VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Skaitmeninės tapatybės valdymas taikant blokų grandinę

Digital Identity Management using Blockchain

Bakalauro darbas

Atliko:	Jurgis Kargaudas	(parašas)
Darbo vadovas:	lekt. Aurimas Šimkus	(parašas)
Darbo recenzentas:	lekt. Andrius Adamonis	(parašas)

Vilnius – 2018

Santrauka

Bus pridėta parašius visą darbą.

Summary

To be added when the whole thesis is finished.

TURINYS

Įvadas

Interneto paslaugos šiais laikais yra neatsiejama žmonių gyvenimo dalis. Norėdami individualizuoti turinį, sustiprinti taikomosios programos saugumą ar siekdami iš anksto išvengti kenkėjiškų tikslų turinčių asmenų ar sukurtų robotų, paslaugų tiekėjai siekia identifikuoti savo naudotojus. Interneto naudotojų skaičiui perkopus 4 milijardus [Min18], o kiekvienam naudotojui vidutiniškai turint po 7 skirtingas socialines paskyras [MM17], asmenų tapatybių valdymas, autentifikavimas ir autorizavimas tampa vis didesniu iššūkiu.

Tapatybių valdymas kelia problemų naudotojams. Bene didžiausi atsiradę keblumai: milžiniškas įsimintinų slaptažodžių kiekis bei sunkumai kontroliuojant savo asmens duomenų sklaidą skirtingose sistemose. Vidutiniškai interneto naudotojas turi 25 slaptažodžių reikalaujančias paskyras ir per dieną turi įvesti 8-is slaptažodžius [FH07]. Susidarius tokiai situacijai, per didelis įsimintinų slaptažodžių kiekis neretai priverčia naudotojus paaukoti saugumą dėl patogumo ir pradėti naudoti tą patį slaptažodį sirtingoms sistemoms [PM03; Sam99]. Naudotojas, turėdamas keletą paskyrų skirtingose sistemose, taip pat praranda dalį savo asmens duomenų kontrolės. Jam tenka pasitikėti taikomosios programos naudojamomis technologijomis ir metodais bei tikėtis, kad jie bus pakankamai saugūs ir stabilūs, o suteikti asmens duomenys nepasieks nepageidaujamų adresatų. Didėjant naudojamų paslaugų kiekiui, naudotojo skaitmeninės tapatybės duomenis turi vis daugiau taikomųjų programų ir bent vienai iš jų patyrus programišių įsilaužimą ar kitokią nesėkmę, jautrūs naudotojo duomenys gali būti paviešinti.

Taikomi skirtingi metodai skaitmeninei tapatybei internete valdyti. Šiais laikais vienas dažniausiai internete sutinkamų sprendimų yra jungtinis (angl. *federated*) tapatybės valdymas ir jo suteikiamas vienkartinis prisijungimas (angl. *single sign-on*). Šis sprendimas leidžia naudotojui pasirinkti vieną tapatybės tiekėją (angl. *identity provider*) ir patikėti jam skaitmeninės tapatybės valdymą. Tokiu būdu naudotojui pakanka turėti paskyrą tik tapatybės tiekėjo sistemoje, o kreipiantis į paslaugas naudotis jau turima tapatybės tiekėjo paskyra. Tačiau šis sprendimo būdas taip pat turi aiškių trūkumų: naudotojas negali prisijungti prie paslaugų, nepalaikančių pasirinkto tapatybės tiekėjo, jo pasiekiamumas tampa vieninteliu nesėkmės tašku (angl. *single point of failure*), naudotojas taip pat praranda dalį savo asmens duomenų kontrolės. Naršantis internete asmuo yra priverstas pasitikėti tapatybės tiekėjo gebėjimu perduoti tik naudotojo leistus asmens duomenis ir tik toms trečiosioms šalims, kurias jis patvirtina. Kaip rodo *Cambridge Analytica* incidentas [Gra18], net didžiosios kompanijos, tokios kaip „Facebook“, ne visada sugeba tai užtikrinti.

Blokų grandinė (angl. *blockchain*) yra nauja alternatyva skaitmeninės tapatybės valdymui. Ši technologija veikia kaip paskirstytų įrašų platforma (angl. *distributed ledger platform*), kurioje kiekvienas įrašas yra nekintamas, o visi užfiksuoti įrašai atspindi tikslią transakcijų istoriją nuo pat grandinės sukūrimo [Baa16]. Saugant tapatybės duomenis šioje grandinėje ir pritaikius reikiamą blokų grandinės pasiekiamumo lygį įrašų rašymui ir skaitymui, asmuo visada žinotų, kokia trečioji šalis gali pasiekti kokius tapatybės duomenis. Kadangi blokų grandinė yra decentralizuota, pritaikius ją skaitmeninės tapatybės valdyme taip pat būtų galima išvengti šioje srityje dažnos vienintelio nesėkmės taško problemos.

Šiame darbe blokų grandinės tinkamumas skaitmeninės tapatybės valdymui tiriamas iš

naudotojo perspektyvos. Pateikus esminius naudotojų poreikius identiteto valdymui, apžvelgiamas dabar naudojamų sistemų gebėjimas įgyvendinti šiuos reikalavimus. Apibendrinus pagrindines neišspręstas naudotojams kylančias problemas, tiriama, kaip blokų grandinė gali padėti jas įveikti, kokie šios technologijos panaudojimo skaitmeniniame tapatybės valdyme pranašumai, trūkumai bei pritaikymo barjerai (angl. *adoption barriers*).

Darbo tikslas - išanalizuoti blokų grandinės tinkamumą skaitmeninės tapatybės valdymui.

Darbe keliami uždaviniai:

1. Išskirti naudotojų poreikius skaitmeninės tapatybės valdymui internete.
2. Apžvelgti skaitmeninės tapatybės valdymo sprendimus, vertinant jų gebėjimą įgyvendinti išskirtas naudotojų reikmes.
3. Įvardyti naudotojams kylančias problemas skaitmeninės tapatybės valdyme.
4. Apibūdinti blokų grandinės technologiją ir jos savybes.
5. Atsižvelgus į blokų grandinės savybes bei esamą technologijos taikymą identiteto srityje, įvertinti blokų grandinės gebėjimą spręsti įvardytas naudotojų problemas.
6. Pateikti blokų grandinės panaudojimo atvejį skaitmeninės tapatybės valdymui ir sukurti jo veikimą demonstruojantį prototipą.
7. Įvertinti pateiktą sprendimą apibūdinant jo privalumus, trūkumus ir galimus pritaikymo barjerus.

1. Skaitmeninės tapatybės valdymo apžvalga

Skaitmeniniame amžiuje tapatybės valdymas internete yra svarbi taikomųjų programų dalis. Šiame skyriuje apžvelgiamas asmenų identifikavimo internete poreikis, skaitmeninės tapatybės valdymo samprata, taikomi tapatybės valdymo modeliai bei pagrindinės naudotojams kylančios problemos. Skaitmeninės tapatybės, autentifikavimo, autorizavimo bei kitų darbe naudojamų terminų aiškinimai pateikiami skyriuje „Sąvokų apibrėžimai“.

1.1. Tapatybės atpažinimo poreikis

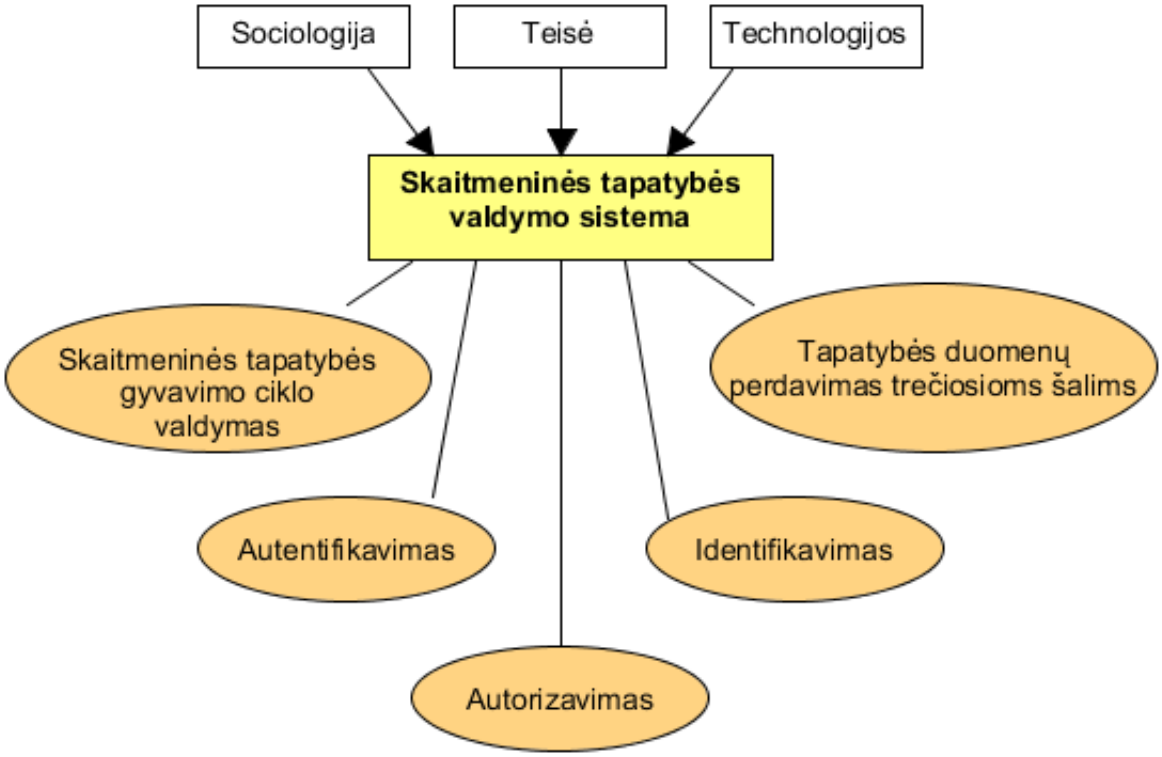
Naudotojo identifikavimas yra reikšminga interneto taikomųjų programų dalis. Paslaugų tiekėjai identifikuoja savo naudotojus norėdami [Ral14]:

- registruoti (angl. *log*) naudotojų veiklą,
- užtikrinti, kad naudotojas iš tikrųjų yra asmuo, kuris sakosi esąs,
- suteikti dalį funkcionalumo tik autorizuotiems naudotojams,
- individualizuoti tinklalapio ar taikomosios programos turinį pagal naudotojo poreikius,
- sukurti paslaugos naudotojų bendruomenę,
- išvengti galimų anoniminių naudotojų atakų.

Dėl išvardytų priežasčių naudotojų identifikavimas atlieka svarbią rolę įvairiose taikomųjų programų srityse - elektroninėje valdžioje, elektroninėje komercijoje, verslo sumanume (angl. *business intelligence*), tyrimuose bei saugume (angl. *homeland security*) [GV09]. Kiekvienas paslaugų tiekėjas turi pasirinkti, kaip autentifikuoti, ir, jei reikia, autorizuoti naudotojus. Programos kūrėjas taip pat turi užtikrinti naudotojo suteiktų duomenų saugumą, o naudotojui tenka rūpintis skirtingų turimų paskyrų priežiūra ir savo duomenų sklaida tarp skirtingų sistemų. Minimus tapatybės atpažinimo skaitmeninėje erdvėje aspektus nagrinėja skaitmeninės tapatybės valdymo disciplina.

1.2. Skaitmeninės tapatybės valdymo samprata

Dėl nuolat vykstančios interneto ir jame esančių paslaugų plėtros tapatybių valdymo uždavinys pastaraisiais metais tapo itin svarbus [GV09]. Skaitmeninės tapatybės valdymo pagrindinis uždavinys yra kontroliuoti tapatybę ir su ja susijusius procesus, tokius kaip autentifikavimas, autorizavimas, prieigų kontrolė, tapatybės gyvavimo ciklo valdymas bei saugus tapatybės atributų perdavimas trečiosioms šalims [CY10; DP08]. Sprendžiant šį uždavinį, sukurta skirtingų skaitmeninės tapatybės valdymo sistemų. Šioms sistemoms įtaką daro kiti tapatybę nagrinėjantys mokslai (pvz. sociologija), taip pat jos gali atlikti keletą skirtingų funkcijų, susijusių su naudotojų tapatybe. 1 paveiksle apibendrintas tapatybės valdymo sistemų kontekstas bei pagrindinės atliekamos užduotys:



1 pav. Skaitmeninių tapatybių valdymo sistemų kontekstas ir užduotys [GV09]

Paveiksle matomos disciplinos turi skirtingą poveikį tapatybių valdymo sistemoms. Sociologija padeda apibrėžti tapatybę ir jos atitikmenį skaitmeninėje erdvėje, teisės mokslas nusako tapatybės duomenų naudojimo reikalavimus, o esamos technologijos formuoja sistemos įgyvendinimo niuansus. Verta pastebėti, kad tapatybės valdymo sistema gali atlikti ne visas diagramoje nurodomas funkcijas, o tik dalį iš jų.

1.3. Naudotojų poreikiai skaitmeninės tapatybės valdymo sistemoms

Skaitmeninės tapatybės valdymas yra plati sritis, kurią galima analizuoti iš skirtingų pusių: paslaugų tiekėjo, tapatybės tiekėjo ar naudotojo. Šiame darbe į skaitmeninių tapatybių valdymą žvelgiama iš naudotojo perspektyvos - kaip skaitmeninio valdymo sistemos atitinka naudotojų poreikius bei lūkesčius. Išskiriamos šios naudotojoms aktualios sistemų savybės:

- atpažinimo duomenų kiekis. Naudotojui vidutiniškai turint 25 paskyras, reikalaujančias slaptažodžių [FH07] bei naudojant nuo 2 iki 12-os el. paštų [GC07], jis tampa priverstas prisiminti vis daugiau slaptažodžių bei identifikatorių. Atsimintinų autentifikavimo duomenų kiekiui augant, naudotojai yra linkę aukoti saugumą dėl patogumo ir naudoti panašius slaptažodžius skirtingose sistemose [PM03; Sam99];
- saugumas. Suteikiant savo asmens duomenis internete naudotojai tikisi, kad jie bus patikimai saugomi ir nepasiekiami programišiams. Tapatybių valdymo sistemos turėtų būti budrios saugumo rizikoms bei viešai skelbti saugumui skirtas priemones ir atliktų saugumo analizių

rezultatus, kad tiek naudotojai, tiek paslaugų tiekėjai galėtų pasitikėti tapatybių valdymo sistemomis [DD08];

- asmens duomenų kontrolė. Naudotojai nesijaučia kontroliuojantys savo asmens duomenų internete [SSH+09]. Dėl to naudotojai pradeda nepasitikėti taikomųjų programų kūrėjais, nes jie pilnai nežino, kokia informacija apie juos kaupiama ir kokioms sistemoms ji perduodama;
- patogumas (angl. *usability*). Naudotojams skaitmeninės tapatybės valdymas neretai yra tik pašalinis mechanizmas, reikalingas norint pasiekti paslaugą [DD08]. Dėl šios priežasties sistemos naudojimosi patogumas yra svarbus - kuo tapatybės valdymas yra labiau integruotas su asmens jau naudojamomis sistemomis, kuo mažiau jis reikalauja papildomo naudotojo įsitraukimo ir kuo suteikia geresnę naudotojo patirtį (angl. *user experience*), tuo labiau naudotojas bus linkęs pasirinkti šį identiteto valdymo sprendimą.

1.4. Skaitmeninės tapatybės valdymo modeliai

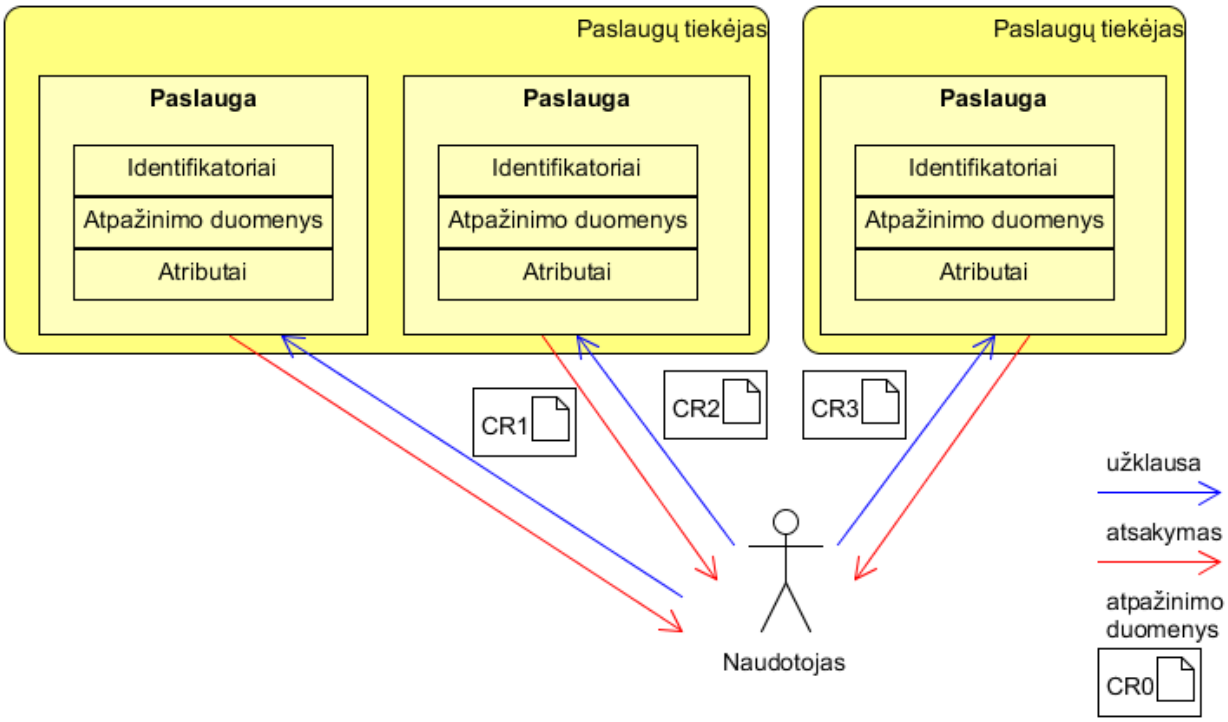
Naudojamų tapatybių valdymo sistemų architektūros bei veikimo principai yra skirtingi. S. Clauß ir M.Köhntopp savo tyrime pastebi, kad nėra vieningo standarto identiteto valdymo sistemoms [CK01]. Šiame skyriuje tiriami 3-ys dažniausiai naudojami identiteto valdymo modeliai. Tiriant kiekvieną modelį, pirmiausia apžvelgti jo bendri veikimo principai. Taip pat apžvelgtos paplitusios modelį įgalinančios technologijos (angl. *enabling technology*), nes modelio realizacijoje taikomi standartai ar protokolai gali turėti įtakos naudotojų poreikiams. Apžvelgus visus modelius, analizuotas jų atitikimas naudotojų lūkesčiams, išvardytiems 1.3 skyrelyje.

