UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA

STORITEV PORAZDELJENEGA SHRANJEVANJA ZASEBNIH DATOTEK

ANDRAŽ CUDERMAN

2020
ZAKLJUČNA NALOGA
CUDERMAN

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Storitev porazdeljenega shranjevanja zasebnih datotek

(Distributed storage service for saving private files)

Ime in priimek: Andraž Cuderman
Študijski program: Računalništvo in informatika
Mentor: prof. dr. Cene Bavec

Koper, julij 2020

Ključna dokumentacijska informacija

Ime in PRIIMEK: Andraž CUDERMAN

Naslov zaključne naloge: Storitev porazdeljenega shranjevanja zasebnih datotek

Kraj: Koper

Leto: 2020

Število listov: 54 Število slik: 18 Število referenc: 43

Mentor: prof. dr. Cene Bavec

Ključne besede: Tehnologija veriženja blokov, porazdeljeni sistemi, Bitcoin, Ethereum, IPFS, kriptografija, porazdeljeno shranjevanje datotek

Izvleček:

Hranjenje datotek preko storitev v oblaku predstavlja nov standard shranjevanja podatkov, saj so datoteke, ki so shranjene na oblaku, dostopne na vseh napravah z dostopom do internetnega omrežja. Obenem so datoteke shranjene na več diskih in posledično zavarovane pred izgubo podatkov. Vendar pa je večina spletnih storitev grajena na arhitekturi odjemalec-strežnik, ki med drugim vsebuje težave s cenzuro, eno samo točko odpovedi ter težave z nadzorom nad zasebnostjo in integriteto datotek. Cilj zaključne naloge je raziskati porazdeljene tehnologije in s pomočjo le-teh rešiti naštete probleme centraliziranega shranjevanja podatkov, obenem pa zagotoviti uporabniku nadzor nad podatki preko kriptografskih tehnik, ki zagotavljajo tako integriteto podatkov kot tudi zasebnost in možnost upravljanja s podatki. V zaključni nalogi je uporabljena tehnologija veriženja blokov, in sicer Ethereum, poleg tega pa še omrežje IPFS, ki omogoča porazdeljeno shranjevanje statičnih datotek. Ustvarjalci sistema IPFS menijo, da bo IPFS v povezavi s tehnologijo veriženja blokov predstavljal prihajajoči standard za porazdeljeno arhitekturo interneta, ki bo grajen predvsem na transparentnosti, s čimer bo uporabnikom vlival zaupanje v aplikacije in jim obenem omogočil nadzor nad zasebnostjo podatkov.

Key document information

Name and SURNAME: Andraž CUDERMAN

Title of the final project paper: Distributed storage service for saving private files

Place: Koper

Year: 2020

Number of pages: 54 Number of figures: 18 Number of references: 43

Mentor: Prof. Cene Bavec, Phd

Keywords: Blockchain, distributed systems, Bitcoin, Ethereum, IPFS, cryptography, distributed file storage

Abstract:

File storage through cloud services represents a new standard for data storage, as files stored in the cloud are accessible on all devices with Internet access. At the same time the files are stored on multiple disks and consequently protected against data loss. However, most online services are built on a client-server architecture, which includes, among other things, censorship issues, a single point of failure, privacy and file integrity control issues. The aim of the paper is to explore distributed technologies and use them to solve the listed problems of centralized data storage while providing the user with control over data through cryptographic techniques that ensure data integrity, privacy and personal data management capabilities. The prototype of distributed storage service uses blockchain technology, namely Ethereum, as well as the IPFS network, which allows distributed storage of static files. The IPFS creators believe that IPFS, in conjunction with blockchain technology, will be an upcoming standard for distributed Internet architecture, built primarily on transparency, giving users confidence in applications while still allowing them to have control over personal data.

ZAHVALA

Zahvaljujem se mentorju dr. Cenetu Bavcu za pomoč in usmerjanje pri izdelavi zaključne naloge. Posebno zahvalo za vso podporo, ki sem jo dobil skozi študijska leta, si zasluži tudi družina.

KAZALO VSEBINE

KAZALO SLIK IN GRAFIKONOV

SEZNAM KRATIC

API Application Programming Interface (aplikacijski programski vmesnik)

AWS Amazon Web Services (Amazonove spletne storitve)

BTC Bitcoin

DHT Distributed Hash Table (porazdeljena razpršena tabela)

ECC Eliptic-curve Cryptography (kriptografija eliptične krivulje)

ETH Ethereum

GHOST Greedy Heaviest Observed Subtree (pohlepno najtežje opaženo poddrevo)

IPFS InterPlanetary File System (medplanetarni datotečni sistem)

IPLD InterPlanetary Linked Data (medplanetarno povezani podatki)

P2P Peer-to-peer (enak z enakim)

SPV Simple Payment Verification (preprosto preverjanje plačila)

UTXO Unspent transaction output (nezapravljen izhod transakcije)

1 UVOD

V zadnjih letih sta v ospredje stopili dve zelo perspektivni tehnologiji v svetu računalništva. Ena izmed njih je računalništvo v oblaku, ki ga dandanes uporablja že praktično vsak uporabnik interneta. Računalništvo v oblaku je tehnologija pri kateri so računalniški viri dostopni preko storitev na spletu. Bistvena prednost računalništva v oblaku je, da so storitve na oblaku dostopne na katerikoli napravi in na katerikoli lokaciji pod pogojem, da ima naprava dostop do interneta. Obenem uporabniki z uporabo storitev v oblaku postanejo neodvisni od strojne opreme, saj preko svojega brskalnika dostopajo do strežnikov. Prednost računalništva v oblaku je tudi zaščita pred izgubo podatkov, saj so podatki shranjeni na večjem številu diskov [8]. Druga tehnologija, ki je v zadnjih letih pridobila na prepoznavnosti, pa je tehnologija veriženja blokov. Tehnologija veriženja blokov je v zadnjih letih postala prepoznavna predvsem zaradi kripto valut. Bistvena prednost te tehnologije je ta, da je arhitektura sistema porazdeljena, kar pomeni, da v sistem ni vključena nobena tretja oseba, ki bi ji morali zaupati. Zaupanje je ključno za storitve kot so denarne transakcije, e-volitve in varnost podatkov [37]. Tehnologija veriženja blokov naj bi bila revolucionarna tehnologija, ki bo v prihodnosti predstavljala standard za spletne storitve.

Namen zaključne naloge je združiti omenjeni tehnologiji in narediti prototip popolnoma porazdeljenega sistema za shranjevanje datotek. Ideja sistema je, da bodo vse datoteke, spletna stran, ki bo omogočala nalaganje dokumentov, in podatkovna baza porazdeljene po več napravah, ki so povezane preko P2P omrežja. Tako bo delovanje storitve popolnoma neodvisno od tretje osebe, storitev pa bo dosegljiva v vsakem trenutku, tudi v primeru izpada kateregakoli elementa sistema. Tehnologija veriženja blokov je ena izmed uporabljenih P2P tehnologij, ki sicer ne omogoča shranjevanja velikih datotek, zato bo poleg tehnologije veriženja blokov uporabljena tudi druga vrsta P2P omrežja, ki omogoča shranjevanje in hiter dostop do datotek.

Tehnologija veriženja blokov, ki sem jo uporabil v prototipu, je tehnologija Etherum, ki je ena izmed najbolj znanih in uporabljenih tehnologij veriženja blokov. Za shranjevanje datotek pa bo skrbel odprtokoden P2P sistem IPFS, ki omogoča porazdeljeno shranjevanje in dostop do datotek. Ena izmed težav v P2P omrežju je, da le-ta ne omogoča nastavljanja pravic dostopa do virov kot so datoteke, zato lahko do njih dostopa vsak udeleženec v omrežju. Dostop do datotek se lahko omeji s posebnimi kriptografskimi tehnikami, ki omogočajo šifriranje datotek na način, da jih lahko dešifrira samo uporabnik.

V tem dokumentu bo najprej opisana tehnologija veriženja blokov in že prej omenjene tehnologije, ki bodo prav tako uporabljene v prototipu. V naslednjih segmentih bo

vsebinsko predstavljen problem, zasnova sistema ter prototip razvite aplikacije. Na koncu bom projekt še ovrednotil in predlagal možne izboljšave za razvito aplikacijo.

2 TEHNOLOGIJA VERIŽENJA BLOKOV

Tehnologija veriženja blokov (angl. *blockchain*), je lahko enostavno interpretirana kot porazdeljena baza podatkov, ki jo kontrolira skupina samostojnih računalnikov. Njihov glavni namen je ohraniti avtoriteto podatkov, kar jim uspe preko soglasja med vsemi vozlišči v porazdeljenem omrežju [37].

2.1 Kaj prinaša tehnologija veriženja blokov?

Če želimo določiti pravo uporabnost tehnologije veriženja blokov, moramo najprej pogledati trenutno stanje storitev, ki so uporabljene v vsakdanjem življenju. Ko kupujemo preko spleta, moramo podatke svoje kartice zaupati tretji osebi kot je na primer banka ali podjetje, s tem pa jim hkrati zaupamo, da bodo iz našega računa vzeli pravo vsoto denarja. Enako je zaupanje pomembno tudi pri plačevanju z gotovino, saj mora prodajalec kupcu zaupati, da je bankovec resničen, obenem pa mora tudi kupec zaupati prodajalcu, da mu je ta vrnil resničen bankovec, ki ga bo lahko uporabil tudi, če nekaj želi kupiti v drugi trgovini. Prav tako prodajalec verjame, da lahko ta bankovec odnese na banko in ga spremeni v digitalni denar, ki ga lahko uporabi pri plačevanju preko spleta oziroma pri plačevanju s kartico. Ljudje moramo stalno zaupati, da tretje osebe oziroma ustanove hranijo evidence naših transakcij in trenutnih stanj v svoji bazi podatkov, obenem pa tudi, da so evidence točne in nespremenjene [24].

Zaupanje v institucije pa ne obstaja zgolj na finančnem nivoju. Primer zaupanja v institucije je tudi knjižnica, ki v svoji centralni bazi podatkov hrani vse knjige, ki jih ima, prav tako pa hrani tudi podatke o svojih članih in vseh knjigah, ki so si jih kadarkoli sposodili. Člani knjižnic morajo zato zaupati knjižnici, da ne bodo delile njihovih podatkov z drugimi posamezniki ali organizacijami. Primer knjižnice je na videz neškodljiv, vendar pa bi se lahko zgodilo, da bi vlada pridobila dostop do teh podatkov in preverjala, katere knjige si člani knjižnice izposojajo. Posameznika, ki bi si izposodil knjigo, ki je vlada ne odobrava, bi lahko označili kot sumljivega in pod drobnogled vzeli njegovo vedenje, lahko pa bi to vodilo tudi v obsežnejšo osebno preiskavo ali aretacijo [24].

V nekaterih državah, kjer je prisotno pomanjkanje zaupanja v podjetja in vlado, je vsaka transakcija tvegana. Če ljudje naložijo denar na banko, so v nevarnosti, da banke njihov denar vzamejo, zato so primorani ob opravljanju velikih transakcij, ki so na primer potrebne za nakup hiše, denar shranjevati v bankovcih, zlatu ali srebru. Tudi, ko posamezniku uspe privarčevati denar za hišo, še vedno tvega, da mu bo prodajalec ukradel denar. Če računi niso izdani, ob gotovinskem poslovanju ne obstaja noben legalni ali

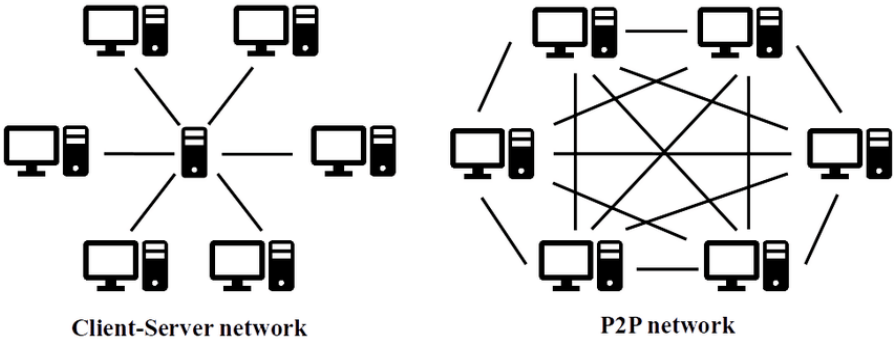
pravni sistem, ki bi dokazal posameznikov nakup, zato je skoraj nemogoče dokazati, da je posameznik hišo dejansko kupil [24].

Tehnologija veriženja blokov je lahko uporabljena tudi za vrsto drugih aplikacij. Ena izmed njih je na primer oskrbovalna veriga za prodajo produktov. Pogodbe in dogodki so lahko dogovorjeni preko podatkov shranjenih na verigo blokov, kar nudi informacije o poreklu blaga in izboljša vidnost logistike ter kakovosti dobavne verige. Obenem pa so lahko dogodki v oskrbovalni verigi povezani z avtomatskimi plačili. Možna uporaba tehnologije veriženja blokov je tudi v shranjevanju dokazov o digitalnih podatkih. Vsi podatki, ki so zapisani na verigo blokov, so nespremenljivi, zato lahko na verigo blokov s pomočjo različnih kriptografskih tehnik shranjujemo dokaze o integriteti dokumentov [43].

Centralizirane baze podatkov in institucije delujejo dobro, ko je v sistemu prisotno zaupanje v pravo, vlado in ljudi. Kljub izpolnitvi omenjenih pogojev pa še vedno lahko pride do izdaje zaupanja, kar lahko povzroči izgubo denarja [24]. Tehnologija veriženja blokov omogoča ustvarjanje aplikacij, ki se izvajajo porazdeljeno, obenem pa ohranjajo soglasje v obliki porazdeljene baze podatkov. Vsi udeleženci v omrežju lahko vidijo in preverijo podatke transakcij, kar nudi transparentnost in ljudem daje zaupanje. Obenem pa tehnologija veriženja blokov ponuja anonimnost, kar prav tako predstavlja veliko prednost te tehnologije [37].

2.2 Peer-to-peer (P2P)

Za razumevanje tehnologije veriženja blokov je najprej potrebno poznavanje modela, na katerem tehnologija sloni. Peer-to-peer (P2P) je tehnologija, ki se je v javnosti prvič pojavila leta 1979, in sicer v sistemu USENET, ki predstavlja storitev, preko katere si lahko uporabniki izmenjujejo novice. Izraz P2P se nanaša na računalnike oziroma naprave, ki so med seboj povezane in so si enakovredne, kar omogoča neposredno interakcijo med napravami. Interakcija poteka brez posrednika, ki je sicer potreben v tradicionalnem modelu odjemalec-strežnik. Arhitektura P2P omrežja omogoča predvsem sodelovanje naprav z namenom izvedbe določene naloge. Naprave naloge izvedejo z združevanjem virov kot so diski za shranjevanje podatkov ali procesorska moč za opravljanje računskih operacij [33].



Slika 1: Arhitektura omrežja odjemalec-strežnik in arhitektura P2P omrežja[1].

P2P sistem mora biti sposoben združiti vire in podatke iz posameznih vozlišč, da lahko opravi zadano nalogo. Da je združevanje virov izvedljivo, mora vsako vozlišče shranjevati nekaj metapodatkov, ki olajšajo iskanje po omrežju. Metapodatki so informacije o podatkih, ki jih trenutno vozlišče shranjuje, obenem pa vključujejo tudi podatke o drugih vozliščih. Metapodatki omogočajo usmerjanje poizvedbe po omrežju do vozlišča, ki shranjuje zahtevan del informacije. Iskana informacija je lahko shranjena na enem vozlišču, lahko pa je razpršena med več vozlišči. Ko so vozlišča, ki shranjujejo iskane dele datotek, identificirana, lahko vozlišče, ki je podatke zahtevalo, direktno komunicira z njimi, da dobi želene podatke [33].

Ena izmed glavnih značilnosti, obenem pa tudi prednosti P2P omrežja, ki to omrežje ločuje od modela odjemalec-strežnik, je simetrična vloga vsakega vozlišča. Vsako vozlišče hkrati opravlja nalogo strežnika in odjemalca. Simetrična vloga pomeni, da lahko vsako vozlišče odda poizvedbo po omrežju, kot bi to sicer storil odjemalec, oziroma ponudi storitev, kot to sicer počne strežnik. Druga velika prednost P2P omrežja je možnost širjenja omrežja, ker omogoča povezavo več tisoč vozlišč. Rezultat tega je možnost izkoriščanja velikega dela računske moči in izkoriščanja prostora na diskih, ki so na voljo na vozliščih v omrežju [33].

Trenutno verjetno najbolj uporabljeno in znano P2P omrežje je omrežje BitTorrent. Namen omrežja BitTorrent je prenos (velikih) datotek, deljenih s strani gruče vozlišč v omrežju, na katere se lahko poveže kdorkoli. Povezava na več vozlišč omogoča prejemanje podatkov iz več virov naenkrat, kar omogoča večjo hitrost prenosa podatkov. BitTorrent lahko tako zamenja velike strežnike z navadnimi (hišnimi) računalniki [29].

