informacija o proizvodima unutar opskrbnog lanca

Informacije o sirovinama i proizvodima unutar opskrbnog lanca dostupne su svim proizvođačima mreže uključujući i vlasnika automobila. Prilikom početnog unosa informacija o sirovini u mrežu, distributer mora unijeti informacije o sirovini kao što su vrsta sirovine, zemlja podrijetla, tvrtka koja je izvršila iskopavanje, cijena i količina. Prije nego se te informacije zapišu u lanac blokova, njima se dodaje jedinstveni broj identifikacije koji će služiti za postupak praćenja proizvodnje. Prikaz strukture sirovine vidljiv je na slici 4.3.

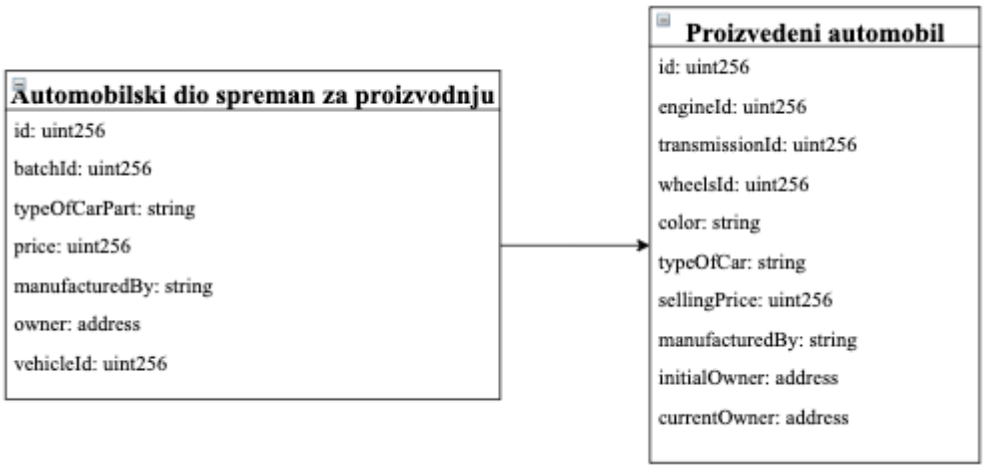
Slika 4.3. *Struktura sirovine*

Nakon što proizvođač automobilskih dijelova odluči kupiti određenu sirovinu za izradu dijelova, spomenuti se identifikacijski broj prenosi čime se stvara poveznica koja ukazuje na slijed proizvodnje. Nakon što sudionik proizvede određeni automobilski dio, to se upisuje na lanac blokova zajedno s informacijama o proizvođaču, vrsti dijela, cijeni te novim identifikacijskim brojem kao i s početnim identifikacijskim brojem zadanim od strane distributera sirovine. Prikaz informacija o sirovini i proizvedenom automobilskom dijeli vidljiv je na slici 4.4.



Slika 4.4. *Informacije o sirovini i proizvedenom automobilskom dijelu*

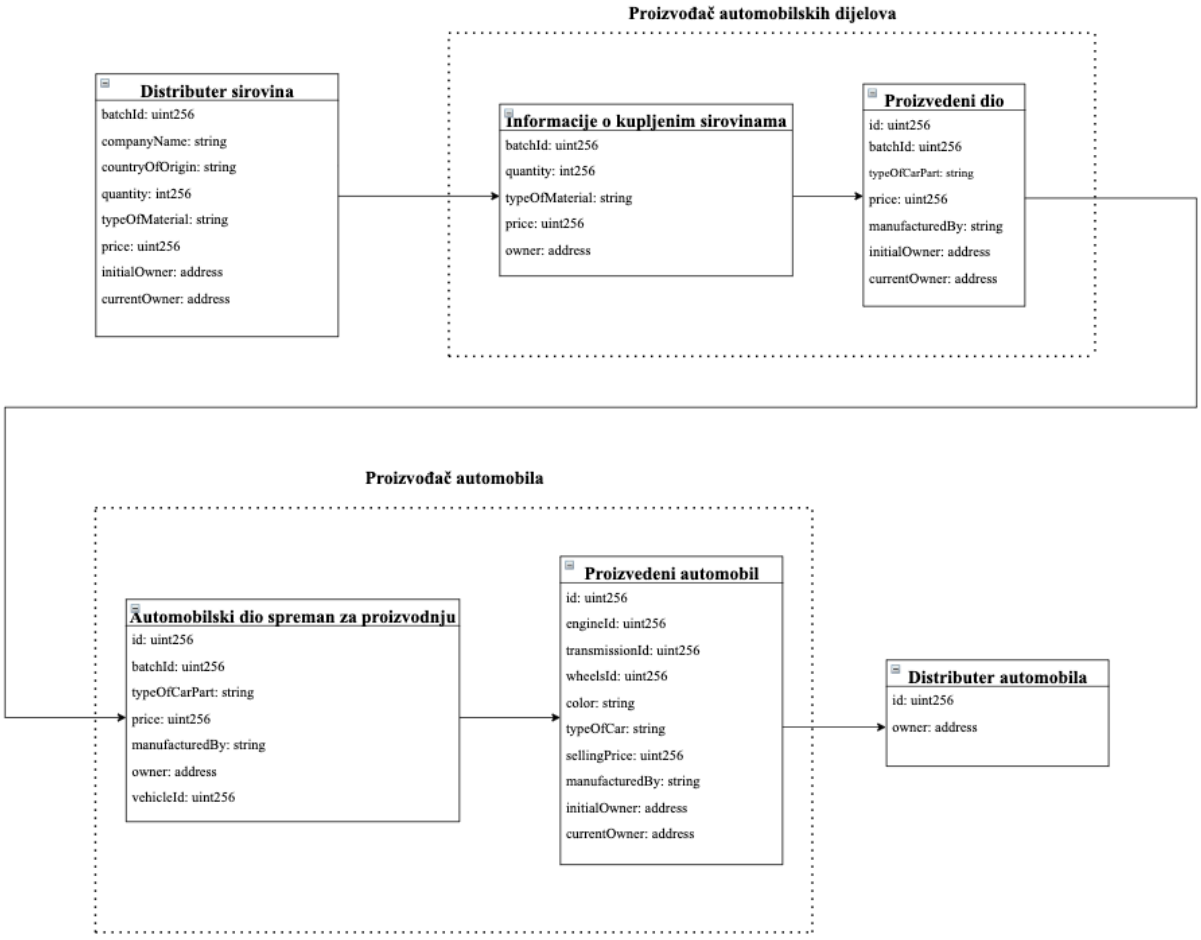
Prilikom kupovine automobilskih dijelova od strane proizvođača automobila vlasništvo tih dijelova se prebacuje s jednog sudionika na drugog te zapisuje na lanac blokova. Nakon što proizvođač kreira novi automobil, zapisuje potrebne informacije o automobilu na lanac blokova. Kao što su, identifikacijski broj automobila, motora, prijenosa i ovjesa, boju i tip vozila te informacije o proizvođaču te prodajnu cijenu. U već zadanu strukturu podataka od strane proizvođača dijelova proizvođač automobila dodaje identifikacijski broj automobila kako bi se osiguralo praćenje svih dijelova, što možemo vidjeti na slici 4.5.



Slika 4.5. *Prikaz informacija nakon proizvodnje automobila*

Kao što je vidljivo na slici 4.6, prilikom kupovine automobila od proizvođača distributer dobiva osnovne informacije o automobilu, kao što je identifikacijska oznaka vozila pomoću koje omogućuje budućem vlasniku praćenje cijelog životnog ciklusa, od početne sirovine do konačnog automobila, što je vidljivo na slici 4.7.

Slika 4.6. *Prikaz strukture kupljenog automobila*



Slika 4.7. *Prikaz redoslijeda informacija opskrbnog lanca od početne sirovine do distributera*

4.3. Sudionici opskrbne mreže temeljene na lancu blokova

Sudionici mreže su unaprijed određeni proizvođači i distributeri koji međusobno razvijaju pametne ugovore odnosno dogovaraju se oko načina funkcioniranja tih ugovora. Prvobitno svaki sudionik odnosno tvrtka u mreži mora posjedovati vlastiti kripto račun (eng. *wallet*). Kripto račun je vrsta virtualnog novčanika koji se kreira pomoću javnog i privatnog ključa, a služi za spremanje kripto valuta te obavljanje transakcija na lancu blokova. Svaki novčanik ima jedinstvenu javnu adresu koja se koristi za uplaćivanje kripto valute određenom sudioniku, a pomoću spomenutog privatnog ključa je omogućen pristup tim kripto valutama. Također, adresa novčanika se koristi prilikom kupovine kripto valute, koja se plaća stvarnim, fiat[4] novcem. Primjerice, cijena 1 Ethera u vrijeme pisanja ovog rada je iznosila 2,469.32 kune. Tek nakon što svaki sudionik mreže kreira vlastiti virtualni novčanik mreža se može podići. Svaki će sudionik mreže podići vlastiti ugovor na mrežu pomoću vlastitog novčanika čime se stvara poveznica prilikom kupovine određenog proizvoda i

[4] Fiat novac – službeno sredstvo plaćanja priznato od strane jedne države, primjerice dolari, euri, kune.

uplate na odgovarajući račun. Također, prilikom objavljivanja ugovora na mrežu, sudionici će koristiti adrese drugih objavljenih ugovora čime će se ugovori međusobno povezati te omogućiti automatiziranu komunikaciju između njih kao i prijenos novca. Na taj će način svaki prijenos vlasništva proizvoda kao i novca biti točan, siguran i automatiziran.

4.4. Testna mreža lanca blokova

Testna mreža lanca blokova [42] simulira stvarni rad mreže, a namijenjena je za testiranje i razvoj pametnih ugovora prije objavljivanja na glavnu mrežu lanaca blokova. Za razvijanje primjera bit će korištena Ethereum mreža lanca blokova. Glavna je prednost testne mreže ta što Ether kriptovaluta nema stvarnu vrijednost (eng. *faux-ether*), a može se dobiti besplatno. Za besplatno dobivanje kriptovalute na testnoj mreži koristi se slavina (eng. *faucet*) koja omogućava preuzimanje kriptovalute. Preuzeta kriptovaluta može se koristiti samo na mreži čija je slavina korištena. Najkorištenije tri testne mreže koje oponašaju rad sličan produkcijskom lancu blokova, prikazane su u tablici 4.1.

Tablica 4.1. Prikaz najkorištenijih Ethereum testnih mreža

Naziv testne mreže	Konsenzusni protokol	Podržane inačice čvorova
Ropsten	Proof of Work	Geth, Parity
RinkeBy	Proof of Authority	Geth
Kovan	Proof of Authority	Parity

Lanac blokova zahtijeva od svih sudionika *peer-to-peer* mreže da pokrenu implementaciju programskog rješenja podržane inačice čvora. Takvo programsko rješenje obično uključuje: potrebne protokole za širenje poruka i transakcija putem mreže, implementaciju virtualnog stroja lanca blokova, komponentu baze podataka koja se koristi za pohranu podataka u lanac blokova i sučelja koja omogućuju interakciju vanjskog programskog rješenja s lancem blokova. Geth i Parity su programska rješenja koja omogućuju pojedino pokretanje čvora na javnoj Ethereum mreži ili bilo kojoj drugoj mreži koja koristi Ethereum protokol. Obje su inačice trenutno održavane i aktivne, no Parity je stekao dobru reputaciju zbog svojih performansi i fleksibilnosti.

Testna mreža Ropsten upravo iz spomenutih razloga podržava obje inačice čvora. Ona najviše nalikuje glavnoj mreži lanca blokova, jer koristi isti konsenzusni algoritam što znači da su čvorovi zaduženi za održavanje mreže. Oni koriste računalnu snagu za generiranje blokova i ažuriranje, a blokovi se rudare u prosjeku svakih 30 sekundi. Na ovoj se mreži Ether može dobiti pomoću rudarenja ili već spomenutih slavina. Budući da se Ether može rudariti na testnoj mreži Ropsten, podložan je *spam* napadima, odnosno valovima beskorisnih transakcija koje zagušuju mrežu. S

obzirom da je Ether besplatan i lako ga je dobiti, lako je stvoriti ovakvo zagušenje. Jedan od takvih napada dogodio se u veljači 2017. godine kada su napadači iskopali velike količine Ethera i neprestano slali prevelike transakcije u mrežu. Granica veličine bloka Ethereuma dizajnirana je da bude fleksibilna i raste s potražnjom, stoga su napadači uspjeli povećati veličinu bloka od 4 milijuna jedinica *gasa*, koliko je tada bila, do nekoliko milijardi. Nakon toga poslali su velike, računski intenzivne, ali beskorisne transakcije u mrežu, potpuno začepivši ograničenje *gasa* i blokirajući rad svih ostalih sudionika mreže. Mreža se srušila i oživila mjesec dana kasnije nakon resetiranja podataka lanca blokova. Zanimljivo je primijetiti da se testna mreža Ropsten razlikuje od glavne mreže, na kojoj svi posjeduju svoj "pravi" Ether, samo po jednoj stavki – međusobnom dogovoru. Unatoč tome što je testna mreža Ropsten jednaka glavnoj mreži njen Ether je bezvrijedan samo zato što je zajednica tako odlučila. Na ovom primjeru možemo primijetiti kako konsenzus zajednice diktira vrijednost imovine, ili točnije, nedostatak vrijednosti imovine.

Testna mreža Kovan pokrenuta je 2017. godine od strane Parity tima nakon napada na testnu mrežu Ropsten. Kovan radi samo s Parity čvorom, za razliku od Ropstena koji podržava i Geth. Umjesto rudarenja s PoW-om, Kovan koristi *Proof of Authority* kao konsenzusni mehanizam. U PoA algoritmu su određeni čvorovi ovlašteni za kreiranje novih blokova i potvrđivanje transakcija, te samo oni mogu izvršiti spomenute operacije. Prosječno vrijeme objavljivanja novog bloka na mreži je 4 sekunde. Kovanov se Ether može dobiti samo tako da ga se zatraži od čvora validatora putem njegove slavine. Zbog mehanizma PoA, Kovan se donekle razlikuje od glavne mreže i zato se ne može smatrati preciznom simulacijom. Unatoč tome, izvrstan je za javno testiranje, pouzdan je i stabilan te imun na *spam* napade. RinkeBy također koristi Proof of Authority konsenzusni mehanizam, ali se razlikuje od testne mreže Kovan po tome što umjesto Parity inačice čvora podržava Geth, a umjesto Aura implementacije koristi implementaciju PoA algoritma nazvanu Clique. Prosječno je vrijeme objavljivanja novog bloka na RinkeBy mreži 15 sekundi.