1.4.1. Izoliuotas tapatybės valdymas

Izoliuotame modelyje paslaugų tiekėjas yra ir tapatybės tiekėjas, nes visos su tapatybės valdymu susijusios operacijos yra atliekamos vieno serverio. Tapatybės duomenų saugojimas, autentifikavimas ir autorizavimas yra įgyvendinti paties paslaugų tiekėjo [CY10]. Kiekvienas naudotojas turi atskirus identifikatorius kiekvienai naudojamai paslaugai. Modelis grafiškai pavaizduotas 2-ame paveiksle.

Pagal izoliuotą modelį, naudotojas turi savo paskyrą kiekvienoje naudojamoje sistemoje. Kiekvieną kartą autentifikuojant ar autorizuojant naudotoją, tai atlieka pats paslaugų tiekėjas, bendraudamas tiesiogiai su naudotoju (jo naršykle). Naudotojui prisijungus prie vieno tinklalapio ir gavus prieigos raktą, jis gali toliau naudotis šiuo tinklalapiu, tačiau prireikus pasinaudoti kita taikomąja programa, tapatybės atpažinimo veiksmai (autentifikavimas, autorizavimas) turi vėl būti atlikti naujoje sistemoje.

Kadangi šis naudotojų autentifikavimo bei autorizavimo modelis naudojamas seniausiai, yra gana nemažai jį įgyvendinusių taikomųjų programų. Apsilankius keleto įmonių interneto tinklalapiuose, pastebėta, kad Lietuvoje naują paskyrą susikurti siūlo „Tiketa“, „Bilietai.lt“, „Pigu.lt“, „Varle.lt“, pasaulyje - „Booking.com“, „Skycop“, „AirBnB“, „CodinGame“ bei kitos platformos (išvardytų įmonių puslapiai tikrinti 2018 metų balandžio 14-ą dieną). Dalis iš jų remiasi ne vien tik savo izoliuotu tapatybės valdymu, bet jau turi į savo sistemas integravę ir papildomų autentifi-



2 pav. Izoliuotas skaitmeninės tapatybės valdymas [CY10]

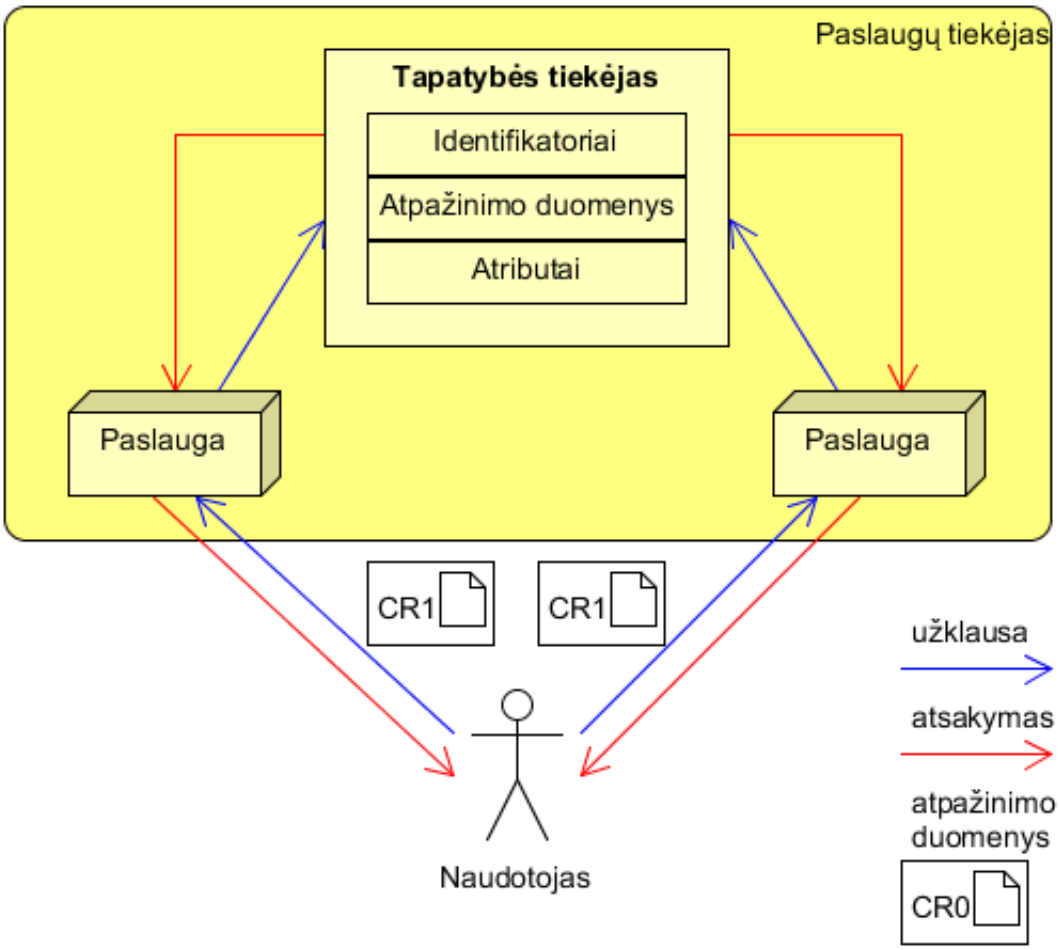
kavimo būdų (pvz. prisijungimą per „Facebook“ ar „Google“).

Realizacijų technologiniai sprendimai dažniausiai nėra viešai prieinami. Šiame modelyje kiekvienas paslaugų tiekėjas yra ir tapatybės tiekėjas, tad nereikia apibrėžti protokolų, duomenų formatų ar kitų detalių, kurios formalizuotų bendravimą tarp pasikliaujančiosios šalies ir tapatybės tiekėjo - visa tai pats nusprendžia ir įgyvendina paslaugų tiekėjas.

1.4.2. Centralizuotas tapatybių valdymas

Centralizuotame skaitmeninių tapatybių valdyme egzistuoja vienas tapatybės tiekėjas, į kurį kreipiasi visos paslaugos, esančios to paties paslaugų tiekėjo domene [JP05]. Kai paslaugų tiekėjui reikia autentifikuoti naudotoją (ar atlikti kitą tapatybės valdymo procesą), jis persiųs naudotojo pateiktus atpažinimo duomenis tapatybės tiekėjui, siekdamas pabaigti procesą [CY10]. Naudotojui šiame modelyje užtenka vienų atpažinimo duomenų, su kuriais jis gali prisijungti prie visų to paties paslaugų tiekėjo paslaugų. Modelio veikimas iliustruotas 3-iame paveiksle.

Centralizuotame modelyje paslaugų tiekėjo ir tapatybės tiekėjo funkcijos tampa atskirtos - tapatybės tiekėjas rūpinasi naudotojo identiteto valdymu, o paslaugų tiekėjas koncentruojasi į paslaugos vystymą. Tai sudaro patrauklesnes sąlygas naudotojui, tačiau taip pat sukuria vieno nesekmės taško (angl. *single point of failure*) sistemą. Tapatybės tiekėjo sistemai tapus nepasiekiamai, naudotojai nebegali naudotis nei viena paslauga tame pačiame domene.



3 pav. Centralizuotas skaitmeninės tapatybės valdymas [CY10]

Centralizuotas modelis tinkamiausias naudoti darbuotojams įmonės ribose arba vieno paslaugų tiekėjo paslaugoms [JP05]. Pateikiami pavyzdžiai abiems šioms realizacijoms.

Viena iš įmonėse naudojamų realizacijų centralizuotam tapatybės valdymui - katalogų prieigos protokolas (angl. *Lightweight Directory Access Protocol*, toliau LDAP), naudojamas pasiekti ir palaikyti informaciją tinkle [Kuk11]. Šis protokolas dažniausiai sujungiamas su aktyviąja direktorija (angl. *active directory*) ir leidžia laikyti kompanijos darbuotojų tapatybės informaciją vienoje vietoje. Taikomosios programos siunčia užklausas į LDAP serverį, kuriose nurodo norimą atlikti veiksmą (pvz. naudotojo autentifikavimą ar naudotojo atributų atnaujinimą).

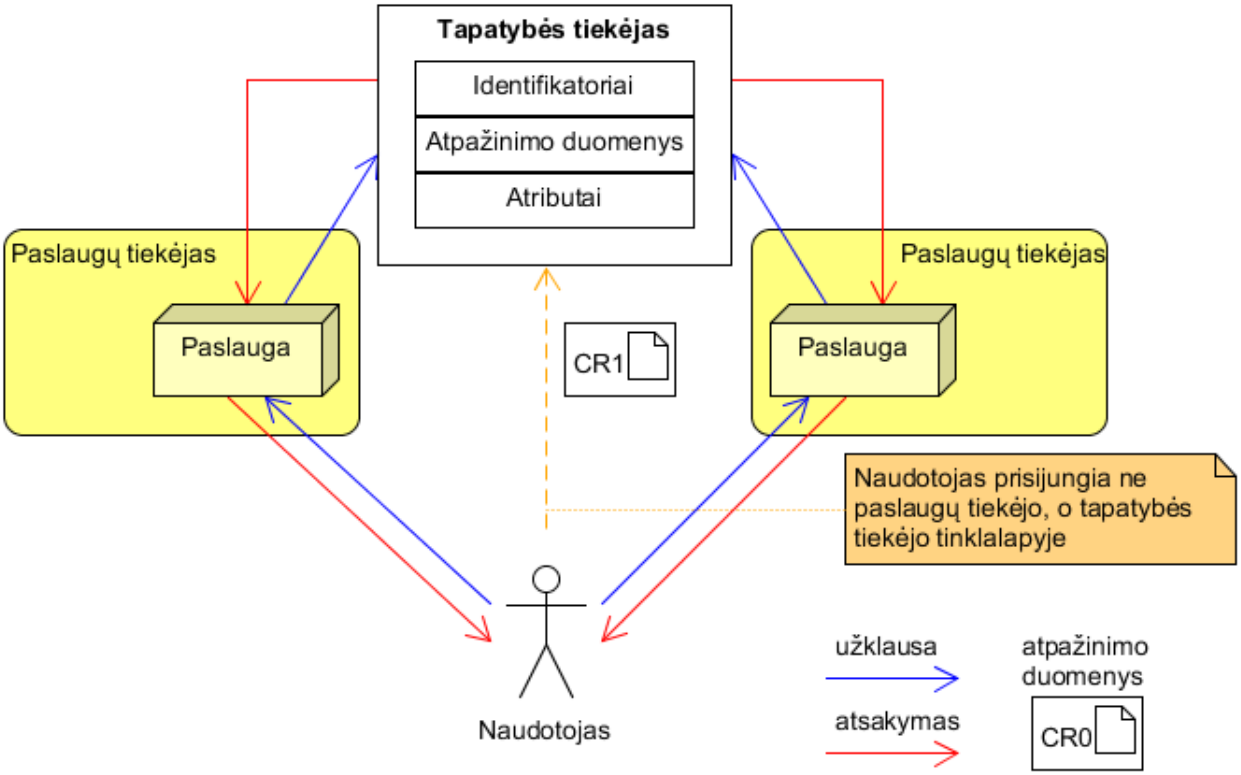
LDAP grįstas vienkartinis prisijungimas leidžia įmonės darbuotojams vieną kartą prisijungti prie tam tikros įmonėje naudojamos programos ir nebekartoti prisijungimo kreipiantis į kitą programą. Tačiau, tai galios tik toms programoms, kurios pasiekiamoms darbuotojams per vidinį intranetą [Kuk11]. Dėl šios priežasties LDAP grįstas centralizuotas tapatybės valdymas retai sutinkamas už įmonių intraneto ribų [Kuk11].

Centralizuotas tapatybės valdymas taip pat gali būti realizuotas ir ne įmonės ribose, jei konkretus paslaugų tiekėjas turi keletą paslaugų, skirtų naudotojams. Tokiu atveju jis gali turėti centralizuotą posistemę tapatybės valdymui, o naudotojui užtenka turėti vieną paslaugų tiekėjo paskyrą visoms įmonės paslaugoms. Tokios realizacijos pavyzdys - „Atlassian“ įmonės paslaugos. Nau-

dotojui pakanka turėti vieną „Atlassian“ paskyrą ir jis gali naudotis skirtingais šio paslaugų tiekėjo produktais, tokiais kaip „Jira“, „Confluence“, „Bitbucket“ bei kitais. Vieną kartą prisijungus prie „Atlassian“ programos (pvz. „Jira“), naudotojas tampa autentifikuotas ir kituose „Atlassian“ tinklalapiuose.

1.4.3. Jungtinis tapatybės valdymas

Ilgą laiką centralizuoto tapatybių valdymo pakako įmonėms turėti patogų, pačios įmonės prižiūrimą tapatybės valdymo sprendimą. Tačiau augant naudojamų taikomųjų programų bei integracijų su trečiųjų šalių aplikacijomis kiekiui, reikėjo sprendimo, leidžiančio identiteto valdymo uždavinius spręsti ne tik vienos organizacijos ribose. Todėl buvo pradėtas naudoti jungtinis (angl. *federated*) tapatybių valdymas.



4 pav. Jungtinis skaitmeninės tapatybės valdymas [CY10]

Jungtinis (angl. *federated*) tapatybių valdymas yra aibė technologijų ir procesų, kurie leidžia sistemoms dalintis tapatybės informacija ir deleguoti tapatybės valdymo užduotis tarp skirtingų paslaugų tiekėjų [MR08]. Šis tapatybių valdymo modelis įgalina naudotojus turėti vienus atpažinimo duomenis, kuriuos gali naudoti skirtingų paslaugų tiekėjų tinklalapiuose. 4 paveiksle pateikiama schema, vaizduojanti šio modelio architektūrą.

Jungtiniame tapatybės valdyme tapatybės tiekėjas yra atskira sistema, su kuria turi integruotis paslaugų tiekėjas. Tapatybės duomenys bei su tapatybe susiję veiksmai yra deleguojami šiai sistemai. Naudotojas turi vieną identifikatorių, su kuriuo prisijungia tiesiogiai tapatybės tiekėjo

puslapyje. Prisijungus šioje sistemoje, naudotojas tampa autentifikuotas visose paslaugose, kurios palaiko šį tapatybės tiekėją [MR08].

Kadangi jungtiniame tapatybės valdyme tapatybės tiekėjas bei paslaugų tiekėjas yra skirtingų kūrėjų sistemos, duomenų apsikeitimui tarp jų sukurti technologiniai standartai. Trys šiuo metu labiausiai paplitę protokolai: „SAML“, „OAuth“ bei „OpenID“ [Kou17]. Šių technologijų apžvalga pateikiama 1-oje lentelėje.

Kiekviena iš minimų technologijų šiandien yra gana plačiai naudojama internete - tai rodo ir didžiųjų kompanijų („Google“, „GitHub“, „Microsoft“) sprendimai pritaikyti jas savo programinėje įrangoje. Pagrindinis šių standartų skirtumas - jų panaudojimo apimtis. „SAML“ pritaikytas visiems tapatybės valdymo veiksmams, „OAuth“ skirtas naudotojui autorizuoti trečiąją šalį pasiekti jo atributus tapatybės tiekėjo platformoje, o „OpenID“ skirtas naudotojų autentifikavimui.

1 lentelė. Jungtinio tapatybės valdymo technologijos

	SAML	**OAuth**	**OpenID**
Dabartinė versija	SAML 2.0	OAuth 2.0	OpenID Connect
Paskirtis	Autentifikavimas, autorizavimas, atributų perdavimas	Autorizavimas	Autentifikavimas
Duomenų perdavimas	HTTP, SOAP	HTTP, REST	HTTP, REST
Duomenų formatas	XML	JSON, JWT	JSON, JWT
Duomenų šifravimas	Yra	Yra	Yra
Tapatybės tiekėjo suteiktų duomenų validavimas	Viešo-privataus rakto infrastruktūra	Neapibrėžta (palikta realizacijai)	Viešo-privataus rakto infrastruktūra
Naudotojo sutikimas perduoti duomenis	Nėra	Yra	Yra
Mobiliųjų programėlių palaikymas	Nėra	Yra	Yra
Naudojančios organizacijos	Salesforce, PingFederate, Oracle Access Manager	Google, Amazon, GitHub	Google, Microsoft, Ping Identity

1.4.4. Skaitmeninės tapatybės valdymo modelių palyginimas

Apžvelgus skirtingus skaitmeninės tapatybės valdymo modelius, 2 lentelėje pateikiamas jų palyginimas. Skirtingi modeliai lyginti pagal 1.3 skyrelyje apibrėžtus naudotojų poreikius.