[1] Vir: https://www.researchgate.net/figure/Client-server-and-P2P-network-models_fig2_333160118, datum dostopa: 30.4.2020

Uporaba P2P omrežij sicer ni zelo razširjena, saj je P2P omrežje v primerjavi s sistemi, ki slonijo na omrežni arhitekturi odjemalec-strežnik, mnogo težje zavarovati. Težave pri varnosti nastanejo predvsem zaradi odprtosti P2P omrežja. Obstaja več nevarnosti, saj se lahko v omrežju pojavijo hudoželjna vozlišča. Zaradi samostojnosti vozlišč vključenih v omrežje, lahko nekatera vozlišča škodujejo omrežju tako, da ne upoštevajo samega protokola delovanja. Primer takšnega škodovanja bi bila situacija, kjer bi eno vozlišče zahteve poizvedb po omrežju preusmerilo na nepravilen način, kar lahko povzroči, da zahtevana poizvedba ne pride do želenega vozlišča in s tem odjemalec ne dobi želenih podatkov. Možno bi bilo tudi, da eno vozlišče zavrača shranjevanje podatkov, spet drugo pa lahko zavrača serviranje poizvedb, čeprav so podatki, ki ustrezajo poizvedbi, na voljo na tem vozlišču. Prav tako bi lahko vozlišča z nenehnim pristopanjem in izstopanjem naredila omrežje nestabilno in nezaupljivo. Vsako vozlišče pa lahko tudi spreminja in briše datoteke. Vozlišče lahko drugim vozliščem pošilja tudi nepotrebne poizvedbe in tako zapolni celotno pasovno širino, ki jo premore vozlišče, kar povzroči, da vozlišča ne morejo servirati pravim poizvedbam, vendar servirajo le vozlišču, ki pošilja nepotrebne zahteve. To je le nekaj od problemov varnosti omrežja, ki pa jo vsaka tehnologija rešuje drugače [33]. Ena izmed tehnologij, ki se zelo dobro spopada z naštetimi problemi, je tehnologija veriženja blokov.

2.3 Kako deluje tehnologija veriženja blokov?

V tem poglavju bo opisano predvsem, kako deluje jedro tehnologije veriženja blokov v sistemu Bitcoin, saj je to prva in najbolj osnovna implementacija, ki predstavlja temelje za večino drugih tehnologij veriženja blokov [24]. V nadaljevanju bo podrobneje predstavljeno delovanje tehnologije Ethereum, ki je bila uporabljena pri izdelavi prototipa.

2.3.1 Prstni odtisi

Za razumevanje tehnologije veriženja blokov je potrebno poznati kriptografsko tehniko prstnih odtisov (ang. hash). Prstni odtisi so podkategorija kriptografije, ki ima specifične lastnosti, ki pripomorejo k varnosti platform kot je tehnologija veriženja blokov. Ideja prstnega odtisa je zgostiti vsebino zapisa v krajši zapis na način, da iz zgoščenega zapisa ni mogoče dobiti prvotnega zapisa. Vsak zgoščen zapis je praktično unikaten, saj v teoriji obstaja zelo majhna možnost, da bi se dva različna zapisa zgostila v enakega. Hkrati je vsak prstni odtis brez vsakega vzorca, preko katerega bi lahko ugotovili, kakšen je prvotni zapis. Prstni odtis ima tudi predpisano dolžino, ki je fiksna za vsako zgoščevanje, vendar pa se dolžina med različnimi algoritmi razlikuje [7]. Ker v prstnih odtisih ni nobenega vzorca, preko katerega bi bilo možno ugotoviti, kakšen je prvotni zapis, je lahko prvotni zapis identificiran zgolj z zgostitvijo vsakega potencialnega originalnega zapisa. Če

predpostavljamo, da obstaja neskončno možnosti za originalni zapis, lahko vidimo, da je iz prstnega odtisa praktično nemogoče dobiti originalni zapis. Kombinacija lastnosti prstnega odtisa omogoča zavarovanje aplikacij z digitalnimi podpisi, preverjanjem integritete podatkov (zaznavanje napak pri prenosu), dokazom dela, avtentikacijo (zgoščevanje gesel), unikatnimi identifikatorji in drugimi načini. Obstaja več algoritmov za generiranje prstnih odtisov, in sicer Ethereum uporablja algoritem Keccak-256, Bitcoin pa algoritem SHA-256 [2].

2.3.2 Transakcije

Transakcije so digitalno podpisana sporočila, ki so zapisana v blokih in so poleg nekaterih metapodatkov, ki so vključeni v glavo bloka, edina, ki lahko spremenijo stanje podatkov na verigi. Transakcije omogočajo pridobivanje informacij o trenutnem stanju na verigi blokov. V tem poglavju bodo predstavljene transakcije, ki potekajo v omrežju Bitcoin, saj uporabljajo drugačen model kot sistem Ethereum. Transakcije v omrežju Bitcoin uporabljajo model ne zapravljenih transakcijskih izhodov (angl. *unspent transaction output* - UTXO), medtem ko Ethereum uporablja računski model (angl. *account model*) [2].

Za ustvarjanje transakcije je potrebna denarnica (angl. *wallet*), ki predstavlja program, ki skrbi za uporabniški račun in omogoča pošiljanje kovancev v obliki valute omrežja. Tehnologija Bitcoin uporablja valuto Bitcoin ali krajše BTC. Vsak račun je zavarovan s strani asimetrične kriptografije, ki omogoča varnost transakcij v omrežju [21].

2.3.2.1 Asimetrična kriptografija

Asimetrična kriptografija predstavlja kriptografsko tehniko, ki za šifriranje in dešifriranje podatkov uporabi različne ključe. Za asimetrično kriptografijo, sta potrebna javni in zasebni ključ, ki sta matematično povezana. Javni ključ je javen in je za namene šifriranja sporočila lahko uporabljen s strani kogarkoli, sporočilo pa je lahko dešifrirano le s pripadajočim zasebnim ključem. Asimetrična kriptografija pa deluje tudi v drugi smeri, in sicer je lahko v primeru šifriranja sporočila z zasebnim ključem sporočilo dešifrirano z javnim ključem. Dešifriranje z javnim ključem omogoča preverjanje pristnosti pošiljatelja, saj ima zgolj pošiljatelj dostop do zasebnega ključa [26].

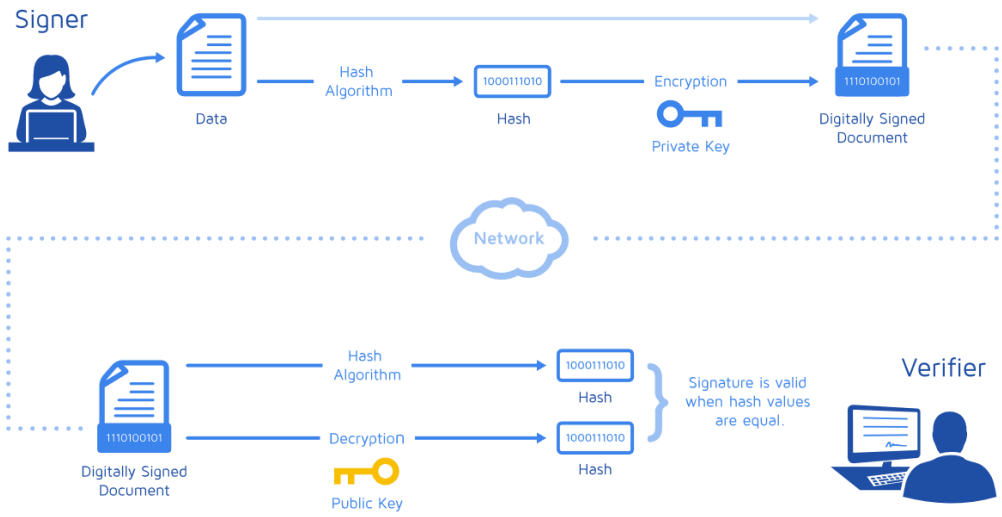
Asimetrična kriptografija je skupaj s tehniko prstnih odtisov tipično uporabljena za digitalne podpise. Digitalni podpis je tehnika preverjanja pristnosti pošiljatelja in integritete podatkov ter hkrati tudi tehnika, ki ovrže možnost, da ustvarjalec digitalnega podpisa zanika svoje dejanje. Hkrati je digitalni podpis enakovredna različica ročnega

podpisa. Asimetrična kriptografija je z namenom digitalnega podpisa uporabljena tudi v tehnologijah veriženja blokov [26].

2.3.2.2 Digitalni podpis

Digitalni podpisi uporabljajo asimetrični par ključev in zgoščevalne funkcije za generiranje prstnega odtisa. Uporabljajo se v primeru, ko želi pošiljatelj prejemniku poslati datoteko, ne želi pa, da kdorkoli to sporočilo med prenosom spremeni, torej želi ohraniti integriteto podatkov, hkrati pa želi tudi dokazati pristnost pošiljatelja, torej dokazati, da je res on pošiljatelj te datoteke. Digitalni podpis ustvari pošiljatelj, preveri pa ga prejemnik. Uporaba digitalnega podpisa je sestavljena iz petih korakov, ki so navedeni v nadaljevanju:

1. Pošiljatelj izračuna prstni odtis datoteke.
2. Pošiljatelj šifrira prstni odtis datoteke z njegovim zasebnim ključem in s tem ustvari digitalni podpis.
3. Datoteka in digitalni podpis sta nato združena in poslana skupaj.
4. Ko prejemnik prejme sporočilo, iz sporočila izvleče digitalni podpis in ga dešifrira s pošiljateljevim javnim ključem. Če je dešifriranje uspešno, to pomeni, da je bila pristnost pošiljatelja overjena.
5. V zadnjem koraku prejemnik izračuna prstni odtis datoteke, in ga primerja s tistim, ki ga je dešifriral v 4. koraku. Če se prstna odtisa ujemata, je s tem overjena integriteta podatkov [9].



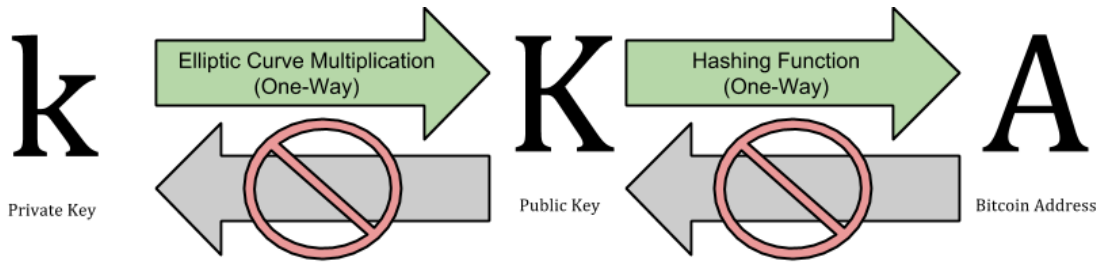
Slika 2: Postopek ustvarjanja in preverjanja prstnega odtisa[2].

[2] Vir: https://medium.com/@meruja/digital-signature-generation-75cc63b7e1b4, datum dostopa: 30.4.2020

2.3.2.3 Račun in kriptografski ključi

Vsak uporabnik pri kreaciji računa dobi zasebni ključ, ki ni shranjen nikjer v omrežju, ampak je skrb zanj v rokah uporabnika. Če uporabnik izgubi zasebni ključ, s tem izgubi tudi vse kovance vezane na ta zasebni ključ [34]. Poleg zasebnega ključa uporabnik uporablja tudi javni ključ ter naslov, ki predstavlja prstni odtis javnega ključa. Naslov se uporablja pri kreiranju transakcij, kjer pošiljatelj pove naslov prejemnika. Naslov je javen v vsakem trenutku. Medtem ko je zasebni ključ znan samo uporabniku, se javni ključ razkrije po vsaki transakciji [36].

Oba sistema, Bitcoin in Ethereum, za generiranje asimetričnega para ključev uporabljata ključe, ki so generirani pri algoritmu kriptografije eliptične krivulje (angl. *eliptic curve cryptography* - ECC). ECC algoritem omogoča generiranje javnega ključa iz zasebnega ključa. Proces generiranja ključa poteka zgolj v eni smeri, kar pomeni, da iz javnega ključa ne moremo dobiti zasebnega [2]. Naslov računa predstavlja prstni odtis javnega ključa in se uporablja predvsem zaradi varnosti zasebnega ključa, saj v primeru, ko se najde luknja v kriptografiji eliptične krivulje, iz naslova ne bo možno takoj dobiti zasebnega ključa, ker je javni ključ neznan do prve transakcije. Nekatere denarnice, ki skrbijo za račune v omrežju, naredijo nov zasebni ključ po vsaki opravljeni transakciji [36].



Slika 3: Generiranje javnega ključa in naslova iz zasebnega ključa[3].

2.3.2.4 Delovanje transakcij

Sistem Bitcoin je namenjen zgolj prenosu kovancev (Bitcoin-ov), zato definicija transakcij znotraj sistema Bitcoin opisuje zgolj interakcijo, v kateri transakcija pove omrežju, da je lastnik kovancev odobril prenos kovancev drugemu lastniku. Nov lastnik lahko te kovance pošlje drugemu računu, s čimer se ustvari veriga transakcij [3].

Vsaka transakcija je sestavljena iz vhoda (angl. *input*) in izhoda (angl. *output*). Vhodi so zapisi, ki računu predstavljajo dolg oziroma strošek, izhodi pa v kontekstu naslova računa

[3] Vir: https://blockgeni.com/bitcoin-keys/ datum dostopa: 30.4.2020

delujejo kot dobroimetje. Izjema med transakcijami je tako imenovana Coinbase transakcija, ki predstavlja prvo transakcijo v bloku in ne vključuje nobenega vhoda temveč zgolj izhode. Coinbase transakcija predstavlja nagrado rudarju za uspešno rudarjen blok, ki se doda v verigo blokov. Z dodajanjem bloka v verigo rudar poskrbi za varnost omrežja [38].

Vsaka transakcija ima lahko več vhodov oziroma izhodov. Če en vhod ne vsebuje dovolj kovancev, se lahko skupaj združi več vhodov, katerih seštevek bo predstavljal zadostno število kovancev, ki so potrebni za transakcijo. V primeru, ko je vsota vhodov prevelika, se ustvari dodatna izhodna transakcija, ki nakaže ostanek kovancev nazaj na pošiljateljev naslov. Na sliki 4 je prikazana transakcija, ki ima en vhod in en izhod [38].

```
Input:
Previous tx: f5d8ee39a430901c91a5917b9f2dc19d6d1a0e9cea205b009ca73dd04470b9a6
Index: 0
scriptSig: 304502206e21798a42fae0e854281abd38bacd1aeed3ee3738d9e1446618c4571d10
90db022100e2ac980643b0b82c0e88ffdfec6b64e3e6ba35e7ba5fdd7d5d6cc8d25c6b241501

Output:
Value: 5000000000
scriptPubKey: OP_DUP OP_HASH160 404371705fa9bd789a2fcd52d2c580b65d35549d
OP_EQUALVERIFY OP_CHECKSIG
```

Slika 4: Primer transakcije z enim vhodom in enim izhodom[4].

Vhodna transakcija v zgornjem primeru (slika 4) je sestavljena iz parametra *prejšna transakcija* (previous tx), ki predstavlja prstni odtis uporabljenega izhoda prejšnje transakcije, *indeks* (index) pa predstavlja specifičen izhod prejšnje transakcije. *Skriptni podpis* (scriptSig) je sestavljen iz dveh komponent, in sicer iz digitalnega podpisa in javnega ključa, kjer se mora prstni odtis javnega ključa ujemati s prstnim odtisom navedenim v uporabljenem izhodu transakcije, v polju *skriptni javni ključ* (ScriptPubKey). Javni ključ je uporabljen za overitev pristojnosti digitalnega podpisa [38].

```
scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
scriptSig: <sig> <pubKey>
```

Slika 5: Elementi skriptnega javnega ključa in skriptnega podpisa[4].

