KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Irmantas Zenkus

VIDUTINIŲ SUMŲ MOKĖJIMO SISTEMOS SUKŪRIMAS IR TYRIMAS

Magistro darbas

Vadovas
prof. dr. E. Sakalauskas

KAUNAS, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. N. Listopadskis
2011 06 02

VIDUTINIŲ SUMŲ MOKĖJIMO SISTEMOS SUKŪRIMAS IR TYRIMAS

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
(parašas) doc. dr. J. Jonaitis
2011 06 02

Recenzentas
(parašas) doc.dr. K.Plukas
2011 06 02

Atliko
FMMM-9 gr. stud.
(parašas) I. Zenkus
2011 06 02

KAUNAS, 2011

Summary

Zenkus I. Creation and research of medium amount payment system : Master's work in applied mathematics / supervisor dr. prof. E. Sakalauskas; Department of Applied mathematics, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2011. – 42 p.

SUMMARY

Traditional models of electronic money, usually used for electronic payment systems are not adapted for small amounts of high-volume transactions. When the product is very low value, such as music, art, transport tickets, traditional schemes are far from ideal. Payments must be made very quickly, without requiring significant system resources. As we know the system resources cost money, and some low-value goods would become cheaper than the transaction costs.

In this work i present the electronic money model based on the conceptual WebCoin electronic payment scheme. Model is extended by using a fully hierarchical proxy signature, Schnorr signature scheme and some small changes in the WebCoin scheme. Closed circuit of bank, customers and vendors is extended to fully hierarchical tree of banks. Blind signature from fully hieararchical electronic signature scheme is substituted with a classic Shnorr electronic signature scheme.

Instead of two denominations of coins used by the WebCoin scheme, I introduce an additional denomination. In this way, I improve a system of micro amounts to medium-sized payments. Medium amount payments in this work are such that the transaction amount does not exceed 100 LTL, but other interpretations are possible. The system is prepared to be used with unlimited amounts, but in offline mode it is more difficult to prevent duplication or counterfeiting of money. Medium amount payments are simply not worth the risk.

Analytical part of this work introduces different electronic money schemes and types of signatures, and review their properties. Methodological section presents models and schemes that are used in this work. It also will present theorems showing properties of schemes. Research part will describe the creation of functioning model and code of the package in which the model is realized.

TURINYS

ILIUSTRACIJŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

ĮVADAS

Tradiciniai elektroninių pinigų modeliai, dažniausiai naudojami elektroninių mokėjimų sistemose nėra pritaikyti didelio kiekio mažų sumų sandoriams. Kai prekė yra labai mažos vertės, pavyzdžiui muzikos kūrinys, straipsnis, transporto bilietas, tradicinės schemos toli gražu nėra idealios. Mokėjimai turi būti atliekami ypatingai sparčiai, nereikalaujant didelių sistemos resursų. Kaip žinia sistemos resursai kainuoja pinigus, todėl itin mažos vertės prekės taptų pigesnėmis negu sandoriui atlikti reikalingų sisteminių resursų kaštai.

Šiame darbe pateikiamas elektroninių pinigų sistema, paremta WebCoin elektroninių mokėjimų konceptualia schema. Sistema išplėsta panaudojant pilnai hierarchinį įgaliojantį elektroninį parašą, Schnorr elektroninio parašo schemą ir įvesti nedideli pokyčiai pačioje WebCoin schemoje. Uždara vieno banko ir daugelio klientų bei pardavėjų schema išplečiama iki pilnai hierarchinio bankų medžio. Aklasis parašas iš pilnai hierarchinio įgaliojančio elektroninio parašo schemos yra keičiamas klasikiniu Shnorr elektroniniu parašu.

Vietoje WebCoin schemoje naudojamų dviejų nominalų monetų įvedamas papildomas nominalas. Tokiu būdu iš mikromokėjimų sistemos gaunu vidutinių sumų mokėjimų sistemą. Vidutiniais mokėjimais šiame darbe laikau tokius, kurių sandorio suma neviršija 100Lt, tačiau galimos ir kitos interpretacijos. Sistemą puikiai galima naudoti ir neriboto dydžio mokėjimams, bet esant atjungties režimui sunkiau apsisaugoti nuo dvigubo pinigo išleidimo ar padirbinėjimo. Vidutinės sumos mokėjimai nėra verti rizikos, kurią užsitraukia nesąžiningas vartotojas.