Za razvoj praktičnog primjera opskrbnog lanca automobilske industrije temeljenog na lancu blokova bit će korištena testna mreža Ropsten zbog svojih sličnosti glavnoj mreži Ethereum lanca blokova.

4.5. Programski jezik Solidity

Solidity je objektno orijentiran, *high-level* programski jezik koji se koristi za implementaciju pametnih ugovora na Ethereum lanac blokova. On se piše statički, podržava nasljeđivanje, biblioteke i kompleksne tipove stvorene od strane korisnika. Nastao je pod utjecajem C++, Pythona i JavaScripta te dizajniran kako bi funkcionirao s Ethereum virtualnim strojem (EVM). On služi za pokretanje Ethereum pametnog ugovora, a radi isključivo s byte kodom, koji se dobiva

korištenjem *application binary interfacea* (ABI). ABI nam služi za kodiranje i dekodiranje Solidity byte koda. Uz korištenje Solidity programskog jezika, za razvoj sustava koristit će se Remix IDE (eng. *integrated development environment).* Remix je integrirano razvojno okruženje koje pomaže u pisanju Solidity pametnih ugovora ravno iz web preglednika ili lokalno na našem stroju. Također, omogućuje testiranje, *debugging* te objavljivanje pametnih ugovora na Ethereum lanac blokova. Remix također dolazi i s ugrađenim UI sučeljem koje će poslužiti za interakciju s pametnim ugovorima. Sučelje će pojednostaviti objavljivanje pametnih ugovora na lanac blokova, prikaz operacija koje su omogućene pametnim ugovorima te stanje proizvoda i mreže. Kao virtualni novčanik koristit će se Metamask novčanik koji je integriran u web preglednik (eng. *plugin*). Metamask je virtualni novčanik za upravljanje, prebacivanje te primanje Ethera ili ERC20 tokena. S obzirom da je integriran u web preglednik, služi kao posrednik između web preglednika i Ethereum lanca blokova [43]. Koristeći Metamask izradit će se četiri zasebna novčanika, od kojih će svaki predstavljati jednog sudionika mreže. Koristeći slavinu koja je dostupna na [44], zatražen je jedan Ether za svaki račun. Upravo će se ti računi koristiti u sljedećem poglavlju kako bi objavili pametne ugovore na mrežu lanca blokova.

5. KONCEPTUALNO PROGRAMSKO RJEŠENJE OPSKRBNOG LANCA TEMELJENOG NA TEHNOLOGIJI LANCA BLOKOVA

U ovom poglavlju prikazana je razrada modela iz prethodnog poglavlja u obliku programskog rješenja kojim se prikazuje upotreba tehnologije lanca blokova u lancima opskrbe. Uz prikaz programskog rješenja napisanog u Solidity programskom jeziku bit će objašnjeni specifični dijelovi programa kao i postupak objavljivanja pametnih ugovora, korištenja samog sustava te prikaz rezultata.

5.1. Postupak razvoja programskog rješenja

Kao što je prethodno rečeno, za razvoj pametnih ugovora bit će korišteno Remix integrirano razvojno okruženje omogućeno unutar web preglednika. Nakon početnog kreiranja SupplyChain datoteke s nastavkom .sol navodi se inačica programskog jezika Solidity koja će biti korištena, prikazano na slici 5.1.

```
1
2  pragma solidity >=0.4.22 <0.7.0;
```

Slika 5.1. *Inačica Solidity programskog jezika*

5.1.1. Pametni ugovor distributera sirovine

Nakon navođenja inačice koja će biti korištena za razvoj, kreiran je prvi pametni ugovor pod nazivom *RawMaterialTrader*. Kao što je rečeno u prethodnom poglavlju, Solidity programski jezik je objektno orijentiran što znači da će definicija pametnog ugovora biti predstavljena kao klasa u objektno orijentiranom jeziku, samo s ključnom riječi *contract*. Prva funkcija koja je kreirana je takozvani konstruktor. To je funkcija koja se izvršava pri kreiranju pametnog ugovora odnosno objavljivanju ugovora na lanac blokova. Ona nam služi za početnu inicijalizaciju samog ugovora te spremanje adrese vlasnika ugovora i adrese proizvođača automobilskih dijelova. Kao što je vidljivo na slici 5.2, adresa vlasnika će biti ona adresa s koje je ugovor objavljen, odnosno adresa virtualnog novčanika s kojeg je ugovor objavljen na mrežu. Za razliku od adrese vlasnika adresa proizvođača automobilskih dijelova bit će predana kao parametar prilikom inicijalizacije pametnog ugovora. Stoga možemo primijetiti deklaraciju varijable tipa *CarPartsManufacturer* koja predstavlja referencu na pametni ugovor proizvođača automobilskih dijelova koji će naknadno biti napisan. Upravo će zbog ove reference biti omogućen pristup određenim funkcijama pametnog ugovora proizvođača automobilskih dijelova unutar pametnog ugovora distributera sirovine. Također, možemo primijetiti da je prilikom deklaracije varijable *owner* korištena ključna

riječ *payable* koja omogućuje transakcije s vlasnikom ugovora. Ona uključuje prijenos Ethera s jedne adrese na drugu, a bez njene uporabe to ne bi bilo moguće.

```
CarPartsManufacturer public partsManufacturer;
address payable owner;

constructor(CarPartsManufacturer carPartsManufacturersAddress) public {
    partsManufacturer = carPartsManufacturersAddress;
    owner = msg.sender;
}
```

Slika 5.2. *Prikaz konstruktora RawMaterialTrader pametnog ugovora*

Korištenjem strukturnog modela distributera sirovine prikazanog na slici 4.3. u prošlom poglavlju, kreirana je struktura podataka koja će služiti za unos novih te prikaz postojećih sirovina u sustavu. Kao što je prikazano na slici 5.3, korištenjem ključne riječi *struct* možemo kreirati vlastite tipove podataka koje grupiraju nekolicinu varijabli. Kao što je vidljivo, kreirali smo strukturu koja sadrži *batchId* tipa *unsigned Integer* odnosno pozitivan cijeli broj, koji će služiti kao jedinstveni identifikacijski broj iskopane sirovine. Također, struktura će sadržavati informacije kao što su ime tvrtke koja je obavila iskopavanje sirovine, zemlju podrijetla, količinu, vrstu sirovine i cijenu. Između ostalog, struktura će sadržavati adresu početnog i trenutnog vlasnika iskopane sirovine. Početno će te dvije adrese biti iste, odnosno referencirat će se na adresu novčanika distributera sirovine. Tek nakon prodaje određene serije sirovina[5] promijenit će se adresa trenutnog vlasnika, dok će početna adresa ostati ista kako bi nam kasnije poslužila za praćenje te serije kroz cijeli opskrbni lanac, točnije kako bi od gotovog automobila mogli pratiti životni ciklus sve do distributera sirovine.

```
struct rawMaterialsInfo {
    uint256 batchId;
    string companyName;
    string countryOfOrigin;
    int256 quantity;
    string typeOfMaterial;
    uint256 price;
    address initialOwner;
    address currentOwner;
}
```

Slika 5.3. *Prikaz strukture sirovine*

[5] Serija sirovina (eng. *Batch*) – količina ili pošiljka dobara proizvedena istovremeno.

Kao što se može primijetiti na slici 5.4, deklarirana su i dva brojača koja će služiti za praćenje kreiranih serija sirovina i kreiranje jedinstvenih identifikacijskih oznaka. Brojač *counterId* će nam služiti za kreiranje novih identifikacijskih oznaka prilikom kreiranja novih serija sirovina tako što će se inkrementirati prije nego se to dogodi. Taj brojač je privatan što znači da neće biti vidljiv tijekom korištenja sustava, dok je *batchCounter*, koji će služiti za prikaz trenutno dostupnih sirovina, javan. U Solidity programskom jeziku sve varijable koje sadrže ključnu riječ *public* automatski dobivaju svoje *getter* funkcije, odnosno funkcije koje služe za prikaz trenutne vrijednosti varijable. Također, korištenjem „mapiranja" stvaraju se ključ-vrijednost parovi (eng. *key-value pair*) koji omogućuju pretraživanje serija sirovina po njihovom jedinstvenom identifikatoru odnosno *batchId*-u. *Mapping* je struktura slična *hash* tablicama samo što *mapping* inicijalizira sve ključeve na početku programa zbog čega se vrijednosti mogu naknadno dodavati, a ključevi ne mogu. U ovom je primjeru „mapiranje" odrađeno pomoću *Integera* čime je omogućeno pretraživanje po jedinstvenom identifikatoru.

```
mapping(uint => rawMaterialsInfo) public rawMaterials;
uint256 public batchCounter = 0;
uint256 private counterId = 0;
```

Slika 5.4. *Deklariranje brojača i „mapiranje" strukture*

Nakon završetka definiranja svih potrebnih varijabli i struktura, započinje pisanje funkcije koja će služiti za unos sirovina u sustav. Funkcija prikazana na slici 5.5, kao parametre prima ime tvrtke koja je obavila iskopavanje, zemlju podrijetla, količinu, vrstu sirovine te cijenu. Kao u većini programskih jezika, parametrima funkcije moraju biti deklarirani tipovi podataka, no uz podatke tipa *string* potrebno je staviti ključnu riječ *storage, memory* ili *calldata*. S obzirom da veličina podatka tipa *string* ovisi o samoj dužini riječi te da se *string* podatak zapravo sastoji od tipova *char* potrebna je referenca na dio memorije u kojoj je spremljen taj podatak. Prilikom rada s takvim tipovima podataka može se koristiti ključna riječ *storage* koja predstavlja trajno zapisivanje vrijednosti u memoriju čime se povećava vrijednost *gasa* koju je potrebno platiti pri takvoj operaciji. Ključna riječ *memory* predstavlja niz bajtova podataka koji se brišu nakon završetka funkcije što znači da je vrlo jeftina, točnije da neće povećavati vrijednost *gasa* koji se mora platiti. Ključna riječ *calldata* zapravo predstavlja *stack* koji sadrži lokalne varijable samo tijekom izvršavanja funkcije što znači da se vrijednost *gasa* neće povećati kao ni kod ključne riječi *memory*. No, za razliku od ključne riječi *memory, calldata* može spremati malu količinu podataka. Upravo iz tog razloga za razvoj svih pametnih ugovora u ovom primjeru korištena je ključna riječ

memory. U funkciji je početno inkrementirana vrijednost dva brojača, potom su proslijeđeni parametri spremljeni u strukturu. Strukturi je dodijeljen identifikator pomoću kojeg je omogućeno pretraživanje i pregledavanje same strukture. Nakon što je transakcija zapisana u lanac blokova i objavljena na mreži, svi sudionici mreže mogu pretraživati dostupne sirovine i njihove informacije.

```
function createRawMaterials(string memory _companyName, string memory _countryOfOrigin, int256 _quantity,
string memory _typeOfMaterial, uint256 _price) public onlyOwner
{
    batchCounter += 1;
    counterId += 1;
    rawMaterials[counterId] = rawMaterialsInfo(counterId, _companyName,
    _countryOfOrigin, _quantity, _typeOfMaterial, _price, owner, owner);
}
```

Slika 5.5. *Funkcija za kreiranje sirovina*

Jednako tako, na slici iznad možemo primijetiti da je prije samog tijela funkcije korišten *modifier* pod nazivom *onlyOwner*. Ove funkcije služe kako bi se implementirale određene restrikcije nad funkcijama koje se pozivaju. Upravo je na taj način spriječeno da bilo koji sudionik mreže unosi lažne podatke o nepostojećim sirovinama kako bi naštetio stvarnom distributeru sirovine. Stoga ova funkcija onemogućava da korisnik koji nije distributer sirovine uopće dođe do dijela koda u kojem se sirovina doista kreira. Kao što je prikazano na slici 5.6, ovu funkciju može pokrenuti samo vlasnik ugovora, odnosno tvrtka koja je objavila taj ugovor na mrežu lanca blokova. Pomoću funkcije *require* provjerava se je li osoba koja je pokrenula postupak kreiranja novih sirovina osoba koja je vlasnik tog ugovora. Funkcija vraća *boolean* vrijednost, odnoso *true* ili *false*. Ukoliko je vrijednost *false*, sudioniku je odmah onemogućen pristup funkciji. Programski jezik Solidity zahtijeva *underscore* na kraju *modifier* funkcije.

```
modifier onlyOwner() {
    require(msg.sender == owner);
    _;
}
```

Slika 5.6. *Modifier koji provjerava je li vlasnik ugovora pokrenuo kreiranje sirovina*

U pametni ugovor distributera sirovina mora se implementirati funkcija za prodaju sirovina drugim riječima funkcija koja će omogućiti ostalim sudionicima mreže kupovinu sirovina. Kao što vidimo na slici 5.7 funkcija implementirana kod distributera sirovina poziva funkciju proizvođača automobilskih dijelova koja će se koristiti za spremanje kupljenih sirovina, a bit će naknadno napisana. Možemo primijetiti kako se koristiti *transfer*, funkcija unutar Solidity programskog jezika koja omogućuje prebacivanje kripto valute s adrese jednog novčanika na drugi. U ovom je slučaju *owner* adresa novčanika vlasnika ugovora koja je postavljena prilikom inicijalizacije

ugovora unutar konstruktora. Pomoću *msg.value* čitamo unesenu vrijednost Ethera prilikom poziva funkcije. Svakako treba primijetiti kako je funkcija *payable* čime je omogućen prijenos valute s jednog računa na drugi. Vrijednost brojača dostupnih sirovina za kupovinu se smanjuje za jedan s obzirom da je jedna serija sirovina prodana proizvođaču automobilskih dijelova. Prodana serija sirovina ostaje i dalje vidljiva kod distributera sirovina, ali vlasništvo se prebacuje na kupca. Prebacivanje vlasništva je vidljivo na zadnjoj liniji funkcije gdje vlasnik sirovine postaje sudionik koji je započeo transakciju sirovina, odnosno pokrenuo funkciju za kupovinu sirovina.

```
function buyRawMaterials(uint256 _batchId) public payable
balanceValidation(tx.origin, rawMaterials[_batchId].price) isValidAmount(msg.value/10**18, rawMaterials[_batchId].price)
{
    partsManufacturer.storeMaterials(rawMaterials[_batchId].batchId, rawMaterials[_batchId].quantity,
    rawMaterials[_batchId].typeOfMaterial, rawMaterials[_batchId].price);
    partsManufacturer.incrementOfMaterials();
    owner.transfer(msg.value);
    batchCounter -= 1;
    rawMaterials[_batchId].currentOwner = tx.origin;
}
```

Slika 5.7. *Funkcija za kupovinu sirovina*

Također, korištena su dva *modifiera balanceValidation* i *isValidAmount.* Pomoću *balanceValidation modifera* provjerava se ima li kupac dovoljno sredstava na računu kako bi kupio sirovinu. Dok se pomoću *isValidAmount modifera* provjerava je li kupac poslao odgovarajuću vrijednost Ethera, kako se ne bi dogodilo da premalo ili previše plati, vidljivo na slici 5.8. Prilikom slanja Ethera događa se pretvorba u Wei, odnosno najmanju jedinicu Ethera, stoga je potrebno prvi parametar *modifiera isValidAmount* prilikom pozivanja podijeliti s 10^{18} kako bi se poslana vrijednost vratila iz Weija u Ether, što je vidljivo na slici 5.7.