Atsižvelgus į palyginimo rezultatus, galima teigti, kad parankiausias naudotojams yra jungtinis tapatybės valdymo modelis. Centralizuotas bei izoliuotas tapatybės valdymas verčia naudotojus kurti ir administruoti naujas paskyras nebandytose sistemose, prisiminti daugybę slapyvardžių bei slaptažodžių (ar silpninti saugumą naudojant tą patį slaptažodį) bei kartoti prisijungimo žingsnius, o tai įkyri naudotojams. Pasak kompanijos „Blue Research“ tyrimo, tinklalapiui prašant kurti naują

paskyrą 54% interneto naudotojų teigia paliksiantys ar negrįšiantys į tokį puslapį, o 26% ieškos kito panašią paslaugą teikiančio tinklalapio, nereikalaujančio naujos paskyros.

2 lentelė. Skirtingų tapatybės valdymo modelių palyginimas pagal naudotojo poreikius

		Izoliuotas	**Centralizuotas**	**Jungtinis**
Atpažinimo duomenys		Kiekvienai paslaugai	Kiekvienam paslaugų tiekėjui	Kiekvienam tapatybės tiekėjui
Patogumas	Prisijungimų kiekis	Tiek, kiek paslaugų	Tiek, kiek paslaugų tiekėjų	Tiek, kiek tapatybės tiekėjų
	Naudotojo patirties vientisumas	Yra	Yra	Nėra
Saugumas	Nukreipimai (angl. redirects)	Nėra	Galimi	Yra
	Vienas nesėkmės taškas	Nėra	Paslaugų tiekėjo tapatybės valdymo modulis	Tapatybės tiekėjas
	Technologiniai standartai	Paslaugos kūrėjo nuožiūra	Paslaugų tiekėjo nuožiūra	SAML 2.0, Oauth 2.0, OpenID Connect
Kontrolė	Naudotojo patvirtinimas duomenis perduodant kitai paslaugai	-	Nėra	Galimas (Oauth 2.0, OpenID Connect)
	Duomenų suteikimas tik jų prireikus	Tai realizavus paslaugos kūrėjui	Tai realizavus paslaugų tiekėjui	Dalinis (dar nenaudotai paslaugai prašant naudotojo patvirtinimo)
	Tapatybės duomenų keitimas	Kiekvienos paslaugos tinklalapyje	Kiekvieno paslaugų tiekėjo tinklalapyje	Kiekvieno tapatybės tiekėjo tinklalapyje

1.5. Naudotojų problemų tapatybės valdyme apibendrinimas

Skaitmeninės tapatybės valdymas taikant jungtinį modelį išsprendžia nemažai problemų: sumažintas naudotojams reikalingų paskyrų kiekis, įsimintinų slapyvardžių ir slaptažodžių gausa, apribotas reikiamų prisijungimų kiekis. Nepaisant to, nemažai problemų naudotojams išlieka. Pagrindinės iš jų:

- vienintelio nesekmės taško situacija. Jungtiniame tapatybės valdyme tapatybės tiekėjas yra vienintelis nesekmės taškas (angl. *single point of failure*). Tapatybės tiekėjui tapus nepasiekiamam, asmuo internete nebegali naudoti visų paslaugų, kurios buvo integruotos su šiuo tapatybės tiekėju.
- kontrolės trūkumas. Organizacijos renka didžiulius kiekius asmens duomenų - „Facebook“ nuo sukūrimo jau sukaupė 300 petabaitų duomenų apie naudotojus [VW14], tačiau naudotojai nėra pakankamai informuoti, kur perduodami jų duomenys. Nors naudojamos technologijos, tokios kaip „OAuth“, išreikštinai prašo naudotojų sutikimo perduodant duomenis naujai paslaugai, tai atliekama tik pradedant naudotis tam tikra paslauga. Taip pat, *CambridgeAnalytica* incidentas [Gra18] rodo, kad šio sutikimo gali neužtekti. Visa tai veda prie to, kad naudotojai jaučiasi nekontroliuojantys savo asmens duomenų internete, norėtų lengviau keisti bei trinti juos, o dalį tapatybės valdymo sistemų naudoja tik todėl, kad neturi kitos išeities [Baa16];
- pasitikėjimo stoka. Naudotojams jungtiniame tapatybės valdyme tenka pasikliauti tiek tapatybės tiekėjų, tiek paslaugų tiekėjų gebėjimu patikimai saugoti ir perduoti tapatybės duome-

nis. Norint naudotojų pasitikėjimo, sistemos turėtų būti skaidrios (angl. *transparent*) apie tai, kaip jos saugo, valdo tapatybės duomenis [Baa16]. Tačiau, kai technologijų specifikacijos palieka vietos nesaugiam protokolų įgyvendinimui [SB12], o paslaugų tiekėjai retai atskleidžia sistemos veikimo detales [Baa16], naudotojų pasitikėjimas senka;

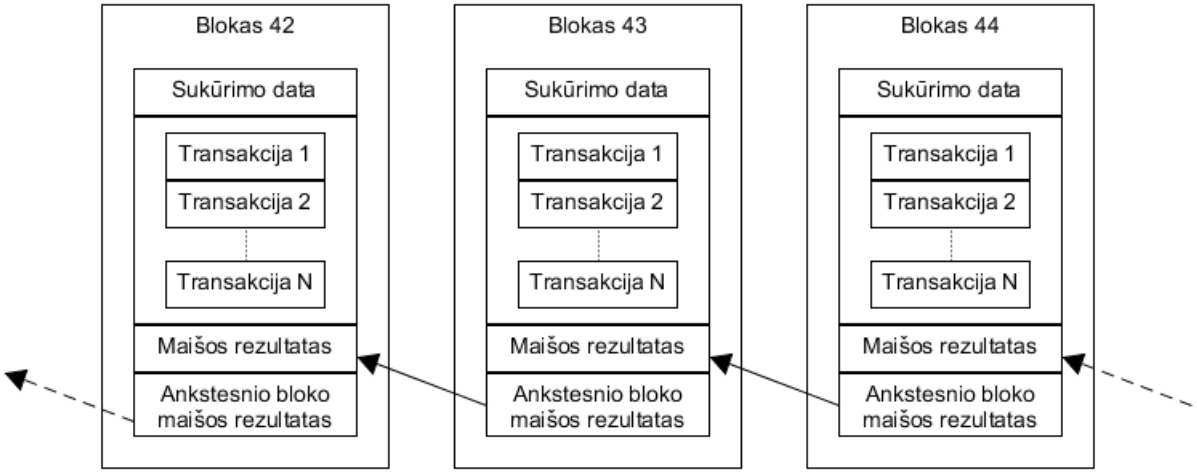
- saugumo iššūkiai. Dėl naudojamų nukreipimų jungtinis tapatybės valdymas yra itin jautrus sukčiavimo (angl. *phishing*) atakoms, tačiau paslaugų tiekėjai dažniausiai vis dar remiasi naudotojų gebėjimu atpažinti netikrus programišių tinklalapius, nors to nerekomenduoja tyrimai [SPM+11]. Taip pat, jungtiniame tapatybės valdyme atsiranda vienas didelės vertės taikinys (angl. *single high value target*) - tapatybės tiekėjas. Įsilaužus į jo sistemą, naudotojų atpažinimo duomenys gali būti pavogti ir panaudoti daugybėje skirtingų paslaugų tiekėjų sistemų.

Pateikti jungtinio tapatybės valdymo trūkumai leidžia teigti, kad tapatybės valdymo sritis vis dar yra tobulintina naudotojų atžvilgiu. Asmens duomenų saugumui taikomųjų programų kūrėjų neretai neskiria pakankamai dėmesio sistemos kūrimo pradžioje (angl. *privacy by design*). Naudotojams trūksta didesnės tapatybės duomenų kontrolės, o dėl galimų saugumo iššūkių, vieno nesekmės taško situacijos ir neatskleidžiamų sistemų įgyvendinimo detalių, pasitikėti paslaugų bei tapatybės tiekėjais yra sudėtinga.

Toliau darbe apibūdinama blokų grandinės technologija. Apžvelgus jos savybes, tiriamas jų potencialas spręsti įvardytas problemas skaitmeninės tapatybės valdyme: vieno nesėkmės taško situaciją bei kontrolės, skaidrumo ir pasitikėjimo trūkumą.

2. Blokų grandinės technologija

Blokų grandinė - tai vieno su kitu susijusių blokų grandinė, kurios blokuose saugomi nekeičiami įrašai [Nak08]. Šią technologiją galima apibūdinti kaip daugybę paskirstytų nekintamų skaitmeninių įrašų (angl. *immutable distributed ledger*), tarpusavyje susietų taikant kriptografiją (blokų grandinės pavyzdys pateikiamas 5 paveiksle). Technologija geriausiai žinoma dėl jos panaudojimo „Bitcoin“ kriptovaliutoje. Šiame skyriuje apžvelgiami pagrindiniai blokų grandinės techniniai aspektai, savybės bei galimi skirtingi variantai.



5 pav. Supaprastintas blokų grandinės modelis

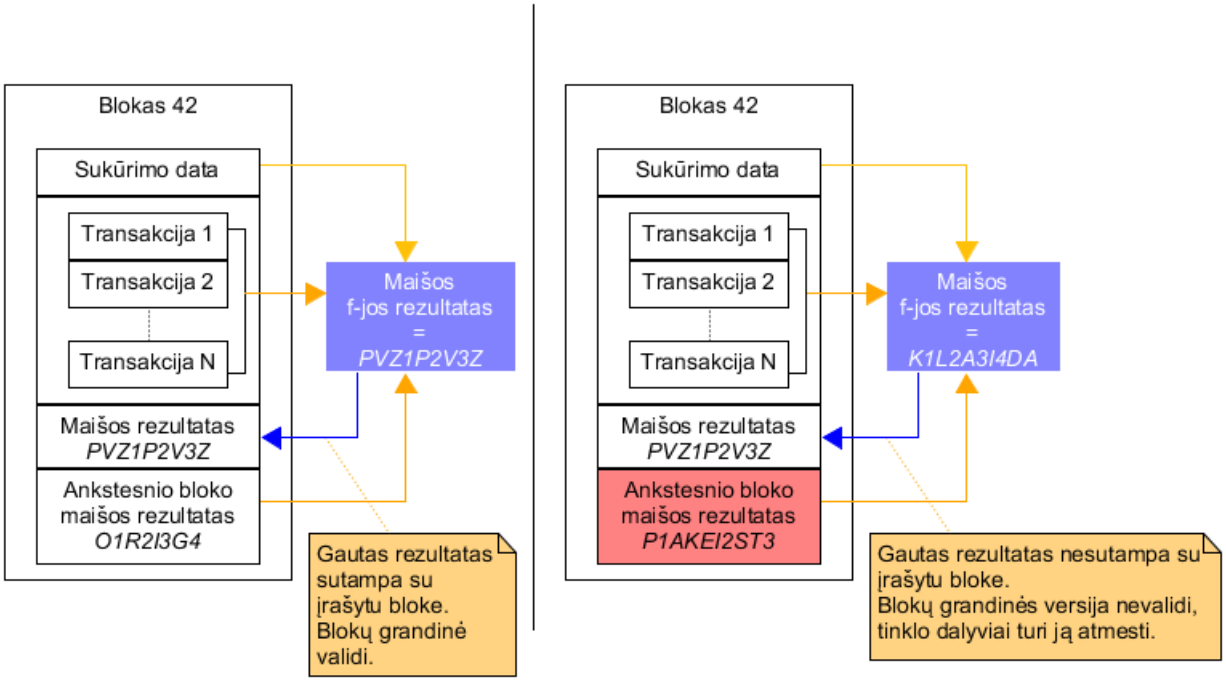
2.1. Nekintamumas

Blokų grandinėje kiekvienas blokas yra sudarytas iš šių dalių:

- transakcijų. Kiekviena transakcija yra duomenys, kuriuos norima saugoti blokų grandinėje. Šie duomenys gali būti bet kokia vertinga informacija: pervesti pinigai, programinis kodas, skaitmeninės tapatybės atributai ar kt. Kiekviena transakcija yra pasirašoma kūrėjo privačiu raktu. Vienas blokas gali turėti vieną arba daugiau transakcijų;
- bloko kriptografinės maišos funkcijos rezultato (angl. *hash*);
- ankstesnio (tėvinio) bloko kriptografinės maišos funkcijos rezultato;
- bloko sukūrimo laiko. Blokai grandinėje saugomi chronologiškai;
- kitų metaduomenų (pvz. bloko eilės numerio, blokų grandinės versijos, *nonce* darbo įrodymui).

Kiekvieno bloko maišos funkcijos rezultatas priklauso nuo jo transakcijų, prieš tai buvusio bloko maišos rezultato ir bloko metaduomenų. Jeigu betkurio bloko duomenys būtų pakeisti, tuomet maišos funkcija sugeneruotų kitokį maišos rezultatą ir būtų lengva patikrinti, kad naujai perskaičiuotas maišos rezultatas nesutampa su bloke esančiu rezultatu. Taip pat, kadangi kiekvienas blokas priklauso nuo prieš tai buvusio bloko, net ir pakeitus vieną iš pirmųjų blokų, pakeitimas

būtų pastebimas pridedant naujus blokus ir būtų galima suprasti, kad turima blokų grandinės versija yra nevalidi (žr. 6 pav.). Tokiu būdu kiekvienas blokų grandinės blokas patvirtina prieš tai buvusio bloko integralumą, taip pasiekiant blokų grandinės nekintamumą (angl. *immutability*), nes perrašyti įrašus blokuose nepastebėtam labai sunku [Nak08].



6 pav. Bloko grandinėje validavimas

2.2. Decentralizuotumas

Blokų grandinės sistema yra decentralizuota - nėra vieno centrinio serverio, kuris vienas turėtų visą blokų grandinę. Sistemą sudaro daugybė blokų grandinės mazgų (angl. *node*), kurie turi visą blokų grandinės kopiją. Šie mazgai yra atsakingi už naujų transakcijų validavimą, blokų su transakcijomis kūrimą, sukurtų blokų priėmimą į blokų grandinę ir pranešimus kitiems mazgams apie naują į grandinę priimtą bloką [Ant14]. Kiekvienas mazgas yra susietas su keletu kitu mazgų. Mazgas, kuris nori pridėti naują bloką (vadinamas *kasėju*), praneša apie jį kitiems mazgams, jie savo ruožtu žinią perduoda kitiems mazgams ir taip ilgainiui kiekvienas mazgas tinkle turi naujausią blokų grandinės versiją.

Kadangi nėra centrinės institucijos, kuri nuspręstų, ar siūlomas blokas yra tinkamas priimti į grandinę, sprendimą bendrai turi priimti visi tinklo dalyviai. Egzistuoja skirtingos taisyklės, vadinamos konsensuso strategijomis (plačiau apie juos 2.4 skyrelyje), kuriomis remdamiesi tinklo mazgai nusprendžia, ar pasiūlytas blokas yra validus. Šios taisyklės apibrėžia, kaip tinklo dalyviai turi įrodyti bloko validumą jį siūlydami į grandinę bei kaip patikrinti kito dalyvio pasiūlyto bloko validumą.

2.3. Tipai

Priklausomai nuo tinklo dalyviams suteikiamų blokų grandinės skaitymo ir rašymo teisių, išskiriami trys pagrindiniai blokų grandinės tipai: vieša, konsorciumo bei privati. Tipų skirtumai pateikiami 3 lentelėje.

3 lentelė. Viešos, konsorciumo ir privačios blokų grandinės palyginimas [ZXD+17]

	Vieša	**Konsorciumo**	**Privati**
Konsensuso nustatymas	Visi kasėjai	Išrinkti tinklo dalyviai	Viena organizacija
Skaitymo teisės	Viešos	Gali būti viešos ar apribotos	Gali būti viešos ar apribotos
Centralizuotumas	Nėra	Dalinis	Yra
Efektyvumas	Mažas	Didelis	Didelis

Kadangi vieša blokų grandinė yra atvira visam pasauliui, visiems matomos ir joje išsaugotos transakcijos. Tai sudaro puikias salygas įrašų auditui, tačiau sumažina naudotojų privatumą. Siekiant išlaikyti tam tikrą privatumo lygį, viešoje grandinėje matomi tik transakcijas atlikusių asmenų vieši raktai [Nak08]. Viešų blokų grandinių pavyzdžiai: „Bitcoin“, „Ethereum“ kriptovaliutų blokų grandinės.

Privačios bei konsorciumo blokų grandinės yra tik dalinai decentralizuotos - jose blokų validavimą ir priėmimą į grandinę atlieka vienas ar dalis tinklo dalyvių. Šios grandinės privalumai: visi validuotojai yra žinomi, grandinės efektyvesnės dėl greičiau priimamų blokų, apribotos blokų skaitymo teisės suteikia didesnį privatumo lygį, o iškilus poreikiui, tinklo dalyviai gali pakeisti ar atšaukti įvykusias transakcijas [But15a]. Konsorciumo ir privačios blokų grandinės labiau tinkamos įmonių vidiniam (ar jungtiniam, pvz. tarp kelių finansinių institucijų) naudojimui. Blokų grandinių karkasų sprendimus įmonėms siūlo „IBM“, „Microsoft“, „Amazon“ [ZXD+17].

2.4. Konsensuso strategijos

Kadangi blokų grandinės sistema yra decentralizuota, nėra centrinės institucijos, kuri nuspręstų, ar naujai siūlomas pridėti į grandinę blokas yra validus (be transakcijų su falsifikuotais duomenimis). Todėl blokų grandinės tinkle taikoma konsensuso strategija, pagal kurią nusprendžiama, ar pridėti naują bloką į grandinę. Apžvelgiamos trys dažnai naudojamos strategijos: darbo, įtakos bei autoriteto įrodymo.