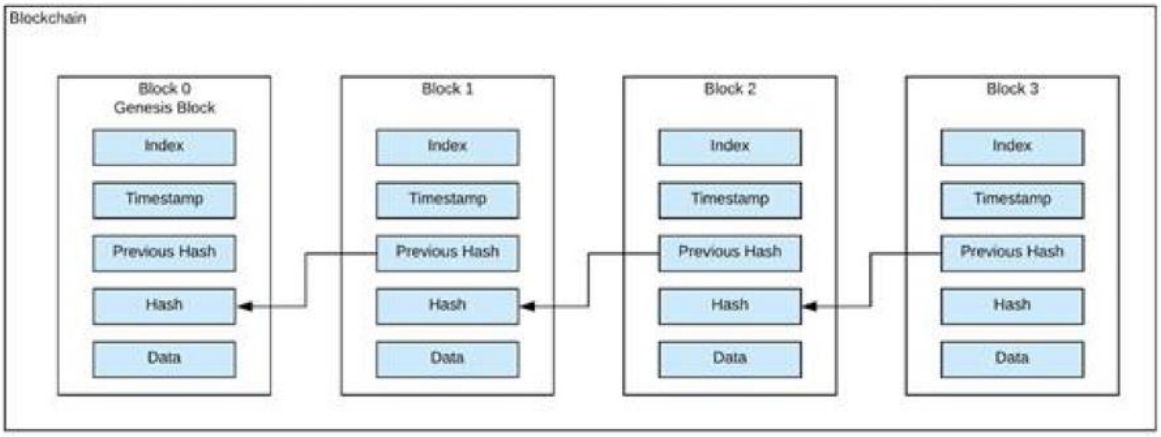
Izhod vsebuje navodila za nadaljnje pošiljanje kovancev med računi. Vsebuje vrednost (angl. value), ki ponazarja število poslanih Satoshijev, pri čemer Satoshiji predstavljajo manjšo enoto Bitcoina (1 Bitcoin = 1000,000,000 Satoshijev). *Skriptni javni ključ* je

[4] Vir: https://en.bitcoin.it/wiki/Transaction, datum dostopa: 30.4.2020

skripta, ki vsebuje pravila, ki morajo biti izpolnjena, da lahko prejemnik zapravi prejete kovance. V primeru transakcije na sliki 4 mora skriptni podpis imeti pravilen digitalni podpis in javni ključ, pri čemer je prstni odtis javnega ključa enak tistemu, ki je naveden v polju *skriptni javni ključ* [5].

2.3.3 Struktura in povezava blokov

Podatkovna struktura tehnologije veriženja blokov predstavlja enostaven seznam, kjer vsak element v seznamu predstavlja svoj blok. Vsak blok v verigi blokov je identificiran s pomočjo prstnega odtisa glave bloka. Poleg svojega prstnega odtisa pa vsebuje tudi prstni odtis prejšnjega bloka, ki predstavlja kazalec na prejšnji blok. Ureditev seznama nas pripelje do verige blokov, ki poteka od prvega bloka, ki je bil dodan na verigo, do zadnjega. Prvi blok v verigi je poimenovan *genesis*, kar v prevodu pomeni izvor [3].



Slika 6: Osnovna struktura verige blokov[5].

Vsak blok je sestavljen iz glave, ki vsebuje metapodatke, poleg glave pa vsebuje tudi dolg seznam transakcij. Število transakcij v bloku ni točno določeno, vendar se število v omrežju Bitcoin običajno giblje okoli 500 transakcij na blok, kar pomeni približno 125mb podatkov transakcij na blok. Velikost glave je omejena na 80 bajtov [3].

Glava bloka v sistemu Bitcoin sestoji iz treh delov in je v veliki meri podobna tudi pri sistemu Ethereum. Prvi del predstavlja kazalec na prejšnji blok preko *prstnega odtisa prejšnjega bloka*. Drugi del je sestavljen iz *težavnosti* (ang. *difficulty*), *časovnega žiga* (ang. *timestamp*) in *uganjenega števila* (angl. *nonce*), ki predstavlja naključno število, ki ga iščejo rudarji. Drugi del metapodatkov se uporablja pri rudarjenju (ang. *mining*). Tretji del

[5] Vir: https://www.spheregen.com/blockchain-technology-basics/, datum dostopa: 30.4.2020

podatkov zajema koren Merklovega drevesa, ki predstavlja podatkovno strukturo, ki je uporabljena predvsem za učinkovit povzetek transakcij v bloku [3].
Vsak blok je enolično določen z unikatnim prstnim odtisom bloka, ki predstavlja rezultat zgoščevanja glave bloka. Prstni odtis glave bloka ni shranjen v samem bloku. Namesto shranjevanja prstnega odtisa v sam blok, prstni odtis glave bloka izračuna vsako vozlišče posebej, in sicer takrat, ko je blok prejet iz omrežja. Prstni odtis bloka vozlišča običajno shranijo v posebno bazo podatkov, kjer shranjujejo metapodatke za bloke. Shranjevanje metapodatkov v posebno bazo podatkov zaradi možnosti indeksiranja baze podatkov omogoča bolj efektivno iskanje blokov [3].

Vozlišča, ki skrbijo za svojo verigo blokov, se imenujejo polna vozlišča. Ko polna vozlišča prejmejo nov blok s strani omrežja, vozlišče najprej preveri, če je blok veljaven, nato pa ga, če se izkaže kot veljaven, pripne na konec verige [15].

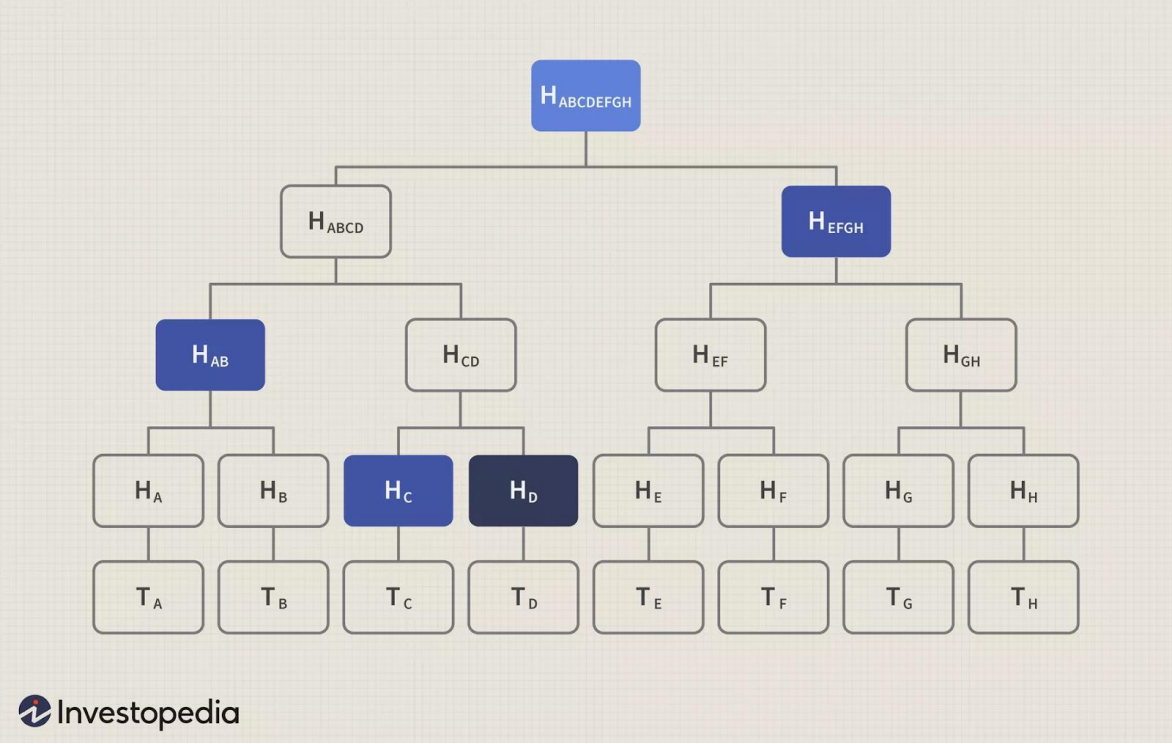
2.3.3.1 Merklovo drevo

Vsak blok v verigi vsebuje koren Merklovega drevesa, ki je sestavljen iz vseh transakcij v bloku. Podatkovna struktura Merklovo drevo je uporabljena predvsem za učinkovito shranjevanje in preverjanje integritete velike količine podatkov. Merklovo drevo je binarno in poda prstni odtis vseh transakcij v bloku. Narejeno je z rekurzivnim zgoščevanjem parov vozlišč, dokler ne dobimo zgolj enega vozlišča. Dobljeno vozlišče se imenuje koren Merklovega drevesa. Liste drevesa predstavljajo prstni odtisi transakcij. [18].

Merklovo drevo potrebuje sodo število listov. Če je število transakcij v bloku liho, se tista transakcija, ki v drevesu nima para, podvoji, njen prstni odtis pa je torej zgoščen sam s sabo. Višina Merklovega drevesa je vedno $\log_2(N)$, kjer N predstavlja število vseh transakcij. Merklovo drevo omogoči, da vozlišče lahko potrdi obstoj transakcije znotraj bloka z zgolj $\log_2(N)$ bitov informacije, ki vodijo od korena drevesa do lista transakcije. Poti od korena drevesa do prstnega odtisa transakcije pravimo tudi avtentikacijska ali Merklova pot, ki povezuje specifično transakcijo s korenom drevesa. Majhna višina drevesa oziroma kratka Merklova pot je predvsem pomembna, ko število transakcij v bloku naraste, saj se logaritem z osnovo dva povečuje precej počasneje kot sam N, kar je pomembno predvsem pri učinkovitem preverjanju ali so transakcije vključene v blok [18].

Vozlišče lahko z Merklovo potjo preveri, ali je transakcija vključena v blok. Na sliki 7 je prikazan primer Merklovega drevesa. Spodnja vrstica na sliki ponazarja transakcije, ki so označene z vrednostmi od T_a do T_h, zgornje vrstice pa predstavljajo prstne odtise, ki gradijo Merklovo drevo. Na sliki 7 je prikazan primer preverjanja, ali se transakcija T_d nahaja v Merklovem drevesu. Za izračunanje korena Merklovega drevesa in s tem dokaza,

da se transakcija Td nahaja v Merklovem drevesu, so potrebni le štirje prstni odtisi, in sicer prstni odtisi vozlišč $H_{ABCDEFGH}$, H_{EFGH}, H_{AB} in H_C, ki so obarvana z modro barvo. Ostala vozlišča na poti so lahko izračunana z zgoščevanjem znanih parov vozlišč (npr. vozlišče H_{CD} predstavlja prstni odtis para vozlišč H_C in H_D) [18]. Proces računanja neznanih prstnih odtisov je zelo hiter, saj je zgoščevanje operacija, ki se izvede v linearnem času [2].



Slika 7: Merklovo drevo[6].

Merklova drevesa so predvsem uporabna v vozliščih, ki niso polna in torej ne vsebujejo celotne verige blokov, temveč vključujejo zgolj glavo bloka. Če ne-polna vozlišča želijo preveriti, ali se določena transakcija nahaja v bloku, to storijo precej hitreje, kot če bi zahtevala celoten blok, saj od polnih vozlišč ne potrebujejo zahtevati vseh transakcij. Namesto celotnega bloka lahko od polnih vozlišč prenesejo zgolj potrebne prstne odtise. Količina prenesene informacije je v tem primeru precej manjša, sam prenos pa hitrejši, kot bi bil prenos vseh transakcij. V omrežju Bitcoin vozlišča za preverjanje plačila (angl. *simple payment verification - SPV*) predstavljajo primer ne-polnih vozlišč. Vozlišča SPV se nahajajo v uporabnikovi denarnici. Ne-polna vozlišča obstajajo, ker je hranjenje celotne verige lokalno prostorsko zelo potrošno, zato vozlišča kot so SPV shranjujejo samo glave blokov. Namen vozlič SPV je preveriti, ali so transakcije res dodane v blok [3].

[6] Vir: https://www.investopedia.com/terms/m/merkle-tree.asp, datum dostopa: 30.4.2020

Glavni namen Merklovih dreves, poleg zmanjševanja količine informacije potrebne za preverjanje ali se transakcija res nahaja v bloku, je predvsem preverjanje, ali podatkovni bloki, ki jih je vozlišče pridobilo s strani drugih udeležencev, so ali niso poškodovani, spremenjeni ali lažni. Bloki vsebujejo kazalce na prejšnji blok preko prstnega odtisa bloka, ki je sestavljen iz prstnega odtisa glave, kjer nastopa tudi koren Merklovega drevesa. Če je v bloku spremenjena ena ali več transakcij, bo koren drevesa čisto drugačen, kar bi pomenilo, da se spremeni tudi prstni odtis bloka. Spremembo v prstnem odtisu bloka bo takoj opazil naslednji blok, saj vsebuje kazalec na prejšnji blok in ugotovil, da se je nekaj spremenilo, saj ima zabeležen izvirni prstni odtis prejšnjega bloka. Naslednji blok torej ne bo več kazal na prejšnjega, kar bi pomenilo, da je vse od spremenjenega bloka naprej neveljavna veriga, kar povzroči, da se veriga skrajša. Sprememba v tem bloku torej ne bo sprejeta, saj je glavna veriga najdaljša [7].

2.3.4 Rudarjenje in soglasje

Rudarjenje je proces, ki omogoča soglasje v omrežju po vseh vozliščih brez centralne avtoritete. Soglasje po vseh vozliščih pomeni, da imajo vsa vozlišča v sistemu enako stanje oziroma verigo blokov. Motivacija za rudarje je nagrada v obliki valute sistema, ki jo dobijo za vsak uspešno zapečaten blok. V omrežju Bitcoin trenutna nagrada znaša 6.25 BTC, nagrada pa se prepolovi na vsakih 210 tisoč zapečatenih blokov. Začetna nagrada leta 2009 je bila 50 BTC. Poleg te nagrade pa rudarji dobijo tudi nagrado, ki predstavlja vsoto vseh stroškov transakcij. Pričakovano je, da bo leta 2140 nagrada za novo zapečaten blok enaka nič, kar pomeni, da bo od takrat naprej nagrada sestavljena le iz stroškov transakcij [12].

Nagrado prejme rudar, ki kot prvi zapečati novi blok. V sistemih Bitcoin in Ethereum se blok zapečati z rešitvijo zahtevnega matematičnega problema, ki se mu reče dokaz dela (angl. *proof of work*) in predstavlja dokaz, da je rudar porabil veliko računske moči. Nagrada je dodeljena preko transakcije, ki se imenuje Coinbase in je dodana s strani rudarja kot prva v novem bloku. Transakcija Coinbase ima enaka polja kot navadna transakcija, le da ne vsebuje nobenega vhoda [14].

Rudarska vozlišča imajo v procesu pridobivanja soglasja dve nalogi, in sicer stalno preverjanje transakcij ter dodajanje novih transakcij v blok [24]. Nov blok, ki vsebuje transakcije, ki do trenutka še niso vključene v verigo blokov, je v sistemu Bitcoin v povprečju rudarjen vsakih 10 minut, pri čemer se te transakcije skupaj z blokom dodajo na verigo blokov. Z dodajanjem novega bloka so transakcije v bloku potrjene, kar dovoljuje novim lastnikom kovancev, da le-te tudi zapravijo [14].

Centralizirani sistemi skrbijo le za eno podatkovno bazo, kar pomeni, da so podatki vedno najnovejši in obstaja zgolj ena verzija podatkov. Lahko rečemo, da imamo v vozlišču eno vozlišče, ki ima soglasje samo s seboj. Težje je zagotoviti porazdeljeno soglasje, saj obstaja več neodvisnih vozlišč, ki niso pod centralno avtoriteto. Za soglasje morajo biti izpolnjeni štirje pogoji, ki se zgodijo neodvisno na vozliščih v omrežju:

1. Neodvisno preverjanje vsake transakcije, ki jo izvede vsako polno vozlišče, na podlagi kriterijev.
2. Neodvisno združevanje transakcij v nove bloke s strani rudarskih vozlišč, skupaj z izračunom dokaza dela.
3. Neodvisno preverjanje novih blokov s strani vsakega vozlišča in združevanje novega bloka v verigo.
4. Neodvisna izbira verige vsakega polnega vozlišča, ki vsebuje največ potrditev dela [3].

2.3.4.1 Neodvisno preverjanje transakcij na strani polnih vozlišč

Ko se transakcija ustvari, je razmnožena po celotnem omrežju. Preden posamezno vozlišče razmnoži transakcijo, jo mora preveriti. Preverjanje med drugim vsebuje tudi pregled digitalnega podpisa. S tem zagotovimo, da so čez omrežje razmnožene le veljavne transakcije, medtem ko so neveljavne zavržene že pri prvem vozlišču, ki jih prejme. Rezultat procesa preverjanja veljavnosti transakcij je bazen veljavnih transakcij v vsakem vozlišču, ki pa so še nepotrjena [3].

2.3.4.2 Rudarska vozlišča

Rudarska vozlišča predstavljajo posebno vrsto vozlišč, ki so lahko polna ali ne. V primeru, da so polna, vsebujejo vso verigo blokov. Rudarska vozlišča delujejo podobno kot polna vozlišča, kar pomeni, da preverjajo transakcije in tako zgradijo svoj bazen nepotrjenih transakcij. Rudarska vozlišča, nasprotno od polnih vozlišč, transakcije tudi združijo in jih dodajo v nov kandidatni blok. Kandidatni blok še ne predstavlja veljavnega bloka, saj ne vsebuje dokaza dela. Kandidatni blok postane veljaven le v primeru, da rudar, ki je ta kandidatni blok ustvaril, prvi najde rešitev za dokaz dela [14].