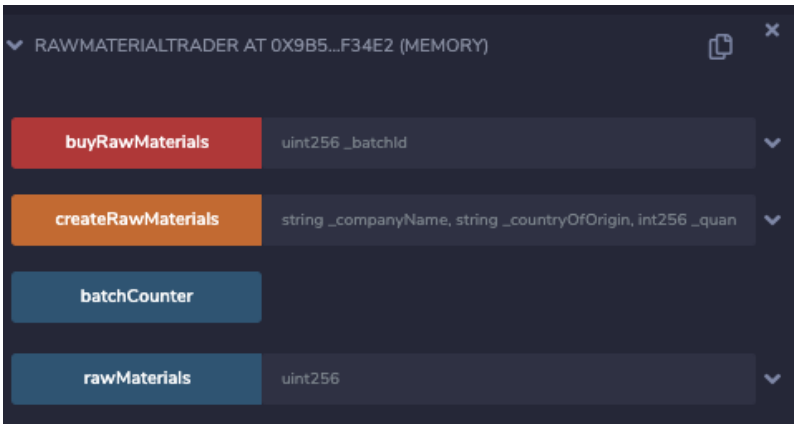
$$1\ Ether = 10^{18}\ Wei \tag{5-1}$$

Ova se pretvorba koristi samo tijekom razvoja pametnih ugovora dok se objavljuju lokalno od strane Remix IDE koji omogućuje 10 računa s 100 Ethera na svakom računu. Kada dođe do objavljivanja ugovora na Ropsten testnu mrežu koristit će se *Metamask* novčanici koji će na računu imati po 1 Ether. Stoga će lokalno cijene proizvoda biti u Etheru, a na Ropsten mreži u Weiju, kako bi se skratio proces punjenja četiri novčanika pomoću slavina, što je vrlo dugotrajan proces.

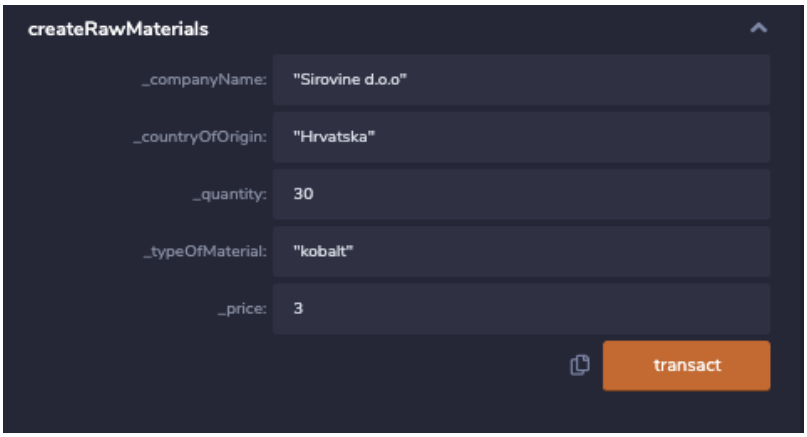
```
modifier balanceValidation(address buyer, uint256 price) {
    require(buyer.balance >= price);
    _;
}
modifier isValidAmount(uint256 sendValue, uint256 requiredAmount) {
    require(sendValue == requiredAmount);
    _;
}
```

Slika 5.8. *Modifieri za provjeru balansa računa i uplaćene kripto valute*

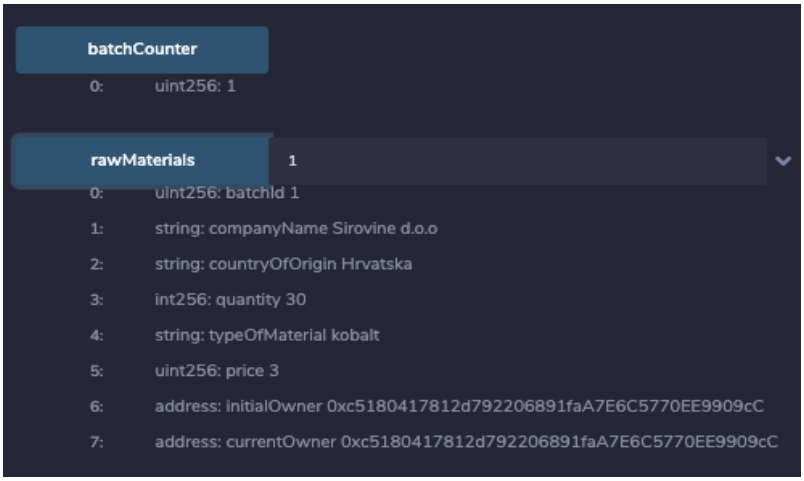
Tim procesom završen je razvoj pametnog ugovora distributera sirovine. Nakon što se kompajlirao programski kod, unutar Remix IDE omogućeno je lokalno objavljivanje pametnog ugovora. Potom se objavljuje ugovor s jednim od 10 računa, čime je zajamčeno UI sučelje koje omogućuje pozivanje svih napisanih funkcija, prikazano na slici 5.8. Na slikama 5.9 i 5.10 prikazan je postupak objavljivanja sirovina u mrežu, trenutno stanje dostupnih sirovina te njihov pregled.



Slika 5.8. *UI sučelje pametnog ugovora distributera sirovine*



Slika 5.9. *Prikaz postupka unosa informacija o sirovini u mrežu*



Slika 5.10. *Prikaz brojača dostupnih sirovina te pregled informacija o sirovini pomoću ID*

5.1.2. Pametni ugovor proizvođača automobilskih dijelova

Potrebno je kreirati novi pametni ugovor koji će sadržavati sve potrebne funkcionalnosti za proizvođača automobilskih dijelova. Prvo se kreira konstruktor sličan prethodnom samo što se sada sprema adresa ugovora proizvođača automobila. Ugovor će biti naknadno napisan, dok je konstruktor prikazan na slici 5.11.

```
CarManufacturer public carManufacturer;
address payable owner;

constructor(CarManufacturer carManufacturersAddress) public {
    carManufacturer = carManufacturersAddress;
    owner = msg.sender;
}
```

Slika 5.11. *Konstruktor pametnog ugovora proizvođača automobilskih dijelova*

Kako bi se omogućilo proizvođaču automobilskih dijelova da prilikom kupnje sirovina spremi potrebne informacije o kupljenoj sirovini kreira se struktura zajedno s mapiranjem. One omogućuju pretraživanje i pregled informacija, a prikazane su na slici 5.12.

```
struct MaterialsInfo {
    uint256 batchId;
    int256 quantity;
    string typeOfMaterial;
    uint256 price;
    address owner;
}
mapping(uint => MaterialsInfo) public boughtMaterials;
```

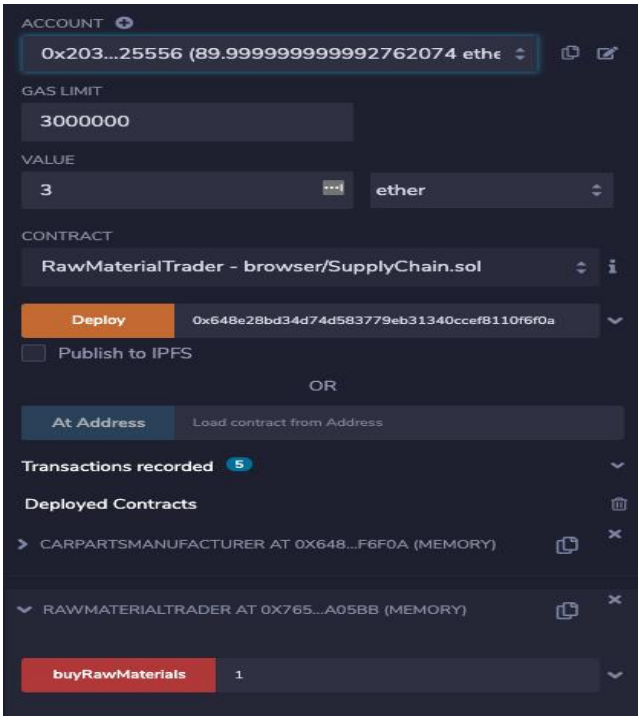
Slika 5.12. *Prikaz strukture kupljenih sirovina i njezinog mapiranja*

Kao što je rečeno, prilikom izrade funkcije za kupovinu sirovina unutar pametnog ugovora distributera sirovina potrebno je napisati funkcije koje će se pozvati kako bi spremile kupljeni proizvod u kreiranu strukturu. Zatim je potrebno inkrementirati brojač koji služi za prikaz grupa sirovina koje su u vlasništvu proizvođača dijelova. Prikaz funkcija za kupovinu i povećanje brojača vidljive su na slici 5.13.

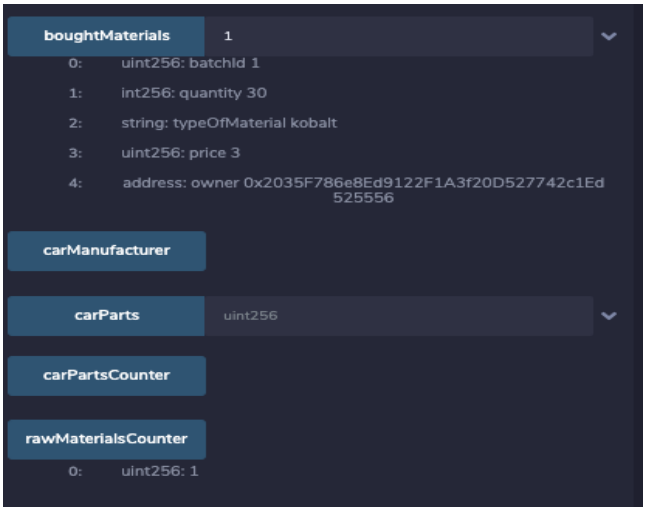
```
function incrementOfMaterials () public {
    rawMaterialsCounter += 1;
}

function storeMaterials (uint256 _batchId, int256 _quantity, string memory _typeOfMaterial, uint256 _price)
public hasEnoughMoney(_price)
{
    boughtMaterials[_batchId] = MaterialsInfo(_batchId, _quantity, _typeOfMaterial, _price, owner);
}
```

Slika 5.13. *Prikaz funkcija za spremanje kupljenih sirovina i povećanje brojača*

Nakon završetka razvoja funkcije prikazane iznad, omogućena je kupovina sirovina od distributera te prijenos vlasništva tih sirovina. Na slikama 5.14 i 5.15 predočen je postupak kupovine sirovina te pregled kupljenih sirovina u vlasništvu proizvođača automobilskih dijelova. Na slici 5.14 treba primijetiti kako se pomoću računa proizvođača automobilskih dijelova započinje postupak kupovine sirovine. Kao parametar funkciji za kupovinu predan je jedinstveni identifikacijski broj koji označava točno određenu seriju sirovina, a u *value input* polje unesena je količina Ethera koja je potrebna za kupovinu, u ovom slučaju 3 Ethera. Na slici 5.15 vidljivo je povećanje brojača pod nazivom *rawMaterialsCounter,* također je prikazan postupak pretraživanja informacije o sirovini pomoću jedinstvenog identifikatora.



