KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Justas Raudonius

VARTOTOJŲ AUTENTIFIKAVIMAS TINKLO PASLAUGOSE NAUDOJANT BLOKŲ GRANDINES

Baigiamasis magistro darbas

Vadovas
Prof. Algimantas Venčkauskas

KAUNAS, 2019

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

VARTOTOJŲ AUTENTIFIKAVIMAS TINKLO PASLAUGOSE NAUDOJANT BLOKŲ GRANDINES

Baigiamasis magistro darbas

Informacijos ir informacinių technologijų sauga (kodas 621E10003)

Vadovas
(parašas) Prof. Algimantas Venčkauskas
(data)

Recenzentas
(parašas) Doc. Nerijus Morkevičius
(data)

Projektą atliko
(parašas) Justas Raudonius
(data)

KAUNAS, 2019

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Kompiuterių katedra

(Studento vardas, pavardė)

Informacijos ir informacinių technologijų sauga (kodas 621E10003)

(Studijų programos pavadinimas, kodas)

„Vartotojų autentifikavimas tinklo paslaugose naudojant blokų grandines“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ____ d.

Kaunas

Patvirtinu, kad mano, **Justo Raudoniaus**, baigiamasis projektas tema **„Vartotojų autentifikavimas tinklo paslaugose naudojant blokų grandines“** yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

(vardą ir pavardę įrašyti ranka) *(parašas)*

Raudonius, J. „Vartotojų autentifikavimas tinklo paslaugose naudojant blokų grandines“. Magistro baigiamasis projektas / vadovas prof. Algimantas Venčkauskas; Kauno technologijos universitetas, Informatikos fakultetas, Kompiuterių katedra.

Kaunas, 2019. 60 p.

SANTRAUKA

Daugeliui žmonių kiekvieną dieną tenka įrodyti savo tapatybę naudojantis tinklo paslaugomis (jungiantis prie pašto, banko, valstybinių institucijų ir kt. tinklalapių). Yra sukurta daugybė įvairių autentifikacijos būdų, pasižyminčių savitais privalumais ir trūkumais, tačiau saugios autentifikacijos problema nėra iki galo išspręsta. Nuolatos ieškoma naujų saugesnių, patikimesnių autentifikacijos būdų. Per pastaruosius kelis metus labai išpopuliarėjo blokų grandinių technologijos, jos jau yra pritaikomos įvairiose srityse: finansų, draudimo paslaugose, medicinoje, daiktų internete ir kt. Taip pat tiriamas ir blokų grandinių pritaikomumas vartotojų autentifikacijai ir identiteto valdymui apskritai.

Šio baigiamojo magistro darbo tikslas yra sukurti vartotojų autentifikavimo tinklo paslaugose sistemą, naudojančią blokų grandines, atsižvelgiant į kitų autorių darbuose pastebėtus trūkumus ir juos ištaisant. Darbo metu analizuojama literatūra vartotojų autentifikavimo, blokų grandinių veikimo ir praktinio panaudojimo tematika. Pasiūlomas blokų grandinėmis paremtas vartotojų autentifikavimo sistemos teorinis modelis ir jis pritaikomas praktiškai. Sukuriamas „Ethereum“ išmanusis kontraktas, skirtas valdyti vartotojų autentifikacijos informaciją ir juos autentifikuoti. Šis išmanusis kontraktas įdiegiamas į sukurtą blokų grandinės tinklą, kurį vartotojų autentifikacijai naudoja dvi paslaugų svetainės. Administratoriui užregistravus vartotoją vienoje paslaugos svetainėje, vartotojas gali prisijungti ir kitoje, nes blokų grandinė sinchronizuoja šią informaciją. Sukurtas prototipas yra ištiriamas, išmatuojant vartotojų autentifikacijos informacijos tikrinimo trukmės priklausomybę nuo registruotų vartotojų skaičiaus ir palyginamas su dažnai naudojamu autentifikavimu naudojant „MySQL“ duomenų bazę.

Raudonius, Justas. *Blockchain Based User Authentication in Network Services*: *Master's* thesis in Information and Information Technology Security / supervisor prof. Algimantas Venčkauskas. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: Information Technology Security.

Key words: authentication, blockchain, network, „Ethereum", smart contract.

Kaunas, 2019. 60 p.

SUMMARY

Everyday a lot of people are required to prove their identity while using network services (e. g., email, bank, government institutions). Currently, there are a lot of various authentication methods, differing by their strengths and weaknesses. Though, the problem of secure authentication remains. During the recent years, blockchain technology has gain a lot of traction and is being experimented with and applied in various fields, such as, finance, insurance, medicine, Internet of things, etc. Blockchain applicability for user authentication and identity management is also being evaluated.

The goal of this master's thesis is to create a blockchain based user authentication system, taking into account problems observed in related work of other authors. Literature on user authentication, blockchain and its practical applicability is analyzed in this thesis. Based on the analysis, a theoretical model for blockchain based user authentication system is proposed. Theoretical model is then implemented practically, and a prototype is created. Prototype consists of a newly developed "Ethereum" smart contract, used for managing users' authentication data and authentication itself. This smart contract is deployed to "Ethereum" blockchain network, which is used by two websites. When administrator registers a new user in the smart contract once, the user can authenticate to both of the websites even though the websites connect to different blockchain network nodes. User authentication information is stored in the blockchain and therefore synchronized between the network's nodes. Eventually, it is analyzed how the time needed to verify user authentication information in the created smart contract corelates with the number of registered users. The results are compared to other popular way of storing user authentication information – "MySQL" database.

TURINYS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

API – aplikacijų programavimo sąsaja (angl. Application Programming Interface);

Autentifikavimas – procesas, kuriuo yra įrodoma esybės identitetas;

Autentifikavimo serveris – vartotojų autentifikavimo informacijos saugojimo ir pateiktos autentifikavimo informacijos tikrinimo serveris;

Autorizavimas – esybės prieigos prie konkretaus ištekliaus teisių nustatymas;

Blokų grandinė – rikiuotas kriptografiškai sujungtų įrašų sąrašas;

Blokų grandinės tinklas – tarpusavyje sujungtų blokų grandines naudojančių mazgų visuma;

Identifikavimas – esybės identiteto (tapatybės) nustatymas;

Identiteto valdymas – vartotojų identifikavimo, autentifikavimo ir autorizavimo visuma;

Išmanusis kontraktas – blokų grandinės tinkle vykdomas programinis kodas;

Maišos funkcija – vienkrypčio duomenų transformavimo funkcija;

Maišos kodas – maišos funkcijos rezultatas;

Paslaugos serveris – tam tikrą paslaugą teikiantis serveris;

PKI – viešojo rakto infrastruktūra (angl. Public key infrastructure);

Privilegijų serveris – esybių prieigos prie išteklių teisių saugojimo serveris;

SSO – vieningo prisijungimo technologija (angl. Single sign-on);

„Ethereum“ – viena iš blokų grandinių technologijų, palaikančių išmaniuosius kontraktus.

ĮVADAS

„Vartotojų autentifikavimas tinklo paslaugose naudojant blokų grandines“ yra baigiamasis magistro darbas, priklausantis „Informacijos ir informacinių technologijų saugos“ programai. Šiame darbe analizuojamos įvairių autentifikacijos metodų savybės, privalumai ir trūkumai, blokų grandinių technologijos, jų veikimo principai ir tiriamas blokų grandinių pritaikomumas vartotojų autentifikacijai tinklo paslaugose.

Darbo problematika ir aktualumas. Darbo **problematika** sietina su egzistuojančia daugybe įvairių autentifikacijos metodų, besiskiriančių savo savybėmis, saugumu ir panaudojamumu skirtingose situacijose. Vis labiau modernėjančiame pasaulyje, žmonėms tenka nuolatos įrodinėti savo tapatybę, kad gautų prieigą prie tinklo ir kitų paslaugų. Tačiau autentifikacijos saugumo problema vis dar neišspręsta, kiekvienais metais viešumoje pasirodo pranešimų apie milijonus nutekintų slaptažodžių, įsilaužimus į sistemas dėl socialinės inžinerijos, silpnai apsaugotų ar neteisingai pritaikytų autentifikacijos mechanizmų ir kt. Dėl šios priežasties, nuolatos gerinami esami ir kuriami nauji autentifikacijos būdai. Apžvelgiant Lietuvoje rašomus baigiamuosius mokslo darbus, pastebėta, kad jie dažniausiai tyrinėja tik su blokų grandinėmis susijusias problemas[1]. O platesniame, pasauliniame diskurse, tyrėjai aptaria ir blokų grandinių pritaikomumą autentifikacijai (tai plačiau pristatoma teorinėje darbo dalyje). Taigi, šis darbas **aktualus** savo naujumu, nes darbo rezultatas – sukurta vartotojų autentifikacijos tinklo paslaugose sistema, naudojanti blokų grandines. Blokų grandinėmis paremtos autentifikavimo sistemos gali būti diegiamos aukštus saugumo reikalavimus turinčiose sistemose (bankų, valstybinių institucijų ir pan.) Darbe analizuojamas blokų grandinių pritaikomumas autentifikacijai, siūlomas teorinis modelis, kuris yra įgyvendinamas sukuriant vartotojų autentifikacijos sistemą, naudojančią „Ethereum“ blokų grandinių tinklą ir išmaniuosius kontraktus, ir pritaikant ją interneto svetainėje.

Darbo tikslas – sukurti blokų grandinėmis paremtą vartotojų autentifikacijos tinklo paslaugose sistemą.

Šiam tikslui pasiekti, iškelti penki **uždaviniai**:

1. Susipažinti ir išanalizuoti identiteto valdymo problematiką ir detaliau panagrinėti autentifikacijos metodus ir su jais susijusias problemas.
2. Įsigilinti į blokų grandinių technologijų veikimo principus, šiuo metu egzistuojančius sprendimus, pritaikant šias technologijas autentifikacijai; identifikuoti esamų sprendimų privalumus ir trūkumus.
3. Sukurti teorinį vartotojų autentifikavimo modelį, vartotojų autentifikacijai naudojantį blokų grandinę.
4. Teorinį modelį praktiškai pritaikyti sukuriant interneto svetainę; kuriant atsižvelgti ir patobulinti teorinėje darbo dalyje pastebėtų kitų darbų autorių autentifikavimo sprendimų trūkumus.
5. Ištirti sukurtą teorinio modelio prototipą: išmatuoti vartotojų autentifikacijos spartą ir palyginti su vienu iš tradicinių sprendimų.

Darbo rezultatas – sukurta vartotojų autentifikacijos sistema, kurios veikimas grįstas blokų grandinių tinklais. Darbe pasirinktas blokų grandinių pritaikymas autentifikacijai užtikrina aukštesnį vartotojų informacijos saugumą.

[1] Bakalauro darbai: Martynas Andriuškevičius „Blokų naršymo įrankis Ethereum platformai“ (Kauno technologijos universitetas, 2018), Tomas Labanauskas „NullVote anonimio balsavimo sistema“ (Kauno technologijos universitetas, 2018); magistro darbai: Aleksas Šulnius „BlockChain sistemų mastelio keitimo savybių tyrimas taikant agentais grįstą modeliavimą“ (Vilniaus universitetas, 2017) ir kt.

Darbo struktūra – darbą sudaro trys dalys. Pirmame skyriuje „Identiteto valdymo ir blokų grandinių technologijų analizė“ analizuojama su autentifikacija ir blokų grandinėmis susijusi literatūra. Antrame skyriuje „Blokų grandinės panaudojimo autentifikacijai modelis“ pristatomas sukurtas vartotojų autentifikacijos modelis, pateikiami galimi šio modelio panaudos atvejai ir veiklos diagramos. Trečiame skyriuje „Ethereum“ išmaniojo kontrakto sukūrimas ir pritaikymas vartotojų autentifikacijai“ pristatoma, kaip antrajame skyriuje pasiūlytas teorinis modelis yra įgyvendinamas praktiškai, naudojant konkrečias blokų grandinių tinklų kūrimo ir interneto svetainių programavimo technologijas, taip pat, sukurtas prototipas yra tiriamas, įvertinant vartotojų autentifikacijos spartą priklausomai nuo sistemoje registruotų vartotojų skaičiaus.

1. IDENTITETO VALDYMO IR BLOKŲ GRANDINIŲ TECHNOLOGIJŲ ANALIZĖ

1.1. Identiteto valdymas

Pasaulis labai sparčiai modernėja ir vis daugiau paslaugų yra perkeliama į elektroninę erdvę arba joje yra kuriamos naujos paslaugos. Taigi vis dažniau tenka įrodyti ir savo tapatybę, tam kad gautume prieigą prie šių paslaugų. Užtikrinant sistemų saugumą neapsieinama be vartotojų identifikavimo, identiteto įrodymo (autentifikavimo) ir jų prieigos prie išteklių valdymo (autorizavimo). Visi šie saugumo užtikrinimo etapai kartu yra vadinami identiteto valdymu. Šio skyriaus įžangoje aptariamos identiteto valdymo sudedamosios dalys, kurių dažniausiai išskiriamos trys: identifikavimas, autentifikavimas ir autorizavimas [1, 2, 3, 4] ir kartais išskiriama ketvirta – apskaita [5, 6].

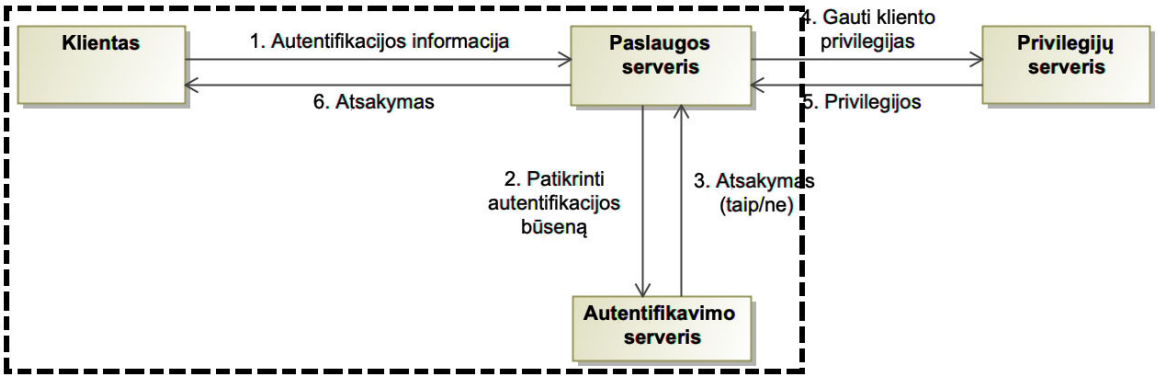
Pirmasis etapas – identifikavimas – skirtas atpažinti vartotoją. Tai gali būti įgyvendinama pateikiant vartotojo vardą, e. paštą, asmens kodą ar kitą, leidžiančią identifikuoti, informaciją [1, 7]. Vien tik identifikavimas neįrodo, kad žmogus yra tuo, kuo teigia esąs [5] ir tik identifikavimo užtektų patikimai suteikti prieigą prie paslaugų, tik tuo atveju, jei vartotojais būtų galima pasitikėti ir jie nebūtų linkę apsimesti tuo, kuo nėra.

Norint užtikrinti aukštesnio lygio saugumą, reikalingas antrasis etapas – autentifikavimas – kuriuo vartotojas įrodo savo tapatybę [1, 2, 5, 7]. Tapatybei įrodyti yra sukurta įvairių būdų, pavyzdžiui, slaptažodis, PIN kodas, elektroninė kortelė [8], mobilusis parašas [9] ar biometriniai [7, 10], kriptografiniai metodai [11]. Priklausomai nuo verslo pobūdžio, norimo saugumo lygio ir kitų poreikių, parenkamas atitinkamas autentifikacijos būdas. Šiuo procesu siekiama užtikrinti, kad prieiga prie sistemos bus suteikta tik konkrečiai esybei ir vienos esybės negalės apsimesti kitomis.

Vartotojui autentifikavusis, paslaugos tiekėjas suteikia prieigą prie paslaugos ir, jeigu reikia, papildomai pritaiko trečiąjį etapą – autorizaciją – kurios metu patikrinamos konkretaus vartotojo prieigos prie reikiamo ištekliaus teisės [1, 5, 7]. Pritaikant visus tris etapus yra užtikrinama, kad prieiga prie išteklių būtų suteikiama tik autorizuotiems vartotojams.

Yra sukurta daugybė įvairių autentifikacijos metodų, kurie pasižymi skirtingomis charakteristikomis, saugos lygiu, pritaikomumu. Klientas, norėdamas gauti prieigą prie serverio teikiamų paslaugų, privalo autentifikuotis, kad įrodytų savo tapatybę. 1 paveiksle vaizduojama tipinė vartotojų autentifikacijos ir autorizacijos schema. Šiame darbe koncentruojamasi į schemoje punktyru apibrauktą dalį – autentifikacijos etapą, o privilegijų serveris įtraukiamas ir vaizduojamas dėl papildomo konteksto. 1 paveiksle vaizduojamoje schemoje, autentifikacija apima tris sistemos dalyvius: klientą, kuris siekia gauti prieigą prie paslaugos; paslaugos serverį, kuris teikia paslaugą ir autentifikavimo serverį, kuriame saugomas prieigą turinčių klientų sąrašas.

Pirmiausia, klientas pateikia autentifikacijos informaciją serveriui. Priklausomai nuo autentifikacijos metodo, tai gali būti vartotojo identifikatorius ir slaptažodis, žetonas, biometrinės kliento savybės atvaizdas ar kt. Paslaugos serveris, gavęs autentifikacijos informaciją, ją patikrina autentifikacijos serveryje. Jeigu to reikalauja sistema, serveris gauna klientui priklausančias privilegijas iš privilegijų serverio ir galiausiai grąžina atsakymą klientui apie tai, ar autentifikacija buvo sėkminga ir pateikia kitą reikiamą informaciją.



1 pav. Tipinė vartotojų autentifikacijos ir autorizacijos schema[2]

Autentifikacijos metodai dažnai yra skirstomi pagal tai, kaip vartotojai juos saugo ir kaip jais naudojasi. Algimantas Venčkauskas ir Jevgenijus Toldinas [7], Lawrence'as O'Gorman'as [8], Laurie A. Jones'as ir kt. [12], Dobromir'as Todorov'as [5] išskiria šiuos identiteto įrodymo būdus:

1. tuo, ką žino – slaptažodžiu, PIN kodu, iš anksto nustatytu atsakymu į saugumo klausimą;
2. tuo, ką turi – banko kortele, aparatine ar programine įranga, suteikiančią prieigą, telefonu;
3. tuo, kas jie yra – šis būdas remiasi žmogaus fizinėmis savybėmis: išvaizda ir kūno geometrija, judesiais, širdies plakimo ypatumais, balsu, rašysena, rašymo klaviatūra charakteristikomis.

Tuo tarpu, Mich'as E. Kabay'us [1] ir Ravi's Sandhu's ir kt. [3] trečiąjį būdą išskiria į du: statinius ir dinaminius. Prie statinių biometrinių savybių priskiria pirštų antspaudus, akies rainelę ir tinklainę, o prie dinaminių – balsą, rašyseną, teksto rinkimo kompiuteriu dinamiką.

Šie autentifikacijos būdai gali būti naudojami tiek pavieniui, tiek kartu. Pagal verslo pobūdį ir jo keliamus saugumo reikalavimus parenkama, kokie būdai turėtų būti naudojami naujai kuriamose sistemose. Taigi yra išskiriami keli saugumo tipai pagal tai, kiek būdų savo tapatybei įrodyti naudoja vartotojai [5]:

1. Vieno lygio – įrodyti tapatybei naudojamas vienas iš trijų anksčiau įvardytų būdų.
2. Dviejų lygių – autentifikacijai naudojami du iš trijų anksčiau įvardytų būdų. Pavyzdžiui, prie sistemos prisijungti užtenka vartotojo vardo ir slaptažodžio, o norint įvykdyti aukštesnio saugumo lygio reikalaujančius veiksmus, reikia autentifikuotis dar vienu būdu: gauti žinutę į telefoną, naudoti kodų generatorių ar kt.
3. Kelių lygių – šis tipas yra toks pat kaip dviejų lygių, tačiau naudojama daugiau nei du lygiai.

Kaip jau minėta, autentifikacijos būdai pasižymi skirtingomis savybėmis, todėl skiriasi jų pritaikomumas. Kiekvienu atveju situacija turėtų būti įvertinta ir parinktas tinkamiausias būdas ar jų kombinacija. Toliau šiame skyriuje detaliau apžvelgiami autentifikacijos būdai, įvardijami jų privalumai ir trūkumai, į kuriuos atsižvelgti būtina kuriant naują saugumo sistemą.

Autentifikacija vartotojų identifikatoriumi ir slaptažodžiu

Seniausias ir populiariausias būdas autentifikuotis yra naudojant vartotojo identifikatorių (pavyzdžiui, slapyvardį, e. pašto adresą, asmens kodą ir pan.) ir slaptažodį [13]. Norėdamas prisijungti

[2] Schema nupiešta darbo autoriaus remiantis [1, 4].

klientas paslaugos serveriui siunčia identifikatorių ir slaptažodį, serveris patikrina, ar turi šiuos duomenis atitinkantį įrašą duomenų bazėje ar kitoje autentifikacijos informacijos saugykloje ir grąžina atitinkamą atsakymą, t. y., ar vartotojui pavyko prisijungti, ar ne.

Šį autentifikacijos metodą įgyvendinti technologiškai nesunku ir tai yra vienas dažniausiai pasitaikančių ir lengviausiai sukuriamų autentifikacijos būdų [14]. Dažniausiai jungiantis prie tinklo paslaugų, serveris pateikia prisijungimo formą su dviem laukeliais: kliento identifikatoriaus ir slaptažodžio. Vartotojui tereikia juos užpildyti ir pateikti formą.

Paslaugos serveris slaptažodžius gali saugoti įvairiais būdais: lokaliame faile [15], duomenų bazėje [16], LDAP [17] serveryje ir kt. Vienas dažniausių būdų – saugoti slaptažodžius ar jų atitikmenis duomenų bazėje. Pavyzdžiui, naudojama lentelė „vartotojai“, kurioje saugomi klientų prisijungimo vardai, eilės numeriai, sukūrimo data, slaptažodžiai ir kita informacija. Pats paprasčiausias slaptažodžio saugojimo formatas – atviras tekstas. Vartotojo registracijos metu pateiktas slaptažodis įrašomas į duomenų bazę, o jungiantis prie paslaugos, pateiktas slaptažodis patikrinamas su išsaugotu duomenų bazėje ir jeigu jie sutampa, prieiga suteikiama. Tačiau saugoti slaptažodžius atviru tekstu yra pavojinga, nes įsilaužimo atveju, piktavaliai gali pavogti visus prisijungimui prie paslaugos reikalingus duomenis.

Norint užtikrinti aukštesnį saugumo lygį, slaptažodžiai neturėtų būti saugomi atviruoju tekstu. Prieš išsaugant duomenų bazėje ar kitoje saugykloje, turėtų būti pritaikyta vienkryptė transformacija, tokiai pačiai pradinei reikšmei visados grąžinanti tokį patį rezultatą, pavyzdžiui maišos funkcija [18]. Maišos funkcijos ir maišos kodai plačiau aptariami 1. 2. 1 poskyryje. Kai duomenų bazėje saugomi slaptažodžių maišos funkcijų, o ne patys slaptažodžiai, įsilaužimo atveju slaptažodžiai nėra atskleidžiami. Norint atkurti slaptažodį iš jo maišos kodo, reikia pasitelkti grubią jėgą (angl. *brute force*), o tai užima daug laiko ir reikalauja daug skaičiavimo galios.

Nesaugus slaptažodžių tvarkymas ir saugojimas gali turėti skaudžių pasekmių, nes žmonės naudojasi daugybe internetinių paslaugų, kuriose reikia registruotis, todėl dažnai naudoja paprastus (silpnus) arba vienodus slaptažodžius [19]. Jeigu yra pavagiamas ir atkuriamas slaptažodis iš vienos paslaugos ir tokie patys prisijungimo duomenys naudojami kitose paslaugose, piktavaliai gali gauti prieigą ir prie jų.

Pagal „Yubico“ užsakytą tyrimą [20], 51 % apklaustųjų teigė naudojantys tokius pačius slaptažodžius skirtingose paskyrose, o vidutiniškai kelis kartus panaudoja 5 slaptažodžius. Virdžinijos technologijų universiteto tyrėjai išanalizavo [21] 28,8 mln. vartotojų 61,5 mln. pavogtų slaptažodžių ir nustatė, kad 38 % žmonių tą patį slaptažodį naudoja daugiau nei vieną kartą.

Galima apibendrinti, kad slaptažodžiai yra patogus būdas jungtis prie interneto paslaugų tiek vartotojams, tiek paslaugų tiekėjams, tačiau silpni slaptažodžiai, slaptažodžių pakartotinis naudojimas, grubios atakos, neteisingas slaptažodžių tvarkymas ir saugojimas kelia didelę riziką.

Išmaniosios kortelės

Pagal Keyth'ą Mayes'ą išmaniosios kortelės yra tos kortelės, kurios turi unikalų identifikatorių, gali būti naudojamos vykdyti elektronines transakcijas (užtikrina kortelėje esančių duomenų saugumą), yra skirtos padidinti sistemos saugumą (palaiko kriptografinius algoritmus ir funkcijas) ir nėra lengvai suklastojamos [22]. Dažniausiai tai būna plastikinės kortelės su magnetine juosta ar mikroprocesoriumi. Vartotojui norint autentifikuotis neužtenka turėti išmaniąją kortelę, tačiau reikia ir įrenginio, kuris gali ją nuskaityti. Dažniausiai tokie įrenginiai įrengiami prie įėjimų į aukšto saugumo zonas. Išmaniąsias korteles naudojant autentifikuotis tinklo paslaugose, galima naudoti kompiuterio klaviatūros priedėlį, gebantį skaityti šias korteles. Vartotojas, norėdamas autentifikuotis sistemoje,

įdeda kortelę į kortelių skaitytuvą ir suveda kortelės PIN kodą. Sistema patikrina PIN kodą ir kortelės galiojimą ir atitinkamai vartotojui suteikia prieigą.

Išmaniosios kortelės pasižymi aukštu saugumo lygiu, nes kortelių neįmanoma pavogti internetu, o net pavogus fiziškai, kortele nepavyks pasinaudoti nežinant jos PIN kodo. Visgi, išmaniąją kortelę visur nešiotis su savimi yra nepatogu, taip pat, naudojant ją namuose, reikia turėti papildomą techninę ir programinę įrangą.

Mobilusis parašas

Dauguma pasaulio gyventojų turi prieigą prie telefonų [23], todėl kuriami jais paremti autentifikacijos metodai. Vienas iš jų – mobilusis parašas. Norėdamas naudotis mobiliuoju parašu vartotojas privalo turėti jį palaikantį mobilųjį telefoną ir pritaikytą SIM kortelę. Vartotojas, jungdamasis prie tam tikros paslaugos, jos puslapyje suveda savo mobilaus telefono numerį ir į telefoną gauna prisijungimo patvirtinimo užklausa. Suvedęs 4–8 skaitmenų ilgio sPIN1 kodą vartotojas prisijungia prie paslaugos, o suvedant 5–8 skaitmenų ilgio sPIN2 kodą, pasirašomos ir patvirtinamos operacijos [24].

Kaip ir išmaniosios kortelės, mobilusis parašas pasižymi aukštu saugumo lygiu, nes jo neįmanoma pavogti internetu. Norint piktavališkai pasinaudoti kito vartotojo mobiliuoju parašu, reiktų gauti prieiga prie šio vartotojo telefono, taip pat žinoti vieną ar abu slaptus sPIN kodus.