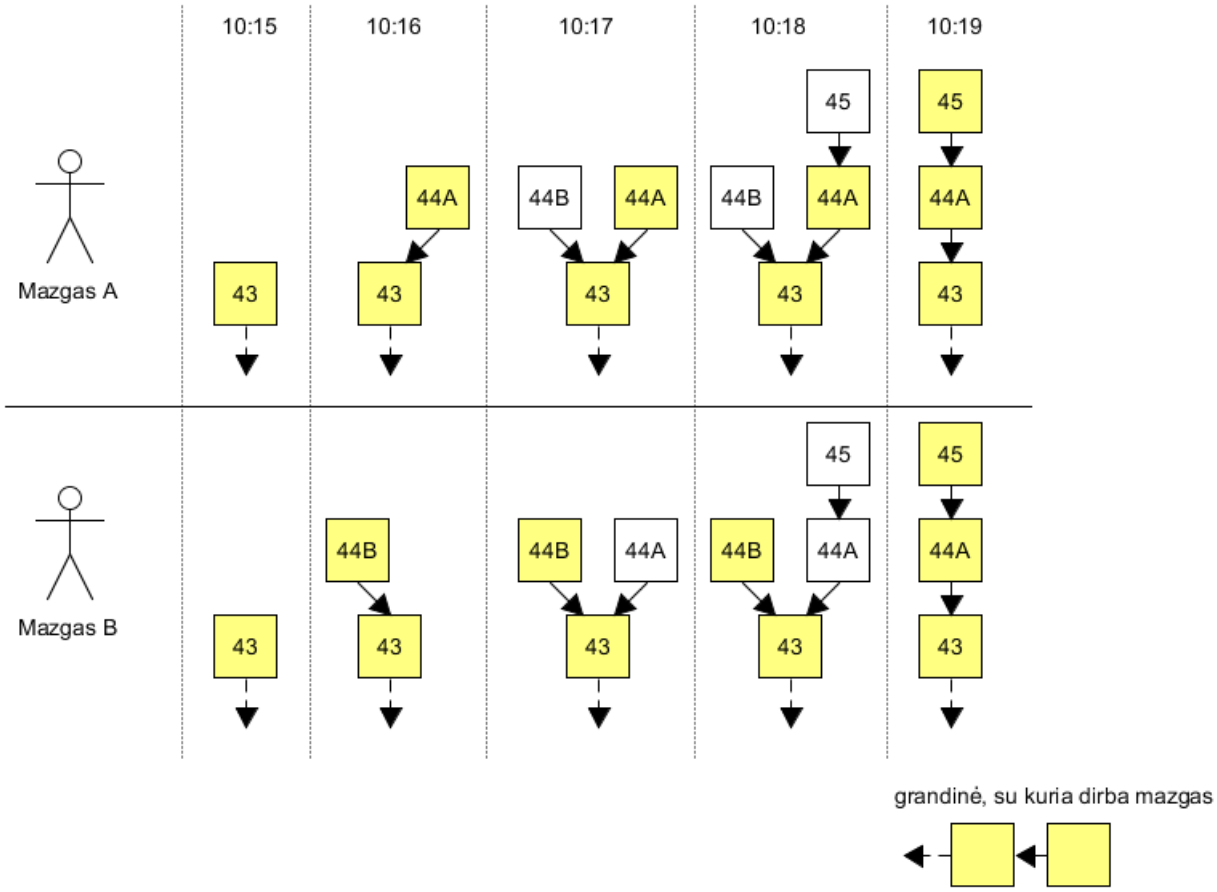
2.4.1. Darbo įrodymo strategija

Darbo įrodymo konsensuso strategija remiasi principu, kad daug pastangų ir resursų į bloko validumo įrodymą įdėjęs tinklo dalyvis nebus linkęs sukčiauti. Šioje strategijoje tinklo dalyvis, norėdamas pridėti bloką į blokų grandinę, turi išspręsti laikui ir resursams imlų matematinį uždavinį (užsiima *bloko kasimu*). Pirmas uždavinio reikšmę radęs tinklo dalyvis praneša apie ją kitiems,

kurie turi patvirtinti, ar ši reikšmė teisinga. Jei tai patvirtinta, tinklo dalyviai patikrina, ar naujojo bloko transakcijos yra validžios. Jeigu jos validžios, blokas pridedamas į grandinę [ZXD+17].

Darbo įrodymo matematinis uždavinys dažniausiai būna paremtas kriptografine maišos funkcija, kurios rezultatą lengvą validuoti, tačiau duomenis, sugeneravusius šį rezultatą, sunku surasti. Uždavinio tikslas - surasti šiuos duomenis. Tinklo dalyviai eikvoja didžiulius kiekius elektros energijos ir laiko, nes radimas būna paremtas duomenų perrinkimu (angl. *brute force*). Dėl šios priežasties rezultato ieškantiems *kasėjams* neretai būna įvesta paskatinimo sistema, kuri teisingą reikšmę radusį *kasėją* apdovanoja piniginiu atlygiu [Nak08].

Kadangi blokų grandinės tinklas yra decentralizuotas, įmanoma situacija, kad labai panašiu metu į grandinę skirtingų mazgų pridėti du validūs blokai. Taip dalis tinklo dalyvių gaus vieną mazgą, o dalis - kitą, o abu jie bus susieti su tuo pačiu prieš tai buvusiu bloku. Tokiu atveju, taikoma ilgiausios grandinės taisyklė (žr. 7 pav.). Mazgai dirba prie pirmiau gauto bloko, tačiau išsaugo kitą gautą bloką kaip šaką. Po to, kai bus gautas dar vienas blokas, jis bus susietas tik su viena iš šakų - taip ši šaka taps ilgesnė. Tuomet ilgesnė šaka paskelbiama aktyviąja grandine, visi su trumpesniąja šaka dirbę mazgai turi pereiti prie aktyviosios grandinės, o atmesto bloko (vadinamo *bloku-našlaičiu*) transakcijos grąžinamos į bendrą transakcijų sankaupą (angl. *transaction pool*) [Nak08]. Realiuose blokų grandinės taikymuose, dažnai laukiama keleto iš eilės einančių naujų blokų, kol *blokas-našlaitis* yra atmetamas. Pavyzdžiui, „Bitcoin“ blokų grandinėje tik sulaukus apytiksliai 6 blokų iš eilės *blokas-našlaitis* gali būti atmestas[ZXD+17].



7 pav. Ilgiausios grandinės taisyklės taikymas

Pagrindinis šios konsensuso strategijos privalumas yra tas, kad dideli kasimo kaštai gali atgrasyti programišius nuo potencialių atakų. Tačiau, taip veltui išeikvojama daugybė elektros energijos - skaičiuojama, kad kasimas „Bitcoin“ ir „Ethereum“ blokų grandinėms kartu sudėjus per dieną sueikvoja elektros energijos, kurios vertė yra apie 1 milijoną dolerių [Eth14]. Taip pat, dėl ilgo uždavinio sprendimo laiko vieno bloko priėmimas į grandinę gali užtrukti - Bitcoin blokų grandinėje tai užima apie 10 minučių [ZXD+17].

2.4.2. Turto įrodymo strategija

Turto įrodymo konsensuso strategija remiasi principu, kad daug blokų grandinės turto turintis kasėjas bus sąžiningas, nes išaiškinus jo nesąžiningumą jis rizikuoja prarasti savo turimą turtą [Baa16]. Šis algoritmas patiki sprendimą priimti tiems tinklo dalyviams, kurie įrodo kad turi daugiausia turto (pvz. blokų grandinės kriptovaliutos). Remiantis tik šia savybe, tokia strategija gali pasirodyti kaip ne itin sąžininga, nes turtingiausias tinklo dalyvis gali būti vienvaldžiu sprendimų priėmėju. Dėl to blokų grandinės tinklai neretai taiko patobulintus šios strategijos variantus: „Peercoin“ papildomai vertina turto amžių, „Blackcoin“ kitą patvirtintoją paskiria pagal atsitiktinę funkciją, kuri atsižvelgia ir į turimą turtą [ZXD+17].

Ši strategija leidžia nebeeikvoti didžiulių elektros kiekių, skirtingai nei darbo įrodymo strategija [Eth14]. Algoritmo efektyvumas taip pat sutaupo laiko ir blokai būna greičiau patvirtinami

ir pridedami į grandinę. Tačiau, dėl praktiškai nulinių bloko *kasimo* sąnaudų, galimos dažnesnės tinklo atakos [ZXD+17].

2.4.3. Autoriteto įrodymo strategija

Autoriteto įrodymo strategija remiasi keleto tinklo dalyvių, kuriems suteikta teisė validuoti naujus blokus, priimamais sprendimais. Ši strategija nebėra tinkama visiškai decentralizuotai blokų grandinei, kurioje būtinas pilnas pasitikėjimo padalinimas [Hos17]. Tačiau ši strategija tinkama privačioms ar konsorciumo blokų grandinėms.

Šis konsensuso mechanizmas remiasi iš anksto išrinktais tinklo dalyviais, kurie bus atsakingi už blokų validavimą. Kiekvieną kartą pridedant naują bloką į grandinę, vienas iš išrinktų validuotojų patvirtins arba atmes pasiūlytą bloką. Siekiant sumažinti galimą kenksmingų patvirtintojų žalą, įvedamos taisyklės, neleidžiančios tam pačiam validuotojui patvirtinti keleto blokų iš eilės [Hos17].

Autoriteto įrodymo strategijoje validuotojams svarbu išlaikyti gerą reputaciją - susigadinus ją, validuotojas gali būti pašalintas iš tinklo. Šis konsensuso mechanizmas leidžia greitai ir su itin mažais ištekliais pridėti blokus, tačiau nėra tinkamas pilnai decentralizuotoms blokų grandinėms. Šią strategiją taiko „Parity“ blokų grandinė [Hos17].

2.5. Išmanieji kontraktai

Išmanusis kontraktas (angl. *smart contract*) - tai kompiuterizuotas transakcijos protokolas, kuris išpildo kontrakto sąlygas, jei patenkinami numatyti kriterij [Sza97]. Išmanųjį kontraktą galima apibūdinti kaip tam tikras sąlygas (išreikštas kodu), kurios, patenkinus reikalavimus, bus įvykdytos automatiškai, be trečiosios šalies priežiūros.

Idėja, pristatyta dar prieš 2000-uosius, šiais laikais yra įgyvendinama pasitelkiant blokų grandinę. Išmanusis kontraktas į blokų grandinę yra patalpinamas kaip transakcija. Tuomet, dėl blokų grandinės viešumo, kodo logika būna atskleista ir kiekvienas norintis gali įsitikinti, ar kontraktas veiks tinkamai. Taip pat, kadangi pats išmanusis kontraktas yra blokų grandinės transakcijoje, jis nebegali būti keičiamas. Tai padidina naudotojų pasitikėjimą, nes jokia centrinė institucija negali pakeisti kontrakto sąlygų. Tačiau, jei kode palikta defektų, jų nebebus galima ištaisyti.

Blokų grandinėje esančių išmaniųjų kontraktų kodas yra vykdomas automatiškai tinkle esančių kasėjų [ZXD+17]. Taip nebelieka vienintelio nesekmės taško situacijos, kai kodą vykdytų vienas centrinis serveris - išmaniojo kontrakto kodas butų nebevykdomas tik tada, jei visi tinklo mazgai išeitų iš tinklo. Už tai, kad tinklo dalyviai naudoja savo kompiuterių resursus išmaniųjų kontraktų kodui vykdyti, jiems dažniausiai suteikiamas piniginis paskatinimas.

Išmaniųjų kontraktų veikimo detalės priklauso nuo blokų grandinės, kurioje jie bus vykdomi. Bene žinomiausia išmaniųjų kontraktų kūrimui pritaikyta platforma - „Ethereum“. 2.5.1 skyrelyje apžvelgiamos pagrindinės išmaniųjų kontraktų veikimo sąlygos.

2.5.1. „Ethereum“ išmanieji kontraktai

2015-aisias pradėjusi veikti „Ethereum“ platforma pristatė blokų grandinę, kuri pritaikyta kurti būseną galinčius keisti išmaniuosius kontraktus su išbaigta (angl. *turing complete*) programavimo kalba [Eth15a]. „Ethereum“ kontraktai vykdomi tinklo dalyvių, „Ethereum“ virtualiosios mašinos pagalba. „Ethereum“ virtualioji mašina gali būti suprantama kaip didelis decentralizuotas kompiuteris, saugantis milijonus objektų, vadinamų paskyromis, kurie turi galimybę turėti savo duomenis, vykdyti kodą bei bendrauti tarpusavyje [Eth14].

„Ethereum“ platforma turi paskyras, kuriomis naudojantis, galima kurti transakcijas bei kviesti išmaniųjų kontraktų funkcijas. Paskyros identifikuojamos pagal unikalų adresą. „Ethereum“ egzistuoja dviejų tipų paskyros [Eth14]:

1. išorinės paskyros (angl. *externally owned accounts*). Jos savininkų valdomos privačiais raktais ir turi eterio balansą;
2. kontraktų paskyros (angl. *contract accounts*). Šios paskyros be balanso turi kodą bei būseną, kuri gali keistis, jei įvykdomos kode aprašytos sąlygos. Kontraktų paskyros yra valdomos kontrakto kodo.

Skatinant tinklo dalyvius vykdyti kontraktų kodą, jiems už šį darbą skiriama piniginė premija - eterio kriptovaliuta. Kiek eterio bus skiriama, priklauso nuo vykdomo kodo sudėtingumo ir jam reikalingų resursų. Dėl to kiekviena kodo operacija „Ethereum“ platformoje yra įvertinama kuro (angl. *gas*) kiekiu, kuris nurodo sąlyginę kainą. Kuro sistema padeda apsisaugoti nuo begalinių ciklų vykdymų - jei vykdant funkciją baigiasi kuras, vykdymas nutraukiamas, o kuras kvietėjui nėra grąžinamas [Eth15a].
Skiriamos dviejų tipų funkcijos: rašymo bei skaitymo. Skaitymo funkcijų kvietimas yra nemokamas - jos tik grąžina esamus kontrakto duomenis. Rašymo funkcijos yra mokamos, nes jos keičia kontrakto būseną. Jų vykdymo metu turi būti sukurta transakcija, kurią patvirtins „Ethereum“ blokų grandinės dalyviai. Kuo didesnis duomenų kiekis bus išsaugotas būsenoje, tuo brangesnė bus operacija.

Funkcijos kvietimą apmoka kvietėjas. Dėl šios priežasties kvietėjas savo paskyroje turi turėti eterio, iš kurio galės padengti kviečiamos funkcijos išlaidas. Norint neversti kvietėjo apmokėti, kontrakto kodas kviečiamos funkcijos pabaigoje gali į kvietėjo paskyrą pervesti jo išeikvotą eterį. Tolimesnėse „Ethereum“ versijose svarstoma įdiegti funkcionalumą, kuris standartizuotų būdą, leisiantį kontraktams patiems apmokėti iškviestų funkcijų išlaidas [But15b].

Bendravimui su vartotojui skirtomis klientinėmis programomis, komunikuojančiomis su blokų grandine (neretai vadinamoms *dapp*), „Ethereum“ platformoje dažniausiai naudojami įvykiai (angl. *events*). Išmaniajame kontrakte galima apibrėžti įvykį ir jo grąžinamus duomenis. Į kontrakto funkcijas tuomet pridedamas kodas, kuris paskelbs įvykį su pasirinktais duomenimis. Visos klientinės programos, besiklausančios šio įvykio, bus informuotos ir gaus įvykyje perduotus duomenis. Taip klientinė programa gali reaguoti į išmaniajame kontrakte įvykusius pokyčius.
Įvykiai „Ethereum“ blokų grandinėje saugomi kaip registrai (angl. *logs*), o paskelbti įvykius kainuoja gerokai mažiau kuro, nei keisti kontrakto būseną. Dėl šios priežasties įvykius galima naudoti

kaip pigesnę alternatyvą duomenims blokų grandinėje saugoti. Tačiau, įvykiuose paskelbti duomenys nėra prieinami iš išmaniojo kontrakto kodo [Eth15a].

2.6. Pavojai ir trūkumai

Blokų grandinės technologija sukuria sąlygas decentralizuotai, nesuteikiant pasitikėjimo vienam ar keliems dalyviams, laikyti nekeičiamus duomenis. Tai atveria įvairių galimybių finansų, daiktų interneto, reputacijos sistemų ar saugumo srityse [ZXD+17], tačiau ši technologija turi ir trūkumų, kurių galimą poveikį būtina įvertinti prieš taikant šią technologiją konkrečioje sferoje. Pagrindinės grėsmės ir trūkumai aptariami šiame skyriuje.

2.6.1. Daugumos ataka

Viešose blokų grandinėse dauguma (>50%) mazgų tinkle turi patvirtinti bloką, kad jis būtų priimtas į grandinę. Potencialus įsilaužėlis gali pateikti falsifikuotą blokų grandinės bloką (pvz. su netikromis transakcijomis), tačiau kol jis neturi daugumos skaičiavimo galios tinkle, šis blokas bus atmestas likusių dalyvių mazge (ir taps *bloku-našlaičiu* taikant ilgiausios grandinės taisyklę). Tačiau, jeigu įsilaužėlis (ar keletas jų) turi daugumą skaičiavimo galios, jis gali dirbti su falsifikuota blokų grandinės versija greičiau negu likę dalyviai tinkle ir taip ilgiausia grandine, prie kurios pereis visi dalyviai, taps jo sukurta grandinė su falsifikuotais blokais [ZXD+17]. Ši ataka nebuvo įgyvendinta prieš žinomiausias „Bitcoin“ bei „Ethereum“ blokų grandines, tačiau buvo įvykdyta prieš „Verge“ blokų grandinę [Sed18].