Slika 5.14. *Prikaz postupka kupovine sirovina*



Slika 5.15. *Prikaz informacija o kupljenoj sirovini*

S obzirom da proizvođač automobilskih dijelova koristi kupljene sirovine kako bi proizveo određene dijelove automobila, potrebna je funkcija koja će to omogućiti. Na početku je potrebna struktura podataka koja će biti korištena za spremanje novo kreiranih dijelova, kao i funkcionalnost pretraživanja kreiranih auto dijelova te brojač koji ukazuje na dostupne dijelove, slika 5.16. Kao što je prikazano, kreira se nova identifikacijska oznaka, jedinstvena za proizvođača auto dijelova, također sprema se jedinstvena identifikacijska oznaka stavljena od strane distributera čime je ostvarena poveznica između kreiranog dijela automobila i sirovine koja je korištena za proizvodnju. Uz jedinstvene identifikacijske oznake, unose se informacije o vrsti automobilskog dijela, njegovoj cijeni, proizvođaču i vlasniku.

```
struct CarPartsInfo {
    uint256 id;
    uint256 batchId;
    string typeOfCarPart;
    uint256 price;
    string manufacturedBy;
    address initialOwner;
    address currentOwner;
}
mapping(uint => CarPartsInfo) public carParts;
uint256 public rawMaterialsCounter = 0;
uint256 public carPartsCounter = 0;
```

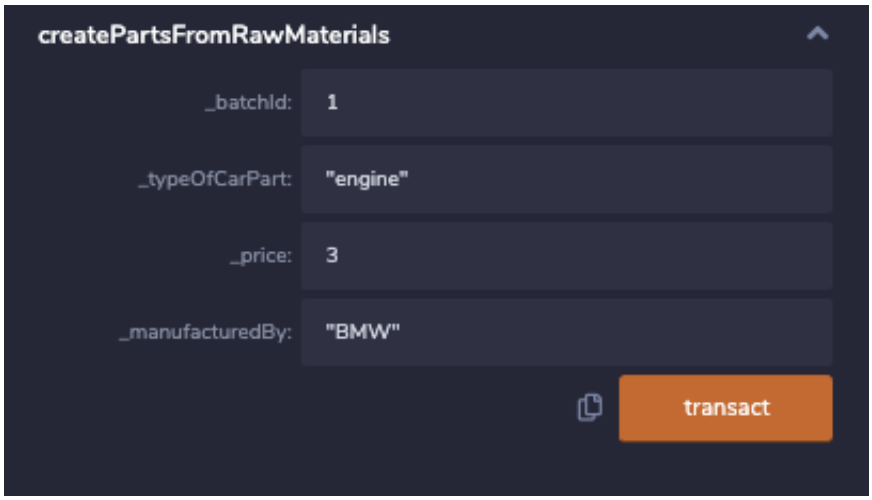
Slika 5.16. *Prikaz strukture, mapiranja i varijabli potrebnih kreiranje dijelova automobila*

Kako su definirane sve potrebne strukture i varijable, implementacija funkcije za kreiranje automobilskih dijelova može početi. Kao što struktura iznad prikazuje, parametri funkcije bit će jedinstvena identifikacijska oznaka serije sirovina pomoću koje se kreira određeni automobilski dio, vrsta dijela, cijena i naziv proizvođača. Nova jedinstvena identifikacijska oznaka automatski je dodijeljena novoj strukturi, pomoću koje će biti omogućeno pretraživanje. Početne vrijednosti *initialOwner* i *currentOwner* bit će postavljene na adresu vlasnika ugovora, sve do trenutka prodaje kada se *currentOwner* adresa mijenja na adresu kupca automobilskog dijela. Slika 5.17 prikazuje prvi korak u kojem se provjerava količina sirovine koju vlasnik posjeduje i ima li dovoljno sirovine da proizvede određeni automobilski dio, u ovom slučaju potrebna količina je predstavljena brojem 10. Ukoliko proizvođač posjeduje dovoljnu količinu sirovine, prvo se povećava brojač dijelova koje posjeduje, zatim se pomoću ranije kreirane strukture spremaju informacije o proizvodu te se konačno oduzima količina utrošena na proizvodnju. Ukoliko nakon procesa proizvodnje više ne postoji ta sirovina, brojač sirovina koje vlasnik ugovora posjeduje se smanjuje, ali ne briše kako bi se osiguralo praćenje proizvodnog ciklusa. U slučaju da se struktura obriše, što se može učiniti korištenjem ključne riječi *delete,* korištena sirovina tijekom proizvodnje više ne bi bila dostupna. Time bi se životni ciklus automobila prekinuo te vlasnik automobila ne bi mogao pratiti cijeli ciklus proizvodnje.

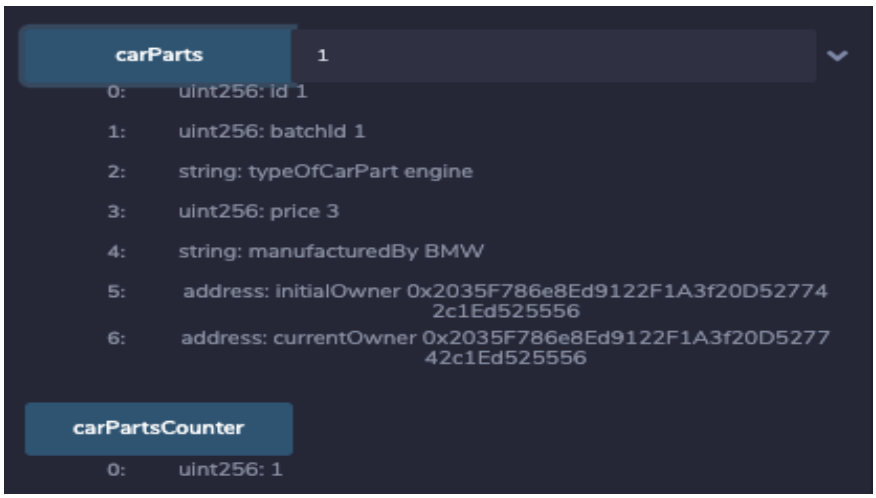
```
function createPartsFromRawMaterials(uint256 _batchId, string memory _typeOfCarPart, uint256 _price,
string memory _manufacturedBy) public
{
    if (boughtMaterials[_batchId].quantity >= 10) {
        carPartsCounter += 1;
        carParts[carPartsCounter] = CarPartsInfo(carPartsCounter, _batchId, _typeOfCarPart,
        _price, _manufacturedBy, owner, owner);
        boughtMaterials[_batchId].quantity -= 10;
        if (boughtMaterials[_batchId].quantity == 0) {
            rawMaterialsCounter -= 1;
        }
    } else {
        assert(false);
    }
}
```

Slika 5.17. *Funkcija za proizvodnju auto dijelova*

Na slikama 5.18 i 5.19 prikazan je postupak kreiranja auto dijelova pomoću kupljenih sirovina te pregled njihovih informacija. Kao što slike prikazuju, potrebno je unijeti identifikator sirovine pomoću kojeg se želi izraditi određeni dio automobila. Također, potrebno je unijeti dodatne informacije o kreiranom dijelu automobila, kao što je naziv kreiranog dijela, njegova prodajna cijena te naziv proizvođača.



Slika 5.18. *Prikaz postupka kreiranja automobilskih dijelova*



Slika 5.19. *Prikaz brojača gotovih dijelova te informacije o kreiranom dijelu*

Kako bi se omogućila prodaja novo kreiranih automobilskih dijelova potrebno je implementirati funkciju koja će služiti proizvođaču automobila za kupnju dijelova koji su potrebni za izradu automobila. Kako se prilikom inicijalizacije ugovora unutar konstruktora sprema adresa pametnog ugovora proizvođača automobila, koji će biti naknadno definiran, omogućena je kupnja auto dijelova od proizvođača dijelova. Stoga se definira funkcija koja, prilikom pozivanja od strane proizvođača automobila, provjerava stanje računa s cijenom dijela koji se kupuje te je li uplaćen točan iznos kripto valute. Ukoliko je, transakcija Ethera se događa pomoću funkcije *transfer*, nakon čega se poziva funkcija unutar pametnog ugovora proizvođača automobila koja služi za prijenos vlasništva. Kupnja određenog dijela automobila događa se preko jedinstvene identifikacijske oznake s obzirom da sudionik kojemu je potreban određeni dio može pregledati raspoložive dijelove kod proizvođača. Nakon transfera kripto valute i prebacivanja vlasništva, broj se gotovih dijelova koji proizvođač posjeduje smanjuje, nakon čega se adresa novog vlasnika upisuje u strukturu kako bi se naznačilo da taj dio više nije u vlasništvu proizvođača. Cijeli proces kupovine automobilskog dijela je vidljiv na slici 5.20.

```
function buyCarPart(uint256 _id) public payable
balanceValidation(tx.origin, carParts[_id].price) isValidAmount(msg.value, carParts[_id].price)
{
    owner.transfer(msg.value);
    carManufacturer.carPartTransfer(_id, carParts[_id].batchId, carParts[_id].typeOfCarPart,
    carParts[_id].price, carParts[_id].manufacturedBy);
    carPartsCounter -= 1;
    carParts[_id].currentOwner = tx.origin;
}
```

Slika 5.20. *Funkcija za kupovinu dijelova automobila*

5.1.3. Pametni ugovor proizvođača automobila

Sljedeći će sudionik opskrbnog lanca proizvodnje automobila biti proizvođač automobila odnosno tvrtka koja sastavlja automobile. Potrebno je kreirati novi pametni ugovor pod nazivom *CarManufacturer* koji će sadržavati sve potrebne funkcionalnosti koje zahtjeva takav proizvođač. Početno je kreiran konstruktor sličan prijašnjim koji služi za spremanje adrese vlasnika ugovora te adrese distributera automobila. Kako bi kupovina auto dijelova bila moguća potrebno je definirati funkciju koja se poziva za spremanje dijelova, vidljiva na slici iznad. Prvo je potrebno kreirati strukturu pomoću koje će se to obavljati kao i sva potrebna mapiranja i varijable, prikazane na slici 5.21. Proizvođač automobila prilikom kupnje dijelova dodjeljuje jedinstveni identifikator kupljenom dijelu. U funkciji iznad vidljivo je da proizvođač automobila dodjeljuje jednak identifikator kao i proizvođač dijelova. Preko jednakog identifikatora se stvara poveznica između proizvođača tih dijelova i automobila koji će biti kreiran pomoću njih. Također, sprema se

identifikator serije sirovina pomoću koje je kreiran dio, vrsta dijela automobila koji je kupljen, njegova cijena, proizvođač te vlasnik. Varijabla pod nazivom *vehicleId* početno će biti postavljena na nulu, ali će se naknadno upisati nakon što se automobil koji je kreiran pomoću tog dijela proizvede.

```
struct partsReadyForManufacturingInfo {
    uint256 id;
    uint256 batchId;
    string typeOfCarPart;
    uint256 price;
    string manufacturedBy;
    address owner;
    uint256 vehicleId;
}
mapping(uint => partsReadyForManufacturingInfo) public partsReadyForManufacturing;
uint256 public completedCarCounter = 0;
uint256 public partsForManufacturingCounter = 0;
```

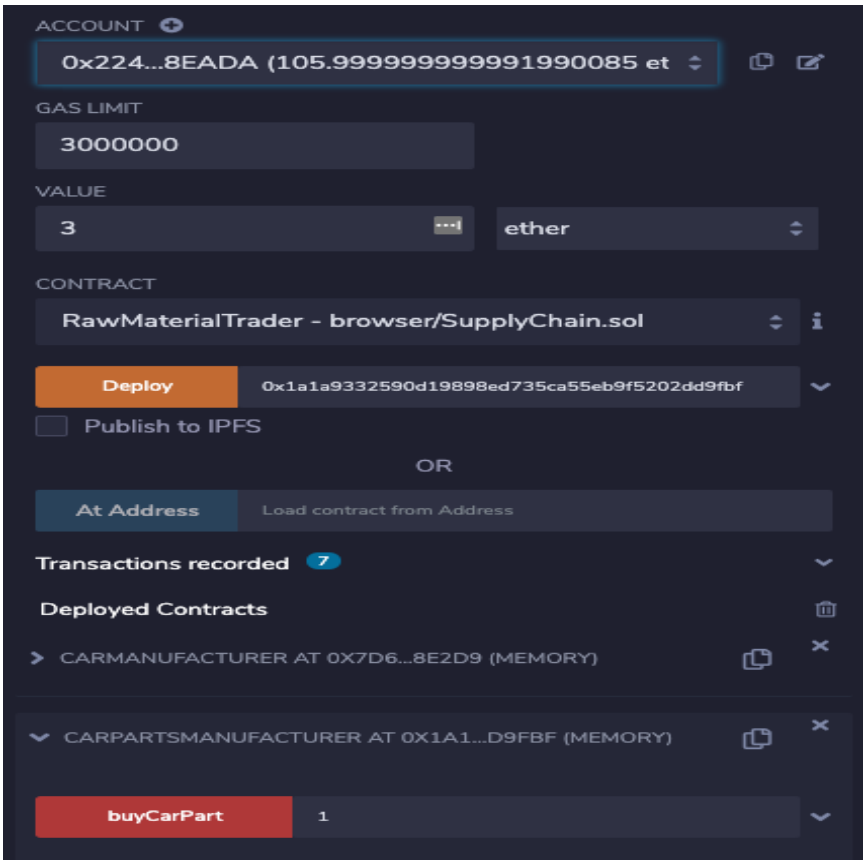
Slika 5.21. *Struktura kupljenih automobilskih dijelova, mapiranje i brojači*

U slijedećem koraku potrebno je definirati funkciju korištenu pri kupovini auto dijelova, koja je vidljiva na slici 5.22. Funkcija prima parametre poslane iz pametnog ugovora proizvođača auto dijelova te koristi iznad definiranu strukturu kako bi je spremila.

```
function carPartTransfer(uint256 _id, uint256 _batchId, string memory _typeOfCarPart, uint256 _price,
string memory _manufacturedBy) public
{
    partsForManufacturingCounter += 1;
    partsReadyForManufacturing[_id] =
    partsReadyForManufacturingInfo(_id, _batchId, _typeOfCarPart, _price, _manufacturedBy, owner, 0);
}
```

Slika 5.22. *Funkcija za spremanje kupljenih auto dijelova*

Na slici 5.23 prikazan je postupak kupovine gotovih dijelova automobila od strane proizvođača automobila, koji koristi gotove dijelove za izradu automobila. Nakon kupovine dijelova te prijenosa vlasništva, proizvođač automobila može provjeravati stanje dijelova koje posjeduje te provjeriti je li određeni dio već iskorišten u proizvodnji ili ne, prikazano na slici 5.24.



