VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERINIO IR DUOMENŲ MODELIAVIMO KATEDRA

Kursinis darbas

Išmaniųjų kontraktų panaudojimas išsaugant ir išduodant e-sertifikatus Ethereum tinkle

Atliko:
Paulius Vilkauskas

Vadovas:
lekt. Linas Būtėnas

Vilnius
2019

Turinys

Sutartinis terminų žodynas

- AES - Advanced Encryption Standard - simetrinės kriptografijos algoritmas
- dApp - decentralizuota aplikacija - aplikacija, veikianti peer-to-peer kompiuterių tinkle
- Ether - Ethereum platformos kriptovalita
- Gas - Mokestis už transakcijas, atliktas Ethereum tinkle
- p2p - peer-to-peer - tinklo modelis, kuriame keitimasis resursais vyksta tiesiogiai tarp tinklo vartotojų

Santrauka

Šiame tiriamojo tipo kursiniame darbe aprašoma Blockchain technologija, aprašoma Ethereum platforma, kriptografija, skaitmeniniai parašai, IPFS sistema bei nagrinėjamas realaus pasaulio, decentralizuotos dokumentų pasirašymo aplikacijos prototipas. Šio darbo tikslas buvo išnagrinėti kaip pasinaudojus Ethereum išmaniaisiais kontraktais galima talpinti ir pasirašinėti e-sertifikatus bei kitus legalius dokumentus.

Raktiniai žodžiai: blockchain, ethereum, ipfs, išmanieji kontraktai, e-sertifikatai, skaitmeninis paraš

Summary

Storage and Issue of E-certificates Using Smart Contracts in Ethereum network

This research-oriented coursework paper contains information about Blockchain technology, Ethereum blockchain platform, cryptography, digital signatures, InterPlanatery File System as well as a review of a real-world document signing decentralized application prototype. The main goal of this paper was to research how Ethereum Smart Contracts could be used to store and sign e-sertificates as well as other legal documents.

Keywords: blockchain, ethereum, ipfs, smart contracts, e-certificate, digital signature

Įvadas

2008 m. grupė žmonių, kurie save indetifikavo kaip Satoshi Nakamato, konceptualizavo anksčiau pasauliui pateiktą blokų grandinės idėją, sukurdami Bitcoin. Pačią blockchain technologiją galima apibūdinti kaip blokų grandinę, kurioje kiekvienas blokas laiko transakcijų duomenis, savo bei praėjusio bloko maišos rezultatą. Grandinę užtvirtina tinklo dalyviai, sprendžiant sudėtingas skaičiavimo problemas. Tinklo dalyviai konkuruoja tinkle, kad kuo efektyviau išspręstų problemą, taip pridėdami naują bloką prie blokų grandinės. Nors Blockchain technologija vis dar stipriai siejama, ir pagrinde žinoma dėl Bitcoin, šiandien yra sunkiai suskaičiuojamas kiekis platformų, besinaudojančiu Blockchain technologija. Viena svarbiausių jų yra Ethereum. Tai atviro kodo platforma, leidžianti kurti decentralizuotas aplikacijas bei naudotis išmaniaisiais kontraktais. Tai yra teisinių kontraktų skaitmeninė versija, veikiant Ethereum tinkle. Išmanieji kontraktai gali saugoti duomenis bei saugiai, patikimai ir automatiškai atlikti tam tikrus, iš anksto apibrėžtus teisinius veiksmus, taip pašalinant reikiamybę kreiptis į teisininkus, ar notarus. Šiame darbe nagrinėjama kaip veikia Blockchain technologija bei Ethereum platforma ir jos palaikoma technologija - Išmanieji kontraktai. Taip pat nagrinėjamas dokumentų pasirašymas bei validavimas, naudojantis įvairiais kriptografijos konceptais. Galiausiai, siekiant išsiaiškinti kaip Išmaniųjų Kontraktų technologiją galima panaudoti, talpinant e-sertifikatus, darbe nagrinėjama papildomos Ethereum technologijos bei aptariamas realaus pasaulio, decentralizuotas dokumentų pasirašymo prototipas ir teoriškai apsvarstomas minimo prototipo praplėtimas.

1. Blockchain

1.1. Blockchain technologija

Iš esmės, Blockchain yra grandinė duomenų struktūrų-konteinerių, vadinamų blokais, kurie laiko atžvilgiu yra suskirstyti į linijinę seką. Blokai laiko įvairius duomenis, savo maišos reikšmę (ang. hash), ankstesnio bloko maišos reikšmę, laiko žymą ir Bitcoin atveju - transakcijų duomenis. Blockchain technologijos istorija prasideda 1979 m., kai Ralph Merkle užpatentavo duomenų struktūrą, vadinamą Merkle medžiu (ang. Merkle Tree). Tai gali būti laikoma labai primityvia blokų grandine. Ši duomenų struktūra veikė patvirtindama ir tvarkydama kompiuterinių sistemų duomenis. Peer-to-peer kompiuterių tarpusavio tinkle duomenų patvirtinimas yra labai svarbus norint užtikrinti, kad duomenų perdavimo metu niekas nebūtų pakitę. Iš esmės Merkle medžiai buvo ir yra naudojami išlaikyti bei įrodyti bendrai naudojamų duomenų vientisumą. 1991 m. „Merkle“ medis buvo naudojamas „saugios blokų grandinės“ sukūrimui - duomenų įrašų serijai, kur kiekvienas grandinės įrašas būtų prijungtas prie praėjusio. Naujausiame grandinės įraše būtų visos grandinės istorija. [1] 2008 m. Grupė žmonių, kurie save indetifikavo kaip Satoshi Nakamato, konceptualizavo paskirstytą blokų grandinę (ang. Distributed blockchain). Joje būtų laikoma saugi duomenų mainų istorija, naudojant peer-to-peer tarpusavio ryšių tinklą duomenų apsikeitimai turėtų laiko žymas bei būtų verifikuoti, tuo pačiu viską valdant autonomiškai be centrinių institucijų. Tai tapo Bitcoin pagrindu.

„Blockchain“ technologija užtikrina dvigubo išlaidų (ang. double-spending) problemos pašalinimą, naudojant viešųjų raktų kriptografiją, kur kiekvienam agentui priskiriamas privatus raktas (laikomas slaptai kaip slaptažodis) ir viešas raktas, bendrai prieinamas visiems kitiems agentams.[7] Transakcija inicijuojama, kai būsimas monetų (skaitmeninių žetonų) savininkas siunčia savo viešąjį raktą pradiniam savininkui. Monetos perduodamos maišos reikšmės skaitmeniniu parašu. Viešieji raktai yra kriptografiškai sugeneruoti adresai, saugomi blokų grandinėje. Kiekviena moneta yra surišta su adresu, o transakcija kripto-ekonomikoje (ang. crypto-economy) yra tiesiog monetų mainai monetomis iš vieno adreso į kitą. Sveikintinas bloko grandinės bruožas yra tai, kad viešieji raktai niekada nėra susieti su realaus pasaulio tapatybe. Transakcijos, nors ir atsekamos, yra vykdomos neatskleidžiant asmens tapatybės; tai yra pagrindinis skirtumas tarp kripto ir įprastinių valiutų.