Savo darbo analitinėje dalyje supažindinsiu su skirtingais elektroninių pinigų ir parašų tipais, apžvelgsiu jų savybes. Metodologinėje dalyje pateiksiu modelius ir schemas, kuriuos naudojau savo darbe. Taip pat pateiksiu teoremas, įrodančias schemų savybes. Tiriamojoje dalyje aprašysiu sukurto modelio veikimo schemą, pateiksiu matematinio paketo kodą, kuriame realizuojamas modelis.

1 ANALITINĖ DALIS

1.1 Mikromokėjimai

(Burton Rosenberg, 2010: 163) Mikromokėjimas yra finansinė operacija su labai mažomis sumomis. Toks terminas atsirado dėl internete esančio mažos vertės turinio pirkimo. 1990 metais buvo susikūręs judėjimas, siekiantis sukurti mikromokėjimų standartus. W3C (pasaulinio tinklo konsorciumas) mėgino įtraukti mikromokėjimus į HTML standartą. Net buvo idėjų sukurti mokėjimų užklausų klaidų kodus. Tačiau bandymai liko bandymais ir mikromokėjimai netapo populiarūs perkant internete esantį turinį. Šiuo metu populiariausias tiekėjas, teikiantis mikromokėjimų paslaugas yra PayPal.

Elektroninių pinigų pionieriai Rivest ir Shamir (1996) pasiūlė mikromokėjimų sistemą, pavadinimu Payword. Pagrindinis tikslas buvo sukurti tokį pinigų modelį, kuris veiktų ypatingai sparčiai ir nereikalautų daug sistemos resursų. Maišos funkcijos veikia arti šimto kartų greičiau negu klasikinės elektroninio parašo schemos. Greitis reikalingas dėl to, kad maža pirkinio vertė nebūtų mažesnė už pačią sandorio savikainą. Be to mažesnės sumos nesudaro sąlygų padirbinėjimui. Juk neapsimoka naudoti superkompiuterį, kad padirbti mažos vertės elektroninį pinigą.

1.2 Įprastos elektroninių pinigų schemos

Kalbant apie elektroninius pinigus turime galvoje kompiuterinius tinklus, internetą, skaitmeniniu būdu saugomą informaciją. Elektroniniai pinigai yra viena įdomesnių kriptografijos sričių, tačiau realus elektroninių pinigų panaudojimas dar tik randa savo nišą. Šioje srityje labiausiai pasižymi Japonija.

Elektroniniai pinigai yra tam tikra sistema, kuri internete sudaro sąlygas keistis pinigais su kitais sistemos dalyviais. Iš esmės sistema gali veikti ne tik internete tačiau ir atjungties režimu.

1.2.1 Centralizuotos sistemos

Vienos sistemos (PayPal, WebMoney, CashU ir kitos) parduoda elektroninius pinigus tiesiai vartotojui, kitos parduoda elektroninę valiutą per elektronines valiutos keityklas.

Hong Konge veikiančioje Octopus kortelėje elektroniniai pinigai veikia panašiai kaip reguliariuose bankuose. Kai bankas gauna pinigus iš kliento, pinigai padedami į banką, o ne į kortelę.

1.2.2 Decentralizuotos sistemos

Decentralizuotų mokėjimų sistemų pavyzdys yra BitCoin. Tai veikianti vartotojas-vartotojui elektroninių pinigų sistema su maksimaliu infliacijos limitu.

1.2.3 Atjungties anoniminės sistemos

Tokiose sistemose pardavėjas neturi bendradarbiauti su banku, priimdamas pinigą iš kliento. Vietoje to pardavėjas surenka pinigus pas save, o vėliau juos perkelia į banką. Iš esmės ir perkėlimas gali būti atjungties rėžimu, nes pardavėjas galėtų fiziškai nunešti duomenų laikmeną į banką ir iškeisti elektroninius pinigus į fizinius. Bet kokiu atveju sistema turi garantuoti, kad pinigai, kuriuos gavo pardavėjas, bus priimti banke. Tokiose sistemose turi būti įvesta saugiklių nuo dvigubo to paties pinigo išleidimo. Taip pat klientas turi būti apsaugotas nuo pardavėjo nesąžiningumo.