2.6.2. Plečiamumas

Blokų grandinės plečiamumas matuojamas pagal du kriterijus: transakcijų pralaidumą ir saugojimo reikalavimus mazgams. Transakcijų pralaidumas, kuris priklauso nuo to, kaip greitai nauji blokai yra pridedami į blokų grandinę, susijęs su taikoma konsensuso strategija. Kuo taikoma strategija leidžia greičiau priimti naują bloką, tuo greičiau transakcijos bus patvirtintos. „Bitcoin“ kriptovaliutos blokų grandinėje, taikančioje darbo įrodymo konsensusą, apdorojoma apie 7 transakcijas per sekundę [ZXD+17], kai „Tendermint“ blokų grandinė, taikanti atsparumo klaidoms konsensusą, teigia galinti apdoroti tūkstančius transakcijų per sekundę [Ten17]. Dėl darbo įrodymo strategijos neefektyvumo blokų grandinės neretai tampa priverstos iškeisti ją į efektyvesnį konsensuso būdą - „Ethereum“ blokų grandinė ketina pereiti prie turto įrodymu grįsto konsensuso [Eth14].

Kita priežastis, dėl ko transakcijos tvirtinamos gana lėtai - blokai turi dydžio apribojimus. Dėl šių apribojimų, tik dalis susikaupusių transakcijų gali būti priimtos į naują bloką, o likusios turi laukti, kol pateks į kitą bloką. Tam spręsti pasitelktos šalutinės blokų grandinės. Jos dalį bloko duomenų iškelia į šalutinę grandinę, taip palikdamos daugiau vietos pagrindinės grandinės bloke. Tokį sprendimą priėmė „Bitcoin“ blokų grandinė, į tinklą pristatydama „SegWit“ protokolą, kuris iškelia skaitmeninių parašų duomenis į atskirą grandinę [Bit16].

Visos blokų grandinės dydis taip pat gali sukelti plečiamumo problemų. Šiuo metu Bitcoin blokų grandinė užima per 100 gigabaitų [ZXD+17]. Siekiant sumažinti blokų grandinės (kurią turi kiekvienas mazgas tinkle) dydį, siūlomi įvairūs sprendimai. Vienas variantas yra mazgams neturėti seniausių blokų grandinės dalių, o senus blokus su transakcijomis iškelti į atskirą duomenų bazę. Kitas sprendimo būdas - pagrindinėje blokų grandinėje laikyti tik transakcijų maišos rezultatus, o pačius transakcijų duomenis iškelti už grandinės ribų (angl. *off-chain*) [LXC+17].

2.6.3. Anonimiškumas

Manoma, kad blokų grandinės išsaugo naudotojų privatumą dėl to, kad transakcijose atskleidžiamas tik naudotojų viešas raktos vietoj tikrosios tapatybės [ZXD+17]. Tačiau, tyrimai rodo, kad iš viešojo rakto galima atsekti tikrąją naudotojo tapatybę [Bar07]. Taip pat, transakcijoje buvę duomenys (pvz. pervestas pinigų kiekis) viešoje blokų grandinėje bus matomas visiems tinklo dalyviams.

Galimi skirtingi sprendimai didesniam anonimiškumui pasiekti. Pirmiausia, reikia atsižvelgti, ar negalima naudoti privačios blokų grandinės, kurioje blokų skaitymo teisės būtų apribotos. Kitas variantas - blokų grandinėje saugoti užšifruotus duomenis, kurių skaitymui reikia turėti dešifravimo raktą. Jeigu tai netinkama išeitis, tuomet transakcijas galima atlikti per tarpininką, kuris išskaidys vieną transakciją į keletą, vykdomų su skirtingais viešaisiais raktais, ir taip bus sunkiau atsekti transakcijos autorių [ZXD+17].

3. Blokų grandinės naudojimas skaitmeninės tapatybės valdyme

Šiame skyriuje nagrinėjamas blokų grandinės technologijos panaudojimas tapatybės valdyme. Tiriama, kaip blokų grandinės savybės gali padėti išspręsti egzistuojančias naudotojų problemas, apžvelgiami technologiją identiteto valdyme jau taikantys projektai.

3.1. Blokų grandinės tinkamumas identiteto valdymui

2 skyriuje apibendrintos naudotojų problemos tapatybės valdyme rodo, kad naudotojams internete trūksta savo asmens duomenų kontrolės, jie turi pasitikėti paslaugų programų veikimu bei tapatybės tiekėjo pasiekiamumu. Viešos blokų grandinės savybės atveria įvairių galimybių, kaip spręsti susidariusias problemas:

1. Decentralizuota sistema eliminuoja vienintelio nesekmės taško situaciją. Pritaikius decentralizuotą blokų grandinę tapatybės valdymo sistemoje, dėl blokų grandinės išskirstymo po skirtingus mazgus tapatybės tiekėjas nebebūtų vienintelis nesekmės taškas identiteto valdymo infrastruktūroje. Tokiu būdu tapatybės informacija būtų prieinama ir tada, kai tapatybės tiekėjas yra nepasiekiamas.
2. Visiems prieinami grandinės įrašai ir sistemos veikimas padidina skaidrumą. Blokų grandinėje saugant naudotojo suteiktas prieigas skirtingoms paslaugoms, naudotojas betkada galėtų matyti paslaugoms suteiktas teises. Prieigų suteikimą aprašius išmaniuosiuose kontraktuose, paslaugos duomenis pasiekti galėtų tik tada, kai naudotojas išreikštinai tai patvirtina.
3. Nekintamumas apsaugo nuo galimų įsilaužimų siekiant pakeisti įrašus. Blokų grandinėje saugant naudotojo atributus ar paslaugų prieigos teises, o rašymo teises į grandinę turint tik naudotojui ar jo pasirinktoms programoms, naudotojas būtų užtikrintas, kad neįvyko jo paties neautorizuotų pakeitimų.

Įvardyti blokų grandinės pranašumai būtų sunkiau įgyvendinami taikant įprastą centralizuotą duomenų bazę. Tokia duomenų bazė būtų kontroliuojama vienos organizacijos ar asmens, tad išliktų vienintelis nesekmės taškas. Įrašų prieinamumu ir nekintamumu bei veikimo skaidrumu turėtų išreikštinai rūpintis duombazės kūrėjai, kai šias savybės užtikrina pati blokų grandinės technologija. Pranašumas, kurį šiuo atveju turi duomenų bazė, yra plečiamumas - saugoti didžiulius kiekius tapatybės duomenų blokų grandinėje yra gana brangu.

2.6 skyrelyje įvardyti pagrindiniai blokų grandinės apribojimai nesudaro didelių kliūčių šią technologiją taikyti skaitmeninės tapatybės valdyme. Anonimiškumo klausimą galima spręsti šifruojant saugomus tapatybės duomenis, o raktus dešifravimui suteikiant tik norimiems asmenims. Blokų grandinės greitaveika neturėti kelti problemų standartinėms tapatybės valdymo sistemos operacijoms [LXC+17], o didėjant įrašų kiekiui dalį jų būtų galima iškelti į šalutinę grandinę (angl. *off-chain*). Daugumos ataka decentralizuotoje viešoje blokų grandinėje išlieka kaip galimas pavojus, tačiau šią riziką galima sumažinti pasirenkant tokią blokų grandinę, kurioje tokios atakos vykdymo kaina būtų per didelė.

3.2. Blokų grandinės projektai tapatybės valdymo srityje

Tiriant blokų grandinės naudojimą skaitmeninės tapatybės valdymo srityje, rasta įvairių technologiją bandančių pritaikyti projektų. Šis skyrius skirtas trumpai apžvelgti autoriaus nuomone įdomiausius projektus, susijusius su tapatybe bei blokų grandinės technologija.

3.2.1. Blockstack

„Blockstack“ projektas yra kompiuterių tinklas, kolektyviai saugantis globalų domenų vardų bei jų viešų raktų registrą. Su šiuo registru, „Blockstack“ įgyvendiną decentralizuotą domenų vardų sistemą, kur kiekvienas norintis gali užregistruoti savo domeną [ASN+17]. „OneName“ projektas, naudojantis „Blockstack“ sistemą, siekia sukurti decentralizuotą tapatybę, kurią susikūręs naudotojas galėtų atsisakyti visų kitų turimų socialinių tinklų („Facebook“, „Google“) tapatybių.

3.2.2. UPort

„Consensys“ įmonės „UPort“ projektas panašus į „OneName“ - jis skirtas naudotojams turėti decentralizuotas skaitmenines tapatybes, registruotas „Ethereum“ tinkle, su kuriomis galėtų atlikti „Ethereum“ transakcijas. Naudotojai su „UPort“ paskyra galėtų prisijungti prie paslaugų bei valdyti savo atributus. Šio darbo rašymo metu „UPort“ turi parengę integracijas paslaugoms ir elektroninę piniginę, skirtą „Ethereum“ raktų ir transakcijų valdymui, tačiau mobilioji programėlė „UPort“ skaitmeninei tapatybei valdyti vis dar ruošiama (angl. *under development*) [Con18].

3.2.3. Tradle

„Tradle“ yra blokų grandine paremtas projektas, skirtas finansinėms institucijoms (bankams, draudimo įmonėms) įgyvendinti „pažink savo klientą“ (angl. *Know Your Customer - KYC*) procesus, suteikiant klientams didesnę asmens duomenų kontrolę. „Tradle“ suteikia klientams programinę įrangą, kuri padės integruoti „Tradle“ platformą su jų įmonėse jau egzistuojančia klientų duomenų saugojimo infrastruktūra. Klientams išreikštinai sutikus, „Tradle“ taip pat suteikia galimybę perduoti asmenų duomenis iš vienos finansinės įmonės į kitą [Baa16].

3.3. Apibendrinimas

Blokų grandinės savybės suteikia pagrindo naudoti šią technologiją skaitmeninės tapatybės valdyme. Decentralizuotumas, skaidrumas bei nekintamumas gali padėti grąžinti naudotojams jų pačių asmens duomenų kontrolę bei panaikinti didžiulę priklausomybę nuo tapatybės ar paslaugų tiekėjų. Jau egzistuojantys projektai, taikantys blokų grandinę identiteto valdyme, įrodo technologijos galimą potencialą šioje srityje.

4. Tapatybės atributų valdymo modelis

Atsižvelgus į blokų grandinės savybes, tinkamas tapatybės valdymo problemoms spręsti, šiame skyriuje pateikiamas vieša blokų grandine paremtas skaitmeninės tapatybės valdymo modelis. Modelis skirtas spręsti 1.5 skyrelyje apibendrintoms naudotojų problemoms. Kadangi tapatybės valdymas apima daug skirtingų procesų, aprašomas modelis apsiriboja asmens tapatybės atributų valdymu, saugojimu ir perdavimu.

Modelio architektūra kurta atsižvelgiant į MIT tyrime [ZNP15] pristatytus teorinius protokolus, kurie apibrėžia mobiliųjų programėlių, naudotojo bei blokų grandinės bendravimą, asmeniui suteikiant telefone saugomą informaciją (pvz. lokacijos duomenis) naujai parsisiųstai programėlei. Aprašomas modelis modifikuoja bendravimą, siekiant jį pritaikyti naudojimui internete, taip pat išplečia funkcionalumą, leidžiant naudotojui pačiam įvesti norimą informaciją.

4.1. Reikalavimai

Naudotojams kylančios problemos tapatybės valdyme remiasi į tai, kad jų tapatybė yra visiškai patikėta valdyti tapatybės tiekėjams. Dėl menko naudotojų įsitraukimo į tapatybės valdymo procesus, asmenys neretai lieka tik pasyvūs stebėtojai.

Siekiant išspręsti šią problemą, kuriamo modelio reikalavimai kurti pagal į naudotoją orientuotos tapatybės (angl. *user-centric identity*) principus. Ši paradigma akcentuoja naudotojus kaip centrinę identiteto valdymo sistemų dalį, perduodant paslaugų ir tapatybės tiekėjų turimą tapatybės kontrolę naudotojams [CY10]. Tokiu būdu, naudotojai turi aktyviau prižiūrėti ir dalyvauti tapatybės valdymo procesuose (dažniausiai naudojant papildomą programinę įrangą), tačiau geriau žino, kaip ir kur yra naudojami jų asmens duomenys.

Taikant į asmenis orientuotos skaitmeninės tapatybės principus, reikalavimai modeliui suformuluoti naudotojų istorijų forma:

1. Kaip interneto naudotojas, aš noriu žinoti, kurios programos turi prieigą prie kurių mano asmens duomenų.
2. Kaip interneto naudotojas, aš noriu kontroliuoti savo asmens duomenis ir pats pasirinkti, kurios mano naudojamos paslaugos gali pasiekti mano asmens duomenis.
3. Kaip interneto naudotojas, aš noriu galimybės lengvai atnaujinti savo asmens duomenis vienoje vietoje.
4. Kaip interneto naudotojas, aš nenoriu, jog mano asmens duomenų pasiekiamumas priklausytų tik nuo vienos trečiosios šalies pasiekiamumo.

Pateiktos naudotojų istorijos padengia 1 skyriuje apibrėžtus asmenų poreikius tapatybės valdymo sistemoms bei išskirtas kontrolės, pasitikėjimo ir skaidrumo trūkumo problemas. Interneto tapatybių valdymo sistemų saugumo atakos (angl. *cross site scripting*, *phishing*) yra plati tema, verta atskiro tyrimo, tad ji šiame modelyje nebus nagrinėjama.

4.2. Modelio dalys

Modelis sudarytas iš trijų dalių: atributų saugyklos išmaniojo kontrakto, paslaugų registro išmaniojo kontrakto bei atributų valdymo programos.

4.2.1. Atributų saugyklos išmanusis kontraktas

Atributų saugyklos išmaniajame kontrakte yra saugomi šifruoti naudotojų atributai bei jų suteikimo logika. Šis kontraktas atsakingas už:

- naudotojo atliekamą prieigų suteikimą. Naudotojas, kviesdamas kontrakto funkciją, gali pasirinkti, ar suteikti konkrečiai paslaugai prieigą prie jos norimo atributo. Prieigos yra valdomos ne visos tapatybės, o atskirų atributų lygmenyje;
- šifruotų atributų saugojimą. Kontrakte saugomi naudotojo tapatybės atributai. Kadangi modelyje naudojama vieša blokų grandinė, joje esantys duomenys prieinami visiems - dėl to saugomi šifruoti atributai. Jeigu naudotojas N yra autorizavęs paslaugą P pasiekti atributą A, tuomet kontrakte išsaugomas N ir P simetrišku šifro raktu (angl. *symmetric encryption key*) užšifruotas atributas A. Taip tik paslauga P ir naudotojas N, turintys šifro raktus, galės perskaityti viešame išmaniajame kontrakte esantį atributą;
- pateikiamas funkcijas paslaugoms bei naudotojui pasiekti atributus. Išmanusis kontraktas pirmiausia atsižvelgia į funkcijos kvietėją. Jei kreipiasi naudotojas, jam grąžinama šifruota atributo reikšmė. Jei kreipiasi paslauga, tikrinama, ar naudotojas yra patvirtinęs jos prieigą norimam atributui. Jei taip, grąžinama šifruota reikšmė. Jei ne, grąžinamas pranešimas, kad prieiga atmesta. Jeigu naudotojas šios paslaugos prieigos dar nesvarstęs, apie paslaugos P siekį gauti naudotojo N atributą A pranešama naudotojui. Kai jis prieigą patvirtins arba atmes, paslauga galės pasiekti atributą A.

Atributai kontrakte atskiriami pagal jų identifikatorius (pvz. 1 - vardas, 2 - telefono numeris ir t.t.). Siekiant minimizuoti blokų grandinėje laikomą informaciją, atributų identifikatorių žodynas nesaugomas pačiame kontrakte. Svarbu, kad šis žodynas būtų viešai prieinamas visoms paslaugoms - jis galėtų būti pateikiamas modelio dokumentacijoje.

4.2.2. Atributų valdymo programa

Atributų valdymo programa yra skirta palengvinti naudotojo bendravimą su blokų grandine. Teoriškai, naudotojas galėtų pats formuoti užklausas, generuoti šifro raktus, perduoti jos paslaugoms, užšifruoti ir dešifruoti siunčiamas reikšmes, tačiau tai nėra patogu. Dėl to ši programa padeda atlikti „nematomą darbą“ bei supaprastina naudotojo sąsają su bloko grandine. Programoje naudotojas gali:

- suteikti arba atmesti paslaugai P prieigą prie atributo A,
- peržvelgti suteiktas prieigas ir keisti jas,
- įvesti, keisti ar trinti atributų reikšmes.

Atributų valdymo programos veikimas panašus į kriptovaliutos piniginės veikimą. Kaip tokia piniginė leidžia naudotojui per vartotojo sąsają pervesti pinigus iš naudotojo sąskaitos į kitą sąskaitą, taip atributų valdymo programa leidžia naudotojui valdyti savo paskyros atributus blokų grandinės išmaniajame kontrakte. Programa veikia naudotojo vardu (angl. *on user's behalf*), turėdama jo privatų raktą - taip ji gali kreiptis į blokų grandinę iš naudotojo paskyros, o išmanusis kontraktas, gavęs užklausą iš atributų valdymo programos, žino, koks naudotojas kreipiasi į funkciją.

Modelyje nėra apibrėžiama, kaip turi būti įgyvendinta ši programa - tai gali būti kompiuterio darbalaukio programa, mobilioji programėlė ar interneto tinklapis.

4.2.3. Paslaugų registro išmanusis kontraktas

Šiame modelyje paslaugos yra identifikuojamos pagal jų blokų grandinės paskyros adresą. Atributų valdymo programa, skaitanti išmaniajame kontrakte laukiančias užklausas, jose mato išsaugotą besikreipusios paslaugos adresą.

