Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Domen Petrič

Uporaba verige blokov v avtomobilski industriji

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Mojca Ciglarič

Ljubljana, 2018

Avtorske pravice. Rezultati magistrskega dela so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljanje ali izkoriščanje rezultatov magistrskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

©2018 Domen Petrič

Zahvala

Zahvaljujem se mentorici izr. prof. dr. Mojci Ciglarič za pomoč, vodenje in nasvete pri izdelavi magistrskega dela. Rad bi se zahvalil tudi svojim bližnjim in prijateljem, ki so me ves čas študija spodbujali in mi stali ob strani.

Domen Petrič, 2018

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
BFT	Byzantine fault tolerance	bizantinska toleranca napak
EVM	etherum virtual machine	Etherumova navidezna naprava
SVM	Support vector machine	metoda podpornih vektorjev
IOT	Internet of things	internet stvari
SVM	Support vector machine	metoda podpornih vektorjev
SGX	Software Guard Extensions	ukaz v naboru procesorjev Intel
REST	Representational State Transfer	tip spletne storitve
API	Application programming interface	vmesnik za namensko programiranje
WHO	World health organization	Svetovna zdravstvena organizacija
TWh	Tera watt hour	teravatna ura
ZIO	Trusted execution environment	zaprto izvajalno okolje

Povzetek

Naslov: Uporaba verige blokov v avtomobilski industriji

Algoritmi oziroma mehanizmi soglasja so ključni košček v sestavljanki verige blokov. Razvoj različnih mehanizmov soglasja je v porastu in prav ta košček bo kritično vplival na splošno adaptacijo in uporabo verig blokov v vsakdanjem življenju. V delu smo preverili, kako delujejo posamezni predlagani mehanizmi soglasij. Primerjali smo dokaz dela in dokaz deleža kot prva implementirana algoritma z dokazom pretečenega časa in dokazom požiga. Primerjali smo tudi ideje algoritmov soglasja, ki trenutno še nimajo praktične implementacije, kot dokaz sreče in dokaz oblasti. Raziskali smo, katere lastnosti mora mehanizem soglasja izpolnjevati, da pomeni izboljšavo tehnologije verige blokov. Ugotovljene lastnosti so nam postale merilo za primerjavo algoritmov soglasja. V delu smo pokazali, da lahko problem, ki ga je smiselno reševati s tehnologijo verige blokov, rešimo, in to celo bolje kot s pomočjo algoritma dokaza dela. Za izdelavo smo uporabili ogrodje Sawtooth z dokazom pretečenega časa, kot mehanizmom soglasja. Implementirali smo beležko vzdrževanj in popravil vozila, dobljeno rešitev pa smo primerjali z obstoječimi rešitvami servisne knjižice ter dokazali uporabnost in boljše delovanje verige blokov z alternativnim algoritmom soglasja.

Ključne besede

veriga blokov, algoritem soglasja, avtomobilska industrija

Abstract

Title: Use of Blockchain in automotive industryConsensus algorithms or mechanisms are a key piece in the blockchain puzzle. The development of different consensus mechanisms is on the rise, and this particular part of blockchain will critically influence the general adaptation and use of blockchains in everyday life. In this thesis, we examined how each proposed consensus mechanism works. We compared the proof of work and proof of stake as the first implemented algorithms with the proof of elapsed time and proof of burn. We also compared the ideas of consensus algorithms that do not currently have a practical implementation, such as proof of luck and proof of authority. We investigated what properties a consensus mechanism needs to fulfil to improve the technology of the blockchain. The established properties became the criteria for comparing consensus algorithms. We have shown that a problem that can be solved with blockchain technology can be implemented even better than with the use of a proof of work. For the implementation, we used the Sawtooth framework with a proof of elapsed time as a consensus mechanism. We implemented a maintenance and repair notebook of the vehicle, compared the solution obtained with the existing solutions of the service booklet, and proved the usability and superior performance of blockchain with alternative consensus an algorithm.

Keywords

blockchain, consensus algorithm, automotive industry

Poglavje 1

Uvod

Živimo v dobi interneta, ko je svet povezan bolj kot kadar koli prej. Interakcija ljudi, ki se nikdar niso fizično srečali, je danes povsem normalna, tako v zasebnem kot poslovnem svetu. Internet je bil ob svojem nastanku implementiran s predpostavko, da se strani, ki si izmenjujejo podatke, poznajo in si zaupajo. S širitvijo interneta danes ta predpostavka ne velja več. Ljudje, ki si izmenjujejo informacije prek interneta, se ne poznajo in si ne zaupajo vedno. Ena izmed možnih rešitev pomanjkanja zaupanja bi lahko bile mednarodne organizacije, ki bi jim uporabniki zaupali in bi jamčile za iskrenost vseh vpletenih strani. Druga rešitev, ki rešuje problem, pa je vzklila iz skupka tehnologij, povezanega v celoto. Prvi je tehnologijo opisal in kasneje implementiral Satoshi Nakamoto v delu Bitcoin: A Peer-to-Peer Electronic Cash System [1]. To je tehnologija, ki se še danes razvija in jo imenujemo veriga blokov. Strokovnjaki analitične hiše Gartner pravijo, da bo tehnologija za splošno rabo zrela v nekaj letih. Na verigo blokov lahko gledamo kot na poseben tip podatkovne baze, katere kopijo hranijo vsa vozlišča, povezana v skupno omrežje. Odločitev zapisovanja podatkov v verigo blokov je težka. Le-ta sicer prinaša zaupanje v podatke, a je vzpostavitev verige blokov zahtevna.

V delu se bomo osredotočili predvsem na točno določen del verige blokov. Nakamoto je za algoritem soglasja uporabil dokaz dela. Ta je zaradi svoje

implementacije računsko zelo zahteven in energijsko potraten. Ob razširitvi omrežja se je algoritem pokazal kot slabo nadgradljiv. V delu se bomo osredotočili na iskanje alternativnega mehanizma soglasja, ki bi soglasje dosegal energijsko učinkoviteje in hitreje. Želimo, da je algoritem razširljiv in varen.

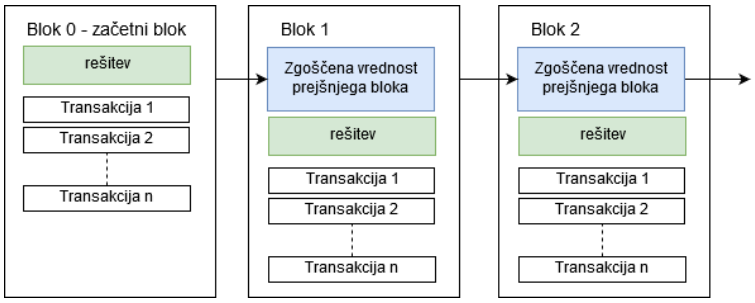
V delu bomo raziskali mehanizme soglasja, ki so že implementirani in tiste, ki so šele idejno zasnovani. Poiskali bomo merila za primerjavo algoritmov soglasja. Mehanizme soglasja bomo nato primerjali in izbrali tiste, ki so najperspektivnejši. Z enim izmed bolje ocenjenih mehanizmom soglasja bomo implementirali sistem za hranjenje podatkov o zgodovini popravil vozila in tako pokazali uporabnost novih mehanizmov soglasja.

V drugem poglavju bomo opisali delovanje verige blokov in različne tipe verig blokov. Skrb za komunikacijo med vozlišči in dogovor vozlišč upravljajo mehanizmi soglasja, ki bodo opisani v tretjem poglavju. V tretjem poglavju bomo analizirali tudi merila za primerjavo algoritmov soglasja. Izvedli bomo primerjavo splošnih implementacij mehanizmov soglasja in jih ocenili. V četrtem poglavju bomo pregledali, po našemu mnenju najperspektivnejša ogrodja v razvoju na področju verige blokov in izbrali najprimernejše za razvoj svojega projekta. Naš projekt se osredotoča na hranjenje zgodovine popravil vozila, ki lahko služi kot primer projekta, primeren za rešitev s pomočjo verige blokov. Projekt bomo implementirali, samo implementacijo pa opisali v petem poglavju. Delo bomo zaključili s šestim poglavjem, kjer bomo poudarili ključne ugotovitve in se ozrli v prihodnost razvoja verig blokov.

Poglavje 2

Veriga blokov

Ideja verige blokov je bila prvič objavljena pod avtorstvom Satoshija Nakamota 1. oktobra leta 2008 v delu Bitcoin: A Peer-to-Peer Electronic Cash System [1]. Šlo je za skupek že prej opisanih tehnologij. Tako je bil Nakamotov dokaz dela opisan že v Backovem članku HashCash [2]. Dokaz dela se je ob povečanju vozlišč v omrežju verige blokov izkazal za neprimernega, zato so različni avtorji predlagali svoje ideje, ki so soglasje dosegale na nove različne načine. Tako so Lamport [3], Castro in Liskov [4] v svojih delih reševali problem bizantinske tolerance napak. Ideja dokaza dela se je deloma prenesla v nov mehanizem, ki je poudarek na procesorski moči zamenjal s poudarkom na prostem diskovnem prostoru. Mehanizem je bil opisan v delu Proofs of space [5]. Neustreznost dokaza dela za obvladovanje soglasja v verigi blokov je pripeljala še do številnih drugih raziskav na področju mehanizmov soglasja. King in Nadal sta tako v verigi blokov PeerCoin prva implementirala dokaz deleža [6]. Podjetje Intel pa je razvilo svoj mehanizem soglasja po imenu dokaz pretečenega časa [7], medtem, ko so v verigi Slimcoin implementirali dokaz zgorelosti [8]. Predlagani so bili tudi drugi algoritmi, kot so dokaz oblasti [9] in dokaz pomembnosti [10]. V letu 2017 pa so Milutinovic, He, Wu in Kanwal objavili članek Proof of Luck: an Efficient Blockchain Consensus Protocol [11], ki preko prejšnjih algoritmov soglasja pripelje do razmisleka o novem mehanizmu soglasja, dokazu sreče. Nekateri avtorji so v razlagah



Slika 2.1: Veriga blokov.

lastne implementacije algoritma soglasja le tega primerjali z obstoječim. Primera takih člankov sta The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication [12] in PBFT vs Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain [9].

Satoshi Nakamoto je v svojem delu poskušal opisati implementacijo sistema, ki bi služil kot nadomestilo bančnega sistema. Sistem je želel odstraniti centralno avtoriteto iz sistema in implementirati sistem, ki vzpostavlja zaupanje med vsemi deležniki ter popolno sledljivostjo in nespremenljivostjo podatkov. Sam dokument je leta 2009 dobil tudi implementacijo algoritma in zaživel. Danes to tehnologijo označujemo kot verigo blokov 1.0 (angl. Blockchain 1.0).

S poslovnega stališča je veriga blokov tehnologija, kjer si enakovredna vozlišča izmenjujejo podatke brez potrebe po centralni avtoriteti. Veriga blokov je porazdeljena baza podatkov, ki ne omogoča popravljanja ali brisanja podatkov, ko so bili ti enkrat potrjeni s soglasjem med večino vozlišč. Veriga blokov je sestavljena iz potrjenih blokov transakcij. Poimenovanje transakcija v tem primeru pomeni spremembo statusa oziroma preprosteje neki zapis, ki ga veriga blokov hrani. V verigi blokov imamo vedno tudi referenco (zgoščeno vrednost) na prejšnji blok, kot prikazuje slika 2.1.

S tem podatkom tehnologija zagotavlja veriženje in nespremenljivost hranjenih podatkov. Izjema je začeni blok (angl. genesis block), ki reference na prejšnji blok nima. Začetni blok je namreč ročno ustvarjen zapis, ki začenja

verigo blokov. Glede na tip verige blokov imamo lahko v bloku še druge podatke, kot so časovni žigi ali rešitve uganke.

V verigo blokov se transakcije dodajajo v bloke, bloki pa se potrdijo v verigo s pomočjo mehanizma porazdeljenega soglasja. Mehanizem porazdeljenega soglasja je najpomembnejši del verige blokov, saj nam preko svojega algoritma zagotavlja, da so podatki vpisani na verigo blokov in varno shranjeni v večino vozlišč, ki sodelujejo v porazdeljenem omrežju verige blokov. Mehanizem soglasja tako nadomešča zaupanje, ki ga v siceršnjih sistemih zagotavljajo centralne institucije (npr. banka, vlada ...).

Prva implementacija verige blokov z imenom Bitcoin je hotela nakazati nov bančni sistem, a uporabnost verige blokov sega širše. Danes obstaja veliko projektov, ki poskušajo verigo blokov uporabiti na drugih področjih. Od tu izhaja uvedba koncepta pametne lastnine (angl. smart property). Ideja uvedbe pametne lastnine omogoča, da materialne in nematerialne stvari elektronsko označimo in jih tako preko pametnih pogodb izmenjujemo za valuto ali druge pametne lastnine. Lastništvo se tako zapiše v verigo blokov in je trajno zapisano, prav tako njegova zgodovina. Pametne pogodbe nam pri izmenjavi pomagajo. Pametna pogodba je zbirka ukazov (računalniški program), ki se samodejno sproži, ko so določeni prej definirani pogoji izpolnjeni.

Projekti za uporabo verige blokov so še v povojih, a glede na trend analitična hiša Gartner v krivulji priljubljenosti tehnologij ocenjuje, da do široke uporabe verige blokov v industriji manjka še približno pet let [13]. Na krivulji popularnosti so verige blokov že prešle vrh priljubljenosti in so v rahlem padanju na ravni tehnologij, ki so dozorele.

Kljub napovedim lahko že v času pisanja tega dela, opazimo prve uspešne projekte na energetskem področju in področju sledenja izvoru produktov, ki verigo blokov že upešno uporabljajo v industriji in ustrezajo pravnim zahtevam.

Projekti uporabljajo najrazličnejše mehanizme soglasja. Veliko verig blokov prve generacije uporablja mehanizem, imenovan dokaz dela. Poleg dokaza dela je eden izmed pogostejših mehanizmov soglasja novejši dokaz deleža.

V nadaljevanju dela bomo pregledali tudi nekatere druge predlagane in implementirane mehanizme soglasja. Zavedati se moramo, da za različnimi mehanizmi stojijo različne ideje o načinu potrjevanja blokov, za vsak mehanizem pa imamo vsaj nekaj različnih implementacij, ki se razlikujejo po varnosti, energetski učinkovitosti, uporabnosti ...

Pri splošni adaptaciji verige blokov se bo razvoj moral usmeriti v probleme, opisane v nadaljevanju.

- **Energetska učinkovitost** - Mehanizem soglasja po imenu dokaz dela je preko rudarjev v letu 2018 za zagotovitev varnosti verige Bitcoina porabil toliko električne energije kot manjše države [14]. Veriga blokov mora postati energetsko učinkovitejša, preden se preseli v splošno rabo.

- **Nadgradljivost** - Na svetu je približno 8 milijard ljudi, ki večinoma večkrat dnevno uporabljajo najrazličnejše transakcije. Za lažjo predstavo poglejmo primer finančnih nakazil preko elektronskih sistemov. Uporaba kreditnih kartic, transakcij in drugih načinov prenakazil denarja se vztrajno veča, nekatere države celo spodbujajo brezgotovinsko poslovanje oziroma nameravajo gotovino v prihodnjih letih ukiniti. Dnevno se na Zemlji zgodi milijarde transakcij. Kako bo veriga blokov to podprla in skrbela, da ob povišanju števila transakcij ne bo prihajalo do preobremenjenosti? V letu 2017, v času rekordnih vrednosti Bitcoina, le ta veriga blokov ni bila sposobna sprotno potrjevati transakcij, kljub temu, da se s kripto valutami ukvarja relativno malo ljudi.

- **Zaupanje ljudi** - V trenutnem sistemu ljudje zaupajo bankam oziroma neki centralni avtoriteti, ki skrbi za prenos dobrin med strankami. Kot ovira pri adaptaciji verige blokov lahko pričakujemo nezaupanje ljudi v samouravnalni elektronski sistem [15].

- **Migracija** - Ob odločitvi za hranjenje podatkov v verigi blokov se je treba vprašati tudi, kaj bomo naredili z obstoječimi podatki. Ali se bodo podatki še vedno hranili v starih sistemih in jih bomo kot take vzdrževali ali se bodo vsi dosedanji podatki migrirali v verigo blokov?

- **Pravni vidik** - Kako bodo države uravnavale uporabo verigo blokov. Kakšni bodo postopki za identifikacijo glede na psevdoanonimnost uporabnikov verige blokov. Pojavi se tudi vprašanje hranjenja in objavljanja osebnih podatkov. Ne želimo si namreč hraniti osebnih podatkov v nespremenljivi, javni in neizbrisljivi podatkovni strukturi.

- **Preprečevanje zlorab** - Delno v povezavi s prejšnjo točko bo treba razviti mehanizme za sledenje finančnemu toku denarja in s tem preprečevati pranje denarja, financiranje terorizma ...

- **Kvantno računalništvo** - Kakšni naj bodo javni in zasebni ključi, da bodo da z razvojem kvantnega računalništva še vedno varni? Kako zahtevne morajo biti uganke pri mehanizmu soglasja dokaza dela, da ne bo mogoče v prihodnosti izračunati alternativne napačne stranske verige in tako spremeniti zgodovine transakcij?

2.1 Overjanje uporabnikov verige blokov

V informacijskih sistemih običajno uporabniki izvedejo identifikacijo s pomočjo uporabniškega imena in gesla, pridobljenega ob registraciji. Tak postopek overitve imenujemo eno-faktorska overitev, ker obstaja samo en podatek, s katerim uporabnik dokaže svojo identiteto. Tak način prijave ni varen za vsa področja, zato se v sodobnih informacijskih rešitvah pogosto uporablja več-faktorska overitev. Za drugega izmed faktorjev se pogosto uporablja fizična lastnina. Po navadi gre za enkratno generiran, časovno spremenljiv žeton. Oseba, ki se vpisuje v sistem, s tem dokaže, da uporabniško ime in geslo nista bili odvzeti pravemu uporabniku, saj ima oseba tudi lastnino uporabnika. Kot tretji faktor se uporabljajo biometrične značilnosti posameznika, kar pokaže še na posameznikovo prisotnost na kraju overjanja. Tu se lahko uporabljajo sistem za skeniranje prstnega odtisa, sistem za skeniranje očesa ali katera izmed človekovih drugih unikatnih lastnosti.

Veriga blokov za overitev uporablja, sistem javnih in zasebnih ključev.

Javni ključ v verigah blokov pogosto pomeni tudi naslov, kjer se hranijo žetoni ali kovanci. Pri zahtevi po novem paru javnega in zasebnega ključa verige blokov večinoma ustvarijo par s pomočjo kriptografije eliptične krivulje.

V Bitcoinu je zasebni ključ naključno izbrana 256-bitna vrednost v intervalu, ki ga določa standard SECP256K1 ECDSA. Javni ključ lahko kasneje pridobimo iz zasebnega ključa, saj nam to omogoča kriptiranje eliptičnih krivulj po standardu SECP256K1. V Bitcoinu se tako pridobljen javni ključ vstavi najprej v zgoščevalno funkcijo SHA256, nato pa rezultat pretvori s pomočjo zgoščevalne funkcije RIPEMD-160. Rezultat zgostitvene funkcije je 160-biten niz, ki predstavlja javni ključ. Temu nizu se nato s sprednje strani doda številka različice. Celoten niz se nato zakodira po kodni tabeli Base58Check. Dobljeni rezultat je niz, dolg med 26 in 35 znaki, ki služi kot naslov denarnice. Podobno se generirajo tudi javni in zasebni ključi ter naslovi denarnic v verigi Ethereum.

