LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

BLOCKCHAIN TEHNOLOĢIJĀ BALSTĪTU SISTĒMU IZVEIDE AR ETHEREUM

BAKALAURA DARBS

Autors: **Mihails Ļevenčiks**
Studenta apliecības Nr.: ml12059
Darba vadītājs: Dr. dat. Guntis Arnicans

RĪGA 2016

Anotācija

Darba mērķis ir Ethereum decentralizētas programmatūras platformas pētīšana. Darbā tiek pētītas tehnoloģijas uz kurām balstās decentralizētās programmatūras platforma, programmatūras izstrādes process tajā un platformas stiprās un vājās puses. Darba praktiskajā daļā, tiek izveidots decentralizētas kopfinansēšanas sistēmas prototips Ethereum platformā.

Atslēgvārdi: Blockchain tehnoloģija, Ethereum platforma, decentralizēta programmatūras platforma, kopfinansēšanas sistēma

Abstract

Development of system based on Blockchain technology using Ethereum

Goal of paper is Ethereum decentralized software examination. Technologies on which decentralized software platforms are made, software development processes within them and strengths and weaknesses of a decentralized platform are examined. In the practical part of the paper decentralized crowdfunding system prototype is developed on Ethereum platform.

Keywords: Blockchain technology, Ethereum platform, decentralized software platform, crowdfunding system

SATURS

APZĪMĒJUMI UN SKAIDROJUMI

Apzīmējums	Skaidrojums
Assembly programmēšanas valoda	Zema līmeņa programmēšanas valoda, kas ir cilvēklasāma mašīnvalodas notācija. Dažādu datoru arhitektūru asamblervalodas nav savietojamas savā starpā.
Atom	Lietojumprogrammas koda redaktors.
Atslēgas-vērtības pāris	Pāris, kurš parasti tiek lietots masīvos un dod iespēju pievienot pāri, meklēt, rediģēt un dzēst pāri pēc tā atslēgas.
Altcoin	Pirmā alternatīva kriptovalūta.
Alternatīva kriptovalūta	Jebkura kriptovalūta, kura nav Bitcoin kriptovalūta.
Blockchain	Savā starpā saistītu bloku virkne, kurā katrs bloks norāda uz iepriekšējo.
Bloks (angļu val. Block)	Datu ierakstu kopums.
Bitcoin	Decentralizēta starplietotāju (peer-to-peer) tiešsaistes maksājumu sistēma ar pilnīgi digitālu valūtu, kuru sauc par Bitcoin.
Bitcoin Weekly	Tīmekļa vietne ar iknedēļās publikācijām par Bitcoin un ar to saistīto sabiedrību.
Bitcoin Magazine	Tīmekļa vietne ar ikmēneša publikācijām, kas aptver visus Bitcoin aspektus un ar to saistīto sabiedrību.
BitTorrent	Failu izplatīšanas protokols, kurā failu lietotāju datori ir apvienoti vienā tīklā un izplata failus sava starpa bez centrāla servera palīdzības.
Casper protokols	GHOST protokola modifikācija, ekonomikas konsensu jeb vienošanas protokols, kas bāzēts uz drošības depozītiem.
C programmēšanas valoda	Zema līmeņa standartizēta programmēšanas valoda.
Decentralizēts	Decentralizēšanas procesa rezultāts, kurā funkcijas, pilnvaras, cilvēki vai kaut kas cits tiek pārdalītas no centrālā atrašanās vietas.
DDoS uzbrukums	Uzbrukums, kas tiek veikts, lai pārtrauktu vai traucētu tīmekļa vietnes, servera vai cita tīmekļa resursa darbību.
Dropbox	Failu izvietošanas pakalpojums, kurš piedāvā datu glabāšanu kompānijas serverī, failu sinhronizāciju un klienta

	programmatūru.
Dapple	Gudro līgumu izstrādes vide, kura atbalsta Solidity programmēšanas valodu.
DNS protokols	Domēnu nosaukumu sistēma (DNS) ir protokols, kas pārveido domēnu nosaukumus par skaitliskajām IP adresēm.
Ethereum	Platforma decentralizētu lietojumprogrammu veidošanai, kas balstīta uz Blockchain tehnoloģiju.
EthereumJS TestRPC	Ethereum platformas klients lietojumprogrammas izstrādei un testēšanai.
Eth kriptovalūta	Ethereum platformas kriptovalūta.
EVM (Ethereum Virtual Machine)	Ethereum platformas virtuālā mašīna.
Eris-PM	Gudro līgumu izstrādes vide.
Embark	Decentralizētas lietojumprogrammas izstrādes vide.
Emacs	Daudzfunkcionālu paplašinājumu teksta redaktoru paaudze.
E-pasts	Elektroniskais pasts ir ziņojumu sastādīšanas, nosūtīšanas un saņemšanas metode, izmantojot elektroniskās sakaru sistēmas.
Go programmēšanas valoda	Formāla valoda, kas kalpo datorprogrammu aprakstam.
GHOST (Greedy Heaviest Observed Subtree)	Blockchain virknes izvēles noteikums.
Google drive	Failu glabāšanas un sinhronizācijas pakalpojums.
Gudrie līgumi (smart contracts)	Algoritms, kas apraksta nosacījumu kopumu, kuru izpilde ietekme uz noteikumiem reālā dzīvē vai digitālas sistēmas.
Izpildes pierādījums (angļu val. Proof-of-work)	Bloka izveides algoritms, kas speciāli izveidots priekš Ethereum.
Indiegogo	Centralizēta starptautiska kopfinansēšanas sistēma.
IP adrese	Unikāls kādas ierīces, kura ir pieslēgta tīklam, identifikators.
Jaucējfunkcija	Algoritms, kas pārveido dažāda garuma tekstu fiksēta garuma izvadē.
JavaScript programmēšanas valoda	Skriptu programmēšanas valoda, kas balstīta uz prototipu koncepta.
Kickstarter	Centralizēta amerikāņu kopfinansēšanas sistēma
Kriptovalūta	Digitālās naudas izteiksmes līdzeklis, kurai nav centrālās

	autoritātes.
Likvīds	Ekonomikas termins, kas apzīmē aktīvu īpašību būt ātri pārdodamiem par cenu, kura ir tuva tirgu cenai.
LLL programmēšanas valoda	Zema līmeņa programmēšanas valoda gudro līgumu izstrādei.
Lisp programmēšanas valoda	Loģiskās programmēšanas valoda.
Linkable ring signatures	Grupas paraksta shēma, kurā grupas loceklis paliek anonīms parakstot kādus datus, jo tas parakstās no grupas vārda.
Mutan programmēšanas valoda	Gudro līgumu augsta līmeņa programmēšanas valoda, līdzīga C programmēšanas valodai.
Merkle koks	Koks (datu struktūra) kriptogrāfijā un datorzinātnē, kurā mezglos atrodas visu apakšmezglu jaucējkods.
Mix izstrādes rīks	Oficiāls Ethereum platformas izstrādes rīks, kurš atbalsta lietojumprogrammas izveidi un testēšanu Solidity programmēšanas valodā.
Miner lietotājs	Lietotājs, kas veido skaitļošanas operācijas Blockchain tīklā.
npm (Node Project Manager)	JavaScript lietojumprogrammu pakešu menedžeris.
Node.js	Platforma, kas galvenokārt paredzēta tīkla lietotņu veidošanai, kā, piemēram, tīmekļa serveris.
Namecoin	Patvaļīgo atslēgas-vērtību kombināciju uzglabāšanas sistēma bāzēta uz Bitcoin tehnoloģijas.
Nonce	Konta skaitlis Ethereum platformā, kas ir vienāds ar konta, kam Nonce pieder, nosūtīto transakciju skaitu.
OS X	Grafisku operētājsistēmu līnija, kuru izstrādā un tirgo kompānija Apple.
Populus	Ethereum gudro līgumu izstrādes vide.
Python programmēšanas valoda	Augsta līmeņa objektorientētā programmēšanas valoda.
SHA-256	Divu līdzīgu jaucējfunkciju saime, ar dažādiem bloka izmēriem
Seprent programmēšanas valoda	Augsta līmeņa programmēšanas valoda gudro līgumu izstrādei.
Solidity programmēšanas valoda	Augsta līmeņa programmēšanas valoda ar JavaScript sintaksi, kas paredzēta Ethereum platformas gudro līgumu izstrādei.
sudo	Programma priekš sistēmas administrēšanas UNIX sistēmās.

SublimeText	Starpplatformu teksta redaktors ar Python lietojumprogrammu programmēšanas interfeisu.
Sharding	Datu bāzes mērogojamības tehnoloģija, kuras būtība ir sadalīt datu bāzi tā, lai katru daļu varētu iznest atsevišķā serverī.
Tjūringa pilnīga valoda	Valoda, kurā iespējams pilnībā simulēt universālo Tjūringa mašīnu.
Truffle izstrādes rīks	Ethereum platformas lietojumprogrammu izstrādes un testēšanas vide.
UNIX	Daudzuzdevumu un daudzlietotāju operētājsistēma, ko sākotnēji izstrādāja AT&T Bell Labs darbinieki. Mūsdienās UNIX ir sadalījusies vairākos atzaros.
Ubuntu	Linux operētājsistēmas distributīvs, kas pamatā orientēts lietošanai personālajos datoros.
VisualStudio	Integrētā izstrādes vide no kompānijas Microsoft.
Visa	Starptautiska maksājumu sistēma.
Windows	Microsoft kompānijā radīta operētājsistēma personāliem datoriem un serveriem.
Zibenīgs tīkls (angl. val. lightning network)	Decentralizēts tīkls, kurš ļauj veidot maksājumus nekavējoties.

IEVADS

Mūsdienās eksistē daudz dažādas informācijas sistēmas un lielākā daļa no tām ir centralizētas. Centralizētajām sistēmām ir liels bojājuma faktors ņemot vērā to, ka tajās eksistē vienots konkrēts kļūmes punkts – ugunsmūris vai tīmekļa vietne. Lietojot centralizēto sistēmu lietotājam ir jāuztic savu informāciju un personiskos līdzekļus kādai personai vai organizācijai. Decentralizētajās sistēmās, gluži pretēji, neeksistē persona vai organizācija, kurai lietotājs būtu spiests uzticēties. Blockchain tehnoloģija ir decentralizētas sistēmas pamats. Pēdējos gados digitālā valūta, kura balstās uz Blockchain tehnoloģiju, kļuvusi plaši izplatīta un populāra. Blockchain tehnoloģijai ir liels potenciāls un uz tās pamata attīstās arī citas platformas šāda veida platformas. Viena no šādam platformām ir Ethereum. Lietojumprogrammas, kas izstrādātas iekš Ethereum platformas, neprasa lietotāju uzticēt savus datus un personiskos līdzekļus izstrādātajam vai kam citam. Darba autors uzskata, ka tuvākajā nākotnē decentralizētas sistēmas kļūs par daļu no sabiedrības ikdienas un Ethereum platforma nav izņēmums, un līdz ar to bakalaura darba izvēlētā tēma ir vērā ņemami aktuāla. Darba tēmas pamatā ir programmatūras izstrādes procesa izpēte Ethereum platformā izpēte un kopfinansēšanas sistēmas prototipa izstrāde Ethereum vidē.

Darba autors izvirzījis mērķi izpētīt un saprast, kā darbojas Ethereum platforma, uz ko tā balstās un, izmantojot iegūtas zināšanas, izstrādāt kopfinansēšanas sistēmas prototipu uz Ethereum platformas. Uzdevumi, kuri ir jāveic, lai sasniegtu izvirzīto mērķi:

- Izpētīt Blockchain tehnoloģiju un to arhitektūru;
- Izpētīt Ethereum platformu un tās arhitektūru;
- Noskaidrot Ethereum platformas stiprās un vājās puses;
- Izstrādāt kopfinansēšanas sistēmas prototipu uz Ethereum platformas.

Darbā izmantotās pētniecības metodes ir literatūras analīze un eksperimentu veidošana. Sakarā ar to, ka Ethereum platforma vēl ir izstrādes stādīja un oficiālu rakstu par to gandrīz neeksistē, tad informācija tika ievākta no visa veida avotiem, priekšrocību sniedzot oficiālajiem avotiem. Ņemot to vērā, darbā aprakstītā informācija pēc kāda laika var kļūt neaktuāla.

1. BLOCKCHAIN TEHNOLOĢIJA

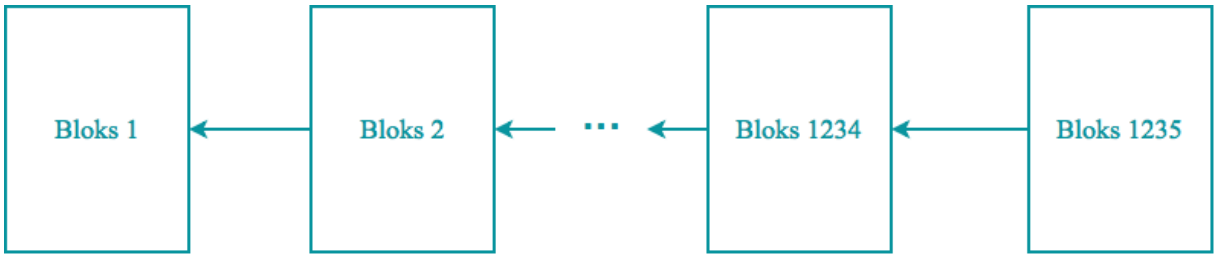
Mūsu dzīve ir tieši saistīta ar naudu un visa veida datiem un dokumentiem. Līdz ar to jākomunicē ar dažādiem starpniekiem, kuri izsniedz naudu, datus un dokumentus, pārbauda to autentiskumu, izsniedz kopijas, kam arī pārbauda uz autentiskumu. Nespējīgi ticēt viens otram, mums ir jātic starpniekiem, kuriem ticēt īstu iemeslu nav, ja tiek akcentēts dokumentu viltošanas un citi riski. Risināt uzticamības problēmu palīdz Blockchain tehnoloģija. Blockchain ir decentralizēta transakcijas datubāze, kura ir kopīga un pieder visiem tīkla mezgliem, kuru neviens nekontrolē un kuru visi tās lietotāji var atjaunināt un uzraudzīt.

Ethereum platforma ir balstīta uz Blockchain tehnoloģiju, tāpēc svarīgi izprast, kas ir Blockchain. Blockchain ir tehnoloģija, kas ļauj droši uzglabāt uzticamus ierakstus. Piemēram, iekš Blockchain var uzglabāt datus par naudas pārskatījumiem. Uz doto brīdi visizplatītākā sfēra, kurā tiek lietota Blockchain tehnoloģija ir kriptovalūta un vispopulārākā kriptovalūta ir Bitcoin. Kopā ar Bitcoin tika izstrādāta uz izpildes pierādījuma (angļu val. proof-of-work) bāzēta Blockchain koncepcija [2]. Kriptovalūtas Blockchain tiek lietots lai fiksēt informāciju par to, kurš, kuram un cik daudz digitālo naudu ir pārskaitījis. Bet iekš Blockchain arī var uzglabāt tirdzniecības licences, apdrošināšanas apliecinājumus, līgumus, datus par kredītiem, par īpašuma tiesības utml. Var teikt, ka visu, ko var uzrakstīt papīrā var ierakstīt iekš Blockchain, tikai Blockchain vidē nav iespējams aizvietot vai viltot ierakstus.

Blockchain koncepcija ir jauna organizēšanas paradigma datu atrašanai, vērtēšanai un visa veida uzskaitāmu vērtību pārskaitīšanai, un ar to iespējams visu cilvēka aktivitāti koordinēt daudz plašākā mērogā, nekā tas ir bijis iespējams līdz šim [1]. Šajā daļā ir aprakstīta Blockchain tehnoloģija un tās darbība.

1.1. Blockchain tehnoloģijas darbība

No tehnoloģijas nosaukuma var saprast, ka Blockchain ir virkne, kura sastāv no datu blokiem. Katrs datu bloks virknē ir saistīts ar iepriekšējo bloku un satur datu ierakstus (1.1. att.). Visi jaunie bloki vienmēr tiek pievienoti virknes beigās [2]. Turpmāk tekstā Blockchain, Blockchain virkne, bloku virkne un virkne tiek lietoti kā sinonīmi.



1.1. att. **Blockchain virkne**

Blockchain virkne ir balstīta uz trim principiem:

- Izplatība;
- Atklātība;
- Aizsardzība.