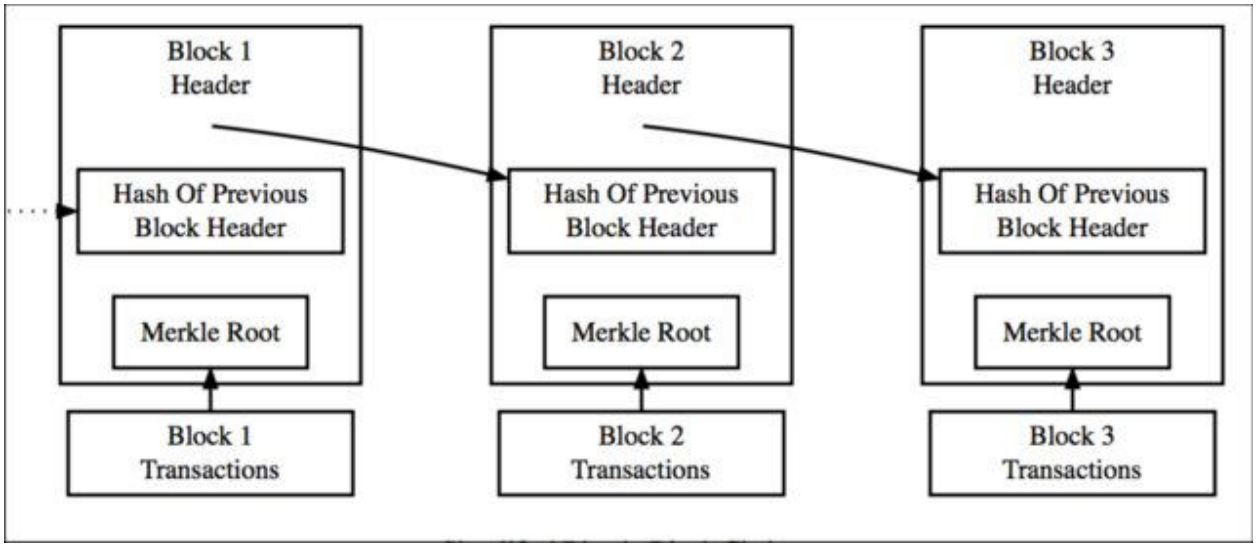
1.2. Kalnakasiai ir skaičiavimo problemų sprendimas

Blokų grandinė yra sandorių įrašų grandinė, kurią tinklo dalyvių pogrupis (taip pat žinomas kaip miners) užtvirtina, sprendžiant sudėtingas matematines skaičiavimo problemas. [7] Šie tinklo dalyviai (toliau - “kasėjai”) intensyviai (ir anonimiškai) konkuruoja tinkle, kad kuo efektyviau išspręstų matematinę problemą, taip pridėdami naują bloką prie blokų grandinės. Bloko atlygis (t. y. naujos monetos) siunčiamas į dalyvio viešąjį adresą. Jei dalyvis nori išleisti šias monetas, jis turi pasirašyti su atitinkamu privačiu raktu. Kai didėja darbo atlikimo (mining) pajėgumas, atitinkamai sudėtingėja ir matematinės problemos. Sudėtingumo lygis yra nustatytas taip, kad naujų blokų gaminimas būtų pastovus - Bitcoin platformoje tai yra apie 10 minučių tarp blokų sukūrimo.

1.3. Maišos funkcijos

Kriptografinė maišos funkcija yra speciali klasė maišos funkcijų turinti tam tikras savybes, kurios leidžia ją naudoti kriptografijoje. Tai matematinis algoritmas, kuris atvaizduoja savavališko

dydžio duomenis į fiksuoto dydžio bitų eilutę (hash) ir yra suprojektuotas būti vienpusio funkcija, t.y. funkcija, kurios yra neįmanoma invertuoti. Įvesties duomenys dažnai vadinami pranešimu (ang. message), o išvestis (maišos vertė arba maiša) dažnai vadinama pranešimo santrauka (ang. digest). Blokų grandinėje maišos funkcijos yra naudojamos atvaizduoti esamą blockchain būseną. Tokiu būdu įvestis atspindi viską, kas įvyko blokų grandinėje, todėl kiekviena tranksakcija iki to momento, kartu su naujais papildomais duomenimis yra pridedami. Tai reiškia, kad rezultatas yra suformuotas visų prieš tai įvykusių transakcijų, įvykusių blokų grandinėje. Kaip buvo minėta anksčiau, menkiausias bet kokios informacijos dalies pakeitimas lemia didžiulį rezultato pokytį, kas užtikrina neginčijamą blokų grandinės technologijos saugumas. Pakeitus bet kokį įrašą, kuris anksčiau įvyko bloko grandinėje, būtų pakeisti visi maišos rezultatai, kas įrašus padaro klaidingais ir pasenusiais. Tai tampa neįmanoma, kai atsižvelgiama į skaidrų bloko grandinės pobūdį, nes bet kokie pokyčiai įvyktų aiškiai matant visam tinklui. Pirmasis bloko grandinės blokas, žinomas kaip šakninis (genesis) blokas, apima savo transakcijas, kurios, sukombinuotots ir validuotos, sukuria unikalų maišos rezultatą. Šis maišos rezultatas kartu su visomis naujomis transakcijomis, kurios yra vykdomos yra panaudoti kaip įvestis naujam maišos rezultatui, kuris yra naudojamas sekančio bloko grandinėje. Tai reiškia, kad kiekvienas blokas yra surištas su prieš tai buvusiu bloku savo maišos rezultatu, kas suformuoja grandinę iki pat šakninio bloko. Dėl to ši technologija ir yra pavadina blokų grandine. Tokiu būdu transakcijos gali būti pridėtos saugiai, kol visi tinklo dalyviai sutaria koks turėtų būti bloko maišos rezultatas.