Šioms savybėms užtikrinti yra naudojama kriptografija ir skaičių teorija. Kai kurios atjungties rėžimo elektroninio parašo schemos pasižymi anonimiškumu, neatsekamumu.

1.2.4 Savybės

Dažnai elektroninių pinigų operacijos yra vykdomos neprižiūrint jokiai kontroliuojančiai institucijai, todėl schemos tenkina tam tikras savybes.

Privatumas - tiksliau tariant duomenų apsauga. Tai neabejotinai svarbu, nes asmeninė vartotojo informacija negali būti laisvai prieinama kitiems sistemos dalyviams ar įsibrovėliams

Vartotojo identifikavimas - apsauga nuo apsimetimo. Bet kuri schema turi užtikrinti, kad kiti dalyviai žinotų su kuo užmegztas ryšys.

Žinučių patikimumas - apsauga nuo klastojimo ir sukeitimo atakų. Sistemos dalyvis turi žinoti, kad siuntėjo žinutė yra lygiai tokia pati, kokią jis siuntė.

Neatsižadėjimas - apsauga nuo vėlesnio sandorio paneigimo. Svarbu, kad būtų užfiksuojamas sandorio faktas.

Paminėtos savybės galėtų būti apibendrinamos kaip identifikacija. Tokios savybės gali būti pasiektos keliais būdais. Populiariausia technika, kuri leidžia tą pasiekti yra žinutės pasirašymas elektroniniu parašu. Tam dažnai naudojami simetriniai raktų protokolai, kai žinutė pasirašoma tik sau pačiam žinomu raktu, o patikrinama visiems žinomu viešu raktu.

1.3 Įgaliojantis parašas

Pirmuosius įgaliojančius parašus pristatė mokslininkai Mambo, Usuda ir Okamoto 1996 metais (Mambo; Usuda; Okamoto: 1996). Vėliau buvo pasiūlyta daugybė įgaliojančių parašų schemų. Įgaliojančio parašo schema leidžia įgaliotiems pasirašytojams pasirašyti žinutes tikrojo pasirašytojo vardu. Tokie parašai greit rado savo nišą ir buvo pritaikyti kai kuriose sistemose. Tai ypač svarbu tokiose sistemose, kur skaičiavimai yra išdalinami dalimis ir reikia įgalioti kitas šalis pasirašyti savo vardu. Be to įgaliojantys parašai gali įtraukti tam tikrus specifinius parašus, taip gaunant naujo tipo įgaliojančius

parašus. Pasirodė tokie parašai kaip ribinis įgaliojantis parašas, įgaliojantis multi-parašas, įgaliojantis aklasis parašas ir kiti.

Įgaliojantis aklasis parašas yra svarbus tokiame scenarijuje: elektroninių pinigų schemoje klientas paprašo banko aklai pasirašyti monetą naudojant aklo parašo schemą. Kai vartotojas norės eiti į kitą banką (dukterinį), jis galės gauti parašą ant savo pinigo pirmojo banko vardu.

1.4 Aklasis parašas

Pirmąjį aklą parašą (Chaum: 1983) 1983 metais pristatė D.Chaum. Tai yra elektroninio parašo atmaina, kuri leidžia žinutės autoriui prašyti pasirašyti kitą žmogų, neatskleidžiant žinutės turinio. Kelio schemos buvo transformuotos į aklo parašo schemas.

Pirmas aklasis parašas buvo paremta RSA schema. T. Okamoto vėliau pasiūlė aklą Schnorr parašą, o D. Pointcheval įrodė jo saugumą.

1.5 Naudojama programinė įranga

Modelio realizacijai pasirinkau matematinį paketą Wolfram Alpha. Mano pasirinkimą lėmė šios savybės:

- Paketas veikia Linux operacinėje sistemoje
- Palaiko operacijas su labai dideliais skaičiais
- Turi įdiegtas funkcijas atsitiktiniam pirminiam skaičiui generuoti
- Turi įdiegtas funkcijas nustatyti ar skaičius pirminis ar ne
- Palaiko didelių skaičių kėlimą laipsniu su moduline operacija

1.6 Darbo uždaviniai

- Konceptualią maišos funkcijomis paremtą elektroninių pinigų schemą paversti veikiančiu modeliu
- Papildyti modelį pilnai hierarchiniu įgaliojančiu parašu
- Papildyti modelį taip, kad patogiau būtų galima vykdyti vidutinio dydžio mokėjimus
- Realizuoti matematinį modelį programos kodu pasirinktame matematiniame pakete