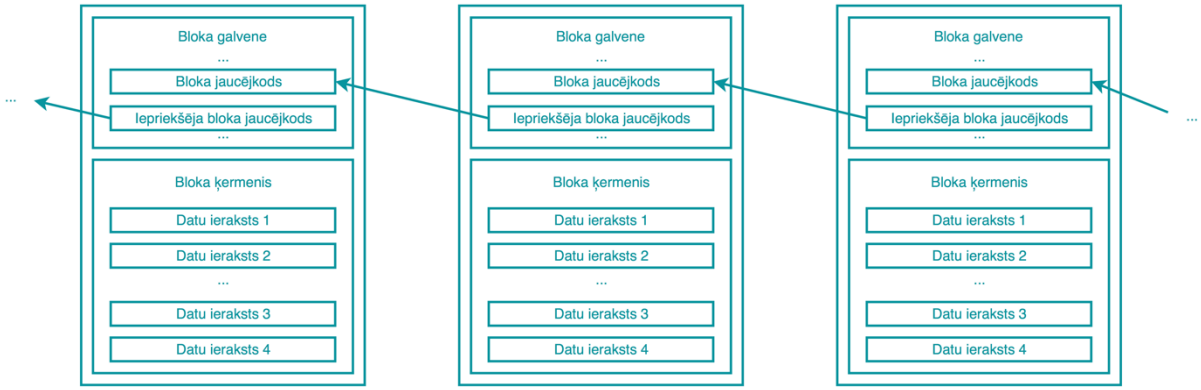
Visi Blockchain lietotāji kopā veido datoru tīklu, kur katrā datorā glabājas Blockchain virknes kopija. Parasti tā ir pilna bloku virknes kopija, bet pēc nepieciešamības var uzglabāt arī tikai daļu no virknes ar datiem, kuri ir nepieciešami konkrētajā datorā. Pamatojoties uz to Blockhain gandrīz nevar salauzt vai izslēgt, jo, kamēr vismaz viens dators uzglāba pilno bloku virkni, Blockchain eksistē. Katrs jauns lietotājs palielina un nostiprina tīklu. Blockchain tīklā visiem datoriem ir vienlīdzīgas tiesības. Visi datu bloki un datu ieraksti tajos ir vienmēr atvērti visiem. Jebkurš var viegli nolasīt datu bloku un redzēt datu ierakstus blokā. Līdz ar to visi dati iekš Blockchain ir viegli pārbaudāmi un tas nozīme, ka nav vajadzības uzticēties citiem tīkla lietotājiem, jo vienmēr var pārbaudīt un garantēti saņemt īsto atbildi. Blockchain vidē tiek lietota šifrēšana, lai aizsargātu tīkla lietotāju datus [1]. Tas palīdz datiem būt vienlaicīgi atklātiem un autentiskiem. To viegli var aprakstīt ar piemēru – visi var redzēt, ka kaut kādam pieder kāds īpašums, bet neviens nevar uzzināt kuram tieši tas pieder kamēr īpašnieks neiedos atslēgu, kas apliecinās, ka īpašums pieder tieši viņam.

1.2. Datu bloka struktūra

Datu bloks Blockchain virknē sastāv no galvenes un ķermeņa. Bloka ķermenis sastāv no saraksta ar datu ierakstiem (1.2. att.) [2].

1.2. att. **Blockchain datu bloka struktūra**

Iekš Blockchain bloki ir savienoti viens ar otru ar jaucējkodu. Katrā blokā galvenē glabājas iepriekšējā bloka jaucējkods. Tas nodrošina aizsardzību pret Blockchain uzlaušanas risku [2].



1.3. att. **Blockchain datu bloku savienojums**

Katra bloka jaucējkods ir aprēķināts pēc visa bloka datiem un iepriekšējā bloka atslēgas. Tas nozīme, ka katra bloka atslēgā ir iešifrēti ne tikai šī bloka dati, bet arī visi iepriekšēji bloki. Jebkāda neliela izmaiņa jebkurā blokā rada izmaiņas šī blokā jaucējkodā, kas savukārt prasa visu sekojošo bloku jaucējkodu maiņu. Tādejādi, redzot visu Blockchain un jaucējkodus, var pārbaudīt datus, pārliecinoties ka bloku secība ir pareiza un nav pazudis bloks vai nav ielikts jauns viltots bloks, kā arī pārbaudīt vai bloka jaucējkods atbilst tajā

uzglabātajiem datiem. Bloka jaucējkodam ir jāapmierina drošības noteikumi, kas nosaka tīkla aizsardzības līmeni. Aizsardzības līmenis var mainīties atkarība no tīkla izmēra un tā palielināšanas. Piemēram, Bitcoin pirmo bloku jaucējkods sākas ar desmit nullēm, tas nosaka jauno bloku izveides sarežģītības līmeni, ieskaitot gan viltus, gan korektos blokus. Kā tiek iegūts jaucējkods, kas sakas ar desmit nullēm ir aprakstīts 1.6. sadaļā.

1.3. Blockchain drošība

1.3.1. Publiskās un privātās atslēgas

Publiskās kriptogrāfijas īpašības:

- Tiek izveidots unikāls pāris no publiskās un privātās atslēgas;
- Publiskā atslēga var tikt pārsūtīta visiem tīkla lietotājiem;
- Privātā atslēga ir droši uzglabāta;
- Visi datu ieraksti ir šifrēti ar privātām atslēgām;
- Jebkurš var pārbaudīt datu ierakstu pēc autentiskuma, lietojot publisko atslēgu;
- Nav iespējams viltot datu ierakstus, nezinot privātās atslēgas [3, 4].

Kriptovalūtās, lai glabātu, saņemtu un sūtītu digitālo naudu ir vajadzīgs maks, kurš sastāv no publiska identifikatora (maka adreses), bilances un privātās atslēgas, kas ļauj veikt darbības ar maku. Publisko identifikatoru un privāto atslēgu var salīdzināt ar ieejas vārdu un paroli, jo ieejas vārdu var zināt arī citi, bet paroli ir uzmanīgi jāuzglabā. Blockchain gadījumā, pazaudējot privāto atslēgu, to nevarēs viņu atjaunināt. Katrs Bitcoin kriptovalūtas lietotājs var izveidot tik daudz makus, cik vēlas. Ar privāto atslēgu var iegūt attiecīgā maka adresi, bet otrādi nevar, par to atbild jaucējfunkcija.

Bitcoin tīklā publiska atslēga ir asociēta ar maka adresi un tiek sūtīta kā daļa no transakcijas, kad tiek sūtīta digitālā nauda [1]. Publiskā atslēga citādi tīklā netiek izplatīta. Ja publiskā atslēga vēl nav lietota transakciju veikšanai, tad parasti tā publiski pieejama nebūs. Publiskā atslēga tiek lietota, lai pierādītu, ka sūtīšana no konkrētās adreses ir likumīga.

1.3.2. Jaucējkods

Blockchain drošība ir balstīta uz jaucējkodiem, ar kuru palīdzību var pārbaudīt datu korektumu un autentiskumu. Jaucējkods ir jaucējfunkcijas darbības rezultāts. Konkrētai datu kopai jaucējfunkcija izdod tieši vienu jaucējkodu, kuram ir atbilstoši divas īpašības:

• Ar jaucējkodu nevar atrast sākotnējo datu kopu;

• Atrast otru tādu datu kopu, kura pēc jaucējfunkcijas darbības izdod tādu pašu jaucējkodu, gandrīz nav iespējams.

Zinot tikai jaucējkodu cilvēks nevar izdarīt nekādu kaitējumu, bet zinot sākotnējos datus var pārbaudīt, vai tie atbilst jaucējkodam. Ar to vienlaicīgi tiek iegūta atklātība un aizsardzība. Vēl viena jaucējkoda īpašība ir tas, ka minimāli mainot sākotnējas datus rezultāta jaucējkods lielā mērā izmainās. To var apskatīt piemērā. Ja sākotnējie dati ir "test string1", tad SHA-256 jaucējfunkcijas rezultāts būs "f7c62d30cb1ccef700b5c971cbf47591d47cde2cf55eb019d994883b5891a9a2". Un nedaudz samainot sākotnējus datus – "test string2" rezultāta iegūsim "33b1c15739b2053a5d66cff01a0059891b690d05959c78c52f6a6055089bef29". Kā var redzēt viena simbola maiņa ir jaucējkoda pilnīgas izmaiņas cēlonis. Viltot bloku virkni vai mainīt datu ierakstus blokā nav iespējams, tāpēc ka mainīsies jaucējkodi, ko redzēs citi tīkla lietotāji. Ja ļaundaris gribēs pievienot savu viltus bloku ar viņam vajadzīgajiem datu ierakstiem caur diviem jau eksistējošiem korektiem blokiem virknē, tad jaunā (viltotā) bloka galvenē ļaundarim jāievieto bloka jaucējkodu un jāpārsūta informāciju par jauno bloku visiem pārējiem tīkla lietotājiem. Lietotāji uzreiz varēs redzēt, ka bloks ir viltots, jo galvenē bloka jaucējkodā nav ņemts vērā, viltotā bloka jaucējkods. Vienīga iespēja maldināt lietotājus ir visu nākamo bloku pēc viltotā bloka atkārtota veidošana, ņemot vērā viltotā bloka izmaiņas. Tas, savukārt, nav iespējams, jo ātrums ar kuru respektīvie lietotāji veido jaunus blokus ir tāds pats, kā ātrums, ar kuru ļaunprātīgie lietotāji veido viltotos blokus, jo tieši tā darbojas bloku veidošanas algoritms [1, 5]. Teorētiski pastāv iespēja, ka var izveidot „uzbrukumu 51%" – kad ļaunprātīgo lietotāju datoru kopēja jauda ir lielāka par lietotāju datoru kopējo jaudu. Praktiski šādu uzbrukumu realizēt ir gandrīz neiespējami. Viltot bloka ierakstu arī nesanāks, jo mainot bloka ķermeni, kurā atrodas datu ieraksti, arī izmainās bloka jaucējkods, bet nākamos blokos jau ir saglabāts pareizs bloka jaucējkods. Līdz ar to jebkurš lietotājs var viegli pārbaudīt to, vai saņemtais bloks īstenībā ir viltots.

1.4. Blockchain tīkls

Visi Blockchain dati tiek glabāti tīkla lietotāju datoros. Visiem tīkla lietotājiem ir vienādas tiesības, līdz ar to katrs var vēlas, tajā skaitā veikt nesekmīgus mēģinājumus maldināt citus tīkla lietotājus. Blockchain tīkla lietotājiem nav vajadzīgs starpnieks, kam būtu jāuzticas (piemēram, banka vai apdrošināšanas kompānija), jo tīklā nav lietotāju ar papildus tiesībām vai iespējām. Praktiski neviens nevar lietotāju aizsargāt labāk par viņu pašu. Ieejot

Blockchain tīklā, lietotājs savienojas ar citiem datoriem tīklā, lai apmainīties ar datiem, kuri sastāv no blokiem ar datu ierakstiem tajos. Blockchain tīkls nav nekādā veida saistīts ar kādu ģeogrāfisku vietu. Tāda ziņa lietotājs no vienas valstis var vienlaicīgi izveidot savienojumu ar lietotājiem no citām valstīm. Tas arī pasargā lietotāju no jebkādam reģionālajām īpatnībām.

Saņemot datus un pārbaudot to korektumu un autentiskumu, katrs lietotājs saglāba tos sev un pa tīklu pārsuta tālāk. Blockchain tīklā vienlaicīgi var eksistēt divi datu veidi – korektie un viltus. Attiecīgi viltus datus tīklā izplata ļaunprātīgie lietotāji un korektos datus izplata respektablie lietotāji. Saņemot viltus datus respektablie lietotāji tālāk tos neizplata. Rezultāta viltus dati izplatās tikai starp ļaunprātīgajiem lietotājiem un respektablie lietotāji izplata tikai korektos datus [1].

1.5. Blockchain tīkla lietotāji

Blockchain tīkla lietotāji iedalās uz divās grupās:

- Parastais lietotājs – veido jaunus ierakstus;
- "Miner" lietotājs – veido blokus ar datiem.

Tā kā bloku veidošana prasa lielus resursus ne visi lietotāji grib un spēj to darīt. Parastie lietotāji veido un izplata tīklā datu ierakstus. Kā piemēru var minēt naudas pārsūtīšanu kriptovalūtas tīklā – lietotājs X sutā 10 digitālās naudas vienības lietotājam Y – visi dati ir atvērti, bet šifrēti [2]. Var redzēt summu, kura ir pārsūtīta, bet kurš un kam sūtījis nevar redzēt, ja sūtītājs un saņēmējs nav izvietojis šos datus publiskā pieejā. Jāpiemin, ka katram lietotājam var būt vairākas privātās atslēgas. Tāda veidā zinot datus par vienu pārskaitījumu var nezināt datus par citu, pat, ja abos ir iesaistīts viens un tas pats lietotājs. "Miner" lietotāji ievāc datu ierakstus, pārbauda tos, ieraksta datu blokos un izplata tīklā. Parastie lietotāji saņem blokus un saglabā pie sevis, lai varētu korekti veidot savus un korekti pārbaudīt citu lietotāju datu ierakstus. Kamēr datu ieraksts nav iekļauts, neviens ieraksts datu blokā nav uzticams. Jebkurš tīkla lietotājs var lietot datu ierakstus, kuri vēl nav iekļauti datu blokā, uzticoties tikai sev, jo ieraksts varētu būt nekorekts, vai viltus, vai arī to var atcelt. Tāpēc parastie lietotāji pārsūta jaunos datu ierakstus, lai beigu beigās tie tiktu pie "Miner" lietotāja, kurš iekļaus tos ierakstu blokā. Tikai tad, kad datu ieraksts ir saglabāts blokā, lietotājs var būt pārliecināts, ka ieraksts ir pārbaudīts, un atcelt to vairāk nevar.

1.6. Bloka galvenes jaucējkods

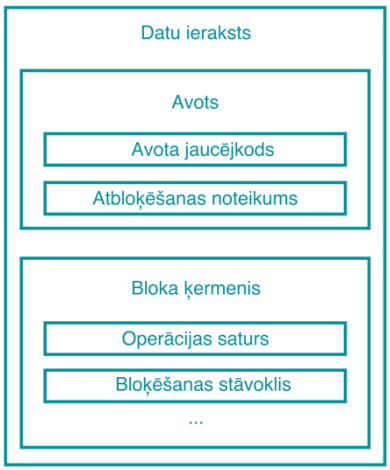
"Miner" ir Blockchain tīkla lietotājs, gluži kā parastie lietotāji, kuri pārbauda un izplata datus, bet viņš arī darbojas ar jauno bloku veidošanu. Saņemot jaunus ierakstu datus no citiem tīkla lietotājiem, "Miner" lietotājs saliek tos kopā, veido jauna bloka galveni un aprēķina bloka jaucējkodu. Bloka galvenē ir lauks *nonce*, kurš ir lietots izpildes pierādījuma algoritmā. Bitcoin pirmo bloku jaucējkods sākas ar desmit nullēm. Tas nosaka jauno bloku izveides sarežģītības līmeni. Ja pēc pirmā aprēķina atslēga nesakas ar desmit nullēm, tad vajag mainīt sākotnējas datus. Lai to panāktu ir lietots *nonce* skaitītājs. Jaucējkoda sākotnēji aprēķina vai šīs lauks ir vienāds ar nulli. Tāpēc "Miner" lietotājs palielina *nonce* vērtību uz vieninieku un aprēķina jaucējkodu atkārtoti [1]. "Miner" lietotājs atkārto šī darbību, kamēr vajadzīgais jaucējkods, kurš atbild noteikumiem (jāsākas ar desmit nullēm), vēl nav atrasts. Tikai tad, kad vajadzīgais jaucējkods ir atrasts, "Miner" lietotājs saglāba bloku un pārsuta viņu citiem lietotājiem. Tad visi datu ieraksti blokā ir apstiprināti un aizsargāti ar jaucējkodu, kuru viltot ir vērā ņemami grūti. Bloka jaucējkodā ir iekodēta iepriekšēja bloka jaucējkods, kuru arī viltot ir ļoti sarežģīti.

Jaucējkoda veidošanā galvenais ir tas, ka šīm procesam nav jāprogresē. Nav svarīgi, kad tiek uzsākta jaucējkoda meklēšana, cik datu ierakstu ir blokā, kādu laiku jaucējkods jau tiek rēķināts, kāds jaucējkoda skaits jau ir aprēķināts – varbūtība atrast vajadzīgo jaucējkodu katrā iterācijā vienmēr ir vienāda. Tas nozīme, ka nevar zināt iepriekšējo jaucējkodu un katram "Miner" lietotājam ir tikai viena iespēja atrast vajadzīgo jaucējkodu, vienkārši atkārtojot vienas un tās pašas darbības – rēķināt, rēķināt, rēķināt, utt. Par katru izveidotu bloku "Miner" lietotājs saņem peļņu. Kurš no "Miner" lietotājiem pirmais atrod jaucējkodu, tas ir izveidojis bloku. Bet pārējie "Miner" lietotāji, kuri arī meklēja jaucējkodu blokam nesaņem neko. Gadījumā, ja vienlaicīgi divi "Miner" lietotāji ir izveidojuši blokus, tad Blockchain virknei būs divi zarojumi, kamēr netiks atrasts nākamais bloks, kurā no diviem zarojumiem tas tika pievienots, un šis zarojums tad arī būs īstais. Otrais zarojums vairāk netiks ņemts vērā, jo Blockchain virknē īstais zars ir visgarākais zars. Tāda veida jaucējkoda meklēšanas procedūra apgrūtina bloku veidošanu, bet tā arī ļoti apgrūtina viltus bloku izveidi, padarot to gandrīz neiespējamu.