Autentifikacija biometrija

Kiekvienas žmogus pasižymi daugybe unikalių savybių, pavyzdžiui, balso tonu, pirštų ir delnų antspaudais, kūno geometrija, akie rainele ir kt. Būtent šios unikalios savybės yra išnaudojamos autentifikuoti sistemų vartotojus [10]. Pagal Gintarą Skersį [25], biometrinės savybės, pagal kurias autentifikuojami vartotojai, turėtų pasižymėti universalumu, unikalumu ir pastovumu. Tai yra, šias savybes turėtų turėti visi žmonės, jos turėtų būti unikalios tarp žmonių ir nekintančios.

Pirmiausia reikiami vartotojo biometriniai duomenys yra užregistruojami sistemoje. Tai vyksta nuskaitant konkrečią vartotojo biometrinę savybę ir iš jos sudarant biometrinį modelį. Šis modelis ir yra saugomas duomenų bazėje ir naudojamas autentifikuoti vartotoją. Kai vartotojas jungiasi prie sistemos, ta pati jo biometrinė savybė yra nuskaitoma ir sudaromas biometrinis modelis, kuris yra palyginamas su registracijos metu sudarytu modeliu. Jeigu modeliai sutampa atsižvelgiant į leistiną paklaidą, vartotojas laikomas autentifikuotu.

Biometrija pasižymi aukštu saugumo lygiu, nes atitinka apibrėžimą „tai, kuo esi“, t. y., teoriškai tik pats vartotojas gali pateikti savo biometrinį atvaizdą ir gauti prieigą prie sistemos savo vardu. Visgi, absoliučiu tikslumu nustatyti ar biometrinės savybės modeliai sutampa yra neįmanoma, todėl įvedamos leistinos modelių nesutapimo paklaidos. Piktavaliai, pasinaudoję šiomis paklaidomis, gali gauti neteisėtą prieigą prie sistemos, todėl leistinos paklaidos turi būti labai mažos. Kita vertus, sumažinus leistinas paklaidas, padidėja klaidingo įgalioto asmens atmetimo dažnumas, t. y., įgalioti vartotojai negali prisijungti prie paslaugos [25]. Sistemose diegiant biometrinį autentifikacijos metodą stengiamasi parinkti optimalią leistiną paklaidą. Apskritai, tinkamai įgyvendinti biometriniai autentifikacijos metodai siūlo aukštą saugumo lygį, tačiau tam tikrais atvejais yra nepraktiški, nes reikalauja sudėtingos įrangos.

Vieningo prisijungimo technologija

Vartotojams naudojant daug skirtingų sistemų, pavyzdžiui, naudojamų tai pačiai paslaugai, kaskart suvedinėti savo prisijungimo duomenis yra problematiška: nepatogu; kai reikia turėti daug

slaptažodžių, žmonės dažnai juos kuria lengvesnius ir, tuo pačiu, mažiau saugius; jautrūs prisijungimo duomenys dažniau siunčiami tinklu, taip pat, juos apdoroja daugiau tarnybų, taip padidėjant rizikai slaptažodžiams būti pavogtiems. Šias problemas sprendžia vieningo prisijungimo technologija (angl., *Single sign-on, SSO*).

Naudojant SSO vartotojas vieną kartą prisijungia prie centrinės autentifikavimo tarnybos, kuri autentifikuoja ir autorizuoja vartotoją kitoms sistemoms. Kai vartotojas per sesijos laikotarpį jungiasi prie antros, trečios ir t. t. paslaugos, kuri naudoja šią tarnybą, prisijungimas įvykdomas automatiškai, naudojant jau užmegztą sesiją [26].

Vieningo prisijungimo sistemos gali veikti dviem būdais: programomis (angl. *script-based*) arba agentais (angl. *agent-based*). Autentifikuojantis su SSO, pagrįstu programomis, vartotojui jungiantis į sistemą, iškviečiama prisijungimo programa, kuri patikrina pateiktus duomenis ir autentifikuoja. Jungiantis prie kitų to paties tinklo sistemų, ta pati programa autentifikuoja automatiškai, prisijungimo duomenų vesti nebereikia. Toks prisijungimo būdas yra nesaugus, nes dažnai slaptažodžiai saugomi arba perduodami tinkle atviru tekstu. Piktavaliai pasinaudoję tinklo šnipinėjimo įranga galia pavogti atviru tekstu perduodamus slaptažodžius ir gauti neteisėtą prieigą [27].

Agentų naudojimas SSO sistemoje yra daug dažnesnis negu aptartas programų būdas dėl savo saugumo. Visos tinklo sistemos, prie kurių leidžiama prisijungti, turi agentus – papildomą programinę įrangą, skirtą autentifikacijai su serveriu. Kai vartotojas jungiasi prie sistemos, agentas įvertina užklausą ir kreipiasi į autentifikacijos serverį, kad patikrintų pateiktų duomenų teisingumą. SSO agentais paremtai autentifikacijai sukurta daugybė protokolų. Vieni populiaresnių: „Kerberos“, „RADIUS“ [27], „Keycloak“ [28], „OneLogin“ [29], SAML [30].

Sandeep’as Sandhu’s [31], Ece’as Cakir’as [26] išskiria šiuos, pagrindinius, vieningo prisijungimo technologijos privalumus:

1. užtenka vieno slaptažodžio – didesnė sauga, vartotojams nereikia turėti daug slaptažodžių, todėl gali sukurti ir įsiminti vieną sudėtingą slaptažodį, mažesni aptarnavimo kaštai;
2. patogumas vartotojams – nereikia daug kartų pateikinėti prisijungimo duomenų;
3. patogumas administratoriams – vartotojas ir susijusius išteklius galima valdyti vienoje vietoje;
4. perkelia jautrių prisijungimo duomenų valdymą iš daugybės sistemų į vieną vietą – teisingai įgyvendinus SSO padidinamas bendras sistemos saugumas;
5. kiekvienai naujai sistemai nereikia kurti viso autentifikacijos funkcionalumo iš naujo, tereikia sukurti prisijungimą per SSO serverį.

Tuo tarpu, Ecer’as Cakir’as [26] pastebi tokius SSO trūkumus:

1. netinkamai įgyvendinta SSO sistema didina saugumo rizikas;
2. sudėtingiau užtikrinti visos sistemos, naudojančios SSO, prieinamumą, nes reikia daugiau laiko rasti klaidas sudėtingoje sistemoje, apimančioje daug posistemių;
3. pavogus prisijungimo duomenis didesnė žala, nes galima prieiti prie daugiau paslaugų negu nenaudojant SSO.

Vieningo prisijungimo technologija suteikia daug privalumų ją naudojant didelėse sistemose su daug posistemių, prie kurių jungiasi tie patys vartotojai. Tokiu atveju žmonėms nereikia kurti ir prisiminti daug skirtingų slaptažodžių, taip pat, yra patogiau suvesti slaptažodį vieną kartą ir gauti prieigą prie visų posistemių. Siekiant užtikrinti sistemos, naudojančios SSO, saugą, administratoriai privalo tinkamai pritaikyti SSO protokolą (-us). Tinkamai naudojama vieningo prisijungimo technologija palengvina vartotojų prisijungimą ir jų duomenų administravimą.

„LDAP“ protokolas

„LDAP“ (angl. *Lightweight Directory Access Protocol*) yra atviras protokolas, skirtas komunikuoti su direktorijos paslaugas teikiančia tarnyba naudojant „TCP/IP“ protokolą. „LDAP“ protokolas gali būti naudojamas skaityti ir rašyti informaciją į direktoriją, autentifikuoti vartotojus ir kt. [32]

„LDAP“ 3 versija palaiko tris autentifikacijos tipus: anoniminę, paprastąją ir paremtą „SASL“ (angl. *Simple Authentication and Secure Layer*). Klientai kreipdamiesi į „LDAP“ serverį ir nepateikiantys autentifikacijos duomenų yra laikomi anonimais ir jiems suteikiamos mažiausios privilegijos. Paprastoji autentifikacija atliekama į „LDAP“ serverį siunčiant kliento identifikatorių ir slaptažodį, serveris patikrina, ar pateiktas slaptažodis sutampa su išsaugotuoju direktorijoje ir grąžina atitinkamą atsakymą. Šie prisijungimo duomenys tinklu yra siunčiami atviru tekstu, todėl patartina naudoti srauto šifravimą [32]. Norint naudoti „LDAP“ protokolą autentifikacijai, reikia turėti ir palaikyti „LDAP“ direktorijos serverį, taip pat, šis protokolas autentifikacijai gali būti naudojamas tik su tais klientais, kurie palaiko „LDAP“ protokolą.

„Kerberos“ protokolas

„Kerberos“ yra tinklo autentifikacijos protokolas, naudojantis kriptografinius raktus ir šifravimą. Tarp kliento ir serverio saugus ryšys užmezgamas net ir nešifruotuose kanaluose naudojant stiprią kriptografiją, klientas įrodo savo tapatybę serveriui ir atvirkščiai prieš pradedant apsikeitimą duomenimis. Po tapatybės įrodymo visa komunikacija gali būti šifruojama taip užtikrinant duomenų privatumą ir vientisumą [33].

„Kerberos“ naudoja autentifikavimo bilietus (specialaus formato kriptografines žinutes), kuriuos klientas naudoja įrodyti savo tapatybę skirtingose paslaugose. Taip išvengiama pakartotinio slaptažodžio siuntimo tinklu, taip pat ir slaptažodžio siuntimo atviru tekstu, pagerinamas vartotojų potyris. „Kerberos“ serveris tinkle naudojamas kaip patikima centrinė autentifikacijos vieta, visos autentifikacijos užklausos siunčiamos per šį serverį. Papildomai saugumui padidinti naudojama abipusė autentifikacija – kai ne tik klientas įrodo tapatybę serveriui, tačiau ir serveris klientui. Taip užtikrinama, kad vartotojų užklausos piktavališkai nebūtų nukreipiamos į kitą „Kerberos“ serverį ir nebūtų bandoma pavogti prisijungimo duomenų [33].

Vartotojui jungiantis prie paslaugos, autentifikacijai naudojančios „Kerberos“, raktų valdymo serveris (angl. *Key Distribution Center*, KDC) sukuria bilietą ir jį užšifruoja vartotojo slaptažodžiu. Vartotojas, iššifravęs bilietą, gali jį naudoti iš bilietų tiekimo serverio (angl. *Ticket Granting Server*) gauti kitus bilietus, skirtus gauti prieigą prie konkrečių išteklių. Šiuos bilietus vartotojas siunčia jungdamasis prie konkrečios paslaugos, kurios serveris patikrina, ar bilietas galiojantis ir atitinkamai suteikia prieigą arba ne [33].

Viešojo rakto infrastruktūra

Viešojo rakto infrastruktūra (angl. *public key infrastructure*, PKI) yra procedūrų rinkinys, skirtas tvarkyti kriptografinius raktus ir kurti, dalintis, naudoti, saugoti ir atšaukti skaitmeninius sertifikatus. Šios procedūros naudojamos užtikrinti saugų informacijos perdavimą elektroninėje erdvėje, užtikrinant, kad informaciją galės perskaityti tik tikrieji gavėjai, ji nebus pakeista ar suklastota [34].

PKI susideda iš kelių dalyvių: klientų, kurie keičiasi informacija, sertifikatų registravimo (angl. *registration* authority, RA), valdymo (angl. *certificate authority*, CA) ir patvirtinimo (angl. *validation authority*, VA) institucijų. Klientas, norėdamas dalyvauti PKI, privalo susigeneruoti kriptografinių raktų porą, kuri atitiks jo tapatybę. Privatusis kriptografinis raktas naudojamas pasirašyti žinutes, taip

įrodant konkrečių raktų nuosavybę ir pseudo-tapatybę. Tačiau tai dar neįrodo tikrosios žmogaus tapatybės, t. y., koks jo vardas, pavardė, asmens kodas. Tapatybės ir viešojo kriptografinio rakto susiejimą vykdo registravimo institucija, kuriai žmogus pateikia identifikuojantį dokumentą (pavyzdžiui, pasą, tapatybės kortelę, banko patvirtinimą ar kitą sutartą galiojantį įrodymą) ir savo viešąjį raktą, o RA suformuoja elektroninį sertifikatą, turintį laiko žymą, žmogaus identifikatorių ir viešąjį raktą, ir pasirašo savo privačiuoju raktu. Sertifikatą persiunčia į CA ir VA, kuris, reikalui esant, patvirtina, kad konkretus viešasis raktas priklauso konkrečiam asmeniui [34].

Viešojo rakto infrastruktūra įgalina patogiai įrodyti savo tapatybę elektroninėje erdvėje. Pastebėtina, kad viešojo rakto infrastruktūrą gali naudoti ne tik žmonės tarpusavyje, bet ir su kompiuteriais arba kompiuteriai tarpusavyje. Pavyzdžiui, PKI sertifikatai plačiai naudojami žiniatinklyje siekiant užtikrinti, kad informacijos (pavyzdžiui, interneto svetainės puslapio) perdavimo metu, ji nebuvo pakeista ar suklastota ir kad ją perdavė būtent ta svetainė, į kurią žmogus kreipėsi [35, 36]. Viešojo rakto infrastruktūra suteikia aukštą saugumo lygį, tačiau reikalauja kriptografinių raktų valdymo vartotojo įrenginyje.

Centralizuoto autentifikavimo sistemos

Šiame poskyryje aptarti autentifikacijos būdai tradiciškai naudojami jungtis prie centralizuotų paslaugų. Tai yra, autentifikacijos serverį ar paslaugą, kurioje įdiegtas autentifikacijos mechanizmas, valdo tam tikra esybė (pavyzdžiui, konkretus asmuo, įmonė, įmonių grupė, valstybė ir pan.) Visų paslaugos vartotojų prisijungimo duomenys saugomi vienoje ar keliose duomenų bazėse, prie kurių priėjimą turi tik tam tikra esybė ir tik ji gali reguliuoti, kas gali registruotis sistemoje, kuriems registruotiems vartotojams suteikti prieigą ir pan. Iš to kyla keletas problemų:

- Įsilaužimo į sistemą atveju, padaroma didelė žala, nes piktavaliai gauna prieigą prie visų saugomų autentifikacijos duomenų. Pavyzdžiui, dažna problema yra įsilaužimai į sistemas, kurios prisijungimui naudoja slaptažodžius. Kiekvienais metais milijonai [37, 38, 39, 40, 41, 42] slaptažodžių ir kitos vartotojų informacijos yra pavagiama per įsilaužimus į įvairias sistemas.
- Kai vartotojų autentifikacija vykdoma centralizuotai ir nenaudojami arba naudojama per mažai atsarginių autentifikacijos serverių, atakos prieš autentifikacijos tarnybą ar serverių sutrikimai nebeleidžia vartotojams prisijungti prie sistemos [43].
- Informacija, kuri valdoma tam tikros vienos esybės, gali būti cenzūruojama. Politinių, ekonominių ar kitokių nesutarimų atveju, sistemos savininkas gali nebeleisti prisijungti tam tikriems vartotojams, tai gali būti žalinga žodžio ir kitoms laisvėms [44].

1.2. Blokų grandinių technologija

1991 metais Stuart'as Haber'is ir W. Scott'as Stornetta [45] pasiūlė idėją, kaip būtų galima užtikrinti elektroninių išteklių vientisumo ir chronologinės sekos apskaitą ir įrodymą. Autoriai iškėlė problemą, kad kitaip nei su fiziniais ištekliais, aptikti elektroninių išteklių klastotes ar pakeitimus yra sudėtinga. Tiesa, dabar, kai technologijos yra gerokai pažengusios lyginant su publikacijos metais, ir egzistuoja aukšto lygio tyrimo metodai, tirti elektroninius išteklius taip pat lengviau.

S. Haber'is ir W. S. Stornetta aprašė laiko ženklinimo metodą remdamiesi „skaitmeninio seifo" (angl. *digital safety-deposit box*) koncepcija: ištekliai siunčiami į centrinę tarnybą, kurioje įrašomas gavimo laikas ir data bei išsaugomas pats dokumentas. Tai iš esmės atitinka pagrindinę sprendžiamos problemos sąlygą, tačiau turi kitų trūkumų. Visų pirma, pažeidžiamas išteklių autorių privatumas – perdavimo arba saugojimo metu piktavaliai gali perimti dokumentą ir sužinoti jo turinį. Pilnų išteklių perdavimas ir saugojimas reikalauja daug laiko, interneto srauto ir saugojimo vietos. Perduodant ar saugant dokumentus jie gali būti sugadinti ir visas laiko ženklinimo procesas prarastų prasmę, nes nebebūtų įmanoma gauti informacijos apie dokumentą – kada jis buvo paženklintas. Be to, tokia sistema yra centralizuota ir tinklo dalyviai turi pasitikėti laiko ženklinimo tarnyba, kuri, jeigu bus kompromituota, gali falsifikuoti žymes, jas kurti praeities ar ateities datoms. Kad būtų sprendžiami šie trūkumai, autoriai pasiūlė du patobulinimus: naudoti maišos funkcijas ir elektroninius parašus. Vietoje to, kad laiko ženklinimo tarnyba gautų ir saugotų pilnus dokumentus, naudojami tų failų maišos funkcijų rezultatai, perduodamų ir saugomų duomenų kiekis sumažėja daug kartų, nes bet kokio dydžio dokumentui suskaičiuojama fiksuoto ilgio (pvz., 256 bitų) reikšmė. Kad būtų įsitikinta, kad reikšmės perdavimo metu iš kliento į tarnybą, ji nebuvo pakeista arba sugadinta, tarnyba paženklina dokumentą, jį pasirašo ir siunčia atgal klientui, kuris patikrina parašo galiojimą ir pasirašytus duomenis. Tačiau liko dar viena problema: tarnyba vis tiek gali falsifikuoti dokumentų žymių datas. Publikacijos autoriai pasiūlė, kad žymės privalo turėti informaciją apie praeitas žymes [45]. Tokiu būdu kuriant žymes, sudaroma grandinė, kurią norint pakeisti, reikia pakeisti visas žymes, esančias po keičiamosios. Pavyzdžiui, jeigu grandinę sudaro n žymių ir norima pakeisti arba įterpti naują po k žymos, tektų perskaičiuoti $n - k$ žymių reikšmes. Norint tą padaryti, reiktų daugiau skaičiuojamosios galios negu tuo metu yra sutelkta originalios grandinės palaikymui. Tinklo dalyviai naująją (suklastotą) grandinę pripažintų tikrąja tik tada, kai ji taptų ilgesne už tikrąją. Bet kuriai nors kiek labiau naudojamai grandinei (pvz., populiariausių kriptovaliutų) tai yra nepraktiška arba neįmanoma [46][47].

Vėliau, 1992 metais, Dave'as Bayer'is, Stuart'as Haber'is ir W. Scott'as Stornetta [48] pasiūlė grandinės metodo patobulinimą, kuriuo apskaitomi ištekliai į grandinę jungiami ne pavieniui, o blokais. Tai leido gerokai sumažinti duomenų srautus ir reikiamos saugojimo vietos kiekį. Sujungus įrašus į blokus, technologija pavadinta blokų grandine (angl. *block chain*, vėliau patapo *blockchain*).

1.2.1. Blokų grandinių veikimo principai

Terminas „blokų grandinė" (angl. *blockchain*) neretai vis dar yra maišomas su „Bitcoin", ypač mažiau su šiomis technologijomis susiduriančių žmonių, kadangi išpopuliarėjo tuo pačiu metu [48]. 2008 metais programuotojas ar jų grupė prisidengę pseudonimu Satoshi'is Nakamoto'a, aprašė virtualios valiutos koncepciją, paremtą blokų grandinės technologija [47], o 2009 ją paviešino išleisdamas(-i) pirmąją kriptovaliutą „Bitcoin" (BTC). Iki tol blokų grandinių technologija nebuvo populiari, todėl šis lygiagretus technologijų paviešinimas ir įvedė painiavos. Iš tiesų, blokų grandinių tinklai yra paskirstyta (angl. *distributed*) P2P (*peer-to-peer*) sistema, kurioje išteklių mainus apskaito ne centrinė tarnyba (kitaip, trečioji šalis), o pačio tinklo dalyviai. Šiame poskyryje detaliau apžvelgiama, kokios technologijos ir principai naudojami kurti ir naudotis blokų grandinėmis.

Kriptografiniai raktai ir adresas

Klientai (vartotojai ar kompiuteriai), norėdami dalyvauti blokų grandinės tinkle, privalo turėti kriptografinių raktų porą, nes šie raktai yra naudojami autentifikacijai tinkle. Sukūręs norimą transakciją, klientas ją pasirašo savo privačiuoju raktu, taip įrodydamas savo tapatybę, ir siunčia transakciją į blokų grandinės tinklą apdorojimui [47].

Kiekvieną klientą, o tiksliau, kriptografinių raktų porą, atitinka adresas blokų grandinės tinkle[3]. Adresas yra unikalus identifikatorius, su kuriuo yra siejamos transakcijos blokų grandinėje. Kriptovaliutų atveju, valiuta yra siunčiama iš vieno adreso į kitą, t. y., transakcija turi siuntėją ir gavėją [47]. „Ethereum“ blokų grandinės išmaniuosiuose kontraktuose (angl. *smart contract*), transakcijose vietoje gavėjo įrašomas kontrakto adresas ir funkcijos pavadinimas bei jos parametrai [49].

Kriptovaliutų atveju, adresas atitinka banko sąskaitą tradicinėje finansų sistemoje. Kiekvienas adresas duotuoju laiko momentu turi, o tiksliau, turi teisę išleisti, persiųsti, tam tikrą kiek kriptovaliutos. Skirtingai nuo banko sąskaitos, kurioje visados yra saugomas galutinis sąskaitos likutis, norint nustatyti, kiek konkretus adresas turi kriptovaliutos, reikia peržiūrėti visas tos valiutos blokų grandinės transakcijas, kur adresas yra gavėjas arba siuntėjas ir susumuoti siųstos ir gautos valiutos kiekius.

„Bitcoin“ blokų grandinėje adresą sudaro 26–35 (daugumos – 34) skaičiai ir raidės. Adresas yra deterministiškai apskaičiuojamas iš viešojo rakto, todėl adresas atitinka viešąjį ir tuo pačiu privatųjį kriptografinį raktą. Pirmiausia, apskaičiuojamas viešojo rakto su priešdėliu „0x04“ maišos kodas naudojant paeiliui SHA256 ir RIPEMD160 maišos funkcijas. Prie gauto rezultato pridedamas blokų grandinės tinklo 1 baito ilgio identifikatoriaus priešdėlis ir apskaičiuojamas maišos kodas naudojant SHA256 du kartus iš eilės. Šio rezultato pirmi 4 baitai yra naudojami kaip kontrolinė suma praeito etapo metu gautai reikšmei. Galiausiai, paeiliui sujungus blokų grandinės tinklo identifikatorių, pirmojo etapo metu gautą reikšmę ir antrojo etapo metu gautą kontrolinės sumos dalį gaunamas adresas [50]. Palyginimui, „Ethereum“ tinkle dalyvio adresas generuojamas panašiai, tačiau naudojama mažiau žingsnių. Apskaičiuojamas dalyvio viešojo rakto maišos kodas naudojant „keccak256“ maišos funkciją ir paimami paskutiniai 20 baitų [49].

Maišos funkcijos

Maišos funkcijos bet kokio ilgio duomenų eilutę paverčia pastovaus ilgio tekstine reikšme. Pavyzdžiui, plačiai paplitusios maišos funkcijos MD5, SHA1, SHA256 grąžina atitinkamai 128, 160, 256 bitų reikšmes. Funkcijos pritaikymas užrašomas $h = H(M)$, kur h – pastovaus ilgio m maišos funkcijos rezultatas (maišos kodas), H – maišos funkcija, M – funkcijos įvestis.

Maišos funkcijos naudojamos pasirašant dokumentus elektroniniu parašu, kaip kontrolinės sumos ir kt., dėl to, kad šios funkcijos yra deterministinės. Tą pačią funkciją pritaikius tai pačiai įvesties reikšmei kelis kartus, rezultatas visada yra tas pats. Tokiu būdu žinant originalią didelio failo kontrolinę sumą, galima lengvai patikrinti, ar failas nebuvo pakeistas. Užtenka pritaikyti funkciją šiam failui ir palyginti gautą reikšmę su originaliąja.

Algoritmas, kuriuo reikšmei M apskaičiuojamas maišos kodas h, yra neimlus skaičiavimo ištekliams. Tuo tarpu, žinant h, apskaičiuoti M yra sudėtinga, nes maišos funkcijos yra vienkryptės – jas pritaikant informacija prarandama. Norint rasti pradinę reikšmę, tai įmanoma padaryti tik imant reikšmes iš eilės, pritaikant maišos funkciją ir palyginant su ieškomos reikšmės h. Kuo funkcijai

[3] Toliau darbe blokų grandinės kontekste „adresas blokų grandinės tinkle“ sutrumpintai vartojamas kaip „adresas“.

būdingas maišos kodas yra ilgesnis, tuo šis procesas yra sudėtingesnis ir tikimybė, kad bus rasta pradinė reikšmė mažesnė.

Maišos funkcijos taip pat pasižymi atsparumu kolizijoms. Sudėtinga rasti du skirtingus M ir M', kur $h = H(M)$, $h' = H(M')$, $h = h'$. Jeigu būtų lengva rasti tą patį maišos kodą turinčias skirtingas įvesties reikšmes, piktavaliai galėtų tuo manipuliuoti. Pavyzdžiui, vartotojas suformuotų teisinį dokumentą M ir elektroniniu parašu pasirašytų jo kontrolinę sumą h, piktavalis pakeistų dokumentą savo naudai (M') ir taip, kad jo kontrolinė suma būtų tokia pati kaip pradinio ir galėtų teigti, kad vartotojas pasirašė suklastotąjį dokumentą. Kuo maišos funkcijos generuojama reikšmė yra ilgesnė tuo atsparumas kolizijoms yra galimai didesnis, nes gali būti daugiau unikalių maišos kodo reikšmių [51].

Transakcijos

Kaip minėta anksčiau, blokų grandinėje informacija saugoma blokuose. Kiekvienas informacijos vienetas saugomas transakcijoje, o transakcijos jungiamos į bloką ir pridedamos prie esamos blokų grandinės.

Populiariausioje pagal kapitalizaciją blokų grandinės implementacijoje kriptovaliutoje „Bitcoin“ [52], transakcijoje saugomas pervedamos valiutos kiekis, siuntėjo ir gavėjo adresai, papildoma protokolo veikimui reikalinga informacija. Tinklo dalyvis, norėdamas pervesti kriptovaliutos kitam dalyviui, suformuoja transakciją, kurioje pateikia jau įvykdytų transakcijų, kurių metu gavo valiutos, maišos kodus, taip įrodydamas, kad turi teisę disponuoti tam tikru valiutos kiekiu. Taip pat siuntėjas privalo įrodyti, kad turi teisę kurti transakcijas iš norimo adreso. Adresas yra susietas su viešuoju (tuo pačiu, ir privačiuoju) raktu, todėl siuntėjas pasirašo transakciją privačiuoju raktu [47].

Transakcijose gali būti saugomi ne tik kriptovaliutų pervedimo įrašai, tačiau ir kitokia informacija. Pavyzdžiui, „Ethereum“ tinkle galima leisti paskirstytai vykdomas programas – išmaniuosius kontraktus (angl. *smart contract*). Išmanieji kontraktai saugo būseną blokų grandinės tinkle, o programos funkcijų iškvietimas gali šią būseną pakeisti. Visos būseną keičiančios užklausos yra saugomos transakcijose, užklausų funkcijų pavadinimai, parametrai įrašomi į transakcijos duomenų laukelį [49]. Siekiant sumažinti išteklių, reikalingų blokų grandinei, poreikį, transakcijos jungiamos į blokus.

Blokai

Pavienės transakcijos atspindi konkrečius informacijos vienetus, tuo tarpu blokai jas apjungia, užtikrina chronologinį nuoseklumą ir vientisumą, o blokai yra jungiami į grandinę. Kiekvienas blokų grandinės blokas turi informaciją apie tai, kada buvo sukurtas, sunkumą, apjungiamas transakcijas, maišos kodą, praeito bloko maišos kodą, kūrėjo adresą ir kt. Ši informacija varijuoja tarp konkrečių blokų grandinės implementacijų [49, 53, 54].