Slika 5.23. *Postupak kupovine gotovih auto dijelova*

Slika 5.24. *Prikaz informacija o kupljenim dijelovima*

Nakon što je omogućena kupovina i transfer auto dijelova, potrebno je napisati funkciju koja će služiti za proizvodnju automobila. U prvom koraku treba napisati strukturu koja će služiti za prikaz informacija o automobilu kao i mapiranje koje će služiti distributerima za pregledavanje dostupnih vozila. U ovom se praktičnom konceptu svaki automobil proizvodi od tri dijela: motora, prijenosa i ovjesa. Kao što se vidi na slici 5.25 proizvedenom automobilu dodjeljuje se jedinstvena

identifikacijska oznaka koja služi za kupovinu, ali i za praćenje informacija o automobilu od strane distributera i kupca. Pomoću jedinstvene identifikacijske oznake kupac dobiva informacije kao što su jedinstvena identifikacija motora, ovjesa i prijenosa, boja i tip vozila, cijena, te informacije o proizvođaču i vlasniku.

```
struct completedCarInfo {
    uint256 id;
    uint256 engineId;
    uint256 transmissionId;
    uint256 wheelsId;
    string color;
    string typeOfCar;
    uint256 sellingPrice;
    string manufacturedBy;
    address initialOwner;
    address currentOwner;
}
mapping(uint => completedCarInfo) public completedCar;
```

Slika 5.25. *Struktura gotovog automobila i njezino mapiranje*

U sljedećem koraku potrebno je kreirati funkciju prikazanu na slici 5.26, pomoću koje će proizvođač automobila koristeći različite kupljene dijelove proizvesti gotovo vozilo. Na početku funkcija uzima sumu vrijednosti svih kupljenih dijelova te ju povećava dva puta čime dobiva prodajnu cijenu automobila. Nakon toga povećava vrijednost brojača ukupnih gotovih vozila, što ujedno predstavlja i identifikacijsku oznaku. Koristeći identifikacijske oznake auto dijelova, proizvođač odabire dijelove koje želi koristi za proizvodnju te kreira automobil pomoću prethodno definirane strukture. Nakon čega umanjuje broj dijelova automobila koje ima u svojem vlasništvu. Na kraju dodjeljuje jedinstvenu identifikacijsku oznaku u strukturu svakog dijela korištenog pri proizvodnji kako bi se osigurala mogućnost praćenja sve do početnog distributera sirovina.

```
function createCar(uint256 _engineId, uint256 _transmissionId, uint256 _wheelsId, string memory _color,
string memory _typeOfCar, string memory _manufacturedBy) public
carPartsValidation(partsReadyForManufacturing[_engineId].typeOfCarPart,
partsReadyForManufacturing[_transmissionId].typeOfCarPart, partsReadyForManufacturing[_wheelsId].typeOfCarPart)
{
    uint256 totalPayedPrice = partsReadyForManufacturing[_engineId].price +
    partsReadyForManufacturing[_transmissionId].price + partsReadyForManufacturing[_wheelsId].price;
    uint256 sellingPrice = totalPayedPrice * 2;
    completedCarCounter += 1;
    completedCar[completedCarCounter] = completedCarInfo(completedCarCounter, _engineId, _transmissionId,
    _wheelsId, _color, _typeOfCar, sellingPrice, _manufacturedBy, owner, owner);
    partsForManufacturingCounter -= 3;
    partsReadyForManufacturing[_engineId].vehicleId = completedCarCounter;
    partsReadyForManufacturing[_transmissionId].vehicleId = completedCarCounter;
    partsReadyForManufacturing[_wheelsId].vehicleId = completedCarCounter;
}
```

Slika 5.26. *Funkcija za proizvodnju automobila*

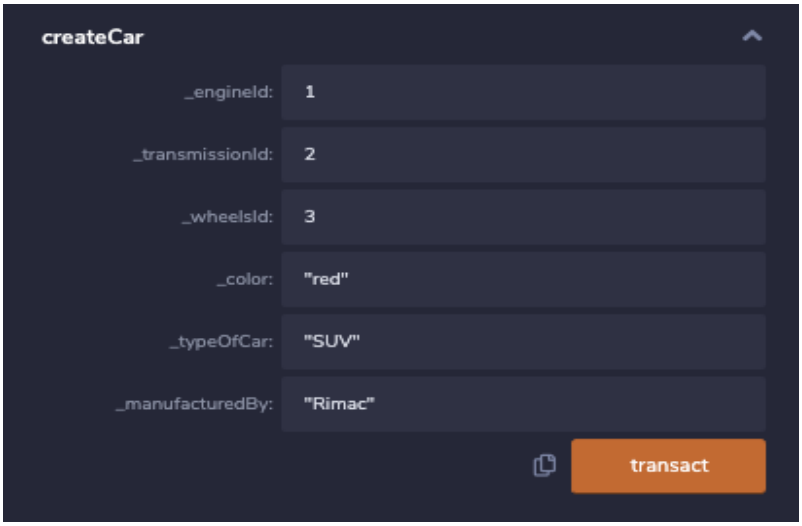
Također, funkcija koristi *modifier* pod nazivom *carPartsValidation* koji služi za provjeru korištenih dijelova pri proizvodnji. Kao što je rečeno, svaki se automobil u ovom primjeru sastoji

od motora, ovjesa i prijenosa, stoga se prilikom poziva funkcije moraju predati jedinstveni identifikatori baš tih dijelova, u suprotnom proizvodnja nije dozvoljena. Prikaz *modifiera* je na slici 5.27. S obzirom da uspoređivanje podataka tipa *string* u Solidity programskom jeziku nije moguće, potrebno je koristi *keccak256* kriptografsku metodu. Na početku pomoću *abi.encodePacked* funkcije pretvorimo predane *string* podatke u niz bajtova, koje nakon toga *hashiramo* pomoću *keccak256* funkcije. Dobivene *hasheve* uspoređujemo te vraćamo odgovarajuću *boolean* vrijednost.

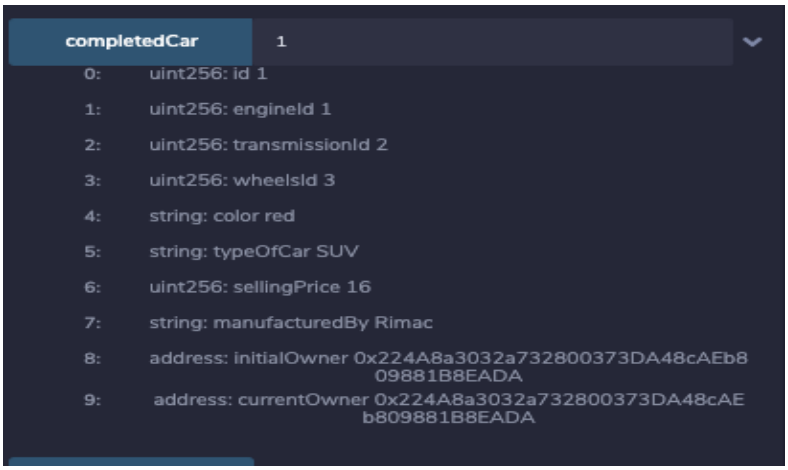
```
modifier carPartsValidation(string memory engine, string memory transmission, string memory wheels) {
    require(keccak256(abi.encodePacked((engine))) == keccak256(abi.encodePacked(("engine"))));
    require(keccak256(abi.encodePacked((transmission))) == keccak256(abi.encodePacked(("transmission"))));
    require(keccak256(abi.encodePacked((wheels))) == keccak256(abi.encodePacked(("wheels"))));
    _;
}
```

Slika 5.27. *Prikaz usporedbe stringova u Solidity jeziku*

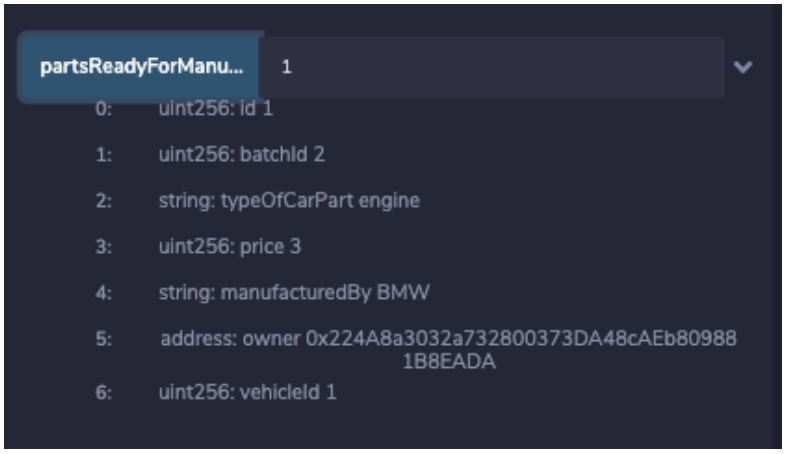
Na slici 5.28 prikazan je postupak proizvodnje automobila koja uključuje tri glavna dijela automobila što su motor, ovjes i prijenos. Kao što je prikazano, potrebno je predati identifikatore određenih dijelova koji će se koristiti prilikom proizvodnje. Također, potreban je unos dodatnih informacija o automobilu kao što su boja, tip i proizvođač automobila. Na slici 5.29 prikazane su informacije o kreiranom automobilu. Jedinstveni identifikatori motora, ovjesa i prijenosa upisani su kao dio informacija čime je omogućeno praćenje proizvodnog ciklusa. Također, izračunata je cijena automobila po kojoj će automobil biti dostupna kupcima. Unutar automobilskog dijela korištenog za proizvodnju automobila dodan je identifikator vozila, slika 5.30.



Slika 5.28. *Postupak proizvodnje vozila*



Slika 5.29. *Prikaz informacija o kreiranom automobilu*



Slika 5.30. *Prikaz upisivanja informacije o vozilu u čijoj proizvodnji je sudjelovao taj dio*

U slijedećem koraku potrebno je napisati funkciju koja će omogućiti kupnju gotovog automobila od proizvođača. Obavljaju se provjere stanja računa distributera te provjera uplate odgovarajuće cijene pomoću *modifiera*. Poziva se funkcija koja će biti naknadno napisana, a služit će za spremanje informacija o kupljenom automobilu. Posljednji korak je prebacivanje sredstava s jednog računa na drugi te prebacivanje vlasništva, prikazano na slici 5.31.

```
function buyCarFromManufacturer(uint256 _vehicleId) public payable
balanceValidation(tx.origin, completedCar[_vehicleId].sellingPrice)
isValidAmount(msg.value/10**18, completedCar[_vehicleId].sellingPrice)
{
    carDistributer.carOwnershipTransfer(completedCar[_vehicleId].id);
    completedCarCounter -= 1;
    owner.transfer(msg.value);
    completedCar[_vehicleId].currentOwner = tx.origin;
}
```

Slika 5.31. *Funkcija za kupovinu automobila*

5.1.4. Pametni ugovor distributera automobila

Potrebno je kreirati četvrti pametni ugovor kojim će biti predstavljen distributer automobila. Taj će ugovor predstavljati informacije o automobilu koje će kupac prilikom kupovine dobiti, u ovom primjeru to je jedinstvena identifikacija automobila. Krajnji će vlasnik automobila upravo preko jedinstvene identifikacijske oznake moći pratiti cijeli životni ciklus automobila. Unutar pametnog ugovora *CarDistributer* kreirana je struktura koja sadrži samo jedinstvenu identifikaciju kupljenog automobila i adresu vlasnika. Na slici 5.32 vidljiv je primjer cijelog pametnog ugovora u kojem je prikazan konstruktor koji se koristi prilikom inicijalizacije ugovora te funkcija za kupnju automobila. Ta se funkcija poziva iz funkcije za kupovinu automobila napisane unutar pametnog ugovora proizvođača automobila, a njezina je svrha spremanje potrebnih informacija o automobilu u definiranu strukturu i povećavanje brojača pomoću kojeg se može pratiti broj automobila u vlasništvu.

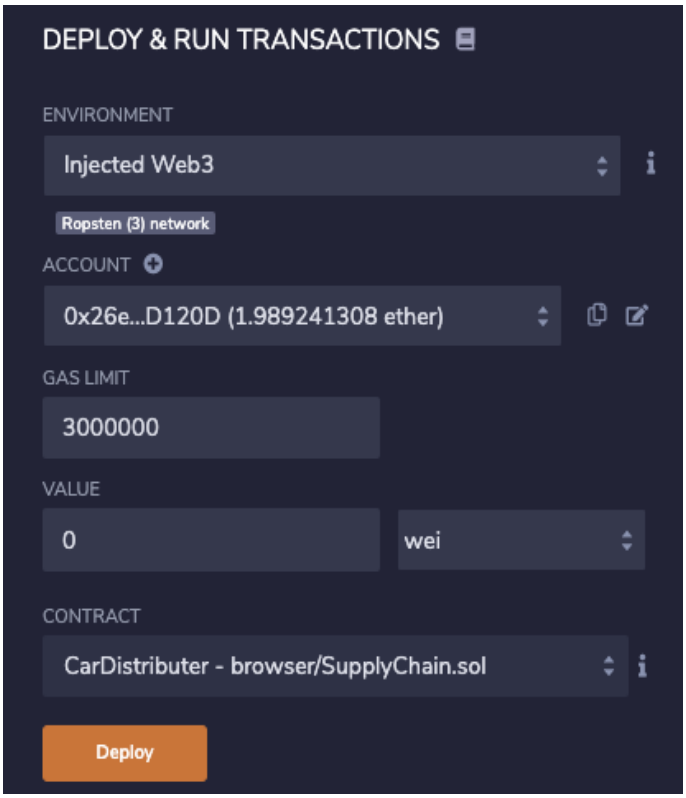
```
contract CarDistributer {
    address payable owner;

    struct carOwnershipInfo {
        uint256 id;
        address owner;
    }
    mapping(uint => carOwnershipInfo) public carOwnership;
    uint256 public carOwnershipCounter = 0;

    constructor() public {
        owner = msg.sender;
    }

    function carOwnershipTransfer(uint256 _vehicleId) public
    {
        carOwnershipCounter += 1;
        carOwnership[carOwnershipCounter] = carOwnershipInfo(_vehicleId, owner);
    }
}
```