2. Ethereum

Ethereum yra atvirai prieinama, decentralizuota, blockchain technologijos paremta platforma, kuri leidžia programuotojams kurti bet diegti decentralizuotas aplikacijas, tokias kaip išmanieji kontraktai. Ethereum yra ne tik platforma, bet ir Turing-complete programavimo kalba, veikianti ant blokų grandinės. Ethereum tikslas yra sujungti ir tobulinti skriptų, „altcoins“ (alternatyvūs Bitcoin) ir „grandininių“ meta-protokolų sąvokas ir leisti kūrėjams kurti savavališkas sutarimu pagrįstas programas, kurios būtų išplėčiamos, lengvai standartizuojamos bei funkciškai išsamios. Ethereum tai daro kuriant tai, kas iš esmės yra kertinis abstraktumo pamatinis sluoksnis: bokų grandinė, turinti integruotą Turing-complete programavimo kalbą, leidžiančią bet kam kurti išmaniuosius kontraktus ir decentralizuotas programėles, kur jie - kūrėjai, galėtų patys kurti savo nuosavybės taisykles, transakcijų formatus bei būsenos pasikeitimo funkcijas. Išmanieji kontraleidžiantis visiems rašyti sumanias sutartis ir decentralizuotas programas, kur jie gali sukurti savo savavališkas taisykles dėl nuosavybės, sandorių formatų ir valstybės pereinamojo laikotarpio funkcijos. Blockchain tinklas yra sudarytas iš daug atšakų, priklausančių tinklo dalyviams, atliekančiams anksčiau minėtas darbo operacijas (mining) bei kitų atšakų, kurios neatlieka jokių matematinių skaičiavimų naujų blokų gaminimui, bet padeda vykdyti išmaniuosius kontraktus bei transakcijas. Šios atšakos yra žinomos kaip EVM (Ethereum virtuali mašina). Kiekviena atšaka yra sujungta su kita tinklo atšaka. Šios tinklo atšakos naudoja peer-to-peer protokolą komunikavimui tarpusavy. Kiekvienas “kasėjas” išlaiko naujausią tinko būseną, turinčia visus grandinės blokus (angl. ledger). Šį konceptą dar galima apibūdinti kaip buhalterinę knygą, kurioje aprašytos visos įvykusios bei validžios transakcijos.

2.1. Ethereum paskyros

[4] Ethereum platformos būsena yra sudaryta iš objektų, vadinamų paskyromis (angl. accounts), kur kiekviena paskyra turi 20 baitų ilgio adresą, o būsenos pasikeitimas yra tiesioginis reikšmės ir informacijos keitimasis tarp paskyrų. Ethereum paskyra turi keturis laukus:

- Bereikšmis skaičius (angl. the nonce), užtikrinantis, kad kiekviena transakcija galėtų būti vykdoma tik kartą.
- Paskyros Ether balansas
- Paskyros kontrakto kodas (jei toks yra)
- Paskyros talpa (numatytuoju atvėju tuščia)

“Ether“ yra pagrindinė „Ethereum“ valiuta, kuri yra naudojama mokėti sandorių mokesčius. Apskritai yra dviejų tipų paskyros: išorės sąskaitos, valdomos privačiais raktais, ir kontraktų sąskaitos, kontroliuojamos pačio kontrakto kodo. Išorinė paskyra neturi jokio kodo; jos savininkas gali siųsti žinutes pasirašant transakciją. Tuo tarpu, kontraktinė paskyra, gavusi žinutę iš išorės, aktyvuoja savo kodą/programą, leidžiant skaityti bei rašyti į vidinę atmintį, siųsti kitas žinutes, ar aktyvuoti kitus kontraktus.

Žinutės ir transakcijos Ethereum platformoje, transakcijos (naudojamos Bitcoin) yra vadinamos žinutėmis. Jos skiriasi nuo Bitcoin transakcijų keletu aspektų:

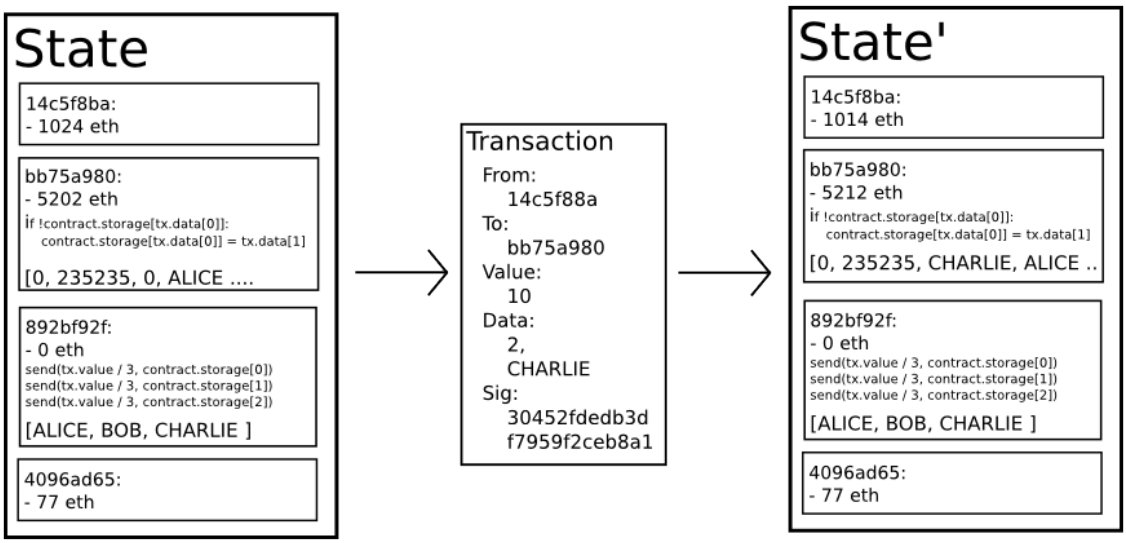
1. Ethereum žinutė gali būti sukuri arba išoriškai, arba kontrakto, kai Bitcoin transakcija sukurta gali būt tik išoriškai.

2. Ethereum žinutė gali laikyti duomenis, kai Bitcoin transakcija gali tik perduoti valiutą.

3. Jei žinutės gavėjas yra kontraktinė paskyra, ji turi galimybė gražinti reikšmę/atsakymą, o tai reiškia, kad Ethereum žinutės apima ir funkcijos konceptą.

Terminas transakcija naudojamas „Ethereum“ sistemoje apibūdinti pasirašytą duomenų paketą, kuris laiko žinutę, siunčiamą iš išorinės paskyros. Transakcijas sudaro žinutęs gavėjas, siuntėją identifikuojantis parašas, ether kiekis ir siunčiami duomenys bei dvi vertės, vadinamos STARTGAS ir GASPRICE. Siekiant užkirsti kelią eksponentiniam smūgiui ir begaliniam kodui (infinite loop), kiekvienai transakcijai reikia nustatyti ribą, kiek skaičiavimo kodo vykdymo etapų ji gali neršti, įskaitant pradinį pranešimą ir bet kokius papildomus pranešimus, kurie atsiranda vykdymo metu. STARTGAS yra ši riba, o GASPRICE yra mokestis, mokamas kasėjui už kiekvieną skaičiavimo žingsnį.