2.2 Pametne pogodbe, pametna lastnina in porazdeljene aplikacije

O verigi blokov 1.0 govorimo, ko govorimo o verigah blokov, katerih namen je zgolj nadomestiti banke. To so verige blokov, pri katerih lahko trgujemo samo s kriptovalutami. V to kategorijo gotovo spada Bitcoin kot prva kriptovaluta.

V drugo generacijo verige blokov spadajo verige, ki se ne ukvarjajo samo s kriptovalutami, ampak tudi z drugimi finančnimi lastninami, kot so obveznice, delnice in podobno. S to generacijo se za finančne lastnine uvaja termin pametna lastnina. Transakcije se izvajajo preko pametnih pogodb.

Uvedba pametne lastnine je sprožila povezavo med fizičnim objektom in njegovo elektronsko referenco. Elektronsko referenco na fizično lastnino imenujemo pametna lastnina (angl. smart property). Pametna lastnina je tako unikatna, ima svojega lastnika in ne more biti dvakrat porabljena ali v lastništvu več oseb. Pametna lastnina se lahko prenese z enega lastnika na

drugega preko pametne pogodbe.

Prve pametne pogodbe je idejno opisal Nick Szabo [16] leta 1996, a je preteklo kar nekaj časa do resnega razvoja na tem področju. Pametna pogodba je bila delno implemetirana že v verigi blokov Bitcoin.

Pametna pogodba je v naprej testirana samostojna programska koda, ki se izvede na verigi blokov varno in nezaustavljivo. Predstavlja poslovno logiko verige blokov.

Pametna pogodba je računalniški program, ki je napisan v programskem jeziku, ki ga ciljni računalnik razume. Program nadomešča pisni poslovni dogovor med različnimi deležniki v pogodbi. Izvede se samodejno in vedno v celoti, ko so vnaprej določena merila izpolnjena. V celoti se mora izvesti tudi v primeru dejavnikov, ki bi tako ali drugače lahko vplivali na izvajanje pogodbe. Pametna pogodba mora biti tako odporna na napake, hkrati pa se mora izvesti v smiselnem času. Pametna pogodba je torej kot atomarni ukaz. Pomembna lastnost pametnih pogodb je tudi njihova determinističnost. Pomembno je, da nam izvedena pametna pogodba vrne vedno enak rezultat, ne glede na katerem vozlišču verige blokov se nahajamo, saj v primeru različnih rezultatov iste pametne pogodbe ne more priti do soglasja vozlišč, ali se blok lahko potrdi ali ne. Posledica tega je zahteva, da je jezik pogodbe deterministične. V ukazih pogodbe se ne smejo uporabljati funkcije, ki bi vrnile različne rezultate (npr. iskanje naključnega števila), hkrati pa mora biti tudi programski jezik primeren, saj mora na primer ne glede na vozlišče enako ravnati pri matematičnih operacijah, kjer se uporabljajo števila z decimalno vejico.

Pametna pogodba je verigi blokov dodala večjo uporabnost, saj omogoča fleksibilnost, programabilnost in nadzor nad akcijami uporabnikov verige blokov glede na poslovno področje [17]. Pametna pogodba je napisana v programskem jeziku, razumljivem predvsem računalnikom in programerjem. Za nadaljnji razvoj pametne pogodbe želijo raziskovalci izdelati prevajalnik, ki bi pametne pogodbe prevedel v človeku prijaznejši jezik in bi tak zapis veljal kot uradni dokument tudi pred sodišči.

Pametne pogodbe se izvajajo na navideznih napravah na posameznih vozliščih. Navidezne naprave se med posameznimi implementacijami verige blokov razlikujejo, a vedno gre za idejo zaprtega deterministčnega Turingovega stroja. Zaprtega v smislu zaprtega izvajalnega okolja, kjer jezik omogoča determinstično končno izvajanje vsake pametne pogodbe. Kot primer navidezne naprave lahko vzamemo Ethereumovo navidezno napravo (EVM), ki ima sklad velikosti 1024 elementov. EMV vpeljuje sistem goriva (angl. gas) za posamezno navidezno napravo. V verigi blokov Ethereum pri vsaki transakciji dodamo poleg plačila transakcije še plačilo goriva. Gorivo služi kot vhodni podatek za navidezno napravo in je omejeno na posamezni ukaz. Gorivo tako predstavlja trajanje izvedbe pametne pogodbe. Tako Ethereum poskrbi, da se vse pametne pogodbe končajo, bodisi uspešno ali v trenutku, ko algoritmu zmanjka goriva. Preprečevanje pogodb, ki bi se izvajale brez konca, varuje omrežje pred napadi zavrnitve storitve, saj so vozlišča v času izvajanja pametne pogodbe nezmožna opravljanja potrjevanja preostalih transakcij na verigi blokov.

2.3 Zasebna in javna veriga blokov

Večina verig blokov, ki jih poznamo, je javna (angl. public). Dostop do podatkov v verigi blokov ima vsak, ki se hoče pridružiti omrežju. Omrežju se v javnih verigah blokov lahko pridruži kdor koli. Javna veriga blokov je za večino primerov uporabe tako tudi najbolj smiselna, saj nudi vse prednosti, ki jih veriga blokov ponuja (transparentnost, zaupanje v podatke, uporabo s strani veliko uporabnikov). Primeri takih verig blokov so Bitcoin, Ethereum ...

Na drugi strani so se pojavile zasebne verige blokov. Prvo je razvil IBM pod imenom project Fabric in jo kasneje predal v nadaljnji razvoj odprtokodni skupini The Linux Foundation. Zasebne verige blokov niso dosegljive na javnih omrežjih,temveč jih imajo lastniki zaprte. Verigi se tako lahko pridružijo le določena vozlišča, katerih dovoljenja mora upravljati centralna av-

toriteta. Smiselnost takih verig blokov je tako velikokrat vprašljiva. Če uporabimo zasebno verigo blokov, se namreč pojavi vprašanje, zakaj za hrambo podatkov ne uporabimo običajne podatkovne baze.

V razvoju so tudi najrazličnejše verige blokov, ki so po svojem značaju nekje med javno in zasebno. Primer take bi bila veriga, kjer bi vsi imeli dostop do podatkov, le določene osebe pa bi imele možnost potrjevanja blokov, vpisa novih transakcij ...

Poglavje 3

Algoritem soglasja

Javne porazdeljene verige blokov potrebujejo samouravnalni sistem, ki deluje na globalni ravni brez enotne centralne avtoritete. Sprejemati in pošiljati mora podatke iz in na vsa vozlišča v omrežju, ki potrjujejo transakcije v verigo blokov. Porazdeljena veriga tako potrebuje učinkovit, pravičen, hiter, varen in zanesljiv algoritem, ki zagotavlja, da so vse opravljene transakcije unikatne in da se vsa vozlišča prej ali slej strinjajo s stanjem verige. Algoritem, ki je sestavljen iz pravil za zagotavljanje vseh zgoraj naštetih lastnosti, se imenuje algoritem oziroma mehanizem soglasja.

Algoritem soglasja (angl. consensus algorithm) je torej računalniški proces, odporen proti napakam, ki predpisuje postopke za zagotovitev dogovora o pravilnosti in enotnosti podatkov oziroma stanju omrežja v porazdeljenih sistemih.

Algoritem soglasja je tako najpomembnejši del tehnologije verige blokov, saj zagotavlja zaupanje v potrjene podatke in poskrbi za dogovor med tisoči vozlišč v sistemu. Kot opisuje Bashir v Mastering blockchain [17], morajo vsi algoritmi soglasja doseči naslednje lastnosti:

- **dogovor (angl. agreement)** - Vsa poštena vozlišča se morajo dogovoriti za enako stanje omrežja.
- **ustavitev (angl. termination)** - Vsa poštena vozlišča končajo proces soglasja in se dogovorijo za končni dogovor.

- **veljavnost (angl. validity)** - Dogovorjeno veljavno vrednost v omrežju, ki jo je predlagalo vsaj eno pošteno vozlišče, privzamejo vsa druga poštena vozlišča.

- **odpornost proti napakam (angl. fault tolerant)** - Algoritem soglasja mora delovati kljub prisotnosti nepoštenih vozlišč (bizantinskih vozlišč).

- **celovitost (angl. integrity)** - Vsako vozlišče lahko predlaga največ eno rešitev v vsakem ciklu dogovarjanja soglasja.

Ločimo več tipov algoritmov soglasij, dva najpogostejša pa temeljita na bizantinski toleranci napak (angl. byzantine fault tolerance) in na določitvi vodje soglasja (angl. leader-based consensus mechanism).

Algoritem bizantinske tolerance napak izhaja iz problema dogovora med bizantinskimi generali. Vsak izmed bizantinskih generalov poveljuje delčku vojske. Vse vojske skupaj obkrožijo mesto, nato pa se morajo generali odločiti, ali bodo mesto napadli ali se bodo umaknili. Za zmago je pomembno, da se skupaj dogovorijo za eno ali drugo akcijo. Problem se pojavi, ker imamo lahko prisotne tudi generale, ki želijo, da akcija propade, zato glasujejo na neoptimalno strategijo ali glasujejo različno glede na to, kateremu generalu podajajo svoj glas. Problem se še poslabša, ker so lahko generali fizično ločeni in morajo svoj glas posredovati preko kurirja. Glas je lahko tako prestrežen, spremenjen ali pa preprosto ne pride do končnega prejemnika. Problem lahko prenesemo na področje verige blokov, kjer posamezno vozlišče predstavlja enega izmed generalov. Vsa vozišča pa se morajo dogovoriti o potrditvi novega bloka. Prvi predlog rešitve je opisal Lamport [3] leta 1987 v delu The Byzantine Generals Problem.

Algoritem določitve vodje soglasja temeljijo na ideji, da se po določenem ključu izbere vozlišče, ki bo naslednje potrdilo blok v verigo. Tako vozlišča medsebojno tekmujejo, kdo bo postal vodja. Lahko gre za loterijski izbor vodje ali za določitev glede na parametre, kot so količina kovancev, starost kovancev ...

Prvi dogovor soglasja, ki je v uporabi še danes, je bil implementiran v verigi blokov Bitcoin. Dokaz dela je še vedno varen algoritem, a je bil zaradi svoje energetske porabe tarča številnih kritik. Za zagotavljanje varnosti podatkov v verigi blokov Bitcoin naj bi v zadnjem letu svetovno porabili toliko električne energije kot majhna država. Trenutna ocena letne porabe električne energije se giblje pri približno 67 TWh [14]. Takšen algoritem ni skladen z željami po svetovnem zmanjšanju izpustov CO_2, kar je pripeljalo do številnih raziskav in iskanja alternativnih algoritmov soglasja. Danes imamo kar nekaj predlaganih rešitev, v množični uporabi pa so le redke. V nadaljnih poglavjih bomo opisali nekatere predlagane algoritme in jih primerjali.

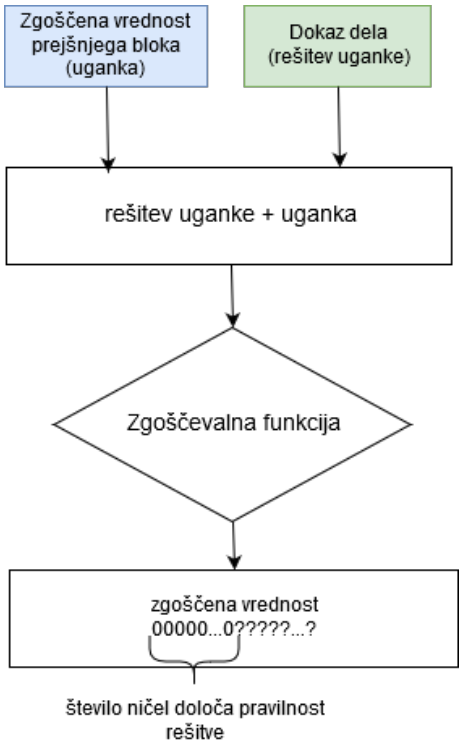
3.1 Dokaz dela

Dokaz dela v osnovi pomeni uporabo procesorskih ciklov za potrditev dobronamernosti zahtevka. Uporabnik z uporabo procesorskih ciklov reši uganko, s čimer dokaže svojo dobronamernost. Dokaz dela izhaja iz članka HashCash [2], ki predlaga dokaz dela za zajezitev napadov zavrnitve storitve (angl. denial of service) in omejevanja pošiljanja neželene e-pošte. Adam Back je leta 1997 predlagal, da bi ob vsakem poslanem e-poštnem sporočilu pošiljatelj rešil uganko z nekaj cikli centralno-procesne enote. Pri velikih količinah poslanih sporočil bi tako uporabniki potrebovali veliko procesorske moči, kar bi podražilo napade in s tem vsaj zmanjšalo njihovo pogostost. Iskanje rešitve uganke na področju verige blokov, predvsem pa na področju kriptovalut, imenujemo rudarjenje.

Dokaze dela v grobem delimo na dva tipa:

- **uganka-odgovor** - Pri tem tipu dokaza dela stranka prejme uganko od omrežja glede na njegovo povpraševanje po storitvi. Rešitev stranka pošlje nazaj omrežju, ki rešitev potrdi ali zavrne in glede na to ponudi dostop do servisa.

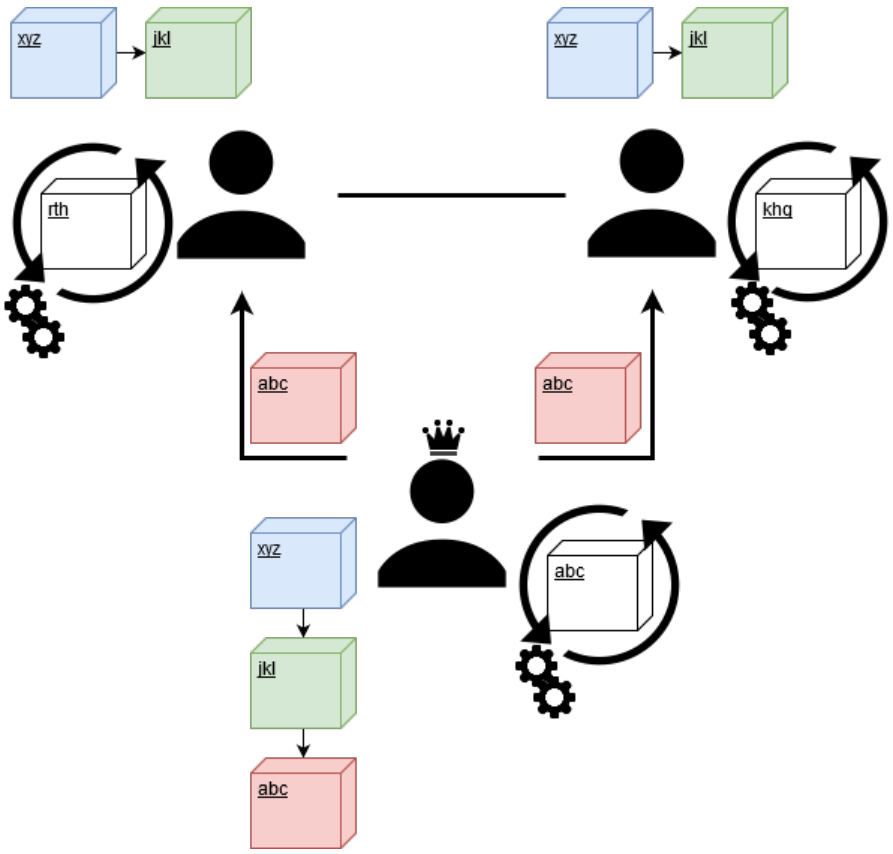
- **rešitev-potrditev** - Pri tipu rešitev-potrditev si pošiljatelj sam zastavi uganko glede na podatke, ki jih hoče poslati prejemniku. Prejemnik, ki



Slika 3.1: Postopek iskanja rešitve uganke.

rešitev sprejme, jo preveri in podatke sprejme v primeru, da se sporočilo in rešitev skladata s protokolom. Ta tip dokaza dela je primernejši za arhitekturo, kjer ni centralne avtoritete in so vsi deležniki v omrežju enakovredni.

Prvi dokaz dela (angl. proof of work) z uporabo v verigi blokov je bil opisan leta 2009 v delu Bitcoin: A Peer-to-Peer Electronic Cash System [1] in je bil prirejen iz dela HashCash [2]. Algoritem soglasja, ki deluje na načelu dokaza dela in je implementiran v verigi blokov Bitcoin, skrbi, da vozlišča vzamejo zgoščeno vrednost prejšnjega bloka, ki ji pripnejo poljuben niz, ki je naključno izbrana rešitev s strani rudarja. Tem vrednostim so nato pripete še transakcije, ki jih želi rudar potrditi. Pred transakcije rudar še vrine transakcijo, ki mu doda na račun nagrado za pravilno potrjeni blok. Dobljeni niz rudar vstavi kot vhoden podatek v zgoščevalno funkcijo. Izhodni niz

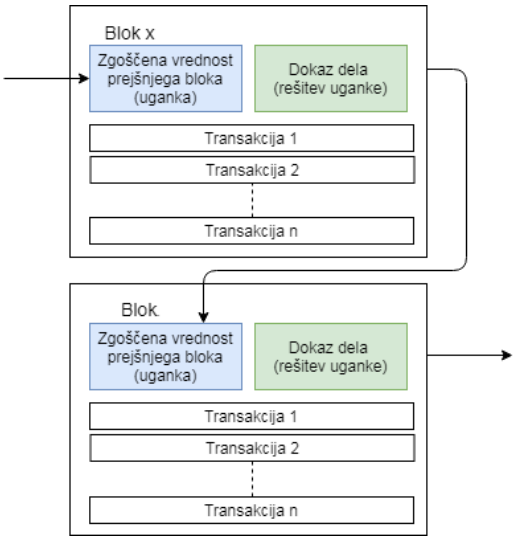


Slika 3.2: Diagram delovanja dokaza dela.

zgoščevalne funkcije mora ustrezati pogojem, s čimer rudar potrdi pravilnost rešitve (Slika 3.1).

V verigi blokov Bitcoin mora biti zgoščena vrednost v dvojiški obliki sestavljena iz določenega števila začetnih bitov ničel. Potrebno število ničel se v algoritmu soglasja v verigi blokov Bitcoin samodejno prilagaja glede na količino potrjenih blokov v prejšnji uri. Zaradi navezave na prejšnje bloke je za kasnejšo spremembo bloka potrebno ponovno preračunavanje vseh blokov od spremenjenega bloka do končnega, kar pa je praktično nemogoče v primeru velikih omrežij.

Če rudar dobi rešitev uganke, le-to vstavi v blok s transakcijami in pošlje rešitev v potrditev drugim vozliščem (rudarjem), ki preverijo veljav-



Slika 3.3: Vezava blokov v verigi blokov.

nost rešitve in blok dodajo v svoj zapis verige, kot prikazuje slika 3.2.