Priklausomai nuo blokų grandinės tinklo tipo, blokai yra sukuriami kas nustatytą laiko tarpą. Pavyzdžiui, „Bitcoin“ tinkle tai yra apie 10 minučių [55]. Siekiant, kad šis laiko tarpas būtų pastovus, blokų „kasimo“ sunkumas koreguojamas pagal tinklo skaičiavimo galią. Kuo daugiau kompiuterių prisijungia ir pradeda „kasti“, tuo sunkumas yra aukštesnis. „Kasimo“ sunkumas reguliuojamas nurodant maksimalią bloko antraštės maišos kodo reikšmę. Jeigu įtraukus transakcijas ir užpildžius kitą bloko informaciją, jo maišos kodas yra didesnis negu nustatyta maksimali reikšmė, blokas nėra priimamas į grandinę. Siekiant keisti bloko, turinčio nekintančią informaciją, maišos kodą, keičiamas jo laukelis „nonce“. Pakeitus „nonce“ reikšmę, pasikeičia bloko maišos kodas ir šis procesas kartojamas tol, kol gaunamas už maksimalią reikšmę mažesnis maišos kodas ir tada blokas pridedamas prie esamos grandinės [56].

Blokų grandinės blokuose saugomos ne tik pačios transakcijos, tačiau ir jų kontrolinė suma. Transakcijoms, įrašytoms į bloką, suskaičiuojama Merkle medžio kontrolinė suma. Kiekvienai transakcijai suskaičiuojamas maišos kodas, jie yra suporuojami ir tada poroms skaičiuojamas maišos kodas. Procesas kartojamas, kol iš visų transakcijų maišos kodų lieka tik vienas maišos kodas, vadinamas Merkle medžio šaknimi (angl. *Merkle Root*) [57].

Blokų jungimas į grandinę užtikrina, kad senesni įrašai (transakcijos ir blokai) negalėtų būti keičiami. Pradedant nuo pirmojo (angl. *genesis*), kiekvienas blokas turi unikalų maišos kodą, kuris atitinka bloko turinį. Kuriant naują bloką, į jo antraštę yra įrašomas praeito bloko maišos kodas, todėl šio naujojo bloko maišos kodas tampa priklausomas nuo praeito, o tiksliau, nuo visų senesnių blokų. Pakeitus bet kurio senesnio bloko bet kurį laukelį, pavyzdžiui, vienoje iš transakcijų įrašius kitą gavėją ar pervedamą kriptovaliutos sumą, to bloko, o tuo pačiu ir visų sekančių blokų, maišos kodai pasikeičia. Norint, kad pakeisti blokai būtų laikomi tinkama, tikrąja grandine, visų pirma, jų maišos kodai turi atitikti tuo metu galiojusio sunkumo reikalavimą, todėl kiekvienam blokui turi būti surastas naujas tinkamas „nonce“, kas reikalauja daug skaičiavimo galios. Visų antra, grandinė su pakeistais blokais turi patapti ilgesne negu esama tikroji, tad piktavaliai, keičiantys praeities įrašus, ne tik turi atlikti daug skaičiavimų ieškodami reikiamų „nonce“ reikšmių, tačiau dar skaičiavimus vykdyti greičiau negu juos vykdo pagrindinėje grandinėje dalyvaujantys kompiuteriai. Šis faktas nulemia, kad praeities įrašus aktyviame blokų grandinės tinkle keisti neapsimoka arba yra išvis neįmanoma [56].

Konsensuso algoritmas

Blokų grandinės tinklo dalyviai komunikuoja tarpusavyje, sinchronizuoja blokų grandinę ir naujas transakcijas, tvirtina blokus ir juos prideda prie esamos grandinės. Grandinės integralumas užtikrinamas naudojant konsensuso algoritmus. Tai yra taisyklės, pagal kurias tinklo dalyviai nusprendžia, kas turi teisę rašyti į grandinę.

Vienas populiariausių konsensuso algoritmų yra „darbo įrodymas“ (angl. *Proof of Work*, PoW). Naudojant šį algoritmą, tinklo dalyvis, norėdamas pridėti naują bloką į grandinę, privalo surasti tokią bloko „nonce“ reikšmę, kad bloko antraštės maišos kodas būtų mažesnis už tuometinį tinklo nustatytą sunkumo reikšmę. Kadangi maišos funkcijos yra vienkryptės, šis procesas yra atsitiktinis ir vienintelis būdas rasti reikiamą „nonce“ reikšmę yra ją parinkti atsitiktinai, skaičiuoti maišos kodą ir kartoti šį procesą, kol bus rastas tinkamas. Algoritmas vadinasi „darbo įrodymas“, nes norint rasti reikiamą „nonce“ reikšmę dažniausiai reikia sutelkti daug skaičiuojamosios galios. Šį algoritmą naudojančios blokų grandinės tinkle sunkumas yra reguliuojamas pagal tuo metu prie tinklo prisijungusių įrenginių skaičiavimo pajėgumą, tad net jeigu konkretus dalyvis skaičiavimui skiria daug skaičiavimo pajėgumų, o kiti dalyviai taip pat pasitelkia galingus kompiuterius, šiam dalyviui šansai rasti kito bloko reikiamą „nonce“ reikšmę yra maži, proporcingi jo ir tinklo skaičiavimo galios santykiui. Iš esmės, kuo dalyvis skiria daugiau skaičiavimo išteklių, tuo jam didesni šansai „iškasti“ kitą bloką ir gauti už tai atlygį. PoW yra neefektyvus išteklių atžvilgiu ne tik dėl to, kad dalyviai lenktyniauja sutelkiamų skaičiavimo pajėgumų atžvilgiu, tačiau ir kai kažkuris iš dalyvių „iškasa“ bloką ir įrašo jį į grandinę, visi kitų dalyvių, kurie buvo pradėję „kasti“ tą patį bloką, t. y., jau išbandę kažkiek „nonce“ reikšmių, pastangos tampa bevertės, nes išbandytos „nonce“ reikšmės ateities blokams nebetiks. Šie dalyviai skyrė išteklius (elektros energiją ir įrangą) skaičiavimams, kurie neatnešė jokios naudos, kitaip tariant, juos iššvaistė [56, 58].

Kitas konsensuso algoritmas – turto įrodymas (angl. *Proof of Stake*, PoS) – kitaip nei PoW, remiasi ne paskirta skaičiavimo galia, o tuo, kiek dalyviai turi kriptovaliutos konkrečiame tinkle. Pavyzdžiui, jeigu dalyvis turi 1 % viso egzistuojančio kriptovaliutos kiekio, jis gali „iškasti“ 1 % blokų.

PoS buvo pasiūlytas kaip alternatyva PoW siekiant padaryti tinklą saugesnį ir sumažinti energijos sąnaudas, reikalingas operuoti blokų grandinės tinklą ir tuo pačiu sumažinti transakcijų kainą. Kuo daugiau dalyvių prijungia skaičiavimo galios prie PoW algoritmą naudojančio tinklo, tuo bendri energijos ištekliai, reikalingi palaikyti tinklą ir patvirtinti transakcijas, auga ir tuo pačiu auga didėja transakcijų kaina. PoS sistemoje dalyviai nelenktyniauja stengdamiesi paskirti kuo daugiau skaičiavimo galios, todėl išlaidos ir transakcijų kaina mažesnė. Taip pat, ilgalaikėje perspektyvoje PoS yra saugesnis, nes norint įgyti didžiąją dalį tinklo valdymo galios, reikia turėti didžiąją arba beveik didžiąją dalį kriptovaliutos. Norint nusipirkti tiek bet kurios labiau naudojamos ir vertingesnės kriptovaliutos ne tik daug kainuotų skaičiuojant esamomis kainomis, tačiau supirkinėjant valiutą dideliais kiekiais, jos kaina kyla [56].

Kuriant naują tinklą (kriptovaliutos ar kitokį) pagal jo poreikius nusprendžiama, kuris protokolas bus naudojamas ir kokias algoritmais tinklas bus grįstas. Gyvuojant ir plečiantis tinklui, jeigu poreikiai pasikeičia, gali būti perrašomas tinklo protokolas ar keičiamas kitas funkcionalumas. Tokiu atveju įvykdomas atšakos pakeitimas (angl. *hard fork*) ir pakeičiamas tinklo veikimas. Po pakeitimo, kiekvienam tinklo dalyviui tenka nuspręsti, kurią atšaką jis palaikys ir joje dalyvaus. Kriptovaliutų atveju, dažnai abi atšakos palaikomos toliau ir valiuta skyla į dvi atskiras valiutas, kaip pavyzdžiui nutiko su „Bitcoin“, kuris skilo į „Bitcoin“ ir „Bitcoin Cash“, „Ethereum“ skilo į „Ethereum“ ir „Ethereum Classic“ ir kt. [59]

Blokų grandinės tinklų tipai

Toshendra Kumar’as Sharma iš „Blockchain Council“ išskiria du blokų grandinės tinklų tipus: atvirus (angl. *permissionless*) ir uždarus (angl. *permissioned*) [60]. Pagrindinis jų skirtumas yra tai, kas gali skaityti informaciją ir pridėti naujus blokus į grandinę.

Atviri blokų grandinės tinklai plačiai naudojami kriptovaliutoms, pavyzdžiui, „Bitcoin“ [47], „Ethereum“ [61], „Zcash“ [62], „Ethereum“ išmaniesiems kontraktams [61] ir kt. Prie šių tinklų gali jungtis visi norintys ir turintys reikiamą fizinę ir programinę įrangą [60]. Dalyviai gali „kasti“ tinklo valiutą, kurti transakcijas (kriptovaliutų atveju – pervesti valiutos kitam dalyviui). Atvirų blokų grandinės tinklų neįmanoma cenzūruoti, nes norint dalyvauti tinkle užtenka skirti skaičiavimo išteklių arba įsigyti valiutos ar jos atitikmens tame tinkle. Kaip jau minėta, tokių tinklų integralumas užtikrinamas naudojant konsensuso protokolus.

Tuo tarpu prieiga prie uždarų blokų grandinės tinklų yra ribojama ir teisė prisijungti suteikiama tik konkretiems dalyviams [60]. Šie tinklai gali būti skirstomi į dvi grupes: vieši ir privatūs [63]. Situacijose, kai reikia valdyti, kas gali rašyti į blokų grandinę, o skaityti leidžiama visiems, naudojami vieši uždari tinklai. Pavyzdžiui, valstybinės institucijos gali saugoti finansines ar darbų ataskaitas blokų grandinės tinkle, kai siekiama skaidrumo ir leidžiama gyventojams informaciją peržiūrėti, tačiau ją įvesti (t. y., kurti transakcijas) turi teisę tik institucijos darbuotojai. Kai prie blokų grandinėje saugomos informacijos reikia riboti prieigą tiek skaitymui, tiek įrašymui, naudojami privatūs uždarieji tinklai [64].

Uždaruose blokų grandinės tinkluose apdoroti transakcijas teisę turi tik numatyti blokų grandinės mazgai, kurių identitetas yra žinomas ir jais pasitikima, todėl nebelieka poreikio naudoti PoW ar kitą algoritmą konsensusui užtikrinti. Tokiu atveju paskata apdoroti transakcijas ir kurti blokus būna ne siekis gauti kriptovaliutos, o prievolė, susitarimas ar kitokia nauda.

1.2.2. Blokų grandinių privalumai

Blokų grandinės plačiai naudojamos ir pritaikomos dėl tam tikrų jų savybių. Daugelis tyrėjų sutaria [65, 66, 67, 68, 69, 70, 71], kad blokų grandinės ir jų tinklai turi išskirtinių savybių ir dėl to gali būti naudojamos įvairiose srityse. Dažniausiai minimi ir svarbiausi privalumai yra šie:

- skaidrumas ir atskaitomybė – kadangi informacija yra įrašoma į blokus, o blokai jungiami į grandinę, jau įrašytų duomenų neįmanoma ištrinti, todėl galima atsekti kas ir kaip keitė blokų grandinėje saugomus duomenis;
- duomenų integralumas – kiekvienas blokų grandinės tinklo mazgas turi pilną blokų grandinės kopiją (išskyrus tam tikrus atvejus), todėl net jeigu viename ar keliuose mazguose saugomi duomenys yra pakeičiami, kiti mazgai šių pakeitimų nepripažįsta, todėl piktavaliai norėdami pakeisti duomenis, tą turėtų padaryti daugumoje tinklo mazgų, kas yra nepraktiška;
- prieinamumas – kadangi blokų grandinės tinklas yra paskirstytas ir vienoda informacija saugoma visuose mazguose, vienam ar keliems mazgams nustojus veikti, tinklas gali toliau funkcionuoti – nuskaitant ir įrašant informacija į blokų grandinę tereikia jungtis prie kito mazgo;
- aukštas pasitikėjimo lygis – į blokų grandinę įrašoma nauja informacija tik tada, kai dalis ar dauguma tinklo mazgų sutaria dėl įrašomos informacijos, naudojami konsensuso algoritmai;
- greitai pasiekiami duomenys – kai tinklą sudaro daug mazgų, norint pasiekti jame saugomą informaciją, galima jungtis prie artimiausio mazgo, taip užtikrinant duomenų pasiekimo spartą;
- privatumas – priklausomai nuo blokų grandinių pritaikymo, gali būti išlaikomas aukštas tinklo vartotojų privatumo lygis, nes norint dalyvauti tinklo veikloje užtenka turėti kriptografinių raktų porą.

Skirtingos blokų grandinių pritaikymo sritys remiasi šiais privalumais. Kitame poskyryje apžvelgiamos blokų grandinių ir jų tinklų pritaikymo sritys.

1.2.3. Blokų grandinių pritaikymo sritys

Per pastaruosius keletą metų blokų grandinių pritaikymas įvairiose srityse išplito. Šiuo metu blokų grandinių tinklai naudojami įvairioms paskirtims nuo finansinių [72] ir draudimo paslaugų [73] iki medicinos [74] ir daiktų interneto [75]. Šiame poskyryje apžvelgiamos kelios blokų grandinių pritaikymo sritys ir kokius privalumus joms teikia blokų grandinių tinklai.

Bankinis sektorius

Bankų paslaugos yra paremtos transakcijomis, kai vieni žmonės perveda pinigus kitiems. Kiekvienas pinigų pervedimas turi gavėją, siuntėją, pervedamą pinigų sumą ir kitą informaciją. Blokų grandinės taip pat yra paremtos transakcijomis, jungiamomis į blokų. Taip pat, priklausomai nuo blokų grandinės tipo, transakcijos gali turėti pinigų pervedimui reikalingą informaciją. Iš tiesų, išpopuliarėjus kriptovaliutoms ir atsirandant naujiems jomis paremtiems atsiskaitymo būdams [76, 77, 78, 79], buvo įrodyta, kad blokų grandinės gali būti naudojamos pinigų operacijoms vykdyti.

Ye Guo'as ir Chen'as Liang'as [72] pažymi, kad blokų grandinių naudojimas bankiniame sektoriuje gali padėti išspręsti įvairias problemas. Dažnai bankų sistemos yra sudėtingos, naudojami sudėtingi kliringo (angl. *clearing*) mechanizmai – jie galėtų būti paprasčiau įgyvendinami naudojant

išmaniuosius kontraktus. Taip pat, naudojant blokų grandinių tinklų būtų galima sumažinti tarpinių transakcijų ir kitų procesų dalyvių skaičių, taip sumažinant veiklos kaštus. Blokų grandinių tinklai pasižymi aukštu duomenų integralumu dėl to, kad duomenys saugomi paskirstytame tinkle, t. y., tinklo mazgai turi pilnas duomenų kopijas, o kad duomenys nebūtų neteisėtai pakeisti, naudojami konsensuso algoritmai.

Draudimas

Draudimo paslaugos yra paremtos dviejų ar daugiau šalių bendru susitarimu. Draudimo įmonė gali teikti paslaugas žmonėms, dalyvauti kaip tarpininkai ir pan. Įvykus draudiminiam įvykiui, draudimo įmonė ištiria aplinkybes, nustato galimus kaltininkus ir apskaičiuoja išmokų dydžius. Šie procesai dažniausiai yra bent dalinai automatizuojami, tačiau vis tiek reikalauja žmonių darbo, dėl to yra lėti [73]. Taip pat, draudimo bendrovės kenčia nuo apgaulių, kai piktavaliai pateikia neteisingą informaciją [80]. Wenting'as Li ir kt. [81], Hiroki's Watanabe'is ir kt. [82] teigia, kad išmanieji kontraktai gali būti naudojami tokiais scenarijais, kai reikia įgyvendinti sudėtingas verslo procesų taisykles ir apsaugos nuo nelegalių ar netinkamų užklausų. Mayank'as Raikwar'as ir kt. [73] pasiūlė blokų grandinės panaudojimo draudimo paslaugoms modelį. Jų modelis paremtas išmaniaisiais kontraktais, kurie apskaito draudimo procesų išteklius ir vykdo transakcijas – klientų registravimą, įmokas už draudimą, išmokas už draudiminius įvykius ir kt. Išmaniuosiuose kontraktuose aprašytos taisyklės užtikrina, kad žmonės negalės neteisingai reikalauti išmokų, o draudimo įmonės negalės išvengti atsakomybės draudiminio įvykio atveju.

Medicina

Ligoninėms, poliklinikoms ir kitoms įstaigoms teikiant medicinines paslaugas, jos susiduria su įvairiais apribojimais ir poreikiais. Visų pirma, duomenys turi būti saugiai saugomi, be galimybės juos lengvai pakeisti. Tai apsaugo nuo piktavališkų kėslų ir užtikrina, kad bus galima patikrinti pacientų įrašų istoriją. Taip pat svarbu, kad būtų užtikrinamas pacientų privatumas, duomenis galėtų peržiūrėti tik autorizuotas personalas. Juan'as M. Roman-Belmonte'as ir kt. [74] pastebi, kad blokų grandinės gali būti pritaikomos medicinos srityje ir padėti įgyvendinti šiuos ir kitus poreikius. Blokų grandinėse saugoma informacija yra nepakeičiama, todėl užtikrinamas duomenų integralumas ir istorinis prieinamumas. Išmanieji kontraktai gali būti naudojami įgyvendinti daugelio gydymo procesų administravimą, automatizuoti tam tikrus sprendimus, užtikrinti, kad tik pacientui sutikus gydymo personalas gaus prieigą prie jo duomenų. Į išmaniuosius kontraktus galima įrašyti informacijos saugojimo ir nuskaitymo, sprendimų priėmimo algoritmus, o blokų grandinės tinklas užtikrina duomenų integralumą ir prieinamumą.

1.2.4. „Ethereum“ išmaniųjų kontraktų veikimas ir jų kūrimas

Blokų grandinių pritaikymo sričių analizės metu nustatyta, kad išmanieji kontraktai suteikia daug privalumų, dažnai jie naudojami „Ethereum“ blokų grandinės tinkle, todėl šiame poskyryje aptariami šių kontraktų veikimo principai ir kūrimas.

„Ethereum“ transakcija yra kriptografiniu raktu pasirašyta tam tikro formato žinutė, siunčiama tarp tinklo dalyvių. Transakcija susideda iš šešių reikšmių: [49]

- „nonce“ – iš siuntėjo adreso jau išsiųstų transakcijų skaičius, skirtas išvengti dvigubo išleidimo (angl. *double-spending*);
- „gasPrice“, „gasLimit“ – skirti nustatyti transakcijos kainą;

- „to“ – transakcijos gavėjo adresas;
- „value“ – pervedamos valiutos kiekis;
- „data“ – specialiai formatuota baitų žinutė, kurioje užkoduotas kviečiamos išmaniojo kontrakto funkcijos identifikatorius ir parametrai, arba naujo kontrakto kodas.

Taip pat, pasirašius transakciją, pridedamos kriptografinės parašo reikšmės. Transakcijos siuntėjas yra ne saugomas atskirai, o nustatomas iš parašo [49].

„Ethereum“ skiriasi nuo daugelio kitų blokų grandinių tuo, kad palaiko ne tik kriptovaliutas (pavyzdžiui, „Ethereum“, „Ethereum Classic“), tačiau ir suteikia galimybę paskirstytai vykdyti programas. Šios programos ir yra vadinamos išmaniaisiais kontraktais. „Ethereum“ blokų grandinė naudoja „Ethereum“ virtualią mašiną (angl. *Ethereum Virtual Machine*, *EVM*), kurioje yra vykdomas kontraktų programinis kodas. EVM palaiko steku paremtą baitinį kodą (angl. *stack-based bytecode language*), atitinkantį Tiuringo mašiną (angl. *Turing-complete*) [49, 83], todėl suteikia galimybę kurti sudėtingas programas. Siekiant patogesnio išmaniųjų kontraktų kūrimo, sukurtos aukštesnio lygio kalbos, kompiliuojamos į EVM baitinį kodą: „Solidity“ [84], „LLL“ [85], „Vyper“ [86], „Bamboo“ [87] ir kt. Objektinė programavimo kalba „Solidity“ yra viena populiariausių „Ethereum“ išmaniųjų kontraktų kūrimo kalbų [88].

Išmanieji kontraktai yra įdiegiami į blokų grandinę kaip transakcijos. Kai jos yra įrašomos į blokus ir prijungiamos prie blokų grandinės, išmanusis kontraktas tampa aktyvus ir tinklo mazgai pradeda jį vykdyti. „Ethereum“ tinkle gali dalyvauti dviejų tipų dalyviai: išorinės paskyros (angl. *Externally Owned Accounts*) – valdomos privačiais kriptografiniais raktais, turi susietą eterio (angl. *ether*, „Ethereum“ blokų grandinės valiuta) kiekį, gali siųsti transakcijas, neturi susieto programinio kodo – ir kontraktų paskyros (angl. *Contract Accounts*) – turi susietą eterio kiekį, turi susietą programinį kodą, turi būseną (angl. *state*). „Ethereum“ išmanusis kontraktas vykdomas, kai yra iškviečiama viena iš aprašytų jo funkcijų, kurias gali kviesti tiek išorinės paskyros, tiek ir kiti kontraktai. Funkcija yra iškviečiama kontrakto adresu siunčiant transakciją su tam tikrais duomenimis – išmaniojo kontrakto funkcijos identifikatoriumi ir parametrais [89].

„Ethereum“ blokų grandinės tinkle dalyvaujantys mazgai vadovaujasi „darbo įrodymo“ konsensuso mechanizmu, todėl blokų kasėjai už prie grandinės prijungtus blokus gauna atlygį, vadinamą „gas“ ir skaičiuojamą eterio dalimis – „wei“. Kviečiant išmaniojo kontrakto būseną keičiančią funkciją, suformuota ir pasirašyta transakcija siunčiama visiems „Ethereum“ tinklo mazgams, įrašoma į blokų grandinę. Tokios funkcijos kvietimo rezultatas matomas ne iš karto, o tik tada, kai transakcija patenka į blokų grandinę [75], todėl kuriant išmaniaisiais kontraktais paremtas sistemas, reikia į tai atsižvelgti. Tuo tarpu, kviečiant funkciją, kuri tik nuskaito išmaniojo kontrakto būseną, būsena nėra keičiama. Tokiu atveju transakcija nėra rašoma į blokų grandinę ir sinchronizuojama su visais tinklo mazgais, o tik įvykdoma lokaliame mazge, kuris iškart grąžina funkcijos kvietimo rezultatą. Tokių funkcijų kvietimas iš tinklo nereikalauja skaičiavimo galios, todėl nėra apmokestinamas [90].

Kompiliuojant „Ethereum“ išmaniuosius kontraktus, kartu yra sukuriama aplikacijų baitinė sąsaja (angl. *application binary interface*, ABI). ABI apibrėžia, kaip turi būti kviečiamos išmaniojo kontrakto funkcijos, t. y., koks funkcijos pavadinimas atitinką kurį baitinį kodą. Šis baitinis kodas yra naudojamas kviečiant išmaniuosius kontraktus naudojant nuotolines procedūrines užklausas (angl. *remote procedural call*, RPC) [83]. RPC leidžia kviesti funkcijas iš interneto svetainių, mobiliųjų programėlių ir kitų programų. „Web3.js“ yra viena iš populiariausių „JavaScript“ kalbos bibliotekų, skirtų kurti interneto puslapių vartotojo sąsajoms. Ši biblioteka naudojama jungtis prie „Ethereum“ blokų grandinės tinklo mazgų, siųsti eterį kitiems tinklo dalyviams, kurti išmaniuosius kontraktus, kviesti kontraktų funkcijas [90].

„Ethereum“ išmanieji kontraktai leidžia kurti sudėtingas programas ir jas paskirstytai vykdyti, taip užtikrinant, kad rezultatas bus patikimas. EVM vykdomi išmanieji kontraktai iš kitų programų pasiekiami per RPC, naudojant ABI. Galimybė sąveikauti su išmaniaisiais kontraktais iš kitų programų ir aplinkų praplečia kontraktų panaudojimo galimybes.

1.2.5. Blokų grandinių pritaikymas autentifikacijai

Blokų grandinės gali būti ir jau yra pritaikomos įvairiose srityse, kaip aptarta 1. 2. 3 poskyryje. Viena iš pritaikymo sričių yra identiteto valdymas. Įvairūs blokų grandinių tinklai, išmanieji kontraktai yra pritaikomi saugoti vartotojų autentifikacijos duomenis (slaptažodžius, viešuosius raktus ir kt.), valdyti vartotojų identitetą. Šiame poskyryje detaliau apžvelgiami du komerciniai identiteto valdymo sprendimai – „Hydro Raindrop“, „uPort“ – naudojantys blokų grandines. Iš šiuo metu rinkoje esančių įvairių blokų grandinės tinklų pritaikymo pavyzdžių [91, 92, 93, 94, 95, 96, 97, 98], pasirinkti būtent šie sprendimai, nes jie yra naudojami realiose sistemose, išbaigti, o jų veikimo principai ir naudojamos technologijos yra viešai apžvelgiamos ir laisvai prieinamos internete.

„Hydro Raindrop“

2018 m. įmonė „The Hydrogen Technology Corporation“ [99] paskelbė techninę „Hydro Raindrop“ karkaso apžvalgą [96] ir pristatė, kaip planuoja naudoti blokų grandinių technologijas pirmiausia vartotojų autentifikavimui ir galimai kitoms su vartotojais susijusioms operacijoms, kaip dokumentų valdymas, programų sauga, dirbtinis intelektas ir kt. Apžvalgoje detaliausiai aptariamas vartotojų autentifikacijos scenarijus ir siūlomos sistemos integravimas į kitas sistemas. „Hydro Raindrop“ gali būti naudojamas įdiegti autentifikaciją į trečiųjų šalių sistemą (toliau – paslauga) arba būti diegiamas kartu su jau egzistuojančia šios paslaugos autentifikacijos procedūra, taip padidinant saugumą (sukuriamas dviejų ar daugiau lygių autentifikacijos mechanizmas). Apžvalgoje aptariamas karkasas yra paremtas „Ethereum“ išmaniaisiais kontraktais ir susideda iš dviejų protokolų – serverinio (angl. *server-side*) ir klientinio (angl. *client-side*).

Autentifikacijai naudojant serverinį protokolą, vartotojas pirmiausia yra užregistruojamas „Hydro“ sistemos išmaniajame kontrakte. Kai vartotojo adresas įrašomas į blokų grandinę, jis tampa registruotu ir gali pradėti naudotis paslauga. Vartotojui jungiantis prie paslaugos, pirmiausia „Hydro“ sistema sugeneruoja tam tikrus transakcijos parametrus ir juos įrašo į blokų grandinę. Šie parametrai yra persiunčiami vartotojui, kuris iš jų suformuoja transakciją, ją pasirašo ir siunčia į blokų grandinę. „Hydro“ nuolatos tikrina ir laukia, kol blokų grandinėje atsiras vartotojo pasirašyta transakcija su anksčiau sugeneruotais parametrais ir jai atsiradus, praneša paslaugai, kad vartotojas įrodė tapatybę ir gali būti laikomas prisijungusiu. Registruojant bandomus ir sėkmingus prisijungimus prie paslaugos blokų grandinėje, galima realiu laiku stebėti sistemą ir reaguoti į piktavališkus veiksmus ir grėsmes. Visgi, autoriai pastebi, kad rašant informaciją (šiuo atveju, transakcijos parametrus) į blokų grandinę, tinklo dalyviai turi atlikti skaičiavimus ir už tai reikia mokėti. Todėl serverinis protokolas tinkamas tik sistemoms, prie kurių vartotojai jungiasi retai.