Slika 5.32. *Prikaz pametnog ugovora kupca automobila*

5.2. Implementacija programskog rješenja

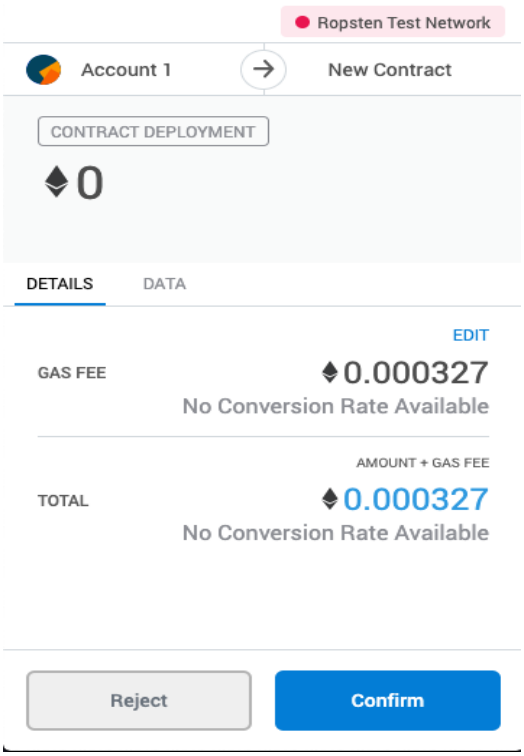
Završetkom razvoja pametnih ugovora i testiranja unutar Remix integriranog okruženja potrebno je objaviti ugovore na lanac blokova. Kao što je rečeno u prethodnom poglavlju koristit će se testna mreža Ropsten za prikaz rješenja. Također, koristit će se četiri kripto novčanika od kojih će svaki novčanik predstavljati jednog sudionika mreže. Kako je objašnjeno prilikom razvoja pametnih ugovora, prilikom objavljivanja određenih pametnih ugovora potrebno je predati adrese već objavljenih ugovora. Ukoliko se pogledaju konstruktori svih pametnih ugovora, može se primijetiti da pametni ugovor distributera automobila jedini ne zahtjeva adresu drugih ugovora. Ugovor

proizvođača automobila zahtjeva adresu distributera automobila, proizvođač dijelova zahtjeva adresu proizvođača automobila, dok distributer sirovina zahtjeva adresu proizvođača dijelova. Upravo iz tog razloga objavljivanje ugovora će teći obrnutim redoslijedom lanca opskrbe. Također, treba napomenuti da prije objavljivanja ugovora na mrežu, iz pametnih ugovora treba izbaciti prebacivanje Weija u Ether s obzirom da svaki novčanik posjeduje između jednog i dva Ethera. Novčanik pod nazivom *Account 1* pripadat će distributeru automobila, novčanik pod nazivom *Account 2* pripadat će proizvođaču automobila i tako dalje. Potom se treba prebaciti u drugu razvojnu okolinu (eng. *environment*). Prilikom razvoja pametnih ugovora korišten je JavaScript Virtual Machine, dok će za objavljivanje ugovora biti korišten Injected Web3. JavaScript VM omogućava pokretanje izoliranog Ethereum čvora lokalno u web pregledniku, što je vrlo korisno za testiranje pametnih ugovora. Injected Web3 koristi Web3 poslužitelja ugrađenog u web preglednik, primjerice Metamask će ugraditi Web3 poslužitelja u preglednik te omogućiti konfiguraciju potrebnu za povezivanje i objavljivanje ugovora na testnoj ili glavnoj mreži Ethereum lanca blokova. Korištena je stranica s izvora [45] kao vrsta *blockchain explorera* odnosno preglednika svih transakcija. Nakon kompajliranja svih ugovora, objavljivanje ugovora na testnu mrežu Ropsten može započeti. Slika 5.33 prikazuje UI sučelje koje se koristi za objavljivanje ugovora. Na slici se vidi korišteno okruženje te adresa novčanika za objavljivanje ugovora.

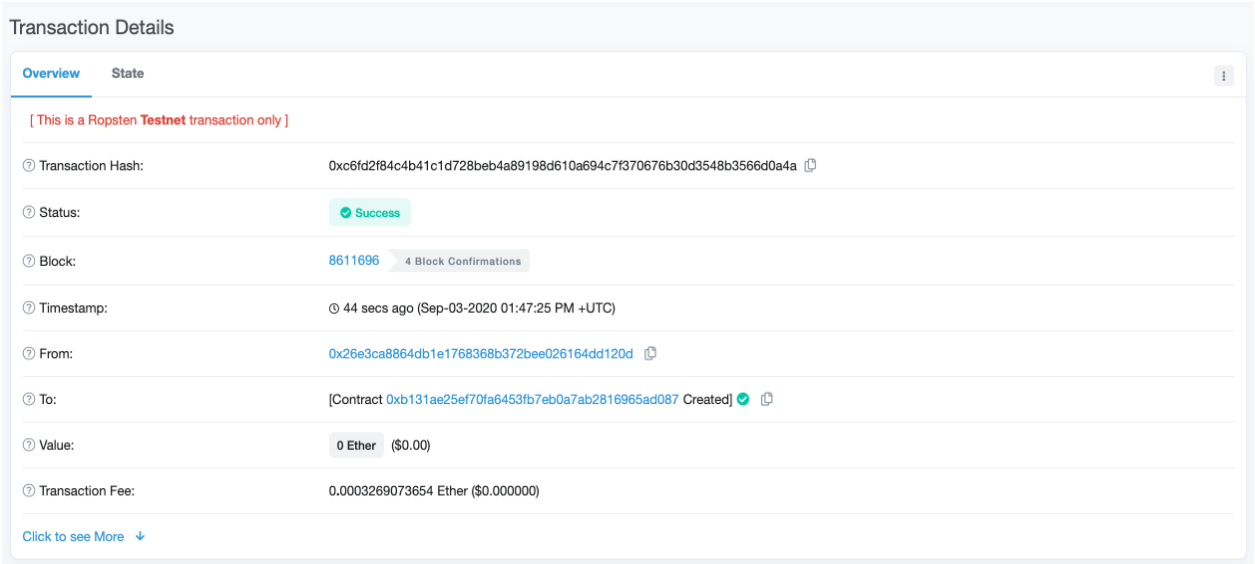


Slika 5.33. *Prikaz sučelja prilikom objavljivanja ugovora*

Nakon klika na gumb *Deploy* iskače prozor *Metamaska* na kojem je prikazana količina *gasa* koju je potrebno platiti za objavljivanje ugovora na mrežu, prikazano na slici 5.34. Prihvaćanjem uvjeta započinje objavljivanje pametnog ugovora na mrežu unutar bloka. Tridesetak sekundi nakon vidljivo je kako je blok izrudaren te objavljen na mrežu, slika 5.35.



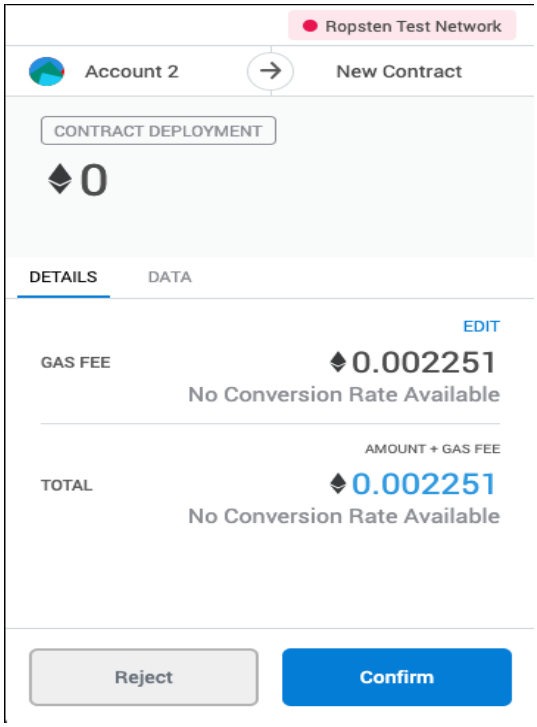
Slika 5.34. *Prikaz količine gasa koju je potrebno platiti prilikom objave pametnog ugovora*



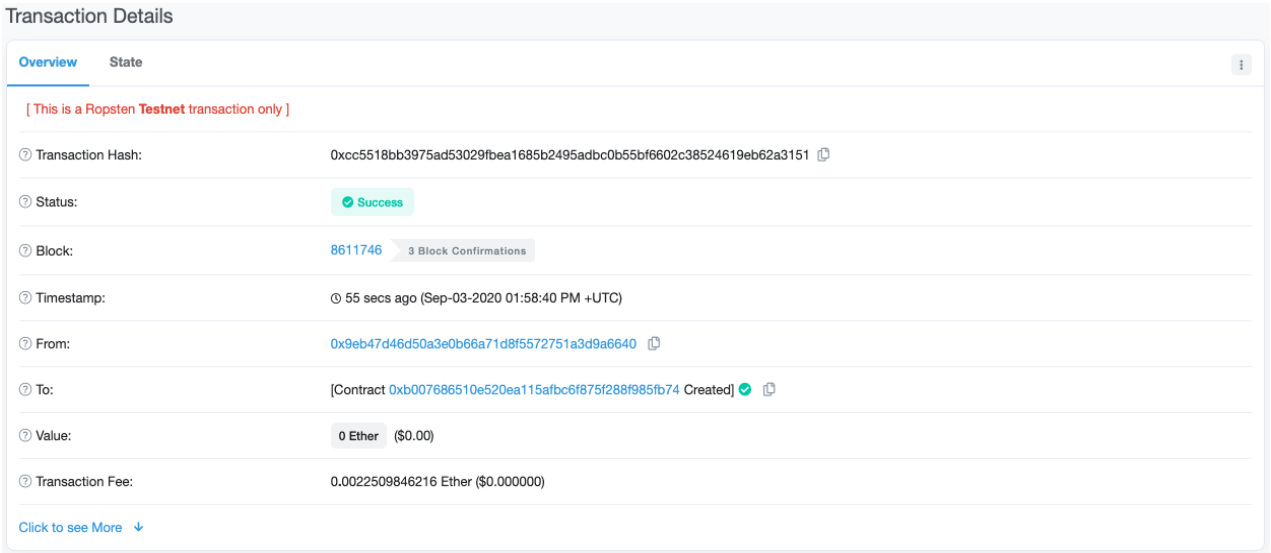
Slika 5.35. *Prikaz izrudarenog bloka*

Nakon što su informacije o pametnom ugovoru dostupne, koristi se prikazana adresa ugovora kako bi se objavio pametni ugovor proizvođača automobila. Slijedi prebacivanje na drugi račun odnosno

drugi novčanik kako bi se objavio slijedeći ugovor. Također, na slici 5.36 treba primijetiti kako je količina *gasa* veća nego količina *gasa* kod objavljivanja pametnog ugovora distributera automobila. Razlog tomu je što je pametni ugovor proizvođača automobila računski kompleksniji od pametnog ugovora distributera. Prikaz izrudarenog bloka vidljiv je na slici 5.37. Isti postupak objavljivanja korišten je i za ostala dva pametna ugovora. Po završetku objavljivanja pametnih ugovora na mrežu dobiva se prikaz svih funkcionalnosti ugovora unutar Remix IDE sučelja.



Slika 5.36. *Prikaz gasa za objavu pametnog ugovora proizvođača automobila*



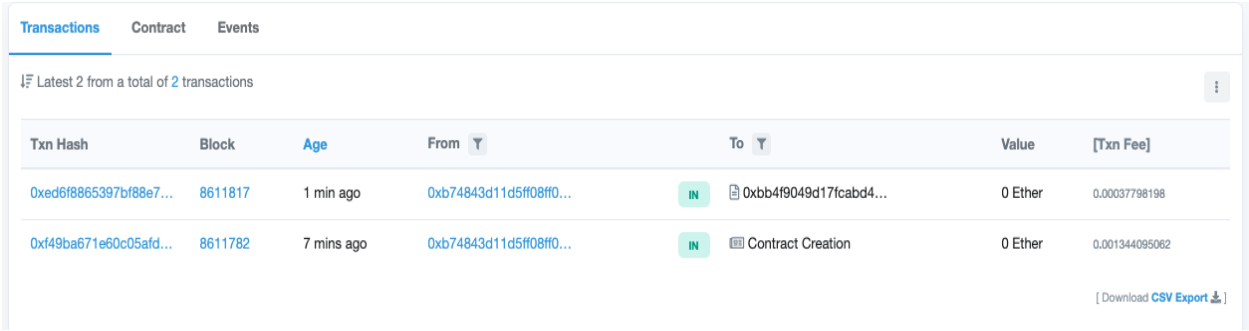
Slika 5.37. *Objavljeni pametni ugovor proizvođača automobila*

6. PREGLED I ANALIZA PROGRAMSKOG RJEŠENJA

U ovom poglavlju će biti analizirano kreirano konceptualno programsko rješenje iz prethodnog poglavlja uključujući njegove prednosti i nedostatke. Analizirat će se njegova implementacija u opskrbne lance automobilske industrije kao i ostale grane u kojima postoji potreba za takvom tehnologijom. Također, bit će prikazan pregled i analiza cijelog područja iskoristivosti tehnologije lanca blokova u opskrbnim lancima.