2.2. Būsenų pasikeitimas



Ethereum būsenos pasikeitimo funkcija APPLY(S,TX) -> S’ gali būti apibrėžtą taip:

1. Patikrinti ar transakcija yra gerai suformuluota (pvz. Ar ji turi teisingą skaičių reikšmių), ar parašas yra validus ir ar bereikšmis kintamasis (nonce) atitinka siuntėjo kintamąjį.

2. Apskaičiuoti transakcijos mokestį šia sandauga: STARTGAS * GASPRICE bei nustatyti siuntėjo adresą iš parašo. Atimti mokestį iš siuntėjo paskyros balanso ir iteruoti siuntėjo nonce. Jei siuntėjo paskyroje nėra pakankamai ether, gražinti klaidą.

3. Priskirti GAS = STARTGAS ir numažinti tam tikrą kiekį mokesčio kiekio kiekvienam transakcijos baitui, skaičiuojant n gas už kiekvieną baitą.

4. Perkelti transakcijos reikšmę iš siuntėjo paskyros į gavėjo paskyra. Jei gavėjo paskyra neegzistuoja - sukurti ją. Jei gavėjo paskyra yra kontraktas, paleisti kontrakto kodą, iki kol jis vis bus įvykdytas, arba baigsis mokestis, skirtas transkaicijai.

5. Jei reikšmės perkėlimas nepavyko dėl siuntėjo nepakankamo balanso, arba mokesčio balanso pabaigos, gražinti visus būsenos pasikeitimus, išskyrus mokesčio apmokėjimus, kurie pridedami į kasėjo paskyros balansą.

6. Kitu atveju, kompensuoti mokestį siuntėjui, už likusį gas ir nusiųsti sumokėtą mokestį kasėjui.

2.3. Išmanieji kontraktai

Visų pirma, kontraktas yra teisinis dokumentas, įpareigojantis dvi ar daugiau šalių, kurios sutinka vykdyti sandorį nedelsiant arba ateityje. Kadangi kontraktai yra teisiniai dokumentai, jie įgyvendinami pagal įstatymus. Sutarties pavyzdys gali būti individualus asmuo, sudarantis sutartį su Draudimo bendrove, kad apdraustų savo sveikatą arba asmuo, perkantis žemės plotą iš kito asmens. Išmanusis kontraktas yra kontraktas, įgyvendintas, įdiegtas ir vykdomas Ethereum aplinkoje. Išmanieji kontraktai yra teisinių kontraktų skaitmeninė versija. Išmanieji kontraktai yra diegiami, laikomi ir vykdomi Ethereum virtualioje mašinoje. Išmanieji kontraktai gali saugoti duomenis. Saugomi duomenys gali būti naudojami įrašinėti informacijai, faktams, balansams ir bet kokiai kitai informacijai, reikalingai realiai pasaulio sutartims įgyvendinti. Išmanieji kontraktai yra labai panašūs į objektinio programavimo (angl. object-oriented programming) klases.Išmanusis kontraktas gali iškviesti kitą išmanųjį kontraktą, tiap pat kaip objektinio programavimo objektas gali sukurti bei naudoti kitų klasių objektus. Išmanųjį kontraktą galima įsivaizduoti kaip mažas programas, turinčias įvairia funkcijas. Galima sukurti kontrakto objektą/egzempliorių ir kviesti funkcijas siekiant peržiūrėti, ar atnaujinti kontrakto duomenis, vykdant tam tikrą vykdymo logiką. Geriausias būdas apibūdinti išmaniuosius kontraktus yra palyginti šią technologiją su gėrymų automatais. Įprastai, norint susitvarkyti su teisiniais dokumentais, žmonės eina pas advokatą ar notarą, moka jiems ir laukia kol gaus reikiamą dokumentą. Naudojantis išmaniaisiais kontraktais pakanka įmesti monetą (pvz. Ether) į automatą ir reikiamas dokumentas automatiškai “įkrentą”į subjekto paskyra. Be to, išmanieji kontraktai ne tik apibrėžia taisykles ir nuobaudas susitarimo klausimu, bet ir automatiškai vykdo šias obligacijas.

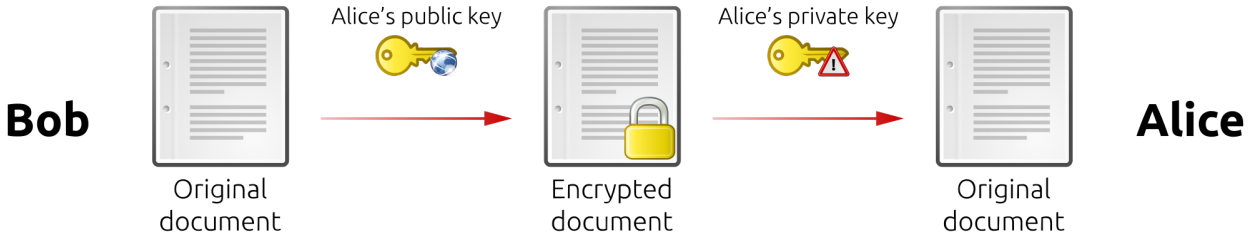
3. Kriptografija

3.1. Simetrinė kriptografija

Simetrinė kriptografija apibūdina procesą, kai užšifravimui ir iššifravimui yra naudojamas tik vienas kriptografinis raktas. Tai reiškia, kad tas pats raktas turėtų būti prieinamas keliems žmonėms, komunikuoti žinutėmis, užšifruotomis šiuo kriptografiniu metodu.

3.2. Asimetrinė kriptografija

Asimetrinė kriptografija reiškia dviejų raktų panaudojimo šifravimui ir iššifravimui procesą. Bet koks raktas gali būti naudojamas šifravimui ir iššifravimui. Pranešimai, užšifruoti su viešuoju raktu, gali būti iššifruoti naudojant privatų raktą, o pranešimai šifruoti privačiu raktu gali būti iššifruoti naudojant viešąjį raktą. Pavyzdžiui: Bob, naudodamas Alice viešąjį raktą, užšifruoja pranešimus ir siunčia juos Alice. „Alice“ gali naudoti savo privatų raktą, kad iššifruotų pranešimą ir gautų jo turinį. „Alice“ viešuoju raktu užšifruotus pranešimus gali iššifruoti tik „Alice“, nes ji turi tik savo privatų raktą ir niekas kitas. Tai yra bendras asimetrinių raktų naudojimo atvejis. Yra ir kitas naudojimas, aprašytas skaitmeninių parašų skyriuje.



4. Skaitmeninis parašas

Skaitmeninis parašas (angl. digital signature) yra labai panašus į asmens parašą ant popieriaus, kadangi abu parašai padeda indetifikuoti asmenį. Be to, tai padeda užtikrinti, kad dokumento/žinutės turinys nėra pakitęs perdavimo metu.