Blok vsebuje zgoščeno vrednost prejšnjega bloka, ki deluje kot referenca, kot prikazuje slika 3.3. V primeru več rešitev oziroma napačnih rešitev se lahko veriga razdeli, saj imajo v omrežju vozlišča različne verige. Čez nekaj blokov se tako privzame za pravo tisto verigo, ki je daljša. V primeru napada z dodajanjem napačnega bloka v verigo bi namreč le nepravična vozlišča nadaljevala z dodajanjem blokov na nepravilno verigo. Če je nepravičnih vozlišč manj kot 50 odstotkov, njihova veriga blokov ne bo najdaljša in bo posledično zavržena. Dokaz dela je bil tako prvi algoritem soglasja, ki je pomembno oblikoval miselnost in dokazal, da je uporaba decentralizirane verige blokov mogoča. Varnost je dokaz dela zagotovil z zahtevo po iskanju pravega niza za zgoščevalno funkcijo. Zahtevnost iskanja dokaza se sicer prilagaja trenutnim razmeram med rudarji in preteklimi bloki, a število procesorskih ciklov za ugotovitev pravilne zgoščene vrednosti je ogromno. Plačilo rudarjev za potrjevanje naslednjega bloka je sestavljeno iz dveh delov. Prvi del je količina kovancev, generirana ob potrditvi novega bloka. Potrjevalec bloka namreč pred vse transakcije vrine zapis, ki na njegovo denarnico doda vrednost tre-

nutne nagrade. Prvi del oziroma nagrada tako pripada rudarju, ki je odkril pravilno rešitev. Drugi del sestavlja vrednost, ki je bila nabrana iz provizij transakcij, ta se razdeli med vse rudarje. Število generiranih kovancev se manjša, kar posledično pomeni manjšo dobičkonosnost za rudarje oziroma povečanje provizij transakcij. Zaradi potrebe po velikih količinah električne energije bodo rudarji primorani zaslužiti več ali se odločiti za prenehanje rudarjenja. Zaradi upada števila rudarjev se zmanjša varnost. Zaradi takega predvidevanja so se avtorji PeerCoina [6] odločili za iskanje mehanizma soglasja, ki ne bi zavaroval podatkov na verigi blokov z električno energijo, ampak z nečim cenejšim. Pomembna želja bi bila tudi pohitritev potrjevanja transakcij, kar je bil tudi problem Nakamotove implementacije dokaza dela v verigi blkov Bitcoin. Dokaz dela bi se sicer lahko spremenilo tako, da bi se blok generiral pogosteje, a s tem bi se zmanjšala zahtevnost uganke oziroma število potrebnih ničel v bitni reprezentaciji zgoščene vrednosti bloka. Vsaka dodatna ničla v preizkusu pomeni kvadratno rast števila povprečnih potrebnih poskusov za rešitev uganke, le-to pomeni tudi kvadratno rast varnosti. Z bloki, potrjenimi pogosteje, bi mehanizmu dokaza dela bistveno zmanjšali varnost.

3.2 Dokaz deleža

Ideja dokaza deleža (angl. proof of stake) se je prvič pojavila na forumih v letu 2011. Leto kasneje sta prvo implementacijo tega algoritma uporabila King in Nadal v Peercoinu [6].

V splošnem dokaz deleža pomeni, da naslednji blok transakcij predlaga vozlišče, ki je izbrano glede na količino kovancev, ki si jih lasti.

V dokazu deleža sta avtorja Peercoina uporabila tudi starost kovancev (angl. coin age). Pomembnost starosti kovancev je uporabil že Nakamoto v verigi blokov Bitcoin leta 2010 in je bila eden izmed parametrov za določanje prioritete transakcij, ni pa nikakor vplivala na varnost verige blokov. Starost kovanca je definirana kot produkt števila kovancev in dolžina časa, odkar

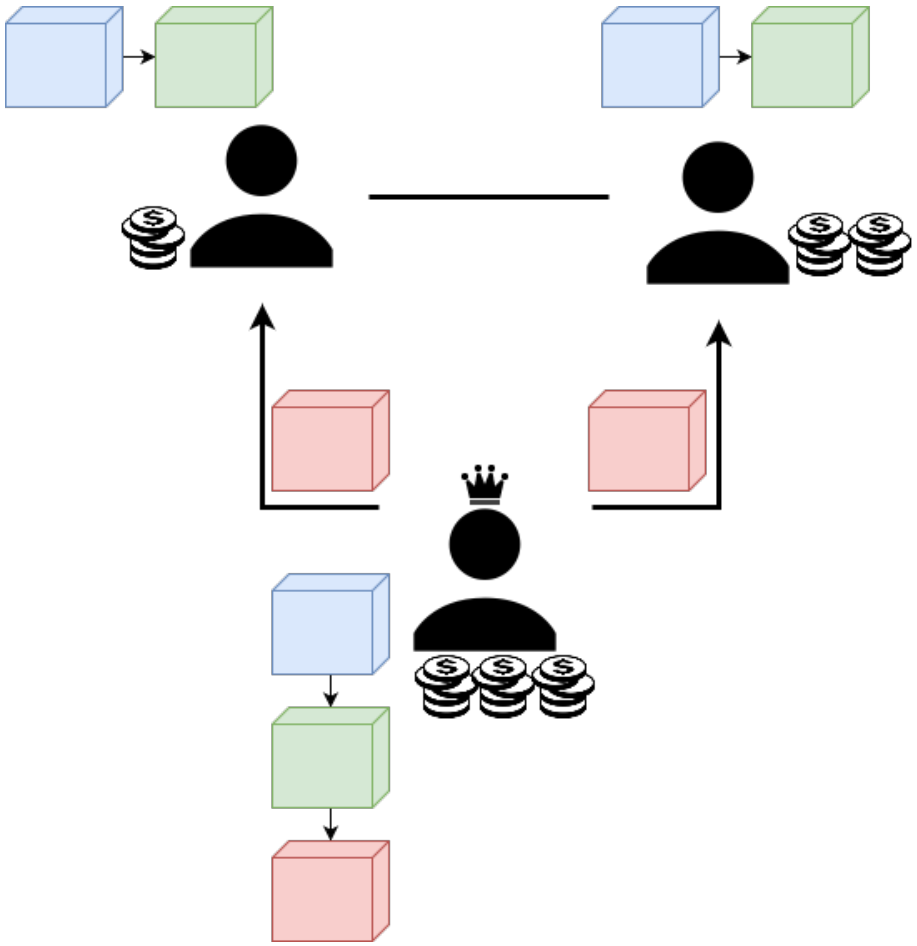
so bili nazadnje premaknjeni. Posledica uporabe starosti kovanca je tudi časovni žig v vsako transakcijo. Dodatno varnost, da ne bi prihajalo do zlorab, predstavlja še časovni žig na vsakem bloku. Delovanje dokaza deleža prikazuje slika 3.4.

Dokaz deleža v nasprotju z dokazom dela ne podeljuje nagrade za potrjen, blok, torej se pri procesu ne ustvarijo novi kovanci in ne prihaja do inflacije. Potreba po nagradi ob potrditvi novega bloka ni potrebna, ker se deterministično določi rudarja, ki bo potrdil naslednji blok. Določitev rudarja se izvede glede na količino njegovih kovancev oziroma delež in starost kovancev. Rudarji so tako plačani le iz provizij transakcij. Zaradi bistveno manjše porabe energije pri potrjevanju bloka transakcij je razmerje med ceno električne energije in ceno kovancev veliko učinkovitejše za rudarje. Nekatere kriptovalute (npr. Ethereum) so tako že spremenile način potrjevanja blokov iz dokaza dela na dokaz deleža, pričakuje pa se, da bo takih sprememb še več.

Za področje varnosti obstajajo teoretične možnosti napada, a so zaradi hitre odzivnosti trga take možnosti minimalne. Za napad bi moral storilec imeti 51 % kovancev, ker bi moral napadalec pri nakupu vložiti velike količine denarja, nato pa bi ob napadu cena kovancev drastično padla, kar bi pomenilo največjo izgubo ravno za storilca. Blok je tako zavarovan z deležem kovancev potrjevalca. V splošnem lahko govorimo, da je zaradi drugačnega načina soglasja treba paziti predvsem na dva nova načina napadov.

Grinding napad (angl. Grinding attack) imenujemo napad, kjer poskušajo nepoštena vozlišča spremeniti izid volitev za potrjevalca bloka. S tem bi si lahko prisvojili več nagrade iz provizij transakcij ali pa bi jim uspelo večkrat porabiti (angl. double spending) iste kovance.

Drugi možni napad, pa se imenuje napad brez tveganja (angl. Nothing as stake attack), pri katerem napadalec poskuša ustvariti nove veje v verigi blokov. V algoritmih dokaza dela, je bilo to onemogočeno, saj morajo vozlišča porabljati procesorski čas za izgradnjo novega bloka, zato je za vozlišča najbolj smiselno, da gradijo le vejo,ki je poštena in se bo obdržala. V dokazu



Slika 3.4: Diagram delovanja dokaza deleža.

deleža pa s sodelovanjem na vseh vejah vozlišče ne izgubi ničesar.

Z metodo dokaza deleža je bilo implementiranih več algoritmov. Najbolj znani so PIVX, Blackcoin, Ouroboros in CASPER. Metodo dokaza deleža je implementiral tudi že omenjeni PeerCoin, a je hkrati obdržal tudi dokaz dela. Algoritem deluje tako, da je težavnost dokaza dela prilagojena faktorju izračunanemu iz deleža kovancev, ki jih ima vozlišče v lasti, in dolžino lastništva teh kovancev.

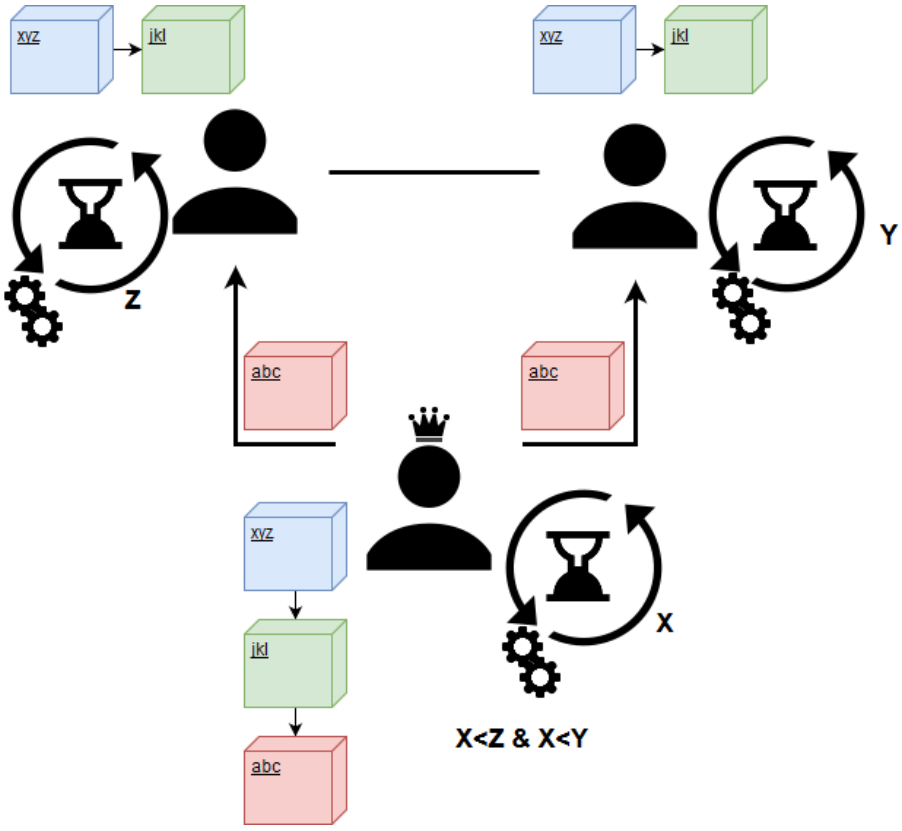
3.3 Dokaz pretečenega časa

Dokaz pretečenega časa (angl. proof of elapsed time) je mehanizem soglasja, ki ga je predlagalo podjetje Intel. Namen je izdelati mehanizem soglasja, kjer določimo vodjo soglasja. Določitev vodje temelji na sistemu loterije. Pomembne lastnosti, ki jih želi mehanizem pokriti, so:

- **pravičnost (angl. fairness)** - Mehanizem mora porazdeljeno loterijo izvajati na čim večjem številu ustreznih vozlišč.
- **vložek (angl. investment)** - Cena postopka izvolitve vodje mora biti sorazmerna z vrednostjo, ki jo varuje.
- **preverljivost (angl. verification)** - Preverjanje, da je izbrani vodja pravilno izbran, mora biti čim bolj preprosto.

Dokaz pretečenega časa je načrtovan na temelju novega varnega ukaza za centralno procesno enoto. Dokaz pretečenega časa tako uporablja ukaz centralno procesne enote za zagotavljanje varnosti in naključnosti procesa izbora vodje, ne da bi za ta namen porabili veliko energije ali posebno strojno opremo [7].

Rudar se pridruži porazdeljenemu omrežju tako, da zažene veljavno programsko kodo verige blokov. Koda v tem trenutku ustvari par javnega in zasebnega ključa. Z ukazom centralno procesne enote nato potrdi zahtevo za pridružitev, ki vsebuje tudi javni ključ in jo pošlje drugim članom v omrežju.



Slika 3.5: Diagram delovanja dokaza pretečenega časa.

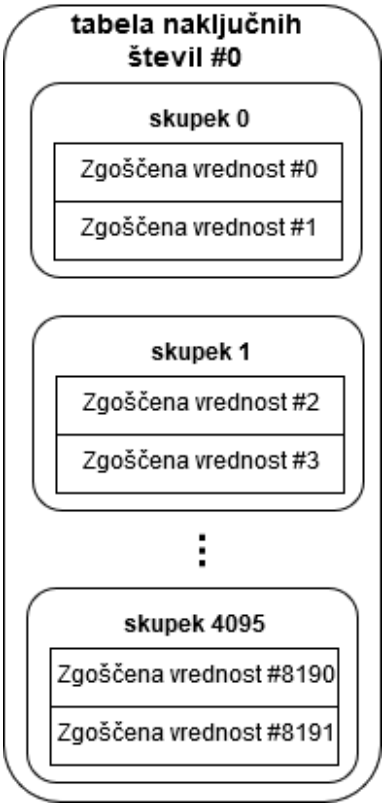
Pri dokazu pretečenega dela najprej vsak izmed rudarjev izvede varen centralno procesni ukaz SGX (angl. Software Guard Extensions). Ta mu vrne naključni čas čakanja. Rudar nato počaka ta naključno določen čas, po preteku pa omrežju pošlje potrdilo o opravljenem čakanju in nov blok, ki ga hoče priključiti verigi blokov. Rudar, ki je pridobil najnižjo vrednostjo čakanja, tako postane vodja soglasja. Drugi potrjevalci v omrežju preverijo integriteto uporabnika, prejet blok in preverijo ali je bil naključno izbrani čas res naključen in ali je bilo čakanje izvedeno. Delovanje prikazuje slika 3.5.

Algoritem za izbor vodje v dokazu pretečenega časa je dober naključni algoritem. Naključnost je enakomerno porazdeljena na celotno število potrjevalcev, a hkrati je verjetnost izvolitve odvisna od količine virov, ki jih potrjevalec nudi. V primeru dokaza pretečenega časa to pomeni število centralno procesorskih enot, ki so na voljo. Zaradi cenovne ugodnosti mehanizma soglasja se pričakuje veliko število sodelujočih potrjevalcev, kar zagotavlja tudi večjo porazdeljenost in s tem večjo varnost. Mehanizem soglasja dokaza pretečenega časa je že implementiran v projektu Hyperledger Sawtooth, na katerem še teče razvoj.

Pomembno se nam zdi opozoriti, da imajo možnost ukaza SGX le procesorji proizvajalca Intel, kar lahko predstavlja prihodnjo težavo za varnost. Gre za zaščiteno idejo podjetja Intel, kar pomeni, da bi za uporabo ideje drugi proizvajalci potrebovali dovoljenje podjetja za proizvodnjo procesorjev z enakimi ukazi. Za druge procesorje trenutno obstaja zgolj knjižnica OpenSGX [18], ki za namene razvoja lahko nadomešča pravi procesor, a je za produkcijska okolja neprimerna.

3.4 Dokaz prostora

Dokaz prostora so prvič opisali v delu Proofs of space [5]. Avtorji so opažali, da ima vsaka naprava vsaj nekaj prostega diskovnega prostora, ki bi lahko bil primeren za mehanizem soglasja, ki bi temeljil na praznih spremenljivih nosilcih podatkov. Branje in zapisovanje na diskovni prostor v primerjavi

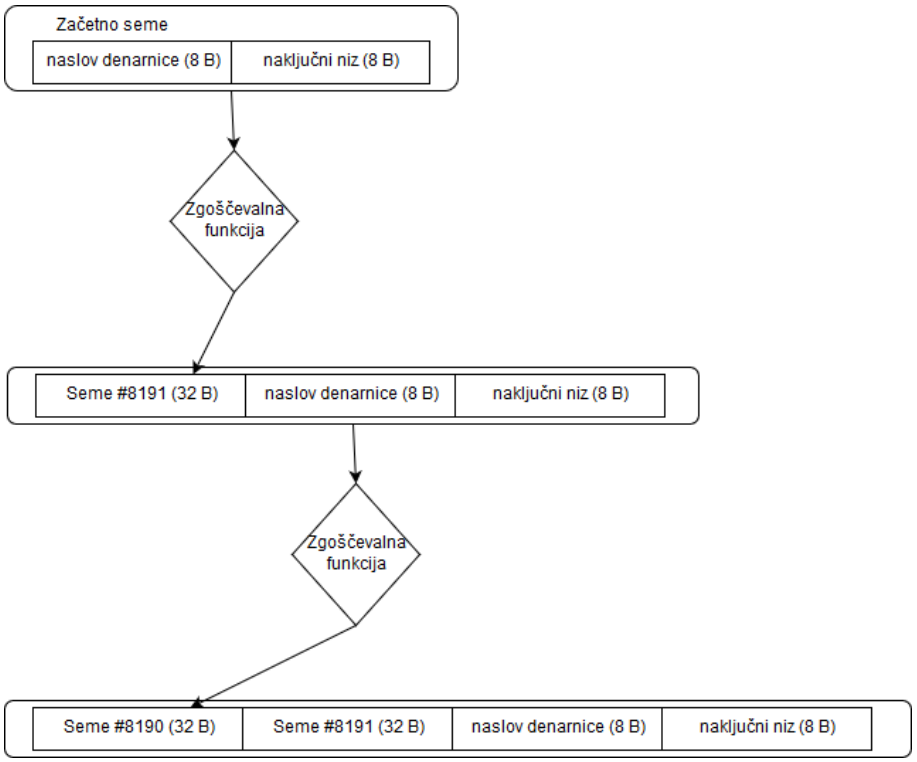


Slika 3.6: Struktura neoptimizirane tabele z naključnimi števili.

z dokazom dela ne porabi, enormnih količin energije in ne potrebujeta namenske strojne opreme. Tudi velikost prostora ni najpomembnejši dejavnik. Prva implementacija v uporabi je zaživela s kriptovaluto Burstcoin.

Algoritem deluje podobno kot dokaz dela. Pri dokazu dela vozlišča v realnem času iščejo zgoščeno rešitev, za vsak blok posebej. Pri dokazu prostora rudar vnaprej preračuna določene zgoščene vrednosti in jih shranjuje na disk. Več kot ima prostora na nosilcih, več možnosti ima, da bo predlagal naslednji blok.