Siekiant išvengti transakcijų rašymo į blokų grandinę autentifikacijos metu ir su tuo susijusių mokesčių, apžvalgoje siūlomas antrasis – klientinis – protokolas veikia šiek tiek kitaip nei serverinis. Pirmiausia, vartotojas mobiliojoje programėlėje susikuria „Hydro“ identifikatorių ir kriptografinius raktus, iš kurių apskaičiuotas adresas kartu su identifikatoriumi yra įrašomas į blokų grandinę. Esminis skirtumas tarp serverio ir klientinio protokolų yra autentifikacijos žingsnyje: klientinio protokolo atveju vartotojas ne siunčia pasirašytą transakciją į blokų grandinę, o tik pasirašo paslaugos sugeneruotą žinutę. Pasirašyta žinutė yra siunčiama į išmanųjį kontraktą, kuris patikrina parašo

galiojimą ir ar pasirašęs vartotojas turi „Hydro“ identifikatorių, t. y., ar yra registruotas sistemoje ir grąžina atitinkamą atsakymą. Autentifikuojantis šiuo metodu, transakcijos nėra rašomos į blokų grandinę kiekvieno prisijungimo metu, todėl tokia sistema labiau pritaikyta paslaugoms su daug vartotojų.

„Ethereum“ išmaniųjų kontraktų naudojimas „Hydro Raindrop“ karkase suteikia galimybę paskirstytai vykdyti kontrakto funkcijas, užtikrinti rezultatų tikrumą naudojant konsensuso mechanizmus. Taip pat, daugelio tinklo mazgų naudojimas užtikrina, kad net nustojus veikti vienam ar keliems mazgams, sistemos funkcionalumas nesutriks. Du skirtingų paskirčių „Hydro Raindrop“ karkasai pasižymi skirtingu funkcionalumu ir privalumais: serverinis protokolas leidžia stebėti prisijungimus ir reaguoti į grėsmes realiu laiku, vykdyti autentifikacijos apskaitą; klientinis protokolas skirtas paslaugoms su daug vartotojų ir kadangi prisijungimo metu informacija nėra rašoma į blokų grandinę, šis protokolas reikalauja mažai tinklo išteklių.

„uPort“

Dr. Christian’as Lundkvist’as ir kt. [100] 2016 m. pristatė identiteto valdymo platformos „uPort“ [92], kuria siekiama užtikrinti asmeninės vartotojų informacijos saugumą, pirminę idėją. Autoriai siekė sukurti tokią platformą, kurioje vartotojai galėtų patys valdyti savo asmeninę informaciją, rinktis, kada suteikti prieigą prie jos trečiosioms šalims. Taip pat, apžvalgoje teigiama, kad perkeliant jautrios informacijos saugojimą iš centralizuotų sistemų į vartotojų įrenginius, būtų užtikrinamas aukštesnis informacijos saugumo lygis, nes piktavaliai norėdami pavogti didelį kiekį informacijos, turėtų įsilaužti į daugelio vartotojų įrenginius. Platforma „uPort“ dalyvauja kaip tarpininkas tarp vartotojų ir trečiųjų šalių, kurios siekia gauti vartotojų informaciją. Vartotojai, naudodamiesi „uPort“ mobiliąja programėle gali jungtis prie paslaugų, siųsti kriptovaliutas, sąveikauti su „Ethereum“ tinkle veikiančiomis paskirstytomis programomis (angl. *decentralized application*) ir kt.

Platforma „uPort“ paremta „Ethereum“ blokų grandine ir išmaniaisiais kontraktais. Pagal minėtą apžvalgą, vartotojui, užsiregistravusiam „uPort“ sistemoje, yra sukuriami du išmanieji kontraktai: valdymo (angl. *controller*) ir tarpininko (angl. *proxy*). Jungiantis prie trečiųjų šalių paslaugos ar atliekant kitas operacijas, vartotojo programėlė su paslauga komunikuoja per šiuos du kontraktus. Valdymo kontrakte saugoma įvairi logika, leidžianti pakeisti savo adresą pačiam ar per nustatytus kontaktinius asmenis. Tuo tarpu, tarpininko kontraktas yra paslaugoms kaip vartotojo tapatybės atitikmuo. Tai reiškia, kad vartotojo tapatybė ir reputacija yra susieta su tarpininko kontrakto adresu, todėl net vartotojui pametus ar pakeitus įrenginį, tarpininko kontrakto adresas nesikeičia ir vartotojas išlaiko savo identitetą paslaugų atžvilgiu.

„Ethereum“ išmaniųjų kontraktų (ir kriptografijos apskritai) naudojimas „uPort“ kūrėjams leido užtikrinti, kad vartotojai patys galės valdyti savo asmens duomenis ir suteikti tik tiek prieigos ir tik toms paslaugoms, kuriems jie nori. Taip pat, pasitelkus papildomus išmaniuosius kontraktus, autoriai pasiūlė schemą, kur su vartotojo tapatybe susiejami trečiųjų šalių (pavyzdžiui, valstybinių institucijų, bankų ir kt.) fizinės vartotojo tapatybės patvirtinimai. Tokiu būdu vartotojas kaip asmuo yra susiejamas su skaitmenine tapatybe (adresu blokų grandinės tinkle) ir prisijungęs prie naujos paslaugos, jis prisijungia ne kaip nauja anoniminė tapatybė, o kaip patvirtintas asmuo.

2. BLOKŲ GRANDINĖS PANAUDOJIMO AUTENTIFIKACIJAI MODELIS

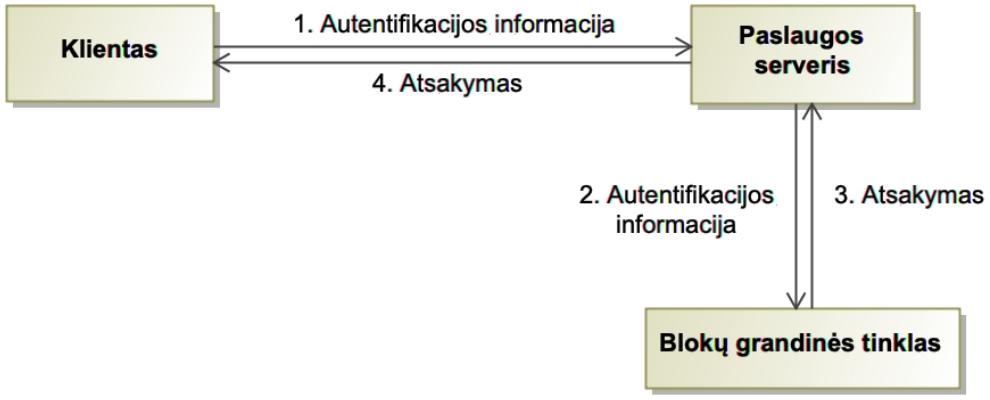
Autentifikacijos metodų analizės metu įvertinti jų privalumai ir trūkumai, taip pat, apžvelgtos problemos, susijusios su centralizuoto autentifikavimo sistemomis. Analizuojant blokų grandines ir jų tinklus, veikimo principus, kitų autorių atliktus darbus, nustatyta, kad blokų grandinės gali būti panaudojamos įvairiems tikslams, įskaitant ir identiteto valdymą. Apžvelgiant kitų autorių sukurtus blokų grandinių panaudojimo autentifikacijai modelius, pastebėti įvairūs trūkumai, todėl kuriant teorinį modelį, siekiama į juos atsižvelgti ir pašalinti.

Kuriamam teoriniam sistemos modeliui iškelti šie reikalavimai:

1. Vartotojų autentifikavimas turi vykti greitai, pateiktos autentifikavimo informacijos tikrinimas turi užtrukti ne daugiau nei pusę sekundės.
2. Vartotojas, norėdamas prisijungti prie paslaugos, neprivalo turėti blokų grandinės tinklo valiutos.
3. Sukurtas modelis gali būti pritaikomas autentifikuoti vartotojus tinklo paslaugose, pavyzdžiui, internetinėse svetainėse.
4. Modelis turi pasižymėti daliniu vieningo prisijungimo sistemos funkcionalumu – vartotojų paskyrų informacija turi būti sinchronizuojama ir prieinama iš bet kurio blokų grandinės tinklo mazgo.

2.1. Bendra sistemos, naudojančios blokų grandinės tinklą vartotojų autentifikacijai, architektūra

1.1 skyrelyje aptarta tipinė autentifikacijos ir autorizacijos schema. Šiame skyriuje kuriant teorinį sistemos modelį, kaip autentifikacijos serveris yra naudojamas blokų grandinės tinklas (žr. 2 pav.) Taip pat, nurodoma, kokią techninę ir programinę įrangą privalo turėti kiekvienas sistemos komponentas. Pastebėtina, kad į kuriamą modelį privilegijų serveris nėra įtraukiamas, nes šio darbo tikslas yra sukurti autentifikacijos sistemą. Naudojant kuriamą autentifikacijos sistemą, privilegijų serverio pasirinkimas nėra svarbus, nes autentifikacijos serveris yra atsakingas tik už vartotojo autentifikavimo etapą, o autorizacijos serveris – autorizavimo etapą.



2 pav. Blokų grandinės pritaikymo autentifikacijai modelis

Prieš pradedant nagrinėti, kaip šioje sistemoje bus vykdoma autentifikacija, vertėtų detaliau apžvelgti kiekvieną sistemos komponentą. Pastebėtina, kad šiame modelyje „Klientas“ atitinka dvi žmonių grupes: paslaugos serverio vartotojus (toliau – vartotojas) ir vartotojų autentifikacijos

informacijos (paskyrų) administratorius (toliau – administratorius). Pirmiausia aptariama vartotojo autentifikacija, vėliau – autentifikacijos informacijos administravimas.

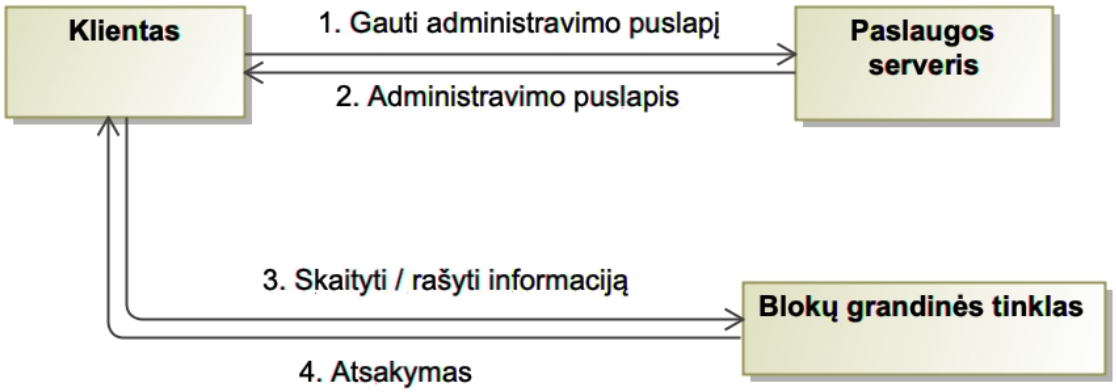
2 paveiksle vaizduojamoje sistemoje vartotojas, norėdamas gauti prieigą prie paslaugos, privalo turėti išmanųjį įrenginį (pavyzdžiui, telefoną, kompiuterį, planšetę ar pan.) su interneto prieiga ir įdiegta programine įranga, leidžiančia pasirašyti žinutes naudojant kriptografinius raktus. Taip pat vartotojas įrenginyje privalo turėti kriptografinių raktų porą, kurią naudos žinutėms pasirašyti ir taip įrodys savo tapatybę paslaugos serveriui.

Paslaugos serveris yra atsakingas už vartotojo autentifikacijos informacijos priėmimą, apdorojimą ir atitinkamo atsakymo grąžinimą. Paslaugos serveris komunikuoja su blokų grandinės tinklu, kad patikrintų, ar klientas yra autentifikuotas. Serveryje turi būti įdiegta „Web3“ standartą palaikanti „API“ biblioteka, kuri yra naudojama jungtis prie blokų grandinės tinklo, nuskaityti informaciją iš blokų grandinės.

Blokų grandinės tinklas yra skirtas saugoti autentifikacijos informaciją ir iš esmės nėra svarbu, kokios blokų grandinės tinklas bus naudojamas. Šiai paskirčiai užtektų net ir vieno blokų grandinės „tinklo“ mazgo, t. y., lokalios blokų grandinės, tačiau blokų grandinės tinklas su daugiau nei vienu mazgu turi daugiau privalumų ir siūlomas modelis remiasi būtent blokų grandinės tinklu, o ne lokalia blokų grandine.

Blokų grandinėje saugoma informacija galėtų būti, pavyzdžiui, prisijungimo vardas ir slaptažodis, tačiau šiame modelyje naudojamas kriptografinis identiteto įrodymas, t. y., naudojami kriptografiniai raktai. Vartotojas, norėdamas įrodyti savo tapatybę, privalės pasirašyti tam tikrą žinutę taip įrodydamas, kad turi konkrečius kriptografinius raktus, kuriuos atitinka adresas blokų grandinės tinkle. Būtent šie adresai ir bus saugomi blokų grandinėje. Pagal poreikį, į blokų grandinę galima papildomai įrašyti kiekvieno vartotojo autentifikacijos statusą („galiojanti“, „atšaukta“), galiojimo pradžios ir pabaigos datas, papildomus to paties kliento adresus ir kitą informaciją. Taip pat pastebėtina, kad saugumo sumetimais teisę rašyti į blokų grandinę privalo turėti tik patikimi asmenys (adresai), o skaitymo iš blokų grandinės leidimai nustatomi pagal konkretaus šio modelio panaudojimo poreikius. Jeigu autentifikuotų asmenų sąrašas nėra slaptas, blokų grandinės tinklą skaitymui galima palikti atvirą, kitu atveju – tinklas privalo būti uždaras.

Administruojant autentifikacijos informaciją siūlomo modelio komponentų tarpusavio sąveikos yra kitokios. Šio proceso bendra architektūra vaizduojama 3 paveiksle. Paslaugos serveris nesijungia prie blokų grandinės tinklo, nes nevykdo jokių skaitymo ir rašymo operacijų. Su blokų grandinės tinklu sąveikauja tik klientas (šiuo atveju, administratorius), jis kuria ir savo privačiuoju kriptografiniu raktu pasirašo transakcijas, kuriomis keičia blokų grandinėje saugomos autentifikacijos informacijos būseną. Administratoriaus įrenginyje turi būti įdiegta „Web3“ standartą palaikanti „API“ biblioteka, kuri yra naudojama jungtis prie blokų grandinės tinklo, nuskaityti informaciją iš blokų grandinės ir kurti transakcijas, t. y., įrašyti naują informaciją į blokų grandinę.



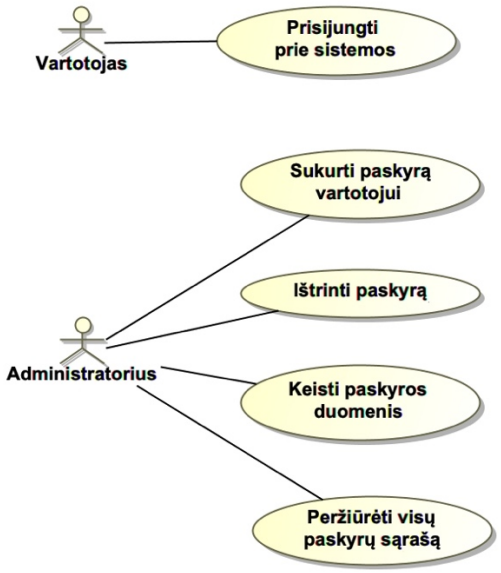
3 pav. Bendra autentifikacijos informacijos administravimo architektūra

Šiame procese paslaugos serveris dalyvauja tik kaip pagalbinė priemonė, leidžianti klientui komunikuoti su blokų grandinės tinklu. Paslaugos serveris pateikia patogų administravimo puslapį ir informaciją kaip prisijungti prie blokų grandinės tinklo, pavyzdžiui, jo prieigos adresą, prievadą ir kt., arba, jeigu naudojamas „Ethereum“ ar kitas išmanusis kontraktas, jo adresą ir aplikacijų baitinę sąsają. Teoriškai, jeigu visą šią informaciją turi pats administratorius, paslaugos serveris šiam tikslui nebėra reikalingas ir administratoriaus įrenginys gali tiesiogiai komunikuoti su blokų grandinės tinklu. Visgi, kuriant sistemą šio modelio pagrindu, patogumo dėlei galima sukurti atskirą tinklalapį, nesusijusį su konkrečia teikiama paslauga ir skirtą tik palengvinti autentifikacijos informacijos administravimą.

Toliau šiame skyriuje bus apžvelgiamos konkrečios sistemos operacijos: autentifikacija, vartotojų registravimas ir paskyrų valdymas.

2.2. Vartotojų ir administratorių panaudos atvejai

Siūlomas teorinis modelis yra pritaikomas autentifikacijai tinklo paslaugose, pavyzdžiui, interneto svetainėje. Tokios sistemos vartotojai gali atlikti vieną veiksmą – prisijungti, o administratoriai turi daugiau panaudos atvejų. Vartotojų ir administratorių panaudos atvejai vaizduojami 4 paveiksle.



4 pav. Vartotojų ir administratorių panaudos atvejai

Administratorius gali atlikti visus su vartotojų paskyromis susijusius veiksmus: sukurti naują, ištrinti esamą, keisti duomenis ir peržiūrėti visų esamų paskyrų sąrašą.

2.2.1. Vartotojų autentifikacija

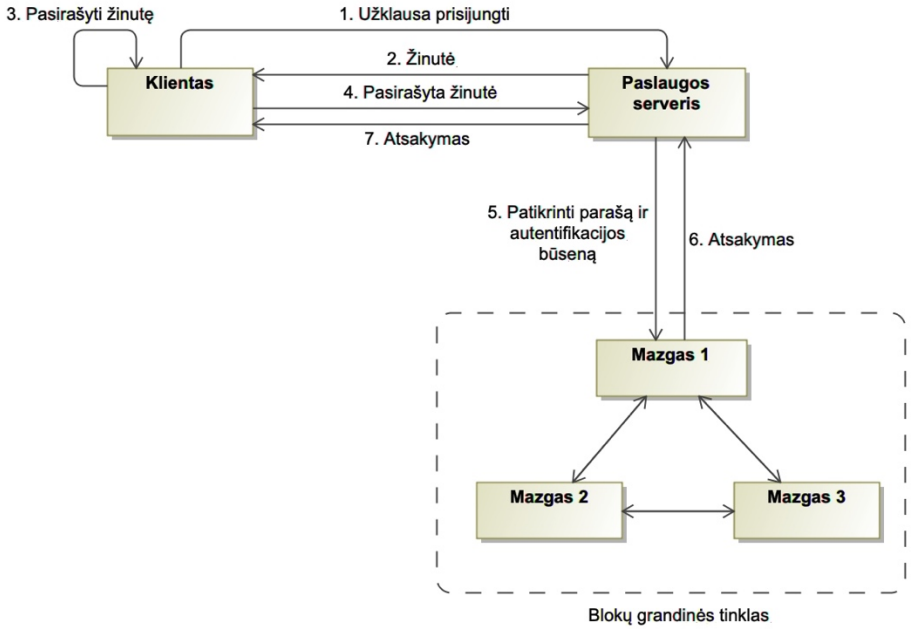
Vartotojas, siekdamas gauti prieigą prie paslaugos, įrodo savo tapatybę privačiuoju kriptografiniu raktu pasirašydamas tam tikrą žinutę. Teoriškai, sistemos funkcionalumui žinutės turinys įtakos neturi, tačiau visados pasirašyti tą pačią žinutę būtų nesaugu. Piktavalis, pavogęs vartotojo pasirašytą žinutę, galėtų apsimesti vartotoju neribotą skaičių kartų ir tai galėtų tęstis tol, kol galiotų vartotojo prieiga prie paslaugos. Norint padidinti saugumą, darbo autorius siūlo unikalią unikalią pasirašomą žinutę formuoti paslaugos ar autentifikacijos serveryje ir tada persiųsti ją į kliento įrenginį pasirašymui. Žinutė turėtų būti ne tik unikali, bet ir galioti tam tikrą nustatytą kiekį kartų arba nustatytą laiko intervalą. Nustatant tam tikrą kiekį kartų, kiek leidžiama prisijungti su ta pačia žinute, serveris turi vykdyti prisijungimų skaičiaus apskaitą lokalioje duomenų bazėje, faile ar kitur.

Administratoriui paslaugų serveryje nustačius, kad žinutė galioja tik vienam prisijungimui, serveris sugeneruoja naują atsitiktinę žinutę ir išsaugo ją kartu su informacija kiek kartų ji buvo panaudota (t. y., sukūrimo metu – 0 kartų) duomenų bazėje. Sugeneruotą žinutę išsiunčia klientui, šis ją pasirašo, autentifikacija įvyksta sėkmingai ir serveris duomenų bazėje atnaujina informaciją, kad ši žinutė panaudota vieną kartą. Jeigu tas pats arba kitas klientas vėl atsiųstų pasirašytą tą pačią žinutę, serveris patikrintų, kad žinutė jau buvo panaudota ir atmestų prisijungimo prašymą. Šis metodas reikalauja papildomos apskaitos paslaugos serveryje, tad siekiant, kad blokų grandinės tinklo panaudojimas autentifikacijai būtų lengviau integruojamas, patogiau būtų naudoti žinutes su galiojimo laiku.

Autentifikacijai naudojant žinutę, kuri turi galiojimo laiką, serveris ją sugeneruoja gavęs vartotojo autentifikacijos užklausą, į žinutę įrašo jos galiojimo pabaigos laiką. Teoriškai įmanoma, kad siunčiant tokią žinutę pasirašyti klientui, jis, ar bet kas kitas, galėtų gana paprastai pailginti žinutės galiojimo laiką – tiesiog pakeisti žinutėje esančią laiko žymą. Žinoma, pačiam vartotojui nėra jokios naudos tai daryti. Tačiau jeigu piktavalis turi priėjimą prie nešifruoto vartotojo interneto srauto, jis gali specialiai pailginti žinutės galiojimą, kad kai vartotojas ją pasirašys ir panaudos autentifikacijai, piktavalis galėtų pavogti parašą ir turėtų daugiau laiko juo pasinaudoti, apsimesdamas tikruoju vartotoju. Siekiant išvengti šios problemos, siūloma naudoti ryšio šifravimą arba paslaugos serveris žinutę turėtų pasirašyti savo privačiuoju kriptografiniu raktu ir siųsti klientui ne pačią žinutę, o jos parašą. Tokiu būdu serveris gali patikrinti kliento pasirašytos žinutės autentiškumą ir klastotės atveju atmesti prašymą prisijungti.

Žinutės panaudojimo laiko ribojimas yra mažiau saugus nei panaudojimo kartų ribojimas. Naudojant laiko ribojimą, žinutė turi galioti priimtiną laiko tarpą, kad vartotojas spėtų ją pasirašyti savo įrenginyje ir išsiųsti atgal į serverį. Tačiau tuo pačiu šis laiko tarpas turi būti kuo trumpesnis, kad piktavaliai pavogę pasirašytą žinutę turėtų mažiau laiko ja pasinaudoti. Žinutės panaudojimo kartų ribojimas yra saugesnis žinutės galiojimo valdymo būdas, nes nustačius, kad žinutė galioja vieną kartą, net pavogus parašą ir bandant jį iškart panaudoti, jis nebegalios, nes vartotojas parašą spės „išnaudoti“ pirmas ir serveris antrą kartą tos pačios pasirašytos žinutės nepriims. Priklausomai nuo verslo pobūdžio ir sistemos saugumo reikalavimų reikia nuspręsti, kurį žinutės galiojimo valdymo būdą naudoti.

Pasirašęs paslaugos serverio sugeneruotą žinutę vartotojas ją ir parašą siunčia atgal į serverį. Serveris patikrina parašą ir jeigu šis yra tikras, o pasirašyta žinutė yra galiojanti, kreipiasi į blokų grandinės tinklą ir nustato, ar žinutę pasirašiusio vartotojo adresas yra autentifikuotų klientų sąraše. Jeigu reikia, serveris taip pat patikrina vartotojo privilegijas ir galiausiai grąžina atsakymą klientui (žr. 5 pav.)



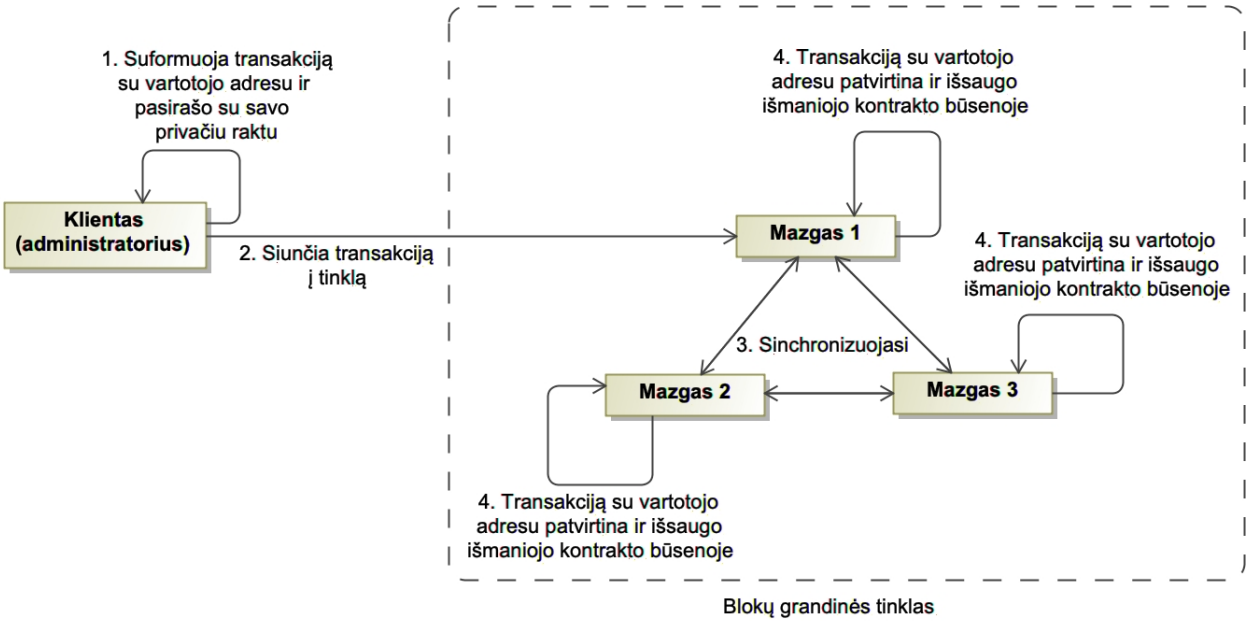
5 pav. Kliento autentifikavimas naudojant blokų grandinės tinklą

Autentifikacijos metu blokų grandinės tinklas naudojamas tik informacijai nuskaityti, todėl procesas vyksta gerokai greičiau negu tuo atveju, jeigu būtų kuriamos transakcijos ir laukiama, kol jos bus įtrauktos į blokus ir patvirtintos.