4.1. Pasirašymas, kai norima validuoti pranešimą

Konceptą, kaip veikia skaitmeniniai parašai galima paaiškinti pavyzdžiu, nagrinėjant žemiau esantį pav. Tarkime, kad siuntėjas/pasirašytojas (angl. Signer) yra Alice, o gavėjas/validuojantis asmuo (angl. Verifier) yra Tomas. Alice nori išsiųsti bei pasirašyti dokumentą Tomui. Pasirašymas vyks taip:

1. Alice paims siunčiamą dokumentą, ir maišos funkcijos pagalba ištrauks dokumento maišos rezultatą (hash)

2. Maišos rezultatą ji užšifruos savo privačiu raktu.

3. Užšifruotą maišos rezultatą ji pridės prie siunčiamo dokumento ir visą galutinį rezultatą (dokumentas + užšifruotas dokumentos maišos rezultatas) išsiųs Tomui.

Tomas, gavęs visą žinutę galės pavaliduoti, kad ji atėjo iš Alice atlikdamas validavimo veiksmus.

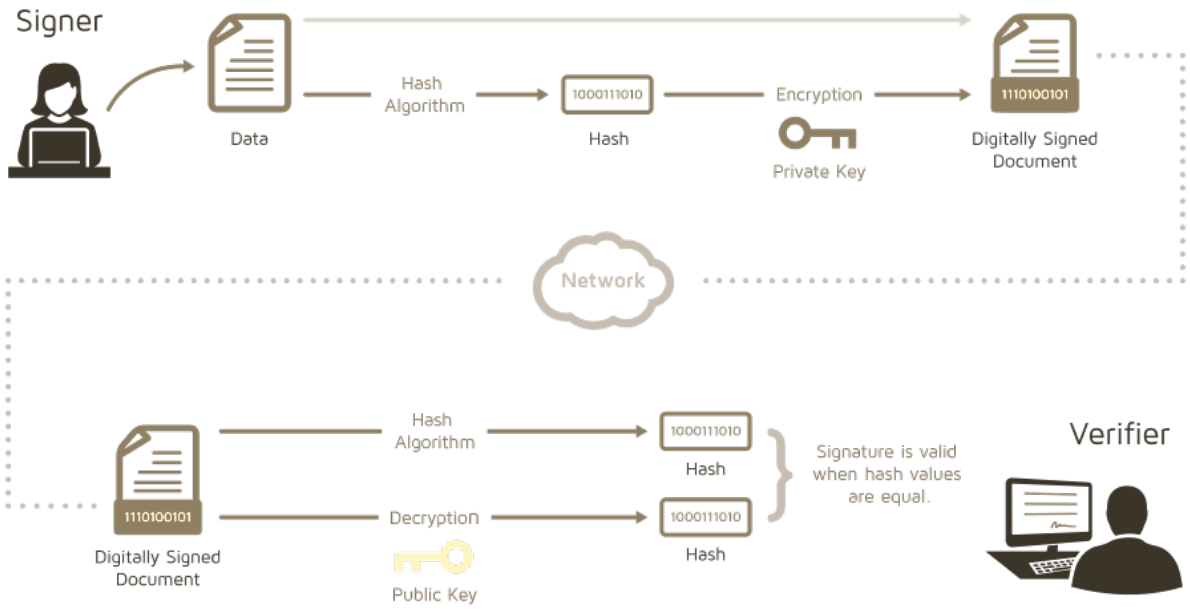
1. Jis atskirs dokumentą nuo užšifruoto maišos rezultato.

2. Pasinaudodamas Alice viešuoju raktu, iššifruos dokumento maišos rezultatą.

3. Panaudojęs maišos funkciją ir patį fokumentą kaip kintamąjį, išgaus dokumento maišos rezultatą.

4. Šioje vietoje Tomas turės du maišos rezultatus: hash x ir hash y. Hash x yra gautas iš Alice kartu su dokumentu, o hash y yra išgautas Tomo, pasinaudojus maišos funkcija. Jei sulyginus hash x ir hash y, jei bus vienodi, tai reikš, kad dokumentas/žinutė buvo nepakeista.

Be to, tai reikš, kad žinutė atėjo iš Alice, nes tik ji maišos rezultatą galėjo užšifruoti savo privačiu raktu. Panašiai skaitmeninis parašas yra naudojamas turto, ar kripto valiutos kaip Ether savininko, pasirašyti transakcijos duomenis.