1.7. Bloka datu ieraksti

No iepriekšējas nodaļas un sadaļas var secināt, ka viltot datu ierakstus blokā vai ievietot viltotu bloku iekš Blockchain ir gandrīz neiespējami. Tā kā visiem tīkla lietotājiem ir vienādas

tiesības, var iedomāties, ka lietotājs X mēģinās izveidot ierakstu, kurā aprakstīs, ka lietotājs Y pārsūta digitālo naudu uz X lietotāja kontu. Bet tas nav iespējams, jo datu ieraksti ir aizsargāti ar apvienošanu datu ierakstu virknē. Katrs bloka ieraksts satur saiti uz iepriekšējo ierakstu (avotu), bloķēšanas stāvokli un atbloķēšanas nosacījumu [2].



1.4. att. **Blockchain datu bloka ieraksta struktūra**

Avots ir saite uz iepriekšēja datu ieraksta rezultātu. Rezultāts savukārt ir nākošā datu ieraksta avots. Operācijas saturā ir operācijas apraksts formalizētā veidā, piemēram, „pārsūtīt 10 digitālās naudas vienības". Katrs rezultāts ir aizsargāts ar bloķēšanas stāvokli. Pievienot jaunu ierakstu, pagarināt datu ierakstu virkni un mainīt informāciju iekš Blockchain var tikai tie lietotāji, kuri var apmierināt atbloķēšanas nosacījumu. Ja lietotājs X pārsūtījis digitālo naudu lietotājam Y ar bloķēšanas stāvokli datu ierakstā tā, ka paroles jaucējkodam ir jābūt noteiktam, tad iztērēt šo naudu lietotājs Y varēs tad un tikai tad, ja izveidos jaunu ierakstu, norādot atbloķēšanas noteikumā vajadzīgo paroli. Atbloķēšanas nosacījuma rezultāts tiks ievietots iepriekšējā ieraksta bloķēšanas stāvoklī un, ja tas ir korekts, tad skaitās, ka datu ieraksts ir korekts, un tas nozīmē, ka „Miner" lietotājs saņems ierakstu un pievienos to blokā. Ja stāvoklis neizpildās, piemēram, ja ir ievadīta nepareiza parole, tad visi respektablie lietotāji nepārsutis šo ierakstu tālāk un „Miner" lietotāji nepievienos ierakstu blokā. Tas nozīme, ka Blockchain tīkls nepieņems operāciju.

Stāvokļu un nosacījumu aprakstīšanai Blockchain virknē tiek lietota programmēšanas valoda, kura ļauj veidot visai sarežģīto loģiku un aprakstīt lietotāju mijiedarbības noteikumus. Datu ieraksts var pārveidot dažus ierakstus-avotus uz ierakstiem-rezultātiem. Blockchain ļauj formalizēt attiecības ne tikai starp cilvēkiem, bet arī starp robotiem un programmām. Šādas attiecības sauc par „gudrajiem līgumiem".

1.8. Blockchain pielietošanas sfēras

Pārsvarā dotajā brīdī Blockchain tehnoloģija tiek izmantota kriptovalūtās, tādās, kā Bitcoin un dažas citos. Tieši uz kriptovalūtas piemēriem tika ieraudzīta Blockchain perspektīva, ierobežojumi un problēmas ar kuriem ir jāsastopas reālajā dzīvē. Blockchain arī tiek izmantots, lai fiksētu tiesības uz māksla darbiem un citām rētām vērtībām, piemēram, dimantiem. Daudzi projekti pasaulē ir sākuši lietot Blockchain un eksperimentēt ar to. Šādus projektus var iedalīt divās grupas – projekti, kuri veido Blockchain infrastruktūru un projekti, kuri lieto Blockchain tehnoloģiju konkrēta mērķa realizēšanai [1]. Pirmā grupa iekļauj Ethereum platformu, kuras mērķis ir radīt ne tikai Blockchain, bet arī daudzfunkcionālu platformu gudrajiem līgumiem.

1.9. Secinājumi

Blockchain ir datubāze, kurā var glabāt ierakstus par objektiem un notikumiem no reālās dzīves. Ierakstus iekš Blockchain ir gandrīz neiespējami viltot, samainīt vai nodzēst. Visus ierakstus, kuri piefiksēti Blockchain virknē var viegli pēc autentiskuma. Blockchain tehnoloģija ir diezgan sarežģīts tehniskais risinājums, kas joprojām turpina attīstīties.

2. ETHEREUM PLATFORMA

Ethereum platformas pamatā ir Blockchain tehnoloģija, kas ļauj veidot decentralizētas sistēmas bez starpniekiem un samazināt izmaksas valūtas, aktīvu un informācijas apmaiņā starp cilvēkiem un uzņēmumiem. Ethereum platformas mērķis ir, apvienojot un izstrādājot koncepciju skriptu veidošanai, alternatīvas kriptovalūtas un gudros līgumus, ļaut visiem, kas vēlas veidot patvaļīgas decentralizētas lietojumprogrammas, kas savukārt vienlaicīgi sniedz mērogojamības, standartizēšanas, izstrādes viegluma un savietojamības īpašības. Iebūvētā Ethereum bloku virknē Tjūringa-pilnīgā programmēšanas valoda dod iespēju ikvienam rakstīt gudros līgumus un decentralizētas lietojumprogrammas, kurās var izveidot savus piederības noteikumus, transakciju formātus un patvaļīgas funkcijas stāvokļa maiņai.

Jāpiemin, ka Ethereum atrodas izstrādes procesā un dotajā brīdī kopā ir ieplānoti seši atjauninājumi. No tiem divi atjauninājumi jau ir iznākuši:

- "Frontier" ir Ethereum tīkls vistīrākajā formā. Satur bloku veidošanas saskarni un gudro līgumu augšupielādes un izpildes iespējas;
- "Homestead" ir "Frontier" versijas uzlabojums, kura mērķis ir parādīt, ka Ethereum platformas izmantošanas risks ir būtiski samazinājies salīdzinājumā ar sākotnējo versiju.

Divi nākoši ieplānotie atjauninājumi:

- "Metropolis" ir atjauninājums, kurš saturēs tīmekļa pārlūkprogrammu Ethereum platformai un decentralizētas lietojumprogrammas veikalu. Atjauninājuma mērķis ir parādīt visas Ethereum tīkla iespējas lietotājiem bez tehniskām priekšzināšanām;
- "Serenity" ir atjauninājums, kas iekļaus tīkla pārslēgšanu no bāzēta uz izpildes pierādījuma Blockchain koncepcijas uz daļas pierādījuma Blockchain koncepciju.

Vēl divi atjauninājumi pagaidām paliek bez nosaukumiem (Ethereum 2.0 un 3.0) [6, 7]. Uz doto brīdi Ethereum platforma jau nodrošina gudro līgumu izveidi un attiecīgi decentralizētas lietojumprogrammas uzbūvi. Šajā daļā ir aprakstīta Ethereum platforma, tas darbība un lietojumprogrammas izveide iekš Ethereum.

2.1. Vēsturiskais ieskats

Bitcoin parādījās 2009. gadā un tas bija brīvi izplātāms unikālas digitālās valūtas kods [1]. Prefikss "Bit" norāda uz BitTorrent failu apmaiņas sistēmu, piemērojot tas ideju naudas pasaulei. Tāpēc Bitcoin tīklam nav centrālais serveris, kurā tiktu uzglabāti un apstrādāti dati.

Tas notiek lietotāju ierīcēs, kas kopā veido lielu datortīklu. Valūtu izsniedz nevis centralizēta iestāde, bet paši lietotāji, kā specifiskas aprēķināšanas loterijas rezultātā, kuras noteikumi ir stingri noteikti vienreiz un nekad nemainīsies. Galvenā Bitcoin ideja ir pilna decentralizēšana. Tā kā kriptovalūtai nav vienotas valūtas izsniegšanas vietas, tad tā ir aizsargāta pret manipulācijām. Kriptovalūta neuzglabājas serverī, kuru varētu uzlauzt vai arestēt. Jebkura valūtas summa tiek pārsūtīta uzreiz, neskatoties uz valsts robežām un valsts noteikumiem, un nav pakļauta kādai obligātajai komisijai.

Ethereum platformas idejas autors ir Vitalik Buterin, kurš ir dzimis 1994. gadā. Šobrīd Vitalik ir pazīstams kriptovalūtas kopienā kā programmētājs un rakstu autors. 2011. gadā viņš strādāja izdevumā Bitcoin Weekly, vēlāk kļūstot par vienu no Bitcoin Magazine dibinātājiem un līdz 2013. gada bija viens no galvenajiem izdevuma autoriem. 2013. gadā novembrī Vitalik paziņoja par darbu jaunā projektā – Ethereum. Viņa ideja bija paņemt Bitcoin principus un padarīt tos elastīgus un plaši piemērojamus, lai tos varētu izmantot jebkurā projektā, ieskaitot tādus kā jaunas kriptovalūtas, failu uzglabāšanas sistēmas, domēna vārdu reģistrs un citos līdzīgos projektos. 2014. gadā projektā piedalījās arī Neal Koblitz – eliptiskas šifrēšanas mehānisma algoritma, kurš tiek izmantots Bitcoin, autors [7]. Vēl viena svarīga persona Ethereum projektā ir Jeffrey Wilck, kurš 2013. gadā sācis īstenot Ethereum ideju Go programmēšanas valodā un bijis Go komandas līderis un galvenais attīstītājs. Ethereum klients Go programmēšanas valodā veiksmīgi palaists 30. jūlijā 2015. gadā, atzīmējot pirmā bloka izveidi un Ethereum platformas palaišanu [8].

2.2. Ethereum platformas darbība

Ethereum platforma balstās uz Blockchain tehnoloģijas. Centralizētas lietojumprogrammas ir uzticamas dēļ sarežģītās drošības sistēmu lietošanas un kvalificētas ekspluatēšanas komandas, kuri ļoti labi pilda savus pienākumus. Blockchain tehnoloģija lieto atšķirīgu modeli, kurā uzticamību var iegūt, pamatojoties uz atvērtā tīkla arhitektūru. Bitcoin ir pirmā digitālā valūta, kura bāzēta uz Blockchain tehnoloģiju. Ethereum dod iespēju pielietot šo tehnoloģiju gandrīz jeb kam, ko var izteikt matemātiski un ar ko var apmainīties tāpat kā ar Bitcoin valūtu, starpniecībai neizmantojot personas vai organizācijas, kuriem būtu jānodrošinātu informācijas drošību un jāsniedz visa veida pakalpojumus. Iekš Ethereum platformas lietotāja dati ir aizsargāti ar kriptogrāfijas algoritmu palīdzību. Ethereum izstrādātāji raksta biznesa loģiku, veidojot gudros līgumus. Gudrie līgumi ir programmatūras vienības, kas īsteno darbību secību. Gudrie līgumi var glabāt datus, saņemt un sūtīt transakcijas, kā arī sadarboties ar citiem gudrājiem līgumiem, nevienam nepakļaujoties. Tos

apkalpo tīkls, un tie ir uzrakstīti programmēšanas valodā, kas ir līdzīga daudziem citiem populāriem programmēšanas valodām. Tā kā Ethereum platformas gudrie līgumi ir arī izplatīti tīklā, tad nav vajadzīga infrastruktūra lietojumprogrammas izplatīšanai un instalēšanai. Lietotāji arī nav ieinteresēti veikt DDoS uzbrukumus Ethereum tīklā, jo par katru transakciju tīklā ir jāmaksā komisijas maksa un, lai veiktu DDoS uzbrukumus, nepieciešams liels naudas apjoms, kas galu galā nokļūs pie "Miner" lietotāja par skaitļošanas operācijas veikšanu.

2.2.1. GHOST protokols

Blockchain tehnoloģijas realizācijām, kurām ir ātra (piemēram, desmit sekundes) bloku izveide, ir drošības trūkums, ņemot vērā lielu nevajadzīgo bloku skaitu. Blokam ir vajadzīgs kāds laiks, lai izplatītos tīklā. Blokiem, kuri netiks iekļauti Blockchain virknē, izveides varbūtības attiecību (procentos) pret bloka, kurš tiks iekļauts virknē, izveidi var aprakstīt ar formulu:

$$\left(\frac{\left(\frac{p}{b}\right)}{\left(1+\left(\frac{p}{b}\right)\right)}\right) * 100\%$$

Kur p ir vidējais bloka izplatīšanas laiks tīklā, b ir vidējais bloka izveides laiks un 1 ir bloka, kurš tiks iekļauts bloku virknē, izveides iespēja(1, jo tāds bloks tiks vienmēr atrasts). Ja bloka izveide vidēji aizņem 600 sekundes, kā tas ir iekš Bitcoin, un bloka izplatīšana tīklā vidēji aizņem 10 sekundes, tad, lietojot iegūto formulu var aprēķināt bloku, kuri netiks iekļauti bloku virknē, izveides varbūtību:

$$\left(\frac{\left(\frac{10}{600}\right)}{\left(1+\left(\frac{10}{600}\right)\right)}\right) * 100\% \approx 1.64\%$$

Bet ja bloka izveide vidēji aizņem 10 sekundes un bloka izplatīšana tīklā arī aizņem 10 sekundes, tad pēc formulas:

$$\left(\frac{\left(\frac{10}{10}\right)}{\left(1+\left(\frac{10}{10}\right)\right)}\right) * 100\% \approx 50\%$$

Var secināt, ka, jo mazāks ir bloka izveides laiks, jo vairāk būs bloku, kuri netiks iekļauti Blockchain virknē. Lielais neiekļauto bloku skaits ietekmē uz tīkla centralizēšanu, jo situācijā, kad tīklā ir 7500 „Miner" lietotāju ar 0.01% skaitļošanas jaudu katrs no kopēja tīkla skaitļošanas jaudas un viens „Miner" lietotājs, vai grupa (*mining pool* – angļu val.) ar 25%

skaitļošanas jaudas no kopējā tīkla skaitļošanas jaudas, tad 75% no laika vienības bloku veido viens no „Miner" lietotājiem, kura skaitļošanas jauda ir 0.01% no tīkla kopējas, un bloks izplatās tīklā 10 sekundes (ja tiek ņemts iepriekšējais pieņēmums). Savukārt 25% no laika vienības bloku veido „Miner" lietotājs (grupa), kuram pieder 25% skaitļošanas jaudas no tīkla kopējās un pēdējais uzzina par jaunu bloku uzreiz, bet visi pārējie tīkla lietotāji vēl par to nezina un turpina veidot aprēķinus. Gadījumā, kad bloka izveide vidēji aizņem 10 sekundes un bloka izplatīšana vidēji arī aizņem 10 sekundes, tad iespēja atrast bloku, kurš netiks iekļauts bloku virknē, ir 50%, kā tiek aprakstīts augstāk. Un "Miner" lietotājs (grupa) ar skaitļošanas jaudu 25% veidos jaunus blokus 25% no laika vienības ar iespēju izveidot bloku, kurš netiks iekļauts bloku virknē, kas ir vienāda ar 0%. Var aprēķināt „Miner" lietotāju (grupas) efektivitātes koeficientus. „Miner" lietotājam ar 25% skaitļošanas jaudas efektivitātes koeficients būs:

$$\left(1 - \left(\frac{\left(\frac{10}{10}\right)}{\left(1 + \left(\frac{10}{10}\right)\right)}\right)\right) * 0.75 + 1 * 0.25 = 0.375 + 0.25 = 0.625$$

Apskatāmā situācijā katram „Miner" lietotājiem (grupai) efektivitātes koeficients būs 0.5. Tad viegli aprēķināt, ka starpība ir:

$$100 - \left(\frac{0.5}{\left(\frac{0.625}{100}\right)}\right) = 100 - 80 = 20\%$$

Starpība 20% apmērā ekonomiski ir ļoti būtiska. Rezultātā „Miner" lietotāji gribēs apvienoties grupās (*mining pool* – angļu val.) un tas noved pie tā, ka Blockhain tīkli ar mazu intervālu starp bloku izveidi tiks kontrolēti ar 1-2 lielām grupām, kas īstenībā ir 51% uzbrukuma realizēšana [9, 12].

GHOST protokols ir virknes izvēles nosacījums, kurš lieto blokus, kuri nav iekļauti virknē, lai sniegtu drošāku un pielāgojamu sistēmu. Protokols tiek piedāvāts 2013. gadā decembrī, un tā galvenā ideja ir tāda, ka jāpievieno kopējā virknes svarā bloki, kuri netiks iekļauti virknē (ir nevajadzīgi) [9].