2 METODOLOGINĖ DALIS

2.1 Modulinė aritmetika ir grupės

Viešo rakto kriptografijos ir elektroninių pinigų skaičiavimuose dažniausiai naudojami skaičiai nuo 0 iki $p-1$, kur p yra didelis pirminis skaičius. Taip pat dažniausiai šie skaičiai yra keliami laipsniais nuo 1 iki q, kur q nusako kitą pirminį skaičių q, kuris dalina skaičių $p-1$. Taigi daugyba ir kėlimas laipsniu vyksta grupėse $Z^*(p)$ ir $G(q)$.

Šios grupės ypatingai svarbios kriptografijoje. Turime aibę $\{1, 2, \ldots, p-1, p-2\}$. Jeigu dauginsime bet kuriuos du elementus iš šios grupės ir sumažinsime gautą skaičių moduline operacija $mod\, p$, gautas skaičius taip pat priklausys tai grupei. Be to jeigu bet kuriam skaičiui k iš aibės egzistuoja kitas skaičius k^{-1}, toks, kad $k \cdot k^{-1} = 1\, mod\, p$, reiškia šis skaičius turi atvirkštinį elementą daugybos atžvilgiu.

Apibrėžimas 1. Elementų aibė vadinama grupe su daugybos operacija $*$, jeigu tenkina šias savybes:

1. Uždarumas - bet kokiems elementams $a, b \in G$, sandaugos operacijos rezultatas taip pat priklauso grupei G
2. Asociatyvumas - operacija $*$ yra asociatyvi, t.y. $(g_1 * g_2) * g_3 = g_1 * (g_2 * g_3)$ visiems $g_i \in G$
3. Neutralus elementas - egzistuoja neutralus elementas e operacijos $*$ atžvilgiu, toks kad $e * g = g * e = g$
4. Atvirkštinis elementas - kiekvienam elementui egzistuoja simetrinis elementas operacijos $*$ atžvilgiu.

$$g * g^{-1} = g^{-1} * g = e$$

čia $g \in G$, $g^{-1} \in G$ simetrinis elementas (atvirkštinis).

Jeigu prie grupės $Z^*(p)$ pridėsime elementą 0, gausime grupę $Z(p)$ susidedančią iš visų liekanų $mod\, p$ ir0. Pastebėsime, kad kiekvienas elementas turi atvirkštinį dėl to, kad p yra pirminis skaičius. Todėl didžiausias bendras daliklis tarp p ir bet kurio grupės elemento bus 1. Ši savybė negaliotų jeigu modulinę aritmetiką taikytume sudėtiniam skaičiui.

Šioje grupėje galime apibrėžti daugiau operacijų. Dalybą iš k galime apibrėžti kaip k^{-1}.

2.2 Diskretaus logaritmo uždavinys

Pirmas paskelbtas darbas apie viešo rakto konstravimą, pagal Diffie ir Hellman yra paremtas diskretaus logaritmo uždaviniu baigtiniame lauke F_p, laikant kad F_p yra baigtinis laukas iš pirminio skaičiaus

elementų kiekio. Patogumo dėlei pakaitomis naudosiu žymėjimus F_p,Z,Z_p. Čia Z_p yra baigtinė grupė su nuliniu elementu.

Teorema. Tarkim p yra pirminis skaičius. Tuomet egzistuoja toks elementas $g \in F_p^*$, kurį pakėlus visais laipsniais gausime visus F_p^* elementus.

$$F_p^* = \left\{1, g, g^2, g^3, \ldots, g^{p-2}\right\}$$

Tokį elementą g vadiname grupės generatoriumi.

Reiškia kiekvienas nenulinis elementas iš F_p yra lygus kažkokiam g laipsniui. Atskirai imant, $g^{p-1} = 1 \, mod \, p$ pagal Fermat mažąją teoremą. Ir nė vienas mažesnis g laipsnis nėra lygus 1.

Apibrėžimas. Tegul g yra grupės F_p generatorius o h yra nenulinis grupės elementas. Diskretaus logaritmo problemos (DLP) uždavinys yra rasti tokį laipsnį x, kad

$$g^x \equiv h \, mod \, p$$

Skaičius x yra vadinamas h logaritmu pagrindu g ir yra žymimas $log_g(h)$.