4.2. Pasirašymas, kai norima paslėpti failo turinį

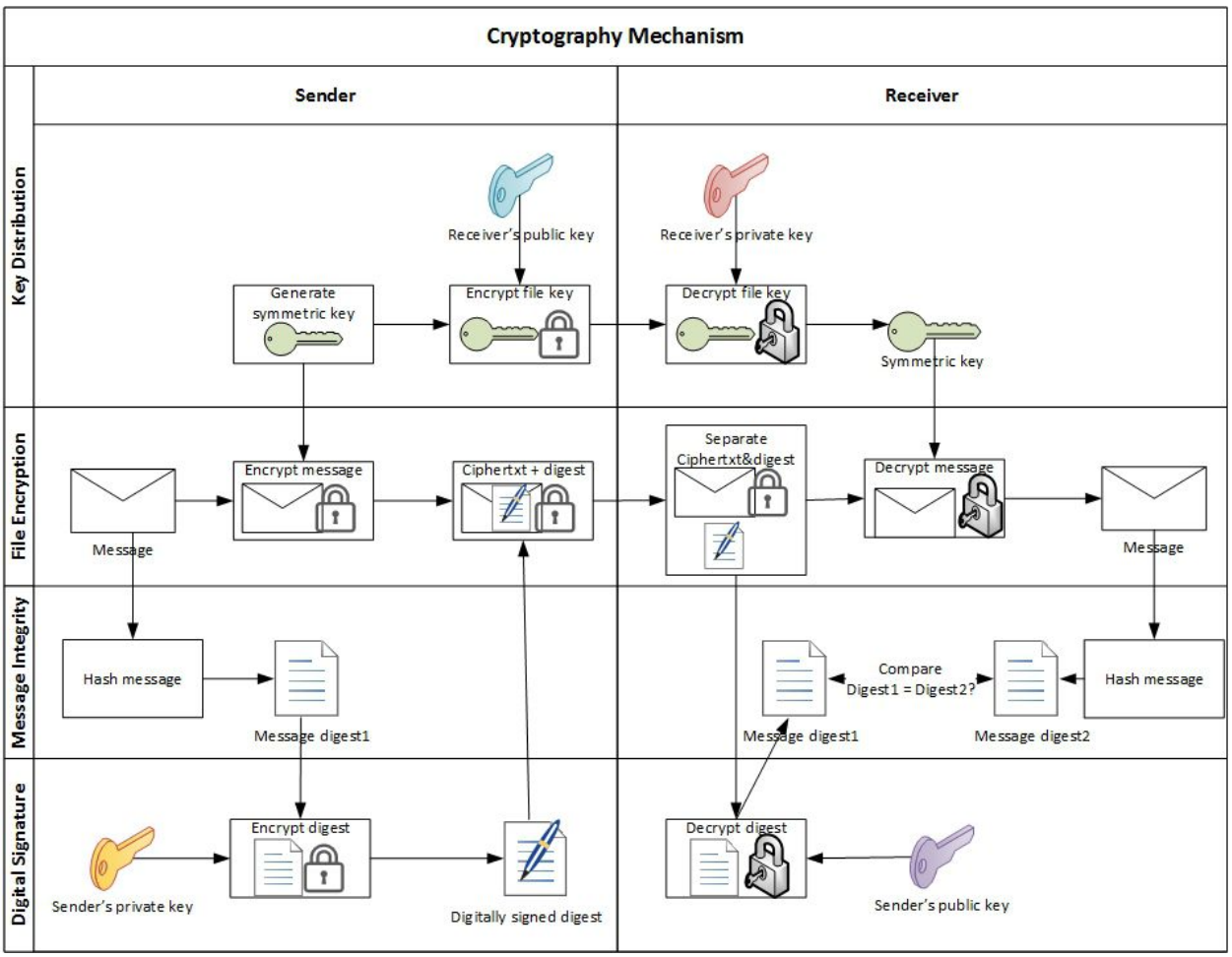
Aukščiau pateiktas skaitmeninio parašo panaudojimo pavyzdys yra labai plačiai naudojamas duomenų perdavime ir komunikacijoj. Tačiau, toks būdas yra neefktyvus, kalbant apie tikrų, jautraus turinio dokumentų (pvz. Pdf formato failų), pasirašymui, kadangi dokumento turinys nėra šifruojamas. Teoriškai tą patį asimetrinės kriptografijos metodą būtų galima panaudoti ir dokumento turinio užšifravimui, tačiau asimetrinė kriptografija reikalauja per daug kompiuterinių resursų, kad pačio dokumento šifravimas viešuoju/privačiu raktu galėtų būti naudojamas efektyviai. Todėl, kai kalbama apie dokumento pasirašyma, kai norima paslėpti ir jo turinį, praktikoje yra naudojamas kur kas efektyvenis būdas - taikoma hibridinė kriptografija. Kitaip tariant, kombinuojami abu, anksčiau minėti kriptografijos metodai - simetrinė kriptografija, kartu su asimetrine kriptografija. Simetrinė kriptografija leidžia efektyviai ir pigiai šifruoti didelius dokumentus, bet pati iš savęs yra nesaugi. Todėl kartu panaudojama ir asimetrinė kriptografija, užšifruoti simterinį raktą bei jau aptartą dokumento maišos rezultatą. Šį procesą galime geriau paaiškinti dar vienu pavyzdžiu. Situacija panaši, kaip ir anksčiau pateiktame pavyzdyje, tačiau šį kartą norima užšifruoti ir patį dokumentą.

Dokumento siuntėjas turėtų atlikti tokius žingsinius:

1. Panaudoti maišos funkciją failui ir išgauti failo maišos rezultatą (Digest 1)
2. Užšifruoti maišos rezultatą savo privačiu raktu
3. Sugeneruoti visiškai atsitiktini simetrinį raktą (pvz. AES-128)
4. Užšifruoti patį failą pasinaudojus sugeneruoti simetriniu raktu
5. Užšifruoti simetrinį raktą gavėjo viešuoju raktu
6. Išsiųsti užšifruotą failą, užšifruotą failo maišos rezultatą bei užšifruotą simetrinį raktą

Dokumento gavėjas, norėdamas iššifruoti failą, bei validuoti visą žinutę atliktu šiuos veiksmus:

1. Iššifruoti simetrinį raktą savo pasinaudojus privačiu raktu
2. Iššifruoti failą pasinaudojus gautu simetriniu raktu
3. Iššifruoti failo maišos rezultatą (Digest 1) pasinaudojus siuntėjo viešuoju raktu
4. Panaudoti maišos funkciją failui ir išgauti failo maišos rezultatą (Digest 2)
5. Palyginti Digest 1 su Digest 2. Jei maišos rezultatai vienodi, vadinasi failas ir visa žinutė buvo nekeista ir yra validi.



4.3. Kelių asmenų pasirašymas

Akivaizdu, kad skaitmeninis parašas labai smarkiai remiasi ne tik kriptografinių raktų technologija bet ir maišos funkcijomis. Vienapusės maišos funkcija suteikia galimybę pasirašyti tik mažą dokumento dalį, kas smarkiai padidina skaitmeninio parašo naudojimo efektyvumą. Be to, tai leidžia parašą laikyti atskirai nuo pačio dokumento. Dėl to galima nesunkiai ir efektyviai valdyti keletą parašų vienam dokumentui - kiekvienas pasirašantysis gali pasirašyti atskirą dokumento maišos rezultatą, o tai leidžia kiekvieną maišos rezultatą - parašą validuoti atskirai. [6]

5. Pagalbinės technologijos

5.1. IPFS

IPFS (InterPlanetary File System) yra peer-to-peer paskirstyta failų sistema, kuri sujungia geriausias ir sėkmingiausias idėjas iš kitų p2p sistemų, tokių kaip DHTs, BitTorrent, Git, SFS. [3] IPFS technologinis indėlis yra supaprastinti, tobulinti ir sujungti patikrintus metodus į vieną darnią sistemą, geresnę už jos dalių visumą. IPFS pateikia naują platformą programų rašymui ir diegimui bei naują didelių duomenų platinimo ir versijavimo sistemą. IPFS atšakos saugo IPFS objektus lokalioje atmintyje. Atšakos jungiasi vienas su kitu ir perduoda objektus. Šie objektai reprezentuoja failus bei kitas duomenų struktūras. Iš esmės, IPFS yra versijuota failų sistema, kuri gali priimti failus ir juos tvarkyti, taip pat juos saugoti kažkur ir po to sekti versijas. IPFS taip pat valdo kaip šie failai juda tinkle, tad IPFS yra paskirstyta failų sistema. Vienas didžiulių, šio p2p hipermedijos protokolo privalumų yra tai, kad puslapiams nebereikia centralizuotos failų talpyklos, kas šiuo metu apibūdina vieną paprasčiausių Interneto principų. Šiuo metu, kai HTTP protokolas yra panaudotas, failai gali būti surasti ir atsisiųsti tik iš vienos lokacijos (kompiuterio ar serverio). Tuo tarpu IPFS leidžia failams būti atsiųstiems iš kelių lokacijų vienu metu, kas šį procesą padaro žymiai greitesniu bei efektyvesniu. Vietoj to, kad medijos objektus (nuotraukas, vaizdo įrašus, straipsnius ir pan.) internete apibūdintų juos talpinantis serveris, IPFS šiuos failus apibūdina jų maišos rezultatais. Šio priėjimo idėja yra ta, kad norint savo naršykle pasiekti tam tikrą puslapį, IPFS klausia viso tinklo “ar kas nors turi ieškomą failą, kuris sutampa su šiuo maišos rezultatu?”, o tinklo atšaka, turinti ieškomą failą, leidžia jį pasiekti. IPFS naudoja turinio adresavimą HTTP sluoksnyje. Tai yra idėjos, kai duomenys yra adresuojami ne pagal lokaciją, o pagal turinį, panaudojimas praktikoje. Tai reiškia, kad failo turinys nustato failo adresą, o ne serveris, kuriame failas yra talpinamas.