Rudar najprej izvede proces načrtovanja (angl. plotting). V procesu ustvari datoteke, ki vsebujejo večjo količino tabel z naključnimi števili (angl. nonce table). Število tabel z naključnimi vrednostmi je odvisno od količine prostora, ki ga je namenil za rudarjenje. Tabela z naključnimi vrednostmi

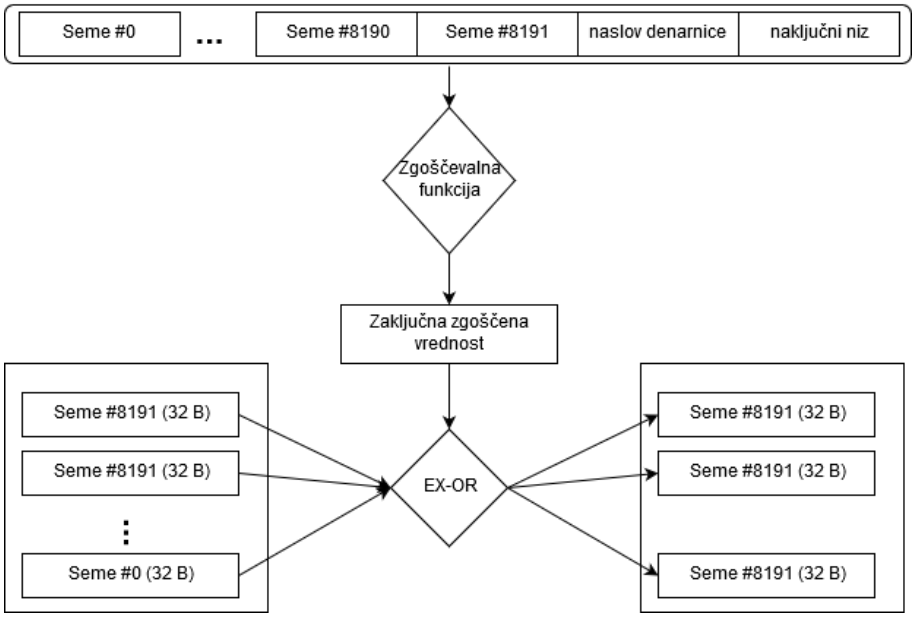


Slika 3.7: Generiranje semen.

sestoji iz 8192 zgoščenih vrednosti, ki zasedejo 256 KB prostora. Posamezna zgoščena vrednost zasede 32 B prostora. V tabeli z naključnimi vrednostmi sta združeni po dve zgoščeni vrednosti v zaporedje skupkov (angl. scoops) parov, kot prikazuje slika 3.6. Proces načrtovanja vzame za vhodni podatek naslov denarnice rudarja in tako doseže unikatnost načrtovanih datotek za vsakega rudarja. Zgoščene vrednosti se preračunajo po naslednjem postopku, opisanem v dokumentaciji [19]:

- Najprej je preračunana zgoščena vrednost za polje 8192, tako da se unikatni naslov denarnice združi z naključnim 8-bajtnim nizom. Iz sestavljenega niza se nato preračuna zgoščena vrednost dolžine 32 bajtov, ki je seme za končno zgoščeno vrednost polja 8192.

- Seme polja 8191 se nato predpne naslovu denarnice in naključnega niza,



Slika 3.8: EX-OR-operacija nad semeni in zgoščeno funkcijo vseh semen.

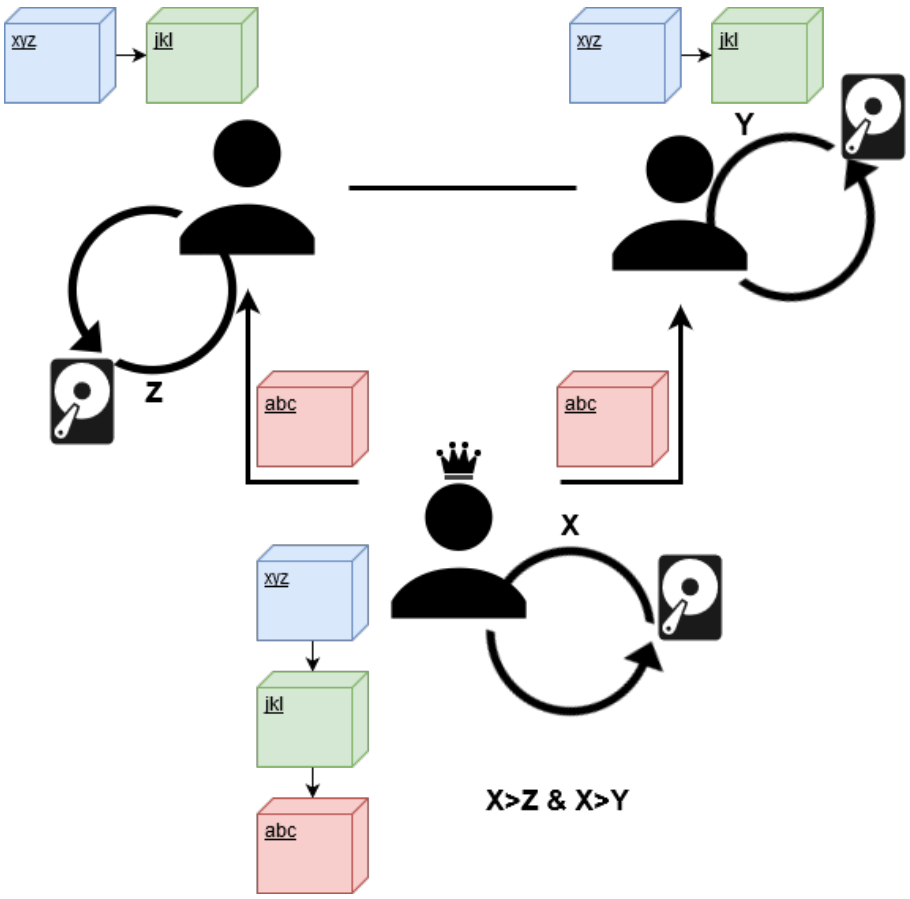
nakar ponovno izvedemo zgoščevanje vrednosti. S tem pridobimo seme za polje 8190.

- Postopek nadaljujemo pri računanju zgoščene prednosti niza, ki ga sestavljajo seme polja 8190, 8191, naslov denarnice in naključni niz, tako kot prikazuje slika 3.7. Tako pridobimo seme za polje 8189.

- Računanje semen izvajamo po opisanem postopku za prvih 128 iteracij. Po 128. iteraciji vrednost vhoda v zgoščevalno funkcijo prilagodimo tako, da je vhodni niz vedno dolžine 4096 bajtov, sestavljen iz niza nazadnje generiranih semen.

- Ko izračunamo semena za vsa polja, sestavimo niz vseh semen in začetno seme. V naslednjem koraku pridobimo zgoščeno vrednost sestavljenega niza.

- Na vsakem semenu sedaj izvedemo ex-or operacijo z zgoščeno vrednostjo vseh semen, dobljeno v prejšnjem koraku. Vsaka pridobljena vre-



Slika 3.9: Diagram delovanja dokaza prostora.

dnost je končna vrednost posameznega polja v tabeli, kot prikazuje slika 3.8.

Načrtovanje lahko obsega še optimizacijo tabel z naključnimi vrednostmi, in sicer tako, da algoritem združi enakoležna polja skupkov vseh tabel z naključnimi vrednostmi in tako dostopa do polja hitreje.

Ob zaključenem načrtovanju se rudar vključi v omrežje in začne tekmovanje za potrditev bloka. Rudar najprej vpraša za zgoščeno vrednost zadnjega bloka, vrednost osnovnega cilja in zaporedno številko naslednjega bloka. Vrednost osnovnega cilja (angl. base target) je vrednost, ki je izračunana iz zadnjih 24 blokov in nastavlja pogostost novega bloka. Zgoščeno vrednost sedaj

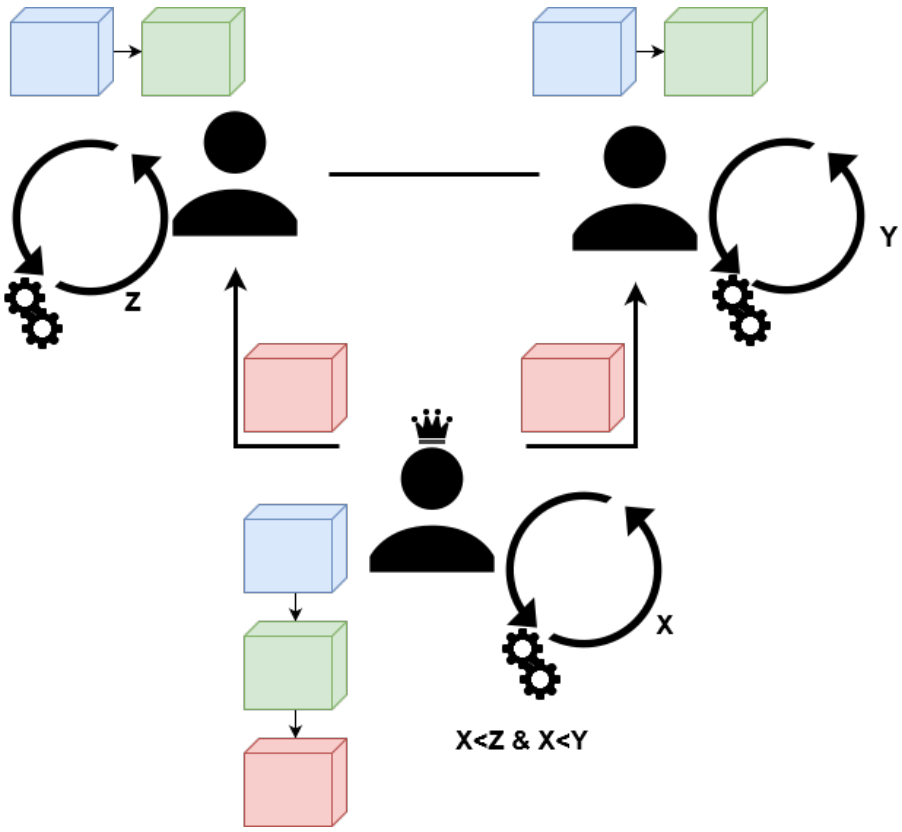
rudar spne z zaporedno številko naslednjega bloka in jo vstavi v zgoščevalno funkcijo. Dobljeno vrednost imenujemo generalna zgoščena vrednost. Vozlišče generalno zgoščeno vrednost deli po modulu 4096 in izračuna ostanek deljenja. Ostanek deljenja določa, v katerem polju skupkov išče zgoščeno vrednost. Zgoščene vrednosti polja skupka sedaj združi z generalno zgoščeno vrednostjo in jo vstavi v zgoščevalno funkcijo. To stori za vsako polje skupka, ki ustreza v vseh tabelah z naključnimi vrednostmi. Dobljene vrednosti, imenovane ciljne vrednosti, algoritem deli z vrednostjo osnovnega cilja. Pridobljeni rezultat je čas, ki mora preteči, preden lahko vozlišče kandidira za pridobitev funkcije potrjevalca. Postopek prikazuje slika 3.9. Cilj vozlišča je, da pridobi čim več vrednosti iz svojih tabel. Vrednosti pošlje v omrežje, v primeru, da je vrednost najnižja v omrežju, se mu dodeli oblikovanje naslednjega bloka. Algoritem ima tudi varnostni mehanizem, ki poskrbi, da je določena največja vrednost čakanja na omrežju oziroma, da imajo rudarji znano trenutno najnižjo vrednost, saj v primeru, da rudar izračuna vrednost, ki je višja od trenutno najnižje, svoje informacije sploh ne pošlje v omrežje.

3.5 Dokaz sreče

Dokaz sreče so opisali avtorji članka Proof of Luck: an Efficient Blockchain Consensus protocol [11]. Delo temelji na isti ideji kot dokaz pretečenega časa, torej na zaprtih izvajalnih okoljih, kot je SGX-ukaz na centralno-procesnih enotah. S tem želijo avtorji poskrbeti za večjo porazdeljenost rudarjenja in se izogniti potrebi po kupovanju namenske strojne opreme za rudarjenje. V izogib energijski potratnosti in časovni kompleksnosti pa uporabijo še dokaz časa in dokaz lastništva. Avtorji ohranjajo tudi dokaz dela kot enega izmed osnovnih gradnikov, ki potrdi blok, vendar v tem primeru ni potrebe po tako veliki zahtevnosti, saj je rudar, ki potrjuje naslednji blok, določen z drugimi načeli in ne s tekmovanjem. Vsi ti prijemi skupaj sestavljajo algoritem dokaza sreče.

Avtorji uporabijo idejo, povzeto po treh gradnikih, dokazu dela, dokazu

pretečenega časa in dokazu lastništva, ki se izvajajo v zaprtem izvajalnem okolju. Dokaz dela nam izračuna zgoščene vrednosti, dokaz časa nam s funkcijo SGX na procesorjih zagotovi, da so rudarji izbrali naključni čas čakanja in ta čas tudi čakali, dokaz lastništva pa nam zagotovi enkratno izvedbo algoritma na enem fizičnem procesorju v vsaki iteraciji potrjevanja bloka. Najprej vzamemo idejo dokaza dela. Algoritem dokaza dela lahko vstavimo v zaprto izvajalno okolje. Kot vhodne podatke mu podamo težavnost in naključni niz, s čimer preprečimo, da bi enkrat dobljeni dokaz še kdaj uporabili. Izhodni podatek je dokaz opravljenega dela. Ideja kot taka preprečuje uporabo namenskih strojnih komponent, zaradi dostopa do izvajalnega okolja. Tako pripomore k večji porazdeljenosti omrežja. Pri tej ideji potrjevanje rešitev ni potrebna, saj je dovolj potrditev, da je dokaz pridobljen v zaprtem izvajalnem okolju. Težava ideje je, da še vedno porabi velike količine električne energije. Z nekaj razmisleka lahko ugotovimo, da je dokaz namenjen samo zaščiti, delo, ki so ga izvajali procesorski cikli, pa je nesmiselno. S to idejo poskušamo procesorske cikle uporabiti bolj smiselno, in sicer z uporabo funkcije SGX. Dokazati želimo, da smo počakali določen čas. V splošnem bi lahko rudarji ponaredili čas, a ker čakamo v zaprtem izvajalnem okolju, lahko trdimo, da je procesor, ki je vrnil potrditev čakanja, ta čas tudi počakal. Med čakanjem lahko s pametno implementacijo algoritma procesorski čas pametno uporabimo. Za preprečitev ponovne uporabe rezultata čakanja, v delu algoritma, ki se izvaja v zaprtem izvajalnem okolju, implementiramo števec, ki se ob vsaki izvedbi poveča. Ideja dokaza časa se zdi energetsko učinkovita, ampak vprašamo se zakaj čakamo čas. Ob širšem pogledu ugotovimo, da s tem le dokazujemo lastništvo nad procesorjem, ki je tega sposoben. Zakaj bi torej transakcije pri potrjevanju čakale nekaj minut, če so lahko potrjene v nekaj sekundah? Tako dobimo algoritem dokaza lastništva. V osnovi algoritem predstavlja, da vsak rudar s svojim procesorjem izvede funkcijo v zaprtem izvajalnem okolju, ki nam potrjuje lastništvo procesorja, rudar pa s tem pridobi srečko za žreb rudarja, ki bo potrjeval naslednji blok. Težava dokaza lastništva je v njegovi razširljivosti. Vsa vozlišča morajo namreč pri-



Slika 3.10: Diagram delovanja dokaza sreče.

dobiti pripravljenost rudarjenja vseh drugih v omrežju. Dokaz sreče je v osnovi ideja, ki nadgrajuje prej omenjene algoritme. V osnovi vsako vozlišče ustvari naključno število, tako imenovani koeficient sreče. Tisto vozlišče, ki ima višji koeficient, ima možnost potrjevanja novega bloka. Algoritem se začne s čakanjem vnaprej določenega časa. V tem času poteka sinhronizacija med vozlišči. Vozlišča prejmejo zadnji potrjeni blok in uskladijo svojo verigo blokov. Po pretečenem čakanju algoritem izbere naključni koeficient sreče na intervalu med 0 in 1. Sledi čakanje, v razmerju 1/koeficient sreče. Gre za optimizacijski korak, saj vozlišča, ki imajo višji koeficient sreče, prej nadaljujejo z izvajanjem. V naslednjem koraku vozlišče potrdi blok in ga s svojim koeficientom sreče pošlje v omrežje kot prikazuje slika 3.10. Vozlišča, ki so čakala dlje, so tako že dobila blok, z višjim koeficientom sreče in tako

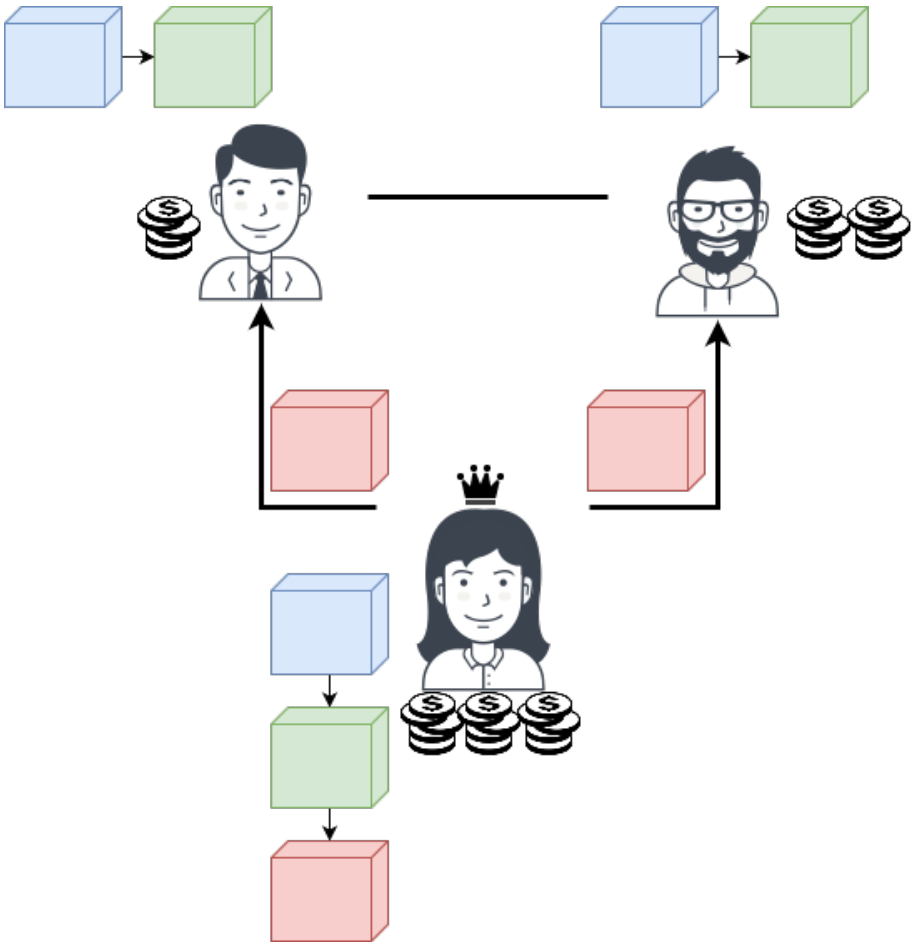
svoje potrditve ne pošljejo. S to optimizacijo tako poskrbimo, da se pri večji količini vozlišč komunikacija med njimi ne poveča drastično.