Pastaba 1. Diskretaus logaritmo uždavinys yra surasti sveikąjį skaičių x, tokį kad $g^x = h \, mod \, p$. Tačiau jeigu yra bent vienas sprendinys, tai jų yra begalybė, nes Fermat mažoji teorema teigia, kad $g^{p-1} \equiv 1 \, mod \, p$. Taigi jeigu x yra lygties $g^x = h \, mod \, p$ sprendinys, tai $x + k(p-1)$ taip pat yra sprendinys visiems k, nes

$$g^{x+k(p-1)} = g^x \cdot \left(g^{p-1}\right)^k \equiv h \cdot 1^k \equiv h \, mod \, p$$

Tokiu būdu $log_g(h)$ yra nusakomas tik pridedant ar atimant tam tikrą kiekį dėmenų $p-1$. Kitais žodžiais tariant $log_g(h)$ yra apibrėžtas moduliu $p-1$. Nesunku įsitikinti, log_g duoda tokią funkciją

$$log_g : F_p^* \longrightarrow \frac{\mathbb{Z}}{(p-1)\,\mathbb{Z}}$$

Kartais konkretumo labui turime omenyje patį diskretų logaritmą kaip sveikąjį skaičių x, esantį tarp 0 ir $p-2$, tenkinantį tapatybę $g^x \equiv h \, mod \, p$.

Pastaba 2. Nesunku įrodyti, kad

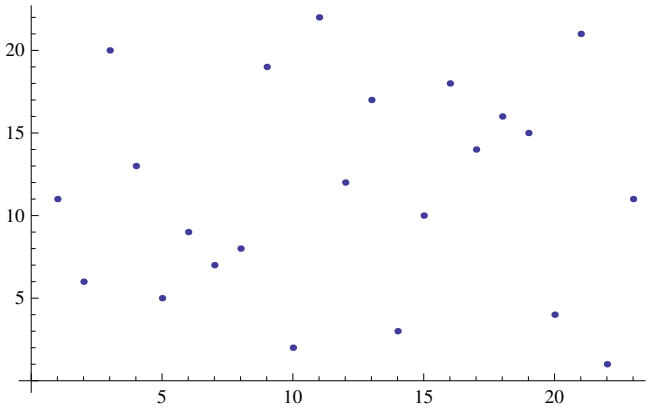
$$log_g(ab) = log_g(a) + log_g(b) \quad visiems \; a, b \in F_p^*$$

Taigi pagrįstai galime $log_g a$ vadinti logaritmu, nes sandauga paverčiama suma taip pat kaip įprastoje logaritmo funkcijoje. Matematikos terminologijoje diskretus logaritmas log_g taip pat yra grupės izomor-

fizmas iš F_p^* į $\frac{\mathbb{Z}}{(p-1)\mathbb{Z}}$.

Pavyzdys. Skaičius $p = 56509$ yra pirminis ir galima patikrinti $g = 2$ yra grupės Z_p generatorius. Kaip apskaičiuotume diskretų logaritmą $h = 38679$? Pats paprasčiausias metodas yra pakelti generatorių laipsniais, kol gausime laipsnį lygų 38679. Būtų sunku tai padaryti rankiniu būdu, tačiau pasitelkus kompiuterį galime rasti, kad $log_p h = 11235$. Galime įsitikinti skaičiuojant $2^{11235} = 38679 \, mod \, p$.

Pastaba 3. Turime ypatingai pabrėžti, kad diskretus logaritmas mažai panašus į įprastinį logaritmą, apibrėžtą realiųjų ar kompleksinių skaičių aibėje. Tačiau terminologija yra pagrįsta, nes abiem atvejais kėlimo laipsniu operacija yra apverčiama, tačiau kėlimas laipsniu moduliu p labai neįprastas, lyginant su įprastu kėlimu laipsniu. Kad laipsnio kėlimo operacija moduliu p atrodo atsitiktinė, galima pastebėti vien žiūrint į kėlimo laipsniais grafiką su moduline operacija 2.1.