Rudarska vozlišča poslušajo omrežje za nove bloke, razširjene po omrežju, kot tudi vsa polna vozlišča. Na rudarska vozlišča prihod novega bloka še toliko bolj vpliva, saj to pomeni, da se je tekmovanje za dodajanje novega bloka končalo, ker je nekdo drug prvi razrešil matematični problem. Obenem to pomeni tudi začetek novega kroga in začetek tekmovanja za naslednji dodani blok [3].

2.3.4.3 Dokaz dela

Cilj rudarja v sistemu Bitcoin in drugih sistemih, kot je na primer tudi Ethereum, je ustvariti veljaven dokaz dela. Osnovna definicija dokaza dela je zgoščevanje glave kandidatnega bloka s spreminjanjem atributa enkratnega števila (nonce), ki se nahaja v glavi bloka, dokler rezultat ne ustreza dogovorjenemu cilju. Cilj dokaza dela je najti prstni odtis glave bloka, ki je manjši od dogovorjene vrednosti. Manjša kot je dogovorjena vrednost, težje je najti ustrezen prstni odtis. Cilj dokaza dela je torej, da se prstni odtis glave začne z dogovorjenim številom ničel. Če je zadan cilj, da se prstni odtis začne s 17 začetnimi ničlami, je primer ustreznega prstnega odtisa:
"00000000000000004dd3426129639082239efd583b5273b1bd75e8d78ff2e8d" [19].

Kot je navedeno tudi v podpoglavju 2.3.1, rezultat prstnega odtisa ne more biti določen vnaprej, prav tako pa ne obstajajo nobeni vzorci možnih rezultatov, zato je edini način za dosego cilja neprestano računanje prstnega odtisa in spreminjanje enkratnega števila, dokler rezultat ne ustreza cilju. Vozlišče, ki prvo ugane ustrezno število, dobi nagrado, dobljen prstni odtis in uganjeno enkratno število pa prestavljata dokaz dela. Rudarji za iskanje ustreznega števila lahko potrebujejo tudi več kot milijardo poskusov. Število ničel, ki jih mora vsebovati cilj, pa je določeno s strani parametra *težavnost*, ki se nahaja v glavi bloka. Težavnost je v omrežju Bitcoin nastavljena tako, da je nov blok zapečaten približno vsakih 10 minut [19]. Težavnost se spreminja dinamično, saj je število rudarjev in s tem same računske moči spremenljivo. Da se ohrani meja približno desetih minut, se mora težavnost ustrezno prilagoditi. Težavnost se prilagodi s strani posameznih vozlišč, in sicer vsako rudarsko vozlišče preveri, koliko časa je bilo potrebnega za rudarjenje zadnjih 2016 blokov. Ob predpostavki, da je za en blok potrebnih 10 minut potem sledi, da je 2016 blokov ustvarjenih v dveh tednih, zato vozlišča preverijo ali je bilo zadnjih 2016 blokov narejenih v manj kot 2 tednih. Če vozlišča ugotovijo, da to drži, se težavnost poveča, v nasprotnem primeru pa zniža [14].

2.3.4.4 Preverjanje novega bloka

Prvi rudar, ki najde ustrezno enkratno število, razširi blok vsem udeležencem v P2P omrežju. Ti preverijo nov blok in ga zavržejo, če ni veljaven, hkrati pa v tem primeru rudarsko vozlišče ne dobi nagrade in je pravzaprav na izgubi, saj je vozlišče porabilo veliko električne energije, da je prišlo do ustreznega enkratnega števila, ki izpolni dokaz dela. Obstaja veliko pravil za preverjanje bloka, ki med drugim preverjajo strukturo bloka in, ali je uganjeno število ustrezno oziroma, ali je dokaz dela izpolnjen. Preverjanje bloka

se dogaja na vsakem vozlišču in preprečuje, da bi rudarji goljufali ali v blok dodali lažne in neveljavne transakcije [3].

2.3.4.5 Dodajanje novega bloka v verigo

Zadnji korak za zagotavljanje soglasja na omrežju je dodajanje novega bloka v verigo [3]. Vsako vozlišče vsebuje tri množice blokov. Dve od teh množic sta povezani v verigo, in sicer glavno in sekundarno verigo, medtem ko tretja množica, ki se imenuje sirota, ni dodana v verigo. Glavna veriga v večini primerov predstavlja množico blokov, ki je v primerjavi s sekundarnimi verigami najdaljša. Lahko se zgodi, da glavna in sekundarna veriga vsebujeta enako število blokov, zato polna vozlišča vzamejo tisto, ki je v procesu izgradnje imela največjo vsoto težavnosti in posledično tudi največji dokaz dela. Ko je nov blok sprejet, ga bo vozlišče poskušalo dodati na glavno oziroma sekundarno verigo preko kazalca na zadnji blok v verigi. Sekundarna veriga je sestavljena iz blokov, ki so se razcepili iz glavne verige, vendar si delijo starša na glavni verigi. Bloki v sekundarni verigi so veljavni in lahko postanejo glavna veriga, če postane sekundarna veriga daljša od glavne. Sekundarne verige nastanejo v primeru, ko sta dva nova bloka narejena ob približno enakem času, saj imata oba kazalec na enakega starša. Oba sta sicer veljavna, zato je prvi blok, ki pride do vozlišča, dodan na glavno verigo, drugi pa na sekundarno. V nekaterih vozliščih se bo zgodilo ravno obratno, saj tehnologija veriženja blokov sloni na P2P omrežju. Drugi blok bo torej prišel k določenemu vozlišču prej kot prvi, zato bo vozlišče drugi blok dodalo na glavno verigo in prvega na sekundarno, s čimer pa se bo porušilo soglasje v sistemu. Pojav, ko nastane sekundarna veriga, se imenuje razpotje (angl. *fork*). V primeru, da se blok doda na sekundarno verigo, vozlišče primerja količino dela sekundarne verige z glavno verigo. Če ima sekundarna veriga več opravljenih dokazov dela, bo le-ta postala glavna veriga, prejšnja glavna veriga, pa bo postala sekundarna. Z izbiro najdaljše veljavne verige kot glavne vsa vozlišča prej ali slej dosežejo soglasje. Tretja množica blokov se imenuje sirota in nastane, če prejet blok nima najdenega starša, torej ustreznega bloka, ki ima enak ustrezen prstni odtis, na katerega kaže kazalec novega bloka. Posledično se ta blok doda v bazen, ki se imenuje sirota, kjer ostane, dokler starš ni dodan na verigo [20].

3 ETHEREUM

Ethereum je tehnologija, ki sloni na tehnologiji veriženja blokov in razvijalcem omogoča razvijanje porazdeljenih aplikacij. Največja razlika med Bitcoin-om in Ethereum-om je v sposobnosti omrežja. Bitcoin ponuja specifično aplikacijo, ki dovoljuje spletna plačila, Ethereum pa je osredotočen na ustvarjanje porazdeljenih aplikacij [34].

Rudarji v omrežju Ethereum delajo z namenom prislužiti si Ether oziroma krajše ETH, ki predstavlja valuto v sistemu Ethereum. Čeprav lahko ta denar tudi zapravijo na enak način, kot pri nakupih z Bitcoin-om, pa je ETH običajno uporabljen iz strani razvijalcev, da plačajo za storitve in stroške transakcij na omrežju [34].

Namen sistema Ethereum je predvsem biti vsesplošni sistem, ki ga je mogoče programirati. Ethereum zaganja navidezni stroj (angl. *virtual machine*), ki je sposoben izvajanja kode. Obenem je sistem tudi Turingovo izpopolnjen (angl. *Turing complete*), kar pomeni, da je lahko Ethereum uporabljen kot vsesplošno uporabljen računalnik, ki lahko izračuna karkoli, pod predpostavko, da ima vozlišče na voljo dovolj spomina [27].

Ethereum, enako kot Bitcoin, deluje kot stroj stanj (angl. *state machine*) ampak namesto, da sledi samo lastnikom kovancev, sledi tudi splošnim podatkom, saj lahko shranjuje kakršnekoli podatke v obliki ključ-vrednost. Programi, ki jih lahko izvajamo nad Ethereum-om, se izvajajo v navideznem stroju, njihova izvedba pa povzroči spremembo stanj. Programi se imenujejo pametne pogodbe (angl. *Smart contracts*) [34].

3.1 Transakcije

Transakcije so digitalno podpisana sporočila, ki so zapisana v blokih in so edina, ki lahko spremenijo stanje podatkov na verigi. Vsaka transakcija pa stane nekaj gas-a. [2] Gas je jedro Ethereuma in ni enak enoti ETH, saj je gas samostojna virtualna valuta s svojim lastnim tečajem. Ethereum uporablja gas za plačevanje stroškov računanja, ki so posledica izvedbe transakcije, poleg stroškov računanja pa tudi vsak bajt podatkov dodanih v transakcijo stane pet enot gas-a. [13] Gas je ločen in neodvisen od Ether-ja, saj mu to omogoča, da sistem zavaruje pred nenehnim spreminjanjem cene Ether-ja [2].

Ethereum za razliko od Bitcoina, ki uporablja ne-zapravljeno transakcijsko izhodno shemo (angl. unspent transaction output), uporablja tako imenovan računsko voden (angl. *account based*) transakcijski model. Računsko voden model je bolj intuitiven, saj deluje enako kot modeli v tradicionalnem bančnem sistemu. Ethereum shranjuje zadnje stanje računov v vsakem trenutku. Po izvedbi transakcije je zadnje stanje računa ustrezno spremenjeno.

Račun je torej lahko obremenjen ali določeno vsoto Ether-a dobi v dobro. Ethereum vsebuje dve vrsti računov, in sicer račun v zunanji lasti ter pogodbeni račun [6]. Računa se med seboj razlikujeta in bosta bolj podrobno predstavljena v nadaljevanju poglavja.

3.1.1 Računi

Ethereum shranjuje zadnje stanje računov vključenih v verigo in je narejeno iz objektov, ki jim rečemo računi. Vsak račun vsebuje svoj unikatni naslov in spremembe stanj, ki predstavljajo direktne obremenitve vrednosti in informacij med računi. Ethereum-ov račun je zgrajen iz štirih polj. Prvo polje se imenuje enkratno število in predstavlja števec, ki je uporabljen kot varovalo pred večkratnim izvajanjem iste transakcije. Drugo polje je stanje na računu v valuti Ether. Tretje polje je koda pogodbe, če je ta prisotna, zadnje polje pa je pomnilnik, ki je na začetku, torej pri kreaciji računa, prazen [39].

Poznamo dve vrsti računov, in sicer take, ki so vodeni s strani zasebnih ključev in pogodbene račune, ki so kontrolirani s kodo pogodbe. Računi vodeni s strani zasebnih ključev so imenovani tudi računi v zunanji lasti (angl. *externally owned accounts*). Računi v zunanji lasti nimajo kode pogodbe in lahko pošiljajo sporočila s pomočjo kreiranja transakcij. V primeru pogodbenega računa, le-to vsakič, ko prejme sporočilo, izvede vnaprej definirano kodo, ki dovoli branje in pisanje v pomnilnik, hkrati pa tudi pošiljanje sporočil ostalim pogodbam [39].

3.1.2 Struktura transakcij

Transakcije so digitalno podpisana sporočila, ki so zapisana v blokih [2], vsaka transakcija pa je sestavljena iz sedmih vhodnih informacij:

1. Enkratno število, ki predstavlja število že potrjenih transakcij.
2. Cena gas-a, ki je število, ki pove, koliko je izvajalec transakcije pripravljen plačati za eno enoto gas-a.
3. Limit gas-a, ki pove, kakšna je maksimalna količina gas-a, ki ga je račun pripravljen zapraviti za izvedbo transakcije.
4. Polje prejemnik, v katerem je napisan naslov ciljnega računa. Prejemnik je lahko zunanje voden račun ali pogodbeni račun.
5. Vrednost, ki predstavlja število enot Ether-a poslanega prejemniku.
6. Podatki, kjer so v večini primerov zapisani podatki, potrebni za ustrezno izvedbo funkcije v pametni pogodbi.
7. Podpis, ki je prisoten kot varovalo in dovoli, da lahko samo lastnik kovancev ustvari transakcijo [2, 25, 13]

3.1.2.1 Enkratno število

Enkratno število je število potrjenih transakcij vezanih na račun [13]. Enkratno število je pomembno v dveh scenarijih, in sicer, ko pošiljatelj želi, da se transakcije zapišejo v pravilnem vrstnem redu ali za preprečevanje ponovljenih transakcij [2].

V primeru, ko pošiljatelj želi, da se transakcije izvedejo v pravilnem vrstnem redu, predpostavimo da sta dve transakciji poslani približno istočasno. Pri prvi transakciji, ki je za pošiljatelja bolj pomembna, je v polju *vrednost* zapisana vsota 8 ETH, pri drugi pa 6 ETH, kljub temu da ima pošiljatelj na razpolago samo 10 ETH. Ker gre za porazdeljen sistem, bo vsako vozlišče v sistemu kot prvo dobilo različno transakcijo. Z enkratnim številom dosežemo, da se bodo vozlišča strinjala in potrdila prvo transakcijo medtem, ko bodo drugo zavrnila [2].

Za primer preprečevanja ponavljanja transakcij predpostavimo, da ima pošiljatelj račun z 100 ETH. Ko pošiljatelj naredi transakcijo, kjer pošlje 2 ETH, jo podpiše in razširi po omrežju. Ko bo pošiljatelj naredil naslednjo transakcijo z 2 ETH, bo ta identična. Vse transakcije so javne, tako da bi lahko tako transakcijo širil kdorkoli, vse dokler pošiljatelju ne zmanjka Eher-ja. Z dodajanjem enkratnega števila se širjenju transakcije brez pošiljateljeve privolitve izognemo, saj je vsaka transakcija drugačna, ker je enkratno število drugačno za vsako transakcijo [2].

Enkratno število v omrežju Bitcoin ni potrebno, saj ne uporablja računsko vodenega transakcijskega modela, vendar uporablja nezapravljeno transakcijsko izhodno shemo.

3.1.2.2 Cena in limit gas-a

Gas je gonilo sistema Ethereum in predstavlja plačilo za opravljeno računsko delo vozliščem med opravljanjem transakcije. Cena gas-a je število, ki predstavlja ceno, ki jo je izvajatelj transakcije pripravljen plačati za eno enoto gas-a. Denarnice lahko parameter cene gas-a, ki je prisoten v vsaki transakciji, spreminjajo, saj višja cena, ki so jo uporabniki pripravljeni plačati, pomeni večjo verjetnost, da je transakcija prej potrjena in s tem dodana v verigo blokov [2].

Druga pomembna informacija v transakciji je limit gas-a, ki predstavlja maksimalno število zapravljenih enot gas-a, ki ga je izvajatelj transakcije pripravljen plačati [13]. V Ethereumu obstajata dve vrsti transakcij, in sicer transakcije med računi v zunanji lasti ter transakcije med računi v zunanji lasti in pogodbenimi računi. Za transakcijo med računi v zunanji lasti je limit gas-a točno poznan za vsako transakcijo in ima vrednost 21000,

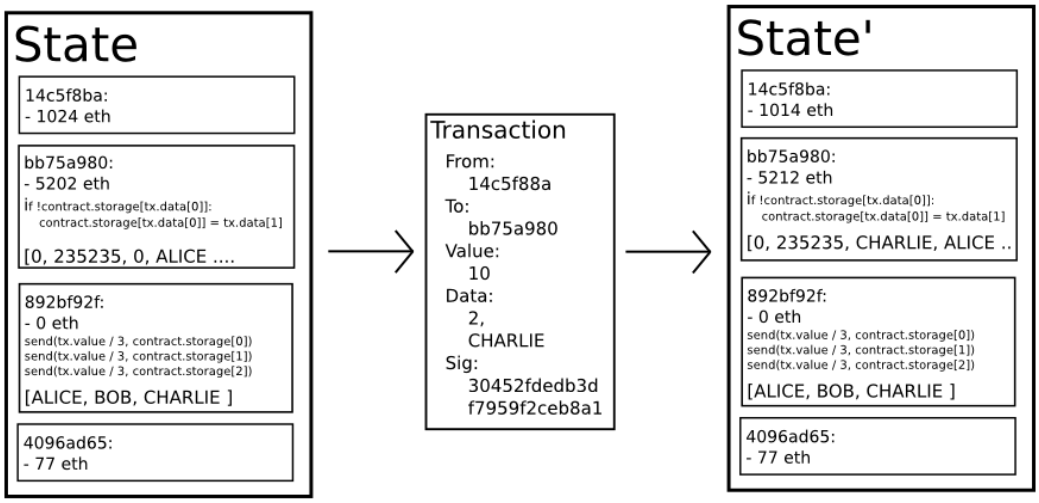
medtem ko za transakcijo med računi v zunanji lasti in pogodbenimi računi limit ni znan, lahko pa je predviden. Točne cene transakcije ni možno določiti, ker imajo lahko pametne pogodbe več pogojev in več poti izvajanja [2]. Cena in limit gas-a sta nastavljena predvsem z namenom odpravljanja neskončnih zank ali drugih računsko zahtevnih delov kode, ki bi pomenila porabo večjih količin gas-a in preobremenitev omrežja [39].