Blok vsebuje transakcije, zgoščeno vrednost prejšnjega bloka in koeficient sreče rudarja. V primeru, da se veriga razdeli na dve veji, zmaga tista, ki ima najvišji skupni koeficient sreče. Po prejetju bloka, ki je kandidat za naslednji blok, vsako vozlišče izvede še nekaj validacij, kot so npr. preračun vse transakcij od začetnega bloka do predlaganega za preprečitev dvojnega porabljanja, validacija resničnega naključnega izbora čakalnega časa potrjevalca in resničnost čakanja, katerega posledica je odpornost proti napadu z manj kot 50 % vozlišč.

3.6 Dokaz oblasti

Dokaz oblasti (angl. proof of authority) je nastal iz želje po zmanjšanju porabe energije dokaza dela in spoznanja, da dokaz deleža ne zadostuje za varovanje verige blokov pred manipulacijami. Pri opazovanju algoritma dokaza deleža so raziskovalci namreč opazili, da nekomu, z velikim deležem premoženja, shranjenega v določeni verigi blokov, lahko to premoženje pomeni le nekaj odstotkov vsega njegovega premoženja, medtem ko drugemu, ki ima enako ali manj premoženja v verigi blokov, to premoženje pomeni večino njegovega premoženja. V tem primeru bo prva oseba prej pripravljena na poskus manipulacije kot druga in bi ji zato morali manj zaupati potrditev bloka. Nezmožnost dokaza deleža, da bi upošteval take težave, je pripeljala do predloga za dokaz oblasti.

Dokaz oblasti uporablja dejstvo, da ima ena oseba natančno eno identiteto. Dokaz oblasti zahteva, da nekdo s pravico potrjevanja blokov svojo identiteto razkrije, kot prikazuje 3.11. Tako potrjevalec ni le anonimna oseba v omrežju, ampak človek, ki za svojimi dejanji stoji in odgovarja zanje. Hkrati lahko to osebo tudi preverimo, da ni zagrešila preteklih kriminalnih dejanj in je zaupanja vreden kandidat za potrditev blokov. Za potrditev osebe kot zaupanja vredno je treba vzpostaviti sistem potrditve, ki temelji na treh



Slika 3.11: Diagram delovanja dokaza oblasti.

pogojih:

- **Identiteta mora obstajati** - Identiteto osebe mora nekdo preveriti. Pri POA Network verigi blokov za ta del predlagajo notarje, ki že imajo podobne vloge v institucijah.

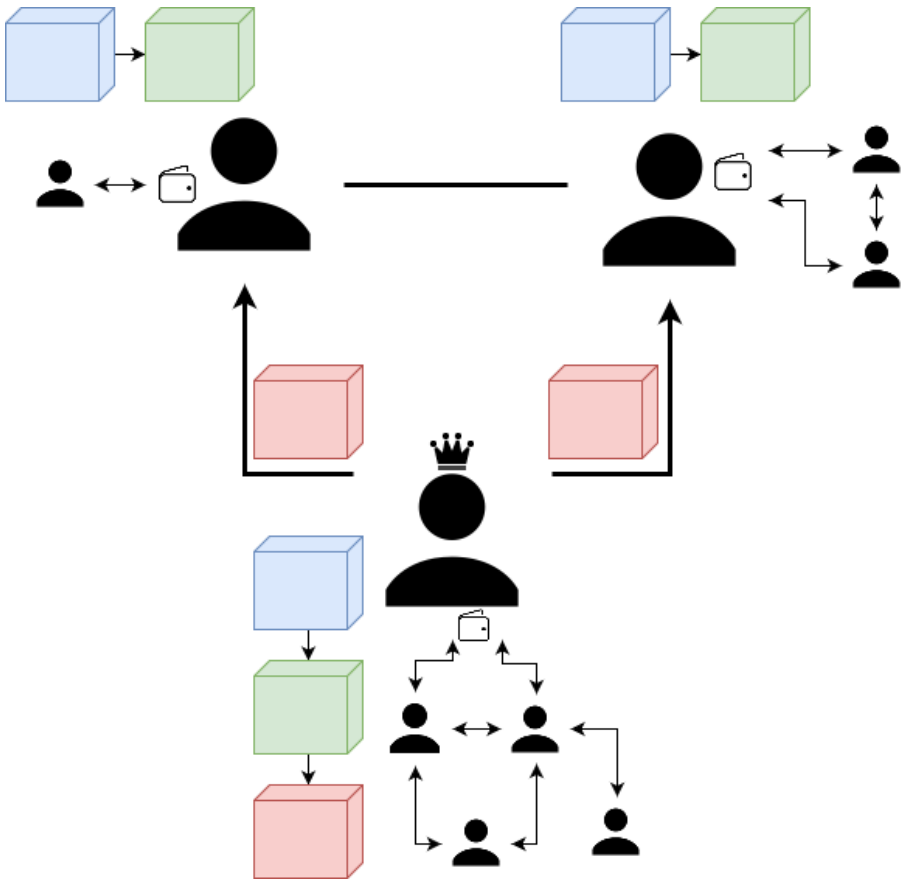
- **Pooblastila morajo biti zaslužena** - Oseba si mora privilegij potrjevanja blokov zaslužiti, izguba statusa pa je neprijetna. Na to lahko vplivajo razni testi s strani notarja, kot je nekaznovanost ali vloženi čas v izdelavo projekta (npr. nudenje uporabniške podpore). V primeru izvedbe napada bi oseba lahko za svoja dejanja odgovarjala tudi na sodišču.

- **Sistem za pridobitev pooblastil mora biti za vse enak** - Sistem pridobitve je za vse enak, saj ga s stališča notarja določa zakon za potrjevanje identitet, medtem ko s strani verige blokov za to poskrbi sama koda v verigi blokov.

Identifikacija in pooblaščeni potrjevalci blokov so mehanizmi, ki verige blokov z algoritmom dokaza oblasti uvrščajo med zasebne ali vsaj hibridne verige blokov. Potrjevalci se krožno izmenjujejo pri predlaganju bloka. Ekipa verige blokov Ethereum je predlagala dve različni implementaciji po imenu Aura in Clique za potrebe zasebnih verig blokov [9]. Implementacija Aura se uporablja v verigi blokov, imenovani Kovan, medtem ko se Clique uporablja v verigi, imenovani Rinkeby. Obe verigi blokov sta še v razvoju.

3.7 Dokaz lokacije

Dokaz lokacije (angl. proof of location) je izpeljanka mehanizma soglasja, ki se uporablja predvsem na področju interneta stvari (IOT). Dokaz lokacije deluje po načelu dokaza dela ali dokaza deleža, vendar v svojo validacijo transakcij vključuje tudi preverjanje, ali so vozlišča omrežja na tistih koordinatah, ki so jih so javila. Vsako vozlišče v omrežju verige blokov je IOT-naprava, ki



Slika 3.12: Diagram delovanja dokaza pomembnosti.

pošlje transakcijo po internetu vsem vozliščem. Po bluetooth povezavi oziroma kateri izmed drugih povezav s kratkim obsegom pa transakcijo pošlje tudi bližnjim napravam. Te nato primerjajo svoje GPS-koordinate in koordinate v transakciji ter ugotovijo, ali je naprava res v njihovi bližini [20].

3.8 Dokaz pomembnosti

Dokaz pomembnosti (angl. proof of importance) sledi ideji dokaza deleža. Mehanizem soglasja določi naslednjega potrjevalca bloka z žrebom. Vozlišča, ki so v sistemu bolj pomembna, imajo več možnosti, da so izbrana in s tem lahko potrjujejo naslednji blok kot prikazuje slika 3.12.

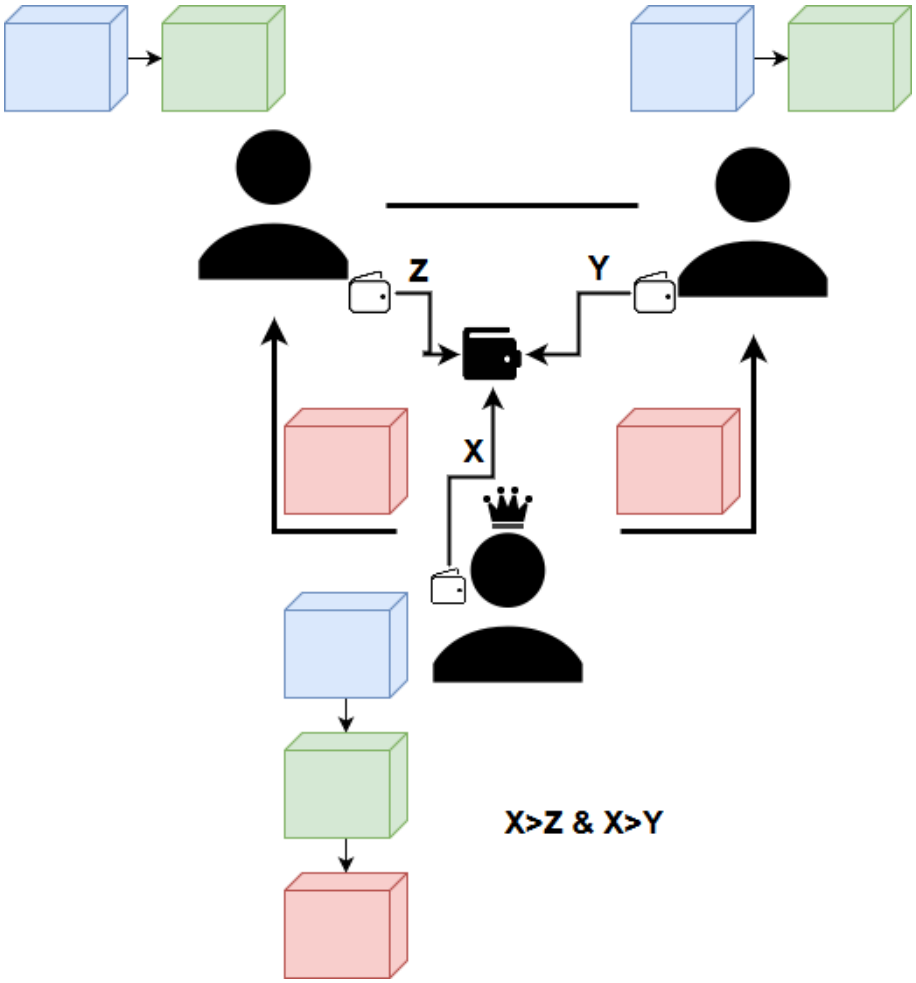
Vsako vozlišče v verigi blokov ima svoj naslov, ki hrani kovance. Glede na transparentnost verige blokov imamo tako možnost s pomočjo teorije grafov preračunati, kako pomembno je vozlišče v grafu. Naslovi denarnic so vozlišča, transakcije med njimi pa povezave. Pri tem upoštevamo število kovancev, vrednost dosedanjih transakcij in pomembnost vozlišč, s katerih so bili kovanci nakazani [10]. Pomembnost vozlišča nam torej vzpostavlja zaupanje v vozlišče. Dokaz pomembnosti za mehanizem soglasja uporablja kriptovaluta NEM.

3.9 Dokaz požiga

Dokaz požiga (angl. proof of burn) je mehanizem soglasja, ki ga je prvi implementiral Slimcoin [8]. Ideja mehanizma soglasja je potrjevanje blokov, ki je hitro in neodvisno od strojne opreme oziroma čim cenejše z vidika realnih vrednosti. Ideja predpostavlja, da že imamo žetone ali kovance, ki so nosilci vrednosti oziroma je za njihovo rudarjenje že bil porabljen čas, energija ali strojna oprema. Običajno se ideja povezuje z bitcoin kovanci, a bi bila lahko implementirana tudi na drugih verigah blokov. Rudar, ki hoče potrjevat naslednji blok, mora narediti transakcijo stare valute na naslov, ki obstaja, a z njega nihče ne more narediti nakazila naprej. S tem se rudar prikaže kot zaupanja vreden, saj je vnaprej daroval svoje premoženje za to, da bi dobil pravico potrjevanja. Rudar, ki je v omrežju izvedel transakcijo z najvišjo vrednostjo oziroma požgal največ kovancev, bo dobil naslednji blok za potrjevanje in vse transakcijske provizije, kot prikazuje slika 3.13.

Dokaz požiga je predlagan tudi kot mehanizem soglasja za sistem zbiranja denarja za novo verigo blokov. Torej vsak, ki hoče potrditi blok, nakaže vrednost zbiralcem denarja, s tem pa pridobi lastništvo oziroma žetone zbiralca.

Težava algoritma je, da kljub temu, da ne porabi veliko energije ali namenske strojne opreme, uporablja stare kovance, ki so bili pridobljeni na ta način., stari kovanci pa so za vedno izgubljeni. Algoritem sicer skrbi, da imajo vsi rudarji enako možnost potrjevanja naslednjega bloka, saj ni potrebna na-



Slika 3.13: Diagram delovanja dokaza požiga.

menska strojna oprema oziroma večja količina kovancev, a v tekmovanju, kdo bo potrjeval naslednji blok, se postavi vprašanje, ali rudar lahko s transakcijami novih kovancev nadomesti vrednost, ki jo je porabil s požigom starih kovancev.

3.10 Merila za primerjavo algoritmov soglasja

Med poglavjem smo ugotovili, da različni algoritmi posedujejo različne lastnosti. Vsi poskušajo odpravljati napake predhodnikov, a lahko na ta način pridobijo nove lastnosti, ki so nezaželene. Zavedamo se, da različni mehanizmi soglasja delujejo na različne načine in so namenjeni različnim verigam blokov. V nekaterih verigah blokov so določene lastnosti potrebne, drugje zaželene, nekje pa morda ne vplivajo na verigo blokov zaradi domene, ki jo veriga blokov ureja. Verige blokov in algoritmi soglasja so v času pisanja dela predmet številnih raziskav, kar pomeni, da se popravki in novi algoritmi pojavljajo pogosto in odpravljajo pomanjkljivosti prejšnjih implementacij ali idej.

Kljub prej opisanim dejavnikom želimo poiskati nekaj lastnosti, tako da bomo lahko algoritme poskusili postaviti na isti imenovalcev in si pomagali pri kasnejšem izboru algoritma soglasja. Merila smo prepoznali tako, da smo tekom raziskovanja prepoznali pomanjkljivosti posameznih algoritmov soglasja. Pomanjkljivosti smo združili v naslednja merila.

- **Energijska učinkovitost** - Želimo si, da bi bil algoritem soglasja tudi energijsko učinkovit. Pri nekaterih algoritmih soglasja porabimo ogromne količine procesorskega časa, ki so namenjene same sebi. Rešitev uganke, ki jo algoritmi soglasja iščejo, nima nobenega drugega namena kot dokaz, da je rudar vložil nekaj energije v zavarovanje verige blokov. Energijska učinkovitost se zdi sploh velika težava, ko opazimo izračune porabe električne energije samo za verigo blokov Bitcoin. Ta veriga blokov letno porabi toliko električne energije kot manjša država [14]. Sprejemljivo bi bilo porabljati energijo v primerih, kjer bi iskanje rešitve na

uganko pripomoglo k računanju težav, za katere se porablja električna energija v vsakem primeru. Primer take težave, ki se dnevno pojavlja, bi lahko bilo napovedovanje vremena, ki se že danes računa na porazdeljenih sistemih. Poleg nizkih energijskih stroškov same potrditve je pomembna tudi nezahtevnost po programski opremi. V primerih, ko proizvedemo veliko strojnih komponent, ki bodo v svojem življenjskem ciklu namenjene zgolj rudarjenju, smo pri proizvodnji teh komponent povzročili ekološko težavo. Primer takih komponent bi lahko bili procesorji, grafične kartice, diskovni prostor.

Zaradi opisanega bodo slabše ocenjeni algoritmi soglasja, ki za svoje delovanje porabijo več procesorske moči ali dopuščajo prednost vozlišč z večjim številom komponent, ki so namenjen in v uporabi zgolj za mehanizem soglasja.

- **Zahteve po strojni opremi** - Za rudarjenje z algoritmom dokaza dela že obstajaja namenska strojna oprema. Tudi brez namenske strojne opreme imamo na trgu drage grafične kartice, ki zmorejo še uspešno rudariti kovance. Posledica takega rudarjenja je manjša decentraliziranost, saj lahko veliki investitorji naberejo veliko količino rudarske moči na kup, kar poveča možnost za napad z več kot 51 % vozlišč. Za dober algoritem soglasja si želimo, da bi bil visoko centraliziran, zato ne sme biti odvisen od namenske ali drage strojne opreme, ampak mora biti neodvisen od strojne opreme oziroma mora biti odvisen od relativno poceni strojne opreme, ki ni namenska in jo ima na voljo skoraj vsak uporabnik računalnika. V primeru, ko bo algoritem potreboval nenamensko strojno opremo in bo vsak udeleženec v omrežju imel možnost pridobiti plačilo za svoj prispevek v omrežje, se bo več ljudi odločalo za rudarjenje, kar posledično privede do velike distribucije, kar na drugi strani omejuje napade.

  V tej kategoriji bodo slabše ocenjeni mehanizmi soglasja, ki dopuščajo uporabo namenske strojne opreme, prilagojene za delo z mehanizmi

soglasja in ne za splošno rabo.

- **Časovna učinkovitost** - Časovno učinkovitost verige blokov lahko merimo med drugim v številu transakcij oziroma zapisov na verigo blokov v sekundi. Število transakcij v sekundi za verigo blokov mora biti skalabilno in mora že sedaj brez težav dosegati vrednosti, ki jih dosegajo obstoječi sistemi. Primer take pomanjkljivosti je število transakcij v verigi Bitcoin. Bitcoin v trenutni implementaciji ponuja teoretično najvišje sedem transakcij na sekundo. VISA, ki obvladuje le del bančnega sektorja, trenutno v viških procesira 10 tisoč transakcij na sekundo [21]. Pri tem je treba upoštevati, da obstajajo sektorji, ki procesirajo bistveno večje število transakcij v sekundi. Število transakcij na sekundo pripomore k hitrejši potrditvi transakcij, kar je izredno pomembno za prihodnost uporabe verige blokov.

  Algoritmi soglasja, ki imajo teoretično majhen vpis potrjenih transakcij v verigi bodo v tej kategoriji ocenjeni slabše.

- **Varnost** - Varnost verige blokov je zelo pomembna. Veriga blokov je bila implementirana z željo, da prinaša zaupanje v zapisane podatke. Večina ljudi sicer nikdar ne bo vedela, kako je zaupanje v verigo blokov implementirano, vendar se bo veriga blokov uporabljala le v primeru, da bo bo zaupanje upravičeno. Zaupanje v verigo blokov bo upravičeno takrat, ko bodo možnosti za napad in s tem popravek zapisov v verigi blokov onemogočene. Velik del zaupanja torej prispeva varnost, ki je omogočena preko mehanizma soglasja. Mehanizem soglasja določa, katere transakcije se zapišejo v verigo blokov. Te transakcije morajo biti pravilne in nespremenljive, ko so enkrat zapisane. Najbolj znana napada, ki ju poskušamo z dobrim algoritmom soglasja preprečiti, sta možnost dvojnega porabljanja in napad na omrežje z več kot 51 % vozlišč.

  Algoritmi, kjer je lažje doseči pogoje za napad z 51 % vozlišč ali z dvojno porabo bodo v tej kategoriji slabše ocenjeni kot tisti, pri katerih

je take pogoje težje ustvariti.