2.1 pav.: Skaičius 11 pakeltas laipsniais nuo 1 iki 23 (mod 23)

Pastaba 4. Šis diskretaus logaritmo problemos apibrėžimas apima ir prielaidą, kad pagrindas g yra grupės generatorius, bet tai nėra griežtas reikalavimas. Bendrai tariant bet kuriems $g \in F_p^*$ ir $h \in F_p^*$ diskretaus logaritmo uždavinys yra rasti tokį laipsnį x, kuris tenkintų $g^x \equiv h \, mod \, p$, laikant kad toks skaičius iš viso egzistuoja.

Apibendrinus dar labiau, vietoj imant baigtinio lauko nenulinius elementus ir dauginant juos tarpusavyje ar keliant laipsniu, galime imti elementus iš bet kurios grupės ir vadovautis grupės savybėmis vietoj daugybos. Tokiu būdų gautume plačiausią diskretaus logaritmo problemos apibrėžimą.

Apibrėžimas 2. Tarkime G yra grupė, kurios daugybos operaciją žymėsime $\star$. Diskretaus logaritmo uždavinys yra grupei G nustatyti tokį sveikąjį skaičių x, visiems g ir h priklausantiems G, kad

$$\underbrace{g \star g \star \ldots \star g}_{x \, kartu} = h$$

2.3 Maišos funkcijos

Asimetrinių parašų schemose, tokiose kaip RSA praktiškai niekada nėra pasirašoma pati žinutė. Žinutė gali būti tekstas, knyga, filmas ar kitas didelės apimties turinys. Jeigu Alisa nori pasirašyti ilgą žinutę, tai gali labai ilgai užtrukti. Tuomet ji turėtų žinutę skaidyti į atskirus blokus ir pasirašinėti juos atskirai. Bronius tikrindamas parašą taip pat turės kiekvieną bloką tikrinti ir lipdyti žinutę iš naujo. Be to elektroninis parašas gali tapti panašios apimties kaip pati žinutė.

Norint apeiti šią problemą, žinutės nėra pasirašomos. Pasirašyti reikia žinutės pėdsaką. Šis pėdsakas gali būti laikomas maišos funkcijos rezultatu. Maišos funkcija yra vadinamas algoritmas, skirtas apskaičiuoti tam tikram rezultatui. Toliau panagrinėsiu maišos funkcijas, rezultatus ir jų vaidmenį kriptografijoje.

2.3.1 Maišos rezultatai ir funkcijos

Matematiškai galima būtų teigti, kad jeigu funkcija f yra maišos funkcija, h yra funkcijos rezultatas, o m yra funkcijos argumentas, tuomet galioja $h = f(m)$. Dydis h yra ribotas, o m galima parinkti neriboto dydžio.

Nekriptografinis atvejis Tipinė situacija, kur taikoma maišos funkcija: Kriptobankas išduoda savo klientams identifikacinius numerius sudarytus iš skaičių. Kadangi bankas turi tik kelis tūkstančius klientų, keturženkliai skaičiai pilnai tenkina jų poreikius. Bet iš patirties bankas žino, kad klientai dažnai pamiršta vieną ar kelis savo sąskaitos numerio skaičius. Todėl Kriptobankas nusprendė vietoje keturženklių numerių išduoti penkiaženklius. Paskutinis skaičius tuo atveju yra maišos rezultatas, gautas iš pirmų keturių skaičių. Jeigu Aldona pamiršta vieną skaičių, maišos funkcija dažniausiai nesutampa ir iškart tampa aišku, kad numeris neteisingas.

Kolizijos Mūsų atveju turime 10000 įėjimo reikšmių (visi keturženkliai), kuriems yra tik dešimt maišos rezultatų (nuo 0 iki 9). Žinoma, kad gali taip nutikti, kad maišos rezultatas yra teisingas, nors skaičiai ne. Toks atvejis kai du ar daugiau argumentų turi tą patį maišos rezultatą, vadinamas kolizija. Naudojant gerą maišos funkciją tokie atvejai praktiškai nepasitaiko, o jeigu pasitaiko, maišos funkcija turi tenkinti šiuos reikalavimus:

- Jeigu testuojame visas įmanomas reikšmes, tas pats maišos rezultatas turėtų pasirodyti daugiau mažiau vienodu dažnumu
- Maišos rezultatas turi pakisti net nuo pačių mažiausių pokyčių funkcijos argumente.

Paprasta maišos funkcija Šiuo atveju Kriptobankas turi kelis pasirinkimus kalbant apie maišos funkcijas. Paprasčiausias jų yra sudėti visus keturženklio skaičiaus skaitmenis ir gautą skaičių dalinti iš dešimties, paliekant liekaną (moduliarinė operacija).