5.2. Infura

Vienas iššukių, su kuriais susiduria Ethereum aplikacijų programuotojai, kurie nori diegti savo aplikacijas, kurios sąveikauja su pačiu Ethereum yra ta, kad jų rašomas kodas turi naudoti pilną tinklo atšaką, dalyvaujančia Ethereum blokų grandinės tinkle. [5] Programuotojui, kuriam svarbiau yra skirti laiką rašant pačias aplikacijas, palaikyti pilną atšaką lokaliai užima daug laiko ir resursų. Šiai problemai spręsti buvo sukurtas įrankis, pavadinimu INFURA. Iš esmės, Infura galima laikyti kaip didžiulę Ethereum atšaką bei servisą, teikiantį lengvai naudojamą API (angl. Application programming interface), kuriuo naudojantis vartotojai nesunkiai gali pasiekti Ethereum tinklą bei IPFS atšakas.

5.3. MetaMask

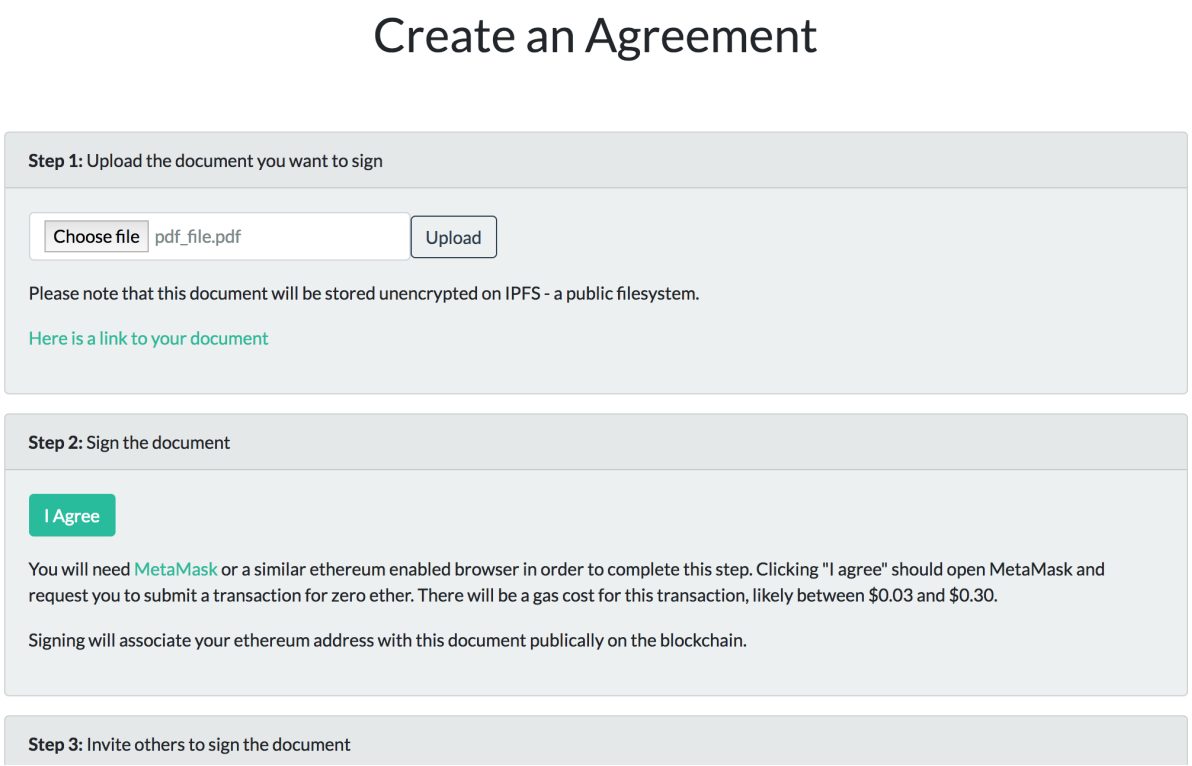
Metamask yra internetinės naršyklės plėtinys, kuris leidžia vartotojams tvirtinti/pasirašinėti išmaniuosius kontraktus, vykdyti transakcijas bei sąveikauti su dApps (decentralizuotomis aplikacijomis), neturint pilnos tinklo atšakos savo kompiuteryje. Vietoj to, MetaMask vartotojams leidžia prisijungti prie INFURA atšakos ir vykdyti išmaniuosius kontraktus toje atšakoje. MetaMask palaiko vartotojo Ethereum piniginę, kurioje yra laikoma Ether valiuta bei leidžia naudoti tą valiutą įvairioms transkacijoms, naudojantis įvairiom decentralizuotom aplikacijom.

6. OpenSign pavyzdys

6.1. Puslapis

OpenSign (sukurta komandos Atchai Digital) yra nemokamas, atviro kodo, decentralizuotos aplikacijos prototipas, sukurtas, panaudojus Embark karkasą, kuris naudoja web3 biblioteką, skirtą komunikuoti su Ethereum atšakomis. [2] OpenSign - tai decentralizuota alternatyva jau esamiems produktams kaip DocuSign ir EchoSign. Iš esmės, OpenSign, vartotojams leidžia talpinti failus į IPFS bei juos pasirašinėti be reikiamybės turėti patikimą trečios šalies atstovą (kas yra būtina, naudojant įprastas dokumentų pasirašymo platformas). OpenSign naudoja anksčiau minėtas papildomas technologijas - pasinaudojus Infura atšakomis, failai yra talpinami į IPFS, priėjimą prie IPFS suteikia MetaMask, o pasirašius patalpintą dokumentą, skaitmeninis parašas yra saugomas išmaniajame kontrakte. OpenSign vartotojams leidžia naudotis trimis funkcijomis:

1. Įkelti dokumentą, kurį yra norima pasirašyti. Dokumentas tuomet yra patalpinamas į IPFS
2. Pasirašyti įkeltą dokumentą. Sutikus pasirašyti (Paspaudus “I Agree”), MetaMask pasiūlys patvirtinti transakciją. Susimokėjus transakcijos mokestį, pasirašančiojo viešasis raktas išsaugotas išmaniajame kontrakte.
3. Pakviesti kitus asmenis pasirašyti ikeltą dokumentą. Vartotojui, ikėlusiam dokumentą, suteikiama nuoroda, kurią jis/ji gali persiųsti kitiems asmenims, kad jie pasirašytų dokumentą. Pakviesti asmenys atlieka tą patį procesą, paminėtą antrame punkte.