- **Porazdeljenost** - Veriga blokov in doseganje soglasja na le-tej morata biti čim bolj porazdeljena. Z večjo porazdelitvijo se veča varnost same verige blokov, hkrati pa večja distribucija pomeni tudi lažji skok v svet verig blokov, hitrejši razvoj aplikacij in sistemov, ki za osnovo vzamejo verigo blokov in posledično tudi hitrejše in boljše sprejetje med ljudmi. Večja porazdeljenost pogosto pomeni tudi večjo priljubljenost, kar poleg hitrejšega sprejetja sistema pomeni tudi hitrejši nastanek regulativ na področju verige blokov in večje zaupanje.

  V tej kategoriji bodo bolje ocenjeni mehanizmi soglasja, ki bodo zaradi svoje implementacije zahtevali visoko porazdeljenost in čim večjo enakost med vozlišči.

- **Razširljivost** - V primeru večje porazdelitve imamo opravka z velikim številom vozlišč. Vozlišča v verigi blokov se poskušajo dogovoriti o vsebini naslednjega bloka. Za določanje naslednjega bloka ne obstaja ena centralna avtoriteta, kar pomeni, da se mora velikokrat vsako vozlišče dogovoriti z drugimi, kakšen naj bo naslednji blok. Pomembno se zdi, da je algoritem soglasja zato razširljiv. Proces usklajevanja ne sme trajati preveč časa, zato je zaželeno, da ima algoritem soglasja optimizacijske procese, ki kljub povečanju omrežja poskrbijo za hitro dosego soglasja. Želimo si algoritem soglasja, ki bo skalabilen tudi na področju hranjenja podatkov, saj želimo, da je sistem vzdržen tudi na dolgi rok. Algoritmi soglasja se bodo morali v določeni točki spopasti s težavo velike količine podatkov v verigi blokov. Že leta 2015 je veriga blokov Bitcoin zrasla za 14 GB [21]. V primeru bančnih transakcij VISA bi letno veriga blokov narasla za 1,42 PB.

  Mehanizmi soglasja, ki predvidevajo optimizacijski proces pri doseganju soglasja, bodo bolje ocenjeni. Za hranjenje podatkov, zapisanih na verigo blokov, vsi algoritmi porabijo približno enako prostora. Vsi namreč zapisujejo reference na prejšnji blok, dodajo časovni žig in podatke

transakcij, zato velikosti zasedenega prostora ne bomo upoštevali.

3.11 Primerjava mehanizmov

V nadaljevanju bomo primerjali algoritme med seboj in utemeljili ocene po posameznih merilih. Pomembno se je zavedati, da merilih niso popolnoma enakovredna. Na merila si moramo postaviti uteži ob iskanju primernega algoritma soglasja. Merila nam lahko različno pomenijo glede na domeno, za katero bomo verigo blokov uporabljali. V spodnji tabeli smo poskušali oceniti algoritme soglasja. Ocene so med 1 in 3, kjer je 1 pomeni problematično na določenem področju, 3 pa dobro. V nadaljevanju so ocene tudi obrazložene.

V tabeli smo izvrgli dokaz lokacije, saj bi ga lahko opredelili le kot vrsto dokaza dela. Poleg tega smo dokaz pretečenega časa, opisanega v delu dokaza sreče [11], enačili z dokazom pretečenega časa, saj gre za isto idejo. Poudariti je treba, da je dokaz oblasti še precej teoretičen in njegovo ocenjevanje ni preveč realno, dokler ne doživi določene splošno uporabljene implementacije. Idejo vseeno primerjamo, ker se nam zdi, da ima možen potencial, saj verjamemo, da bodo za splošno sprejetje uporabe verige blokov uporabniki le-te potrebovali identifikacijo ali vsaj pseudo-identifikacijo. S strani države in prava bi bilo težko verjetno, da bi uporabniki ostali anonimni.

- **Energetska učinkovitost:** Energetska učinkovitost posameznega algoritma je odvisna od porabljene energije na blok, poleg tega pa je pravilno, da upoštevamo tudi proizvodnjo strojnih komponent, ki so proizvedene namensko za rudarjenje. Dokaz dela je problematičen zaradi samega iskanja zgoščenih vrednosti. Tudi primeri algoritmov, ki imajo manjšo zahtevnost kot veriga Bitcoin, še vedno preračunajo veliko zgoščenih vrednosti, ki se v svetovnem merilu hitro nabirajo. Za računanje le-teh se porablja ogromno električne energije. V ta namen rudarji tudi kupujejo namensko strojno opremo, za hlajenje takih naprav pa je treba še dodatno zagotoviti energijo. Z energetskega stališča

Tabela 3.1: Tabela ocen primerjav mehanizmov soglasja.

	Energetska učinkovitost	Zahteve po strojni opremi	Časovna učinkovitost	Varnost	Porazdeljenost	Razširljivost	**Skupaj**
Dokaz dela	1	1	1	1	1	3	**8**
Dokaz deleža	3	3	3	2	1	1	**13**
Dokaz pretečenega časa	3	3	3	3	3	2	**15**
Dokaz prostora	2	2	3	2	2	3	**14**
Dokaz sreče	3	3	3	3	3	3	**18**
Dokaz oblasti	3	3	3	3	1	3	**16**
Dokaz pomembnosti	3	3	3	3	1	3	**16**
Dokaz požiga	2	3	3	2	1	3	**12**
Dokaz dela v ZIO*	1	3	3	3	3	1	**14**
Dokaz lastništva v ZIO*	3	3	1	3	3	3	**16**
ZIO=zaprto izvajalno okolje							

se nič drugače ne obnaša rudarjenje dokaza dela v zaprtem izvajalnem okolju.

Boljše se po tem merilu izkaže dokaz prostora in dokaz požiga. Prvi ravno tako kot dokaz dela preračuna veliko zgoščenih vrednosti v začetni fazi načrtovanja, kasneje pa preračunavanja ne izvaja več. Na verigah blokov, ki trenutno obstajajo z dokazom prostora, prihaja do tekmovanja med rudarji. S tem se količine diskovnega prostora, rezerviranega za rudarjenje, večajo, kar povzroča nesmiselno proizvodnjo nosilcev podatkov. Dokaz požiga v potrjevanja nakazuje kriptovalute, tudi tiste, ki so bile pridobljene z rudarjenjem s potratnimi algoritmi soglasja. Iz tega sledi, da je potrjevanje tako potratno kot kovanci, ki so bili pri postopku požgani.

Dokaz deleža, dokaz pretečenega časa, dokaz sreče, dokaz lastništva in dokaz pomembnosti so okoljsko najbolje sprejemljivi algoritmi.

- **Zahteve po strojni opremi:** Danes izvajajo rudarji dokaz dela s pomočjo namenske strojne opreme. Če ne gre za namensko strojno opremo, se rudarjenje izvaja na dragih grafičnih karticah. Z opisanimi značilnostmi podamo dokazu dela najslabšo oceno.

  Dokaz prostora sicer ne zahteva posebne strojne opreme za svoje potrjevanje, a v želji po večjem zaslužku rudarji kupujejo ogromne količine diskovnega prostora. Algoritem se je iz želje po algoritmu soglasja, kjer bo vsak oddal svoj neporabljen prostor, preselil v tekmovanje, kdo lahko kupi več prostora in je cenovno še vedno vzdržljiv. Tako zaradi praktičnega vidika dokaz prostora spada v boljšo skupino kot dokaz dela.

  V najboljši skupini so vsi drugi algoritmi soglasja. Nekateri se naslanjajo na trenutno imetje. Ti so dokaz deleža, dokaz pomembnosti in dokaz požiga. Na drugi strani imamo dokaz oblasti, ki se naslanja na javno razkritje rudarja. Tretja skupina algoritmov pa so algoritmi, ki za svoje izvajanja zahtevajo poseben ukaz na procesorju za zaprto izva-

jalno okolje. Taki algoritmi so sicer vezani na posebno strojno opremo, ampak ker gre za strojno opremo, ki je relativno poceni in dosegljiva v vsakem procesorju, od najnižjega cenovnega ranga do najvišjega. Taki algoritmi so dokaz pretečenega časa, dokaz sreče, dokaz dela in dokaz lastništva v zaprtem izvajalnem okolju.

- **Časovna učinkovitost:** Dokaz dela je s strani časovne učinkovitosti najslabši, saj je za naslednji blok potrebno določeno število časa oziroma veliko preračunov zgoščenih vrednosti. Sama težavnost algoritma je sicer prilagodljiva, a hitrejše iskanje rešitev pri tem algoritmu pomeni tudi problematično varnost. Dokaz dela v zaprtem izvajalnem okolju s stališča količine potrjenih blokov ni bistvena izboljšava. V tem primeru lahko zahtevnost rahlo spustimo, saj bomo s tem ohranili podobno varnost verige, a še vedno bo za rudarjenje potrebno veliko preračunavanja zgoščenih vrednosti, čemur pa splošni procesorji, ki nudijo zaprto izvajalno okolje, niso namenjeni.

  Pomemben premislek velja še za dokaz prostora. Dokaz prostora že vnaprej preračuna vse potrebne zgoščene vrednosti, med samim potrjevanjem bloka pa le poišče svoje najboljše rešitve. Pri časovni učinkovitosti se posvečamo zlasti časovnemu obdobju med dvema potrjenima blokoma in minimumu tega obdobja brez tveganja za druga merila. Dokaz prostora tako kljub začetnemu časovnemu vložku spada v boljše algoritme soglasja glede na merilo časovne učinkovitosti. V to kategorijo spadajo še vsi preostali algoritmi, saj večino časa med bloki porabijo za usklajevanje vozlišč med seboj. Posamezne implementacije algoritmov pa vsebujejo tudi optimizacijske korake, ki pripomorejo k hitrejši uskladitvi omrežja.

- **Varnost:** Pri varnosti se osredotočamo predvsem na nevarnost z 51 % vozlišč v omrežju in napad z dvojnim porabljanjem. Napade na programsko kodo v tej značilnosti ne upoštevamo kot napade na točno določeno implementacijo verige blokov. Največja možnost je napad na

verige blokov, ki imajo implementiran algoritem dokaza dela, saj v primeru premalo porazdeljene verige lahko pride do množice rudarjev, ki še preostale rudarje prepričajo o pravilnosti daljše napadalčeve verige. Napadalec lahko ob tvorbi omrežja z namensko strojno opremo doseže hitro razvejitev in kasneje vsili lastno zgodovino verige blokov. Ta napad lahko prepreči že samo izvajanje rudarjenja v zaprtem izvajalnem okolju, še boljša pa je čim večja porazdeljenost omrežja.

Podoben napad, le težje verjeten, lahko napadalec poskuša izvesti s pomočjo verig blokov z dokazom prostora, le da za to potrebuje zelo veliko skupnega diskovnega prostora v svojem napadalnem omrežju. Težave z varnostjo lahko imata tudi dokaz deleža in dokaz požiga. V obeh primerih se namreč lahko zgodi, da napadalčev delež v valuti zaseda le manjši del vsega njegovega premoženja, zato je možnost napada, kjer bi daroval del premoženja v eni valuti in s tem pridobil več druge valute, možen.

V skupino dobrih algoritmov spadajo dokaz pretečenega časa, dokaz sreče, dokaz pomembnosti, dokaz lastništva in dokaz dela v zaprtem izvajaltnem okolju ter dokaz oblasti. Zadnji je dobra rešitve zaradi javnega razkritja rudarjev in posledično pravnih posledic, ki bi jih tak napad imel. V primeru algoritmov soglasja, ki uporabljajo zaprto izvajalno okolje, pa lahko računamo na zelo veliko porazdeljenost, zaradi česar so napadi zelo oteženi.

- **Porazdeljenost:** Pri porazdeljenosti omrežja nas zlasti zanima kako so razporejeni rudarji in njihova moč. Koliko rudarjev obstaja in koliko moči ima vsak izmed njih na voljo. Želeli bi si porazdelitev, kjer bi vsak v omrežju glasoval za naslednji blok in bi bili vsi v omrežju enakovredni. Zaupamo namreč, da je pravilna resnica na verigi blokov tista, v katero zaupa večina vozlišč. Večje, kot je število vozlišč, težje je z napačno verigo prepričati polovico vozlišč.

  Slabo porazdelitve v tej kategoriji imajo dokaz dela, dokaz deleža in

dokaz oblasti. Dokaz oblasti svojo varnost prenaša na identifikacijo in pravni sistem. V primeru dokaza oblasti bi rudarjenje izvajali le posamezniki, ki so se pripravljeni razkriti javnosti, takih pa, menimo, da je relativno malo. Dokaz dela je lahko slabo porazdeljen zaradi namenske strojne opreme. Po drugi strani je lahko slaba porazdelitev tudi pri dokazu deleža, saj je napadalčev delež v posamezni verigi zanemarljiv v primerjavi z njegovim premoženjem, čeprav je glede na druga vozlišča velik. Tako lahko napadalec napade verigo blokov, saj mu nekaj odstotkov imetja ne pomeni toliko, kot jih bo s tako dejavnostjo pridobil. Podobna težava je pri dokazu požiga in dokazu pomembnosti. Z več premoženja lahko namreč do določene stopnje dvigneš svoj status in je verjetneje, da si izbran za potrjevalca naslednjega bloka.

Manj verjeten je dvig statusa, ko gre za diskovni prostor, saj ob velikem številu naprav na svetu, ki premorejo vsaj nekaj prostega prostora, lahko rečemo, da velike količine diskovnega prostora zelo malo pripomorejo k verjetnejšemu izboru za potrjevanje bloka.

Veliko distribucijo nam omogoča ukaz v zaprtem izvajalnem okolju, ki je na širokem spektru splošnonamenskih procesorjev. Posledica tega je razširjenost algoritma med vsemi računalniki v omrežju, v katerem imajo vsi enake možnosti potrjevanja naslednjega bloka. Taki algoritmi so dokaz pretečenega časa, dokaz sreče, dokaz dela in dokaz lastništva v zaprtem izvajalnem okolju.

- **Razširljivost:** Razširljivost je odvisna od časa, pretečenega med dvema blokoma. Običajno je ta čas namenjen uskladitvi vozlišč o trenutnem stanju verige blokov in določitvi potrjevalca naslednjega bloka in njegovih transakcij. Uskladitev vozlišč o trenutnem stanju je časovno podobno zahtevna. V nekaterih implementacijah se sicer pojavijo določene optimizacije, a vsa vozlišča morajo pridobiti sliko verige blokov, preden začnejo tekmovanje za potrjevanje naslednjega bloka.

  Tako je razširljivost v veliki meri odvisna le od določitve potrjevalca

bloka. Pri dokazu dela je to postopek, ki hkrati tudi zavaruje verigo blokov. Zahtevnost uganke pri dokazu se sicer lahko spreminja, a le do mere, kjer varnost ne trpi bistvenih posledic. Tako je postopek dokaza dela najmanj razširljiv. Prav tako se isti algoritem v zaprtem izvajalnem okolju ne izkaže za bistveno boljšo alternativo.

V srednje dobro skupino spada algoritem dokaza pretečenega časa. Pri majhnem številu vozlišč se namreč lahko zgodi, da je čakanje na naslednji blok relativno dolgo. S povečanjem števila vozlišč pa bi moralo biti potrjevanje hitrejše. Posledično bi lahko ta algoritem uvrstili tudi med boljše v kategoriji razširljivosti.

Preostali algoritmi, ki določijo potrjevalca s pomočjo žreba, so hitri in razširljivi.

Poglavje 4

Ogrodja s področja verig blokov

V naslednjem poglavju bomo opisali nekatera ogrodja s področja verig blokov, za katera menimo, da bi lahko bila prava ogrodja na poti uveljavljanja uporabe verige blokov v družbo. Najprej bomo pogledali projekt Quorum, ki je v času zaključevanja tega dela izdal prvo testno različico svojega ogrodja. Projekt se ukvarja z verigo blokov, namenjeno denarnim transakcijam. Za potrjevanje blokov uporablja več mehanizmov soglasja. S svojo trenutno zmogljivostjo potrjevanja transakcij se dokazuje kot resen korak v razvoju. Poleg omenjenega projekta bomo opisali še dve ogrodji, razvijajoči se v projektu Hyperledger. Obe ogrodji odlikujejo modularnost mehanizma soglasja, zmogljivost potrjevanja transakcij in možnost široke adaptacije na različne domene.

V svetu obstaja še več drugih ogrodij verig blokov, kot so verige blokov Bitcoin, Etherum in Ripple, a jih zaradi njihove nizke pretočnosti potrjevanja transakcij in nezmožnosti uporabe alternativnega mehanizma soglasja ne bomo opisovali.

4.1 Hyperledger

Hyperledger je projekt, ki se je oblikoval v skupnosti The Linux foundation skupnosti. V skupini poleg razvijalcev sodelujejo tudi podjetja, ki se

ukvarjajo z informacijskimi tehnologijami, bančništvom, financami in drugo industrijo, ki bi lahko uporabljala tehnologijo verige blokov. Razvoj se izvaja na več vzporednih projektih, ki poskušajo s svojimi izdelki približat uporabo verig blokov industriji. Pod okriljem skupnosti se razvija več ogrodij in orodij na področju verige blokov. Med največjimi in trenutno že dobro razvitimi projekti so ogrodja, s katerimi je možno razviti lastno, domeni prilagojeno verigo blokov. Pomembna projekta sta:

- **Fabric** je bil prvi izmed projektov Hyperledger. Ns začetku se je razvijal pri podjetju IBM, nato pa je bil predan v razvoj skupnosti. Ideja projekta Fabric je izdelati modularno verigo blokov, kjer lahko mehanizem soglasja ali druge komponente verige blokov poljubno zamenjamo. V primerjavi z običajnimi varigami blokov Fabric ponuja možnost zasebnih in zaupnih transakcij, ki so v svetu industrije včasih nujne. Rešitev stremi tudi k veliki razširljivosti in varnosti. Veriga blokov Fabric omogoča možnost zasebne verige oziroma verige, kjer ima posameznik dovoljenja videti določen del podatkov, ne pa vseh. Primeri takih verig blokov bi bili tisti na področju medicine, financ, lastništva, itd. Fabric z algoritmom bizantinske tolerance napak zmore potrjevanje več kot 10 000 transakcij na sekundo [22].

  Hyperledger Fabric se že uporablja na nekaterih projektih znotraj različnih podjetij, in sicer kot zasebna veriga blokov.

- **Sawtooth** je projekt, katerega začetni razvoj je prevzelo podjetje Intel. Sawtooth med drugim lahko uporablja algoritem dokaza pretečenega časa, za kar potrebuje Intelov procesor, katerega ukazni nabor vsebuje ukaz SGX. Ukaz SGX Intel vgrajuje v procesorje vseh cenovnih razredov, saj želijo, da lahko pri soglasju sodeluje čim več vozlišč, kar povzroči veliko razpršenost omrežja in s tem večjo varnost. Ukaz za svoje delovanje zalo malo obremeni procesor, zato so zanj primerni vsi zmogljivostni razredi procesorjev, hkrati pa je možnost izvedbe zaklenjena na natanko en glas na procesor, saj ima vsak izdelan procesor

svojo edinstveno identifikacijsko številko.

Sawtooth stremi tudi k čim večji uporabnosti in prijaznejšemu razvoju, zato cilja na možnost razvoja v čim več programskih jezikih. V tem trenutku polno ali delno podporo že omogoča za programske jezike C++, Go, Java, JavaScript, Python in Rust.