3.1.2.3 Vrednost in podatki

Glavni tovor transakcije je v poljih *vrednost* (angl. *value*) in *podatki* (angl. *data*). Transakcije lahko vsebujejo samo *vrednost*, samo *podatke* ali oboje. Transakcija samo z *vrednostjo* je plačilo, transakcija samo z *podatki* je tako imenovan poziv, transakcija z obojim pa je plačilo in poziv [2]. Do poslanih podatkov lahko dostopa vsaka pametna pogodba in jih ustrezno shrani v pomnilnik. Običajno polje *podatki* vsebuje informacije, ki so potrebne, da se določena funkcija v pametni pogodbi izvede [25].

3.3 Sprememba stanj

Vozlišča v omrežju Ethereum poleg vse zgodovine transakcij vsebujejo tudi zadnje stanje vsakega računa, torej zadnje stanje vsakega računa v zunanji lasti in pogodbenega računa [1].



Slika 8: Primer spremembe stanja na Ethereum-ovi verigi blokov[7].

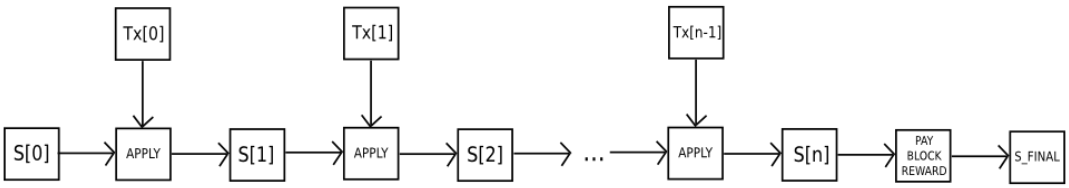
[7] Vir: https://github.com/ethereum/wiki/wiki/White-Paper, datum dostopa: 30.4.2020

Transakcija na sliki 8 prikazuje spremembo stanja, kjer račun z Ethereum-ovim naslovom 14c5f88a pošlje vrednost 10 ETH in podatke 2 ter CHARLIE računu bb75a980. V tem primeru je prejemnik pogodbeni račun, zato se na vozliščih izvede program, in sicer v tem primeru program na drugo mesto v polju vnese ime CHARLIE [39].

Če je transakcija ustrezno strukturirana, je digitalni podpis ustrezen ter ima pošiljatelj na računu dovolj gas-a in ETH, ki ga je navedel kot poslano vrednost, bo rudar transakcijo dodal v blok. Z dodajanjem v blok se bo transakcija izvršila, posledica izvršitve transakcije pa bo sprememba stanja. Ko bo transakcija potrjena, bo s tem tudi povečala pošiljateljevo enkratno število. Če je transakcija neuspešna, se vse potencialne dosedanje spremembe stanj povrnejo v prejšnje stanje. Plačila stroškov transakcij se ne povrnejo v prejšnje stanje ampak se ti stroški dodajo na rudarjev račun [39].

3.4 Veriga blokov in rudarjenje

Ethereum-ova ter Bitcoin-ova veriga, sta si zelo podobni, najbolj pa se razlikujeta v tem, da Bitcoin hrani le kopijo seznama transakcij, Ethereum pa shranjuje oboje, in sicer tako seznam transakcij kot tudi zadnje stanje [39].



Slika 9: Sprememba stanja verige blokov po dodanem bloku[8].

Preverjanje blokov se zgodi na vsakem polnem vozlišču v omrežju. Osnovno preverjanje bloka se začne s preverjanjem, ali kazalec na prejšnji blok obstaja in, ali je pravilen, torej ali starš pripada verigi. Drugi korak je preverjanje podatkov v glavi [39]. Podatki v glavi so praktično enaki kot tisti, ki so opisani v poglavju 2.3.3. Pomemben dodan atribut v glavi bloka je limit gas-a, ki predstavlja vrednost, ki določa kolikšna je največja vsota porabljenega gas-a vseh transakcij v bloku [13]. Naslednji korak je sestavljen iz preverjanja dokaza dela, ki ga je bloku dodal rudar. Po končanem preverjanju dokaza dela vozlišča preverijo Merklov koren, ki je pripet v glavi bloka. Za preverjanje Merklovega korena mora polno vozlišče najprej preveriti trenutno stanje v verigi po dodanem novem

[8] Vir: https://github.com/ethereum/wiki/wiki/White-Paper, datum dostopa: 30.4.2020

bloku. Grajenje zadnjega stanja poteka na način, kot ga prikazuje slika 9. S[0] prikazuje stanje verige pred dodanim novim blokom, Tx pa je seznam z n transakcijami. Po vsaki transakciji v bloku se stanje verige spremeni, zato je za vse i med 0 in n-1, stanje S[i + 1] enak prejšnjemu stanju, torej stanju S[i], kateremu je dodana transakcija i. Med dodajanjem transakcij stanjem, vozlišča preverjajo limit gas-a podan v glavi bloka in v primeru, da se med izvajanjem transakcije ta limit preseže, blok zavrnejo. Postopek vozlišča ponavljajo, dokler ne pridejo do stanja S[n], kateremu prištejejo še nagrado za rudarja za uspešno zapečaten blok in s tem dobijo končno stanje po dodanem bloku, ki je na sliki označeno z S_FINAL. Na koncu vozlišča še preverijo, ali je koren Merklovega drevesa stanja S_FINAL enak korenu v glavi bloka. Če se Merklova korena ujemata, je blok veljaven [39].

Ob prvem pogledu se postopek preverjanja bloka proti Bitcoin-ovem sistemu zdi precej potrošen, saj mora vsako polno vozlišče poleg vseh transakcij shranjevati stanje računov, vendar je v realnosti učinkovitost postopkov precej podobna. Razlog za podobno učinkovitost je dejstvo, da je stanje shranjeno v drevesni obliki. Po vsakem bloku je le manjši del stanja, in s tem tudi drevesa, deležen spremembe, zato so isti deli drevesa lahko uporabljeni večkrat, in sicer s pomočjo kazalcev, ki so prstni odtisi poddreves. Zadnje shranjeno stanje tudi omogoča, da ne shranjujemo celotne zgodovine transakcij [39].

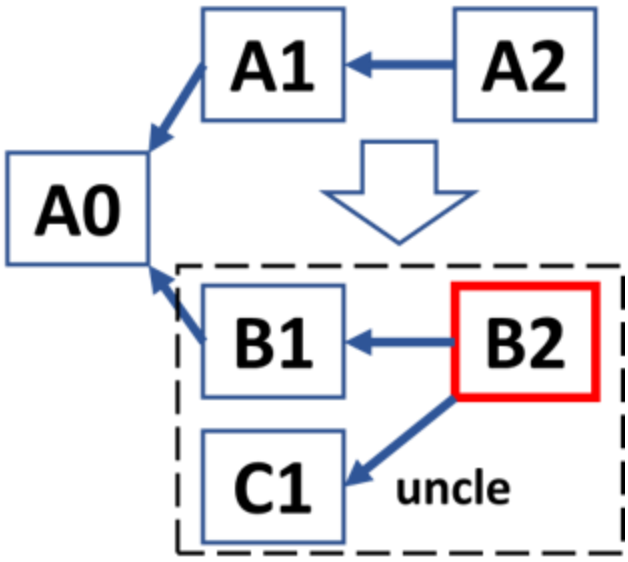
Blok v omrežju Ethereum je v povprečju rudarjen vsakih 10 do 19 sekund. To je veliko hitreje kot v omrežju Bitcoin, kjer je čas rudarjenja približno 10 minut. V Ethereumu je težavnost nastavljena nižje, kar omogoča hitrejše računanje uganjenega števila in s tem hitrejše spreminjanje stanj, vendar to pomeni, da lahko nastaneta dva večja problema varnosti [30].

Bloki potrebujejo nekaj časa, da se razširijo po omrežju. Ko rudar A najde dokaz dela za svoj kandidatni blok, ga takoj pošlje po omrežju, vendar rudar B za ta blok ne izve v istem trenutku, saj se mora blok prenesti od rudarja A do rudarja B, kar traja nekaj časa. V času prenosa bloka lahko B naredi dokaz dela za svoj blok. Ker je bil blok rudarja A razpršen po omrežju pred blokom rudarja B, bo večina vozlišč v verigo dodala blok rudarja A, medtem pa bo blok rudarja B postal sirota in s tem ne bo pripomogel k varnosti omrežja, saj bo njegov blok neporabljen. Druga nevarnost je povezana s centraliziranostjo. Če ima rudar A oblast nad 30% vse računske moči v omrežju (angl. *hashpower*) in B nad 10%, bo A imel možnost produciranja sirote v 70% primerov, rudar B pa v 90 % primerov. Ko je čas generiranja blokov majhen, bo imel A veliko več možnosti, da bo morebitna razpotja razrešil v svojo korist. S kombiniranjem teh dveh problemov, bo prej ali slej prišlo do tega, da bo eno vozlišče imelo zadosten delež računske moči, da bo lahko prevzelo kontrolo nad rudarskim procesom [39]. Prevzetje kontrole nad rudarskim procesom bi dovolilo

manipuliranje verige, kar bi ob omogočilo dvojno zapravljanje istega denarja in zavračanje transakcij, ki prihajajo iz določenega računa [43].

Za reševanje navedenih problemov Ethereum uporablja posebno verzijo protokola, ki se imenuje pohlepno najtežje opaženo poddrevo (angl. *Greedy Heaviest Observed Subtree* - GHOST). GHOST reši težavo z vključitvijo sirot v računanje najdaljše verige, v nasprotju z vključevanjem zgolj starša in njegove predhodnike. Sirotam, ki so vključene v računanje, Ethereum pravi strici. Ti so dodani v izračun, ki določa, katera veriga ima največ dokazov dela. Vsak blok lahko kaže na nič ali več stricev. Kazalec na strice je dodan v glavo bloka. Motivacija za dodajanje strica v blok je nagrada za vsako vključeno siroto. Obenem tudi rudar, ki ustvari siroto, dobi nagrado [39].

Na sliki 10 sta prikazani dve verigi, A in B. Bloka A1 in B1 sta povzročila razpotje. Veriga A je bila izbrana kot glavna veriga in B kot sekundarna. Kasneje se je vključila še sirota C1, ki jo je blok B2 vključil kot strica, zato je postala najtežja veriga, B pa je postal glavna veriga.



Slika 10: Primer razpotja in kazalca na strica[9].

Ethereum uporablja prilagojeno verzijo GHOST protokola, in sicer blok omeji tako, da lahko vsebuje kazalec na strica za največ 7 generacij nazaj. Prilagoditev, da so strici lahko stari le 7 generacij, je bila dodana iz dveh razlogov. Dovoljenje, da je lahko stric iz kjerkoli, bi povzročilo veliko zapletov pri računanju, kateri strici vsebovani v bloku so

[9] Vir: https://medium.com/ibbc-io/ethereum-uncles-how-family-makes-you-stronger-d6e7aaef7b2b, datum dostopa 30.4.2020

veljavni. Drugi razlog pa je, da ta prilagoditev spodbudi rudarja, da rudari na glavni veji in ne veji napadalca. Rudar bi potencialno lahko delal sirote za starejše generacije v verigi blokov. To bi pomenilo, da bi jih napadalec lahko vključil v svojo verigo, saj bi predstavljale predhodnik v napadalčevi verigi. Sirote, ki so vključene v verigo, morajo biti namreč predhodniki novega bloka, stric pa je lahko uporabljen samo enkrat [39].

Ethereum v prihodnosti namerava zamenjati dokaz dela z dokazom o vložku (angl. proof of stake). Prednost dokaza o vložku je predvsem v tem, da porabi manj električne energije, poleg tega pa poveča varnost omrežja in obenem zmanjša možnost centralizacije omrežja, saj bi bil vsak potencialen napad na omrežje zelo drag [32].

3.5 Pametne pogodbe

Pametne pogodbe omogočajo razvijalcem ustvarjanje porazdeljenih programov, ki omogočajo porazdeljen strežnik in porazdeljeno bazo, namesto da se razvijalci zanašajo na centralni strežnik in centralno bazo podatkov [25].

Pametne pogodbe so aplikacije, ki so zgrajene na tehnologiji veriženja blokov. Pametne pogodbe so programi, ki se izvajajo med validacijo bloka. Fizično se koda zapisana v pametnih pogodbah izvaja na vseh vozliščih, ki preverjajo blok, torej na rudarskih in polnih vozliščih [25]. Morda je izraz pametna pogodba zavajajoč, saj ne gre za pogodbo, vendar se izraz pametna pogodba nanaša na nespremenljiv deterministično računalniški program, ki se izvaja na vozliščih v omrežju. Nespremenljiv se nanaša na to, da se kode pametne pogodbe ne da spremeniti, ko je enkrat objavljena na omrežje. Edini način za spremembo pogodbe je objava nove. Deterministične pa so pogodbe zato, ker je za isti kontekst na Ethereum-ovem navideznem stroju, izhod vedno enak. Kontekst je sestavljen iz trenutnega stanja na verigi blokov in konteksta transakcije, ki je zagnala program [2].

Pametne pogodbe so omejene in niso primerne za aplikacije, ki procesirajo informacije na strežnikih in zahtevajo hitro procesiranje. Pametne pogodbe so počasne, saj so vezane na čas rudarjenja bloka in niso dobre za aplikacije, ki se izvajajo v realnem času. Enako niso primerne za aplikacije, ki shranjujejo veliko podatkov, saj je shranjevanje velike količine podatkov na verigo blokov cenovno drago [25].

Pametne pogodbe so Turingovo popolne, kar pomeni, da lahko izvajajo zanke v neskončnost, saj Turingova popolnost predpostavlja neskončno velik spomin. V izogib neskončnim zankam mora vsaka transakcija nastaviti limit, ki določa največjo vrednost gas-a, ki ga lahko transakcija porabi. Ko je limit dosežen se operacija zaključi. Gas se izračuna glede na računsko moč, ki je potrebna za izvedbo pametne pogodbe [25].

3.5.1 Objava pametne pogodbe

Vsaka pametna pogodba se mora pred uporabo namestiti na Ethereum-ovo omrežje. Namestitev transakcije ni nič drugega kot generiranje transakcije z kodo aplikacije. Koda pametne pogodbe je vključena v parameter *podatki* kot bajtna koda. Transakcija za namestitev pametne pogodbe je poslana naslovu "0x0", tako da vozlišča razumejo, da se kreira nova pametna pogodba. Polje *podatki* v tem primeru vsebuje vso logiko pametnih pogodb [25].

Vsaka pogodba je unikatno identificirana preko Ethereum-ovega naslova, ki je izpeljan iz kreacijske transakcije, in sicer kot funkcija računa kreatorja transakcije in enkratnega števila. Pametni pogodbi ne pripadajo nikakršni kriptografski ključi. Lastniki Ethereum-ovih računov lahko pošiljajo transakcije pogodbi, to pa je tudi edini način, s katerim lahko izvedemo kodo v pametni pogodbi. Pametne pogodbe lahko tudi pošiljajo sporočila med seboj [2].

Pametne pogodbe se lahko tudi uničijo, kovanci (ETH), ki jih drži pripadajoč pogodbeni račun, pa se nato nakažejo na poljuben naslov, ki ga navede tisti, ki uniči pogodbo, oziroma na naslov, ki je naveden ob kreaciji pametne pogodbe. Celotna zgodovina transakcij pa sicer ostane na verigi blokov tudi po uničenju [25].

4 INTERPLANETARY FILE SYSTEM (IPFS)

Medplanetarni datotečni sistem (IPFS) je odprtokoden P2P sistem, katerega cilj je preiti v porazdeljeno arhitekturo interneta. Porazdeljen internet predstavlja internet, katerega povezave in datoteke nikoli ne umrejo, hkrati pa nobena tretja oseba ne nadzoruje podatkov. IPFS je vsebinsko naslovljen sistem (angl. *content-addressed system*), kar pomeni, da je zahtevana datoteka naslovljena preko prstnega odtisa datoteke, medtem ko je internet IP naslovljen sistem (angl. *IP-addressed system*). Ko je datoteka dodana na omrežje IPFS, vozlišče kot odgovor vrne prstni odtis dodane datoteke, preko katerega je omogočen dostop do datoteke. Dostopanje do datotek preko prstnega odtisa predstavlja dober način dostopanja do datotek, saj je zagotovljeno, da je uporabnik pridobil enako in nespremenjeno datoteko. Vsaka sprememba v datoteki namreč povsem spremeni prstni odtis [35].

Prednost vsebinsko naslovljenega sistema je, da ta ni odvisen od enega samega strežnika, katerega izpad bi pomenil onemogočen dostop do podatkov. Obenem je vsebinsko naslovljen sistem hitrejši, saj potrebne datoteke dobi od najbližjega vozlišča, ki ima zahtevane datoteke v omrežju [35]. Vsako vozlišče v omrežju nudi in shranjuje datoteke, nihče jih ne zgolj zahteva. Prednost takega sistema, podobno kot tudi drugih P2P sistemov, je, da je veliko težje prepovedati dostop do spletnih strani. Države ali organizacije uporabnikom ne morejo onemogočiti dostopa do določene spletne strani, kot je to na primer storila Turčija, ki je leta 2017 državljanom prepovedala dostop do spletne strani Wikipedia [42].