GHOST protokols palīdz izvairīties no 51% uzbrukuma problēmas, jo, neskatoties uz to, kāds ir galvenā zara efektivitātes koeficients – 75% vai 5% – ļaunprātīgam lietotājam vajadzēs pārvarētu visa tīkla svaru [11]. Oriģinālā GHOST protokolā netiek piešķirta atlīdzība par blokiem, kuri netiks iekļauti bloku virknē, un tāpēc tas nepilnīgi atrisina tīkla centralizēšanas problēmu. Ethereum platformā ir pielietota modificēta GHOST protokola realizācija un tā ir definēta sekojoši:

- Blokam jānorāda uz vecāku (ja nav pirmais virknes bloks) un uz nulli vai vairāk tēvočiem;
- Tēvocim, kurš ir iekļauts blokā X, jābūt sekojošām īpašībām:
    - Tēvocim jābūt tiešam bērnam kādam n paaudzes X sencim, kur $2 \leq n \leq 7$;
    - Tēvocis nevar būt B sencis;
    - Tēvocim jābūt korekta bloka galvene, bet nav jābūt iepriekš pārbaudītam vai pat derīgam blokam;
    - Tēvocim jāatšķiras no visiem tēvočiem, kas iekļauti iepriekšējos blokos vai X blokā;
- Par katru tēvoci Y blokā X „Miner" lietotājs, kurš, izveidojis bloku X, saņem papildus 3.125% atlīdzību un „Miner" lietotājs, kurš izveidojis Y saņem 93.75% no parastās atlīdzības (100%) par bloka izveidi [9, 10].

Praksē tiek lietota modificēta realizācija, jo pilna GHOST protokola versija ievērojami sarežģītu. Aprēķinu, vai kāda dota bloka tēvoči ir korekti. Un arī tāpēc, ka pilna GHOST versija ar eksistējošo Ethereum platformas kompensēšanu par blokiem, kuri netiks iekļauti virknē, noved pie stimula trūkuma veidot jaunus blokus galvenās virknes zarā, nevis uzbrucēja zarā.

2.2.2. Bloku izveides centralizēšanas problēma

Algoritms, kurš ir pielietots Bitcoin tīklā bloku veidošanai nav aizsargāts pret divu veidu centralizāciju. Pirmkārt, eksistē ierīces ko sauc par lietojumprogrammatūras specifisku integrēto shēmu (application-specific integrated circuits jeb ASIC – angļu val.), kuri ir izstrādāti speciali, lai veidotu Bitcoin blokus. Tādām ierīcēm ir daudz reiz vairāk skaitļošanas jaudas nekā pārējiem ierīcēm Bitcoin tīklā [1]. Līdz ar to bloku veidošana iekš Bitcoin tīkla nav pilnīgi decentralizēta, jo tīkla lietotājiem nav vienlīdzīgas iespējas un eksistē "Miner" lietotāji ar ASIC ierīcēm, kuriem ir daudz vairāk skaitļošanas jaudas nekā pārējiem lietotājiem. Jo lielāka ir "Miner" lietotāja skaitļošanas jauda attiecība pret tīkla kopējas skaitļošanas jaudu, jo lielāka ir lietotāja ietekme uz tīklu. Otrkārt, "Miner" lietotāji Bitcoin tīklā galvenokārt neveido paši bloku pārbaudi, bet gan uzticas lietotāju grupām, kuras izsniedz bloku galvenes. 2014. gadā aprīlī trīs lielākas "Miner" lietotāju grupas kontrolēja apmēram 50% no skaitļošanas jaudas Bitcoin tīklā. "Miner" lietotāji var pārslēgties uz citu grupu 51% uzbrukuma gadījumā, bet tas nepadara situāciju vairāk uzticamu [13].

Ethereum platformas izstrādātāji vēlas atrisināt aprakstītas augstāk aprakstītas problēmas ar algoritmu, kurā "Miner" lietotāji saņem nejaušu daļu no stāvokļa datiem, aprēķina dažas nejaušas transakcijas no pēdējiem N blokiem un atgriež rezultāta jaucējkodu. Tā kā Ethereum līgumi var uzglabāt sevī jebkāda veida aprēķinus, tad ASIC ierīcei priekš Ethereum tīkla ir jābūt ierīcei, kas spēj darīt jebkāda veida aprēķinus, nevis noteikta veida aprēķinus, kā tas ir Bitcoin gadījumā, citiem vārdiem sakot, ierīcei jābūt jaudīgu procesoru. Protams, paliek iespēja veidot līgumus, kas varētu padarīt nederīgu aprēķināšanai konkrētu ASIC ierīci un ASIC ierīces ražotājiem būtu izdevīgi lietot šo iespēju, lai apturētu konkurentus. Tādā veidā šis risinājums ir vairāk pielāgojams ekonomiski-sociāls risinājums, nevis tehnisks. Bloku veidošanai ir vajadzīga pieeja visai bloku virknei, kas rada nepieciešamību uzglābāt veselo Blockchain virkni un spēt pārbaudīt katru transakciju. Tāda pieeja padara "Miner" lietotāju grupas pastāvēšanu nevajadzīgu. Aprakstītais algoritms vēl ir izstrādes stadijā, un tāpēc var rasties grūtības līgumu lietošanā kā bloku veidošanas algoritmus, bet šī pieeja varētu atrisināt bloku izveides centralizēšanas problēmas.

2.2.3. Ethereum konti

Iekš Ethereum Blockchain stāvoklis sastāv no kontiem. Katram kontam atbilst tās divdesmit baitu adrese. Stāvokļa maiņas funkcija ir valūtas vai informācijas pārsūtīšana starp kontiem. Ethereum konts satur četrus laukus:

- Nonce – skaitītājs, kurš vajadzīgs lai nodrošinātu katras transakcijas vienreizējo izpildi;
- Bilance – tekošā konta bilance Eth kriptovalūtā;
- Līguma kods, ja eksistē līgums, kurš piesaistīts kontam;
- Konta krātuve – tukša pēc noklusējuma.

Eth kriptovalūta arī tiek lietota lai maksāt komisiju par transakciju veikšanu. Platformā ir pieejami divi kontu veidi:

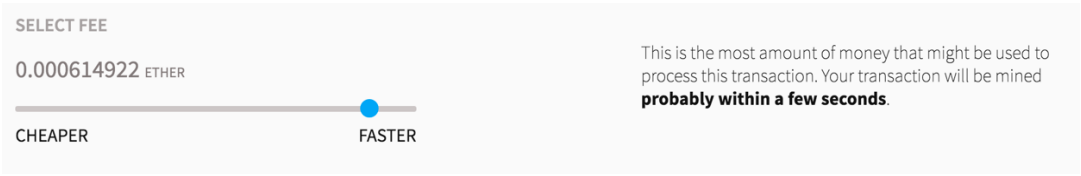
- Konts, kuru kontrolē īpašnieks. Tas pats kas ir Bitcoin sistēmā – konts, kura vienīgā iespēja ir naudas pārsūtīšana (izejošo transakciju veidošana). Tādā veidā taisot un pārrakstot transakcijas no šī konta var sūtīt ziņojumus.
- Konts, kurš ir kontrolēts ar kontam piesaistīta līguma kodu. Katrā reizē, kad konts saņem ienākošo transakciju, tiek palaists līguma kods, kurš var dod iespēju transakcijai pievienot un lasīt no konta krātuves informāciju, sūtīt citas transakcijas un veidot citus līgumus [14].

Jāpiemin, ka konta izveides laikā lietota parole ir teksta virkne, ar kuru tiek šifrēts konts. Iekš Ethereum neeksistē iespējas atjaunot paroli un nevar iegūt pieeju konta līdzekļiem, tas zaudēšanas gadījumā.

2.2.4. Transakcijas

Ethereum transakcijas ir līdzīgas Bitcoin transakcijām, bet ar trim būtiskām atšķirībām. Pirmkārt, transakcijas iekš Ethereum var veidot gan ārējie avoti (cilvēki), gan līgumi, bet iekš Bitcoin transakcijas var veidot tikai cilvēks. Otrkārt, būtiska priekšrocība ir tas, ka Ethereum transakcija var saturēt datus. Treškārt, Ethereum transakcijas saņēmējs, ja tas ir konts, kurš ir kontrolēts ar līgumu, var atbildēt transakcijas autoram uz transakciju. Tāpēc Ethereum transakcijas var salīdzināt ar parastām funkcijām, kuras, saņemot ieejas datus, pēc darbības atgriež rezultātu.

Ar transakciju Ethereum platformā apzīmē ziņojuma datus, kuri tiks sūtīti no ar ārējo avotu kontrolējama konta. Transakcijas satur datus par saņēmēju, sūtītāja parakstu, sūtama Eth kriptovalūtas vērtību un sūtamu informāciju, kā arī divu mainīgo vērtības – STARTGAS un GASPRICE. Transakcijas sūtītājs uzstāda ierobežojumu uz skaitļošanas operācijas skaitu (STARTGAS mainīgais), kas iekļauj sākotnējo skaitļošanas operāciju un visas no tās radītas skaitļošanas operācijas, lai izvairītos no bezgalīgiem cikliem. Savukārt GASPRICE mainīgais ir komisija, kuru “Miner” lietotājs saņem par katru skaitļošanas operāciju, to nosaka sūtītājs kā GAS/Eth vērtību. Ja GAS/Eth vērtība, kuru uzstādīja sūtītājs, neapmierina nevienu "Miner" lietotāju, tad transakcija netiek izpildīta. Ethereum makā GAS/Eth vērtību var izvelēties no noteikta diapazonā (2.1. att.).



2.1. att. **Komisijas maksas uzstādīšanas Ethereum mākā**

Ja transakcijas izpildes brīdī GAS vērtība beidzas, tad visas izmaiņas tiek atsauktas un notiek atgriešana uz sistēmas sākotnējā stāvoklī (uz stāvokli, kas bija pirms transakcijas sākuma), bet komisija, kura tiek samaksāta “Miner” lietotājiem netiek atgriezta atpakaļ sūtītājam. Gadījumā, kad transakcija tiek pārtraukta un GAS vērtība nav vienāda ar nulli, tad GAS vērtības atlikums tiek atgriezts sūtītājam. Tādējādi līgumu autori nodrošina līguma koda izpildi ar ieguldīto GAS vērtību (līdzīgi mobilā telefona bilances papildināšanai).

Kontiem, kuri ir kontrolēti ar līgumiem, ir tāda pat funkcionalitāte, kura ir pieejama kontiem, kurus kontrolē īpašnieks (cilvēks), ieskaitot iespēju sūtīt transakciju vai ziņojumu, un veidot citus līgumus. Iekš Ethereum eksistē ziņojumi, kuri ir līdzīgi transakcijām, kurus var veidot tikai gudrie līgumi, bet nevar ārējie avoti (cilvēki). Ziņojumi satur datus par sūtītāju, saņēmēju, sūtamo Eth kriptovalūtas vērtību, sūtamo informāciju un STARTGAS vērtību. Transakcijas un ziņojumi var veidot līgumus. Jauna līguma, kuru veido transakcija vai ziņojums, adrese tiek rēķināta no *nonce* vērtības un informācijas, kas glabājās transakcijā vai ziņojumā. Tīkla lietotājiem nav jāuztraucas par to, kāds ir saņēmēja konta tips, nav svarīgi vai tas ir līgums, vai ārējais avots [9].

2.2.5. Tīkla stāvoklis

Kā tiek minēts 2.2.4. nodaļā, Ethereum tīkla stāvoklis sastāv no kontiem. Stāvokļa maiņas funkcija ir valūtas vai informācijas pārsūtīšana starp kontiem. Ethereum platformas stāvokļa maiņas funkcija ir aprakstāma kā $APPLY(S, TX) \rightarrow S'$, kur S ir pašreizējais tīkla stāvoklis, TX ir transakcija vai ziņojums kas pielietojama pašreizējajam tīkla stāvoklim, un S' ir tīkla stāvoklis pēc transakcijas izpildes. Apskatīsim daļu no koda, vienkārša Ethereum sistēma domēna vārdu reģistrēšanai (2.2. att.).

```
if (domainNames[name] == address(0x0)) {
    domainNames[name] = value;
}
```

2.2. att. **Domēna vārda reģistrēšanas funkcijas daļa**

Jāņem vērā, ka līguma kods tiek rakstīts zema līmeņa programmēšanas valodā(EVM kods), bet piemēra kods ir uzrakstīts Solidity augsta līmeņa programmēšanas valodā, kurš tiek kompilēts uz EVM kodu. Sīkāk par Ethereum programmēšanas valodu un EVM tiek aprakstīts 2.3.1. un 2.3.2. nodaļās.

Pieņemsim, ka konta krātuve ir tukša un transakcijā tiek sūtītas 15 Eth mērvienības un līgums ir nodrošināts ar 3000 GAS vienībām ($STARTGAS$ ir 3000), $GASPRICE$ ir 0.00106, un vēl papildus tiek sūtīta sekojošā nākoša informācija: [’10.10.10.10’, ‘example.com’]. Tad uz šī piemēra Ethereum stāvokļa maiņas funkcijas darbība būs sekojoša:

1. Tiek pārbaudīts transakcijas formāts, digitāla paraksta korektums un *nonce* lauka vērtība tiek salīdzināta ar sūtītāja konta *nonce* vērtību – tām ir jāsakrīt. Ja kaut kas nav kārtībā, tad tiek atgriezta kļūda un funkcijas darbība apstājas.
2. Pēc digitāla paraksta tiek noskaidrota sūtītāja adrese. Tiek aprēķināta komisijas vērtība pēc formulas $STARTGAS * GASPRICE$, kas piemērā būs $3000 * 0.00106 = 3.18$ Eth,

un tiek pārbaudīts, vai sūtītājam kontā ir vismaz 3.18 Eth mērvienības. Ja sūtītāja konta bilance ir mazāka par komisijas summu, tad tiek atgriezta kļūda. Pretējā gadījumā no sūtītāja konta bilances tiek atņemta komisijas summa (3.18 Eth) un konta *nonce* vērtība tiek palielināta uz vienīnieku.

3. GAS mainīgajam tiek piešķirta $STARTGAS$ vērtība (3000) un par katru baitu, kuru apstrādājis "Miner" lietotājs tiek vākts noteikts daudzums no GAS vērtības. Pieņemsim, ka transakcija ir 182 baitu gara un komisija par vienu baitu ir 5, tad 182*5=910. Tad no GAS vērtības tiek atņemts 910 un paliek 2090 vienības.
4. No sūtītāja konta tiek atņemti 15 Eth (summa noradīta transakcijā) un tie tiek pievienoti saņēmēja kontam. Ja saņēmēja konts neeksistē, tad tas tiek izveidots. Ja saņēmēja konts ir kontrolēts ar līgumu, tad tiek palaists līguma kods. Tālāk ir divas iespējas – vai nu koda izpildes gaitā beigsies GAS vērtība, vai kods veiksmīgi izpildīsies.
5. Pieņemsim, ka saņēmēja konts ir kontrolēts ar līgumu. Tiek palaista koda izpilde. Piemēra paņemtā koda daļa tikai pārbauda vai saņēmēja konta krātuvē pēc adreses '10.10.10.10' ir informācija vai nav. Ja informācijas pēc norādītās adreses nav, tad tiek pievienota 'example.com' vērtība, kura būs pieejama pēc adreses '10.10.10.10'. Pieņemsim, ka koda izpilde maksā 187 GAS vienības, tādā gadījumā paliek $2090 - 187 = 1903$ vienības.
6. Ja GAS vērtība ir lielāka par nulli, tad koda veiksmīga izpildes gadījumā, tad tiek aprēķināta summa pēc kursa Eth/GAS (noradīts $GASPRICE$ mainīgajā) un atgriezta sūtītājam, un iztērētā summa tiek pārskaitīta "Miner" lietotājam. Sūtītājam tiek atgrieztas $1903 * 0.00106 = 2.01718$ Eth vienības, bet "Miner" lietotājs saņem $3.18 - 2.01718 = 1.16282$ Eth [9, 20].

Ja sūtītāja kontā nav noradītas transakcijā pārsūtamas summas (Eth mērvienībās) vai koda izpildes laikā beigusies GAS vērtība, tad visas izmaiņas tiek atsauktas, izņemot komisijas izmaksu "Miner" lietotājam. Gadījumā, kad saņēmēja konts nav kontrolēts ar līgumu, tad piektais solis (apskatītā piemērā) tiek ignorēts. Tādā gadījumā komisijas vērtība ir atkarīga tikai no transakcijas garuma (trešais solis piemērā) un nekāda informācija krātuvē netiks pievienota. Jāpiemin, ka konti, kuri tiek kontrolēti ar līgumiem var paši noradīt $STARTGAS$ vērtību ziņojumiem, kurus rada kods. Kļūdas gadījumā atgriešana notiek uz stāvoklī kodā, kurā tiek radīts šis ziņojums (kurā notikusi kļūda), nevis līgumā koda sākumā. Tas dod iespēju līgumiem noteikt aprēķināšanas soļu skaitu katrā izveidojamajā ziņojumā.

2.2.6. Mērogojamība

Mērogojamības jautājums ir svarīgs, jo parastā bloku virknē, piemēram Bitcoin virknē, katru transakciju vajag apstiprināt katram tīkla lietotājam. Tas ir viens no galvenajiem Blockchain principiem, kurš padara to par tik uzticamu mehānismu transakciju veikšanai. Tomēr tāda pieeja izraisa problēmu – viss tīkls ir tik pat spēcīgāks, cik katrs atsevišķais dators tīklā.