Arhitektura Sawtootha omogoča pisanje različnih aplikacij na isto verigo blokov ter pisanje poslovnih pravil v programskem jeziku in izvajanje na verigi blokov. Omogoča tudi možnost različnih ravni dovoljenj za uporabnike in spreminjanje mehanizma soglasja. Za hitrejše procesiranje transakcij ima Sawtooth vgrajeno analizo transakcij, ki ugotavlja, katere izmed transakcij so medsebojno neodvisne in se lahko vpišejo istočasno in s tem odpravlja pomanjkljivost drugih verig blokov z nizkim številom transakcij.

4.2 Cypherium

Project Cypherium je trenutno v razvoju in poskuša prilagoditi dokaz dela. Avtorji projekta trdijo, da je dokaz dela zaradi svoje narave med najbolj varnimi postopki, a so njegove težave razširljivost, energijska potratnost, hkrati pa napake z več kot tretjino zlonamenrnih vozlišč, ko govorimo o manjših omrežjih [23]. Avtorji želijo ustvariti verigo blokov, ki bo imela možnost dinamičnega določanja parametrov algoritma potrjevanja in bo dovolj hitra, da bi zadostila potrebam bančnega sektorja po nekaj 10 000 transakcijah na sekundo. V svojem delu so se osredotočili na dejstvo, da je tekmovanje z dokazom dela pomembno le zaradi določitve potrjevalca bloka. V ta namen predlagajo dve verigi blokov. Prva veriga blokov je veriga potrjevalcev, kjer vozlišča tekmujejo tekmujejo kdo bo potrjeval blok. Druga veriga pa je sestavljena iz dejanskih potrjevnih blokov transakcij. Prvo verigo tako upravlja algoritem dokaza dela, za urejenost druge pa algoritem bizantinske tolerance napak. Napad na transakcije bi se lahko zgodil le v primeru, da bi bilo napoštenih vozlišč več kot 1/3 v omrežju.

Veriga blokov potrjevalcev izbere za vsako transakcijo komisijo in vodjo, ki vsak blok transakcij potrdi in je znana v naprej. Po potrjenem bloku transakcij tako ni treba čakati nekaj naslednjih blokov, da lahko potrdimo, da so transakcije potrjene. Potrditev transakcije je instantna. Avtorji članka razvijajo algoritem dokaza dela tako, da je vezan na zaledni pomnilnik, s čimer želijo doseči odpornost proti namenski strojni opremi za rudarjenje in tako omogočiti široko razpršenost rudarjev. Projekt je trenutno še v razvoju, avtorji pa so že omenili, da so v testnem okolju dosegli več kot 10 000 transakcij na sekundo.

4.3 Izbira ogrodja

Kot boljšega izmed navedenih smo izbrali ogrodje Sawtooth. Ogrodje Cypherium je še v razvoju in ga še ne moremo uporabljati. Ogrodje Fabrik je v svojih lastnostih primerejše za zasebne verige blokov, medtem ko je ogrodje Sawtooth primerno tudi za javne verige blokov. Obe ogrodji sta modularni in imata visoko prepustnost potrjevanja transakcij. Projekt Sawtooth je zanimiv tudi zaradi že implementiranega mehanizma soglsja dokaza pretečenega časa.

Poglavje 5

Implementacija knjižice vzdrževanja vozila

V delu želimo pokazati delovanje enega izmed bolje ocenjenih mehanizmov soglasja na praktični domeni. Avtomobilska industrija oziroma hranjenje zgodovine vozila se nam zdi kot domena, ki bi jo lahko implementirali s pomočjo verige blokov. Za ogrodje smo si izbrali Hyperledger Sawtooth, ki lahko soglasje upravlja z dokazom pretečenega časa. Dokaz pretečenega časa smo v poglavju 3.11 ocenili kot enega izmed boljših danes poznanih algoritmov.

5.1 Opis problema

Danes je na svetu v uporabi več kot 1,2 milijarde motornih vozil [24]. Vsa omenjena prevozna sredstva moramo sprotno vzdrževati glede na navodila proizvajalca vsaj enkrat letno, v nekaterih primerih tudi pogosteje. Pravilno vzdrževanje vozila je zelo pomembno za njegovo delovanje, kar odraža tudi njegova trenutna vrednost na trgu. Poleg rednih vzdrževalnih del tako imenovanih servisov na vozilu pogosto opravimo tudi druge preglede ali popravila vozila, ki niso označena v navodilih proizvajalca. Primeri takih postopkov so nadgradnje, popravila zaradi prometnih nesreč ali popravila, ki

niso vezana na standardno vzdrževanje avtomobila. Predpisani servisi vozil so običajno vpisani v za to namenjeno servisno knjižico, ki jo proizvajalec priloži ob nakupu vozila. Zgodi se lahko, da tudi običajna vzdrževalna dela v servisni knjižici niso opisana, saj lahko lastnik vozila servise opravlja pri nepooblaščenem serviserju vozil. V primerjavi z rednim vzdrževalnim delom na vozilih, ki so precej dobro popisana, se neredna dela po navadi ne popisujejo. V primeru vestnega lastnika je edini dokaz o delu na vozilu račun izvajalca.

Z večjimi težavami kot ob samem lastništvu se z nazabeleženim vzdrževanjem na prevoznem sredstvu srečamo ob prodaji. Ob prodaji oziroma nakupu rabljenega vozila bi kupec, prihodnji lastnik rad čim podrobneje poznal zgodovino prevoznega sredstva. Prodajalec vozila lahko sicer predloži servisno knjižico, če je ta izpolnjena, v njej pa ni izpolnjenega marsikaterega podatka o avtomobilu, ki bi mogoče prihodnjega lastnika zanimali. Prodajalec vozila lahko sicer ustno preda tudi ostalo zgodovino vozila kupcu ali jo zaradi lastnega interesa po višji ceni vozila zamolči. Kupec je tako primoran zaupati v ne nujno resnično zgodovino vozila. Poleg omenjenega problema, kjer lastnik zamolči del zgodovine vozila, obstajajo tudi kriminalne združbe, ki prevoženo razdaljo vozila zmanjšajo in prodajajo vozilo kot bistveno novejše, kot le-to je. Posledica takih dejanj je še večje nezaupanje med prodajalci in kupci vozil.

Odgovor na opisane težave vidimo v možnosti zapisa podatkov dela na avtomobilu v verigo blokov. Vsak zapis mora vsebovati identifikacijo vozila, trenutno stanje števca vozila, opis dela na vozilu, datum dela na vozilu, identifikacijo podjetja oziroma osebe, ki je vnesla zapis, in časovni žig.

Kupec bi tako lahko pregledal zgodovino vozila, ugotovil zamolčana vzdrževalna dela, popravljeno stanje števca itd. Iz podatkov na verigi blokov bi bilo namreč možno razbrati, kdaj in kje je bilo delo na vozilu opravljeno. Kupec lahko tako zaradi javne dostopnosti podatkov, zapisanih na verigo blokov, jasno vidi, ali je sosledje zapisov smiselno, ali se stanje števca smiselno ujema s stanjem ob zadnjem delu na avtomobilu, in morebitne napake vozila, ki so

bile zaznane na zadnjih pregledih vozila, a niso bile odpravljene.

5.2 Izbira ogrodja in mehanizma soglasja

Iz opisa težave lahko izluščimo, da v procesu vzdrževanja in prodaje vozila sodeluje veliko deležnikov, ki si med seboj ne poznajo in si ne zaupajo. Največje nezaupanje predstavlja posredovana zgodovina vzdrževanja vozila s strani prodajalca kupcu. Centralna avtoriteta, ki bi beležila vsa dela na vozilu, ne obstaja. Upoštevati je treba, da se vozila ne prodajajo samo znotraj države, ampak v velikih merah vozila potujejo tudi med državami, torej bi v namen hrambe podatkov morebitna centralna avtoriteta morala biti mednarodna. Če bi se pojavilo več deležnikov, ki bi centralno zbirali podatke zgodovine vozil, bi tako delovanje postalo problematično, saj bi morali vsakemu vozilu prilagoditi poizvedovanje. Poleg vsega naštetega bi bili vpisi, poizvedbe in druge storitve centralnih avtoritet dražje. Želimo si, da bi bili podatki o zgodovini vozila varno shranjeni in neizbrisljivi, preprosto dostopni, a bi kljub temu zaščitili identiteto deležnikov, kot so lastniki vozil oziroma tisti, ki so ga upravljali.

Za nov zapis v zgodovini vozila ni pomembno, da je takoj shranjen, njegovo shranjevanje lahko traja nekaj minut. Pomembno je, da se vsi zapisi shranijo in so neizbrisljivi ter nespremenljivi. Radi bi hranili vsako delo na vozilu, hkrati pa je vozil na svetovni ravni veliko, kar pomeni, da moramo omogočiti veliko število zapisov. Na svetu je bilo leta 2015 po podatkih WHO približno 1,7 milijarde registriranih vozil [24]. Vsako vozilo ima redni vsaj enkrat letno, iz česar lahko preračunamo, da bi naša veriga blokov v povprečju samo za servise morala uspešno potrditi 57 transakcij na sekundo. Realna številka, bi bila gotovo višja, saj se poleg rednih servisov vozila vzdržujejo tudi na popravilih. Določen odstotek vozil zaradi pogostejše uporabe potrebuje pogostejše vzdrževanje. Tako lahko predvidevamo, da bi bilo potrebno število transakcij vsaj dvojna vrednost izračunane.

Za izdelavo informacijskega sistema za hranjenje podatkov o zgodovini vo-

zila smo zaradi vseh prej naštetih težav uporabili tehnologijo verige blokov. Svoj prototip smo zgradili na ogrodju Hyperledger Sawtooth z uporabo dokaza pretečenega časa kot algoritem soglasja. Ogrodje Sawtooth smo izbrali, ker nam s svojo modularnostjo omogoča, da lahko kadar koli mehanizem soglasja zamenjamo z drugimi mehanizmi soglasja. Poleg navedenega je bilo ogrodje Sawtooth med drugim razvito z mislijo na uporabnost v oskrbovalnih verigah. Hrambo podatkov o izdelkih in njihovi oskrbovalni poti lahko namreč primerjamo s hrambo podatkov o vozilu in njegovih oskrbovalnih delih. Primerno se nam zdi tudi, da različni deležniki posedujejo različne dostope oziroma vloge in s tem stopnjo pravic pri sodelovanju v verigi blokov. Veriga Sawtooth sama po sebi ne deluje z žetoni, zato ima nizek potencial za visoko razpršenost vozlišč. Nizka razpršenost vozlišč lahko predstavlja težavo zaradi vozlišč, ki bi bila zlonamerna in bi poskušala spremeniti vsebino verige blokov. Transakcije bodo tako potrjevala le vozlišča, ki imajo pravice za potrjevanje in so jim v primeru, da poskušajo vstaviti zlonamerne bloke, pravice takoj odvzete. Vsi deležniki, ki potrjujejo ali vpisujejo podatke, se identificirajo s svojim javnim in zasebnim ključem.

Nizka razpršenost vozlišč in veriga brez žetonov morata biti za potrjevalce blokov poceni, a hkrati ne smeta ogrožati varnosti podatkov. Glede na posebno uporabo lahko trdimo, da visoke razširljivosti in velike porazdeljenosti za svojo verigo blokov ne potrebujemo. Zaradi vseh naštetih lastnosti se nam zdi dokaz pretečenega časa, ki je že implementiran v verigi Sawtooth, primerno izhodišče za naš projekt. Modularnost implementaciej verige blokov Sawtooth pa nam omogoča, da lahko kasneje v primeru primernejšega mehanizma soglasja le-tega zamenjamo.

Vlogo potrjevalcev bi tako v produkcijskem okolju prepustili proizvajalcem avtomobilov, ki bi se identificirali s svojim javnim ključem. Vloga je častna in je proizvajalcem v korist, saj je natančna zgodovina avtomobila v korist zadovoljstva lastnika z izdelkom in ima tako večjo možnostjo za nakup avtomobila istega proizvajalca. Poskus vnašanja zlonamernih podatkov v verigo pa bi pomenil očrnitev proizvajalca. Postopek potrjevanja transakcij

je v primeru dokaza pretečenega časa poceni v primerjavi z dokazom dela. Potrjevanje tako zahteva le nekaj ciklov strojne opreme in nekaj električne energije, kar je sprejemljivo, saj ni potrebe po namenskih dodatnih kapacitetah računalniške moči, ampak se lahko potrjevanje izvaja vzporedno z drugimi procesi, ki se izvajajo v računalniških centrih proizvajalcev.

Izvajalci del na avtomobilih svoje storitve o avtomobilih vpisujejo z identifikacijo svojega javnega ključa. Izvajalci del lahko svoj zasebni in javni ključ pridobijo in javno objavijo svoj javni ključ ter s tem pokažejo, koliko vozil jim je zaupanih. Sam vpis vzdrževalcu ne predstavlja dodatnega dela, saj je tudi v trenutnem sistemu vpisoval in potrjeval servisne knjižice vozil.

Za pregled verige blokov uporabnik ne potrebuje identifikacije. Zapisi v verigi blokov so javni in jih lahko pregleduje kdor koli. Tako lahko kdor koli izdeluje najrazličnejše preglede ali aplikacije, ki črpajo podatke iz verige blokov. Taki pregledi bi uporabnikom omogočili pregled zgodovine avtomobila, pregled splošne kakovosti avtomobilov po proizvajalcih, pregled del po delavnicah in njihovo obiskanost.

Ogrodje verige blokov Hyperledger Sawtooth, ima mehanizem volitev, ki se dogaja na isti verigi blokov. Tako lahko spreminjamo vozlišča nastavitve verige blokov ali nastavitve algoritma soglasja. Če bi želeli spremeniti obliko ali tipe vsebine na verigo blokov, lahko to storimo brez težav, saj lahko na obstoječo verigo blokov vstavimo novejši potrjevalec pravilnosti vsebine.

5.3 Delovanje ogrodja Sawtooth

Ogrodje Sawtooth je implementirano zelo modularno. Sestavljeno je iz številnih komponent, štvilne izmed njih pa so namenjene kasnejšemu nadomeščanju z novimi [25]. Za potrebe našega dela je pomembno, da prepoznamo naslednje module, prikazane na sliki 5.1.

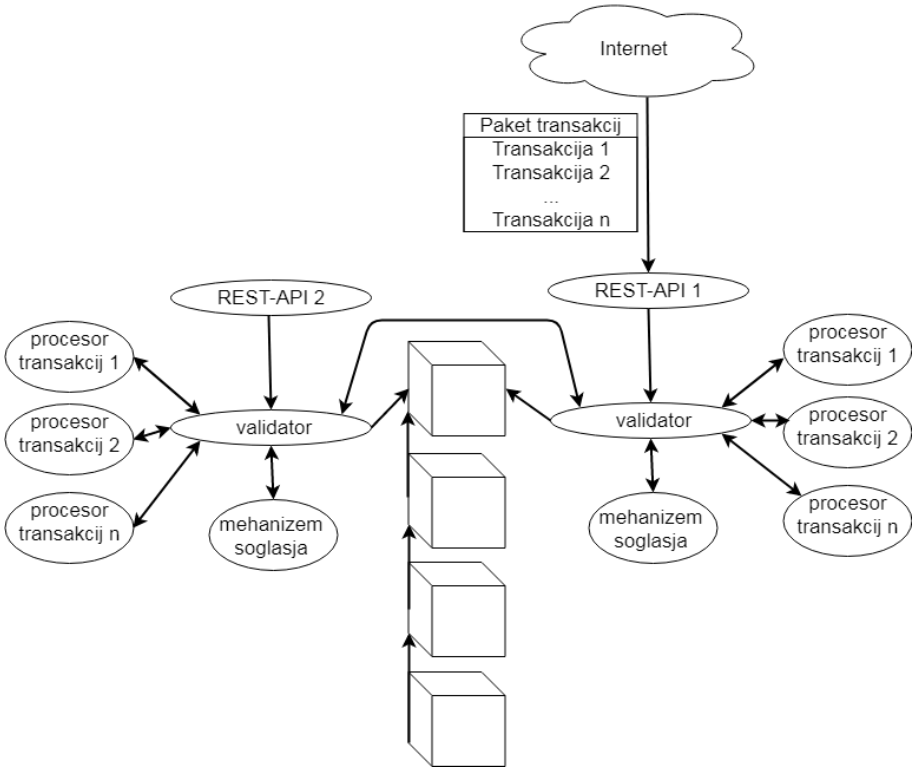
- **REST-API** - Sistem, ki sprejema podatke z interneta, ki so jih poslale stranke. Sistem preveri pravilnost poslanega sporočila in prejetje vseh potrebnih podatkov, kot so javni ključi pošiljatelja in podobno.

- **validator** - Validator je potrjevalec bloka. REST-API mu poda prejete podatke, validator pa preveri paket transakcij in transakcije razdeli med procesorje transakcij glede na njihovo družino in različico družine. Ob ponovnem prejetnju transakcij s strani procesorja transakcij dobi vsebinsko potrditev pravilnosti transakcije. Če potrditve ne pridobi paket transakcij zavrne, v nasprotnem primeru pa poskuša paket transakcij preko mehanizma soglasja potrditi. V postopku komunicira z drugimi validatorji in tako skupaj vpišejo paket transakcij v naslednji blok v verigi blokov.

- **procesor transakcij** - Procesor transakcij ima predpisano družino in različico družine transakcij, ki jih sprejema. Validator mu pošlje njegove tipe transakcij. Validator transakcijo obdela in jo ponovno sestavi v transakcijo. V tem koraku lahko dodamo dodatno poslovno logiko nad prejetimi podatki. Obdelano transakcijo vrne validatorju.

- **mehanizem soglasja** - Mehanizem soglasja je prav tako modularen sistem v ogrodju Sawtooth. Omogoča nam zamenjavo poljubnega algoritma soglasja. Algoritem soglasja lahko sami implementiramo, na voljo so pa tudi tri različice in četrta v razvoju. Na voljo imamo razvojni mehanizem soglasja, ki je namenjen razvoju na ogrodju. Kasneje lahko mehanizem soglasja zamenjamo z enim izmed vseh algoritmov dokaza pretečenega časa. Za testiranje lahko uporabimo algoritem, ki v visokem programskem jeziku implementira ukaz SGX, kar nam omogoča test dokaza pretečenega časa, za produkcijsko okolje pa je primerna le implementacija dokaza pretečenega časa, ki uporablja ukaz SGX na procesorju, za kar moramo imeti tudi dovolj zahtevane strojne opreme.

Vsak zapis v verigo Sawtooth se začne s paketom transakcij, ki jih uporabniki pošiljajo na rest-api ogrodja. Vsak paket transakcij vsebuje najmanj eno transakcijo, lahko pa vsebuje več transakcij, ki so lahko iz različnih družin in različic transakcij. Paket transakcij se potrdi in zapiše na verigo blokov le v primeru, da so vse transakcije bloka pravilne in potrjene. Če je katera



Slika 5.1: Grafični prikaz arhitekture ogrodja Sawtooth.

izmed transakcij napačna se nobena transakcija v paketu ne potrdi. Rest-api sprejme podatke o transakcijah in jih poda validatorju, ki transakcije razdeli glede na družine in različice primernim procesorjem transakcij. Ogrodje ima že vnaprej spisanih nekaj procesorjev transakcij. Najpomembnejši procesor transakcij je procesor za nastavitve. Ta sprejema transakcije, ki v verigo blokov zapišejo trenutne nastavitve verige blokov. Svoj vnaprej implementiran procesor transakcij imajo še nastavitve mehanizma soglasja, nastavitve dostopov in dovoljenj ter procesor, ki skrbi za iskanje blokov in transakcij po verigi blokov. Po želji lahko vključimo samo tiste procesorje, ki so za našo verigo potrebni. Poleg že implementiranih je tukaj smiselno dodati še naš lastni procesor transakcij ali več njih. Ta skrbi poskrbi za zapis podatkov, za katere smo se odločili, da jih bomo zapisovali na verigo blokov. V procesor transakcij lahko dodamo tudi lastno poslovno logiko, ki dobljene podatke obdela in jih zapiše v transakcije. V primeru pravilnosti transakcije vrne validatorju, ki s pomočjo priklopljenega algoritma soglasja s drugimi validatorji transakcije potrdi.