6.2. Išmanusis kontraktas

OpenSign naudojamas išmanusis kontraktas turi tris funkcijas: addDocument() - (pridėti dokumentą), signDocument() - (pasirašyti dokumentą) bei getSignatures() - (gražinti parašus). Naršyklėje, paspaudus “I Agree”, pasirašant dokumentą bei patvirtinant transakciją per MetaMask, OpenSign aplikacija iškviečia addDocument() ir signDocument() metodus. Iškvietas addDocument() metodas, išmaniajame kontrake sukuria Document struktūros objektą, kuris yra laikomas documents[] masyve. “documents[]” masyvo elementai yra atpažystami pagal unikalų maišos rezultatą, sudarytą iš IPFS failo adreso maišos rezultato bei tuometinės laiko žymos. Į tai svarbu atkreipti dėmesį, nes pačiame išmaniajame kontrakte nėra talpinamas failo IPFS adresas, o talpinamas tik unikalus, patalpintą faila indetifikuojantis kodas, kuris iškvietus signDocument() metodą, yra surišamas su pasirašančiojo viešuoju raktu.

```
pragma solidity ^0.4.17;

contract OpenSign{
    struct Document {
        uint timestamp;
        bytes ipfs_hash;
        address[] signatures;
    }
    mapping(address => bytes[]) public users; //maps addresses to agreement id
    mapping(bytes32 => Document) public documents; //maps keccak256(agreement_id) hashes to documents

    function addDocument(bytes id, bytes ipfs) public {
        users[msg.sender].push(ipfs); //Add document to users's "signed" list
        address[] memory sender = new address[](1);
        sender[0] = msg.sender;
        documents[keccak256(id)] = Document(block.timestamp, ipfs, sender);
    }

    function signDocument(bytes id) public {
        users[msg.sender].push(id);
        documents[keccak256(id)].signatures.push(msg.sender);
    }

    function getSignatures(bytes id) public view returns (address[]) {
        return documents[keccak256(id)].signatures;
    }
}
```

6.3. OpenSign prototipo praplėtimas e-sertifikatų talpinimui

Šio darbo rašymo metu, OpenSign yra prototipo stadijoje, tačiau tai yra puikus pavyzdys, kaip pasinaudojus Ethereum išmaniaisiais kontraktais bei papildomis Ethereum tinkle naudojamomis technologijomis, galima talpinti bei pasirašinėti dokumentus. Remiantis OpenSign prototipu, kaip baze didesnei platformai, būtų galima sukurti sertifikatų talpinimo sistemą. Šiuo metu OpenSign leidžia talpinti dokumentą, jį pasirašyti bei validuoti, kad dokumentas yra pasirašytas. Norint šį prototipą praplėsti svarbių dokumentų talpinimui, visų pirma reiktų talpinamą dokumentą užšifruoti. Tam galima pasitelkti simetrinio šifravimo technologija, aptarta kriptografijos bei skaitmeninio pasirašymo skyrelyje. Pasinaudojus simetrinius AES šifro raktu, galima efektyviai bei greitai paslėpti jautrius sertifikato duomenis, prieš talpinant failą į IPFS. Patį AES raktą, skritą ne tik užšifruoti, bet ir iššifruoti failą, būtų galima talpinti į išmanųjį kontraktą, kaip patalpintodokumento objekto atributą. Dokumento objektą galima praplėsti pagal aplikacijos naudojimo poreikius. Šiuo

metu Document objektas turi tris atributus - timestamp (laiko žyma), ipfs_hash (failo unikalus maišos rezultatas), signatures[] masyvas (pasirašiusiųjų viešųjų adresų masyvas). Kaip minėta prieš tai, vienas šių atributų galėtų būti ir failo AES šifro raktas. Be to, paprastai sertifikatas turi savo subjektą - savininką. Esant reikiamybei, kontrakte galėtų būti talpinama informacija apie savininką, pavyzdžiui, vardas, pavardė ir pan. Viena galima OpenSign prototipo spraga yra ta, kad failai patalpinti į IPFS yra pasiekiami vartotojui išdavus pilną failo IPFS adresą - IPFS URL kartu su failo maišos rezultatu. Tai kelia riziką sertifikato adresui būti išplatintam (kadangi adresas priklauso nuo failo maišos rezultato, pats adresas niekada nesikeis, nebent bus pakeistas failo turinys), kai intencija yra juos saugoti slaptai ir priėjimą suteikti tik tam tikriems asmenims. Vienas galimų būdų spręsti šią problemą yra nesuteikti viso adreso vartotojui. Vietoj to, programos viduje parsisiųsti ieškomą dokumentą ir tik tuomet dokumentą pateikti vartotojui.

Išvados ir rekomendacijos

Pagrindinis šio tiriamojo darbo tikslas buvo išnagrinėti kaip pasinaudojus Ethereum išmaniaisiais kontraktais galima talpinti ir pasirašinėti e-sertifikatus bei kitus legalius dokumentus. Norint suprasti kaip veikia išmanieji kontraktai bei dokumentų pasirašymas, buvo nagrinėjama ne tik išmaniuosius kontraktus palaikanti platforma Ethereum ir Blockchain technologija, bet ir skaitmeniniai parašai, kriptografijos konceptai bei pagalbinės Ethereum technologijos. Tyrimo kulminacija - decentralizuoto, dokumentų pasirašymo aplikacijos prototipo, naudojančio išmaniuosius kontraktus - OpenSign nagrinėjimas ir teorinis svarstymas apie minimo prototipo praplėtimą, e-sertifikatų talpinimui.

Literatūros šaltiniai

[1] Digital trends. what is blockchain.
https://www.digitaltrends.com/computing/what-is-a-blockchain.

[2] Atchai. Opensign.
https://github.com/atchai/opensign.

[3] Juan Benet. Ipfs - content addressed, versioned, p2p file system.
https://arxiv.org/abs/1407.3561.

[4] Vitalik Buterin. A next-generation smart contract and decentralized application platform, 2013.
https://github.com/ethereum/wiki/wiki/White-Paper.

[5] ConsenSys. Looking for an easy way to connect your dapp to ipfs and ethereum?
https://media.consensys.net/looking-for-an-easy-way-to-connect-your-dapp-to-ipfs-and-ethereum-c78

[6] Jorge Ortiz Olga Russakovsky, Adam Sadovsky. Modern cryptography: Theory and applications.
https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/cryptography/digsig.html.

[7] Marc Pilkington. Blockchain technology: Principles and applications.
https://papers.ssrn.com/sol3/Papers.cfm?abstract_id=2662660.