2.3.2 Kriptografinė maišos funkcija

Norėdama pasirašyti žinutę, Aldona gali pritaikyti žinutei maišos funkciją, tačiau jeigu panaudotų paprasčiausią atvejį, kai skaitmenys yra sudedami ir pritaikoma moduliarinė operacija, algoritmas būtų ypač nesaugus, nes perėmus žinutę būtų labai nesunku sugalvoti kitą žinutę kuri duotų tą patį maišos rezultatą. Tokiu būdu Aldona pasirašytų žinutę, kurios nerašė.

Jeigu Aldona šiuo atveju panaudotų kurią nors maišos funkciją kuri nėra taikoma kriptografinėse sistemose, ši problema vis tiek išliktų. Taip yra nes kriptografinėms maišos funkcijoms yra keliami aukštesni reikalavimai.

Reikalavimai kriptografinei maišos funkcijai Tokie reikalavimai keliami funkcijoms, naudojamoms elektroniniam parašui:

- Kiekvienas maišos rezultatas turi pasirodyti vienodu dažnumu, bet kuo rečiau
- Mažas argumento pokytis turi duoti didelį rezultato pokytį
- Per priimtiną laiką turi būti labai sunku arba neįmanoma surasti koliziją

Maišos funkcija, tenkinanti šiuos reikalavimus vadinama kriptografine maišos funkcija. Atitinkamai rezultatas vadinamas kriptografiniu maišos funkcijos rezultatu. Šiuolaikinės maišos funkcijos neveikia minėtu sumos pagrindu, o yra paremtos simetrinio rakto kriptografija. Visi algoritmai dalina žinutę į blokus ir dirba su kiekvienu bloku atskirai po kelis kartus. Kadangi kriptografinės maišos funkcijos turi veikti greičiau nei elektroninio parašo skaičiavimas, jos yra paremtos elementariomis dvejetainėmis operacijomis. Todėl žinutę visada reikia įsivaizduoti ne kaip tekstą ar skaičių, o kaip dvejetainėje sistemoje užrašytą eilutę. Tipinis maišos funkcijos rezultatas yra 160 bitų, o argumento dydis yra nesvarbus.

Reikia pažymėti, kad maišos funkcija nieko neužkoduoja ir nėra jokių raktų. Taigi bet kas gali paskaičiuoti žinutės maišos rezultatą, išskyrus kelias išimtis tam tikrais atvejais.

Todėl iš principo sukeitimo ataka veikia su bet kuria maišos funkcija ir bet kuriuo argumentu. Vienintelis atsakas, kuriuo gali pasinaudoti Aldona - pasirinkti maišos rezultatą tokio dydžio, kad būtų užtikrinamas saugumas. Kuo ilgesnis maišos rezultatas, tuo daugiau skirtingų žinučių turi būti patikrinta, kad būtų įmanoma rasti koliziją. Pavyzdžiui jeigu norima surasti atitinkamą žinutę kuriai buvo pritaikyta 160 bitų funkcija, turi būti išmėginama 2^{160} maišos rezultatų. Jeigu tarsime, kad rezultatą pasiektume

išmėginę pusę rezultatų, vis tiek turėsime išmėginti $2^{159} = 7.3 \cdot 10^{47}$ rezultatų, kuriuos mėginant skaičiuoti pačiu geriausiu superkompiuteriu turėtų trukti ilgiau, negu numatoma egzistuoti visatai.

Gimtadienio ataka Pamėginkime įsivaizduoti: kiek kambaryje turi būti žmonių, kad būtų 50% tikimybė, kad bent du žmonės bus gimę tą pačią dieną?

Atsakymas gana įdomus: 22. Tai reiškia, kad jeigu surinksime kelias grupes po 22 žmones, tai daugiau negu pusėje jų bus du žmonės, gimę tą pačią dieną.

Panagrinėkime nuodugniau. Tarkim metai turi d dienų. Tuomet žmonių skaičius, kurį reikia surinkti norint turėti porinius gimtadienius su 50% tikimybe bus mažesnis už $\sqrt{d}$.