Ethereum platformas izstrādātāji plāno atrisināt aprakstīto problēmu, kas ļaus platformai palielināt transakciju skaitu bezgalīgi. Viena no pieejam problēmas risināšanai ir ieviest tā saucamus "zibenīgus tīklus" (*lightning network* – angļu val.), kuru ideja ir nevis iekļaut katru transakciju bloku virknē, bet darīt to tikai gadījumā, kad viens no transakcijas dalībniekiem atsakās sadarboties [17]. Tas dod iespēju uzglabāt mazāk informācijas bloku virknē. Otra pieeja problēmas risināšanai ir *sharding* tehnoloģijas pielietojums. *Sharding* ir datu bāzes mērogojamības tehnoloģija, kurā viena liela datu bāze tiek sadalīta mazākās. Pieejas ieviešanas process tiks sadalīts soļos. Pirmā datu bāzes sadalīšana ieplānota ar Ethereum 2.0 īstenošanu un pēc tam tiks izstrādāta Ethereum 3.0, kurai būs īpašība sadalīties bezgalīgi. Paralēli tiks ieviesta daļas pierādījuma koncepcija iekš Ethereum Blockchain virknes. Starpposmā, lai padarītu parēju vienmērīgāku tiks pielietots hibrīda modelis, pateicoties Casper līgumiem. Casper ir protokols ekonomiskas vienprātības jeb saskaņas panākšanai, kurš balstās uz drošības garantijas depozītiem. Pamatojoties uz Ethereum platformas izgudrotāja vārdiem, rezultātā katrs tīkla lietotājs uzglabās ne vairāk ka 0.1-1% no visām transakcijām [16]. Tas nozīme, ka nebūs vajadzības uzglabāt pilno Blockchain virkni. Uz doto brīdī Bitcoin Blockchain aizņem 83.28 gigabaitus un izmērs pieaug apmēram par vienu megabaitu stundā [18]. Ja Bitcoin tīkls veidotu dažus tūkstošus darījumu sekundē, kā, piemēram, Visa, tad Bitcoin tīkla Blockchain pieaugtu līdz dažiem gigabaitiem stundā. Ethereum platformai varētu būt līdzīgas problēmas, jo tās Blockchain vēl satur lietojumprogrammas, kas strādā uz Ethereum platformas (nevis tikai kriptovalūtu kā Bitcoin). Līdz ar to aprakstītās augstāk pieejas ievērojami uzlabos Ethereum tīkla mērogojamību.

2.2.7. Konfidencialitāte

Ethereum platforma jau piedāvā jau vairāk nekā 100 decentralizētas lietojumprogrammas un pozicionē sevi kā milzīgs pasaules datoru tīkls, bet pašlaik Ethereum trūkst konfidencialitātes, kas ir viena no datora priekšrocībām [19]. Gredzena paraksti ar saistīšanas iespēju (*linkable ring signatures* - angļu val.) ir konfidencialitātes problēmas

risinājums, kuru agrāk piedāvāja Ethereum platformas izgudrotājs. Tādā pieejā labi strādās balsošanas lietojumprogrammās, jo paraksts nav saistīts ar konkrētu lietotāju, bet ir saistīts ar grupu, kurā lietotājs ir dalībnieks. Sistēma nezinās, ka paraksts pieder konkrētam lietotājam, bet zinās, ka paraksts pieder kādam grupas, kurai pieder lietotājs. Tomēr, nobalsot divas reizes nav iespējams, jo gadījumā, ja viens paraksts būs pielietots divreiz – sistēmā par to uzzinās [15].

2.2.8. Ethereum kriptovalūta

Ethereum tīklā ir iebūvēta kriptovalūta – Ether (Eth), kas kalpo diviem mērķiem:

1. Nodrošināt likvīdu kriptovalūtu digitālo aktīvu apmaiņai;
2. Nodrošināt transakcijas apmaksas mehānismu.

Ethereum kriptovalūta iekļauj sevī šīs nominālās vērtības:

- 1 – wei;
- 10^3 – lovelace;
- 10^6 – babbage;
- 10^9 – shannon;
- 10^{12} – szabo;
- 10^{15} – finney;
- 10^{18} – ether.

Tas ir padziļinātāka versija salīdzinājumā ar Euro un Euro centieim. Ethereum platformas izstrādātāji sagaida, ka ether tiks lietots parastajās transakcijās, finney – mikro transakcijās, szabo un wei – tehniskas diskusijās par protokolu, atlikušie, iespējams, būs nepieciešami vēlāk, bet pagaidām nav iekļauti Ethereum klienta lietojumprogrammās. Ether kriptovalūta tika palaista 2015. Gadā, un to varēja iegadāties par Bitcoin kriptovalūtu. Tādā veidā Ether kriptovalūtas pārdošana bija Ethereum platformas finansējums. Ar nopelnīto naudu tika finansēta tālāka Ethereum platformas izstrāde un attīstība. Bitcoin kriptovalūta, kura iegūta Ether kriptovalūtas pardošanas rezultātā tiks lietota, lai samaksātu algas un prēmijas Ethereum platformas izstrādātajiem, kā arī tiks investētā dažādos projektos Ethereum ekosistēmā [9, 19].

Ethereum platforma nodrošina iespēju veidot savu kriptovalūtu. Savas kriptovalūtas izveide aprakstīta 2.3.6.4. nodaļā.

2.3. Lietojumprogrammas izveide iekš Ethereum platformas

2.3.1. Ethereum virtuāla mašīna

Ethereum platformas izpildes modelis nosaka, kā tīkla stāvoklis mainās pēc izpildes bait koda virknes ar komandām. Izpildes modelis tiek noteikts izmantojot formālu virtuālas mašīnas stāvokļa modeli – Ethereum Virtuāla Mašīna (Ethereum Virtual Machine – angļu val.) vai saīsināti EVM. Platformas līgumu kods tiek rakstīts zema līmeņa orientētā uz baitu valodā. Patiesībā līgumu kods tiek rakstīts augsta līmeņa programmēšanas valodā un pēc tam tiek kompilēts uz EVM kodu. Augsta līmeņa programmēšanas valodas, kas ir pieejamas Ethereum platformai, ir aprakstītas 2.3.3. nodaļā. Ethereum baitu kods sastāv no baitu secības, kur katrs baits apzīmē kaut kādu darbību. Darbības lieto trīs vietas informācijas uzglabāšanai:

- Steks (last-in-first-out), kurā var ierakstīt un no kura var lasīt vērtības;
- Atmiņa, patvaļīga izmēra baitu masīvs;
- Līguma krātuve, kurā informācija glabājās pēc atslēga-vērtība principa. Krātuvē, salīdzinājumā ar steku un atmiņu dati var uzglābāties ilgstoši.

Pēc darbības izpildes kā rezultātu kods var atgriezt masīvu no baitiem. No koda ir pieeja naudas summai (Eth), saņēmējam un informācijai no ienākošas transakcijas vai ziņojuma. Koda izpildes laikā pašreizējais stāvoklis glabājas kopā, kura sastāv no:

- Bloka stāvokļa, kurš satur informāciju par visu kontu bilancēm un stāvokļiem;
- Transakcijas vai ziņojuma datiem;
- Līguma koda;
- Līguma krātuves;
- Steka;
- Baita kārtas numura;
- GAS vērtības.

Katrā koda izpildes solī tiek ņemts pēc kārtas nākamais baits un tiek veikta darbība, kura atbilst šīm baitam. Katra no darbībām sava veidā ietekmē uz kopu. Piemēram, ADD darbība izņem no steka divus elementus un ieliek atpakaļ šo elementu summu. Ethereum virtuālajai mašīnai ir neatkarīgs datu uzglabāšanas modelis. Dati tiek uzglabāti asociatīvā masīvā. Tā kā dati glabājas kā daļa no tīkla stāvokļa un tiek uzturēti tādā atmiņa, kā cietie diski, tad, izslēdzot datoru, uzglabājamie dati netiks zaudēti [9, 20].

2.3.2. Tjūringa pilnīga programmēšanas valoda

Ethereum Virtuāla Mašīna ir Tjūring-pilnīga un tas nozīme, ka tā var realizēt jebkādu aprēķinu, kāds vien ir iedomājams, ieskaitot bezgalīgus ciklus. Bezgalīgus ciklus var veidot lietojot kodā tādas operācijas, kuras atgriežas atpakaļ kodā uz konkrētu vietu. Šādus ciklus var veidot arī izsaucot no viena līguma otru. Bezgalīgo ciklu lietošana var raisīt problēmas "Miner" lietotājiem. Ja kodā tiks izveidots bezgalīgs cikls, tad "Miner" lietotājs apstrādās šo kodu bezgalīgi. Ethereum platforma atrisina šī problēmu ar nepieciešamību norādīt maksimālo aprēķinu soļu skaitu, un ja šīs limits ir pārkāpts, tad aprēķināšana beidzas, bet komisijas par aprēķināšanu paliek "Miner" lietotājiem. Piemēram, ļaunprātīgs lietotājs izveido līgumu ar bezgalīgo ciklu un sūta transakciju, kura aktivizē šo līgumu. "Miner" lietotājs izpildot līguma kodu iekļūst bezgalīgā ciklā, bet kādā brīdī GAS vērtība kļūst par nulli un koda izpilde apstājas. Transakcija, kura aktivizēja līgumu paliek aktīva un "Miner" lietotājs saņem komisijas maksājumu par aprēķiniem, bet ļaunprātīgā lietotāja plāns neizdevās un tas tikai iztērēja naudu. Pateicoties $STARTGAS$ mainīgajam "Miner" lietotājs var zināt cik aprēķina soļu skaitu ir konkrētā līguma kodā. Tā kā tad, kad GAS vērtība kļūst par nulli un koda izpilde apstājas, visas izmaiņas tiks atgrieztas atpakaļ, tad nav jāuztraucas par to, ka kāda operācija neizpildīsies līdz galam. Piemēram, nauda tiks noņemta no vienā konta, bet netiks pievienota citam. Ja EVM nebūtu Tjūring-pilniga, tad iespēja veidot bezgalīgus ciklus būtu tikt un tā, jo Ethereum dod iespēju līgumiem veidot jaunus līgumus. Līdz ar to nav jēgas atteikties no Tjūring-pilnas EVM [9, 20, 21].

2.3.3. Ethereum programmēšanas valodas

Ethereum Virtuālā Mašīna, kura ir aprakstīta 2.3.1. nodaļā, atbalsta zema līmeņa orientēta uz steka programmēšanu orientētu baitu valodu. Tomēr eksistē arī augsta līmeņa programmēšanas valodas, kuras kompilējas uz EVM kodu. Dotajā brīdī priekš Ethereum ir pieejamas sekojošās programmēšanas valodas un no tām trīs ir augsta līmeņa programmēšanas valodas:

Seprent – līdzīga Phyton programmēšanas valodai. Sakumā Seprent tiek kompilēts uz LLL valodu un tikai tad uz EVM kodu. Seprent valodu izstrādāja kā papildus projektu Ethereum platformas izgudrotājs – Vitalik Buterin. Pirms tiek izlaista Solidity programmēšanas valoda Seprent bija populārākā no pieejamajām valodām Ethereum lietojumprogrammas izveidei [22, 25].

Solidity – programmēšanas valoda, līdzīga JavaScript programmēšanas valodai. Uz doto brīdī Solidity ir populārākā un visvairāk attīstītā programmēšanas valoda no visām priekš Ethereum pieejamajiem. Visi oficiālie Ethereum lietojumprogrammas izstrādes piemēri ir uzrakstīti ar Solidity. Darba autors uzskata Solidity valodas dokumentāciju par vislabāko no pieejamajām priekš Ethereum programmēšanas valodu dokumentācijām. Solidity dokumentācijā viss ir skaidri un saprotami izskaidrots, sakot ar to kas ir gudrie līgumi un beidzot ar to, kas ir EVM. Valodai ir pieejamas vairākas daudz integrācijas iespējas, pat pieejams izstrādes rīks caur pārlūkprogrammu ar iespēju izvelēties Solidity valodas versiju [22]. Vairāk par izstrādes rīkiem priekš Ethereum ir aprakstīts 2.3.4. nodaļā.

Mutan – līdzīga C programmēšanas valodai [23]. Mutan programmēšanas valodas dokumentācija vairāk ir izveidota izstrādātajiem, kuriem ir pieredze programmēšanā, un iesācējiem būs grūti saprast šī valodu. Var teikt, ka priekš Mutan programmēšanas valodas dokumentācijas ir vairāk špikeris nekā dokumentācija.

LLL (Lisp Like Language) – zema līmeņa programmēšanas valoda līdzīga Assembly valodai [24]. LLL valodas sintakse ir līdzīga Lisp valodai. Būtībā LLL ir iesaiņojums tiešai lietojumprogrammas izveidei EMV kodā.

Izstrādātajiem ir iespēja izvelēties programmēšanas valodu līdzīgu jau sev pazīstamām programmēšanas valodām. Tas atvieglo Ethereum lietojumprogrammas izveidi. Katrai aprakstītajai programmēšanas valodai tīmeklī var atrast dažādus koda piemērus, kas palīdz labāk izprast Ethereum lietojumprogrammas izveidi. Darba autors uzskata, ka programmēšanas valodas izvēlei priekš Ethereum jābalstās uz to, ar kuru valodu izstrādātajam strādāt ir vieglāk.

2.3.4. Izstrādes rīki

Decentralizētas lietojumprogrammas priekš Ethereum platformas ir iespējams izstrādāt dažādos izstrādes rīkos. Sadaļā tiek aprakstīti izstrādes rīki, kuri pieejami Ethereum lietojumprogrammas izveidei.

Mix ir oficiāls Ethereum izstrādes rīks, kurš ļauj veidot gudros līgumus un decentralizētas lietojumprogrammas priekš Ethereum bloku ķēdes. Priekš Mix izstrādes rīka ir pieejama dokumentācija ar piemēriem. Mix iekļauj sevī Solidity programmēšanas valodas koda atkļūdotāju un ir pieejams priekš Windows, OS X un Ubuntu operētājsistēmām. Uz doto brīdī pēdējā aktuālā Mix rīka versija ir 1.2.4. Pēc darba autora pieredzes, izstrādes rīka uzstādīšana gan uz Windows, gan uz OS X ir viegla un neprasa specifiskas zināšanas. Mix izstrādes rīks nodrošina scenāriju izmantošanu, lai testētu un atkļūdot līgumus. Scenārijs ir

lokāla Blockchain ķēde, kur bloku izveide notiek bez izpildes pierādījuma metodes, jo citādāk testēšana būtu ļoti lēna. Scenārijs sastāv no transakcijām, kuras tiek secīgi veiktas. Parasti scenārijs sakas ar transakcijām līguma vai lietojumprogrammas izveidei. Scenāriju arī var pievienot jaunas transakcijas lai testēt un atkļūdot Ethereum lietojumprogrammu. Kā arī scenāriju var modificēt rediģējot vai dzēšot transakcijas. Katras transakcijas informācija scenārijā ir pieejama apskatei. Ar Mix izstrādes rīku var veidot gan lietojumprogrammas loģiku, gan lietojumprogrammas ekrānsaskarnes failus [26].