Nors adresas yra viešas ir unikaliai identifikuoja kiekvieną blokų grandinės dalyvį, turintį paskyrą, tačiau iš adreso nustatyti tikrąją dalyvio tapatybę yra sudėtinga - pats adresas nėra tiesiogiai susiejamas su tikru asmens identifikatoriumi (pvz. asmens kodu ar paslaugos įmonės kodu). Tokiu būdu, atributų valdymo programoje matant tik paslaugos adresą dalyvis gali nežinoti, kokią paslaugą jis autorizuoja.

Šiai problemai spręsti modelyje yra atskiras išmanusis kontraktas, kuris funkcionuoja kaip paslaugų registras. Kiekviena paslauga, besinaudojanti modeliu, atlieka vieną transakciją į šį registrą, joje nusiųsdama savo pavadinimą bei slaptą kodą, sugeneruotą atributų valdymo programos. Kai atributų saugyklos kontrakte bus išsaugota nauja paslaugos P užklausa pasiekti atributą, atributų valdymo programa pagal P viešą raktą gali kreiptis į šį registrą ir įsitikinti, ar P tikrai užsiregistravo (rasti iš P paskyros atliktą transakciją) ir iš kodo nuspręsti, ar tai tikrai yra paslauga, kuria asmuo naudojasi (taip užsiregistravę, bet kodo neturintys programišiai negali apsimesti kita įmone). Atributų valdymo programa, remdamasi paslaugų registru, gali atfiltruoti ir naudotojui pateikti tik validžias paslaugų užklausas.

Verta pastebėti, kad paslaugų registras nėra vienintelis įmanomas būdas autentifikuoti paslaugas. Galimos alternatyvos plačiau apžvelgiamos modelio vertinimo skyriuje.

4.3. Pagrindiniai naudojimo atvejai

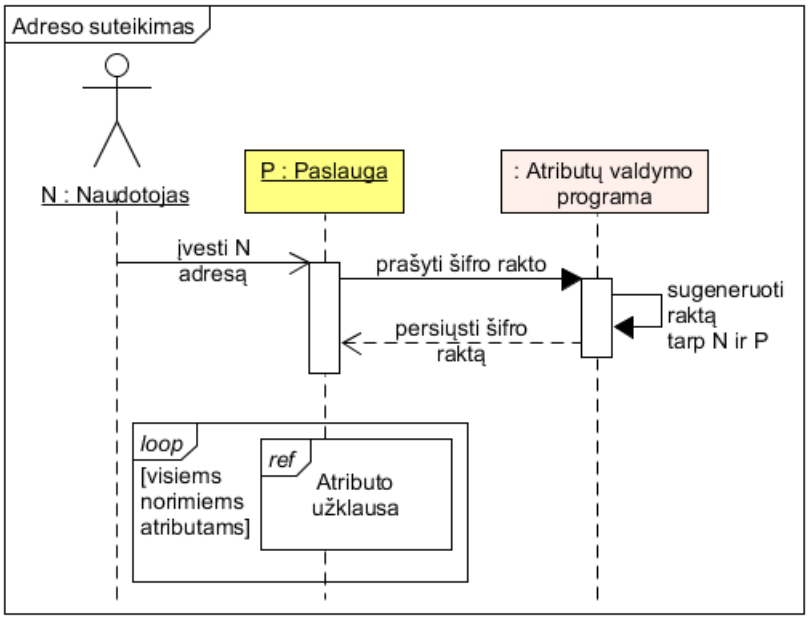
Šiame skyriuje UML sekų diagramomis pateikiami pagrindiniai modelio panaudos atvejai (angl. *use cases*).

4.3.1. Naudotojo adreso suteikimas paslaugai

Kai paslauga nori pasiekti naudotojo asmens duomenis, ji paprašo naudotojo pateikti jo paskyros adresą (naudotojo patogumui, adresas rodomas atributų valdymo programoje), kuris yra viešas naudotojo identifikatorius blokų grandinėje. Su gautu adresu paslauga P gali kreiptis į blokų

grandinę ir prašyti norimų naudotojo atributų (žr. 8 pav.). Gavusi šį adresą, paslauga kreipiasi į atributų valdymo programą gauti šifro raktą, kurio reikės norint dešifruoti iš blokų grandinės gautas atributų reikšmes.

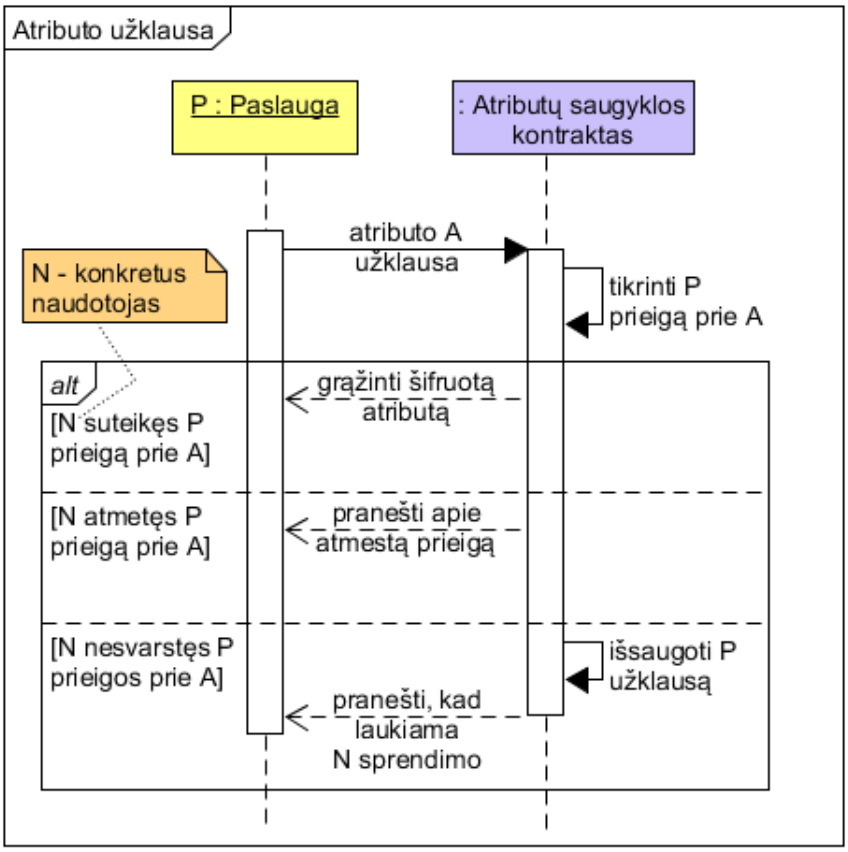
Modelis neapibrėžia, kada naudotojas N pateikia adresą paslaugai P. Svarbu, kad paslauga P gautų šį adresą ir unikalų šifro raktą tarp N ir P, nes be jų P negalės gauti ir perskaityti norimų tapatybės duomenų. Šis suteikimas galėtų būti atliekamas prisijungimo metu, įtraukiant papildomą žingsnį.



8 pav. Naudotojo adreso suteikimas paslaugai

4.3.2. Atributo užklausa

Paslauga P, norinti gauti tam tikrą asmens atributą (pvz. gimimo datą), kreipiasi į atributų saugyklos išmanųjį kontraktą (žr. 9 pav.). Kontraktas tuomet patikrina, ar ši paslauga turi prieigą prie pageidaujamo atributo. Jei turi, tuomet grąžina šį atributą. Jis užšifruotas paslaugos P turimu šifro raktu, tad paslauga gali jį dešifruoti ir perskaityti duomenis. Jei naudotojas atmetęs prieigą, apie tai pranešama paslaugai. Jei naudotojas dar nesvarstęs šios prieigos, išsaugoma laukianti (angl. *pending*) paslaugos P atributo A užklausa ir laukiama naudotojo sprendimo. Naudotojas šią laukiančią užklausą galės pamatyti savo atributų valdymo programoje ir ten atlikti sprendimą.



9 pav. Paslaugos atliekama atributo užklausa

4.3.3. Paslaugos autorizavimas

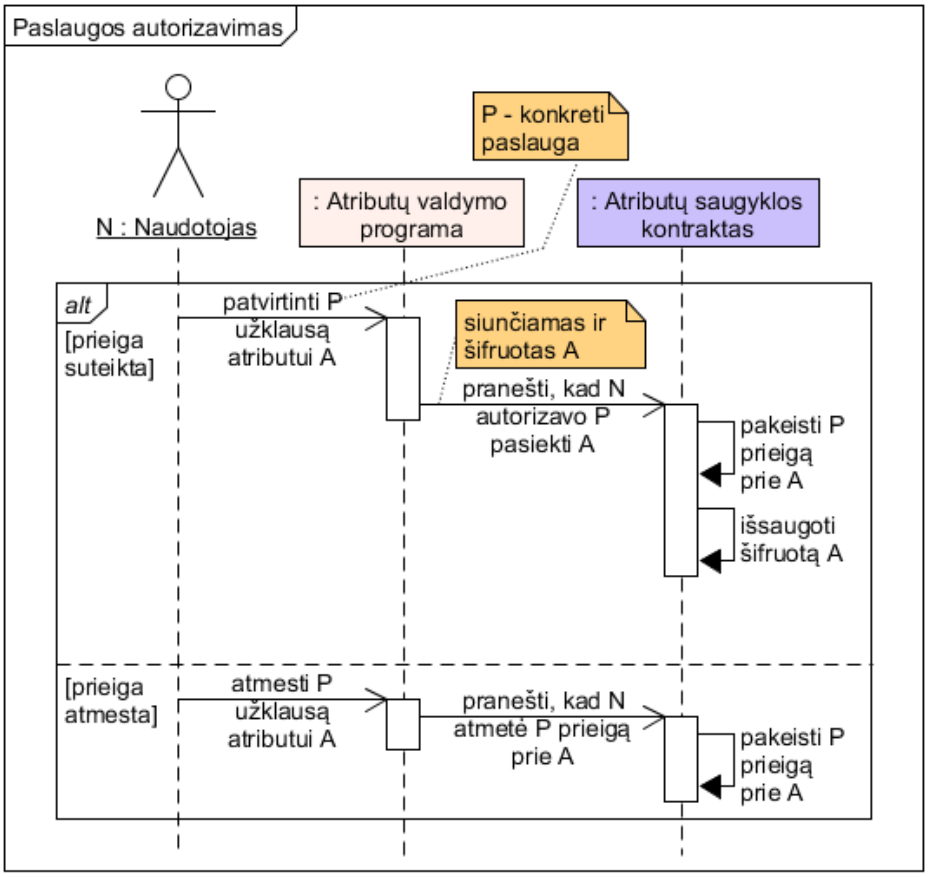
Atributų valdymo programa, stebinti atributų saugyklos kontraktą, praneša naudotojui apie naujas paslaugų užklausas. Tuomet naudotojas gali pasirinkti, ar suteikti, ar atmesti prieigą prie norimo atributo A paslaugai P. (žr. 10 pav.). Atributų saugyklos kontraktas asmeniui leidžia autorizuoti tik savo naudojamas paslaugas[1].

Jei naudotojas sutinka suteikti prieigą, tuomet jis įveda atributo reikšmę ir atributų valdymo programa užšifruoja ją. Užšifravus reikšmę, duomenys nusiunčiami į išmanųjį kontraktą ir jame pažymima, kad paslauga P naudotojo N autorizuota pasiekti atributą A.

Jei naudotojas prieigą atmeta, tuomet blokų grandinėje pažymima, kad paslauga P nėra autorizuota pasiekti naudotojo N atributą A.

Jei naudotojas vėliau nuspręstų pakeisti savo sprendimą (pvz. panaikinti suteiktas prieigas prie atributo A paslaugai P), jis šį paslaugos autorizavimo procesą galėtų pakartoti ir jau esamai prieigai.

[1] ar naudotojo N1 įrašų nebando pakeisti N2, atskiriama iš to, kas kreipiasi į funkciją



10 pav. Naudotojo atliekamas paslaugos autorizavimas

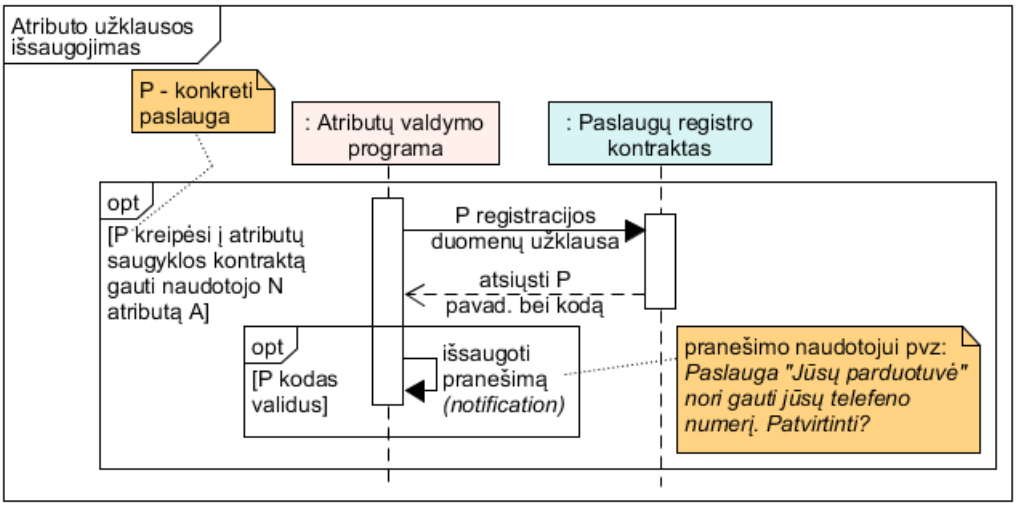
4.3.4. Pranešimas naudotojui apie naują atributo užklausą

Norint išsaugoti naują atributo užklausą, kurią turės patvirtinti naudotojas, atributų valdymo programa turi sužinoti apie ją iš atributų saugyklos kontrakto. Tai galima pasiekti klausantis kontrakto paskelbiamų įvykių[1]. Jeigu programa „neišgirdo“ realiu laiku paskelbto įvykio, ji gali kreiptis į kontraktą ir pasiekti anksčiau paskelbtus įvykius.

Paskelbtame užklausos įvykyje pateikiamas besikreipusios paslaugos P viešas raktas, norimo atributo A identifikatorius bei kurio naudotojo N atributo paslauga nori. Pagal P viešą raktą programa kreipiasi į paslaugų registro išmanųjį kontraktą ir gauna P registracijos duomenis (žr. 11 pav.). Validavus P registraciją (P paslaugų registro kontrakte įrašė tinkamą kodą), programa praneša naudotojui N apie naują P užklausą. Šioje užklausoje N matys P pavadinimą ir norimą gauti atributą - iš to naudotojas galės nuspręsti, ar patvirtinti, ar atmesti šią prieigos užklausą.

Kadangi registraciją paslaugai P pakanka validuoti vieną kartą, atributų valdymo programa gali prisiminti sėkmingą P registraciją ir tolimesnėse užklausose registracijos nebevaliduoti.

[1]jei naudojama blokų grandinė neturi įvykių mechanizmo, atributų valdymo programa gali periodiškai kreiptis į kontraktą ir tikrinti, ar jame yra naujų išsaugotų užklausų.



11 pav. Pranešimas naudotojui apie naują atributo užklausą

5. Tapatybės atributų valdymo modelio prototipas

Tapatybės atributų valdymo modelis sudarytas iš 3 dalių: atributų saugyklos kontrakto, paslaugų registro kontrakto bei atributų valdymo programos (žr. 4.2.1 sk.). Siekiant patvirtinti galimą modelio veikimą, „Solidity“ programavimo kalba „Ethereum“ blokų grandinei realizuoti atributų saugyklos bei paslaugų registrų kontraktai.

5.1. Pasirinktos technologijos

Išmaniesiems kontraktams programuoti pasirinkta naudoti „Ethereum“ platformą ir „Solidity“ programavimo kalbą. Priežastys tam:

- programuotojų bendruomenė. Nuo 2015-ųjų veikianti „Ethereum“ platforma turi nemažą palaikančių asmenų bendruomenę, kuri prisideda prie pačio „Ethereum“ atviro kodo vystymo bei atsako į pradedančiųjų klausimus;
- technologijos branda. Kai išmaniųjų kontraktų programavimas yra vis dar gana nauja sritis, „Ethereum“ technologija, veikianti nuo 2015-ųjų, yra gana gerai dokumentuota, o rekomenduojama kontraktams rašyti programavimo kalba „Solidity“ [Eth14] turi keletą skirtingų kūrimo karkasų programavimo aplinkai paruošti [Eth16].

Išmanieji kontraktai kurti naudojant „Solidity“ programavimui skirtą „Truffle“ kūrimo karkasą, taikant „ganache-cli“ lokalią blokų grandinę testavimui. Kontraktų funkcijos išbandytos jas kviečiant iš sukurto testinio „React“ puslapio. Kodas pasiekiamas „GitHub“ repozitorijoje `https://github.com/Ubaldas11/blockchain-attribute-management`.

5.2. Atributų saugyklos kontraktas

Kontraktas skirtas naudotojų atributų saugojimui bei prieigų suteikimui realizuoti. Kontraktas sudarytas iš kontrakto saugyklos, kurioje saugomos koduotos atributų reikšmės ir jų prieigos, įvykių bei funkcijų. Kontrakto kodas pateikiamas priede nr. 2.

Kontrakte apibrėžti šie įvykiai:

1. *AccessRequested*. Skirtas pranešti, kad paslauga P prašo prieigos prie naudotojo N atributo A. Šio įvykio laukia atributų valdymo programa ir jį gavusi, gali pranešti naudotojui N apie naują paslaugos P užklausą.
2. *AccessChanged*. Skirtas pranešti, kad naudotojas N pakeitė paslaugos P prieigą prie atributo A. Paslaugos, laukiančios šio įvykio, jį gavusios sužino apie galimai suteiktą arba panaikintą prieigą prie atributo A.