2.2.2. Naujo vartotojo registracija

Norėdamas prisijungti prie sistemos, vartotojas pirmiausia turi būti joje užregistruotas. Vartotojus registruoja ir jų įrašus tvarko administratorius. Kaip aptarta 2. 1 skyrelyje, dėl patogumo gali būti naudojama papildoma sistema, suteikianti informacijos apie blokų grandinės tinklą. Taip pat administratoriaus įrenginys gali turėti visą reikalingą informaciją ir, tokiu atveju, papildoma sistema nenaudojama, todėl tolimesnėse schemose ji yra praleidžiama ir procesai vaizduojamai pradedant transakcijos formavimu ir pasirašymu. Suformavęs transakciją su registruojamo vartotojo adresu ir kitais (jeigu reikalingi) papildomais duomenimis ir pasirašęs savo privačiuoju kriptografiniu raktu, administratorius ją siunčia į vieną iš mazgų[4], kuris transakciją sinchronizuoja su kitais blokų grandinės tinklo mazgais. Tada transakcija patvirtinama ir įtraukiama į blokų grandinę, taip įrašant autentifikacijos duomenis. Šis procesas vaizduojamas 6 paveiksle.

[4] Blokų grandinės mazgas gali būti lokalus arba nutolęs. Siūlomo modelio fizinė schema detaliau aptariama 3.6 skyrelyje.

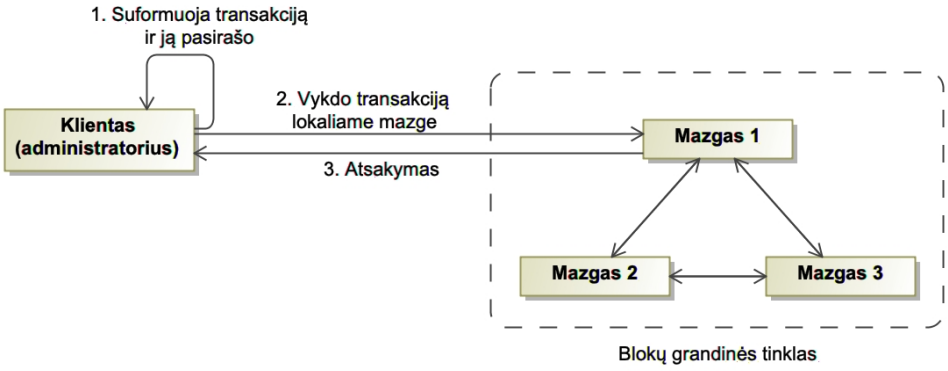


6 pav. Naujo vartotojo registracija blokų grandinės tinkle

Pagal tai, kokia blokų grandinė yra naudojama, turėtų būti taikoma rašymo į blokų grandinę kontrolė. Pavyzdžiui, galima naudoti uždarąjį blokų grandinės tinklą ir rašymo teises suteikti tik konkretiems adresams. Jeigu naudojami išmanieji kontraktai, rašymo teises galima valdyti ir kontrakto lygmenyje – į kontrakto kodą įtraukti atitinkamą patikrą.

2.2.3. Autentifikuotų vartotojų sąrašo gavimas

Administratorius gali ne tik registruoti vartotojus, bet ir peržiūrėti visą jų sąrašą ir informaciją, susijusią su jų paskyromis. Pagal tai, kokia blokų grandinė yra naudojama, skaitymo operacija skiriasi. Naudojant pavienę blokų grandinę, nuskaitymas vykdomas prisijungiant prie vieno iš mazgų ir nuskaitant visus blokus ir susumuojant visą transakcijų informaciją į galutinę būseną (žr. 7 pav.) Ši galutinė būsena ir yra visų registruotų vartotojų sąrašas.

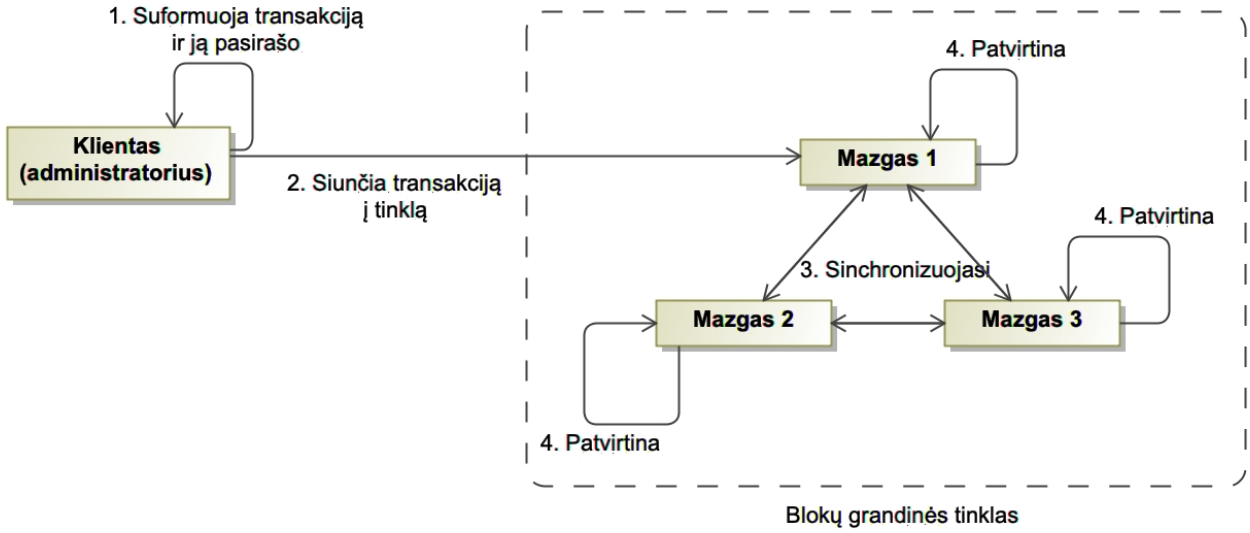


7 pav. Informacijos nuskaitymo operacija blokų grandinės tinkle

2.2.4. Vartotojų registracijos būsenos keitimas

Laikui bėgant kai kurie vartotojai gali nebetekti teisės naudotis paslauga, todėl reikia numatyti būdą ištrinti jų paskyras. Administratorius, norėdamas ištrinti paskyrą, suformuoja transakciją su atitinkamais duomenimis – vartotoju adresu ir požymiu, kad registracija atšaukiama, ir ją pasirašo.

Pasirašytą transakciją siunčią į vieną iš mazgų ir toliau procesas vyksta taip pat kaip 2. 2. 2 skyrelyje aprašytame procese (žr. 8 pav.)



8 pav. Vartotojo paskyros ištrynimas

Net ir ištrynus vartotojo paskyrą, blokų grandinėje pirmiausia bus matoma informacija, kad konkretus vartotojas buvo registruotas, nes nuskaitant blokų grandinėje esančią transakcijų informaciją, tai daroma iš eilės, todėl įrašas, kad vartotojas užregistruotas, bus randamas pirmiau nei įrašas, kad vartotojo registracija yra atšaukta. Visgi baigus visų transakcijų tikrinimą, konkretaus vartotojo būsena bus pakeista pagal paskutinę jam priskirtą reikšmę, šiuo atveju – jog registracija yra atšaukta.

Registracijos būsenos valdymas ir su paskyra susijusios informacijos saugojimas plačiau aptariamas 2.3 skyrelyje.

2.3. Papildomos vartotojo paskyros informacijos saugojimas

Blokų grandinę naudojant autentifikacijos sistemoje, svarbiausia yra tinkamai saugoti pagrindinę vartotojų prisijungimo informaciją. Kaip aptarta 1. 1 skyrelyje, prisijungimo informacija gali būti unikalus identifikatorius ir slaptažodis, viešasis kriptografinis raktas, biometriniai duomenys ir kt. Tačiau priklausomai nuo sistemos reikalavimų, galima saugoti papildomą informaciją, susijusią su vartotojo paskyra sistemoje. Pavyzdžiui, galima saugoti, kada paskyra buvo sukurta ir iki kada galioja, vartotojo rolę ir sudėtingesnes prieigos valdymo taisykles: leisti jungtis tik tam tikrą savaitės dieną ar dienų intervalą, tik tam tikromis valandomis ir panašiai.

Pastebėtina, kad priklausomai nuo naudojamos blokų grandinės, prieigos taisyklių pritaikymą galima įdiegti arba į paslaugos serverį, kuris pagal iš blokų grandinės gautus paskyrų duomenis nuspręs, ar konkreti paskyra duotuoju laiko momentu galioja, arba į blokų grandinės tinklą, kai naudojami išmanieji kontraktai. Išmanieji kontraktai gali būti suprogramuoti tiek priimti, saugoti ir grąžinti duomenis, tiek ir juos interpretuoti ir suformuoti atitinkamą atsakymą.

Į blokų grandinę taip pat galima įrašyti papildomus autentifikuotų vartotojų adresus. Jeigu vartotojai naudoja kelis įrenginius, šių įrenginių adresų priskyrimas vienai paskyrai įgalina vartotojus prie paskyros prisijungti naudojant bet kurį iš įrenginių.

3. „ETHEREUM“ IŠMANIOJO KONTRAKTO SUKŪRIMAS IR PRITAIKYMAS VARTOTOJŲ AUTENTIFIKACIJAI

3.1. Modelio architektūra ir naudojama programinė įranga

2. 1. poskyryje jau aprašyta bendra siūlomo sprendimo architektūra, o šiame poskyryje bus patikslinama, kokios technologijos naudojamos kuriant baigiamojo darbo sistemos prototipą. Taip pat, aprašomas sukurto prototipo tyrimas.

Autentifikacijos duomenims saugoti pasirinkta naudoti ne gryną (angl. *plain*) blokų grandinę, o išmaniuosius kontraktus, nes juos galima pritaikyti individualiai sistemai. Galimybė vykdyti programinį kodą blokų grandinės tinkle, leidžia ne tik patogiai saugoti duomenis įvairiomis struktūromis, tačiau ir manipuliuoti duomenimis. Dėl jau įrodyto panaudojamumo įvairiose srityse, šio darbo išmaniųjų kontraktų platforma buvo pasirinkta „Ethereum“ blokų grandinė.

Kuriamame prototipe paslauga, prie kurios jungiasi vartotojai, yra interneto svetainė, vartotojus autentifikuojanti naudojant „Ethereum“ išmanųjį kontraktą. Paslaugos tinklalapiui kurti pasirinkta programavimo kalba „JavaScript“. Ši programavimo kalba paranki tuo, kad ja galima ir kurti tinklalapius (šiai užduočiai atlikti bus naudojamas „AdonisJs“ karkasas), ir naudoti ją kuriant vartotojo sąsają. „JavaScript“ kodo biblioteka „Web3.js“ buvo pasirinkta šio prototipo kūrimui, nes ji įgalina vartotojus pasirašyti žinutes savo kriptografiniais raktais, jungtis prie blokų grandinės tinklo ir kt. Taip pat šią biblioteką paslaugos serveris naudoja jungtis prie blokų grandinės, kai yra tikrinama, ar vartotojas yra autentifikuotas. „Web3.js“ pritaikoma ne tik kuriant vartotojo sąsają, bet ir paslaugos serverio vidiniame (angl. *back-end*) programiniame kode.

Kaip aptarta 2.1 poskyryje, siekiant palengvinti autentifikacijos informacijos administravimą, galima sukurti tinklalapį, suteikiantį patogią administravimo aplinką. Šiame prototipe administravimo ir paslaugos tinklalapiai dėl patogumo sujungiami į vieną. Prieš naudodamiesi svetaine, vartotojas ir administratorius pirmiausia privalo prisijungti. Po prisijungimo serveris nustato, ar prisijungė paslaugos vartotojas, ar administratorius ir nukreipia į atitinkamus puslapius, kuriuose rodoma atitinkama informacija.

Prisijungimas prie paslaugos įvyksta vartotojui savo privačiuoju kriptografiniu raktu įrodžius savo tapatybę. Kriptografiniai raktai yra saugomi vartotojo kompiuteryje ir gali būti valdomi įvairių naršyklės plėtinių ar programų, pavyzdžiui, naršyklės plėtinio „MetaMask“, kuris ir yra pasirinktas naudoti šiame darbe. „MetaMask“ leidžia patogiai valdyti kriptografinius raktus (galima tiek generuoti naujus raktus ir saugoti savo kompiuteryje, tiek ir naudoti USB kriptografinius raktus), pasirašyti žinutes, kurti „Ethereum“ blokų grandinės transakcijas ir kt.

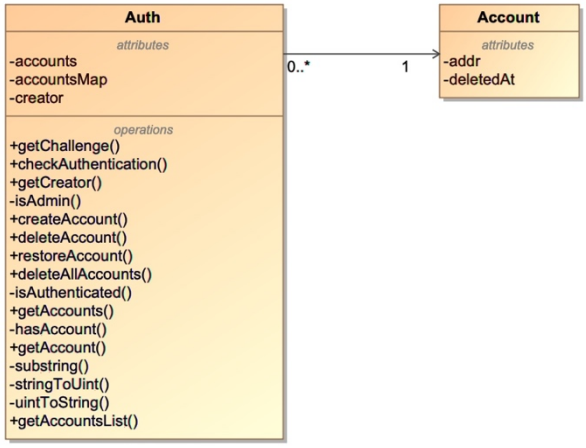
3.2. Išmaniojo kontrakto, skirto valdyti vartotojų autentifikacijos informaciją, kūrimas

Kaip aprašyta 3. 1 poskyryje, šiame darbe vartotojų autentifikacijos informacijai saugoti nuspręsta naudoti išmaniuosius kontraktus dėl jų lankstumo ir universalumo. Prototipo veikimui pademonstruoti, sukurtas išmanusis kontraktas „Auth“, kuriame yra saugomas registruotų vartotojų sąrašas su papildoma informacija. Taip pat aprašytos funkcijos, reikalingos registruoti naujus vartotojus, keisti esamų vartotojų informaciją, patikrinti, ar konkretus vartotojas autentifikuotas ir t. t. Šiame skyrelyje bus apžvelgiama sukurto išmaniojo kontrakto „Auth“ struktūra ir kaip jame saugoma informacija.

Išmanusis kontraktas „Auth“ parašytas programavimo kalba „Solidity“, skirta kurti išmaniuosius kontraktus „Ethereum“ blokų grandinei. Kuriant išmanųjį kontraktą naudotas „Truffle Suite“

programavimo įrankių rinkinys. Šiame rinkinyje esantis įrankis „Ganache“ panaudotas sukurti lokalią blokų grandinę, o „Truffle“ – įdiegti sukurtą išmanųjį kontraktą į blokų grandinę. Taip pat „Truffle“ panaudotas rašyti ir leisti automatinius testus, siekiant užtikrinti, kad išmanusis kontraktas veiktų teisingai. Sėkmingai sukūrus reikiamą išmanųjį kontraktą, naudojant „Docker“ platformą sukurtas blokų grandinės tinklas iš dviejų mazgų ir į jį įdiegtas šis kontraktas.

Prototipui sukurtas „Ethereum“ išmanusis kontraktas „Auth“ susideda iš trijų savybių (angl. *properties*), 16-os funkcijų ir vienos savitos duomenų struktūros. Savita duomenų struktūra „Account“ atitinka vartotojo paskyrą ir susideda iš dviejų laukelių: „address“ tipo „addr“, kuriame įrašomas vartotojo adresas, ir „uint“ tipo „deletedAt“, skirto saugoti informaciją, kada vartotojo paskyra buvo ištrinta. O norint atkurti paskyrą, tereikia „deletedAt“ laukeliui priskirti pradinę reikšmę „0“. Visų registruotų vartotojų adresai saugomi bendrame masyve „accounts“, o paskyrų informacija duomenų struktūros „Account“ pavidalu saugoma masyve „accountsMap“. Į šį masyvą įrašomi naujų vartotojų duomenys ar keičiami esami duomenys, naudojant tam skirtas ir specialiai sukurtas išmaniojo kontrakto funkcijas „createAccount“, „deleteAccount“, „restoreAccount“ ir pan. Funkcija „checkAuthentication“ naudojama patikrinti, ar vartotojas turi teisę prisijungti prie paslaugos. Išmaniojo kontrakto kūrėjo adresą saugo savybė „creator“. Šiame prototipe kūrėjas yra laikomas administratoriumi. Išmaniojo kontrakto „Auth“ struktūra vaizduojama 9 paveiksle.



9 pav. Išmaniojo kontrakto „Auth“ struktūra

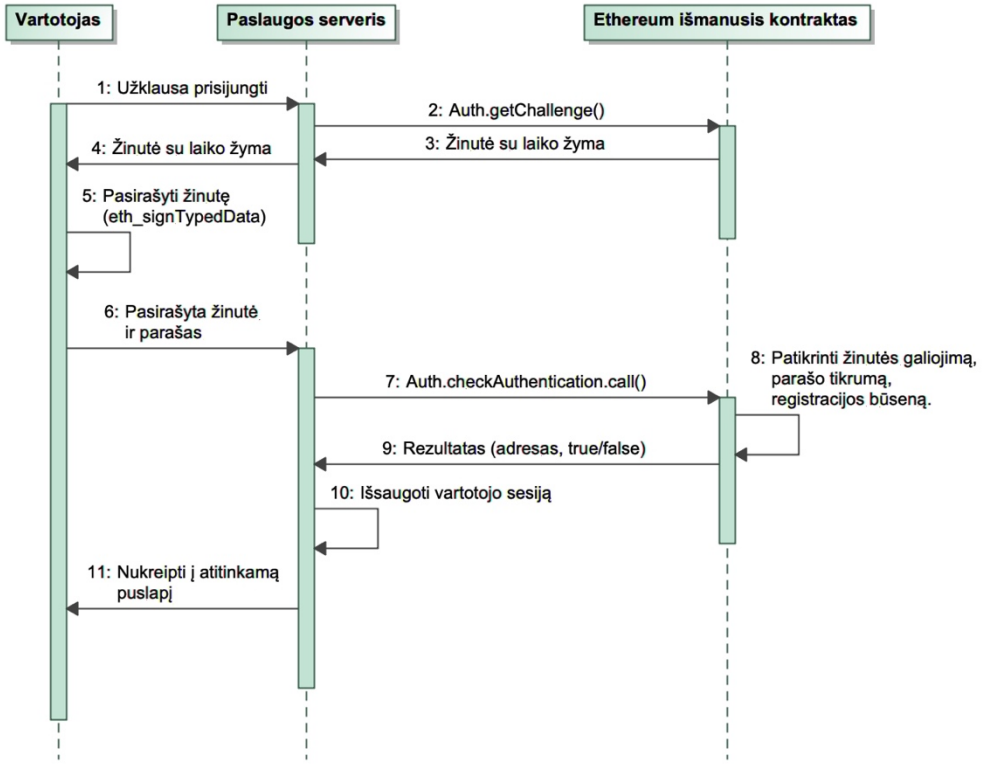
3.3. Vartotojų autentifikacija

Vartotojas, norėdamas naudotis paveikslų dalinimosi tinklalapiu, pirmiausia privalo prisijungti prie sistemos. Atėjęs į tinklalapį vartotojas nukreipiamas į prisijungimo puslapį. Kuriame pirmiausia yra prašoma vartotojo adreso: vartotojo interneto naršyklėje atsiranda „MetaMask“ langas, prašantis duoti sutikimą tinklalapiui gauti vartotojo adresą. Davus sutikimą, įrenginys siunčia užklausą paslaugos serveriui. Paslaugos serveris kreipiasi į išmanųjį kontraktą „Auth“ ir šis grąžina žinutę[5], kurią vartotojas turi pasirašyti. Žinutę paslaugos serveris persiunčia klientui. Klientui gavus žinutę, iškviečiama kodo bibliotekos „Web3“ funkcija „eth_signTypedData“, naršyklėje atsiranda kitas „MetaMask“ langas, kuriame prašoma pasirašyti šią žinutę. Pasirašius žinutę, ji, kartu su parašu, siunčiama į serverį, kuris, perduoda pasirašytą žinutę į išmanųjį kontraktą „Auth“. Kontraktas patikrina ar žinutės galiojimo laikas nepasibaigęs, taip pat patikrina parašo autentiškumą, iš jo gauna kliento

[5] Prototipe naudojama žinutė, galiojanti ribotą laiko tarpą.

adresą, jį patikrina saugomame registruotų vartotojų sąraše ir paslaugos serveriui grąžina atsakymą: koks yra iš parašo nustatytas vartotojo adresas ir ar jis autentifikuotas.

Kadangi šios operacijos metu vykdomas tik duomenų nuskaitymas iš blokų grandinės, o ne rašymas, transakcija nėra sinchronizuojama su kitais blokų grandinės tinklo mazgais. Lokalus tinklo mazgas įvertina transakciją ir pateikia rezultatą, šiuo atveju – ar vartotojas yra registruotas sistemoje, ar nėra. 10 paveiksle vaizduojama autentifikacijos proceso sekos schema.



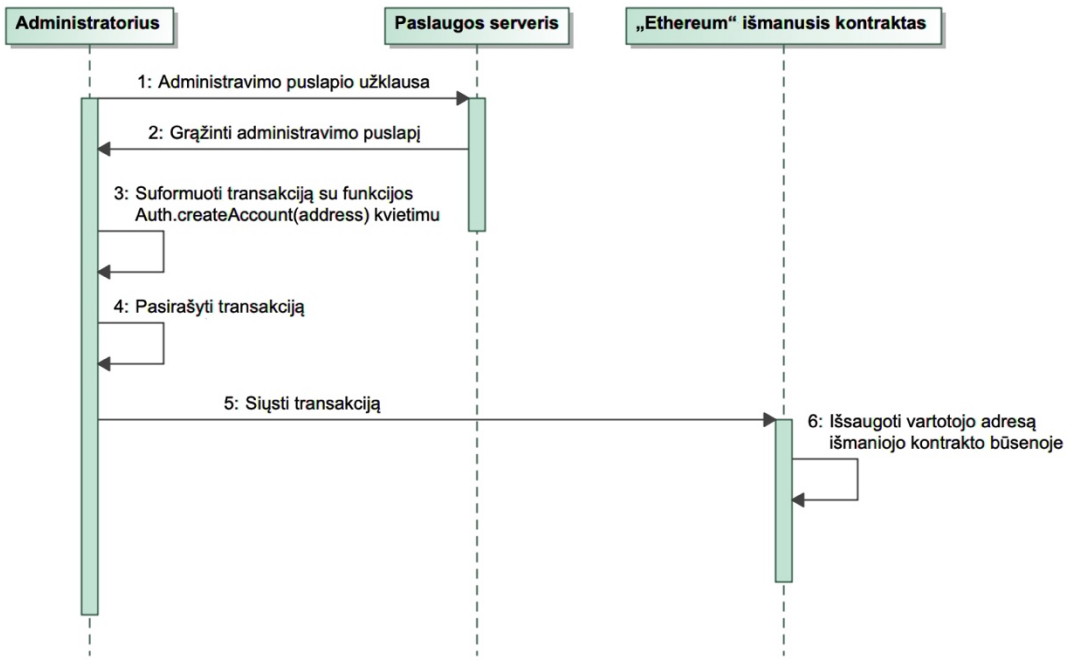
10 pav. Autentifikacijos proceso sekos schema

Po to, kai yra nustatomas vartotojo adresas ir autentifikacijos būsena, ši informacija yra išsaugoma sesijos kintamajame. Vartotojui toliau naršant po tinklalapį, pagal jo rolę yra rodomi atitinkami navigacijos punktai.

3.4. Naujo vartotojo registravimas ir registruotų vartotojų sąrašo peržiūra

Prieš vartotojui jungiantis prie paslaugos, pirmiausia jam turi būti sukuriama paskyra sistemoje. Vartotojų registravimą ir kitą administravimą vykdo specialias teises turintis administratorius.

Administratorius, prisijungęs prie paslaugos tinklalapio, gauna prieigą prie vartotojų administravimo puslapio, kuriame gali peržiūrėti jau registruotus vartotojus ir jų būsenas (paskyra aktyvi ar ištrinta) ir kurti naujas vartotojų paskyras. 11 paveiksle vaizduojama vartotojo registravimo proceso sekos schema. Naujas vartotojas registruojamas į puslapyje pateiktą adreso laukelį įvedant registruojamo vartotojo adresą ir paspaudžiant mygtuką „Kurti“. Puslapio „JavaScript“ kodas suformuoja transakciją ir į ją įtraukia įvestą adresą ir kviečiamos „Auth“ išmaniojo kontrakto funkcijos pavadinimą – „createAccount“. Po to, suformuota transakcija yra pasirašoma ir siunčiama į blokų grandinės tinklą. Tinkle transakcija sinchronizuojama ir tinklo mazgai ją įtraukia į blokų grandinę. Nuo tos akimirkos, kai transakcija yra įrašoma į blokų grandinę, vartotojas tampa registruotu ir gali jungtis prie paslaugos.

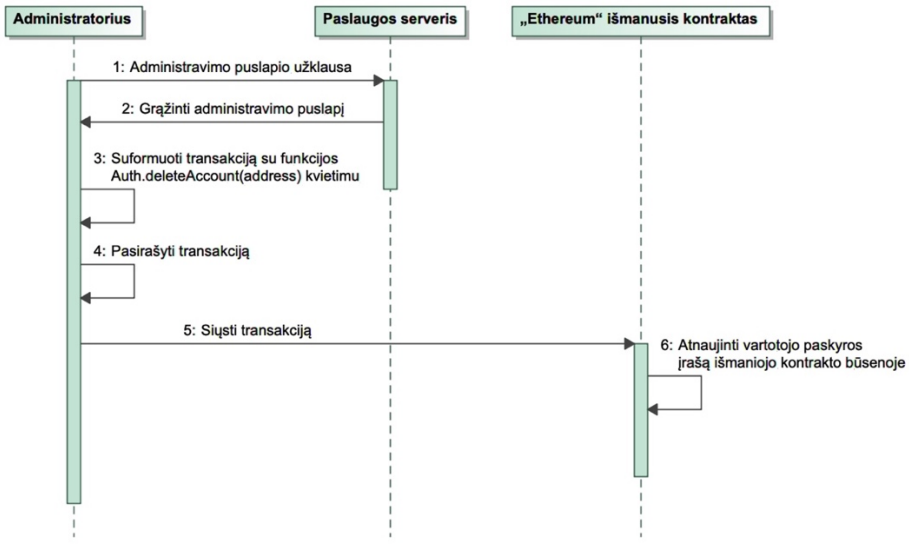


11 pav. Vartotojo registravimo proceso seka

Egzistuojančių vartotojų paskyrų peržiūra vykdoma kiek kitaip – suformuota transakcija su išmaniojo kontrakto „Auth“ funkcija „getAccounts“ yra siunčiama į lokalų blokų grandinės tinklo mazgą, kuris įvykdo nurodytą funkciją ir pateikia rezultatą. Norint gauti autentifikuotų vartotojų sąrašą, į blokų grandinę nereikia įrašyti naujų duomenų, todėl transakcija su kitais tinklo mazgais nėra sinchronizuojama ir išmaniojo kontrakto būsena nėra keičiama.

3.5. Vartotojo paskyros ištrynimas

Be jau anksčiau aptartų funkcijų, šio baigiamojo darbo prototipe administratoriui numatoma ir dar viena funkcija: trinti esamas vartotojų paskyras (žr. 12 pav.) Sukurtame prototipe ištrynus paskyrą, ji yra ne visiškai pašalinama iš išmaniojo kontrakto „Auth“ būsenos, o pažymima kaip ištrinta, į jos laukelį „deletedAt“ įrašant ištrynimo laiką „Unix“ laiko formatu.



12 pav. Vartotojo paskyros ištrynimo seka

Išmaniajame kontrakte „Auth“ paskyros laukelis „deletedAt“ yra naudojamas paliekant galimybę ištrintą paskyrą atkurti. Jeigu trinant vartotojo paskyrą, ji būtų visiškai pašalinama iš išmaniojo kontrakto būsenos, norint vartotojui vėl suteikti prieigą, jį reikėtų registruoti iš naujo. Tokiu atveju būtų prarandama informacija apie tai, kad vartotojas jau buvo registruotas. Taip pat, jeigu šį modelį naudojančios sistemos išmaniajame kontrakte būtų naudojama daugiau laukelių, t. y., saugoma daugiau informacijos vartotojų paskyrose, visa ši informacija ištrynimo metu būtų prarandama ir paskyrą kuriant iš naujo visą informaciją reikėtų įvesti pakartotinai.