Datoteke na IPFS so permanentne, kar pomeni, da jih ni možno izbrisati, vendar pa bodo na omrežju ohranjene le toliko časa, kolikor se bodo udeleženci na P2P sistemu zanje zanimali in jih posledično shranjevali na svojem vozlišču. Če nihče ne kaže zanimanja za določeno datoteko, bo ta datoteka izgubljena. Enaka pomanjkljivost se sicer pojavi tudi na trenutnem internetu, ki mu pravimo tudi web 2.0, saj je mogoče izbrisati vsebino iz strežnikov [42].

4.1 Delovanje tehnologije IPFS

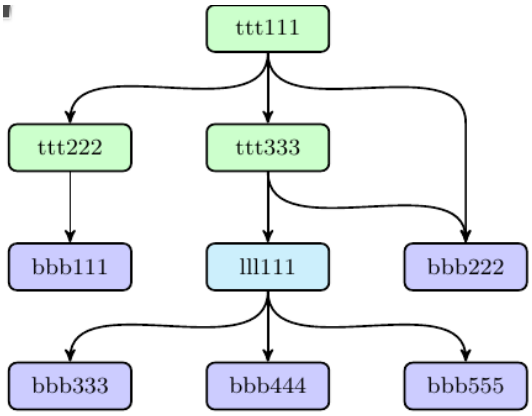
IPFS je sistem, ki sloni na P2P omrežju in je namenjen shranjevanju datotek. Vozlišča v sistemu IPFS v svojem lokalnem spominu shranjujejo IPFS objekte. Objekti predstavljajo datoteke in druge podatkovne strukture, kot so na primer Merklovi grafi. Vozlišča so povezana med seboj, z namenom prenašanja objektov [17].

4.1.1 Povezava vsebin

Jedro sistema IPFS je del sistema, ki se imenuje medplanetarno povezani podatki (angl. *inter planetary linked data - IPLD*), kjer so datoteke povezane med seboj preko Merklovih povezav, ki tvorijo Merklov graf. Merklov graf je usmerjen acikličen graf, vendar v splošnem ni nič drugega kot Merklovo drevo, ki je predstavljeno v poglavju 2.3.3.1. Graf se od drevesa razlikuje v tem, da za graf ni nujno, da je uravnotežen, hkrati pa tudi vozlišča, ki niso listi, torej vmesna vozlišča med listi in korenom, lahko shranjujejo podatke. Uporaba strukture Merklovega grafa omogoča vsebinsko naslavljanje, saj je vsebina unikatno identificirana s prstnim odtisom. Vse datoteke so odporne na spreminjanje, saj se v primeru spremembe dela datoteke, koren Merklovega grafa spremeni, spremembo korena grafa pa bodo zaznali ostali udeleženci v omrežju. Naslavljanje preko vsebine nam omogoča tudi, da v omrežju ni duplikatov, saj imata dve enaki datoteki enak koren Merklovega grafa [28].

Grajenje Merklovega grafa poteka na sledeč način. Najprej se naložena datoteka razdeli na več manjših delov, ki se imenujejo bloki. Vsak blok, ki sestavlja datoteko, ima izračunan prstni odtis, iz katerih je nato preko povezav zgrajen Merklov graf. Ko sistem datoteko razdeli na bloke, jo poveže preko ustreznih povezav v Merklovem grafu in s tem omogoča, da je vsak blok shranjen na drugem vozlišču. Shranjevanje delov datoteke na različna vozlišča omogoča prejemanje datoteke s strani več vozlišč hkrati, kar poveča hitrost prenosa podatkov. Gre za enako idejo, kot je omenjena v poglavju 2.2 pri protokolu BitTorrent, kjer se datoteke prejemajo na enak način torej z več vozlišč hkrati [17].

Poleg izboljšanja hitrosti prenosa podatkov, je dodatna prednost Merklovih grafov ter razdeljevanja datotek v bloke ta, da si dve podobni datoteki lahko delita dele Merklovih grafov, kar pomeni, da lahko deli Merklovih grafov kažejo na isto podskupino podatkov. Deljenje delov grafa je predvsem uporabno pri prenašanjih velikih baz podatkov kot so genomski zapisi, saj takšen način delovanja močno izboljša prenos datotek. Prenos datotek je izboljšan, ker je potrebno prenesti zgolj dele, ki so novi oziroma nespremenjeni, namesto ustvarjanja popolnoma novih datotek na omrežju [42].



Slika 11: Merklov graf [17].

```
> ipfs file-cat <ttt111-hash> --json
{
  "data": ["tree", "tree", "blob"],
  "links": [
    { "hash": "<ttt222-hash>",
      "name": "ttt222-name", "size": 1234 },
    { "hash": "<ttt333-hash>",
      "name": "ttt333-name", "size": 3456 },
    { "hash": "<bbb222-hash>",
      "name": "bbb222-name", "size": 22 }
  ]
}
```

Slika 12: Izpis Merklovega grafa po izvedbi v sistemu IPFS [17].

Slika 11 prikazuje primer Merklovega grafa. Razvidno je, da graf ni nujno binaren, kot je to potrebno pri Merklovem drevesu. Slika 12 ponazarja izpis grafa iz omrežja IPFS za vozlišče *ttt111*. Polje *Data* ponazarja podatke o vozliščih, do katerih ima vozlišče *ttt111* povezave. *Tree* prikazuje kazalec na drugo drevo, preko prstnega odtisa vozlišča, medtem ko *blob* označuje končen blok, ki vsebuje del datoteke, kot je prikazano na sliki 13.

```
> ipfs file-cat <bbb222-hash> --json
{
  "data": "blob222 data",
  "links": []
}
```

Slika 13: Primer vozlišča z podatki [17].

4.1.2 Odkritje vsebin

Za iskanje vozlišč, ki shranjujejo želene datoteke, IPFS uporablja porazdeljene razpršene tabele (angl. *distributed hash table* - *DHT*). Razpršena tabela je baza, ki vsebuje

podatkovni par, ki je sestavljen iz ključev in vrednosti. Porazdeljena razpršena tabela je razpršena tabela, ki je porazdeljena med vsemi udeleženci v P2P omrežju [42].

Ključi v porazdeljeni razpršeni tabeli v sistemu IPFS so prstni odtisi. Vsako vozlišče v omrežju je odgovorno za podmnožico razpršilnih tabel. Ko vozlišče prejme zahtevo, lahko takoj odgovori, če ima zahtevan podatek. V nasprotnem primeru pošlje zahtevo naprej drugemu vozlišču, dokler se ne najde vozlišče, ki lahko na zahtevo odgovori. Obstaja več možnih principov za iskanje odgovora, če prvo vozlišče nima zahtevane vrednosti. IPFS za iskanje odgovora na zahtevo uporablja princip, kjer v primeru, da vozlišče nima odgovora na zahtevo, vozlišču, ki je oddalo povpraševanje, pošlje informacijo o vozlišču, ki bo z večjo verjetnostjo imelo želeno informacijo [11]. Prednost iskanja objektov prek porazdeljenih razpršenih tabel je predvsem razširljivost (angl. *scalability*), kar pomeni da je sistem funkcionalen tudi v primeru več tisoč vozlišč. Natančno iskanje končnega vozlišča, ki ima iskano informacijo v sistemu IPFS, zahteva največ $\log_2(N)$ korakov, pri čemer je N število vozlišč v omrežju [22]. Porazdeljene razpršene tabele so tudi odporne na napake zaradi redundance in so torej dostopne tudi, če vozlišča odidejo oziroma se pridružijo omrežju. Poleg tega so lahko zahteve naslovljene kateremukoli drugemu vozlišču, če je neko vozlišče počasno ali nedostopno. Porazdeljene razpršilne tabele pomagajo tudi pri uravnavanju obremenitev, saj v nobenem primeru ni zgolj eno vozlišče zadolženo za obdelavo vseh zahtev, kot bi bilo v primeru, da bi razpršilno tabelo imel samo centralni strežnik [11].

Vsako vozlišče v omrežju ima svoj unikatni identifikator, ki se imenuje identifikator vozlišča (angl. *Peer ID*). Identifikator vozlišča je prstni odtis dolžine *n* in je po dolžini enak ključu v porazdeljeni razpršilni tabeli. Podmnožica porazdeljenih razpršilnih tabel, ki jih vsebuje vozlišče, se imenuje vedro (angl. *bucket*) in vsebuje povezave med ključi in vrednostmi. Ključ predstavlja prstni odtis vsebine, ki je lahko prstni odtis bloka datoteke oziroma prstni odtis dela Merklovega grafa, vrednost pa je lahko identifikator vozlišča, ki shranjuje to datoteko, ali pa direktno objekt, če je velikost objekta manjša od 1 kb. Vedro za ključe vsebuje prstne odtise, ki se začnejo s predpono identifikatorja vozlišča. Predpona identifikatorja vozlišča je spremenljive dolžine. Vsako vozlišče vsebuje tudi seznam drugih vozlišč, ki so razporejena po bližini, ki je določena glede na identifikator vozlišč. Če je dolžina prstnega odtisa n, potem ima vozlišča razdeljena v n-1 seznamov, kjer so v prvem seznamu vozlišča, ki imajo prvi bit identifikatorja različen, medtem ko so v zadnjem seznamu, torej v n-1-tem seznamu, vozlišča, ki se v identifikatorju razlikujejo po zadnjem bitu. Najbližja so si tista vozlišča, ki si delijo čim večjo predpono enakih bitov [11].

Iskanje po porazdeljenih razpršilnih tabelah poteka na naslednji način. Ko vozlišče prejme zahtevo, najprej v svojem vedru preveri, ali vsebuje odgovor za zahtevano vrednost. Če je

zahtevan objekt prisoten, bo odgovoril z objektom, v nasprotnem primeru pa bo odgovoril z lokacijo bližjega vozlišča, gledano po zahtevani vrednosti, saj ima to vozlišče več možnosti, da vsebuje zahtevano informacijo. Opisan proces se ponavlja, dokler eno od vozlišč ne odgovori z zahtevanim odgovorom [11].

Vozlišča v omrežju IPFS potrebujejo sistem usmerjanja zahtev po omrežju s pomočjo porazdeljenih razpršilnih tabel, da lahko najdejo omrežne naslove drugih vozlišč in vozlišča, ki lahko servirajo iskane objekte [17]. Pri iskanju vsebine je najprej narejena poizvedba za iskanje vozlišč, ki shranjujejo datoteko preko prstnega odtisa vsebine. Odgovor na prvo poizvedbo so identifikatorji vozlišč, ki shranjujejo dele iskane datoteke. Za pridobitev datoteke je potreben mrežni naslov oziroma lokacija vozlišč, katerih identifikatorje smo dobili po prvi poizvedbi, zato je potrebno še eno iskanje po porazdeljeni razpršilni tabeli. V drugem zahtevku po porazdeljeni razpršilni tabeli je torej iskan mrežni naslov vozlišča preko identifikatorja vozlišča. V prvi poizvedbi pa identifikator vozlišča preko prstnega odtisa vsebine [42].

Potem, ko je lokacija blokov znana, je potrebna povezava z vozlišči, ki hranijo iskane bloke. Za ta namen IPFS uporablja modul Bitswap, ki omogoča povezavo z vozlišči, ki hranijo zahtevane bloke. Po povezavi z vozlišči, ki hranijo bloke datotek, tisto vozlišče, ki blok želi, vozliščem pošlje seznam želja, v katerem navede želene bloke, vozlišča pa kot odgovor pošljejo bloke, ki jih lahko ponudijo. Ko bloki prispejo, je z zgoščevanjem prejetega bloka in primerjanjem dobljenega prstnega odtisa z zahtevanim možno preveriti integriteto bloka. Z ujemanjem prstnih odtisov je potrjeno, da je blok nepoškodovan. Končna datoteka je dobljena z združevanjem prejetih blokov. Bloki so združeni s sledenjem po Merklovem grafu [11, 42].

5 PROTOTIPNA IMPLEMENTACIJA STORITVE PORAZDELJENEGA SHRANJEVANJA ZASEBNIH DATOTEK

Trenutno v računalništvu prevladuje trend računalništva v oblaku, ki predstavlja tehnologijo, pri kateri so računalniški viri dostopni preko storitev na internetu. Bistvena prednost računalništva v oblaku je, da so storitve na oblaku dostopne na katerikoli napravi in na katerikoli lokaciji, obenem pa uporabniki postanejo neodvisni od strojne opreme, saj do strežnikov dostopajo preko svojega brskalnika [1]. Ena od aplikacij, ki je grajena na podlagi tehnologije računalništva v oblaku je tudi spletno hranjenje podatkov. Primer aplikacij za spletno shranjevanje sta na primer Dropbox in Google Drive. Trenutna implementacija storitev večinoma sloni na arhitekturi odjemalec-strežnik in zagotovo dobro deluje, vendar ima nekaj slabosti. Kot omenjeno tudi v poglavju 4, je težava pri arhitekturi odjemalec-strežnik predvsem cenzura. Vlada lahko prepove dostop do določenih spletnih strani ali pa onemogoči širjenje informacij. Drugi, ki je hkrati verjetno tudi največji problem, pa je ta, da podjetja običajno za shranjevanje datotek ne uporabijo svojih strežnikov vendar uporabijo storitve drugih podjetij, ki to omogočajo. Z uporabo drugega podjetja izgubijo nadzor nad datotekami, kar za sabo pripelje tudi izgubo kontrole nad zasebnostjo datotek [4]. Ena večjih težav z arhitekturo odjemalec-strežnik je tudi ena sama točka odpovedi, kar pomeni, da v primeru, ko sta strežnik ali podatkovna baza zaradi kakršnihkoli razlogov nedostopna, tudi vsebina, ki je hranjena na strežniku, ne bo dostopna. Trenutno je večina aplikacij grajenih na osnovi Amazonovih spletnih rešitev (angl. *Amazon web services - AWS*). Posledično bi izpad Amazonovih strežnikov pomenil nedostopnost velike večine spletnih storitev [31].

Rešitve za omenjene težave ponuja porazdeljeno omrežje. Porazdeljeno omrežje omogoča možnost razbitja datotek na dele, ki so razpršeni po več vozliščih v omrežju. Razpršitev datotek na več vozlišč reši problem ene same točke odpovedi, saj je datoteka dostopna na več kot enem vozlišču, vsako vozlišče pa lahko tudi sestavi datoteko s povpraševanjem drugih vozlišč, kot to dela sistem IPFS. V porazdeljenem sistemu je praktično nemogoče onemogočiti dostop do vsebine, saj bi to pomenilo preprečitev dostopa do vseh vozlišč v omrežju. Ker se vozišča omrežju lahko kadarkoli priključijo in kadarkoli spremenijo svoj mrežni naslov, je praktično nemogoče prepovedati dostop do vseh vozlišč. Slabost shranjevanja datotek v porazdeljenih sistemih pa je, da so datoteke javno dostopne vsem vozliščem v omrežju. Za preprečitev branja datotek drugim vozliščem razvijalci uporabljajo različne kriptografske tehnike, ki jih bom tudi sam uporabil v implementaciji prototipa storitve porazdeljenega shranjevanja zasebnih datotek. V prototipu je uporabljena tudi tehnologija veriženja blokov, ki predvsem zagotavlja integriteto podatkov, kar pomeni, da zagotavlja, da bo vrnjena datoteka enaka naloženi. Med drugim pa tehnologija veriženja blokov preko prstnega odtisa datoteke omogoča iskanje izvora datoteke, saj je

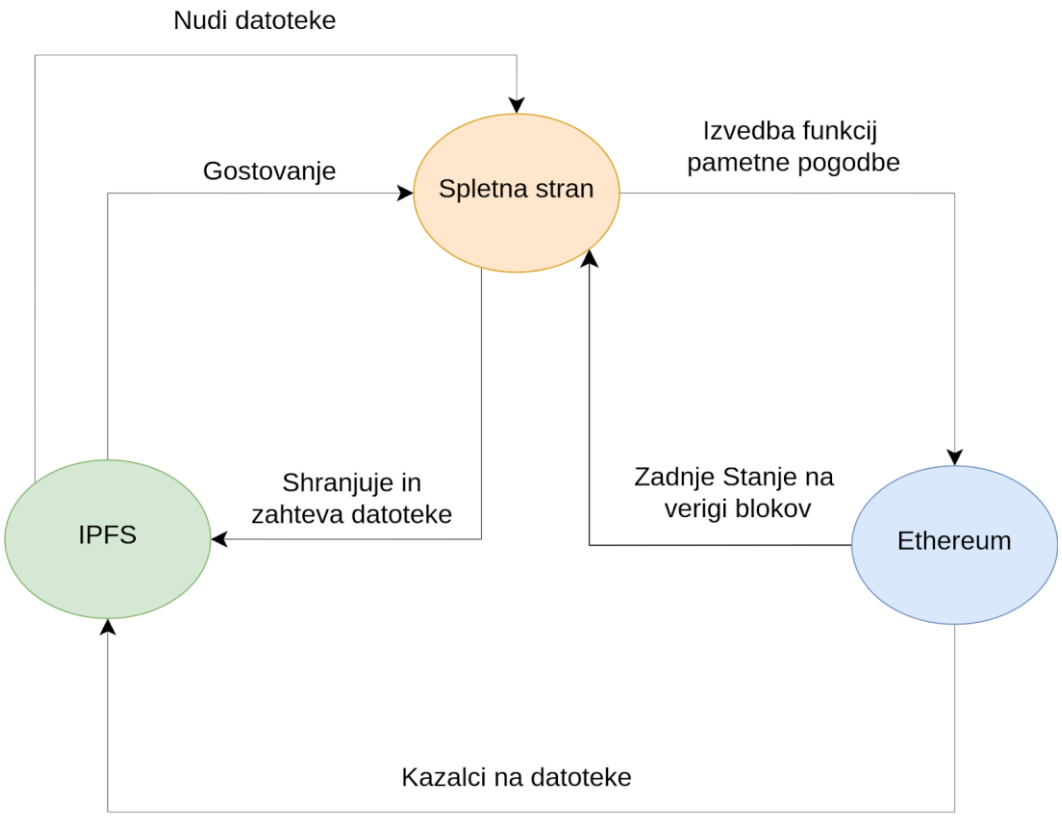
možno določiti, kdo je prvi naložil datoteko in kako se je potem delila naprej, kot v primeru oskrbovalne verige opisane v poglavju 2.1.