Kontrakte apibrėžtos šios viešos funkcijos:

1. *grantAccess*. Skirta naudotojui N suteikti paslaugai P prieigą prie atributo A. Į funkciją paduodamas A identifikatorius, P adresas bei N ir P šifro raktu užkoduota A reikšmė. Funkcijoje keičiama siuntėjo atributo reikšmė - taip užtikrinama, kad naudotojas N negalės pakeisti naudotojo N1 atributo. Funkcija paskelbia *AccessChanged* įvykį;

2. *removeAccess*. Skirta naudotojui N panaikinti paslaugos P prieigą prie atributo A. Į funkciją paduodamas A identifikatorius ir P adresas. Kaip ir *grantAccess* atveju, keičiama siuntėjo atributo prieiga. Funkcija paskelbia *AccessChanged* įvykį;
3. *requestAccess*. Skirta paslaugai P pranešti, kad ji nori prieigos prie naudotojo N atributo A reikšmės. Į funkciją paduodamas A identifikatorius bei N adresas. Funkcija paskelbia *AccessRequested* įvykį;
4. *getAttribute*. Skirta gauti naudotojo N paslaugai P skirtą atributą A. Į funkciją paduodamas A identifikatorius, N adresas bei P adresas. Jei funkciją iškvietė N (t.y., pats naudotojas kreipiasi gauti suteiktą atributą A), grąžinamas siekiamas atributas. Jei kreipiasi paslauga, galimos 3 grąžinamos reikšmės. Jei prieiga naudotojo suteikta - grąžinama A reikšmė. Jei prieiga atmesta - pranešama, kad paslauga neautorizuota. Jei prieiga naudotojo dar nebuvo svarstyta, prašoma pirma kreiptis į *requestAttributeAccess* funkciją.
   Funkcija realizuota kaip vaizdo (angl. *view*) funkcija - ji nekeičia kontrakto būsenos, todėl jos kvietimas yra nemokamas.

Kadangi įvykių paskelbimas yra pigesnis nei kontrakto būsenos keitimas, tai išnaudota paskelbiant apie naują prieigos užklausą (vietoj alternatyvos pakeisti kontrakto būseną ir nurodyti, kad paslauga kreipėsi). Jei paslauga arba atributų valdymo programa realiu laiku „neišgirdo“ įvykio, ji vėliau gali pasiekti visą įvykių sąrašą (angl. *log)*.

5.3. Paslaugų registro kontraktas

Kontraktas skirtas užsiregistravusių paslaugų sąrašui saugoti. Paslaugų siųstuose įrašuose laikomas registracijos kodas bei paslaugos pavadinimas. Kontrakto kodas pateikiamas priede nr. 3.

Kontraktą sudaro dvi funkcijos:

1. *register*. Skirta paslaugai užsiregistruoti. Į funkciją paduodamas registracijos kodas bei paslaugos pavadinimas, kurie išsaugomi kontrakto būsenoje.
2. *getService*. Skirta atributų valdymo programai išgauti duomenis apie besikreipusią paslaugą. Funkcijai paduodamas paslaugos adresas. Funkcija realizuota kaip vaizdo (angl. *view*) funkcija - ji nekeičia kontrakto būsenos, todėl jos kvietimas yra nemokamas.
   Atributų valdymo programa kreipiasi į šią funkciją tuomet, kai gauta nauja paslaugos P atributo A užklausa. Iš gauto kodo programa gali atskirti, ar ši paslauga nėra apsimetelė (žr. 4.2.3 skyrelį). Jei kodas nesutampa, atributo prieigos užklausą galima ignoruoti. Jei kodas sutampa, gautą pavadinimą galima rodyti atributų valdymo programoje ir pranešti apie prieigos užklausą naudotojui.

6. Tapatybės atributų valdymo modelio vertinimas

Šiame skyriuje vertinamas 4 skyriuje pristatytas blokų grandine paremtas tapatybės atributų valdymo modelis: kokie jo privalumai, trūkumai, pritaikymo barjerai.

6.1. Privalumai

6.1.1. Naudotojui suteikta atributų valdymo kontrolė

Naudotojas su atributų valdymo programa gali autorizuoti paslaugas pasiekti jo atributus, esančius blokų grandinėje. Programoje taip pat galima pamatyti visus įvestus atributus ir paslaugas, kurios yra autorizuotos juos pasiekti. Naudotojas taip pat gali pakeisti anksčiau priimtą sprendimą ir programa atnaujins prieigą blokų grandinėje. Naudotojui nusprendus pakeisti atributo reikšmę, jis tai padaro atributų valdymo programoje, o ši kreipiasi į blokų grandinę ir joje atnaujina esančias reikšmes. Kaip galėtų atrodyti tokia atributų valdymo programa pateikiama priede **??**.
Išvardytos funkcijos padengia 4.1 skyrelio 1, 2 ir 3 reikalavimus.

6.1.2. Decentralizuoti asmens duomenys

Tapatybės duomenis saugant blokų grandinėje, atributai tampa pasiekiami nepriklausomai nuo vienos trečiosios šalies pasiekiamumo. Taip tiek paslaugų tiekėjai, tiek naudotojas gali bet-kada pamatyti suteiktus atributus tol, kol bent vienas blokų grandinės mazgas yra pasiekiamas. Taip pat, nors duomenys decentralizuoti, dėl jų šifravimo jie išlieka prieinami tik naudotojui ir jo pasirinktoms paslaugoms. Šis privalumas įgyvendina 4.1 skyrelio 4-ą reikalavimą.

6.1.3. Sumažintas reikalingas pasitikėjimas trečiosiomis šalimis

Kadangi tapatybės atributai yra saugomi viešoje, decentralizuotoje blokų grandinėje, kurios išmaniųjų kontraktų logika prieinama visiems, naudotojams nebereikia pasitikėti vien tik paslaugų bei tapatybės tiekėjais - jų atributai šiuo atveju priklauso nuo korektiško išmaniųjų kontraktų veikimo.

Verta pastebėti, kad pristatytame modelyje naudotojui vis dar reikia pasitikėti atributų valdymo programos veikimu. Tačiau, ši programa pati asmens duomenų nesaugo (tik persiunčia įvestas ir grąžina blokų grandinėje esančias reikšmes). Pasitikėjimui padidinti ši programa taip pat turėtų būti viešai prieinamo atviro kodo.

Taip pat, paslaugų tiekėjai, parsisiuntę atributą iš blokų grandinės, vis dar gali jį išsisaugoti. Tačiau, galimybė paslaugai pačiai nesaugoti naudotojo duomenų turėtų būti pagrindinė paskata naudotis šiuo modeliu - taip naudotojai labiau pasitikės šia paslauga. Galimas išsisaugojimo pavojus pastebimas ir [ZNP15] tyrime, kurio autoriai aptaria ir galimybę neleisti paslaugai pasiekti pačio duomens, o atlikti reikalingus skaičiavimus pačiame blokų grandinės tinkle.

6.2. Trūkumai

6.2.1. Paslaugų autentiškumo tikrinimas

Modelyje aprašytas paslaugų registras padeda apsisaugoti nuo programišių, galinčių apsimesti paslaugomis. Šio registro veikimas nėra itin patogus, nes kiekviena paslauga turi gauti sugeneruotą kodą iš atributų valdymo programos ir įrašyti jį į atskirą išmanųjį kontraktą. Tai reikalauja daugiau integracijų (tarp paslaugos, atributų valdymo programos ir papildomo išmaniojo kontrakto) bei papildomo naudotojo pasitikėjimo atributų valdymo programa: jai patikėta generuoti kodą bei atlikti paslaugos registracijos tikrinimą.

Galimos kelios alternatyvos paslaugų identifikavimui užtikrinti. Galima būtų naudotis jau egzistuojančiais identiteto registravimo projektais, kuriuose paslaugos užregistruotų savo tapatybę (pvz. „NameCoin“ ar 3.2 skyrelyje minimu „Blockstack“). Kitas būdas, aptariamas [Baa16] tyrime, būtų turėti registrą, kuriame paslaugų autentiškumą patvirtintų patikima trečioji šalis (pvz. bankas ar valdžios įstaiga). Tokiu atveju, paslaugos turėtų autentifikuotis šios institucijos sistemoje, o ji įrašytų į paslaugų registrą apie sėkmingą paslaugos registraciją. Tokiu atveju pasitikėjimas būtų perkeliamas pasirinktoms institucijoms.

6.2.2. Nepatikimos atributų reikšmės

Paslaugų tiekėjai iš blokų grandinės gauna reikšmes, kurias įvedė pats naudotojas. Tai nesudaro jokių keblumų, kai jų teisingumą gali patikrinti pati paslauga (pvz. atsiųsti žinutę į telefono numerį), tačiau kai to atlikti negalima, paslaugai tenka pasitikėti naudotojo įvesta reikšme.

Tai būtų galima tobulinti, jei šias reikšmes galėtų verifikuoti atsakingos institucijos (pvz. bankas). Tuomet, šios institucijos turėtų teikti papildomą paslaugą, kurioje naudotojui paprašius, jos galėtų atlikti užklausą į blokų grandinę ir patvirtinti, kad atributo reikšmė yra teisinga, o tai būtų išsaugota išmaniajame kontrakte. Tokiu atveju, į blokų grandinę besikreipianti paslauga galėtų matyti, ar šis atributas yra verifikuotas. Panašus tvirtinimo mechanizmas nagrinėjamas [Baa16] tyrime.

6.2.3. Kaina

Blokų grandinės išmaniųjų kontraktų funkcijų kvietimai kainuoja pinigus. Įprastu atveju, funkcijos vykdymą apmoka funkcijos kvietėjas. Aprašytame modelyje kvietimus į blokų grandinę vykdo tiek paslaugų tiekėjas, tiek atributų valdymo programa. Kai naudotojas dėl didesnės kontrolės gali sutikti mokėti už atributų valdymo programos naudojimą, nėra aišku, ar paslaugų tiekėjai būtų pasiryžę mokėti už funkcijų kvietimus.

Vienas iš galimų sprendimų - kviečiant funkciją paslaugai, ją apmokėtų ne paslaugos tiekėjas, o pats kontraktas. Taip kontraktas būtų apmokamas iš naudotojo lėšų, o jos naudojamos paslaugai kviečiant kontrakto funkcijas. Tačiau, tokį mechanizmą turi palaikyti pasirinkta blokų grandinė, taikant jį reiktų apsisaugoti ir nuo galimų pavojingų pakartotinių kvietimų, kurie būtų skirti tik išeikvoti kontrakto lėšas.

kai prototipą pakodinsiu, pažiūrėsiu kiek kainuoja kuro funkcijos, jas gal reiks įmest

6.3. Pritaikymo barjerai

Pagrindiniai sunkumai, kurie kiltų tokį modelį taikant praktikoje, yra galimas paslaugų nenoras bendradarbiauti bei menka blokų grandinės technologijos branda.

Paslaugų tiekėjams, norint pradėti naudoti šį modelį savo paslaugose, reikia susikurti blokų grandinės paskyrą bei suintegruoti savo sistemas su blokų grandinės išmaniaisiais kontraktais bei atributų valdymo programa. Tai gali būti palengvinta sukuriant viešai prieinamą programinę įrangą, paruoštą naudoti paslaugų tiekėjams, tačiau paslaugų kūrėjai vis tiek turi nuspręsti naudoti šį modelį. Pagrindinė paskata jiems yra didesnis naudotojų pasitikėjimas suteikiant jiems galimybę kontroliuoti savo duomenis bei pagerintas asmens duomenų pasiekiamumas juos iškėlus į decentralizuotą blokų grandinę.

Blokų grandinė yra vis dar gana nauja technologija. Vieša blokų grandinė gali būti pažeista daugumos atakos, nemažai skeptikų teigia, kad pagrindinis technologijos panaudojimas išliks tik kriptovaliutoms, technologijos plečiamumas (angl. *scalability*) aktyviai tiriamas. Taip pat, 2016-aisiais metais išmaniojo kontrakto „The DAO“ trūkumą išnaudojusi ataka privertė „Ethereum“ blokų grandinę atlikti išsišakojimą (angl. *hard fork*), kuris pakeitė transakcijų istoriją ir grąžino įsilaužėlio pavogtą kriptovaliutą jos savininkams [Lei17]. Tai rodo, kad ypatingomis sąlygomis blokų grandinės nekintamumas gali būti pažeistas

Nepaisant išvardytų priežasčių, technologijos populiarumas neblėsta. Blokų grandinės technologiją pradeda naudoti didžiosios kompanijos („Microsoft“ bei „IBM“ siūlo blokų grandinę kaip paslaugą [ZXD+17]). Taip pat, vis daugiau programuotojų kuria išmaniuosius kontraktus (darbo rašymo metu „Ethereum“ blokų grandinėje yra 25374 verifikuoti išmanieji kontraktai [Eth15b]).

Blokų grandinė dar nėra brandi, ilgai taikoma technologija. Ji aktyviai bandoma tiek startuolių, tiek korporacijų, vis dar intensyviai tiriama mokslininkų. Technologiją naudojančių programų sėkmė nulems, ar asmenys sutiks naudotis blokų grandine paremtomis programomis.

Rezultatai ir išvados

Darbo rezultatai:

1. Apžvelgti pagrindiniai skaitmeninės tapatybės valdymo būdai internete.
2. Įvardytos esminės naudotojams kylančios problemos tapatybės valdyme: nepakankama asmens duomenų kontrolė, mažėjantis pasitikėjimas paslaugų duomenų valdymu ir didelė priklausomybė nuo tapatybės tiekėjų.
3. Pristatyta blokų grandinės technologija. Nustatyta, kad galimybė kurti decentralizuotus, veikimą atskleidžiančius išmaniuosius kontraktus blokų grandinėje suteikia pagrindo naudoti blokų grandines ir išmaniuosius kontraktus skaitmeninės tapatybės valdymo srityje.
4. Remiantis į naudotoją orientuotos tapatybės (angl. *user-centric identity*) principais, parengti reikalavimai skaitmeninės tapatybės atributų valdymo modeliui, akcentuojantys įvardytas naudotojų problemas.
5. Paruoštas blokų grandine paremtas tapatybės atributų valdymo modelis, įgyvendinantis iškeltus reikalavimus ir suteikiantis asmeniui galimybę kontroliuoti savo asmens duomenis bei autorizuoti paslaugų prieigą prie jų. Modelis įvertintas išskiriant jo privalumus, trūkumus bei galimus pritaikymo barjerus.
6. Sukurtas pristatyto modelio prototipas. „Solidity“ programavimo kalba realizuoti „Ethereum“ blokų grandinei pritaikyti išmanieji kontraktai, įgyvendinantys paslaugų autorizavimo ir atributų saugojimo logiką. Paruošti mobiliosios programėlės, skirtos naudotojui palengvinti sąveiką su blokų grandine, maketai.

Darbo išvados:

Skaitmeninės tapatybės valdymas yra sritis, kurioje naudotojai neturi pakankamai kontrolės. Dideli kiekiai asmens duomenų yra perduodami tarp programų neinformuojant naudotojų ar nutekinami dėl programišių įsilaužimų. Skaitmeninio identiteto centralizavimas tapatybės tiekėjų sistemose sukuria situaciją, kurioje naudotojas tampa visiškai priklausomas nuo tapatybės tiekėjo bei yra priverstas pasitikėti juo.

Blokų grandinės technologija gali padėti išspręsti susidariusias problemas. Decentralizuotai saugant asmens duomenis, panaikinama priklausomybė nuo tapatybės tiekėjo. Viešuose, visiems prieinamuose išmaniuosiuose kontraktuose aprašius paslaugų autorizavimo logiką, naudotojas gali įsitikinti, kad jis geba kontroliuoti paslaugų turimas prieigas prie jo asmens duomenų. Dėl kontrakto kodo nekintamumo, asmuo gali būti tikras, kad laikui bėgant sąlygos nepasikeis ir naudotojo priimti autorizavimo sprendimai išliks.

Aprašytas blokų grandinės taikymas tapatybės valdymo sistemoje turi iššūkių. Decentralizuotas kodo vykdymas yra gana brangus, įmanomos blokų grandinės tinklo daugumos atakos gali atgrasyti potencialius naudotojus, o tapatybės duomenų iškėlimas į blokų grandinę verčia paslaugų tiekėjus kurti papildomą integraciją. Nepaisant to, galimybė grąžinti asmenims jų duomenų kontrolę internete ir atgauti naudotojų pasitikėjimą gali būti pakankamos paskatos taikyti blokų grandinę skaitmeninės tapatybės valdyme.

Literatūra

[Ant14] Andreas M. Antonopoulos. Mastering bitcoin : unlocking digital cryptocurrencies. 2014, p. 272. ISBN: 1449374042.

[ASN+17] Muneeb Ali, Ryan Shea, Jude Nelson ir Michael J Freedman. Blockstack Technical Whitepaper. Disertacija, 2017. URL: https://blockstack.org/whitepaper.pdf.

[Baa16] Djuri Baars. Towards Self-Sovereign Identity using Blockchain Technology. Disertacija, University of Twente, 2016. URL: http://essay.utwente.nl/71274/1/Baars%7B%5C_%7DMA%7B%5C_%7DBMS.pdf.

[Bar07] Jaume Barcelo. User Privacy in the Public Bitcoin Blockchain. 6(1), 2007. URL: https://pdfs.semanticscholar.org/549e/7f042fe0aa979d95348f0e04939b2b451f18.pdf.

[Bit16] Bitcoin Wiki. Segregated Witness, 2016. URL: https://en.bitcoin.it/wiki/Segregated%7B%5C_%7DWitness (tikrinta 2018-05-02).

[But15a] Vitalik Buterin. On Public and Private Blockchains - Ethereum Blog, 2015. URL: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/ (tikrinta 2018-05-02).

[But15b] Vitalik Buterin. Understanding Serenity, Part I: Abstraction - Ethereum Blog, 2015. URL: https://blog.ethereum.org/2015/12/24/understanding-serenity-part-i-abstraction/ (tikrinta 2018-05-19).