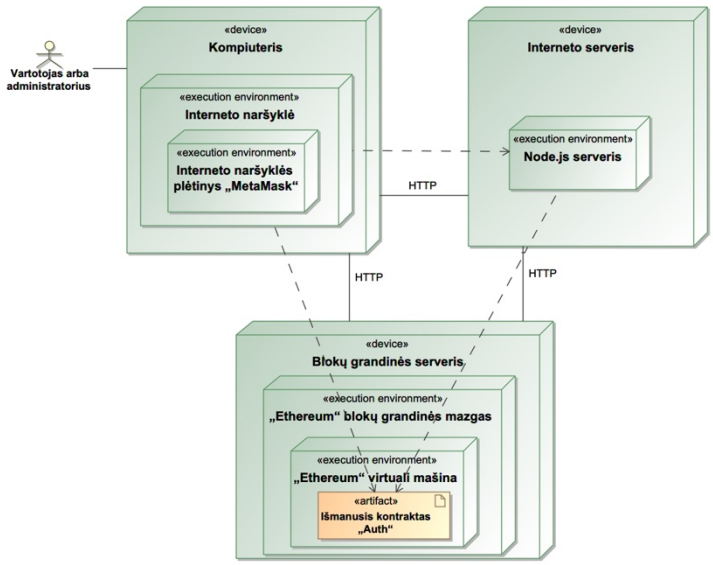
Administratorius, norėdamas ištrinti vartotojo paskyrą, pirmiausia turi ją susirasti administravimo puslapyje, kur pateikiami visi registruoti vartotojai. Prie pasirinkto vartotojo paspaudus mygtuką „Trinti“ ir palaukus kol suformuota transakcija bus įrašyta į blokų grandinę, šio vartotojo paskyra yra ištrinama.

Kaip minėta, vartotojo paskyros būsena (aktyvi ar ištrinta) valdoma paskyros laukeliu „deletedAt“. Jeigu šio laukelio reikšmė yra didesnė už 0, tuomet išmaniojo kontrakto funkcija „checkAuthentication“ grąžina neigiamą reikšmę, t. y., vartotojui prisijungti neleidžiama. Administratorius gali atkurti paskyrą vartotojų paskyrų valdymo lange suradęs reikiamos paskyros eilutę ir paspaudęs eilutėje esantį mygtuką „Atkurti“. Paspaudus mygtuką suformuojama transakcija, nurodant išmaniojo kontrakto „Auth“ funkciją „restoreAccount“ ir parametrą „vartotojo adresas“. Po to transakcija siunčiama į blokų grandinės tinklą, sinchronizuojama ir, įrašius į bloką, pridedama prie grandinės, o išmaniojo kontrakto būsenoje konkrečios paskyros laukelio „deletedAt“ reikšmė nustatomą į „0“ ir vartotojo paskyros būsena vėl tampa „aktyvi“, vartotojas vėl gali prisijungti prie paslaugos.

3.6. Sukurto „Ethereum“ išmaniojo kontrakto pritaikymas paslaugos tinklalapyje

Aprašomame prototipe blokų grandinė atlieka lokalios duomenų bazės ir parašo tikrinimo funkcijas. Į ją įrašomi ir iš jos nuskaitomi (tikrinant, ar vartotojas turi teisę prisijungti prie paslaugos) vartotojų autentifikacijos duomenys, kriptografiškai patikrinamas parašas. Tačiau blokų grandinės suteikia dar daugiau privalumų, kai naudojamas blokų grandinės tinklas. Sujungus du ar daugiau blokų grandinės mazgų, šie sudaro paskirstytą tinklą ir nuolatos sinchronizuoja blokų grandinę tarpusavyje, užtikrindami, kad visuose mazguose esanti informacija yra vienoda.

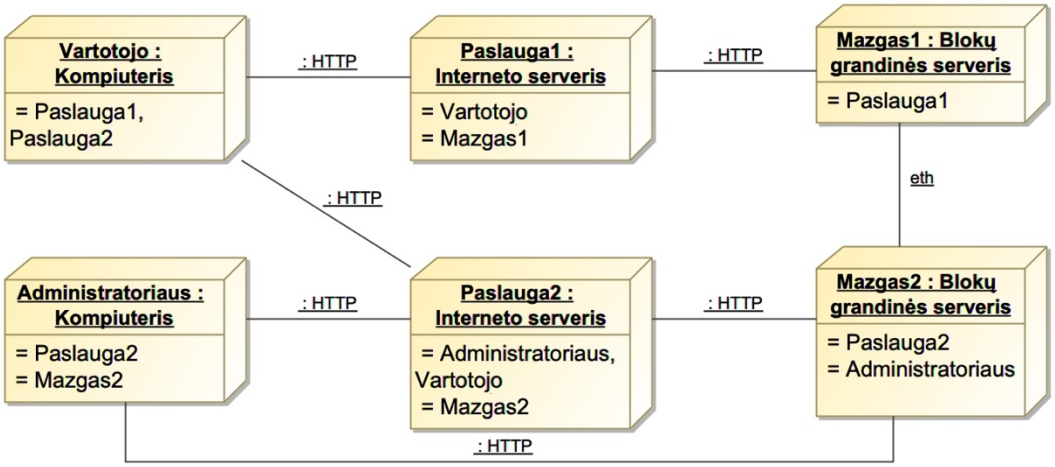
Anksčiau šiame skyriuje išmaniojo kontrakto kūrimas ir pritaikymas buvo vykdomas viename „Ethereum“ blokų grandinės mazge, naudojant blokų grandinių kūrimo įrankį „Ganache“. Šis įrankis yra skirtas naudotis tik išmaniųjų kontraktų kūrimo ir testavimo tikslais lokalioje aplinkoje. Naudojant sukurtą modelį su išmaniuoju kontraktu „Auth“ realioje (angl. *production*) aplinkoje, blokų grandinės tinklą gali sudaryti keli ir daugiau mazgų. Pavyzdžiui, prie tinklo gali jungtis daug paslaugų tiekėjų, kurie gali palaikyti atskirus šio tinklo mazgus. 13 paveiksle vaizduojama sistemos diegimo diagrama su nurodytais įrenginiais, vykdymo aplinkomis, priklausomybėmis ir komunikavimo būdais.



13 pav. Sistemos diegimas: įrenginiai, vykdymo aplinkos, priklausomybės ir komunikavimo būdai

13 paveiksle vaizduojamą blokų grandinės serverį gali turėti ir palaikyti kiekvienas paslaugos tiekėjas arba vienas tiekėjas gali naudotis kito tiekėjo serveriu.

Kaip minėta anksčiau, realioje aplinkoje bus daugiau nei vienas blokų grandinės serveris / mazgas, todėl verta panagrinėti kaip atrodytų tokios sistemos diegimo diagrama. 14 paveiksle vaizduojama diegimo schema su konkrečiais 13 paveiksle vaizduojamų komponentų vienetais.



14 pav. Sistemos diegimo diagrama vaizduojant konkrečius komponentų vienetus.

Šioje pavyzdinėje diegimo schemoje vaizduojamos dvi atskiros paslaugos, vartotojų autentifikacijai naudojančios tą patį blokų grandinės tinklą, tačiau skirtingus tinklo mazgus. Pirmiausia, administratorius, naudodamasis antrosios paslaugos (Paslauga2) pagalbiniu administravimo puslapiu, sukuria vartotojo paskyrą. Kadangi antrosios paslaugos serveris administratoriui nurodo kaip pasiekti antrąjį blokų grandinės tinklo mazgą (Mazgas2), vartotojo paskyros sukūrimo transakcija ir yra siunčiama į šį mazgą. Tada transakcija sinchronizuojama blokų grandinės tinkle, galiausiai įtraukiama į blokų grandinę ir Mazgas1 taip pat gauna ką tik užregistruoto vartotojo autentifikacijos informaciją.

Kai autentifikacijos informacija yra įrašoma į blokų grandinę, vartotojas gali jungtis prie bet kurios šį blokų grandinės tinklą naudojančios paslaugos. Pavyzdžiui, kaip vaizduojama 14 paveiksle, vartotojas gali sėkmingai prisijungti tiek prie paslaugos „Paslauga1“, tiek prie „Paslauga2“.

Automatinė blokų grandinės sinchronizacija tarp tinklo mazgų užtikrina, kad bet kuri paslauga, naudojanti šį tinklą, galės sėkmingai autentifikuoti registruotus vartotojus. Tokiu būdu užtikrinamas dalinis vieningo prisijungimo technologijos funkcionalumas – vartotoją administratorius gali užregistruoti vieną kartą, o vartotojas gali jungtis prie visų paslaugų, naudojančių tą patį išmanųjį kontraktą, net ir jungiantis prie skirtingų blokų grandinės mazgų. Visgi, tai nėra pilnas SSO funkcionalumas, nes šiame modelyje „Ethereum“ išmanusis kontraktas nesaugo vartotojų sesijos informacijos, todėl vartotojui jungiantis prie kiekvienos paslaugos, reikia pasirašyti naują žinutę.

3.7. Sukurto prototipo tyrimas

Teorinio modelio įgyvendinimo praktiškai metu, sukurta veikianti vartotojų autentifikavimo sistema, naudojanti „Ethereum“ išmanųjį kontraktą, ir įdiegta į dvi paslaugų svetaines, vartotojų autentifikacijai naudojančias skirtingus blokų grandinės mazgus. Registruotas vartotojas, naudodamasis „MetaMask“ plėtiniu, gali sėkmingai prisijungti prie bet kurios paslaugos svetainės. O administratoriui nereikia registruoti vartotojo atskirai kiekvienoje svetainėje – užtenka užregistruoti vieną kartą ir vartotojo paskyros informacija yra automatiškai sinchronizuojama visuose „Ethereum“ blokų grandinės mazguose. Siekiant įvertinti vartotojų potyrį jungiantis prie paslaugos, aptarnaujančios daug vartotojų, nuspręsta išmatuoti, kiek laiko išmaniajam kontraktui užtrunka patikrinti vartotojo parašą ir registracijos būseną, priklausomai nuo sistemoje registruotų vartotojų skaičiaus.

Vartotojui jungiantis prie sukurtos svetainės, jis pasirašo išmaniojo kontrakto suformuotą žinutę ir parašą siunčia patikrinti paslaugos serveriui. Kadangi parašo tikrumo ir vartotojo būsenos patikrą atlieka išmanusis kontraktas, o ne paslaugos serveris, nuspręsta matuoti laiką nuo parašo ir žinutės išsiuntimo į išmanųjį kontraktą iki atsakymo, ar vartotojas autentifikuotas, gavimo. Tokiu būdu tiriama būtent išmaniojo kontrakto, kuris atlieka esmines autentifikavimo proceso funkcijas, sparta ir ribojama tyrimo aplinkos įtaka rezultatams.

Vartotojų autentifikavimo tyrimui atlikti pirmiausia buvo sugeneruota 100000 „Ethereum“ paskyrų ir jų privatūs raktai ir adresai išsaugoti lokaliame faile. Naudojant iš anksto paruoštą failą su paskyrų duomenimis, tyrimas vyksta greičiau. Autentifikacijos spartai matuoti sukurta „JavaScript“ funkcija, kuri pagal nustatytus tyrimo parametrus iš anksčiau sukurto paskyrų duomenų failo užregistruoja reikiamą kiekį paskyrų „Ethereum“ išmaniajame kontrakte, parenka atsitiktinę paskyrą ir jos privačiuoju raktu pasirašo žinutę. Galiausiai sukurta tyrimo funkcija išmatuoja, kiek trunka kreipimasis į išmaniojo kontrakto funkciją, kuri patikrina parašą ir vartotojo registracijos būseną. Matavimai atliekami nustatytu intervalu didinant registruotų vartotojų kiekį iki nustatyto galutinio kiekio. Tyrimo rezultatai išvedami į ekraną ir įrašomi į failą, kuris vėliau naudojamas rezultatų vertinimui.

Siekiant objektyviai įvertini gautus rezultatus, taip pat buvo atliktas eksperimentas, kai paslaugos vartotojų autentifikavimui naudojama lokali „MySQL“ duomenų bazė, dažnai naudojama interneto svetainėse. Šiam tyrimui sukurta kita funkcija, kuri veikia panašiai kaip pirmoji – sukuriamas reikiamas kiekis paskyrų, parenkama atsitiktinė, jos privačiuoju raktu pasirašoma žinutė ir galiausiai matuojama, kiek laiko užtrunka patikrinti žinutės galiojimą, parašo tikrumą ir registracijos būseną. Ši funkcija matuoja, kiek „Node.js“ aplinkoje ir „MySQL“ duomenų bazėje užtrunka tokios pačios operacijos kaip sukurtame „Ethereum“ išmaniajame kontrakte.

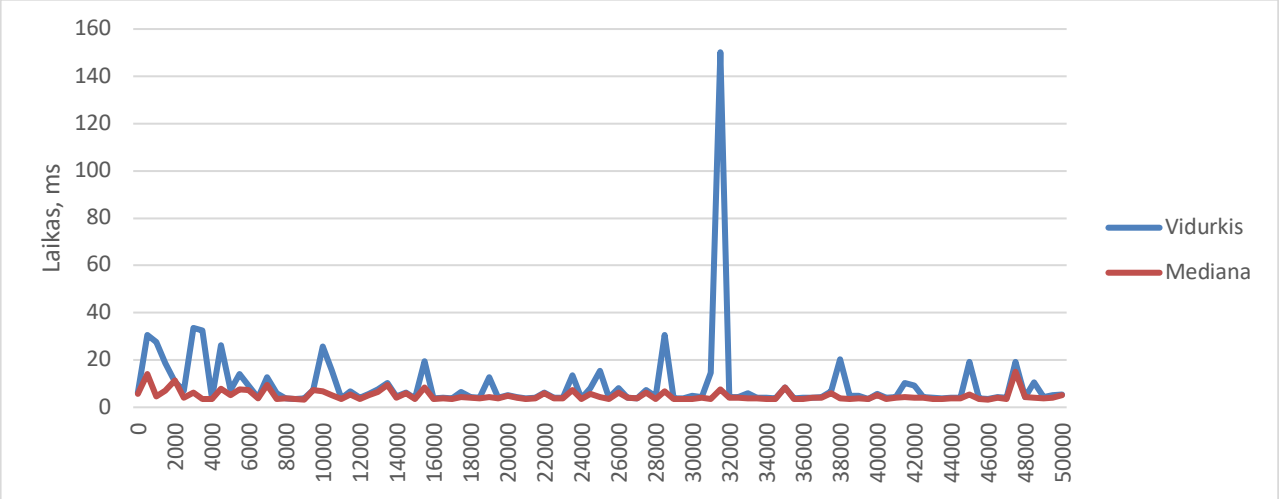
Tyrimas atliktas naudojant šią techninę ir programinę įrangą:

- 2015 m. laidos MacBook Pro nešiojamasis kompiuteris su 2,2 GHz Intel Core i7 procesoriumi, 16 GB 1600 MHz DDR3 atminties;

- macOS Mojave v10.14.4 operacinė sistema;
- Node v10.15.3 programavimo aplinka;
- Truffle v5.0.6;
- Solidity v0.4.24;
- Docker v18.09.2;
- Docker Compose v1.23.2;
- web3.js v1.0.0-beta.37;
- truffle-contract v4.0.10;
- ethereumjs-util v6.1.0;
- ethereumjs-wallet v0.6.3;
- AdonisJs v5.0.13;
- MySQL v5.7.

Tiriama sistema buvo paleista naudojant „Docker“ platformą. Sudarytas dviejų mazgų „Ethereum“ blokų grandinės tinklas, į jį įdiegtas sukurtas išmanusis kontraktas „Auth“. Tyrimo metu visos operacijos buvo atliekamos jungiantis prie vieno iš mazgų. Tiriant autentifikaciją su „Node.js“ ir „MySQL“, ši aplinka taip pat sukurta naudojant „Docker“ platformą. Tyrimo metu stengtasi, kad kiti kompiuterio procesai darytų kuo mažesnę įtaką rezultatams, dėl to buvo išjungtos jo metu nereikalingos programos. Taip pat, kad išmaniajame kontrakte būtų sukurtos paskyros, reikia bent viename iš mazgų įjungti kasimo režimą, t. y., kad būtų kuriami nauji blokai ir informacija rašoma į blokų grandinę. Tuo metu labai apkraunami kompiuterio procesoriai, todėl siekiant, kad kasimas darytų kuo mažesnę įtaką tyrimo rezultatams, sukūrus matavimui reikiamą paskyrų kiekį, kasimas buvo sustabdomas, palaukiama 10 sekundžių, kad nukristų procesorių apkrovimas ir tada atliekami matavimai. Prieš kuriant naujas paskyras, reikalingas kitam matavimui, kasimas vėl buvo įjungiamas ir šis ciklas kartojamas, kol buvo atlikti visi matavimai.

Tiriant prototipą buvo atliekami autentifikacijos spartos matavimai, kai sistemoje registruota nuo 0 iki 50000 vartotojų. Matavimai atlikti su 500 paskyrų intervalu, kiekvieną matavimą kartojant 10 kartų. 1 priede pateikti tyrimo rezultatai vartotojus autentifikuojant „Ethereum“ išmaniuoju kontraktu, o 2 priede – „JavaScript“ serveriu „Node.js“ ir duomenų baze „MySQL“. 15 paveiksle grafiku pateikiami vartotojų autentifikacijos prototipe sukurtu „Ethereum“ išmaniuoju kontraktu rezultatai.

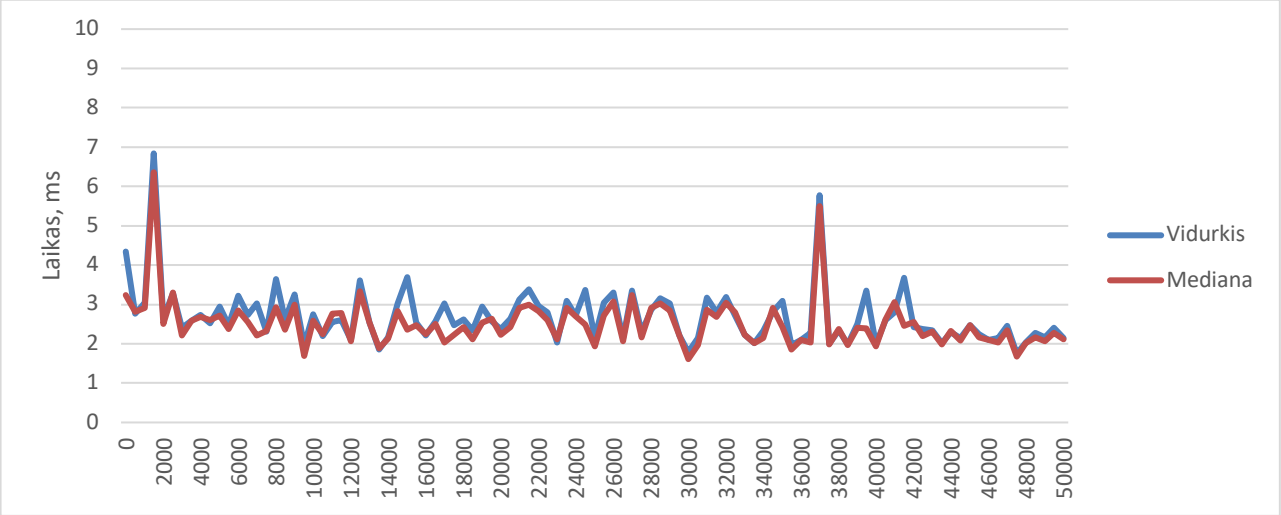


15 paveikslas. Vartotojų autentifikavimo „Ethereum“ išmaniuoju kontraktu tyrimo rezultatai.

Kaip matoma grafike, ties maždaug 500, 3000, 28500, 31500 registruotų paskyrų, vidutinė autentifikavimo trukmė (*Vidurkis*) yra gerokai didesnė, negu kitais atvejais. Detaliau paanalizavus pradinius matavimų rezultatus (1 priedas), pastebėta, kad kai kuriose matavimų grupėse (t. y., esant tam tikram registruotų vartotojų skaičiui) viena ar kelios reikšmės yra kelis šimtus ar kelis tūkstančius procentų didesnės negu dauguma kitų tos pačios grupės matavimų. Tyrimas atliktas asmeniniame kompiuteryje, todėl tikėtina, kad kai kurių matavimų metu procesorius buvo labiau užimtas ir užklausa į blokų grandinės mazge esantį išmanųjį kontraktą užtruko ilgiau. Kadangi ši užklausa užtrunka vos kelias milisekundes, net trumpas vėlinimas gerokai iškreipia rezultatus.

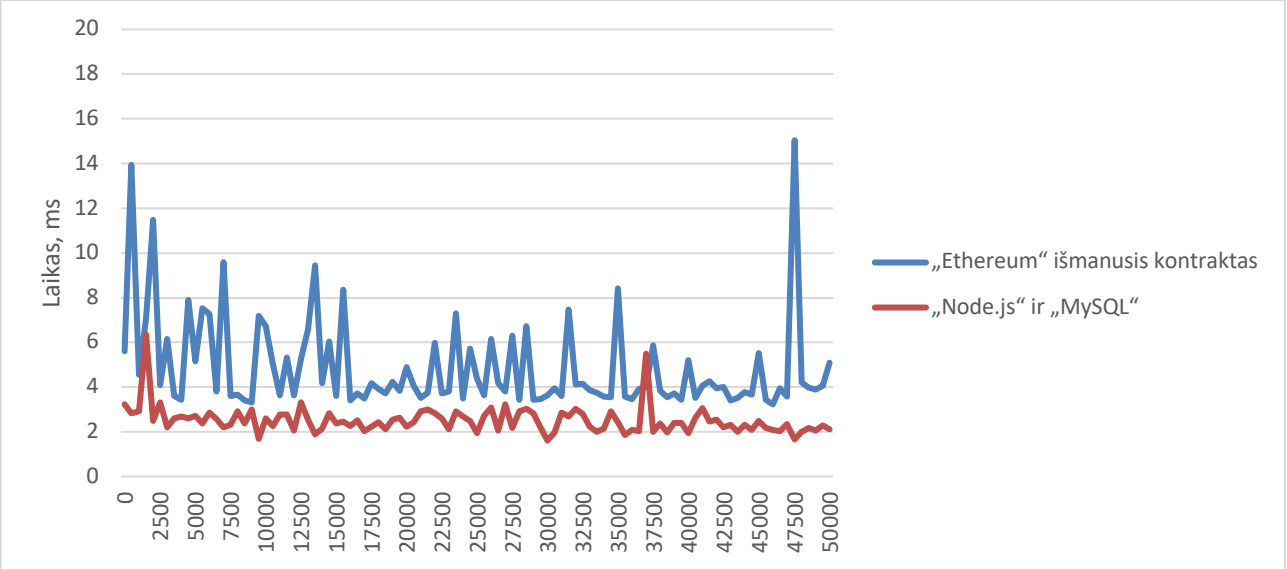
Siekiant tiksliau įvertinti matavimus ir sumažinti nuokrypių įtaką, taip pat suskaičiuotos ir rezultatų medianos. 15 paveiksle vaizduojamas medianų grafikas yra kur kas nuoseklesnis. Vertinant bendrą tendenciją, didinant registruotų vartotojų skaičių nuo 0 iki 50000 pastebima nekintanti autentifikavimo operacijos trukmė. Tai yra, vartotojų kriptografinių parašų ir registracijos būsenų tikrinimo „Ethereum“ išmaniajame kontrakte trukmė nepriklauso nuo registruotų vartotojų skaičiaus ir yra apie 4,88 ms.

Kaip minėta, autentifikavimo išmaniuoju kontraktu tyrimo rezultatai buvo palyginti su autentifikavimo naudojant „Node.js“ ir „MySQL“ tyrimo rezultatais. 16 paveiksle matomas šių rezultatų grafikas.



16 pav. Vartotojų autentifikavimo „Node.js“ aplinkoje ir „MySQL“ duomenų bazėje tyrimo rezultatai.

Vartotojų autentifikacija „Node.js“ aplinkoje ir „MySQL“ duomenų bazėje vidutiniškai užtrunka 2,51 ms. 17 paveiksle vaizduojamame grafike lyginamos abiejų tyrimų rezultatų medianos.



17 pav. Vartotojų autentifikavimo „Node.js“ aplinkoje ir „MySQL“ duomenų bazėje trukmės palyginimas su autentifikavimu „Ethereum“ išmaniuoju kontraktu.

Lyginant šių tyrimų rezultatus, matoma, kad vartotojų parašo ir registracijos būsenos tikrinimas naudojant „Node.js“ ir „MySQL“ yra beveik du kartus spartesnis nei tikrinimas naudojant sukurtą „Ethereum“ išmanųjį kontraktą. Tačiau iš vartotojo perspektyvos šis skirtumas yra nereikšmingas, nes kelių milisekundžių pokytis žmonėms nėra pastebimas.

Kadangi tyrimo objektas yra išmanusis kontraktas, tyrimo planuotas taip, kad rezultatai būtų kuo mažiau priklausomi nuo kodo vykdymo aplinkos ar programos, kuri naudoja šį išmanųjį kontraktą. Šis tyrimas atliktas „Node.js“ serverio aplinkoje, nes ji yra naudojama prototipe svetainei kurti, o autentifikuojant vartotoją prie blokų grandinės jungiamasi naudojant kodo bibliotekas „Web3.js“ ir „truffle-contract“. Todėl galima teigti, kad kuriant šiomis technologijomis paremtą autentifikavimo sistemą ir naudojant prototipo kūrimo metu sukurtą išmanųjį kontraktą, šio tyrimo rezultatai atspindi tikėtiną tokios sistemos veikimo spartą.