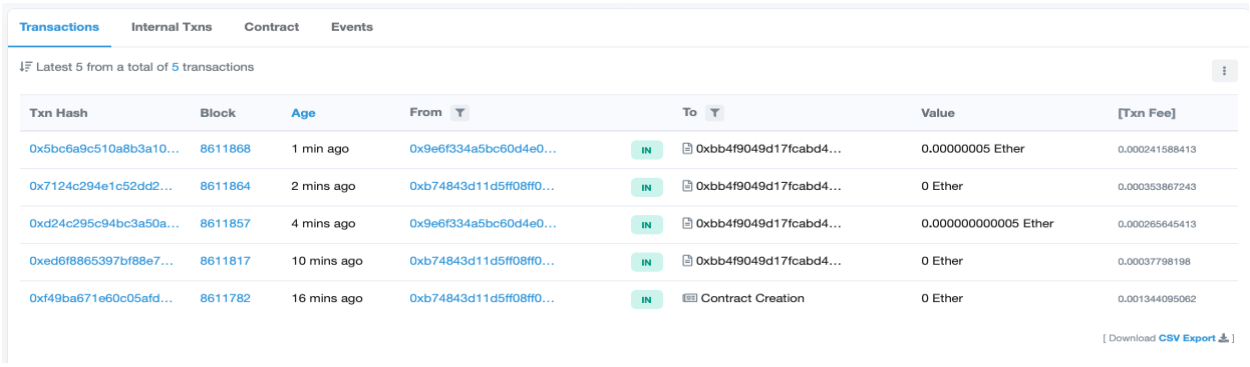
6.1. Primjer uporabe konceptualnog programskog rješenja

Nakon završetka objavljivanja svih potrebnih pametnih ugovora na Ethereum mrežu lanca blokova distributer sirovina može započeti s unosom informacija o sirovinama u sustav. Postupak unosa sirovina i njihovih informacija jednak je kao i u prethodnom poglavlju. Za razliku od prethodnog unosa ovaj put se informacije spremaju u lanac blokova pri čemu se plaća određeni iznos *gasa*. Svakako treba napomenuti kako prikaz informacija o sirovini te količini sirovina koje sudionik posjeduje neće biti vidljiv sve dok se ta informacija ne upiše u lanac blokova. Lokalno su te informacije bile dostupne odmah nakon unosa, dok je na lancu blokova potrebno pričekati da se te informacije spreme u blok te izrudare kako bi bile dostupne. Pregledom svih transakcija unutar ugovora vidljivo je da je unos sirovine u mrežu zapisan. Na slici 6.1 možemo vidjeti dvije transakcije. Početna transakcija, ona donja, označava objavljivanje ugovora na mrežu, dok druga transakcija označava objavljivanje sirovine unutar sustava. Upravo na ovakav način može se osigurati praćenje svih transakcija na mreži te njihova provjera.



Slika 6.1. *Prikaz svih transakcija unutar pametnog ugovora*

Nakon unosa sirovina u mreži, čime postaju vidljive ostalim sudionicima sustava, sirovine su dostupne za kupovinu. Proizvođač može pregledati informacije o sirovinama i odlučiti koje želi kupiti. Na slici 6.2 u listi svih transakcija unutar pametnog ugovora, vidljivo je da je određena sirovina kupljena te koliko Ethera je koštala. Također prikazan je dodatni unos sirovine u sustav te njegova kupovina.



Slika 6.2. *Lista transakcija unutar pametnog ugovora*

Na ovakav će se način sve transakcije unutar mreže pohranjivati, čime će biti omogućen pregled svih informacija o proizvodnom ciklusu automobila. Tako korisnik može biti siguran da svi dijelovi automobila dolaze od službenog proizvođača, odnosno da je proizvod originalan. U ovakvom sustavu krajnji korisnik automobila dobiva jedinstveni identifikacijski broj pomoću kojeg može saznati sve potrebne informacije o kupljenom automobilu. Pomoću jedinstvenog identifikatora pruženog od strane distributera, vlasnik može pretražiti sve proizvedene automobile kod proizvođača, te dobiti informacije kao što su jedinstveni identifikacijski brojevi motora, ovjesa i prijenosa, boju i tip automobila te vlasnika istog. Pomoću jedinstvenih identifikacijskih brojeva motora, ovjesa i prijenosa vlasnik može pogledati sve informacije o dijelovima. Pomoću tih oznaka vlasnik dolazi do proizvođača dijelova, koji mu pruža informacije o distributeru sirovina od kojih je taj dio nastao. Time vlasnik dobiva jedinstveni identifikator serije sirovina koja je korištena pri proizvodnji, što ga dovodi do posljednjeg, odnosno prvog, sudionika proizvodnog procesa, a to je distributer sirovina čime se završava proizvodni proces.

6.2. Analiza konceptualnog programskog rješenja

Korištenjem programskog jezika Solidity za pisanje pametnih ugovora na Ethereum lancu blokova kreiran je opskrbni lanac unutar automobilske industrije. Opskrbni se lanac sastoji od četiri pametna ugovora od kojih svaki predstavlja jednog sudionika proizvodnog ciklusa. Iako se Solidity programski jezik još razvija i nema podršku određenih kompleksnih tipova podataka, metoda, kao ni *floating* brojeva odnosno decimalnih brojeva, za kreiranje ovakvog sustava Solidity programski jezik s Remix integriranim okruženjem uspješno omogućava jednostavnu i brzu implementaciju pametnih ugovora. Cijeli proces razvoja dodatno je olakšan postojećim UI sučeljem koje nudi Remix integrirano okruženje. Razvojem konceptualnog primjera opskrbnog lanca automobilske industrije temeljenog na tehnologiji lanca blokova prikazana je transparentnost koja se nudi korištenjem ove tehnologije. Također, prikazano je rješavanje problema legitimnosti podataka i nepromjenjivosti podataka što predstavlja jedne od glavnih problema tradicionalnih

opskrbnih lanaca. Korisnici opskrbnog lanca temeljenog na tehnologiji lanca blokova mogu digitalizirati predmete opskrbnog lanca, bilo da se radi o materijalima, prehrambenim proizvodima, komponentama ili certifikatima usklađenosti, označavanjem proizvoda određenim skupom informacija. Informacije se bilježe na lanac blokova te u spoju s relevantnom pratećom tehnologijom, sudionici lanca blokova mogu pratiti kretanje proizvoda kroz fizički lanac opskrbe. Kako proizvod putuje kroz različite razine opskrbnog lanca, nove se informacije bilježe u realnom vremenu na lanac blokova. Transparentnost u lancima opskrbe ima brojne prednosti koje mogu stvoriti ili uništiti reputaciju proizvoda. Omogućuje tvrtkama da osiguraju jasan revizijski trag koji će im pomoći u upravljanju rizikom i smanjenju kvarova u opskrbnom lancu. Također, omogućavaju jednostavno praćenje etičkih politika trgovanja i osiguravaju ispunjavanje regulativa industrije. S obzirom na pogodnosti koje tehnologija lanca blokova pruža opskrbnom lancu sve je veći porast potražnje za uporabu tehnologije lanca blokova u različitim sektorima, što nije iznenađujuće s obzirom na sposobnost jednostavnog pronalaženja, provjere izvora i tretmana robe unutar opskrbnog lanca. Od opskrbnih se lanaca temeljenih na lancu blokova očekuje:

- Pružanje stvarnih i potrebnih informacija kupcu prilikom kupovine određenog proizvoda
- Jednostavniju regulaciju pronalaženja izvora proizvoda i postupanja s njim unutar industrija
- Pojednostavljeno i precizno identificiranje neispravnih dijelova ili krivotvorene robe za proizvođače i dobavljače proizvoda

Ovakav sustav bi na jednostavan način omogućio mogućim kupcima automobila temeljitu i istinitu provjeru podataka unutar proizvodnog ciklusa automobila. Na takav bi se način kupcima automobila ulijevalo povjerenje prilikom kupovine da uistinu kupuju automobil proizveden sa svim originalnim dijelovima te provjerenim i verificiranim proizvođačima. Također, ovakav sustav bi pridonio stvaranju povjerenja prilikom kupovine polovnih automobila od distributera kao i od drugih ljudi. Konceptualno rješenje omogućuje kupcima jednostavnu provjeru podataka preko jedinstvenog identifikacijskog broja koji dobivaju prilikom kupovine automobila. Taj se jedinstveni identifikacijski broj može prikazati i kao vrsta QR koda što bi dodatno pojednostavilo kupcima pretraživanje podataka o automobilu, jednostavnim skeniranjem pomoću mobitela. S obzirom na trajnost podataka unutar lanca opskrbe podaci o proizvodnji automobila bit će dostupni godinama.

6.3. Moguća poboljšanja i nedostatci sličnih programskih rješenja

Upravo zbog trajnosti podataka koja postoji u opskrbnim lancima temeljenim na tehnologiji lanca blokova dolazimo do problema skalabilnosti takvih sustava. U prikazanom konceptu opskrbni lanac se sastoji od četiri sudionika, dok stvarni opskrbni lanci sadrži puno veći broj sudionika. S obzirom da će svaki od sudionika često zapisivati podatke u lanac blokova dolazi do ubrzanog povećanja broja blokova u lancu. Nadalje, svaki će od čvorova mreže posjedovati trenutno stanje lanca blokova što u velikim opskrbnim lancima znači veliki broj istih zapisa. U usporedbi s klasičnim centraliziranim bazama podataka spremanje velike količine podataka u lanac blokova vrlo je neefikasno. Zbog toga što lanac blokova zauzima puno više memorije, potrebno je puno više sklopovlja za realizaciju takve mreže. Samim povećanjem broja potrebnog sklopovlja za funkcioniranje mreže povećava se i trošak koji je potreban za realizaciju i održavanje mreže. Također, integriranje manjih dobavljača, poput lokalnih tvrtki, u lanac opskrbe temeljen na lancu blokova neće biti lako s obzirom na nedostatak tehničke stručnosti i / ili infrastrukture. Velika je vjerojatnost da će upravo to biti zajednički problem brojnih opskrbnih lanaca temeljenih na lancu blokova.

U realiziranom konceptualnom primjeru početak proizvodnje automobila kreće iskopavanjem sirovina te unošenjem informacija o sirovinama u sustav. Iz toga slijedi da bi ovaj ljudski unos, koji dolazi u ključnoj točki vremenskog slijeda opskrbnog lanca, mogao biti pogrešan, izostavljen ili izmišljen pa bi mogle biti potrebne dodatne provjere kako bi se osiguralo da je ono što se unosi u lanac blokova točno. Praktični primjer je izrađen na Ethereum lancu blokova, što znači na javnoj mreži koja koristi PoW konsenzusnu metodu. Ključni aspekt transparentnosti ovisi ne samo o integritetu podataka, već i o njihovoj sposobnosti da se ne mijenjaju. Opskrbni lanac temeljen na lancu blokova ne bi trebao biti javan odnosno dostupan svima. Na javnim mrežama aspekt nepromjenjivosti podataka proizlazi iz oslanjanja na PoW i činjenicu da bi trebala ogromna računalna snaga za izmjenu podataka prethodnog bloka. Opskrbni lanci unutar industrija trebaju funkcionirati na privatnoj mreži lanca blokova koja će biti osigurana dodatnim slojem autentifikacija kao i autorizacije. Na taj će način sudionici zaštititi svoje podatke od javnosti i od pristupanja neovlaštenih sudionika u mrežu. U slučaju da je infrastruktura lanca blokova (posebno čvorova) pod nadzorom jedne strane postoji mogućnost promjene podataka koji se nalaze u lancu blokova, na što treba obratiti pozornost. Također, korištenjem PoW konsenzusne metode stvaraju se veliki utrošci energije prilikom rudarenja blokova. Početno, svi čvorovi koji žele sudjelovati moraju posjedovati specifična sklopovlja poput procesora i grafičkih kartica, kako bi pokušali izrudariti blok. Jednako tako, rudari istovremeno pokušavaju izrudariti isti blok, a samo jedan to

uspijeva, čime se stvara energetski zahtjevna mreža. Upravo se iz tog razloga Ethereum lanac blokova planira prebaciti na PoS metodu za postizanje konsenzusa unutar mreže.

Prilikom razvoja velikih opskrbnih lanaca temeljenih na lancu blokova treba posvetiti pažnju testiranju sustava. Jednom kada je pametni ugovor objavljen, ne postoji način da ga se izmjeni. Stoga je od velike važnosti dobro testirati sustav prije objavljivanja. Trenutni testovi i koncepti predviđaju da će tehnologija lanca blokova podržavati složene i raznolike globalne opskrbe lance. Ovom modelu svojstvena je vjerojatnost da će konkurentski proizvođači, dobavljači i logističke tvrtke biti sudionici istog lanca blokova čime će imati pristup istom zapisu podataka, a samim time i informacijama konkurenata. Sukladno tome, iako tehnologija koja podupire lanac opskrbe može pružiti veću transparentnost i povjerenje, oni koji povezuju digitalne opskrbne lance morat će razmotriti na koji način će upravljati pravima pristupa informacijama za korisnike koji žele osigurati da njihovi podaci neće biti dijeljeni s konkurentima. U opskrbnim lancima gdje se pristup podacima ne može prilagođavati, što je sve izazovnije povećanjem broja korisnika u opskrbnom lancu, postoji potencijal da stranke budu manje spremne dijeliti vrijedne podatke unutar opskrbnog lanca te da korisnici preispituju istinitost podijeljenih podataka. To bi, naravno, moglo oštetiti vrijednost lanca opskrbe temeljenog na tehnologiji lanca blokova čiji se integritet i vrijednost u konačnici temelje na samoj legitimnosti podataka koje pruža svaki sudionik. Potencijalne koristi od transparentnosti i povjerenja koje lanac blokova nudi opskrbnom lancu su jasne. Međutim, ove prednosti treba uravnotežiti s ograničenjima koja su svojstvena relativno novoj tehnologiji. Konkretno, trošak je implementacije tehnologije lanca blokova trenutno značajan. Postoje i potencijalna infrastrukturna ograničenja koja mogu značiti da se eksponencijalni rast potreban za potporu globalnim lancima opskrbe ne može održavati ili servisirati kako bi se osigurala dovoljna brzina da zadovolji potrebnu funkcionalnost. U svrhu transparentnosti treba pažljivo upravljati unosom podataka u lanac blokova te načinom prikupljanja podataka i obradom istih. Također od ključne je važnosti da svi sudionici opskrbnog lanca koriste tehnologiju lanca blokova.