Truffle ir izstrādes rīks un testēšanas sistēma priekš Ethereum. Izstrādes rīks ir pieejams priekš Windows un UNIX operētājsistēmām. Truffle ir komandrindas izstrādes rīks, un tam nav pieejams grafiskais lietotāja interfeiss kā tas ir Mix izstrādes rīkā. Izstrādātajiem, kuri pazīstami ar komandrindu nesagādās grūtības Truffle rīka lietošanā. Uzstādīt rīku var tikai ar *npm* (Node Project Manager) palīdzību, kurš iet kopā ar node.js [28, 29]. Savukārt lai uzstādītu Truffle rīku iepriekš jāuzstāda node.js. Kad node.js ir uzstādīta, tad uzstādīt Truffle var, palaižot terminālā (UNIX gadījumā) nākamu komandu:

```
Michaels-MacBook-Pro:mist michael$ sudo npm install -g truffle
```

2.3. att. **Truffle izstrādes rīka uzstādīšanas komanda UNIX operētājsistēmās**

Vai komandrindā(Windows gadījumā), atverot to ar administratora tiesībām, nākamo komandu:

```
C:\Users\Asus>npm install -g truffle
```

2.4. att. **Truffle izstrādes rīka uzstādīšanas komanda Windows operētājsistēmā**

Tālāk, izveidojot mapi projektam un pārvietojoties tajā, var izveidot Truffle projektu. Komandas gan Windows, gan UNIX operētājsistēmām ir vienādas, izņemot to, ka priekš UNIX varbūt būs mepieciešāms pievienot komandas sakumā $sudo$, tiesību trūkuma dēļ. Nākamā komanda izveido Truffle projektu:

```
Michaels-MacBook-Pro:truffle michael$ truffle init
```

2.5. att. **Truffle projekta izveides komanda**

Pēc komandas palaišanas tiks izveidotas projektam nepieciešamās mapes un faili. Līgumu failus ar kodu nepieciešams ielikt mapē $contracts$. Līguma failiem jābūt ar paplašinājumu $.sol$ un uzrakstītiem Solidity programmēšanas valodā. Nākamā komanda kompilē projekta līgumus:

```
Michaels-MacBook-Pro:truffle michael$ truffle compile
```

2.6. att. **Truffle projekta līgumu kompilēšanas komanda**

Pēc kompilēšanas līgumi tiks ievietoti projekta mapē $environments/development/contracts$. Līgumu testēšanai ir nepieciešama vēl kas – Ethereum klients testēšanai un

izstrādei, saucama EthereumJS TestRPC [27]. Tas simulē lokālo Blockchain tīklu, kā tas notiek Mix izstrādes rīkā. To var uzstādīt ar komandu:

```
Michaels-MacBook-Pro:truffle michael$ sudo npm install -g ethereumjs-testrpc
```

2.7. att. **EthereumJS TestRPC uzstādīšanas komanda**

Pēc uzstādīšanas jāpārbauda, ka ports, kurā darbojas EthereumJS TestRPC sakrīt ar portu Truffle projekta uzstādījumu failā. Truffle projekta mape satur failu ar nosaukumu $truffle.js$, kurš satur projekta uzstādījumus. Pēc pārbaudes kompilētos līgumus var ieviest lokālajā tīklā palaižot komandu:

```
Michaels-MacBook-Pro:truffle michael$ truffle deploy
```

2.8. att. **Truffle projekta līgumu ieviešana tīklā komanda**

Tālāk var testēt līgumu lokālā Ethereum tīklā. Komandas tīkla testēšanai ir pieejamas EthereumJS TestRPC oficiāla dokumentācijā. Pārbaudīt lietojumprogrammas ekrānsaskarnes failu darbību ar Truffle palīdzību nevar, lai to sasniegtu, ir nepieciešams uzstādīt papildus rīkus. Darba autors aprakstīja uzstādīšanu un līgumu kompilēšanu, lai paradītu, ka tas nav tik grūti kā varētu pirmajā brīdī likties. Daudzas citas pieejamās Truffle rīka komandas un iespējas darbā aprakstītas netiks, bet tās ir pieejamas un labi izskaidrotas rīka oficiālā dokumentācijā.

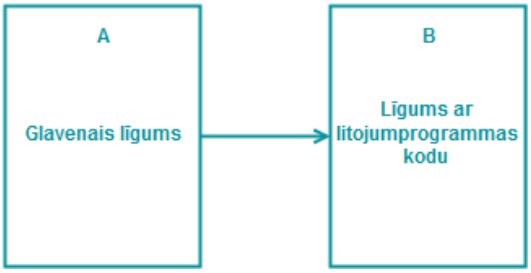
Izņemot oficiālo Ethereum izstrādes rīku eksistē vēl daži citi, kuri ir ļoti līdzīgi Truffle rīkam. Piemēram, Dapple – komandrindas izstrādes rīks, kas atbalsta Solidity programmēšanas valodu. Tas tiek uzstādīts tāpat kā Truffle izstrādes rīks un nodrošina tādas pašas iespējas, izņemot automātisku līguma koda testēšanu [30]. Truffle un Dapple rīkiem ir ļoti līdzīgi izstrādes rīki:

- Populus – gudru līgumu izstrādes rīks, kurš ir uzrakstīts Python programmēšanas valodā [31];
- Eris-PM(Package Manager) – izstrādes rīks kas ļauj veidot līgumus gan lokālajos, gan publiskajos Blockchain tīklos [32];
- Embark – decentralizētas lietojumprogrammas izstrādes rīks, kurš uzrakstīts JavaScript programmēšanas valodā [33].

Ethereum līgumu kodu var rakstīt parastā koda redaktorā, pēc tam kompilējot to kādā no izstrādes rīkiem. Piejāmi Solidity programmēšanas valodas sintakses izcelšana spraudņi priekš tādiem koda redaktoriem kā SublimeText, Atom, Visual Studio un Emacs [34].

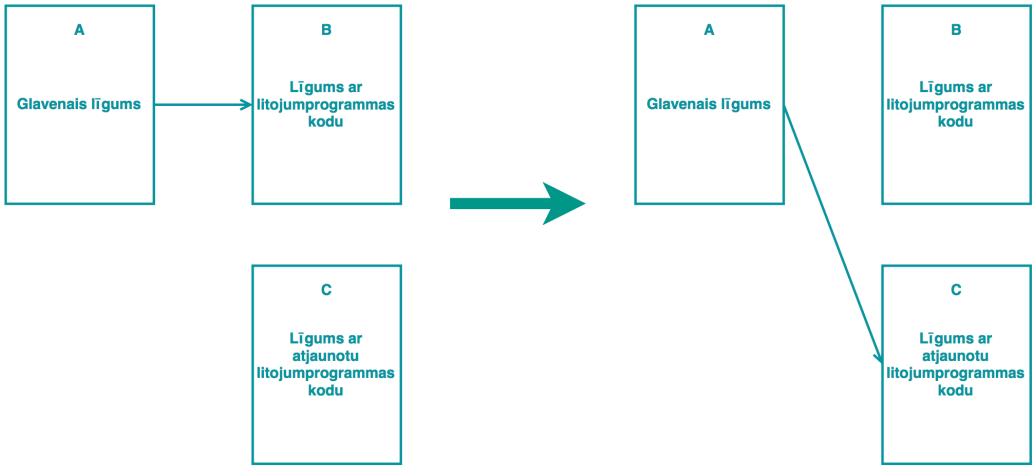
2.3.5. Lietojumprogrammas atjaunošana

Ethereum lietojumprogrammas koda atjaunošana ir aktuāls jautājums, jo, sakarā ar to, ka līgumi tiek iekļauti Blockchain ķēde un tos nevar mainīt, attiecīgi nevar mainīt līguma kodu. Atjaunošanas problēmu var atrisināt izmantojot dažus līgumus lietojumprogrammas realizācijai. Tā kā dati līguma krātuvē var tikt atjaunināti, tad krātuvē var uzglabāt līguma, kurš satur lietojumprogrammas kodu, adresi.



2.9. att. **Lietojumprogrammas realizācija ar koda atjaunošanas iespēju**

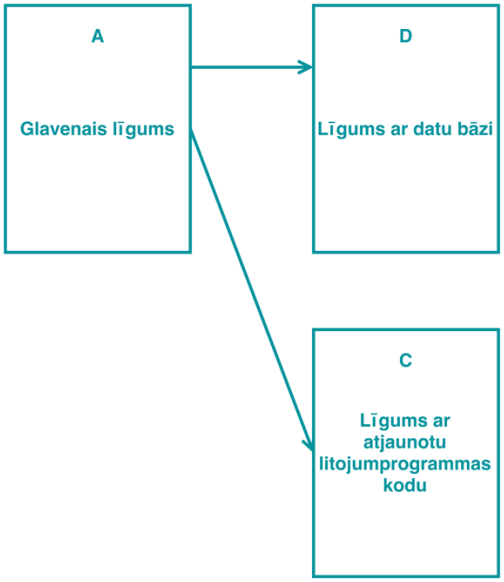
2.9. attēlā redzama aprakstītā pieeja, kurā ir viens galvenais līgums (A), kurš netiks mainīts un transakcijas (un ziņojumi) tiks sūtītas uz viņa adresi. Galvenajā līgumā (A) krātuvē tiks uzglabāta adrese uz līgumu (B) ar lietojumprogrammas kodu. 2.10. attēlā redzama līguma maiņas situācija. Kad būs nepieciešams atjaunot lietojumprogrammas kodu, izstrādātājs izveidos jaunu līgumu (C) ar atjaunotu kodu un nosūtīs transakciju galvenajam līgumam (A) ar jaunā līguma (C) adresi. Protams, pieejai mainīt līguma, ar lietojumprogrammas kodu, adresi jābūt tikai no konkrētā lietotāja konta, jo citādāk katrs, kurš velēsies, varēs samainīt līguma adresi un tā kaitēs lietojumprogrammas lietotājiem.



2.10. att. **Lietojumprogrammas koda atjaunošana**

Eksistē vēl viena problēma – kad līguma, kurš satur kodu, (B) adrese galvenā līguma (A) krātuvē tiek mainīta uz jaunā līguma (C) adresi, tad iepriekšējais līgums (B) vairs nav pieejams galvenajā līgumā (A), un tādēļ nav pieejama tā līguma (B) krātuve, kurā var glabāties dati, kas ir, nepieciešami lietojumprogrammas korektai darbībai, dati. Līdz ar to var

veidot datu uzglabāšanu atsevišķā līgumā (D) un sniegt piekļuvi tam no jauna līguma (C), kurš satur atjaunotu lietojumprogrammas kodu. Šī pieeja ir redzama 2.11. attēlā.



2.11. att. **Lietojumprogramma ar atsevišķu līgumu datu uzglabāšanai**

Var arī uzglabāt visus datus galvenajā līgumā (A), neveidojot atsevišķu līgumu priekš datiem (D). Tā kā pieeja atjaunot līguma, ar lietojumprogrammas kodu, adresi būs tikai konkrētam lietotājam, tad lietojumprogramma vairs nebūs decentralizēta. Tādu problēmu iespējams atrisināt, lietojot galvenā līguma (A) pārvaldībai decentralizēto autonomu organizāciju. Decentralizētas autonomās organizācijas aprakstītas 2.3.6.2. nodaļā.

2.3.6. Lietojumprogrammas

2.3.6.1. Decentralizēta failu uzglabāšana

Uz Ethereum platformas ir iespējams uzbūvēt sistēmas priekš tiešsaistes failu glabāšanai, līdzīgas Google Drive un Dropbox, tikai decentralizētas. Tādas sistēmas dod lietotājiem iespēju augšupielādēt failus speciālā mapē, kura pēc tam tiek sinhronizēta ar kompānijas serveri. Nedaudzi tiešsaistes failu glabāšanas risinājumi piedāvā bezmaksas kontus, kuriem ir vairāk par divdesmit gigabaitiem vietas failu glabāšanai. Dažas cenas par pakalpojumiem salīdzināmas ar cieto disku cenām. Ethereum platformas līgumi var tikt pielietoti decentralizētas failu glabāšanas sistēmas izveidei, kur ikvienam tīkla lietotājam būs iespēja nopelnīt izīrējot sava cietā diska vietu. Uz Ethereum tāda sistēma varētu strādāt līdzīgi BitTorrent failu koplietošanas protokolam, kur viens fails (vai arī faila daļa) tiek uzglabāts dažādās ierīcēs un, kad lietotāji savienojas viens ar otru, tad tie apmaiņas ar informāciju par to, kādi dati katram eksistē, un ja kādam no lietotājiem nav kādas faila daļas, tad tas pieprasa

neeksistējošo faila daļu. Saņemot faila daļu lietotājs pārbauda tas jaucējkodu un paziņo pārējiem, ka viņam arī ir šīs fails, lai visi pārējie lietotāji arī varētu lejupielādēt failu no lietotāja ierīcēs. Tipiska Ethereum decentralizēta sistēma failu uzglabāšanai varētu strādāt sekojoši:

1. Līgums, saņemot informāciju, sadalīs to blokos. Katrs bloks tiks šifrēts. Un no visiem blokiem tiks veidots Merkle koks. Līgumā glabāsies transakcijas sūtītāja adresē un norādīs uz koka sakni;
2. Līgums izveidos jaunu līgumu, kurš katrus N blokus ņems, izmantojot nejaušu indeksu no Merkle koka (lietojot iepriekšējā bloka jaucējkodu kā avotu nejaušajiem skaitļiem), un pārsūtīs X Eth mērvienības pirmajam, kurš pierādīs bloka uzglabāšanas īpašumtiesības zem apskatāma indeksa. Pēc tam varēs pārbaudīt vai fails eksistē. Fails eksistēs, ja jaunais līgums kādam kaut ko maksas.

Kad faila īpašnieks velēsies dabūt failu, viņš varēs lietot mikro maksājumu protokolu, maksājot par kādu nosacītu kilobaitu skaitu. Neskatoties uz to, ka faila īpašniekam ir jāpaļaujas uz lielu nejaušo personu skaitu, kuri uzglabā failu, risks varētu pat neeksistēt, pielietojot noslēpuma atdalīšanas tehnoloģiju [35, 36, 37].

2.3.6.2. Decentralizētas autonomas organizācijas

Decentralizēta autonoma organizācija vai saīsināti DAO ir virtuāla organizācija, kuru veido Ethereum tīkla lietotāji. DAO piemīt sekojošā īpašība – pēc noteiktas locekļu skaita piekrišanas, piemēram, vairāk par 50%, lēmumi var tikt pieņemti uz organizācijas vārda. Lēmumi varētu tikt pieņemti, piemēram, par organizācijas uzkrājumu tērēšanu vai organizācijai piederošas lietojumprogrammas koda modificēšanu, kā tiek aprakstīts 2.3.5. nodaļā. Tāpat arī ar locekļu nobalsošanu DAO kods varētu pēc nepieciešamības tikt mainīts. Decentralizētas autonomas organizācijas ir ļoti līdzīgas parastām organizācijām, izņemot to, ka DAO rīkojumi notiek uz Blockchain tehnoloģijas pamata. Ja DAO locekļiem ir vienāds balsu svars, tad var būt problēma ar to, ka viens lietotājs var piedalīties balsošanās vairākkārtīgi, ja tiem ir vairāki konti Ethereum tīklā un ja DAO akcijas ir brīvi tirgotas. To varētu atrisināt veidojot valdes locekļu grupu, kurā tiktu iekļauti cilvēki, kuriem var uzticēties, un tad lēmumus pieņems valdes locekļi, nevis visi DAO locekļi kopā. Locekļu balsu svars arī varētu būt atkarīgs no loceklim piederošo DAO akciju skaita, bet tāda gadījumā balsošana nebūs godīga. Vienkārša DAO līguma īstenošanā ir trīs transakciju tipi:

- $[0, i, K, V]$ – reģistrē piedāvājumu ar numuru i, kas satur atslēgas-vērtības pāri ar vērtību V (varētu būt līguma, ar jaunu DAO kodu, adrese), kuru ir jāievieto krātuvē

pēc indeksa K (atslēga pēc kuras tiek sasniegta līguma, ar tagadējo DAO kodu, adrese);

- $[0, i]$ – reģistrē balsi par piekrišanu piedāvājumam ar numuru i;
- $[2, i]$ – piedāvājums ar numuru i, ja ir savākts nepieciešamais balsu skaits, stāvas spēkā.

Decentralizētā autonomā organizācijā varētu būt realizēta iespēja balsot ne tikai par koda izmaiņu vai transakcijas veikšanu no organizācijas vārda, bet arī par biedru dzēšanu vai pievienošanu, kā arī varētu tikt realizēta balsu deleģēšana – locekļi var nodot savu balsi citiem locekļiem, tādā veidā dodot iespēju uzticēt koda atjaunošanu vai jaunu biedru pievienošanu speciālistiem. DAO krātuvē nepieciešams uzturēt sarakstu ar organizācijas biedriem, var arī uzturēt reģistru par visām izmaiņām līguma krātuvē (kodā) un sarakstu ar visiem, kuri balsoja par kādam izmaiņām [9].

2.3.6.3. DNS reģistrēšanas sistēma

Uz Ethereum platformas nav grūti izveidot decentralizēto DNS sistēmu. DNS serveri pārveido domēna vārdu par IP adresi. Saknes DNS serveri ir tīmekļa infrastruktūras pamata elements. Pirmā alternatīvā kriptovalūta (*altcoin* - angļu val.) ir Namecoin un tika izveidota ar mēģinājumu lietot Blockchain līdzīgu Bitcoin ķēdei, lai izveidotu decentralizētu domēna vārdu reģistrēšanas sistēmu [1, 9]. Uz Ethereum tāda sistēma varētu būt realizēta attiecīgi 2.12. attēlam.

```
function register(string name, string value) {
    if (domainNames[name] == address(0x0)) {
        domainNames[name] = value;
    }
}
```

2.12. att. **Domēna vārdu reģistrēšanas funkcija**

Pirmā koda rinda satur funkcijas $register()$ definēšanu, kura saņem divus argumentus, kur $name$ ir domēna vārds un $value$ ir IP adrese, uz kuru domēna vārdam ir jānorāda. Otrā koda rindā tiek pārbaudīts, vai līguma krātuvē eksistē atslēga ar domēna vārdu, kurš tiek saņemts transakcijā. Trešā rinda izpildīsies ja atslēga ar saņemto domēna vārdu vēl neeksistē krātuvē un tad krātuvē tiks pievienots jauns ieraksts ar saņemtu atslēgas-vērtības pāri – domēna vārds un IP adrese. Piemēra kods apraksta to, ka informācija varbūt pievienota, bet nevar būt mainīta vai dzēsta. Jebkurš var piereģistrēt domēna vārdu un domēna vārdam tiks piešķirta IP adreses vērtība, un šī reģistrācija būs mūžīga. Sarežģītāks līgums domēna vārdu reģistrēšanai varētu sevī iekļaut iespēju mainīt domēna vārda IP adresi un īpašnieku.