5.4 Izvedba

Postopek izgradnje smo začeli s pridobitvijo izvorne kode preproste elektronske denarnice, ki jo kot primer uporabe nudi zbirnik programske kode ogrodja Sawtooth.

Celoten sistem smo oblikovali v vsebnikih Docker, ki nam omogočajo simuliranje večih neodvisnih računalniških sistemov v enem računalniškem sistemu. Za postavitev celotnega nabora sistemov s tremi vozlišči smo morali urediti še datoteko, ki jo podamo kot vhodno datoteko orodju za delo z vsebniki Docker compose.

S sistemom vsebnikov smo zgradili omrežje, ki vsebuje tri validatorje. Postavitev prvega postavi celotno nastavitev validatorjev in mehanizma soglasja. Vsakemu izmed validatorjev nato postavimo tudi lasten servis rest-api in več procesorjev transakcij. Obvezno moramo postaviti vsaj tri vsebnike s

procesorji transakcij. Vsak validator potrebuje procesor transakcij za:

- nastavitve verige blokov, ki ga je ogrodje vnaprej implementiralo
- procesor transakcij mehanizma soglasja v našem primeru dokaza pretečenega časa, ki ga je ogrodje prav tako implementiralo
- ter procesorja transakcij, ki smo ga implementirali sami.

V procesorju transakcij za nastavitve lahko nastavljamo, katere družine in različice transakcij bo validator sprejemal, nastavitve, kako poteka glasovanje in katera vozlišča lahko glasujejo. Preko omenjenega procesorja transakcij urejamo nastavitve mehanizma soglasja, ki nam omogočajo nastavitve števila transakcij oziroma paketov transakcij na blok in nastavitve. Procesor transakcij mehanizma soglasja uporablja ogrodje za upravljanje in statuse validatorjev.

Želimo si, da potrjevanje blokov izvajajo le vozlišča, ki predstavljajo proizvajalce vozil, zato je treba dodati še procesor transakcij, namenjen nastavitvam identitet. Ob zagonu lahko tako dodamo novo pravilo v verigo blokov, ki poskrbi, da lahko potrjevanje blokov izvajajo le validatorji. V ta namen v pravilo vpišemo javne ključe vozlišč, ki jim je dovoljeno potrjevanje blokov. Če se kateri izmed javnih ključev spremeni ali se katero izmed vozlišč pokaže kot zlonamerno, ga kasneje lahko odstranimo s seznama z glasovanjem validatorjev na verigi blokov.

Nadaljevali smo z implementacijo procesorja transakcij za svoj sistem. V sistem smo implementirali tri funkcije, in sicer dodajanje zapisa o delu, brisanje zapisa o delu in vnos novega avtomobila.

Vse funkcije imajo naslednje skupne parametre:

- VIN - Edinstvena identifikacijska številka avtomobila. Podatke o avtomobilu shranjujemo na naslov, ki je sestavljen iz te številke.
- delavec - Javni ključ osebe ali delavnice, ki je vpis izvedla.
- datum dela - Datum, ko je bilo delo opravljeno.

- delo - Predstavlja številke del iz registra.
- opis - Polje, v katerega lahko vpisovalec vpiše poljubno kratko opombo.
- stanje števca - Polje, kamor se vpiše stanje števca na dan dela.
- časovni žig - Polje, ki se samodejno nastavi, ko je zapis uspešno pregledal procesor transakcij.

Funkcija vnosa novega avtomobila sprejeme še dodatna parametra:

- proizvajalec - Parameter, ki nam pove proizvajalca avtomobila.
- model - Parameter, ki nam pove, za kakšen model avtomobila gre.

V primer vnosa novega vozila se v polje delo samodejno vpiše številka 0. Prav tako se ob vnosu vozila samodejno vpiše stanje števca 0. Ob vnosu vozila se preveri, ali vozilo z identično VIN-številko še ne obstaja.

Funkcije dodajanja in brisanja števila so zelo podobne. Brisanje je mišljeno kot popravek vpisovanja. V primeru, da bi delavec vnesel napačno popravilo, ga lahko izbriše v celoti ali delno tako, da vnese enak zapis le s funkcijo brisanja. Procesor transakcij v tem primeru vsem številkam v polju delo vrine negativen predznak. Tak zapis se shrani v verigo blokov.

Za testiranje smo implementirali še uporabniški vmesnik v ukazni vrstici, s pomočjo katerega vnesemo nove zapise. Uporabniški vmesnik sestavi vnos v pravilen zapis in ga pošlje na rest-api. Omogoča tudi iskanje zadnjega vnesenega zapisa na določenem naslovu.

5.5 Testiranje

Sistem je oblikovan po opisanem v poglavju 5.4. Zaradi programske napake na delu ogrodja naše testno okolje ne bo vsebovalo modula za identifikacijo in pravice vozlišč, kar pa ne bi smelo bistveno vplivati na rezultate testiranja. Omeniti je treba še, da naš sistem pošilja v paketu transakcij vedno le eno

Tabela 5.1: Tabela rezultatov testiranj.

	število transakcij na blok		
število vozlišč	**10**	**30**	**70**
3	94 t/min	121 t/min	135 t/min
5	98 t/min	128 t/min	168 t/min

transakcijo. Sistem bomo testirali tako, da bomo na rest-api validatorjev poslali transakcije in opazovali koliko transakcij bo sistem zapisal v eni minuti. Iz uporabniških izkušenj smo ugotovili, da podpora ogrodju v primerih, ko več validatorjev zapisuje na isti naslov, ne deluje najbolje. V realnem svetu to ne bi smela biti težava, zato se na to ne bomo ozirali. V namen testiranja bomo tako na vsak validator pošiljali zapise, ki delujejo za različna vozila, kar bi moralo preprečiti težave z usklajevanjem transakcij s skupnimi atributi. Teste bomo izvajali na različnem številu vozlišč, in sicer v omrežju s tremi in petimi vozlišči. Drugi atribut, ki ga bomo spreminjali bo število transakcij na blok. Poskušali bomo z 10, 30 in 70 transakcij na blok. Vsebnike smo poganjali na virtualnem okolju z 10 GB RAM-a in šestimi dodeljenimi jedrnimi nitmi procesorja Intel i7-3770.

Sistem smo testirali na okolju, ki je v veliki meri namenjeno testiranju in razvoju in ne produkcijski rabi. Rezultati bi se verjetno bolj spreminjali, če bi implementacijo sistemov postavili na fizične računalnike. Dokaz pretečenega časa, ki je trenutno uporabljen, bi morali zamenjati z različico, ki deluje na Intelovih procesorjih. V teh primerih bi pričakovali boljše rezultate testiranj, ki bi z različnimi parametri pokazali zelo različne rezultate.

Eden izmed projektov, ki so pripravljeni na uporabo v produkciji, je veriga, ki temelji na ogrodju Sawtooth in skrbi za zapisovanje medicinskih podatkov. Po navedbah podjetja, ki se ukvarja z razvojem, je njihova implementacija dosegla do 550 transakcij na sekundo [26], kar menimo, da bi bilo več kot dovolj za implementacijo knjižic z zgodovino popravil vozila.

Tabela 5.2: Tabela primerjave rešitve s trenutnim stanjem.

	papir	**digitalna**	**veriga blokov**
cena	nizka	srednja	srednja
dostopnost	nizka	srednja	visoka
ažurnost	slaba	boljša	najboljša
spremenljivost podatkov	mogoča	skoraj nemogoča	nemogoča

5.6 Analiza

V naslednji tabeli bomo primerjali rešitve, ki obstajajo z našo rešitvijo težave zapisovanja del na vozilu. Obstoječe rešitve predstavljajo servisne knjižice v papirnati obliki, nekateri proizvajalci avtomobilov pa servisno knjižico vodijo v digitalni obliki.

- cena - Cena knjižice na papirju je nizka. Tovarna natisne servisno knjižico v velikem številu izvodov po nizki ceni. Digitalna spletna knjižica zahteva od proizvajalca več denarja, saj mora za izvedbo poleg spletnega vmesnika kupiti še strojno opremo in vzdrževati sistem. Naša rešitev, prav tako potrebuje nekaj strojne opreme, ob dodelavi svoje rešitve pa bi potrebovali le še vzdrževanje sistema.

- dostopnost - Papirnata servisna knjižica obstaja v enem izvodu. Lastnik jo mora imeti pri sebi. Pri digitalni obliki servisne knjižice lahko lastnik vozila do nje dostopa kjer koli. V primeru svoje rešitve lahko do zgodovine vozila dostopa kdor koli v primeru, da ima identifikacijsko številko vozila.

- ažurnost - V servisni knjižici so običajno vpisani le servisi vozila, ki so predpostavljeni s strani proizvajalca vozila. V digitalni knjižici so poleg rednih servisov vpisana še druga dela na vozilu, ampak le v primeru, da so opravljena na pooblaščenem servisu. Naša rešitev dovoljuje vpise

vseh vzdrževalcev. Vpisujejo se lahko najrazličnejši dogodki vozila, kot so na primer tudi tehnični pregledi.

- spremenljivost podatkov - Pri papirnati obliki servisne knjižice se lahko podatki vpišejo kadar koli tudi za nazaj, ne da bi to kdo opazil. Digitalna oblika servisne knjižice lahko s svojo implementacijo zagotovi, nezmožnost urejanja podatkov za nazaj, oziroma bi se tako urejanje opazilo. Naša rešitev z verigo blokov dovoljuje možnost vnosa za nazaj, starih zapisov pa se z gotovostjo ne da spreminjati. Vpisi za nazaj se z lahkoto opazijo.

Naša rešitev deluje na javni verigi blokov, kjer je potrjevanje podatkov zaupano le peščici izbrancev, ki so vredni zaupanja. Tako vemo, da so podatki varno shranjeni, a hkrati vsem javno dostopni. Sistem sicer dopušča vnos podatkov za nazaj, a bi bil tak vnos viden in takoj sumljiv. Pomembna prednost naše rešitve je možnost vnosa del na avtomobilu, ki jih vnesejo neuradni servisi, avtokleparji, tehnični servisi in drugi deležniki, ki so upravljali avtomobil. Cenovno je naša rešitev primerljiva z obstoječo digitalno različico, ki jo ima že vsak proizvajalec.

5.7 Nadaljnji razvoj

Projekt ima dobro osnovo, vendar bo za produkcijsko rabo potrebno še veliko dela. V nadaljevanju bi bilo treba okolje oblikovat tako, da bi bilo nameščanje preprosteje. Sistem bi bilo treba namestiti na fizično strojno opremo in uporabiti implementacijo dokaza pretečenega časa, ki se izvaja na Intelovih procesorjih. S takim sistemom bi bilo treba poiskati najboljše parametre nastavitve mehanizma soglasja in verige blokov. Poleg tega, bi bilo treba dodati še možnost celotne zgodovine vozila in ne samo zadnje, kot sedaj to dovoljuje samo ogrodje. Rešitev iskanja celotne zgodovine bi zahtevalo precej razmisleka. Možne rešitve so vodenje podatkovne baze s s transakcijami, ki omenjajo identifikacijo vozila, ali preiskovanje vseh transakcij in iskanje

tistih, ki vozilo omenjajo. Potrebno bi bilo tudi iskanje po drugih podatkih. Primer takega podatka bi lahko bil, serviser vozila, model in proizvajalec vozila ter drugi. Razviti bi bilo treba tudi uporabniški vmesnik, ki bi deloval v brskalniku ter bi bil uporabniku prijazen in vedno dostopen.

Poglavje 6

Zaključek

V delu smo najprej opisali osnove verig blokov. V nadaljevanju smo se osredotočili na enega izmed najpomembnejših delov verige blokov, imenovanega mehanizem soglasja. Raziskali smo, kako potrjevanje blokov v verigo izvaja prva veriga blokov Bitcoin in pri tem raziskali delovanje dokaza dela. Zaradi pomanjkljivosti dokaza dela smo raziskali preostale mehanizme soglasja. V delu smo opisali delovanje dokaza deleža, dokaza prostora, dokaza požiga, dokaza pretečenega časa, dokaza oblasti, dokaza sreče in dokaza pomembnosti. Ob raziskovanju mehanizmov soglasja smo dobili podatke o pomanjkljivostih posameznih mehanizmov in sestavili merila, s katerimi smo primerjali algoritme soglasja med seboj. Ugotovili smo, da lahko sicer primerjamo idejne osnutke mehanizmov soglasja, a sta pomembni tudi točna implementacija mehanizma soglasja in implementacija verige blokov, v kateri algoritem soglasja operira. Glede na merila smo ugotovili, da se v idejni implementaciji najbolje izkaže dokaz sreče, a algoritem še nima svoje implementacije. Med bolje ocenjenimi algoritmi je dokaz pretečenega časa, ki smo ga izbrali za uporabo v svojem praktičnem prikazu. Nadaljevali smo s pregledom nekaj ogrodij, ki se razvijajo na področju verige blokov in obljubljajo visoko zmogljivost transakcij, modularnost in dobro razširljivost v primeru širjenja omrežja vozlišč. Izbrali smo ogrodje Sawtooth, ki uporablja dokaz pretečenega časa za mehanizem soglasja in rešili problem beleženja popravil in vzdrževanj vozila.

Dokaz pretečenega časa nam je v našem zaprtem okolju uspešno potrjeval transakcije z zapisi v verigo blokov. Poiskali smo tudi podoben projekt, implementiran na istem ogrodju, in preverili, ali bi ogrodje tudi v realnem okolju zadostilo našim potrebam. Svojo implementacijo in koncept uporabe le-te smo v nadaljevanju primerjali z obstoječimi sistemi servisnih knjižic.

V padalnem razvoju bi se lahko osredotočili na prenos projekta na fizična vozlišča, podkrepljena s pravo strojno opremo, in testirali zmožnosti verige blokov. Zanimiva posodobitev bi bila nato tudi implementacija drugih mehanizmov soglasja, predvsem mehanizma soglasja dokaza sreče. Pomemben tehnični iziv razvoja bi bila implementacija iskanja po vsebini, zapisani na verigi blokov.

Menimo, da imajo verige blokov velik potencial v sodobni družbi, a pomembno se bo zavedati, v katerih primerih bo smiselna uporaba verig blokov in v katerih primerih bo bolje uporabljati relacijske in nerelacijske podatkovne baze. Danes verige blokov s mehanizmi soglasja, kot jih poznamo, še niso dosegle zrelosti za splošno rabo, a verjamemo, da se s trenutnim zanimanjem za razvoj trenutek zrelosti in s tem splošne uporabe približuje.

Dodatek A

Programska koda

https://github.com/domenpetric/vehicleLogger

Literatura

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2008.
URL https://bitcoin.org/bitcoin.pdf

[2] A. Back, Hashcash - a denial of service counter-measure, 2002.
URL http://www.hashcash.org/papers/hashcash.pdf

[3] M. L.LAMPORT, R.SHOSTAK, The byzantine generals problem, 1982.
URL https://people.eecs.berkeley.edu/~luca/cs174/byzantin\
e.pdf

[4] B. L. M. Castro, Practical byzantine fault tolerance and proactive recovery, in: ACM Transactions on Computer Systems, 2002, p. 398–461.

[5] V. K. K. P. S. Dziembowski, S. Faust, Proofs of space, 2013.
URL https://eprint.iacr.org/2013/796.pdf

[6] S. N. S. King, Ppcoin: Peer-to-peer crypto-currency with proof-of-stake, 2012.
URL https://pdfs.semanticscholar.org/0db3/8d32069f3341d34\
c35085dc009a85ba13c13.pdf

[7] Sawtooth documentation, 2018.
URL https://sawtooth.hyperledger.org/docs/core/releases/latest/architecture/poet.html

[8] P4Titan, Slimcoin a peer-to-peer crypto-currency with proof-of-burn, 2014.
URL https://eprints.soton.ac.uk/415083/2/itasec18_main.pdf

[9] R. B. F. L. A. M. V. S. S. Angelis, L. Aniello, Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain, 2018.
URL https://eprints.soton.ac.uk/415083/2/itasec18_main.pdf

[10] Nem technical reference, 2018.
URL https://nem.io/wp-content/themes/nem/files/NEM_techRe\
f.pdf

[11] H. W. M. K. M. Milutinovic, W. He, Proof of luck: an efficient blockchain consensus protocol, 2016.
URL https://dl.acm.org/citation.cfm?id=3007790

[12] M. Vukolić, The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication, Springer International Publishing, 2016, pp. 112–125.
URL https://doi.org/10.1007/978-3-319-39028-4_9

[13] Gartner, 5 trends emerge in the gartner hype cycle for emerging technologies, 2018, 2018.
URL https://www.gartner.com/smarterwithgartner/5-trends-\
emerge-in-gartner-hype-cycle-for-emerging-technologies-2018/

[14] Bitcoin energy consumption index.
URL https://digiconomist.net/bitcoin-energy-consumption

[15] Berkeley, Appliad inovation revew, 2016.
URL http://scet.berkeley.edu/wp-content/uploads/AIR-2016\
-Blockchain.pdf

[16] N. Szabo, Smart contracts: Building blocks for digital markets, 1996.
URL http://www.fon.hum.uva.nl/rob/Courses/InformationIn\

Speech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html

[17] I. Bashir, Mastering blockchain, 2017.

[18] P. Jain, S. Desai, S. Kim, M.-W. Shih, J. Lee, C. Choi, Y. Shin, T. Kim, B. B. Kang, D. Han, OpenSGX: An Open Platform for SGX Research, in: Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, 2016.

[19] Burst technical inforamtion, 2018.
URL https://burstwiki.org/wiki/Main_Page

[20] F. Z. G. Brambilla, M. Amoretti, Using blockchain for peer to peer proof of location, 2015.
URL https://arxiv.org/pdf/1607.00174.pdf

[21] M. Swan, Blockchain blueprint for new economy, 2015.

[22] Z. Hintzman, Comparing blockchain implementations, 2017.
URL https://www.nctatechnicalpapers.com/Paper/2017/2017-comparing-blockchain-implementations/download

[23] Cypherium: A scalable and permissionless smart contract platform, 2018.
URL https://www.cypherium.io/landing/download/cypherium-whitepaper.pdf

[24] Registered vehicles data by country, 2015.
URL http://apps.who.int/gho/data/node.main.A995

[25] Sawtooth documentation, 2018.
URL https://sawtooth.hyperledger.org/docs/core/releases/1.0/introduction.html

[26] M. Miliard, Change healthcare's enterprise blockchain tech now available for hospitals, practices, payers, 2018.
URL https://www.healthcareitnews.com/news/change-health\
cares-enterprise-blockchain-tech-now-available-hospitals-p\
ractices-payers