7. ZAKLJUČAK

Opskrbni lanci temeljeni na tehnologiji lanca blokova mogu uvelike doprinijeti transparentnosti podataka čime će smanjiti ljudske pogreške, prevare i kršenje dogovorenih uvjeta. Lanac blokova može osigurati kontinuitet informacija u opskrbnim lancima kroz svoja svojstva nepromjenjivosti i nepobitnosti. Upravo tehnologija lanca blokova može pružiti rješenje sigurnosnih problema unutar lanaca opskrbe i zajamčiti kontrolu integriteta i transparentnosti proizvoda te njihovih sadržaja. Vrijeme provjere valjanosti, integriteta i povjerljivosti podataka može se skratiti upravo implementacijom tehnologije lanca blokova koja sama po sebi sadržava svojstva valjanosti, integriteta i povjerljivosti. Dakle, mreža opskrbnog lanca poboljšana tehnologijom lanca blokova može biti još veća i sigurnija na globalnoj razini od one tradicionalne.

U diplomskom radu razvijeno je konceptualno programsko rješenje opskrbnog lanca automobilske industrije temeljenog na lancu blokova, a realizirano je pomoću pametnih ugovora napisanih unutar Ethereum lanca blokova. Razvojem praktičnog primjera prikazana je transparentnost koju nudi tehnologija lanca blokova kao i rješenje problema legitimnosti i nepromjenjivosti podataka, što su glavni problemi globalnih opskrbnih lanaca. Implementacijom tehnologije ubrzan je postupak prebacivanja vlasništva između sudionika mreže te vremena koje je potrebno za prijenos sredstava s jednog računa na drugi. Praćenjem cijelog proizvodnog ciklusa automobila smanjuje se mogućnost slučajne ili namjerne prevare i povećava se povjerenje kod mogućih kupaca automobila. Također, sustav omogućava proizvođačima jasan revizijski trag koji će im pomoći u smanjenju broja mogućih kvarova i boljem upravljanju rizika. Između ostalog, omogućava jednostavno praćenje etičkih politika trgovanja i ispunjavanje potrebnih regulativa industrije. Realizirani praktični primjer opskrbnog lanca definitivno je primjenjiv i u drugim granama industrije koje zahtijevaju povećanje sigurnosti, transparentnosti i povjerenja između svih sudionika sustava.

LITERATURA

[1] M. Gupta, Blockchain for Dummies, IBM, John Wiley & Sons, Inc., Hoboken, New York, 2017.

[2] S. Nakamoto, Bitcoin: A Peer-to_Peer Electronic Cash System, 2008., dostupno na: https://bitcoin.org/bitcoin.pdf [28. svibanj 2020.]

[3] L. Lamport, R. Shostak, M. Pease, The Byzantine Generals Problem, ACM Transactions on Programming Languages and Systems, No. 3, Vol. 4, str. 382. – 401., srpanj 1982.

[4] V. Gramoli, From Blockchain Consensus Back to Byzantine Consensus, Future Generation Computer Systems, Vol. 107, str. 760. – 769., lipanj 2020.

[5] A. Gervais, V. Glykantzis, G. Karame, H. Ritzdorf, K. Wust, S. Čapkun, On the Security and Performance of Proof of Work Blockchains, Proceedings of the 2016., ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016., str. 3.–16.

[6] S. King, S. Nadal, PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stale, 2012.

[7] S. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, V. Sassone, PBFT vs Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain, Research Center of Cyber Intelligence and Information Security, Sapienya University of Rome, University of Southampton, 2018.

[8] M. Castro, B. Liskov, Practical Byzantine Fault Tolerance, Laboratory for Computer Science, Massachusetts Institute of Technology, 1999.

[9] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, W. Shi, On Security Analysis of Proof-of-Elapsed-Time (PoET), Department of Computer Science, University of Houston, Stabilization, Safety, and Security of Distributed Systems: 19th International Symposium, Vol. 10616, str. 282. – 297., 2017.

[10] J. Aumasson, Serious Cryptography: A Practical Introduction to Modern Encryption, Open WorldCat, 2018.

[11] A. Khan, S. Basharat, M. Riaz, Analysis of asymmetric cryptography in information security based on computational study to ensure confidentiality during information exchange, 2018.

[12] W. Diffie, M. Hellman, New Directions in Cryptography, IEEE Trasactions on information theory, Vol. 22., No. 6., studeni 1976.

[13] R. L. Rivest, A. Shamir, L. Adleman, A Method for Obtaining Digital Signatures nad Public-Key Cryptosystems, Communications of the ACM, Vol. 21, No. 2, str. 120. – 126., veljača 1978.

[14] L. Henzen, J. Aumasson, W. Meier, R. Phan, VLSI Characterization of the Cryptographic Hash Function BLAKE, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 19, no. 10, str. 1746-1754, 2011.

[15] S. Nobin, Difference between blokchain and digital signature, Medium, 2019., dostupno na: https://medium.com/@smsnobin77/difference-between-blockchain-and-digital-signature-311a91b1445c , [10. rujan 2020.]

[16] R. Merkle, A Digital Signature Based on a Conventional Encryption Function, Advances in Cryptology – CRYPTO '87, Vol. 293, str. 369. – 378., 1988.

[17] K. Scarfone, N. Roby, P. Mell, D. Yaga, Blockchain Technology Overview, National Institute of Standards and Technology, listopad 2018.

[18] D. Tapscott, A. Tapscott, Blockchain Revolution: How the Technology behind Bitcoin is Changing Money, Business, and the World, Portfolio / Penguin, 2016.

[19] N. Szabo, Formalizing and Securing Relationships on Public Networks, First Monday, Vol. 2, No. 9, rujan 1997.

[20] G. Peters, E. Panayi, Understanding Modern Banking Ledgers Through Blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money, SSRN Electronic Journal, studeni 2015.

[21] J. Wisner, K. Tan, G. Leong, Principles of Supply Chain Management: A Balanced Approach, South-Western, No. 3, 2011.

[22] What is a Supply Chain, Corporate Finance Institute, dostupno na: https://corporatefinanceinstitute.com/resources/knowledge/strategy/supply-chain/ , [10. rujan 2020.]

[23] A. Litke, D. Anagnostopoulos, T. Varvarigou, Blockchains for Supply Chain Management: Architectural Elements and Challenges Towards a Global Scale Deployment, Logistics, Vol. 3, No. 1, siječanj 2019.

[24] N. Vyas, A. Beije, B. Krishnamachari, Blockchaon and the Supplz Chain: Concepts, Strategies and Practical Applications, Kogan Page, 2019.

[25] D. Tapscott, Supply Chain Revolution: How Blockchain Technology is Transforming the Global Flow of Assets, Open WorldCat, 2020.

[26] A. Tapscott, Financial Services Revolution: How Blockchaion is Transforming Money, Markets, and Banking, Open WorldCat, 2020.

[27] T. Bocek, B. Rodrigues, T. Strasser, B. Striller, Blockchains Everywhere - A Use-case of Blockchains in the Pharma Supply-Chain, Communication Systems Group, Department of Informatics, University of Zurich, 2017.

[28] C. Hulseapple, Block Verify Uses Blockchains to End Counterfeiting and 'Make World More Honest', 2015., dostupno na: https://cointelegraph.com/news/block-verify-uses-blockchains-to-end-counterfeiting-and-make-world-more-honest [18. kolovoz 2020.]

[29] J. Huertas, H. Liu, S. Robinson, Eximchain: Supply Chain Finance solutions on a secured public, permissioned blockchain hybrid, Whitepaper Eximchain, ožujak 2018.

[30] Modum, Blockchain in Supply Chain – A Pharmaceutical Use Case Example, 2019., dostupno na: https://modum.io/news/blockchain-supply-chain-pharmaceutical-use-case-example [19. kolovoz 2020.]

[31] J. Rohr, SAP Helps Swiss Post and Modum Build Blockchain Solutions for Temperature-Controlled Logistics, 2019., dostupno na: https://news.sap.com/2019/03/swiss-post-modum-blockchain-solution-temperature-logistics/ [19. kolovoz 2020.]

[32] VeChain Foundation, VeChain Whitepaper 2.0, VeChain Foundation, 2019., dostupno na: https://www.vechain.org/whitepaper/#bit_65sv8 [19. kolovoz 2020.]

[33] T. Hardin, D. Kotz, Amanuensis: Information Provenance for Health-Data Systems, Information Processing & Management Special Issue: Blockchain for Information Systems Management and Security, svibanj 2020.

[34] F. Yiannas, A new era of food transparency powered by blockchain, Blockchain for Global Development, Innovations: Tehnology, Governance, Globalization, Vol. 12, No. 1, srpanj 2018.

[35] IBM, Using blockchain technology to address worldwide food safety, 2017., dostupno na: https://techxplore.com/news/2017-08-blockchain-technology-worldwide-food-safety.html [18. kolovoz 2020.]

[36] IBM, Maersk and IBM Unveil First Industry-Wide Cross-Border Supply Chain Solution on Blockchain, 2017., dostupno na: https://www-03.ibm.com/press/us/en/pressrelease/51712.wss [18. kolovoz 2020.]

[37] E. Petersson, K. Baur, Impacts of Blockchain Technology on Supply Chain Collaboration, Jönköping International Business School, Jönköping University, svibanj 2018.

[38] S. Aich, S. Chakraborty, M. Sain, H. Lee, H. Kim, A Review on Benefits of IoT Integrated Blockchain based Supply Chain Management Implementations across Different Sectors with Case Study, International Conference on Advanced Communications Technology (ICACT), veljača 2019.

[39] Ambition2039: Our path to sustainable mobility, Daimler, 2019., dostupno na: https://www.daimler.com/documents/investors/news/capital-market-releases/daimler-mercedes-benz-ir-release-en-20190513.pdf [25. kolovoz 2020.]

[40] J. Melville, From Stone to Phone: Modern day cobalt slavery in Congo, Byline times, 2020, dostupno na: https://bylinetimes.com/2020/06/19/from-stone-to-phone-modern-day-cobalt-slavery-in-congo/ [25. kolovoz 2020.]

[41] D. Miehle, D. Henze, A. Seitz, A. Luckow, B. Bruegge, PartChain: A Decentralized Traceability Application for Multi-Tier Supply Chain Networks in the Automotive Industry, IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON), str. 140 – 145., travanj 2019.

[42] M. Pustišek, N. Bremond, A. Kos, Electric Switch with Ethereum Blockchain Support, ipsitransactions, 2018.

[43] Metamask, dostupno na: https://metamask.io/ , [28. kolovoza 2020.]

[44] Metamask Ether Faucet, dostupno na: https://faucet.metamask.io/ , [28. kolovoza 2020.]

[45] Etherscan, Robsten Explorer, dostupno na: https://ropsten.etherscan.io/ , [2. rujan 2020.]

SAŽETAK

U ovom diplomskom radu analizirana je tehnologija lanca blokova te mogućnost njezine primjene u globalnim opskrbnim lancima. Prikazana je struktura blokova i kriptografske metode unutar lanca blokova. Osim toga analizirani su pametni ugovori i konsenzusne metode korištene za postizanje dogovora između čvorova mreže. Predstavljeni su problemi tradicionalnih globalnih opskrbnih lanaca koje tehnologija lanca blokova može riješiti. Izrađen je model sustava i konceptualno programsko rješenje opskrbnog lanca automobilske industrije temeljeno na tehnologiji lanca blokova. Praktični je primjer izrađen pomoću pametnih ugovora napisanih u programskom jeziku Solidity te objavljen na Ethereum mreži lanca blokova. Ostvareni praktični primjer omogućuje praćenje proizvodnog ciklusa automobila od sirovine do distributera. Pomoću jedinstvenog identifikacijskog broja kupcu je omogućen pregled životnog ciklusa automobila, uključujući informacije o svakom automobilskom dijelu i sirovini korištenom za proizvodnju. Ispitivanjem rješenja utvrđeno je da opskrbni lanci temeljeni na tehnologiji lanca blokova pružaju kupcima stvarne i potrebne informacije o proizvodu prilikom kupovine, pojednostavljuju pronalaženje izvora proizvoda, omogućuju provjeru postupanja s proizvodom prilikom proizvodnje, te pružaju verifikaciju svih proizvođača unutar opskrbnog lanca čime se stvara povjerenje kod kupaca.

Ključne riječi: automobilska industrija, konsenzus, lanac blokova, opskrbni lanac, pametni ugovori.

ABSTRACT

Tittle: Supply chain monitoring system in the automotive industry based on blockchain technology

The master thesis analyses blockchain technology and possible use cases of its application in global supply chains. The block structure and cryptographic methods within the blockchain are presented, while smart contracts and consensus methods used for reaching agreements between network nodes are analysed. The problems of global supply chains that blockchain technology can solve are presented. A conceptual model and software solution based on blockchain technology have been developed for the supply chain of the automotive industry. A solution was created using smart contracts written in Solidity programming language and deployed on the Ethereum blockchain network. Created solution enables monitoring of car production cycles, from raw materials to car distributors. Using unique identification number, customers can view the whole life cycle of the car, including information about each car part and raw material used in production. Testing the created solution found that supply chains based on blockchain technology provide customers with necessary product information at the time of the purchase, simplify finding product source, verify product handling during manufacturing and provide verification for all manufacturers within the supply chain creating necessary trust with customers.

Keywords: automotive industry, consensus, blockchain, supply chain, smart contracts.

ŽIVOTOPIS

Lovro Pejčić rođen je 13. veljače 1996. godine u Osijeku. Završava osnovnu školu „Dobriša Cesarić“ 2011. godine te iste upisuje Jezičnu gimnaziju u Osijeku. Po završetku srednje škole ostvaruje upis na preddiplomski sveučilišni studij Elektrotehničkog fakulteta u Osijeku, smjer računarstvo. Preddiplomski sveučilišni studij računarstva završava 2018. godine te stječe zvanje prvostupnika inženjera računarstva (univ. bacc. ing. comp.). Obrazovanje iste godine nastavlja na diplomskom studiju automobilskog računarstva i komunikacija na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Od 1. veljače 2019. godine radi kao praktikant u tvrtki Barrage na poziciji *backend* razvojnog inženjera.

_____

Lovro Pejčić

PRILOZI

Prilog 1. Dokument diplomskog rada

Prilog 2. Pdf diplomskog rada

Prilog 3. Programsko rješenje pametnih ugovora napisanih u programskom jeziku Solidity za opskrbni lanac automobilske industrije temeljen na tehnologiji lanca blokova