IŠVADOS

1. Teorinėje darbo dalyje buvo aptartas identiteto valdymas, įvardytos dažniausios blokų grandinių taikymo sritys, išanalizuoti praktikoje taikomi autentifikacijos metodai. Autentifikacija gali būti atliekama įvairiais metodais: suvedus identifikatorių ir slaptažodį, PIN kodą, pasinaudojus elektronine kortele, mobiliuoju parašu, nuskaičius biometrines vartotojo savybes, kriptografiškai ir t. t. Nors blokų grandinių idėja dar 1991 metais Stuart'o Haber'io ir W. Scott'o Stornetto buvo pasiūlyta kaip sprendimas saugiai elektroninių išteklių apskaitai, tačiau tik šiuo metu pastebima tendencija, kad sparčiai populiarėja blokų grandinių technologijos. Kuriant įvairias blokų grandinėmis grįstas sistemas, dažnai renkamasi naudoti išmaniuosius kontraktus – paskirstytai blokų grandinėse leidžiamas programas (pavyzdžiui, „Ethereum“, „HyperLedger Burrow“ ar pan.) Išmaniųjų kontraktų veikimo principas grįstas konsensuso algoritmais, t. y., tas pačias užklausas vykdo visi (arba daugiau nei vienas) tinklo mazgai, kurie lygiagrečiai apskaičiuoja užklausos rezultatą, todėl tokie rezultatai yra patikimi. Šie kontraktai patrauklūs sistemų kūrėjams, nes leidžia lanksčiai kurti sistemas, pritaikant jas vartotojų poreikiams. Šiuo metu blokų grandinėmis paremtos sistemos yra naudojamos didelio saugumo reikalaujančiose srityse, pavyzdžiui, kuriant sistemas bankams, draudimui, medicinos įstaigoms ar valstybinėms institucijoms.
2. Blokų grandinės paremtos duomenų jungimu į grandinę, taip užtikrinant joje saugomų įrašų nepakeičiamumą. Blokų grandinių tinklai naudoja konsensuso algoritmus ir taip nusprendžia, kokia informacija bus įrašyta į blokų grandinę ir apsaugo nuo neteisėto blokų grandinėje saugomos informacijos pakeitimo. Analizuoti praktiniai blokų grandinės tinklų pritaikymo atvejai rodo, kad ši technologija gali būti lanksčiai taikoma kuriant itin didelio saugumo reikalaujančias sistemas. Darbe aptarti tokių sistemų pritaikymo atvejai: „uPort“, „Hydro Raindrop“. Abiem šiais atvejais yra naudojami „Ethereum“ išmanieji kontraktai, tačiau skiriasi šių sistemų funkcionalumas. Pagrindiniai „uPort“ privalumai yra vartotojo duomenų šifravimas, užtikrinant, kad prieiga prie šių duomenų bus suteikiama tik vartotojui leidus; vartotojo paskyros ir jo fizinės tapatybės susiejimas. Priklausomai nuo „Hydro Raindrop“ konfigūracijos, ši sistema arba leidžia stebėti prisijungimus prie sistemos ir reaguoti į grėsmes realiu laiku, vykdyti autentifikacijos apskaitą, arba reikalauja mažiau tinklo išteklių, nes prisijungimo metu informacija nėra rašoma į blokų grandinę. Tačiau pažymėtina, kad tarp šiuo metu rinkoje egzistuojančių autentifikavimo sprendimų galima pastebėti ir trūkumų: autentifikacijos metu informacija rašoma į blokų grandinę, todėl prisijungimo procesas ilgiau užtrunka, labiau apkraunamas tinklas, o vartotojai privalo turėti tinklo valiutos; nenaudojami vartotojo pasirašomos žinutės kontrolės mechanizmai, todėl pasirašytą žinutę gali pavogti piktavaliai ir taip apsimesti vartotoju.
3. Darbe siūlomas teorinis autentifikacijos modelis sukurtas remiantis esamais blokų grandinėmis paremtais autentifikacijos sprendimais. Kuriant modelį buvo atsižvelgta į pastebėtus jau egzistuojančių modelių trūkumus ir stengtasi juos pašalinti. Prisijungimo procesas pagreitintas, išspręsta tinklo valiutos turėjimo problema ir mažiau apkraunamas blokų grandinės tinklas, nes siūlomame teoriniame modelyje autentifikacijos metu informacija nėra įrašoma į blokų grandinę, o vietoje to vartotojai

savo įrenginyje pasirašo „Ethereum“ išmaniojo kontrakto suformuotą žinutę ir šį parašą patikrina išmanusis kontraktas. Aptariant teorinį modelį, pasiūlyti du sprendimo būdai, kaip galima geriau apsaugoti vartotojų duomenis nuo piktavalių: unikali vartotojui suformuota žinutė gali galioti tam tikrą kiekį kartų arba nustatytą laiko intervalą.

4. Teorinį autentifikacijos modelį pritaikant praktikoje, buvo sukurtas „Ethereum“ išmanusis kontraktas, skirtas valdyti vartotojų autentifikacijos informacijai. Šis kontraktas buvo įdiegtas į sukurtą blokų grandinės tinklą. Galutinis rezultatas įdiegtas darbo metu sukurtoje interneto svetainėje. Pagal technologijas, reikalingas autentifikacijos modeliui sukurti, jį galima skirstyti į keturias dalis: kliento įrenginys, paslaugos interneto serveris, paslaugos duomenų bazės serveris, blokų grandinės tinklas. Klientas savo išmaniajame įrenginyje prie sistemos jungiasi naudodamas „Google Chrome“ (ar ja paremtą kitą) interneto naršyklę ir joje įdiegtą „MetaMask“ plėtinį. Paslaugos interneto serverio kūrimui panaudotas „JavaScript“ serveris „Node.js“, kodo karkasas „AdonisJs“ (interneto svetainės kūrimui) ir biblioteka „Web3.js“ (jungtis prie blokų grandinės tinklo). Sukurtame modelyje naudojama „MySQL“ duomenų bazė, kurioje saugomos vartotojų prieigos prie išteklių teisės. „Ethereum“ blokų grandinėje įdiegtas išmanusis kontraktas, parašytas „Solidity“ kalba, o diegimui į tinklą naudotas įrankis „Truffle“. Blokų grandinės tinklo mazgai sukurti naudojant „Docker“ platformą. Pritaikius visas išvardintas technologijas buvo pasiektas šio baigiamojo darbo tikslas – sukurta blokų grandinėmis paremta vartotojų autentifikacijos tinklo paslaugose sistema.
5. Sukurtas prototipas buvo ištirtas, įvertinta, kiek laiko užtrunka pagrindinis autentifikacijos proceso etapas – vartotojo kriptografinio parašo ir registracijos būsenos patikrinimas – naudojant sukurtą „Ethereum“ išmanųjį kontraktą, priklausomai nuo registruotų vartotojų skaičiaus. Palyginimui taip pat buvo atliktas toks pats tyrimas vietoje išmaniojo kontrakto naudojant „Node.js“ aplinką ir „MySQL“ duomenų bazę. Tyrimo metu autentifikacijos sparta buvo matuojama sistemoje esant nuo 0 iki 50000 registruotų vartotojų, matavimus atliekant su 500 registruotų vartotojų intervalu, po 10 kartų. Išanalizavus gautus rezultatus, nustatyta, kad vartotojų autentifikacijos informacijos tikrinimas „Ethereum“ išmaniajame kontrakte trunka apie 4,88 ms, o naudojant „Node.js“ ir „MySQL“ – 2,51 ms. Tai yra, pirmuoju atveju tikrinimas užtrunka beveik du kartus ilgiau, tačiau iš vartotojų perspektyvos kelių milisekundžių skirtumas nėra pastebimas, todėl galima teigti, kad autentifikavimo „Ethereum“ išmaniuoju kontraktu sparta prilygsta tradiciškam autentifikavimo metodui, kaip duomenų saugyklą naudojant „MySQL“ duomenų bazę.
6. Sukurta autentifikacijos sistema pasižymi atsparumu tinklo sutrikimams – blokų grandinės tinklas yra sudarytas iš keleto (plečiant sistemą būtų galima ją sudaryti iš keliolikos, keliasdešimt ir pan.) mazgų, todėl sutrikus vieno mazgo veiklai sistema galima toliau sėkmingai naudotis, o tinkle saugoma informacija nėra prarandama. Prototipe yra naudojamas konsensuso algoritmas, kuris užtikrina, kad praktiškai nebūtų įmanoma suklastoti blokų grandinėje saugomų autentifikacijos duomenų. Pažymėtina, kad sukurtą modelį dar galima tobulinti, pavyzdžiui, autentifikacijos sistemą pritaikyti ir kitų interneto naršyklių vartotojams (o ne tik „Google Chrome“); sistemą kurti naudojant privatų uždarąjį blokų grandinės tinklą (pavyzdžiui, naudojant

„HyperLedger Burrow“, o ne viešą atvirąjį „Ethereum“). Taip pat nereikėtų pamiršti, kad ypač internetinės technologijos kinta labai sparčiai, todėl galima drąsiai teigti, jog ateityje atsiras naujų iššūkių, susijusių su vartotojų autentifikacija.

LITERATŪRA

1. KABAY, M.E. *Identification, Authentication and Authorization on the World Wide Web.* 1997.

2. VRBANEC, T. ir HUTINSKI, Ž. *Data protection: identifications and authentication in applications and protocols.* 2011.

3. BOSWORTH, Seymour, KABAY, M. E. ir WHYNE, Eric. *Computer Security Handbook, Set.* Somerset: Wiley, 2009. ISBN 9780471716525.

4. VENČKAUSKAS, Algimantas ir KAZANAVIČIUS, Egidijus. *Informacinių technologijų saugos metodai.* Vilnius: UAB „TEV“, 2011.

5. TODOROV, Dobromir. *Mechanics of User Identification and Authentication Fundamentals of Identity Management.* Auerbach Publications, 2007.

6. METZ, Chris. AAA protocols: authentication, authorization, and accounting for the Internet. *IEEE Internet Computing*. 1999, vol. 3, nr. 6, pp. 75-79. ISSN 1089-7801.

7. VENČKAUSKAS, Algimantas ir TOLDINAS, Jevgenijus. *Kompiuterių ir operacinių sistemų sauga.* Kaunas: UAB „Vitae Litera“, 2012. ISBN 9786090203613.

8. O'GORMAN, L. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*. 2003, vol. 91, nr. 12, pp. 2021-2040. ISSN 0018-9219.

9. *Mobilusis elektroninis parašas* [interaktyvus]. [žiūrėta: 2019 04 19] Prieiga per: https://www.telia.lt/m-parasas.

10. ZHANG, David. *Biometric Solutions.* Boston, MA: Springer US, 2002. ISBN 9781461510536.

11. WATKINS, Steve G. *An Introduction to Information Security and ISO27001.* IT Governance Publishing, 2008. ISBN 1905356684.

12. JONES, Laurie, ANTÓN, Annie ir EARP, Julia. Towards understanding user perceptions of authentication technologies. *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*. Oct 29, 2007, pp. 91-98.

13. PANDYA, Dwiti, RAM, Khushboo, THAKKAR, Sneha, MADHEKAR, Tanvi ir THAKARE, B. S. An Overview of Various Authentication Methods and Protocols. *International Journal of Computer Applications*. 2015, vol. 131, nr. 9, pp. 25-27. ISSN 0975-8887.

14. GOLLMANN, Dieter. *Computer Security.* Wiley, 2011.

15. MORRIS, Robert ir THOMPSON, Ken. Password security: a case history. *Communications of the ACM*. 1979, vol. 22, nr. 11, pp. 594-597. ISSN 0001-0782.

16. MIRANTE, Dennis ir CAPPOS, Justin. Understanding Password Database Compromises. *Https://Isis.Poly.Edu/~jcappos/Papers/Tr-Cse-2013-02.Pdf*. 2013.

17. ARKILLS, Brian. *LDAP Directories Explained: An Introduction and Analysis.* 1-oji laida. United States: Addison-Wesley Professional, 2003. ISBN 9780201787924.

18. *OWASP Password Storage Cheat Sheet* [interaktyvus]. 2019 [žiūrėta: 2019 01 12]. Prieiga per: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Password_Storage_Cheat_Sheet.md.

19. *81% of Company Data Breaches Due to Poor Passwords | TraceSecurity* [interaktyvus]. 2018 [žiūrėta: 2019 02 03]. Prieiga per: https://www.tracesecurity.com/blog/articles/81-of-company-data-breaches-due-to-poor-passwords.

20. Yubico. Yubico's 2019 State of Password and Authentication Security Behaviors Report. *Business Wire*. 2019.

21. WANG, Chun, JAN, Steve, HU, Hang, BOSSART, Douglas and WANG, Gang. The Next Domino to Fall: Empirical Analysis of User Passwords
across Online Services. *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. 2018 03 13, pp. 196-203.

22. MAYES, Keith and MARKANTONAKIS, Konstantinos. *Smart Cards, Tokens, Security and Applications.* New York, NY: Springer, 2007. ISBN 0387721975.

23. *How Many People Have Phones Worldwide? (2019 Data)* [interaktyvus]. [žiūrėta: 2019 04 17]. Prieiga per: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world.

24. *Mobilusis parašas* [interaktyvus]. [žiūrėta: 2019 05 07]. Prieiga per: https://www.bite.lt/privatiems/paslaugos/elektroninis-parasas.

25. SKERSYS, Gintaras. *Informacijos sauga.* Vilnius: UAB „TEV“, 2011. ISBN 978-609-433-077-3.

26. CAKIR, E. *Single Sign-On. Risks and Opportunities of Using SSO (Single Sign-On) in a Complex System Environment with Focus on Overall Security Aspects.* Linnaeus University, 2013.

27. PELTIER, Thomas R., PELTIER, Justin ir BLACKLEY, John A. *Information security fundamentals.* Boca Raton [u.a.]: Auerbach, 2005. ISBN 9780849319570.

28. *Keycloak* [interaktyvus]. [žiūrėta: 2019 04 07]. Prieiga per: https://www.keycloak.org.

29. *Microsoft Account Single Sign-On (SSO)* [interaktyvus]. [žiūrėta: 2019 05 07]. Prieiga per: https://www.onelogin.com/connector/microsoft-account.

30. RAGOUZIS, N., HUGHES, J., PHILPOTT, R. ir MALER, E. *Security Assertion Markup Language (SAML) V2.0 Technical Overview.* , 2006.

31. SANDHU, S.S. *Single Sign On Concepts & Protocols.* 2004. Prieiga per: https://www.sans.org/reading-room/whitepapers/authentication/single-sign-concepts-protocols-1352.

32. HOWES, Timothy A., SMITH, Mark C. ir GOOD, Gordon S. *Understanding and Deploying LDAP Directory Services.* Addison-Wesley Professional, 2003. ISBN 9780672323164.

33. GARMAN, Jason. *Kerberos: The Definitive Guide.* O'Reilly, 2003. ISBN 0-596-00403-6.

34. CHOUDHURY, Suranjan, BHATNAGAR, Kartik ir HAQUE, Wasim. *Public Key Infrastructure Implementation and Design.* 1st-oji laida. New York, NY, USA: John Wiley & Sons, Inc, 2002. ISBN 0764548794.

35. WEAVER, Alfred C. Secure Sockets Layer. *Computer*. 2006, vol. 39, nr. 4, pp. 88-90. ISSN 0018-9162.

36. *HTTPS encryption on the web – Google Transparency Report* [interaktyvus]. [žiūrėta: 2019 05 12]. Prieiga per: https://transparencyreport.google.com/https/overview?hl=en.

37. *Security Notice* [interaktyvus]. [žiūrėta: 2019 05 12]. Prieiga per: https://www.zomato.com/blog/security-notice.

38. *Password security alert – 8tracks blog* [interaktyvus]. [žiūrėta: 2019 05 12]. Prieiga per: https://blog.8tracks.com/2017/06/27/password-security-alert/.

39. *Security Issue February 2019: FAQ – 500px Support Center* [interaktyvus]. [žiūrėta: 2019 05 07]. Prieiga per: https://support.500px.com/hc/en-us/articles/360017752493-Security-Issue-February-2019-FAQ.

40. *XSplit Password Reset Alert | XSplit Blog* [interaktyvus]. [žiūrėta: 2019 05 07]. Prieiga per: https://www.xsplit.com/blog/xsplit-password-reset-alert.

41. *2016 Data Security Incident | Uber Newsroom* [interaktyvus]. [žiūrėta: 2019 05 07]. Prieiga per: https://www.uber.com/newsroom/2016-data-incident/.

42. *Have I Been Pwned: Pwned websites* [interaktyvus]. [žiūrėta: 2019 05 07]. Prieiga per: https://haveibeenpwned.com/PwnedWebsites.

43. MIRKOVIC, Jelena ir REIHER, Peter. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*. 2004, vol. 34, nr. 2, pp. 39-53. ISSN 0146-4833.

44. WEAVER, Kari D. Censorship in the Digital Age the World Over. Iš: Advanced Methodologies and Technologies in Government and Society Hershey, PA, USA: IGI Global, 2019, pp. 481-492. ISBN 9781522576617.

45. HABER, Stuart ir STORNETTA, W. Scott. How to time-stamp a digital document. *Journal of Cryptology*. 1991, vol. 3, nr. 2. ISSN 0933-2790.

46. Cresitello-Dittmar, Ben. *Application of the Blockchain For Authentication and Verification of Identity.* 2013.

47. NAKAMOTO, Satoshi. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008.

48. TRANQUILLINI, Andrea. BLOCKCHAIN:
WHAT IS IT? *Journal of Securities Operations & Custody*. 2016, vol. 8, nr. 4, pp. 287-291. ISSN 1753-1802.

49. GAVIN, Dr ir CTO, Wood. ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER. 2014.

50. *Technical background of version 1 Bitcoin addresses - Bitcoin Wiki* [interaktyvus]. [žiūrėta: 2019 01 12]. Prieiga per:
https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses.

51. SCHNEIER, Bruce. *One-Way Hash Functions*. Iš: *Applied Cryptography*. 2-oji laida. 2015. ISBN 9781-119183471.

52. *Cryptocurrency Market Capitalizations | CoinMarketCap* [interaktyvus]. [žiūrėta: 2019 04 03]. Prieiga per: https://coinmarketcap.com.

53. *Block – Bitcoin Wiki* [interaktyvus]. [žiūrėta: 2019 05 02]. Prieiga per:
https://en.bitcoin.it/wiki/Block#Block_structure.

54. *Blockchain Model – Block* [interaktyvus]. [žiūrėta: 2019 05 02]. Prieiga per:
https://docs.neo.org/developerguide/en/articles/blockchain/block.html.

55. *Blockchain Explorer - Search the Blockchain | BTC | ETH | BCH* [interaktyvus]. [žiūrėta: 2019 05 12]. Prieiga per: https://www.blockchain.com/explorer.

56. ANTONOPOULOS, Andreas M. *Mastering Bitcoin.* 2-oji laida. Sebastopol: O'Reilly, 2017. ISBN 1491954388.

57. *Merkling in Ethereum* [interaktyvus]. 2015 [žiūrėta: 2019 05 10]. Prieiga per:
https://blog.ethereum.org/2015/11/15/merkling-in-ethereum.

58. BALIGA, Arati. *The Blockchain Landscape* [interaktyvus]. 2016 [žiūrėta: 2019 05 02]. Prieiga per: https://pdfs.semanticscholar.org/c826/b333dfb04e3053a7c2cb3b881bff1d952942.pdf.

59. KIFFER, Lucianna, LEVIN, Dave and MISLOVE, Alan. Stick a fork in it. *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. Nov 30, 2017, pp. 94-100.

60. *Advantages and Disadvantages of Permissionless Blockchain* [interaktyvus]. 2018 [žiūrėta: [žiūrėta: 2019 05 02]. Prieiga per: https://www.blockchain-council.org/blockchain/advantages-and-disadvantages-of-permissionless-blockchain.

61. *Ethereum White Paper* [interaktyvus]. 2014 [žiūrėta 2019 04 20]. Prieiga per: https://github.com/ethereum/wiki/wiki/White-Paper.

62. *Funding, Incentives, and Governance* [interaktyvus]. 2016 [žiūrėta: 2019 04 20]. Prieiga per: https://z.cash/blog/funding/.

63. Weizhi Meng, TISCHHAUSER, Elmar Wolfgang, Qingju Wang, Yu Wang ir Jinguang Han. When Intrusion Detection Meets Blockchain Technology: A Review. *IEEE Access*. 2018, vol. 6, pp. 10179-10188. ISSN 2169-3536.

64. *On Public and Private Blockchains* [interaktyvus]. 2015 [žiūrėta: 2019 04 21]. Prieiga per: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains.

65. ATZORI, Marcella. BLOCKCHAIN TECHNOLOGY AND DECENTRALIZED GOVERNANCE: IS THE STATE STILL NECESSARY?. *Journal of Governance and Regulation*. 2017, vol. 6, nr. 1. ISSN 2220-9352.

66. UNDERWOOD, Sarah. Blockchain beyond bitcoin. *Communications of the ACM*. 2016, vol. 59, nr. 11, pp. 15-17. ISSN 0001-0782.

67. CAI, Yuanfeng and ZHU, Dan. Fraud detections for online businesses: a perspective from blockchain technology. *Financial Innovation*. 2016, vol. 2, nr. 1, pp. 1-10. ISSN 2199-4730.

68. ALKETBI, Ahmed, NASIR, Qassim ir TALIB, Manar Abu. Blockchain for government services - Use cases, security benefits and challenges. *2018 15th Learning and Technology Conference (L&T)*. Feb 2018, pp. 112-119.

69. ATLAM, Hany F., ALENEZI, Ahmed, ALASSAFI, Madini O. ir WILLS, Gary B. Blockchain with Internet of Things: Benefits, Challenges, and Future Directions. *International Journal of Intelligent Systems and Applications*. 2018, vol. 6, pp. 40-48.

70. MACLEISH, John. *Understanding Blockchain Technology And What It Means for Your Business.* 1-oji laida. New York: Fawcett Columbine, 1991. ISBN 9780449906934.

71. MYLREA, Michael ir GOURISETTI, Sri Nikhil Gupta. Blockchain for smart grid resilience: Exchanging distributed energy at speed, scale and security. *2017 Resilience Week (RWS)*. Sep 2017, pp. 18-23.

72. GUO, Ye and LIANG, Chen. Blockchain application and outlook in the banking industry. *Financial Innovation*. 2016, vol. 2, nr. 1, pp. 1-12. ISSN 2199-4730.

73. RAIKWAR, Mayank, MAZUMDAR, Subhra, RUJ, Sushmita, SEN GUPTA, Sourav, CHATTOPADHYAY, Anupam ir LAM, Kwok-Yan. A Blockchain Framework for Insurance Processes. *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. Feb 2018, pp. 1-4.

74. ROMAN-BELMONTE, Juan M., DE LA CORTE-RODRIGUEZ, Hortensia ir RODRIGUEZ-MERCHAN, E. Carlos. How blockchain technology can change medicine. *Postgraduate Medicine*. 2018, vol. 130, nr. 4, pp. 420-427. ISSN 0032-5481.

75. Seyoung Huh, Sangrae Cho ir Soohyung Kim. Managing IoT devices using blockchain platform. *2017 19th International Conference on Advanced Communication Technology (ICACT)*. 2017, pp. 464-467.

76. *Ethereum Debit Card* [interaktyvus]. [žiūrėta: 2019 05 08]. Prieiga per: https://uquid.com/ethereum-debit-card.

77. *MCO Visa Card* [interaktyvus]. [žiūrėta: 2019 05 08]. Prieiga per: https://crypto.com/en/cards.html.

78. *BitPay – Welcome to the future of payments* [interaktyvus]. [žiūrėta: 2019 05 08]. Prieiga per: https://bitpay.com/.

79. *Buy, Sell & Accept Cryptocurrencies* [interaktyvus]. [žiūrėta: 2019 05 08]. Prieiga per: https://coingate.com/.

80. NATH, Indranil. Data Exchange Platform to Fight Insurance Fraud on Blockchain. *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. Dec 2016, pp. 821-825.

81. LI, Wenting, SFORZIN, Alessandro, FEDOROV, Sergey ir KARAME, Ghassan. Towards Scalable and Private Industrial Blockchains. *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. Apr 2, 2017, pp. 9-14.

82. WATANABE, Hiroki, FUJIMURA, Shigeru, NAKADAIRA, Atsushi, MIYAZAKI, Yasuhiko, AKUTSU, Akihito ir KISHIGAMI, Jay. Blockchain contract: Securing a blockchain applied to smart contracts. *2016 IEEE International Conference on Consumer Electronics (ICCE)*. Jan 2016, pp. 467-468.

83. DANNEN, Chris. *Introducing Ethereum and Solidity.* Brooklyn, New York, USA: Apress, 2017. ISBN 978-1-4842-2534-9.

84. *Solidity* [interaktyvus]. [žiūrėta: 2019 05 09]. Prieiga per: https://solidity.readthedocs.io/en/develop/index.html.

85. *LLL* [interaktyvus]. [žiūrėta: 2019 05 09]. Prieiga per: https://lll-docs.readthedocs.io/en/latest/.

86. *Vyper* [interaktyvus]. [žiūrėta: 2019 05 09]. Prieiga per: https://vyper.readthedocs.io/en/latest/.

87. *Bamboo* [interaktyvus]. [žiūrėta: 2019 05 09]. Prieiga per: https://github.com/pirapira/bamboo.

88. BARTOLETTI, Massimo ir POMPIANU, Livio. An Empirical Analysis of Smart Contracts: Platforms, Applications, and Design Patterns. 2017, pp. 494-509. ISSN 9783-319702780.

89. *Ethereum Homestead Documentation* [interaktyvus]. [žiūrėta: 2019 05 09]. Prieiga per: http://ethdocs.org/en/latest/index.html.

90. *web3.js - Ethereum JavaScript API – web3.js 1.0.0 documentation* [interaktyvus]. 2019 [žiūrėta: 2019 05 09]. Prieiga per https://web3js.readthedocs.io/en/1.0.

91. *Civic Secure Identity Ecosystem* [interaktyvus]. [žiūrėta: 2019 05 11]. Prieiga per: https://www.civic.com.

92. *uPort* [interaktyvus]. [žiūrėta: 2019 05 11]. Prieiga per: https://www.uport.me.

93. *iTrue Platform - Self-Sovereign Biometric Identity for DWeb* [interaktyvus]. [žiūrėta: 2019 05 11]. Prieiga per: https://itrue.io.

94. *Blinking.id – Blockchain-based Digital ID solution* [interaktyvus]. [žiūrėta: 2019 05 11]. Prieiga per: https://blinking.id.

95. *Self-Sovereign Identity for more Freedom and Privacy - SelfKey* [interaktyvus]. [žiūrėta: 2019 05 11]. Prieiga per: https://selfkey.org.

96. *HYDRO Raindrop: Public Authentication On The Blockchain.* [interaktyvus]. 2018 [žiūrėta: 2019 05 11]. Prieiga per: https://www.hydrogenplatform.com/Hydro_Raindrop_White_Paper.pdf.

97. *#4 Authentication and Authorization* [interaktyvus]. [žiūrėta: 2019 05 11]. Prieiga per: https://medium.com/building-a-dapp-on-ethereum/4-authentication-and-authorization-b6719455c94e.

98. *hoxxep/Ethereum-2FA: Two factor authentication through an Ethereum contract* [interaktyvus]. 2017 [žiūrėta: 2019 05 11]. Prieiga per: https://github.com/hoxxep/Ethereum-2FA.

99. *Hydrogen* [interaktyvus]. 2019 [žiūrėta: 2019 05 11]. Prieiga per: https://www.hydrogenplatform.com.

100. LUNDKVIST, Christian, HECK, Rouven, TORSTENSSON, Joel, MITTON, Zac ir SENA, Michael. *UPORT: A PLATFORM FOR SELF-SOVEREIGN IDENTITY.* 2016 [žiūrėta: 2019 05 11]. Prieiga per: http://blockchainlab.com/pdf/uPort_whitepaper_DRAFT20161020.pdf.

PRIEDAI

1. priedas. Vartotojų kriptografinio parašo ir registracijos būsenos tikrinimo „Ethereum“ išmaniuoju kontraktu spartos priklausomybės nuo registruotų vartotojų skaičiaus tyrimo rezultatai.