Opis prototipa je dostopen tudi v video obliki na povezavi: https://youtu.be/GP5oyiu7wNs

5.1 Arhitektura

Prototip sloni na osnovi dveh omrežij, in sicer Ethereum-ove verige blokov in omrežja za gostovanje statičnih datotek IPFS. Na Ethereum-ovo omrežje je naložena pametna pogodba, ki služi shranjevanju podatkov o datotekah, medtem ko IPFS shranjuje naložene datoteke. Obenem IPFS gostuje tudi spletno aplikacijo, tako da sistem ne sloni na nobenem centralnem strežniku. Za potrebe zaključne naloge je Ethereum-ova veriga blokov postavljena lokalno na osebnem računalniku. Lokalna veriga je postavljena s pomočjo programa Ganache, ki lokalno postavi verigo blokov. Pri inicializaciji Ethereum-ovega sistema Ganache ustvari 10 računov, pri čemer ima vsak od njih na začetku na voljo 100 ETH. Ganache opravi tudi rudarjenje blokov.

Spletna aplikacija je torej gostovana na sistemu IPFS in obenem povezana z vozliščem v Ethereum-ovem omrežju in vozliščem v omrežju IPFS.



Slika 14: Arhitektura aplikacije.

5.2 Simetrična kriptografija

Za dosego zasebnosti datotek morajo biti uporabnikove datoteke šifrirane na način, da si jih ne more ogledati nihče drug kot le lastnik datoteke. Eden od načinov za učinkovito šifriranje velikih datotek je uporaba simetrične kriptografije. Simetrična kriptografija sestoji iz enega kriptografskega ključa, ki je enak za šifriranje in dešifriranje. Simetrična kriptografija se uporablja za šifriranje večjih podatkov, saj je simetrična kriptografija hitrejša in po uporabi računske moči bolj učinkovita kot asimetrična kriptografija [40]. Eden od simetričnih kriptografskih algoritmov, je algoritem AES, ki je uporabljen v prototipu in je eden izmed najpogosteje uporabljenih algoritmov. Za uporabljanje algoritma AES je potreben kriptografski ključ in inicializacijski vektor. Šifriranje datotek poteka v blokih, ki so lahko dolžine 128, 192 ali 256 bitov. Inicializacijski vektor (IV) je 16 bitno naključno število, pri čemer je pomembno, da se IV pri šifriranju z enakim ključem uporabi le enkrat. Uporaba inicializijskega vektorja omogoča, da za enake podatke dobimo različno šifrirano sporočilo, kar onemogoči napadalcu, da preko vzorcev pridobi kriptografski ključ, ki je bil uporabljen za šifriranje [23].

5.3 Pametna pogodba

Kot je opisano v poglavju 3.5, so pametne pogodbe programi, ki se izvajajo na polnih in rudarskih vozliščih v omrežju. Funkcija v pametni pogodbi se izvede preko transakcije, ki tudi spremeni stanje pametne pogodbe in za izvedbo potrebuje določeno vsoto gas-a. Pametne pogodbe pa poleg funkcij, ki spremenijo stanje, vsebujejo tudi funkcije, ki delujejo kot bralci (angl. *getter*) in ne spremenijo stanja na verigi blokov, vendar le vrnejo zadnje stanje brane spremenljivke.

Pametna pogodba, ki je uporabljena v prototipu, je napisana v jeziku Solidity. Solidity je objektno orientiran jezik namenjen pisanju pametnih pogodb. Pametna pogodba, ki je potrebna za aplikacijo, vsebuje dve spremenljivki, katerih stanje je shranjeno na verigi blokov. Pametna pogodba shranjuje podatke o naloženih datotekah, ki so prikazani na sliki 15, in sicer pod strukturo *FileStruct,* in javne ključe za pripadajoče račune. Vsak uporabnik ima lahko več naloženih datotek, zato so shranjene v podatkovno strukturo polje. Kot je prikazano na sliki 15, imata obe spremenljivki *files* in *publicKey* pred imenom besedo javno (angl. *public*), kar avtomatsko ustvari funkcijo za enostavno dostopanje do stanja spremenljivk.

```
struct FileStruct {
  string file_type;
  string encrypted_aes_key;
  string encrypted_iv;
  string file_hash;
  string file_name;
}

mapping(address => FileStruct[]) public files;
mapping(address => string) public publicKey;
```

Slika 15: Podatkovni strukturi v pametni pogodbi.

Pametna pogodba pa vsebuje tudi funkcije za spreminjanje vrednosti. Funkcije v pametni pogodbi, ki so potrebne za delovanje prototipa, so namenjene nastavljanju javnega ključa in dodajanju informacij o naloženih datotekah. Funkcije so zelo enostavne in zgolj spremenijo stanje spremenljivk, saj celotno procesiranje poteka na spletni strani. Kljub temu pa so transakcije precej potrošne z gas-om, saj transakcije za dodajanje datotek vsebujejo zelo dolge kriptografske ključe in prstne odtise (cena transakcije je pri trenutni povprečni ceni gas-a približno 0.80€).

Struktura funkcij v pametni pogodbi je zelo podobna. Ena od funkcij je funkcija za dodajanje javnega ključa, ki na mesto naslova pošiljatelja doda njegov javni ključ. Vsaka funkcija kot vhod dobi specificirane parametre, kot je prikazano na sliki 15 (public_key). Vsak vhodni parameter ima določen tip in mesto, kamor naj ta parameter shrani. Parameter je možno shraniti na tri mesta, in sicer:

- Pomnilnik (angl. *storage*), ki shrani parameter v polje pomnilnik (angl. *storage*), ki je del vsakega računa in se ohranja med klici funkcij.
- Spomin (angl. *memory*), pri čemer je parameter shranjen v spomin in je dostopen samo med izvedbo trenutne transakcije, operacija pa je cenejša, kot shranjevanje v pomnilnik in zato predstavlja priporočljivo izbiro.
- Sklad (angl. *stack*), ki omogoča dostop le do zadnjega parametra., pri čemer je cena shranjevanja na sklad enaka ceni shranjevanja v spomin [16].

Podatki, ki so navedeni kot parametri transakcij, so razbrani iz polja *podatki*, ki je del vsake transakcije [16]. Vhod v transakcijo predstavljajo tudi vhodni podatki, ki so razbrani iz drugih polj transakcije, kot sta na primer pošiljana vrednost Ether-ja (msg.value) in naslov pošiljatelja (msg.sender), ki je tudi uporabljen v funkciji na sliki 15.

```
function addPublicKey(
  string memory _public_key
) public {
    publicKey[msg.sender] = _public_key;
  }
```

Slika 16: Funkcija za dodajanje javnega ključa.

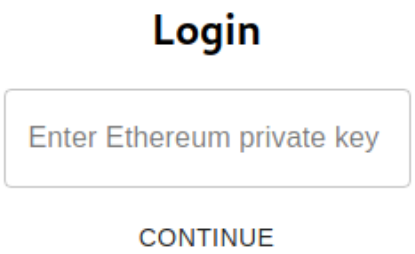
5.4 Programska vmesnika za Ethereum in IPFS

Ethereum ponuja vmesnik za programiranje aplikacij (angl. *Application programming interface - API*), ki se imenuje web3.js in vsebuje kolekcijo knjižnic, ki dovolijo interakcijo z Ethereum-ovim vozliščem [41]. API ponuja možnosti kot so povezava na vozlišče v omrežju in možnosti, ki nam pomagajo pri uporabljanju računa in ustvarjanju transakcij. Knjižnica ponuja dve možnosti za interakcijo s pametno pogodbo. Ena od možnosti je interakcija preko ustvarjanja transakcije z metodo pošlji (angl. *send*). Metoda pošlji je uporabljena za izvedbo funkcij znotraj pametne pogodbe, ki spremenijo stanje verige in za izvedbo potrebujejo gas. Druga možnost interakcije s pametno pogodbo je preko metode klic (angl. *call*). Z metodo klic lahko izvajamo funkcije pametne pogodbe, ki ne spremenijo stanja verige blokov in delujejo kot bralci oziroma nekaj izračunajo in vrnejo rezultat. Funkcije klicane z metodo klic ne potrebujejo kreiranja transakcij, saj ne spremenijo stanja verige blokov. Funkcije, ki se kličejo z metodo klic, se izvajajo samo na vozlišču, na katerega je aplikacija neposredno povezana in za izvedbo ne potrebujejo nič gas-a. Nobeno drugo vozlišče s klicem ne potrebuje biti seznanjeno, saj funkcije ne spremenijo stanja verige.

Tako kot Ethereum tudi IPFS ponuja svoj API, ki omogoča vrsto funkcij za interakcijo z omrežjem. V prototipu se IPFS API uporablja za povezavo na vozlišče, nalaganje in branje datotek. Pri branju datoteke API vrača bloke datoteke v pravilnem vrstnem redu.

5.5 Spletna stran

Spletna stran je sestavljena iz dveh strani, in sicer prijavne stvari in nadzorne plošče, kjer uporabnik upravlja z svojimi datotekami. Na prijavno stran uporabnik prispe ob obisku aplikacije, kjer je potrebno vpisati zasebni ključ Etherum-ovega računa. Aplikacija ne zahteva registracije, saj za vpis uporabnik potrebuje le Ethereum-ov račun, ki je unikaten za vsakega uporabnika in ponuja vse, kar je potrebno za delovanje aplikacije.



Slika 17: Prijavna stran.

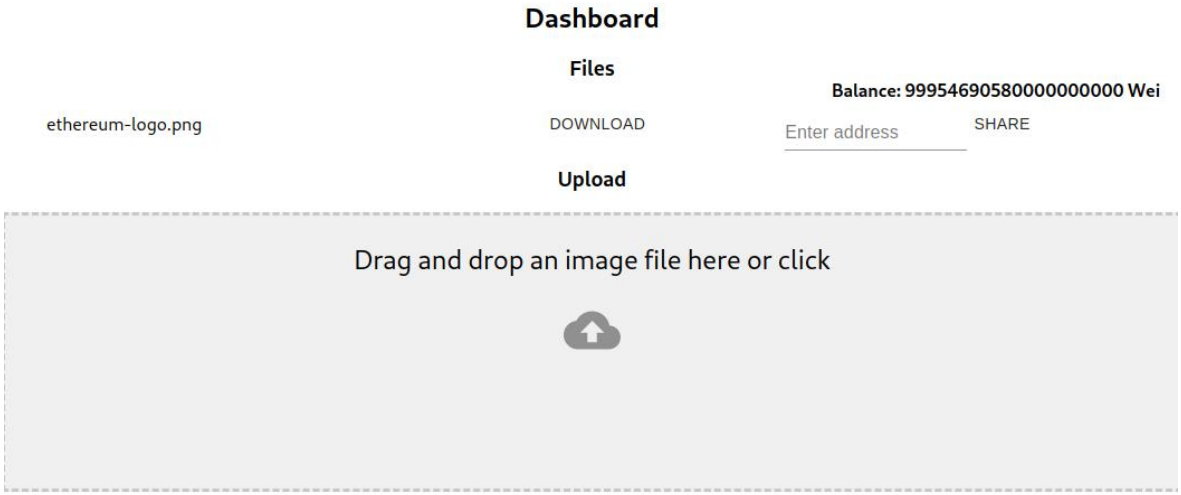
Ko uporabnik pritisne gumb *naprej* (angl. *Continue),* se aplikacija poveže s polnim vozliščem v verigi blokov in vozliščem v sistemu IPFS. Ethereum-ovo vozlišče, na katerega se aplikacija poveže, bo torej prvo, ki bo prejelo ustvarjeno transakcijo in jo preverilo. Če je transakcija veljavna, jo bo dodalo v bazen nepotrjenih transakcij in razpršilo naprej po omrežju, če transakcija ni veljavna, pa bo transakcijo zavrglo. Na podoben način bo delovalo tudi vozlišče IPFS, na katerega se aplikacija poveže. Vozlišče bo razdelilo datoteko na bloke in jo shranilo tako v svoj pomnilnik, kot tudi na nekaj bližnjih vozlišč.

Po prijavi torej aplikacija prejme zasebni ključ računa. Kot je pojasnjeno v poglavju 2.3.2.3, Ethereum uporablja kriptografijo elpitične krivulje, ki omogoča, računanje javnega ključa in naslova računa iz zasebnega ključa. Javni ključ ni poznan verigi blokov, dokler se ne ustvari transakcija, saj to reši teoretični varnostni problem, ki predpostavlja, da bi bilo v primeru luknje v kriptografiji eliptične krivulje možno dobiti zasebni ključ in s tem tudi dostop do upravljanja z računom. Aplikacija narejena v okviru zaključne naloge sicer krši omenjeno varnostno pravilo, saj po prijavi ustvari transakcijo, ki doda javni ključ na verigo blokov preko pametne pogodbe. Javni ključ je uporabljen za funkcijo deljenja datotek z drugimi uporabniki. Za kršitev protokola sem se odločil, ker je nevarnost le teoretična, vendar pa je zato priporočljivo, da se potencialni uporabniki ne prijavijo s svojim glavnim računom, ki ga uporabljajo za večino transakcij, vendar za uporabo aplikacije ustvarijo račun z minimalnim številom Ether-ja, potrebnim za uporabljanje aplikacije.

Ob vsaki ustvarjeni transakciji mora uporabnik med drugim navesti prejemnika. V aplikaciji so vse transakcije namenjene pametni pogodbi. Naslov pametne pogodbe postane znan ob inicializacijski transakciji z naslovljenim prejemnikom 0x0. Inicializacijsko transakcijo pogodbe ustvari račun v zunanji lasti, ki tudi plača stroške transakcije.

Po prijavi in uspešnem dodajanju javnega ključa na verigo blokov uporabniku aplikacija omogoči dostop do nadzorne plošče, ki dovoli nalaganje in prenos datotek. Poleg nalaganja in prenašanja datotek ima uporabnik tudi možnost deljenja datotek. Poleg omenjenih

funkcij aplikacija uporabniku tudi pokaže trenutno stanje na računu. Stanje je prikazano v enoti Wei, ki je manjša enota za Ether. (1 ETH = 10^{18} Wei).



Slika 18: Nadzorna plošča.

Nalaganje datotek se začne z izbiranjem datoteke na uporabnikovem računalniku. Ko je datoteka naložena, aplikacija ustvari simetričen kriptografski AES ključ za šifriranje datotek. Ključ za šifriranje je za vsako datoteko drugačen, kar omogoči enostavno deljenje datotek z drugimi uporabniki. Poleg AES ključa aplikacija generira tudi naključen inicializacijski vektor. Zatem šifrira datoteko z uporabo zgeneriranega ključa in incializijskega vektorja. Šifrirana datoteka je potem naložena na sistem IPFS. S šifriranjem datoteke je doseženo, da nihče drug razen uporabnika ne more brati vsebine datoteke. Ko je datoteka naložena, IPFS vrne prstni odtis datoteke, pod katerim je datoteka shranjena v omrežju. Ko je datoteka dodana na IPFS, mora aplikacija na verigo blokov shraniti še prstni odtis, ime, tip datoteke, AES ključ, ki ga je aplikacija uporabila za šifriranje in inicializacijski vektor.