Aprakstīto kodu arī varētu lietot, piemēram, autentificēšanas sistēmā, kur atslēga būs e-pasta adrese un vērtība būs parole.

2.3.6.4. Savas kriptovalūtas izveide

Tā kā kriptovalūtas vienīga veicamā darbība ir no konta X atņemt A vienības un pievienot A vienības Y kontam, tad to viegli var realizēt uz Ethereum platformas. Protams kriptovalūtas līguma kodā ir jāpārbauda, ka X kontam ir vismaz A vienības pirms transakcijas sakuma un ka tieši X iniciēja transakciju. Koda piemērs Serpent programmēšanas valodā kriptovalūtas līguma loģikas realizēšanai ir redzams 2.13. attēlā.

```
function send(address recipient, uint value) {
    if (accBalances[msg.sender] >= value) {
        accBalances[msg.sender] = accBalances[msg.sender] - value;
        accBalances[recipient] = accBalances[recipient] + value;
    }
}
```

2.13. att. **Kriptovalūtas sūtīšanas funkcija**

Pirmajā koda rindā tiek definēta $send()$ funkcija, kura saņem kā ienākošus parametrus divus argumentus $recipient$ – konta adrese, kam tiks sūtīta valūtas summa un $value$ – sūtāmas valūtas daudzums. Otrā koda rindā tiek pārbaudīts vai sūtītājam ir pieejamas vismaz $value$ kriptovalūtas vienības. Trešā un ceturtā koda rindas izpildās tikai, ja sūtītājam ir pieejamas vismaz $value$ kriptovalūtas vienības. Trešajā koda rindā no sūtītāja konta tiek atņemtas $value$ kriptovalūtas vienības un ceturtā koda rindā $value$ kriptovalūtas vienības tiek pievienotas $recipient$ kontam. Protams, piemērā kodā trūkst sākotnēja valūtas piešķiršanu un funkcijas, kura ļaus pieprasīt informāciju par patvaļīga adreses bilanci. Nedaudz paplašinātākas kriptovalūtas realizācija ir pieejama 1. Pielikumā. Koda aprakstīta EuroCoin kriptovalūtas izveidi uz Ethereum platformas.

Nedaudz pārveidojot kriptovalūtas līguma kodu to varēs izmantot kā organizācijas akcijas zīmi. Uzstādot kādu laiku uz zīmes pārdošanu par Eth kriptovalūtu pēc līguma iekļaušanas Blockchain ķēdē var pievienot zīmēm vērtību. Pēc zīmes pārdošanas beigšanas jaunas zīmes netiks radītas. Tādas zīmes var piederēt DAO, kas aprakstīta 2.3.6.2. nodaļā, un tās var pārstāvēt organizācijas akcijas.

2.4. Secinājumi

Neskatoties uz to, ka Ethereum platforma vēl atrodas izstrādes stadijā, tā jau sniedz lietojumprogrammas izstrādes iespējas. Ethereum protokols tieši neatbalsta nekādas lietojumprogrammas, bet ar Tjūringa-pilnīgu programmēšanas valodu līgumi varbūt izveidoti patvaļīgi un tos var uzskatīt par lietojumprogrammām. Atšķirībā no precīzi specializētiem projektiem uz Blockchain pamata, Ethereum platformai ir plaši specializēta arhitektūra. Augsta līmeņa programmēšanas valodas eksistence padara Ethereum lietojumprogrammas izstrādi vieglāku un līdzīgu parasto lietojumprogrammas izstrādei, kurai daudzi pieraduši. Ethereum ir daudz vairāk nekā parasta kriptovalūta un platformai ir liels potenciāls nākotnē.

3. Kopfinansēšanas tiešsaistes sistēmas izstrāde

Kopfinansēšanas tiešsaistes sistēmas ir tīmekļa resursi, kuri piedāvā reģistrētiem lietotājiem iespēju izveidot projektu, ar tā mērķa aprakstu un noradīt naudas summu, kuru vēlas savākt projekta īstenošanai. Kopfinansēšanas tiešsaistes sistēmas piedāvā reģistrētiem un nereģistrētiem lietotājiem iespēju nosūtīt noteiktu naudas summu un/vai nenoteiktu naudas summu konkrētam projektam. Tādas sistēmas piemērs ir Kickstarter un Indiegogo sistēmas. Pirms veidotie projekti kļūst publiski pieejami tos pārbauda sistēmas administratori. Pēc kopfinansēšanas projekta veiksmīgas beigšanas, savāktā nauda tiek pārskaitīta projekta autoram, lai tas varētu īstenot projektu. Parasti par projekta finansēšanu, parasti, lietotāji saņem kādas simboliskas balvas (krekls, uzlīmes un citas), ierīci vai pakalpojumu, kura īstenošanai tiek izveidots projekts. Balva ir atkarīga no summas, ar kuru lietotājs nofinansējis projektu. Dažreiz, var teikt, ka lietotājs pircis ierīci, vai pakalpojumu, kurš būs pieejams tikai projekta finansēšanas veiksmīgas pabeigšanas gadījumā. Pieejamas kopfinansēšanas sistēmas ir centralizētas un tas nozīme, ka lietotājiem jāuztic savu naudu trešajai personai. Piemēram, kamēr netiks savākta projekta realizēšanai vajadzīgā naudas summa, jau uzkrātā nauda atrodas pie kopfinansēšanas īpašnieka.

Uz Ethereum platformas ir iespējams izstrādāt decentralizēto kopfinansēšanas sistēmu, kurā lietotajiem nevajadzēs uzticēties trešajai personai un par projekta finansēšanu lietotāji varēs dabūt projekta akcijas zīmes, kuras ir vērtīgākas par simboliskām balvām. Protams, arī uz Ethereum var izveidot kopfinansēšanas sistēmu ar iespēju par projekta finansēšanu dabūt projektā aprakstīto ierīci vai pakalpojumu, bet lietotājiem būs jāuzticas projekta autoriem tāpat kā centralizētās sistēmās. Var secināt, kā decentralizēta kopfinansēšanas sistēmā, salīdzinājumā ar centralizēto, lietotājs papildus saņem iespēju dabūt projekta akcijas zīmes un viņa nauda nav kontrolēta ar trešo personu, bet ir kontrolēta ar līgumu. Decentralizētā sistēmā projektu apstiprināšanai par kļūšanu publiskiem var lietot balsošanas sistēmu. Balsošanā var piedalīties vai nu jebkurš Ethereum tīkla lietotājs, vai arī kāda lietotāju grupa.

Darba autors izvēlējās izstrādāt decentralizēto kopfinansēšanas sistēmas prototipu uz Ethereum platformas, lai labāk saprastu platformas darbību un izstrādes iespējas tajā. Darba autors ir iepriekš izstrādājis kursa darbu par tiešsaistes ziedošanas sistēmas arhitektūru izveidi, un tas dos iespēju salīdzināt centralizētas un decentralizētas sistēmas izstrādi. Kopfinansēšanas sistēmas prototipa izstrādē tiks lietota darba otrajā daļā apgūtā un aprakstītā informācija. Ņemot vērā to, ka Ethereum platforma vēl ir tikai izstrādes stadijā, tad šīs sistēmas izstrāde ir tikai pārbaude, lai izpētītu un iepazītos ar to, uz ko Ethereum platforma ir spējīga.

3.1. Sistēmas apraksts

Darba autors noteica īpašības, kurām jāpiemīt izstrādājamajai sistēmai:

- Sistēmai jāpieder un jābūt pārvaldāmai ar DAO;
- Decentralizētas autonomas organizācijas, kurai piederēs sistēma, peļņas daļu ir jādala starp visiem kontiem, kuriem pieder šīs DAO akcijas zīmes, proporcionāli akcijas zīmes daudzumam.
- Pirms projekts kļūst pieejams finansēšanai, tam jābūt apstiprinātam ar kopfinansēšanas sistēmas DAO locekļu balsošanu;
- Sistēmā jābūt iespējai atgriezt naudu lietotājiem, kuri finansēja projektu, gadījumā ja projekts nesavāc nepieciešamo naudas summu;
- Tīkla lietotājiem un līgumiem (ieskaitot DAO) jābūt iespējai izveidot projektu sistēmā;
- Ar projekta veiksmīgas finansēšanas beigšanu jābūt izveidotai DAO, kurai projekts piederēs, un tas akcijas zīmēm. Katram projekta finansētājam jābūt piešķirtām projekta DAO akcijas zīmēm proporcionāla izmēra finansēšanas summai Eth kriptovalūtā. Pēc projekta DAO akcijas zīmes izsniegšanas finansētājiem jaunas zīmes netiks radītas. Summai, kuru projekts savāca Eth kriptovalūtā jābūt piešķirtai projekta DAO. Projekta DAO jābūt pārvaldīšana ar projekta izveidotāju;
- Tīkla lietotājiem un līgumiem (ieskaitot DAO) jābūt iespējai finansēt projektu sistēmā;

3.1.1.Sistēmas lietotāji

Eksistēs divas sistēmu lietotāju grupas – lietotāji, kuri veidos projektus un lietotāji, kuri finansēs projektus. Viens un tas pats lietotājs vienlaicīgi varbūt abu grupu loceklis.

3.1.2.Sistēmas funkcijas

Izstrādājamā kopfinansēšanas sistēma saturēs sekojošas funkcijas:

- $createProject$ – projekta izveides funkcija, kura kā ieeja datus saņems projekta nosaukumu, aprakstu, ilgumu un nepieciešamo projekta realizēšanai Eth kriptovalūtas summu;
- $acceptProject$ – funkcija, kura saņems transakcijas tikai no sistēmas īpašnieka (DAO) un uzstādīs projektu par publiski pieejamu finansēšanai;

- $fundProject$ – saņems transakcijas ar ieejas datiem par projektu, kuru sūtītājs vēlas finansēt un finansējuma summu Eth kriptovalūtā;
- $removeProject$ – funkcija projekta dzēšanai, tā saņems transakcijas ar projekta identifikatoru tikai no sistēmas īpašnieka (DAO) un arī varētu būt izsaucama no citas sistēmas līguma funkcijas. Nepieciešama projektu, kurus neapstiprināja administrācija vai kuri nesavāca nepieciešamo summu realizēšanai, dzēšanai. Projektus nav jāuzglabā sistēmā, jo viss, kas ticis iekšā Ethereum bloku virknē, paliek tur uz visiem laikiem, attiecīgi, projektu izveides vēsturi varēs redzēt apskatot datus blokus, kuri iet pirms bloka, kurš jau nesatur tikko izdzēsto projektu;
- $returnFunds$ – funkcija atgriež savākto naudas finansētājiem, ja projekts nav savācis nepieciešamo Eth kriptovalūtas summu noteikta laikā. Transakcijas funkcija saņems tikai no sistēmas īpašnieka (DAO). Kā ieejas datus funkcija saņems projekta identifikatoru;
- $createProjectDAO$ – saņems transakcijas tikai no sistēmas īpašnieka (DAO) un izveidos jaunu DAO, kurai piederēs projekts, ja projekta finansēšana pabeidzas veiksmīgi. Kā ieeja datus funkcija saņems projekta identifikatoru. Kā jauno DAO īpašnieku funkcija uzstādīs projekta izveidotāju;
- $createTokens$ – funkcija izveidos jaunas DAO akciju zīmes un piešķirs katram kontam, kurš finansēja projektu, akciju, zīmju skaitu, proporcionālu finansētajai summai. Ieejā funkcija saņem adresi, kurai piederēs akciju zīmes, un projekta identifikatoru;
- $sendFunds$ – sūtīs decentralizētai autonomai organizācijai projekta finansēšanas laikā savāktu Eth kriptovalūtas summu (izņemot sistēmas komisijas maksu), kurai šis projekts piederēs. Transakcijas funkcija pieņems tikai no kopfinansēšanas sistēmas īpašnieka (DAO) un no sistēmas funkcijām. Ieejas dati būs sūtama Eth kriptovalūtas summa un DAO, kurai pieder projekts, adrese;
- $payInterest$ – funkcija sūtīs daļu no summas (sistēmas peļņu), kuru savāca projekts, sistēmas īpašniekam (DAO). Sistēmas pelņa tiks uzstādīta sistēmas kodā kā konstanta, piemēram, divi procenti no kopējās summas, kuru savācis projekts. Transakcijas ar ieejas datiem par projektu, funkcija saņems no sistēmas īpašnieka (DAO) vai sistēmas funkcijām.

Ethereum līgumu funkcijas ir lietojumprogrammas darbību noteikumi, kurus nevar mainīt, jo nevar mainīt līguma kodu (izņemot 2.3.5. nodaļā aprakstīto).

3.2. Programmēšanas valodas izvēle

Kopfinansēšanas sistēmas izstrādei darba autors izvēlējas Solidity augsta līmeņa programmēšanas valodu. Pamatojoties uz aprakstīto 2.3.3. nodaļā, Solidity ir populārākā un visvairāk attīstītā programmēšanas valoda no visām priekš Ethereum pieejamajām. Tā tiek lietota visos oficiālajos koda piemēros, kā arī Solidity sintakse ir līdzīga JavaScript programmēšanas valodas sintaksei, kuru pārzina darba autors. Visi līgumu koda piemēri no pielikumiem, kas minēti darbā, darba autors rakstījis Solidity programmēšanas valodā un programmēšanas valodu mainīšanu kopfinansēšanas sistēmas prototipa izstrādei neuzskata par labu pieeju, ņemot vērā to, ka visas priekš Ethereum platformas pieejamas augsta līmeņa programmēšanas valodas tik un tā tiks kompilētas līdz zema līmeņa programmēšanas valodas kodam.

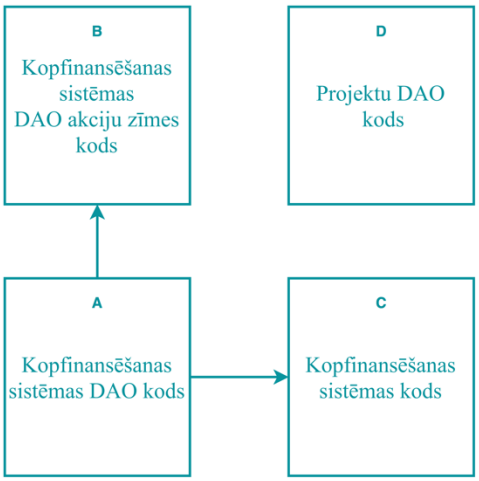
3.3. Izstrādes rīka izvēle

Darba autors nolēma izstrādāt sistēmu lietojot Mix izstrādes rīku, jo tas ir oficiāls Ethereum izstrādes rīks, kurš atbalsta Solidity programmēšanas valodu un iekļauj sevī lokālo Blockchain ķēdi lietojumprogrammas testēšanai un atkļūdošanai. Vēl viens izvēles pamatojums ir tas, ka Mix izstrādes rīks ir vienīgais no 2.3.4. nodaļā aprakstītajiem rīkiem, kuram ir lietotāja saskarne, kas ievērojami atvieglo sistēmas izstrādi un testēšanu.

3.4. Sistēmas arhitektūra

Ethereum lietojumprogrammas arhitektūras var apskatīt kā līgumu hierarhijas. Attēlā 3.1. ir redzama kopfinansēšanas sistēmas prototipa arhitektūra. Līgums A satur decentralizētas autonomas organizācijas kodu, kurai piederēs kopfinansēšanas sistēma. Savukārt līgums B satur kodu, kurš apraksta DAO, kurai piederēs sistēma, akciju zīmes. Kad tiks palaista kopfinansēšanas sistēma noteiktu laiku būs iespēja iegadāties DAO, kurai sistēma piederēs, akcijas zīmes, tādējādi pievienojot sistēmai vērtību. Līdzīgi kā Kickstarter un Indiegogo kopfinansēšanas sistēmas arī izstrādājamā sistēma var pelnīt naudu, ņemot no veiksmīgi pabeigtajiem projektiem procentus (viena līdz piecu procentu apjomā). Pelņa tiks lietota DAO darbinieku algas izmaksai un komisiju apmaksai, veicot transakcijas no DAO vārda, piemēram, atgriežot Eth kriptovalūtas vērtību kontiem par projekta, kurš nesavāc nepieciešamu summu, finansēšanu. Daļa no pelņas varētu tikt sadalīta starp DAO akcijas zīmes īpašniekiem. Līgums C satur kopfinansēšanas sistēmas kodu un datus par projektiem.

Tā kā Solidity programmēšanas valoda nodrošina iespēju veidot datu struktūras, tad nav nepieciešamības veidot atsevišķus līgumus datu glabāšanai un tos var uzglabāt galvenajā līguma krātuvē, lai izvairītos no liekām transakcijām, tādejādi ietaupot laiku transakcijas apstrādei un naudu transakcijas komisijas apmaksai.