Registruotų vartotojų skaičius	Matavimų reikšmės, laikas, ms	Vidurkis, ms	Mediana, ms
0	4,93, 5,10, 5,14, 5,46, 5,56, 5,64, 5,66, 5,81, 8,61, 10,10	6,20	5,60
500	7,46, 9,54, 9,95, 11,32, 11,80, 16,07, 17,32, 25,33, 42,59, 155,32	30,67	13,94
1000	3,36, 3,76, 3,78, 4,19, 4,25, 4,87, 17,83, 56,92, 74,07, 103,21	27,62	4,56
1500	4,63, 5,19, 5,25, 5,27, 6,37, 7,78, 15,54, 29,30, 43,12, 63,11	18,56	7,08
2000	4,10, 4,76, 5,53, 8,40, 9,33, 13,63, 14,69, 15,01, 15,95, 16,64	10,80	11,48
2500	3,44, 3,52, 3,66, 3,94, 4,05, 4,15, 4,41, 5,12, 6,26, 31,31	6,99	4,10
3000	4,79, 4,84, 4,98, 5,54, 5,91, 6,40, 6,63, 8,47, 132,92, 154,39	33,49	6,16
3500	3,32, 3,32, 3,38, 3,43, 3,61, 3,61, 3,68, 5,19, 102,07, 192,92	32,45	3,61
4000	3,21, 3,28, 3,33, 3,40, 3,40, 3,46, 3,59, 3,82, 3,88, 6,76	3,81	3,43
4500	4,45, 4,75, 5,27, 5,62, 6,99, 8,80, 19,55, 48,73, 58,09, 100,58	26,28	7,90
5000	4,39, 4,42, 4,75, 4,83, 5,15, 5,16, 5,54, 5,55, 8,14, 24,01	7,19	5,16
5500	4,26, 4,45, 4,92, 5,12, 5,52, 9,51, 9,62, 9,92, 15,38, 73,31	14,20	7,52
6000	4,69, 5,48, 6,00, 6,95, 7,03, 7,52, 8,51, 14,23, 15,74, 16,97	9,31	7,28
6500	3,16, 3,52, 3,61, 3,62, 3,76, 3,82, 3,90, 4,44, 5,50, 6,40	4,17	3,79
7000	3,24, 3,60, 3,83, 4,12, 8,64, 10,54, 17,82, 20,93, 26,97, 28,50	12,82	9,59
7500	3,21, 3,36, 3,52, 3,53, 3,54, 3,65, 7,27, 9,47, 10,43, 12,83	6,08	3,60
8000	3,18, 3,29, 3,40, 3,52, 3,59, 3,70, 3,73, 3,88, 4,11, 5,72	3,81	3,65
8500	2,96, 2,97, 3,14, 3,15, 3,36, 3,44, 3,48, 3,52, 3,84, 6,33	3,62	3,40
9000	3,04, 3,08, 3,19, 3,22, 3,31, 3,34, 3,42, 3,61, 4,32, 6,04	3,66	3,33
9500	4,88, 5,10, 5,72, 5,81, 7,17, 7,20, 7,25, 7,48, 7,48, 15,75	7,38	7,19
10000	4,72, 4,88, 5,12, 5,39, 5,98, 7,46, 9,38, 17,10, 17,34, 179,25	25,66	6,72
10500	3,54, 3,55, 3,98, 4,21, 5,01, 5,09, 8,33, 19,32, 25,62, 77,12	15,58	5,05
11000	3,14, 3,25, 3,29, 3,34, 3,55, 3,72, 3,81, 4,14, 4,74, 6,09	3,91	3,64
11500	4,68, 4,73, 4,86, 4,97, 5,15, 5,50, 6,40, 7,77, 8,59, 15,22	6,79	5,33
12000	3,36, 3,44, 3,44, 3,59, 3,62, 3,63, 3,65, 4,32, 4,39, 6,44	3,99	3,63
12500	4,91, 4,91, 4,92, 5,03, 5,18, 5,34, 5,35, 5,46, 5,93, 9,87	5,69	5,26
13000	3,43, 5,25, 5,57, 6,00, 6,18, 6,98, 7,24, 7,29, 13,49, 14,74	7,62	6,58
13500	6,92, 7,96, 8,60, 8,61, 8,73, 10,19, 10,75, 11,52, 11,58, 17,75	10,26	9,46
14000	3,24, 3,28, 3,32, 3,54, 3,88, 4,45, 4,53, 4,67, 7,37, 8,07	4,64	4,17
14500	3,87, 4,04, 4,08, 5,68, 5,81, 6,27, 6,86, 8,13, 8,73, 9,02	6,25	6,04
15000	3,25, 3,30, 3,52, 3,53, 3,54, 3,63, 3,65, 3,92, 3,93, 6,23	3,85	3,59
15500	4,01, 5,24, 5,33, 6,30, 6,59, 10,14, 28,96, 33,36, 34,99, 60,25	19,52	8,37
16000	3,27, 3,29, 3,32, 3,32, 3,33, 3,48, 3,51, 4,05, 4,54, 6,24	3,84	3,41
16500	3,24, 3,33, 3,54, 3,57, 3,72, 3,72, 4,48, 4,49, 4,73, 6,14	4,10	3,72
17000	3,16, 3,31, 3,31, 3,39, 3,48, 3,49, 3,76, 3,85, 3,93, 7,02	3,87	3,49
17500	3,25, 3,37, 3,43, 3,62, 4,19, 4,19, 8,01, 9,79, 12,46, 13,05	6,54	4,19

18000	3,57, 3,58, 3,71, 3,74, 3,90, 3,93, 4,03, 4,86, 5,13, 7,64	4,41	3,92
18500	3,28, 3,35, 3,48, 3,56, 3,68, 3,77, 3,85, 3,90, 4,05, 6,84	3,98	3,73
19000	3,29, 3,75, 3,83, 4,01, 4,12, 4,31, 4,51, 7,79, 11,87, 78,48	12,60	4,22
19500	3,23, 3,30, 3,45, 3,74, 3,75, 3,90, 3,91, 4,32, 4,42, 5,77	3,98	3,83
20000	4,27, 4,42, 4,46, 4,65, 4,84, 4,91, 5,21, 5,38, 5,46, 9,13	5,27	4,88
20500	3,41, 3,42, 3,45, 3,50, 4,04, 4,05, 4,34, 4,35, 6,65, 7,07	4,43	4,05
21000	3,26, 3,36, 3,37, 3,41, 3,44, 3,62, 3,74, 3,76, 3,89, 6,90	3,88	3,53
21500	3,03, 3,04, 3,30, 3,31, 3,73, 3,74, 3,82, 3,99, 5,46, 6,39	3,98	3,74
22000	3,73, 3,93, 4,13, 5,87, 5,96, 5,99, 6,04, 6,62, 9,97, 10,46	6,27	5,98
22500	3,07, 3,14, 3,32, 3,37, 3,55, 3,88, 3,92, 4,13, 4,48, 7,22	4,01	3,72
23000	3,45, 3,50, 3,59, 3,65, 3,78, 3,79, 3,97, 4,21, 4,96, 5,99	4,09	3,79
23500	5,37, 5,62, 6,81, 6,90, 7,18, 7,41, 7,43, 17,63, 21,68, 48,95	13,50	7,30
24000	2,96, 3,19, 3,27, 3,33, 3,44, 3,53, 3,87, 4,04, 4,20, 6,16	3,80	3,49
24500	5,04, 5,06, 5,30, 5,60, 5,68, 5,77, 6,00, 6,19, 12,97, 23,83	8,14	5,73
25000	3,32, 3,58, 3,60, 3,67, 4,10, 4,68, 5,20, 5,62, 6,14, 113,26	15,32	4,39
25500	3,31, 3,50, 3,55, 3,57, 3,60, 3,67, 3,68, 3,81, 4,78, 7,05	4,05	3,64
26000	5,16, 5,30, 5,32, 5,45, 6,05, 6,24, 6,53, 9,56, 11,50, 20,99	8,21	6,15
26500	3,56, 3,57, 3,71, 4,03, 4,15, 4,21, 4,43, 4,44, 4,79, 4,86	4,18	4,18
27000	2,93, 3,16, 3,24, 3,43, 3,67, 3,91, 3,99, 4,12, 4,46, 6,14	3,91	3,79
27500	3,81, 3,84, 4,41, 6,06, 6,11, 6,48, 6,54, 6,96, 14,24, 15,39	7,38	6,30
28000	3,18, 3,34, 3,37, 3,39, 3,41, 3,43, 3,73, 4,36, 4,50, 13,29	4,60	3,42
28500	3,75, 4,21, 5,02, 5,73, 6,56, 6,91, 6,94, 7,80, 14,84, 242,92	30,47	6,74
29000	3,27, 3,28, 3,30, 3,32, 3,43, 3,44, 3,61, 3,87, 4,48, 6,16	3,82	3,44
29500	3,19, 3,21, 3,29, 3,30, 3,45, 3,47, 3,61, 3,87, 4,05, 6,70	3,81	3,46
30000	3,17, 3,22, 3,37, 3,39, 3,61, 3,62, 3,68, 3,97, 4,46, 17,42	4,99	3,62
30500	3,65, 3,70, 3,77, 3,88, 3,88, 4,02, 4,14, 4,15, 5,86, 6,49	4,35	3,95
31000	3,36, 3,53, 3,57, 3,58, 3,59, 3,60, 3,69, 3,99, 5,40, 111,43	14,57	3,60
31500	5,24, 5,35, 5,43, 5,58, 6,99, 7,95, 8,05, 8,33, 124,60, 1324,29	150,18	7,47
32000	3,13, 3,48, 3,66, 3,72, 4,08, 4,16, 4,68, 4,94, 6,14, 6,26	4,43	4,12
32500	3,44, 3,51, 3,52, 3,61, 4,13, 4,19, 4,23, 4,25, 4,66, 6,84	4,24	4,16
33000	3,64, 3,71, 3,72, 3,83, 3,86, 3,87, 4,30, 7,07, 8,58, 16,39	5,90	3,87
33500	3,30, 3,36, 3,41, 3,61, 3,73, 3,76, 3,94, 3,94, 4,26, 6,33	3,96	3,75
34000	3,19, 3,35, 3,37, 3,49, 3,56, 3,57, 4,22, 4,50, 6,12, 6,44	4,18	3,57
34500	3,32, 3,37, 3,39, 3,53, 3,55, 3,57, 3,65, 3,77, 3,87, 6,88	3,89	3,56
35000	5,12, 6,53, 7,20, 7,22, 7,96, 8,87, 9,19, 9,31, 10,47, 11,12	8,30	8,42
35500	3,20, 3,23, 3,24, 3,32, 3,47, 3,67, 3,88, 4,04, 4,11, 6,90	3,91	3,57
36000	3,19, 3,31, 3,32, 3,38, 3,42, 3,51, 3,57, 3,92, 3,95, 7,72	3,93	3,47
36500	3,42, 3,49, 3,67, 3,76, 3,78, 4,06, 4,06, 4,09, 4,15, 6,87	4,14	3,92
37000	3,55, 3,75, 3,87, 3,90, 3,96, 4,04, 4,26, 4,53, 4,62, 6,27	4,28	4,00
37500	4,57, 5,06, 5,12, 5,18, 5,47, 6,25, 7,02, 7,10, 10,54, 11,62	6,79	5,86
38000	3,40, 3,41, 3,44, 3,73, 3,76, 3,87, 4,56, 4,85, 9,70, 161,90	20,26	3,82
38500	3,16, 3,39, 3,42, 3,44, 3,49, 3,61, 3,81, 4,31, 9,62, 10,00	4,83	3,55
39000	3,26, 3,54, 3,58, 3,59, 3,68, 3,73, 3,85, 3,96, 4,13, 14,39	4,77	3,71
39500	3,13, 3,15, 3,19, 3,23, 3,40, 3,44, 3,53, 3,56, 3,65, 5,96	3,62	3,42

40000	4,69, 4,70, 4,91, 5,11, 5,19, 5,22, 5,23, 5,70, 7,83, 8,07	5,67	5,21
40500	3,19, 3,19, 3,38, 3,52, 3,53, 3,53, 3,60, 3,70, 5,61, 7,57	4,08	3,53
41000	3,60, 3,67, 3,73, 3,88, 4,02, 4,12, 4,13, 4,78, 5,02, 6,96	4,39	4,07
41500	3,31, 3,46, 3,97, 4,04, 4,12, 4,41, 6,58, 20,51, 25,85, 26,53	10,28	4,27
42000	3,01, 3,13, 3,44, 3,51, 3,90, 3,99, 4,08, 4,45, 7,32, 55,01	9,18	3,95
42500	3,34, 3,36, 3,86, 3,87, 3,99, 4,03, 4,03, 4,81, 6,32, 6,44	4,41	4,01
43000	3,29, 3,32, 3,34, 3,35, 3,36, 3,41, 3,60, 3,70, 5,31, 6,57	3,93	3,39
43500	3,04, 3,20, 3,31, 3,37, 3,51, 3,53, 3,78, 3,81, 3,85, 6,97	3,84	3,52
44000	3,34, 3,45, 3,67, 3,71, 3,72, 3,84, 3,91, 4,15, 4,66, 5,99	4,04	3,78
44500	3,44, 3,46, 3,51, 3,57, 3,62, 3,69, 3,74, 3,76, 4,16, 6,44	3,94	3,66
45000	3,86, 4,49, 4,70, 4,88, 5,50, 5,51, 6,96, 7,31, 16,99, 131,48	19,17	5,51
45500	3,14, 3,19, 3,24, 3,33, 3,37, 3,49, 3,49, 3,56, 3,74, 6,21	3,68	3,43
46000	3,00, 3,04, 3,11, 3,12, 3,16, 3,29, 3,34, 3,75, 3,91, 5,54	3,53	3,23
46500	3,36, 3,42, 3,51, 3,53, 3,82, 4,07, 4,29, 4,92, 6,45, 6,61	4,40	3,95
47000	3,29, 3,39, 3,43, 3,54, 3,55, 3,58, 3,91, 3,95, 3,99, 6,52	3,92	3,57
47500	5,55, 6,46, 7,54, 8,10, 13,54, 16,54, 18,48, 24,73, 41,22, 50,39	19,26	15,04
48000	3,50, 3,64, 3,86, 3,98, 4,14, 4,28, 4,37, 4,49, 4,72, 5,72	4,27	4,21
48500	2,91, 2,92, 3,44, 3,57, 3,61, 4,32, 4,71, 4,80, 12,73, 61,62	10,46	3,97
49000	3,60, 3,75, 3,79, 3,83, 3,85, 3,91, 4,04, 4,29, 4,63, 6,76	4,25	3,88
49500	3,51, 3,66, 3,82, 3,86, 3,94, 4,16, 4,85, 6,36, 7,43, 9,36	5,10	4,05
50000	4,60, 4,69, 4,71, 4,86, 4,93, 5,22, 5,35, 5,36, 5,53, 8,72	5,40	5,08

2. priedas. Vartotojų kriptografinio parašo ir registracijos būsenos tikrinimo „Node.js“ aplinkoje ir „MySQL“ duomenų bazėje spartos priklausomybės nuo registruotų vartotojų skaičiaus tyrimo rezultatai.

Registruotų vartotojų skaičius	Matavimų reikšmės, laikas, ms	Vidurkis, ms	Mediana, ms
0	2,00, 2,52, 2,53, 2,70, 2,94, 3,52, 4,64, 5,09, 6,36, 11,08	4,34	3,23
500	2,12, 2,13, 2,29, 2,61, 2,67, 2,96, 2,98, 3,03, 3,07, 3,75	2,76	2,82
1000	2,19, 2,33, 2,65, 2,72, 2,88, 2,94, 3,15, 3,17, 3,32, 5,05	3,04	2,91
1500	1,75, 1,80, 4,01, 4,72, 5,38, 7,33, 7,82, 8,66, 13,29, 13,63	6,84	6,36
2000	1,54, 2,01, 2,07, 2,25, 2,41, 2,58, 2,73, 3,04, 3,15, 4,51	2,63	2,50
2500	1,80, 1,95, 2,60, 2,63, 3,08, 3,53, 3,54, 3,90, 3,93, 5,98	3,29	3,31
3000	1,51, 1,67, 1,92, 1,92, 2,08, 2,33, 2,42, 2,46, 3,44, 4,32	2,41	2,21
3500	1,79, 1,92, 2,09, 2,12, 2,43, 2,74, 2,85, 2,90, 3,00, 4,00	2,58	2,59
4000	1,78, 1,94, 2,18, 2,43, 2,65, 2,70, 2,71, 3,40, 3,64, 3,93	2,74	2,68
4500	1,76, 2,28, 2,46, 2,55, 2,60, 2,61, 2,65, 2,71, 2,78, 2,80	2,52	2,61
5000	1,72, 1,74, 1,80, 2,04, 2,56, 2,87, 3,53, 4,08, 4,17, 4,99	2,95	2,72
5500	1,74, 1,84, 2,29, 2,31, 2,33, 2,40, 2,67, 2,86, 2,97, 3,24	2,47	2,37
6000	1,95, 2,10, 2,20, 2,22, 2,25, 3,45, 4,03, 4,10, 4,94, 5,00	3,22	2,85
6500	1,90, 2,37, 2,38, 2,42, 2,50, 2,62, 2,65, 2,70, 2,76, 5,02	2,73	2,56
7000	1,55, 1,78, 1,79, 1,99, 2,18, 2,23, 2,24, 2,74, 5,00, 8,72	3,02	2,21
7500	1,39, 1,49, 1,62, 2,08, 2,27, 2,37, 2,68, 2,78, 2,87, 3,58	2,31	2,32
8000	1,44, 2,04, 2,15, 2,22, 2,30, 3,55, 3,91, 4,67, 6,52, 7,64	3,64	2,93
8500	1,85, 1,89, 2,15, 2,22, 2,33, 2,39, 2,45, 3,08, 3,42, 4,40	2,62	2,36
9000	1,47, 1,83, 1,96, 2,15, 2,62, 3,35, 3,51, 4,17, 5,17, 6,28	3,25	2,99
9500	1,52, 1,56, 1,57, 1,62, 1,65, 1,73, 1,76, 2,11, 2,34, 3,79	1,97	1,69
10000	1,85, 1,92, 2,31, 2,37, 2,46, 2,71, 2,92, 3,59, 3,66, 3,73	2,75	2,59
10500	1,39, 1,58, 1,72, 1,94, 1,97, 2,52, 2,58, 2,63, 2,73, 2,89	2,20	2,25
11000	1,38, 1,66, 1,88, 1,93, 2,66, 2,87, 2,99, 3,12, 3,46, 3,63	2,56	2,77
11500	1,62, 1,89, 2,03, 2,23, 2,68, 2,87, 2,89, 2,90, 3,14, 3,77	2,60	2,78
12000	1,57, 1,66, 1,72, 1,89, 1,89, 2,25, 2,26, 2,51, 2,52, 2,63	2,09	2,07
12500	2,36, 2,36, 2,61, 2,64, 2,93, 3,73, 3,95, 4,33, 4,80, 6,46	3,62	3,33
13000	1,56, 1,75, 1,91, 2,32, 2,41, 2,64, 2,70, 3,04, 3,53, 3,53	2,54	2,53
13500	1,62, 1,67, 1,67, 1,71, 1,79, 1,96, 1,99, 2,01, 2,02, 2,12	1,86	1,88
14000	1,61, 1,83, 1,94, 1,97, 2,05, 2,21, 2,34, 2,40, 2,62, 2,65	2,16	2,13
14500	1,88, 2,13, 2,40, 2,63, 2,73, 2,93, 3,04, 3,61, 4,15, 4,73	3,02	2,83
15000	1,69, 1,75, 1,84, 2,03, 2,21, 2,51, 3,26, 3,30, 3,91, 14,53	3,70	2,36
15500	1,73, 1,77, 1,85, 1,86, 2,11, 2,83, 3,10, 3,17, 3,19, 3,68	2,53	2,47
16000	1,83, 2,01, 2,06, 2,06, 2,24, 2,26, 2,29, 2,40, 2,40, 2,61	2,22	2,25
16500	1,57, 1,82, 1,97, 2,00, 2,48, 2,54, 2,61, 2,87, 3,31, 4,48	2,57	2,51
17000	1,50, 1,76, 1,81, 1,90, 1,98, 2,10, 2,13, 2,35, 6,50, 8,17	3,02	2,04
17500	1,26, 1,60, 1,78, 2,07, 2,10, 2,35, 2,45, 3,54, 3,81, 3,84	2,48	2,23
18000	1,67, 2,00, 2,30, 2,35, 2,39, 2,44, 2,81, 3,22, 3,22, 3,75	2,62	2,42
18500	1,75, 1,87, 1,90, 2,00, 2,01, 2,21, 2,53, 2,60, 3,01, 3,56	2,34	2,11

19000	1,91, 1,99, 2,26, 2,38, 2,49, 2,59, 3,21, 3,66, 3,80, 5,14	2,94	2,54
19500	1,60, 1,68, 1,93, 2,20, 2,36, 2,90, 3,06, 3,19, 3,32, 3,47	2,57	2,63
20000	1,75, 1,88, 1,91, 2,10, 2,11, 2,34, 2,36, 2,53, 3,32, 3,48	2,38	2,23
20500	1,65, 1,73, 2,17, 2,22, 2,36, 2,48, 2,65, 2,66, 2,83, 5,52	2,63	2,42
21000	1,49, 1,66, 1,96, 2,02, 2,28, 3,54, 3,98, 4,67, 4,82, 4,84	3,13	2,91
21500	1,74, 1,85, 2,20, 2,53, 2,65, 3,34, 3,71, 3,96, 5,33, 6,55	3,39	3,00
22000	1,83, 2,33, 2,33, 2,43, 2,73, 2,93, 3,23, 3,72, 3,77, 4,33	2,96	2,83
22500	1,96, 2,06, 2,16, 2,26, 2,47, 2,74, 3,15, 3,68, 3,70, 3,86	2,80	2,61
23000	1,60, 1,69, 1,82, 1,85, 2,05, 2,18, 2,21, 2,24, 2,30, 2,39	2,03	2,12
23500	1,78, 2,22, 2,51, 2,62, 2,75, 3,09, 3,44, 3,61, 3,76, 5,16	3,09	2,92
24000	1,47, 2,16, 2,17, 2,18, 2,60, 2,80, 3,04, 3,13, 3,29, 4,23	2,71	2,70
24500	1,98, 2,10, 2,28, 2,40, 2,47, 2,50, 2,61, 4,61, 4,96, 7,64	3,36	2,49
25000	1,44, 1,54, 1,65, 1,85, 1,91, 1,95, 2,13, 2,14, 2,88, 4,13	2,16	1,93
25500	1,84, 1,97, 2,53, 2,62, 2,68, 2,75, 2,90, 3,58, 4,66, 4,90	3,04	2,72
26000	2,03, 2,34, 2,52, 2,54, 2,79, 3,37, 3,93, 4,13, 4,66, 4,77	3,31	3,08
26500	1,81, 1,91, 1,93, 2,03, 2,03, 2,11, 2,20, 2,34, 2,43, 2,51	2,13	2,07
27000	1,99, 2,32, 2,34, 2,97, 3,16, 3,32, 3,92, 3,97, 4,45, 5,08	3,35	3,24
27500	1,48, 1,69, 1,75, 1,97, 1,99, 2,35, 2,36, 2,62, 2,84, 3,86	2,29	2,17
28000	1,88, 1,93, 2,34, 2,84, 2,87, 2,96, 3,08, 3,17, 3,42, 4,07	2,86	2,92
28500	1,67, 2,35, 2,36, 2,48, 2,55, 3,51, 3,73, 4,13, 4,33, 4,49	3,16	3,03
29000	1,97, 1,98, 2,08, 2,13, 2,80, 2,88, 2,94, 4,15, 4,46, 4,81	3,02	2,84
29500	1,55, 1,86, 1,95, 2,06, 2,13, 2,28, 2,33, 2,49, 2,63, 2,90	2,22	2,21
30000	1,47, 1,54, 1,57, 1,57, 1,61, 1,61, 1,68, 1,93, 2,19, 2,73	1,79	1,61
30500	1,52, 1,71, 1,75, 1,87, 1,93, 1,98, 2,25, 2,66, 2,84, 2,92	2,14	1,96
31000	1,53, 1,91, 2,20, 2,21, 2,24, 3,49, 4,14, 4,46, 4,76, 4,81	3,18	2,87
31500	2,02, 2,04, 2,08, 2,26, 2,53, 2,82, 3,37, 3,52, 3,61, 3,62	2,79	2,68
32000	1,61, 2,12, 2,45, 2,48, 2,81, 3,26, 3,64, 3,91, 4,67, 4,92	3,19	3,04
32500	1,99, 2,06, 2,40, 2,67, 2,76, 2,81, 2,83, 3,06, 3,19, 3,44	2,72	2,79
33000	1,57, 1,71, 1,83, 1,90, 2,19, 2,26, 2,29, 2,59, 2,82, 3,09	2,23	2,23
33500	1,46, 1,72, 1,81, 1,95, 2,00, 2,01, 2,11, 2,15, 2,38, 2,49	2,01	2,01
34000	1,71, 1,92, 2,06, 2,07, 2,08, 2,19, 2,51, 2,68, 2,82, 3,17	2,32	2,14
34500	1,67, 2,21, 2,34, 2,64, 2,85, 2,96, 3,07, 3,12, 3,36, 3,94	2,82	2,91
35000	1,81, 2,17, 2,23, 2,36, 2,40, 2,45, 2,92, 3,00, 4,38, 7,17	3,09	2,43
35500	1,59, 1,59, 1,62, 1,81, 1,84, 1,86, 2,12, 2,32, 2,39, 2,55	1,97	1,85
36000	1,71, 1,79, 2,02, 2,02, 2,09, 2,10, 2,19, 2,28, 2,33, 2,46	2,10	2,10
36500	1,57, 1,92, 1,93, 1,95, 1,99, 2,07, 2,38, 2,47, 2,77, 3,64	2,27	2,03
37000	1,71, 2,28, 2,39, 2,49, 5,04, 5,95, 8,64, 9,05, 10,04, 10,25	5,78	5,50
37500	1,55, 1,62, 1,69, 1,90, 1,98, 2,00, 2,10, 2,15, 2,18, 2,86	2,00	1,99
38000	1,62, 2,10, 2,17, 2,19, 2,36, 2,38, 2,41, 2,51, 2,55, 3,23	2,35	2,37
38500	1,53, 1,77, 1,78, 1,83, 1,89, 2,05, 2,16, 2,21, 2,23, 2,45	1,99	1,97
39000	1,77, 1,96, 2,16, 2,28, 2,29, 2,51, 2,66, 2,71, 3,19, 3,60	2,51	2,40
39500	1,84, 1,84, 1,99, 2,12, 2,24, 2,54, 3,74, 4,39, 5,82, 6,99	3,35	2,39
40000	1,55, 1,57, 1,66, 1,68, 1,71, 2,16, 2,30, 2,36, 2,42, 2,53	1,99	1,94
40500	1,87, 1,95, 1,99, 2,12, 2,44, 2,79, 2,80, 2,89, 3,15, 4,00	2,60	2,62

41000	1,96, 2,03, 2,24, 2,74, 3,03, 3,08, 3,08, 3,20, 3,27, 3,30	2,79	3,06
41500	1,92, 1,93, 2,09, 2,11, 2,37, 2,53, 2,74, 2,75, 3,61, 14,71	3,68	2,45
42000	1,56, 1,62, 2,09, 2,28, 2,41, 2,68, 2,75, 2,76, 2,89, 3,21	2,43	2,55
42500	1,51, 1,81, 1,88, 1,98, 2,18, 2,19, 2,32, 2,43, 2,59, 4,82	2,37	2,19
43000	1,55, 1,83, 2,08, 2,17, 2,31, 2,31, 2,35, 2,47, 2,53, 3,93	2,35	2,31
43500	1,49, 1,59, 1,73, 1,83, 1,83, 2,15, 2,28, 2,33, 2,38, 2,48	2,01	1,99
44000	1,60, 1,66, 2,18, 2,24, 2,25, 2,38, 2,46, 2,52, 2,78, 2,83	2,29	2,32
44500	1,64, 1,75, 1,95, 2,01, 2,05, 2,11, 2,13, 2,34, 2,51, 3,02	2,15	2,08
45000	1,87, 1,88, 2,06, 2,07, 2,41, 2,54, 2,69, 2,87, 3,11, 3,15	2,47	2,48
45500	1,55, 1,58, 1,60, 2,00, 2,02, 2,29, 2,37, 2,51, 3,16, 3,40	2,25	2,16
46000	1,63, 1,67, 1,70, 2,04, 2,08, 2,11, 2,17, 2,41, 2,42, 2,68	2,09	2,10
46500	1,58, 1,66, 1,93, 1,99, 2,01, 2,04, 2,15, 2,17, 2,28, 3,60	2,14	2,03
47000	1,61, 1,97, 2,04, 2,14, 2,32, 2,38, 2,39, 2,56, 2,84, 4,35	2,46	2,35
47500	1,57, 1,60, 1,63, 1,66, 1,66, 1,67, 1,67, 1,68, 2,03, 2,31	1,75	1,67
48000	1,77, 1,78, 1,80, 1,82, 1,95, 2,06, 2,19, 2,25, 2,30, 2,33	2,03	2,01
48500	1,78, 1,82, 1,92, 2,03, 2,08, 2,23, 2,43, 2,71, 2,81, 2,84	2,27	2,16
49000	1,75, 1,80, 1,83, 1,85, 2,03, 2,10, 2,28, 2,31, 2,76, 2,90	2,16	2,07
49500	1,71, 2,14, 2,16, 2,17, 2,27, 2,29, 2,29, 2,45, 2,71, 3,78	2,40	2,28
50000	1,43, 1,75, 1,84, 2,04, 2,07, 2,16, 2,27, 2,31, 2,57, 2,96	2,14	2,12