[Cam04] L.J. Camp. Digital identity. IEEE Technology and Society Magazine, 23(3):34–41, 2004. DOI: 10.1109/MTAS.2004.1337889. URL: http://ieeexplore.ieee.org/document/1337889/.

[CY10] Yuan Cao Yuan Cao ir Lin Yang Lin Yang. A survey of Identity Management technology. 2010 IEEE International Conference on Information Theory and Information Security:287–293, 2010. DOI: 10.1109/ICITIS.2010.5689468. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5689468.

[CK01] S Clauß ir M Køhntopp. Identity Management and its Support of Multilateral Security. Computer Networks, 37 (2):205–219, 2001.

[Con18] Consensys. UPort webpage, 2018. URL: https://www.uport.me (tikrinta 2018-05-09).

[DD08] Rachna Dhamija ir Lisa Dusseault. The Seven Flaws of Identity Management: Usability and Security Challenges. IEEE Security & Privacy Magazine, 6(2):24–29, 2008-03. ISSN: 1540-7993. DOI: 10.1109/MSP.2008.49. URL: http://ieeexplore.ieee.org/document/4489846/.

[DP08] M. Dabrowski ir P. Pacyna. Generic and Complete Three-Level Identity Management Model. 2008 Second International Conference on Emerging Security Information, Systems and Technologies, p. 232–237. IEEE, 2008-08. DOI: 10.1109/SECURWARE.2008.18. URL: http://ieeexplore.ieee.org/document/4622588/.

[Eth14] Ethereum Foundation. Ethereum Wiki, 2014. URL: https://github.com/ethereum/wiki/wiki.

[Eth15a] Ethereum Foundation. Ethereum White Paper, 2015. URL: https://github.com/ethereum/wiki/wiki/White-Paper.

[Eth15b] Etherscan. Ethereum Contracts with Verified Source Codes, 2015. URL: https://etherscan.io/contractsVerified (tikrinta 2018-05-08).

[Eth16] Ethereum Foundation. Solidity Documentation, 2016. URL: https://solidity.readthedocs.io/en/v0.4.23/.

[FH07] Dinei Florencio ir Cormac Herley. A large-scale study of web password habits. Proceedings of the 16th international conference on World Wide Web - WWW '07, p. 657, New York, New York, USA. ACM Press, 2007. ISBN: 9781595936547. DOI: 10.1145/1242572.1242661. URL: http://portal.acm.org/citation.cfm?doid=1242572.1242661.

[GC07] Benjamin M. Gross ir Elizabeth F. Churchill. Addressing Constraints: Multiple Usernames, Task Spillage and Notions of Identity. CHI '07 extended abstracts on Human factors in computing systems - CHI '07, p. 2393, New York, New York, USA. ACM Press, 2007. ISBN: 9781595936424. DOI: 10.1145/1240866.1241013. URL: http://portal.acm.org/citation.cfm?doid=1240866.1241013.

[Gra18] Kevin Granville. Facebook and Cambridge Analytica, 2018. URL: https://www.nytimes.com/2018/03/19/technology/facebook-cambridge-analytica-explained.html (tikrinta 2018-03-28).

[GV09] Uwe Glässer ir Mona Vajihollahi. Identity Management Architecture. 9:97–116, 2009. URL: https://link.springer.com/content/pdf/10.1007%7B%5C%7D2F978-1-4419-1325-8.pdf.

[Hos17] Adam Hose. Rolling your own Proof-of-Authority Ethereum consortium, 2017. URL: http://blog.enuma.io/update/2017/08/29/proof-of-authority-ethereum-networks.html (tikrinta 2018-05-01).

[JP05] Audun Jøsang ir Simon Pope. User Centric Identity Management. AusCERT Conference, 2005. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.1563%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf.

[Kou17] Sherif Koussa. Differentiating Federated Identities: OpenID Connect, SAML v2.0, and OAuth 2.0, 2017. URL: https://www.softwaresecured.com/differentiating-federated-identities-openid-connect-saml-v2-0-and-oauth-2-0/ (tikrinta 2018-04-22).

[Kuk11] Ado Kukic. Definitive guide to Single Sign-On (SSO), 2011. URL: https://resources.auth0.com/definitive-guide-to-single-sign-on/.

[Lei17] Matthew Leising. The Ether Thief, 2017. URL: https://www.bloomberg.com/features/2017-the-ether-thief/ (tikrinta 2018-05-08).

[LXC+17] Sin Kuang Lo, Xiwei Xu, Yin Kia Chiam ir Qinghua Lu. Evaluating Suitability of Applying Blockchain. 2017 22nd International Conference on Engineering of Complex Computer Systems (ICECCS):158–161, 2017. DOI: 10.1109/ICECCS.2017.26. URL: http://ieeexplore.ieee.org/document/8292816/.

[Mic07] Microsoft Developer Network. Access Tokens, 2007. URL: https://msdn.microsoft.com/en-us/library/Aa374909.aspx (tikrinta 2018-04-07).

[Min18] Miniwatts Marketing Group. World Internet Users Statistics, 2018. URL: https://www.internetworldstats.com/stats.htm (tikrinta 2018-03-28).

[MM17] Jason Mander ir Felim McGrath. Global Web Index Social. Tech. atask., 2017. URL: https://cdn2.hubspot.net/hubfs/304927/Downloads/GWI%20Social%20Summary%20Q3%202017.pdf.

[MR08] E Maler ir D Reed. The Venn of Identity: Options and Issues in Federated Identity Management. IEEE Security and Privacy, 6(2):16–23, 2008. ISSN: 15407993. DOI: 10.1109/MSP.2008.50. URL: http://innovbfa.viabloga.com/files/IEEESecPriv%7B%5C_%7D%7B%5C_%7D%7B%5C_%7DVenn%7B%5C_%7Dof%7B%5C_%7DIdentity%7B%5C_%7D%7B%5C_%7D%7B%5C_%7D2008.pdf.

[MS07] Tewfiq El Maliki ir Jean-marc Seigneur. A Survey of User-centric Identity Management Technologies. International Conference on Emerging Security Information Systems and Technologies:12–17, 2007. DOI: 10.1109/SECURWARE.2007.6. URL: http://ieeexplore.ieee.org/xpls/abs%7B%5C_%7Dall.jsp?arnumber=4385303.

[Nak08] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. Disertacija, 2008. URL: www.bitcoin.org.

[PM03] Andreas Pashalidis ir Chris J. Mitchell. A taxonomy of single sign-on systems. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2727 LNCS:249–264, 2003. ISSN: 03029743.

[Ral14] Raluca Budiu. Login Walls Stop Users in Their Tracks, 2014. URL: https://www.nngroup.com/articles/login-walls/ (tikrinta 2017-06-10).

[RR12] V. Radha ir D. Hitha Reddy. A Survey on Single Sign-On Techniques. Procedia Technology, 4:134–139, 2012. ISSN: 22120173.

[Sam99] V Samar. Single sign-on using cookies for Web applications. Proceedings IEEE 8th International Workshops on Enabling Technologies Infrastructure for Collaborative Enterprises WET ICE99:158–163, 1999. ISSN: 10801383.

[SB12] San-Tsai Sun ir Konstantin Beznosov. The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems. Proceedings of the 2012 ACM conference on Computer and communications security, p. 378–390, 2012. URL: http://delivery.acm.org/10.1145/2390000/2382238/p378-sun.pdf?ip=193.219.95.141%7B%5C&%7Did=2382238%7B%5C&%7Dacc=ACTIVE%20SERVICE%7B%5C&%7Dkey=1FA3353941FE8055.0BB7C649D41C6C66.4D4702B0C3E38B35.4D4702B0C3E38B35%7B%5C&%7D%7B%5C_%7D%7B%5C_%7Dacm%7B%5C_%7D%7B%5C_%7D=1524915309%7B%5C_%7D.

[Sed18] Kai Sedgwick. Verge Is Forced to Fork After Suffering a 51% Attack - Bitcoin News, 2018. URL: https://news.bitcoin.com/verge-is-forced-to-fork-after-suffering-a-51-attack/ (tikrinta 2018-05-02).

[SPM+11] San-Tsai Sun, Eric Pospisil, Ildar Muslukhov, Nuray Dindar, Kirstie Hawkey ir Konstantin Beznosov. What makes users refuse web single sign-on? Proceedings of the Seventh Symposium on Usable Privacy and Security - SOUPS '11:1, 2011. DOI: 10.1145/2078827.2078833. URL: http://dl.acm.org/citation.cfm?doid=2078827.2078833.

[SSH+09] C Satchell, G Shanks, S Howard ir J Murphy. Identity crisis: user perspectives on multiplicity and control in federated identity management, 2009. DOI: 10.1080/01449290801987292. URL: http://www.tandfonline.com/action/journalInformation?journalCode=tbit20%20https://doi.org/10.1080/01449290801987292.

[Sza97] Nick Szabo. Nick Szabo – The Idea of Smart Contracts, 1997. URL: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/idea.html (tikrinta 2018-05-04).

[Ten17] Tendermint. Tendermint - Blockchain Consensus, 2017. URL: https://tendermint.com/ (tikrinta 2018-05-02).

[VW14] Pamela Vagata ir Kevin Wilfong. Scaling the Facebook data warehouse to 300 PB | Engineering Blog | Facebook Code | Facebook, 2014. URL: https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb/ (tikrinta 2018-05-03).

[ZNP15] Guy Zyskind, Oz Nathan ir Alex 'Sandy' Pentland. Decentralizing Privacy: Using Blockchain to Protect Personal Data. 2015 IEEE Security and Privacy Workshops, p. 180–184. IEEE, 2015-05. ISBN: 978-1-4799-9933-0. DOI: 10.1109/SPW.2015.27. URL: http://ieeexplore.ieee.org/document/7163223/.

[ZXD+17] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen ir Huaimin Wang. Blockchain Challenges and Opportunities : A Survey. International Journal of Web and Grid Services, 2017. URL: https://www.researchgate.net/profile/Hong-Ning%7B%5C_%7DDai/publication/319058582%7B%5C_%7DBlockchain%7B%5C_%7DChallenges%7B%5C_%7Dand%7B%5C_%7DOpportunities%7B%5C_

%7DA%7B%5C_%7DSurvey/links/59d86d50a6fdcc2aad0a2f2a/Blockchain-Challenges-and-Opportunities-A-Survey.pdf.

Sąvokų apibrėžimai

Adresas - blokų grandinės kontekste tai identifikatorius, blokų grandinėje identifikuojantis paskyrą. Paskyra gali būti asmens arba išmaniojo kontrakto.

Atpažinimo duomenys (angl. *credentials*) - tai duomenys, skirti asmens autentifikavimui. Jie gali būti asmens atributai (pvz. biometriniai duomenys, tokie kaip pirštų antspaudai ar balsas) ar sugalvota informacija (pvz. slapyvardis ir slaptažodis). Dažniausiai internete naudojami atpažinimo duomenys yra identifikatoriaus (slapyvardžio ar el. pašto) ir slaptažodžio pora [MR08].

Atributas - charakteristika, susieta su esybe (pvz. fiziniu asmeniu). Galimi fizinio asmens atributai: gimimo data, vardas, ūgis, pirštų antspaudai [Cam04]. Atributas gali būti laikinas (pvz. adresas) arba nuolatinis (pvz. asmens kodas).

Autentifikavimas - tai procesas, kurio metu patvirtinama sąsaja tarp tapatybės ir jos identifikatoriaus (t.y., įrodoma, kad asmuo iš tikrųjų yra tas, kas sakosi esąs) [Cam04; Kuk11]. Šiam patvirtinimui naudojami atpažinimo duomenys. Autentifikavimo pavyzdys: *tavo pateiktas slapyvardis ir slaptažodis patvirtina, kad tu esi Jonas Jonaitis*.

Autorizavimas - tai procesas, kurio metu leidžiama arba draudžiama asmeniui atlikti konkretų veiksmą, priklausomai nuo jo identifikatoriaus ar atributo [Cam04]. Pavyzdys: *kadangi tau yra daugiau nei 18 metų, tu gali nusipirkti energetinį gėrimą*.

Identifikatorius - tai atributas, kuris vienareikšmiškai susiejamas su jį pateikiančiu asmeniu ir kurį sunku arba neįmanoma pakeisti. Fizinio asmens identifikatoriaus pavyzdys galėtų būti gimimo data (žmogus gali apie ją meluoti, tačiau gimimo datos pakeisti neįmanoma) [Cam04]. Skaitmeninio identifikatoriaus pavyzdys yra naudotojo elektroninio pašto adresas.

Identifikavimas - tai procesas, kurio metu asmuo susiejamas su jo identifikatoriumi [Cam04]. Identifikavimo pavyzdys yra asmens ir jo vardo susiejimas: *tu esi Jonas Jonaitis*.

Paslaugų tiekėjas (angl. *service provider*) - tai betkokia taikomoji programa, kuri suteikia naudotojui tam tikrą paslaugą ar norimą turinį. Galimi paslaugų tiekėjai yra interneto tinklapiai, susirašinėjimo programos ar kitos taikomosios programos, į kurias kreipiasi naudotojas [PM03; Sam99]. Paslaugų tiekėjas gali turėti vieną ar kelias paslaugas, kurioms reikia tapatybės valdymo funkcijų. Darbe paslaugų tiekėjas dar gali būti vadinamas pasikliaujančiąja šalimi (angl. *relying party*).

Prieigos raktas (angl. *token*) - tai objektas, identifikuojantis skaitmeninę tapatybę [Mic07]. Šis raktas būna išduodamas tapatybės tiekėjo ir skirtas identifikuoti naudotoją. Raktas būna prisegtas prie visų autentifikuoto naudotojo užklausų ir leidžia paslaugos tiekėjui žinoti, koks naudotojas kreipiasi.

Skaitmeninė tapatybė - abstrakti fizinės esybės reprezentacija, sudaryta iš aibės esybės nuolatinių ar laikinų atributų, kurie susiejami su fizine esybe [Cam04; GV09]. Fizinė esybė gali būti fizinis arba juridinis asmuo. Šiame darbe, jei nenurodyta kitaip, kalbama apie fizinio asmens skaitmeninę tapatybę.

Skaitmeninės tapatybės valdymas (angl. *digital identity management*) - tai veiksmų, skirtų kontroliuoti tapatybę ir su ja susijusius procesus, visuma [DP08]. Į tai įeina autentifikavimas, autorizavimas, prieigų kontrolė, tapatybės gyvavimo ciklo valdymas bei saugus tapatybės atributų

perdavimas trečiosioms šalims [CY10].

Tapatybės tiekėjas (angl. *identity provider*) - servisas ar taikomoji programa, skirta koordinuoti su tapatybe susijusius duomenis tarp naudotojų, jų naršyklių bei paslaugų tiekėjų [Kuk11]. Pagrindinės tapatybės tiekėjo funkcijos: infrastruktūros naudotojų tapatybės duomenims apdoroti sukūrimas ir užklausų iš paslaugų tiekėjų bei naudotojų apdorojimas [CY10].

Vienkartinis prisijungimas (angl. *single sign on*) - tai mechanizmas, kuris naudoja vieną tapatybės patvirtinimo veiksmą suteikti naudotojui prieigą prie susijusių, bet nepriklausomų taikomųjų programų ar interneto svetainių, neprašant naudototojo iš naujo prisijungti prie kiekvienos iš jų konkrečios sesijos metu [RR12].

Priedas nr. 1

Atributų saugojimo ir autorizavimo kontraktas

```
pragma solidity ^0.4.23;

contract UserAttributeStore {
    struct ServiceAttribute {
        bool accessGranted;
        string value;
    }
    mapping (address => mapping(uint => mapping(address => ServiceAttribute)))
     private attributes;

    event AccessRequested (
        address userAddress,
        address serviceAddress,
        uint attributeId
    );
    event AccessChanged (
        address userAddress,
        address serviceAddress,
        uint attributeId,
        bool status
    );

    function grantAccess(uint attributeId, address serviceAddress, string
   value) public {
        changeAccess(attributeId, serviceAddress, true, value);
    }

    function removeAccess(uint attributeId, address serviceAddress) public {
        changeAccess(attributeId, serviceAddress, false, "X");
    }

    function changeAccess(uint attributeId, address serviceAddress, bool
   granted, string value) private {
        attributes[msg.sender][attributeId][serviceAddress].accessGranted =
   granted;
        attributes[msg.sender][attributeId][serviceAddress].value = value;
        emit AccessChanged(msg.sender, serviceAddress, attributeId, granted);
    }

    function requestAccess(uint attributeId, address userAddress) public {
        emit AccessRequested(userAddress, msg.sender, attributeId);
    }

    function getAttribute(uint attributeId, address userAddress, address
   serviceAddress)
```

```
41          public
42          view
43          returns (string)
44      {
45          if (msg.sender == userAddress) {
46              return attributes[msg.sender][attributeId][serviceAddress].value;
47          }
48          //to avoid additional 'initialized' property, length check used
49          if (bytes(attributes[userAddress][attributeId][serviceAddress].value).
    length == 0) {
50              return "Please request access first";
51          } else {
52              if (attributes[userAddress][attributeId][serviceAddress].
    accessGranted) {
53                  return attributes[userAddress][attributeId][serviceAddress].
    value;
54              } else {
55                  return "Access denied";
56              }
57          }
58      }
59  }
```

Priedas nr. 2

Paslaugų registro kontraktas

```
pragma solidity ^0.4.23;

contract ServiceRegister {

    struct Service {
        string registrationCode;
        string name;
    }
    mapping (address => Service) private services;

    function register(string registrationCode, string name) public {
        services[msg.sender].registrationCode = registrationCode;
        services[msg.sender].name = name;
    }

    function getService(address serviceAddress) public view returns (string, string) {
        return (services[serviceAddress].registrationCode, services[serviceAddress].name);
    }
}
```