Šifriranje datoteke ne bi imelo smisla, če bi bil ključ, ki je uporabljen za šifriranje datoteke, shranjen neposredno na verigo blokov, kjer je dostopen vsem, zato ga je potrebno šifrirati. AES ključ je šifriran s kriptografijo eliptične krivulje, in sicer je za šifriranje uporabljen javni ključ računa. Zaradi potencialne dodatne varnosti je šifriran tudi inicializacijski vektor, ki ga sicer ni nujno šifrirati. Po uspešnem nalaganju datoteke se stran osveži in prikaže naloženo datoteko pod naslovom *datoteke* (angl. *files*). Za vsako datoteko ima uporabnik dve možnosti, in sicer prenos ter deljenje datoteke. Vse informacije o naloženih datotekah prejme preko klica funkcije pametne pogodbe. Klic funkcije za zahtevo trenutnega stanja v verigi ne stane nič gas-a, saj metoda samo prebere trenutno stanje.

Prenos datoteke poteka v obratni smeri kot nalaganje datotek. V tem koraku aplikacija pozna že vse informacije o datotekah, ki so prenesene iz verige blokov. Ko uporabnik pritisne gumb *prenos* (angl. *download*), se najprej dešifrirajo kriptografski ključi, ki so bili uporabljeni za šifriranje datoteke. Dešifiranje ključev se naredi z algoritmom eliptične krivulje, in sicer je kot kriptografski ključ uporabljen zasebni ključ. Če je dešifriranje uspešno, se datoteka preko prstnega odtisa zahteva iz omrežja IPFS. Ko je datoteka prejeta, se ta dešifrira in nato shrani na uporabnikov računalnik. Aplikacija omogoča nalaganje datoteke kateregakoli tipa.

Pri deljenju datoteke mora uporabnik navesti naslov računa prejemnika in pritisniti gumb *deli* (angl. *share*). Prvi korak pri deljenju datoteke je klic funkcije pametne pogodbe, in sicer je za deljenje potreben prejemnikov javni ključ. Javni ključ bo nedosegljiv, če se prejemnik še nikoli ni prijavil v aplikacijo, zato bo v tem primeru aplikacija vrnila napako in deljenje ne bo mogoče. Naslednji korak je dešifriranje ključa in inicializacijskega vektorja, ki sta bila uporabljena pri nalaganju datoteke. Dešifriran ključ in inicializacijski vektor aplikacija šifrira s prejemnikovim javnim ključem in ju skupaj s prstnim odtisom, imenom in tipom datoteke preko transakcije zapiše na verigo blokov. Pri tem postopku postane jasno, da je pomembno, da je vsaka datoteka šifrirana s svojim AES ključem, saj bi bilo v nasprotnem primeru ob deljenju datoteke prejemniku le-te omogočeno, da si ogleda vse datoteke pošiljatelja, saj bi s tem izvedel tudi pošiljateljev AES ključ. Prejemnik lahko po prejetju datoteke z njo upravlja na enak način kot pošiljatelj.

Po vsaki transakciji aplikacija osveži trenutno stanje Ethereum-a, kar uporabniku omogoča, da si lažje predstavlja, koliko ETH zapravi za vsako transakcijo, saj se za potrebe transakcije ETH spremeni v gas.

5.6 Ovrednotenje prototipa

Ena glavnih pridobitev tehnologije veriženja blokov je transparentnost in z njo zaupanje v aplikacijo. Arhitektura sistema omogoča, da uporabnik zagotovo dobi enako datoteko kot jo je naložil, saj IPFS uporablja vsebinsko naslovljeno iskanje datotek, zato mora aplikacija za prenos datoteke poznati njen prstni odtis, ki je shranjen na verigi blokov in je posledično nespremenljiv.

V aplikaciji bi bilo možno dodati še eno funkcijo, in sicer shranjevanje prstnega odtisa nešifrirane datoteke, s čimer bi izvedeli, ali je neka datoteka naložena na IPFS duplikat že obstoječe ali ne, hkrati pa bi lahko izvedeli tudi originalni izvor datoteke. Trenutno je možno izvedeti izvor datoteke le pri deljenju datoteke in ne tudi takrat, ko uporabnik še enkrat naloži enako datoteko. Aplikacija od uporabnika zahteva zasebni ključ računa, pri

čemer je skrb za zasebni ključ na uporabnikovi vesti, zato mora biti zelo previden, kateri aplikaciji ga zaupa. Zaupanje je možno pridobiti s transparentnostjo, ki ne bi smela obstajati le na nivoju tehnologije veriženja blokov, temveč tudi spletni strani, ki je dosežena z objavo kode kot odprtokodne, kjer lahko uporabniki razvijalcem tudi postavljajo vprašanja ali jih opozorijo, če najdejo potencialno varnostno luknjo. Obenem so uporabniki lahko prepričani, da se koda v aplikaciji ne bo spremenila, saj je nemogoče spremeniti tako pametno pogodbo, kot tudi aplikacijo saj bo, ko spremenimo del kode, aplikacija na IPFS objavljena pod drugim prstnim odtisom. S tem bo uporabnik takoj obveščen o spremembi aplikacije.

V sistemu je največja pomanjkljivost ta, da vozlišča v IPFS lahko pozabijo na naloženo datoteko, če se datoteke dlje časa ne zahteva, saj jih bo zbiralec smeti (angl. *garbage collector*) izbrisal iz vozlišč. Brisanju datotek bi se lahko izognili z motiviranjem vozlišč v sistemu IPFS, da označijo datoteko kot trajno shranjeno na vozlišče.

Motiviranje vozlišč, da trajno shranjujejo datoteko bi bilo lahko doseženo s plačevanjem pri prenašanju datoteke za vsak prenešen blok in plačevanjem vozlišč glede na trajanje hranjenja datoteke. Za spremljanje trajanja hranjenja datoteke bi potrebovali poseben algoritem, ki bi deloval kot dokazilo o lastništvu. Eno od možnih dokazil bi bilo lahko narejeno s pogodbo preko pametne pogodbe, kjer bi se uporabnik z gostiteljem datoteke dogovoril za ceno hranjenja datotek za določeno časovno obdobje. Najemnik prostora dogovorjeno vsoto nakaže na pametno pogodbo, ki bo denar tekom dogovorjenega obdobja nakazala gostitelju. Pametna pogodba bi na vsakih N zapečatenih blokov od gostitelja datotek zahtevala dokaz, da še vedno shranjuje datoteko.

Sistem Sia je eden od obstoječih sistemov za porazdeljeno shranjevanje datotek. Dokaz lastništva v sistemu Sia je sestavljen iz dela datoteke in delov Merklovega drevesa. Gostitelj dokaz pošlje v transakciji. Vozlišča ob prejetju transakcije zaženejo kodo v pametni pogodbi, ki preveri, ali se dokaz dopolnjuje v pravi koren Merklovega drevesa. Če se dokaz dopolnjuje v ustrezni koren, je lastništvo datoteke dokazano. Pametna pogodba mora torej samo poznati datotečni koren Merklovega drevesa. Za vsak veljaven dokaz gostitelj datotek prejme določeno vsoto kovancev. Če dokaz ni veljaven, je gostitelj kaznovan za določeno vsoto kovancev, zniža se mu ugled, potencialno pa ga je mogoče tudi izločiti iz omrežja. Sprotni dokazi, torej dokazi vsakih N blokov, tudi preverijo, koliko časa je vozlišče na voljo, Če vozlišče ni na voljo, ne bo zaznalo, da mora poslati dokaz, zato bo njegov ugled padel. Dober ugled je pomemben pri sklepanju pogodb, saj so datoteke, ki jih gostitelj gosti, uporabniku na voljo samo takrat, ko je gostitelj na voljo. Posledično bo vozlišče, ki ima slab ugled, težko dobilo stranke [10].

6 ZAKLJUČEK

Dokument predstavlja pregled tehnologij, ki bi lahko v prihodnosti preoblikovale arhitekturo interneta, kot ga poznamo danes. Mislim, da bo zasebnost vedno bolj v ospredju, kar bo tehnologijam, ki slonijo na porazdeljeni arhitekturi, dalo še večji pomen. Vendar pa imajo tehnologije kot je IPFS še veliko varnostnih lukenj in vprašanj, večina sistemov pa je dostopna šele v testni verziji. Porazdeljeni sistemi prinašajo veliko vprašanj glede varnosti v omrežju. Tehnologija veriženja blokov je v tej smeri revolucionarna in je v kombinaciji z obstoječo arhitekturo že uporabljena v nekaterih sistemih, več sistemov pa bo to tehnologijo verjetno uporabljalo kmalu. Prehod na novo arhitekturo se ne bo zgodila čez noč, vendar bo tehnologija v ospredje prihajala postopno. Nova doba bi nam prinesla stalno dostopnost do virov, dostop do osebnih podatkov bo zaradi uporabe kriptografskih tehnik v uporabnikovih rokah, obenem pa tehnologije zagotavljajo tudi integriteto podatkov.

Tehnologija veriženja blokov predstavlja dobro izhodišče za aplikacije, ki bodo skozi transparentnost gradile na zaupanju v produkt. Aplikacija narejena v okviru projektne naloge je le ena izmed njih. Uresničitev le-te bi velike podatkovne centre zamenjala z navadnimi hišnimi računalniki, kar bi znižalo ceno hranjenja podatkov in zaradi redundance in transparentnosti predstavljalo bolj varen način shranjevanja datotek. Menim, da se tehnologije za spletno hranjenje datotek, ki so grajene na modelu odjemalec-strežnik, dandanes med uporabniki še vedno pogosto uporabljajo predvsem zaradi nepoznavanja tehnologije veriženja blokov. Obenem obstoječe tehnologije zaenkrat omogočajo več možnosti za manipulacijo z datotekami in boljšo uporabniško izkušnjo, vseeno pa menim, da ima tehnologija veriženja blokov velik potencial in širok spekter uporabe.

7 LITERATURA IN VIRI

[1] A. Hertig, *Ethereum 101*, 2017, Dostopno na https://www.coindesk.com/learn/ethereum101/how-ethereum-works (datum dostopa 30.4.2020).

[2] A. M. Antonopoulos in G. Wood, *Mastering ethereum: building smart contracts and dapps*, O'reilly Media, California, 2018.

[3] A. M. Antonopoulos, *Mastering bitcoin: Programming the open blockchain*, O'reilly Media, California, 2017.

[4] A. Rosic, *Centralized vs Decentralized Storage: Redefining Storage Solutions with Blockchain Tech*, 2020, Dostopno na https://blockgeeks.com/guides/centralized-vs-decentralized-storage-redefining-storage-solutions-with-blockchain-tech/ (datum dostopa 30.4.2020)

[5] B. Asolo, *scriptPubKey & scriptSig Explained*, 2018, Dostopno na https://www.mycryptopedia.com/scriptpubkey-scriptsig/ (datum dostopa 30.4.2020).

[6] B. Curran, *Comparing Bitcoin & Ethereum: UTXO vs Account Based Transaction Models*, 2018 (datuma dostopa 30.4.2020).

[7] C. Bavec, *Osnove managementa informacijskih tehnologij*, Univerza na primorskem, FAMNIT, Koper, 2019.

[8] *Cloud computing: A complete guide*, Dostopno na https://www.ibm.com/cloud/learn/cloud-computing (datum dostopa: 30.4.2020).

[9] Codenotary, *What are Digital Signatures and How do They Work?*, 2019, Dostopno na https://medium.com/faun/what-are-digital-signatures-and-how-do-they-work-195b18c4f42c (datum dostopa: 30.4.2020).

[10] D. Vorick in L. Champine, *Sia: Simple Decentralized Storage*, 2014, Dostopno na https://sia.tech/sia.pdf (datum dostopa: 30.4.2020).

[11] *Distributed Hash Tables (DHTs)*, 2020, Dostopno na https://docs-beta.ipfs.io/concepts/dht/ (datum dostopa 30.4.2020).

[12] E. Hong, *How Does Bitcoin Mining Work?*, 2020, Dostopno na https://www.investopedia.com/tech/how-does-bitcoin-mining-work/, (datum dostopa: 30.4.2020)

[13] G. Wood, *Ethereum: A secure decentralised generalised transaction ledger, Ethereum project yellow paper*, 2014, Dostopno na https://ethereum.github.io/yellowpaper/paper.pdf (datum dostopa: 30.4.2020).

[14] I. Bashir, *Mastering Blockchain: Distributed ledger technology, decentralization, and smart contracts explained*, Packt Publishing Ltd, Birmingham, 2018.

[15] Information Resources Management Association, *Digital Currency: Breakthroughs in Research and Practice*. IGI Global, New York City, 2018.

[16] *Introduction to Smart Contracts*, Dostopno na https://solidity.readthedocs.io/en/develop/introduction-to-smart-contracts.htm (datum dostopa 30.4.2020)

[17] J. Benet, *Ipfs-content addressed, versioned, p2p file system (draft 3)*, arXiv preprint, 2014.

[18] J. Frankenfield, Merkle Tree, 2020, Dostopno na https://www.investopedia.com/terms/m/merkle-tree.asp (datum dostopa 30.4.2020).

[19] J. Frankenfield, Proof of Work, 2018, Dostopno na https://www.investopedia.com/terms/p/proof-work.asp (datum dostopa 30.4.2020).

[20] K. Dugan, *Blockchain for Beginners: Understand the blockchain basics and the foundation of Bitcoin and cryptocurrencies*, Independently published, 2018. [blockchainBeginners]

[21] M. D'Aliessi, *How Does the Blockchain Work?*, 2016, Dostopno na https://onezero.medium.com/how-does-the-blockchain-work-98c8cd01d2ae (datum dostopa: 30.4.2020).

[22] M. Dufel, *Distributed Hash Tables And Why They Are Better Than Blockchain For Exchanging Health Records*, 2017, Dostopno na https://medium.com/@michael.dufel_10220/distributed-hash-tables-and-why-they-are-better-than-blockchain-for-exchanging-health-records-d469534cc2a5 (datum dostopa 30.4.2020).

[23] M. Dworkin, *Recommendation for block cipher modes of operation. methods and techniques*, NIST, Gaithersburg, 2001.

[24] M. Gates, *Blockchain: Ultimate guide to understanding blockchain, bitcoin, cryptocurrencies, smart contracts and the future of money*, Wise Fox Publishing, 2017.

[25] M. Grincalaitis, *Mastering Ethereum*, Packt Publishing Ltd, Birmingham, 2019. [25]

[26] M. Rouse, *asymmetric cryptography (public key cryptography)*, 2020, Dostopno na https://searchsecurity.techtarget.com/definition/asymmetric-cryptography (datum dostopa: 30.4.2020).

[27] M. Swan, *Blockchain: Blueprint for a new economy*, O'reilly Media, California, 2018.

[28] *Merkle Distributed Acyclic Graphs (DAGs)*, 2020, Dostopno na https://docs-beta.ipfs.io/concepts/merkle-dag/ (datum dostopa 30.4.2020).

[29] N. Antonopoulos, *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications: Models, Methodologies and Applications*, IGI Global, New York City, 2010.

[30] P. Siriwardena *The Mystery Behind Block Time*, 2017, Dostopno na https://medium.facilelogin.com/the-mystery-behind-block-time-63351e35603a (datum dostopa 30.4.2020).

[31] P. Timmalacherla, *The Top 3 Issues of The Centralized Internet*, 2019, Dostopno na: https://medium.com/@IamPremt/the-top-3-issues-of-the-centralized-internet-1db59d5e495e, (datum dostopa 30.4.2020)

[32] *Proof of stake FAQ*, 2019, Dostopno na https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ (datum dostopa 30.4.2020).

[33] Q. H. Vu, M. Lupu, in B. C. Ooi, *Peer-to-peer computing: Principles and applications*, Springer, New York City, 2009.

[34] R. Ozier, *Ethereum - The Insider Guide to Blockchain Technology, Cryptocurrency and Mining* Ethereum, CreateSpace Independent Publishing Platform, 2017.

[35] S. Raval, *Decentralized applications: Harnessing Bitcoin's blockchain technology*, O'reilly Media, California, 2016.

[36] Sheinix, *What is a Bitcoin address, 2018, Dostopno na* https://medium.com/coinmonks/what-is-a-bitcoin-address-6c822c857004 (datum dostopa 30.4.2020).

[37] T. Laurence, *Introduction to Blockchain Technology: The many faces of blockchain technology in the 21st century*, The Netherlands, Van Haren Publishing, 2019.

[38] *Transaction*, 2019, Dostopno na https://en.bitcoin.it/wiki/Transaction (datum dostopa 30.4.2020).

[39] V. Buterin, *Ethereum white paper: a next generation smart contract & decentralized application platform*, 2014, Dostopno na https://github.com/ethereum/wiki/wiki/White-Paper (datum dostopa: 30.4.2020).

[40] W. Stallings, *Cryptography and network security: Principles and Practice*, Sedma izdaja, Pearson Education Limited, Harlow, 2017.

[41] *web3.js - Ethereum JavaScript API*, Dostopno na https://web3js.readthedocs.io/en/v1.2.6/ (datum dostopa 30.4.2020)

[42] *What is IPFS?*, 2019, Dostopno na https://docs-beta.ipfs.io/concepts/what-is-ipfs (datum dostopa 30.4.2020).

[43] X. Xu, I. Weber, in M. Staples, *Architecture for blockchain applications: Principles*. Springer, Berlin, 2019.