3.1. att. **Kopfinansēšanas sistēmas arhitektūra**

Tā kā veiksmīgas projekta finansēšanas gadījumā vajadzēs izveidot DAO, kurai piederēs projekts un savākta Eth kriptovalūtas summa, tad vēl nepieciešams līgums D, kura kods tiks lietots, lai veidotu jaunas DAO.

3.5. Sistēmas izstrāde

Attēlā 3.2. ir redzama kopfinansēšanas sistēmas koda daļa, kurā ir aprakstīta projekta datu glabāšana sistēmā. Pirmajā rindā tiek deklarēts mainīgais kurš saturēs atslēgas-vērtības pārus, kur atslēga ir projekta unikāls identifikators un vērtība ir datu struktūra, kura saturēs datus par projektu. Projekta datu struktūra saturēs datus par projekta izveidotāju, projekta nosaukumu un aprakstu, administratora komentārus, projekta ilgumu, nepieciešamo projekta realizēšanai summu Eth kriptovalūtā, jau savākto summu Eth kriptovalūtā, informāciju par to, kurš un kādu summu projektam finansējis, un ierakstu par to , vai projekts ir publiski pieejams finansēšanai vai nav.

```
mapping (uint => Project) private projects; //funding projects

struct Project {
    address owner; //project owner address
    string name; //project name
    string description; //project description
    string adminComment; //admin comment about project and project owner
    uint duration; //project duration in days
    uint goal; //project goal in Eth
    uint funded; //already collected amount in Eth
    mapping (address => uint) funds; //address that fund the project with funded amount un Eth
    bool accepted; //is project accepted by admins
}
```

3.2. att. **Kopfinansēšanas projektu glabāšanas struktūra**

Datus par to, kurš un cik nofinansēja darba, tiks uzglabāti atslēgas-vērtības pāros, lai projekta beigšanas veiksmīgas finansēšanas gadījumā zinātu, cik projekta DAO akciju zīmes piešķirt konkrētajam sūtītājam, vai, projekta beigšanas neveiksmīgas finansēšanas gadījumā, zinātu, cik vienības Eth kriptovalūtā atgriezt konkrētajam sūtītajam. 3.1.2. nodaļā aprakstītas funkcijas lietos mainīgo $projects$, lai piekļūt projektu datiem.

Īpaša uzmanība jāpievērš jauna līguma priekš DAO, kurai piederēs projekts, izveidei no kopfinansēšanas sistēmas koda. Jauna DAO līguma izveides koda daļu var redzēt 3.2. attēlā. Eksistē divi līgumi:

- $FundingSystem$ – kopfinansēšanas sistēmas līgums;
- $ProjectDAO$ – decentralizētas autonomas organizācijas līgums, kurš tiks lietots jaunu DAO, kuriem piederēs projekti, izveidei.

Līgums $FundingSystem$ satur funkciju $createProjectDAO$, kura aprakstīta 3.1.2. nodaļā. Kā ieejas parametru funkcija saņem projekta identifikatoru. Koda rindiņa iekš $createProjectDAO$ funkcijas apraksta jauna līguma izveidi. Tiek veidots līgums $ProjectDAO$, kura konstruktors saņem kā ieejas parametru tās īpašnieka adresi. Koda daļa $projects[projectId].owner$ iegūst projekta, kura identifikators padots funkcijai, izveidotāju. Kā jauna līguma izveides rezultāts tiek atgriezta izveidotā līguma adrese, kura

tālāk tiek izmantota (lietojot $sendFunds$ funkciju), lai pārsūtīt uz to Eth kriptovalūtas summu, kuru projekts savācis. Papildus jauna līguma adrese tiek izmantota arī, lai izveidoti jaunas DAO akciju zīmes – funkcija $createTokens$.

```
contract FundingSystem {
    ...
    function createProjectDAO(uint projectId) public {
        ...
        address projectDAOAdress = new ProjectDAO(projects[projectId].owner);
        ...
    }
    ...
}

contract ProjectDAO {
    ...
    function ProjectDAO(address owner) {
        ...
    }
    ...
}
```

3.3. att. **Jauna līguma izveides funkcija**

Koda darbības rezultātā izveidota jauna decentralizēta autonoma organizācija, kurai pieder projekts un kuras īpašnieks ir projekta izveidotājs. Kopfinansēšanas sistēmas izstrādes laikā darba autors vēl izveidoja DAO un DAO akcijas zīmes līgumus, lietojot kā piemēru iepriekš izveidotas savas kriptovalūtas un zīmes līgumu kodu.

3.6. Kopfinansēšanas sistēmas nākotnes uzlabojumi

Izstrādātā kopfinansēšanas sistēma var tikt un būs papildināta un uzlabota. Nākotnes uzlabojumi un papildinājumi varētu būt sekojoši:

- Nodrošināta iespēja kā projekta īpašnieku lietojot jau eksistējošu un projekta izveidotājam piederošu DAO;
- Ieviests iespēja mainīt kopfinansēšanas sistēmas īpašnieku;
- Izveidota funkcija komisijas, kuru saņem sistēma par veiksmīgu projekta finansēšanu, summas maiņai;
- Ieviests saraksts ar adresēm, kurā varēs ievietot ļaunprātīgus sistēmas lietotājus adreses.

Ethereum platforma uz doto brīdī nodrošina tikai lietojumprogrammas loģikas uzturēšanu, bet lietotāja saskarni ir jārealizē ar citu rīku palīdzību. Izveidot lietotāja saskarni var lietojot web3.js JavaScript programmēšanas valodas bibliotēku, kas ļauj komunicēt ar Ethereum tīklu [38]. Lietotāja saskarnes izveide arī ir svarīga lieta kopfinansēšanas sistēmas uzlabošanā, lai tai varētu tikt klāt no parastas pārlūkprogrammas.

3.7. Secinājumi

Decentralizēta kopfinansēšanas sistēma nav daudz uzticamāka par centralizētajām, jo jebkurš var izveidot kontu Ethereum tīklā, un, izveidojot sistēmā projektu, veiksmīgas projekta finansēšanas gadījumā saņems visu nofinansēto naudu sev un varēs to izmantot ļaunprātīgos nolūkos. Ja Ethereum eksistētu reģistrs, kurš uzglabātu reālas personas vai uzņēmumus un tiem piederošus kontus, tad būtu iespēja noteikt projekta izveidotāju un ļaunprātīgas rīcības gadījumā ciestu personas vai uzņēmuma reputācija. Reputācijas datus arī varētu uzglābāt iekš Ethereum. Projektu veidotāji būs ieinteresēti ne maldināt lietotājus, kuri finansēs projektu, jo pēc projekta apraksta tie varēs uzzināt daudz svarīgas informācijas par projekta veidotāju. Lietotāji, kuri finansēs projektu vairāk uzticēsies projekta veidotājiem, ja tie būs reālas organizācijas.

Ethereum līgumi ir līdzīgi klasēm citās programmēšanas valodās. Mix izstrādes rīks ļoti labi pilda savus pienākumus, sniedzot Ethereum lietojumprogrammas izstrādes platformu ar iespēju veikt testēšanu. Izstrādes rīks izceļ koda sintaksi, un tajā ir ērti lasīt kodu. Vēl liels pluss Mix izstrādes rīkā ir kļūdas ziņojuma attēlojums uzreiz uzrakstot koda rindu nepareizi. Tas palīdz uzreiz novērst un izlabot kļūdas.

Noslēgums

Darba autors izpētīja Blockchain tehnoloģiju un Ethereum platformu. Tika pētītas arī Ethereum lietojumprogrammas izstrādes iespējas. Noteiktie uzdevumi tika paveikti, un uzstādītais mērķis tika sasniegts – ir izpētīta Ethereum platformas un ir izstrādāts kopfinansēšanas sistēmas prototips, balstoties uz tam noteiktajām prasībām.

Apkopojot veikto izpēti un no tās gūtās atziņas, galvenie darba autora secinājumi ir šādi:

- Blockchain ir datubāze, kurā var glabāt ierakstus par objektiem un notikumiem no reālās dzīves;
- Blockchain tehnoloģija joprojām ir diezgan sarežģīts tehniskais risinājums, kas turpina attīstīties;
- Ethereum platformai ir plaši specializēta arhitektūra un uz tās pamata var veidot dažādu veidu projektus;
- Lietojumprogrammu izstrādei Ethereum vidē pieejamo programmēšanas valodu sintakse ir ļoti līdzīga vairākām populārām programmēšanas valodām, kas atvieglo izstrādi;
- Ethereum platformai ir liels potenciāls un neskatoties uz to, ka tā vēl atrodas izstrādes stadijā, uz tās jau ir iespējams veidot funkcionējošas decentralizētas sistēmas.

Izstrādāto kopfinansēšanas sistēmu nākotnē plānots attīstīt un uzlabot, lai paplašinātu pieredzi sistēmu izstrādē uz Ethereum platformas. Darba autors turpinās sekot Ethereum platformas izstrādei un pielietot iegūtas zināšanas praksē.

Literatūras saraksts

1. Melanie Swan, Blockchain, O'Reilly Media, 2015 – 152 lpp.
2. Bitcoin: A Peer-to-Peer Electronic Cash System [tiešsaiste] – [atsauce 29.05.2016]. Pieejams: https://bitcoin.org/bitcoin.pdf
3. R.C. Merkle, «Protocols for public key cryptosystems,», In *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, 1980 – 122-133 lapas.
4. Bitcoin Part 3 – Hashes, Public Key Cryptography "for Dummies" and the Block Chain – CFB Scientific Translations and Consulting [tiešsaiste] – [atsauce 04.05.2016]. Pieejams: http://www.cfbtranslations.com/bitcoin-part-3-hashes-public-key-cryptography-for-dummies-and-the-block-chain/
5. Block hashing algorithm – Bitcoin Wiki [tiešsaiste] – [atsauce 24.05.2016]. Pieejams: https://en.bitcoin.it/wiki/Block_hashing_algorithm
6. Ethereum: The Road Ahead – Google Slides [tiešsaiste] – [atsauce 19.05.2016]. Pieejams: https://docs.google.com/presentation/d/1JKKef7NVjgtRZFsTcAteS4VtHG0aYu7TApZhWUENbWQ/preview?slide=id.gf6277fbfe_0_13
7. Elliptic Curve Cryptography [tiešsaiste] – [atsauce 07.05.2016]. Pieejams: http://www.infosecwriters.com/text_resources/pdf/Elliptic_Curve_AnnopMS.pdf
8. About the Ethereum Foundation [tiešsaiste] – [atsauce 27.05.2016]. Pieejams: https://www.ethereum.org/foundation
9. White Paper – ethereum/wiki [tiešsaiste] – [atsauce 26.05.2016]. Pieejams: https://github.com/ethereum/wiki/wiki/White-Paper#modified-ghost-implementation
10. What is GHOST and what is its relationship to Frontier and Casper? – Ethereum Stack Exchange [tiešsaiste] – [atsauce 25.05.2016]. Pieejams: http://ethereum.stackexchange.com/questions/314/what-is-ghost-and-what-is-its-relationship-to-frontier-and-casper
11. Accelerating Bitcoin's Transaction Processing [tiešsaiste] – [atsauce 28.05.2016]. Pieejams: http://eprint.iacr.org/2013/881.pdf
12. Toward a 12-second Block Time – Ethereum [tiešsaiste] – [atsauce 23.05.2016]. Pieejams: https://blog.ethereum.org/2014/07/11/toward-a-12-second-block-time/
13. Основатель Ethereum: теперь мы сможем масштабировать систему [tiešsaiste] – [atsauce 19.05.2016]. Pieejams: http://www.coinfox.ru/novosti/5172-ethereum-founder-now-we-can-scale-the-system-2

14. Account types and transactions | Ethereum Frontier Guide [tiešsaiste] – [atsauce 26.05.2016]. Pieejams: https://ethereum.gitbooks.io/frontier-guide/content/account_types.html
15. Основатель Ethereum: теперь мы сможем масштабировать систему [tiešsaiste] – [atsauce 26.05.2016]. Pieejams: http://www.coinfox.ru/novosti/5172-ethereum-founder-now-we-can-scale-the-system-2
16. State Channels – an explanation : ethereum [tiešsaiste] – [atsauce 29.05.2016]. Pieejams: https://www.reddit.com/r/ethereum/comments/3tcu82/state_channels_an_explanation/
17. Bitcoin Lightning решит проблему масштабируемости блокчейна[tiešsaiste] – [atsauce 24.05.2016]. Pieejams: https://bitnovosti.com/2015/04/13/bitcoin-lightning-network/
18. Bitcoin, Litecoin, Namecoin, Dogecoin, Peercoin, Ethereum stats [tiešsaiste] – [atsauce 13.05.2016]. Pieejams: https://bitinfocharts.com
19. State of the Dapps [tiešsaiste] – [atsauce 11.05.2016]. Pieejams: http://dapps.ethercasts.com
20. ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER HOMESTEAD DRAFT [tiešsaiste] – [atsauce 12.05.2016]. Pieejams: http://gavwood.com/Paper.pdf
21. Ethereum State Transition Function [tiešsaiste] – [atsauce 24.05.2016]. Pieejams: https://github.com/ethereum/wiki/wiki/White-Paper#ethereum-state-transition-function
22. Serpent – ethereum/wiki [tiešsaiste] – [atsauce 26.05.2016]. Pieejams: https://github.com/ethereum/wiki/wiki/Serpent
23. GitHub – obscuren/mutan: Compiler & Language definition for the Ethereum project. [tiešsaiste] – [atsauce 23.05.2016]. Pieejams: https://github.com/obscuren/mutan
24. Contract development – LLL language [tiešsaiste] – [atsauce 20.05.2016]. Pieejams: http://ethereum.stackexchange.com/questions/348/is-lll-still-used-as-language
25. Contracts — Ethereum Homestead 0.1 documentation. [tiešsaiste] – [atsauce 28.05.2016]. Pieejams: http://www.ethdocs.org/en/latest/contracts-and-transactions/contracts.html#serpent
26. Dapps — Ethereum Homestead 0.1 documentation. [tiešsaiste] – [atsauce 26.05.2016]. Pieejams: http://www.ethdocs.org/en/latest/contracts-and-transactions/developer-tools.html#dapp-directories

27. Ethereumjs/testrpc: Fast Ethereum RPC client for testing and development. [tiešsaiste] – [atsauce 30.05.2016]. Pieejams: https://github.com/ethereumjs/testrpc
28. ConsenSys/truffle: A development framework for Ethereum. [tiešsaiste] – [atsauce 29.05.2016]. Pieejams: https://github.com/Consensys/truffle
29. Truffle documentation [tiešsaiste] – [atsauce 28.05.2016]. Pieejams: http://truffle.readthedocs.io/en/latest/
30. Nexusdev/dapple: A tool for contract system developers. [tiešsaiste] – [atsauce 28.05.2016]. Pieejams: https://github.com/nexusdev/dapple
31. Populus 1.0.0 documentation. [tiešsaiste] – [atsauce 28.05.2016]. Pieejams: http://populus.readthedocs.io/en/latest/
32. Eris Industries Documentation. [tiešsaiste] – [atsauce 28.05.2016]. Pieejams: https://docs.erisindustries.com/documentation/eris-package-manager/
33. Embark – Framework for Ethereum Dapps [tiešsaiste] – [atsauce 27.05.2016]. Pieejams: https://iurimatias.github.io/embark-framework
34. Solidity — Solidity 0.2.0 documentation. [tiešsaiste] – [atsauce 30.05.2016]. Pieejams: https://solidity.readthedocs.io/en/latest/
35. Ethereum Community Forum – Decentralized 'Dropbox' example [tiešsaiste] – [atsauce 26.05.2016]. Pieejams: https://forum.ethereum.org/discussion/1097/in-the-decentralized-dropbox-example-how-is-trading-a-currency-for-reading-a-file-done
36. c3D Compatibility Documentation [tiešsaiste] – [atsauce 27.05.2016]. Pieejams: https://github.com/project-douglas/eris/blob/master/docs/Structure.md
37. Основы криптографии: Учебное пособие. [tiešsaiste] – [atsauce 09.05.2016]. Pieejams: http://virtua.nsaem.ru:8001/mm/2008/000097884.pdf
38. Ethereum/web3.js: Ethereum Compatible JavaScript API. [tiešsaiste] – [atsauce 30.05.2016]. Pieejams: https://github.com/ethereum/web3.js

Bakalaura darbs „Blockchain tehnoloģijā balstītu sistēmu izveide ar Ethereum” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____ Mihails Ļevenčiks

Rekomendēju/nerekomendēju darbu aizstāvēšanai

Vadītājs: Dr. dat. Guntis Arnicans _____ __.05.2016.

Recenzents: Dr. hab. dat., prof. Audris Kalniņš

Darbs iesniegts Datorikas fakultātē __.05.2016.

Dekāna pilnvarotā persona: vecākā metodiķe Ārija Sproģe _____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

_____. prot. Nr. _____.

Komisijas sekretārs(-e): _____