Kuo tai susiję su kriptografinėmis funkcijomis? Paprasta: jeigu kenkėjas nori sužinoti, kiek vidutiniškai jam reikia išmėginti žinučių, kol jis ras koliziją, jam tereikia prisiminti gimtadienio uždavinį. Tokiu atveju įmanomi maišos funkcijos rezultatai atitinka dienas metuose, o testuojamų žinučių kiekis atitinka žmonių kiekį. Todėl jeigu maišos funkcija yra n ilgio, tuomet yra 2^n įmanomų rezultatų. Tuomet kenkėjas suras antrą tokią žinutę po vidutiniškai $2^{n/2}$ bandymų.

Ataka, kuri remiasi gimtadienio problema yra vadinama gimtadienio ataka. Jeigu maišos rezultatas būtų 128 bitų ilgio, tuomet kenkėjui reiktų 2^{64} bandymų, t.y. 10^{19}. Paprastu kompiuteriu tai truktų dešimtis tūkstančių metų, tačiau superkompiuteriai šią užduotį įveiktų kur kas greičiau, todėl dažniausiai naudojamas bent 160 bitų ilgio maišos rezultatas.

2.4 Schnorr identifikacijos protokolas

Schnorr identifikacijos protokolas yra paremtas diskretaus logaritmo uždaviniu. Tarkim klientas nori įrodyti pardavėjui, kad žino savo slaptą raktą x. To reikia tam, kad įrodyti jog esi tas žmogus kurio reikia, o ne kažkuris kitas, nes niekas kliento slapto rakto žinoti negali.

Klientas siunčia pardavėjui atsitiktinį skaičių w. Pardavėjas atsako atsiųsdamas jam iššūkį - pavyzdžiui skaičių c. Tuomet klientas, norėdamas įrodyti savo tapatybę atsako

$$y = w + x \cdot c \, mod \, q$$

čia q yra bendras sutartas pirminis skaičius. Tačiau šioje procedūroje reikia įvesti papildomų veiksmų, nes šiuo atveju pardavėjas sužinotų c, w ir y o to pasekoje ir x. Reikia taip suformuoti skaičiavimus, kad y būtų teisingas atsakymas į užduotį, tačiau nepavyktų sužinoti x. Pakeliame generatorių $g \in G(q)$ o rezultatui pritaikome $mod \, p$.

$$g^y = g^w g^{xc} \, mod \, p$$

Tuomet pardavėjui bus siunčiamas $g^w = a$. Tarkim kad kliento viešasis raktas yra $h = g^x mod\, p$. Kadangi pardavėjas žino kliento viešą raktą, tai

$$g^y = a \cdot h^c mod\, p$$

Tokiu būdų pardavėjas gavęs iš kliento a, siunčia jam iššūkį c ir gauna atsakymą y. Pardavėjas žinodamas p, q, h, g gali patikrinti ar

$$g^y = a \cdot h^c mod\, p$$

2.5 Schnorr parašo protokolas

Parašo protokolas yra panašus į identifikacijos protokolą. Čia pardavėjas nesiunčia iššūkio c . Pirkėjas skaičiuoja iššūkį naudodamas maišos funkciją. Tarkim klientas nori nusiųsti žinutę M pardavėjui. Pirma jis pasirenka atsitiktinį skaičių w ir suskaičiuoja $a = g^w mod\, p$, kaip identifikacijos protokole. Tuomet žinutė M ir a yra duodami maišos funkcijai kaip argumentas

$$c = H\,(M, a)$$

Klientas tuomet suskaičiuoja $y = w + x \cdot c\, mod\, q$ ir siunčią parašą (c, y), a ir žinutę M pardavėjui, kuris tikrina ar

$$g^y = a \cdot h^c mod\, p$$

tuo pačiu pardavėjas suskaičiuoja maišos rezultatą $H\,(M, a)$ ir patikrina, ar

$$c = H\,(M, a)$$

Jeigu gaunamos lygybės - parašas teisingas.

Kadangi abi šalys naudoja tą pačią H funkciją, reikia kas kart keisti atsitiktinį skaičių w, nes kitu atveju būtų įmanoma atskleisti x. Pavyzdžiui taip

$$x = \frac{(y1 - y2)}{(c1 - c2)}$$

2.6 Webcoin: elektroninių mikromokėjimų schema

Šiame skyriuje aprašoma mikromokėjimų sistema vadinama WebCoin. Ji yra efektyvi, saugi, lengvai įgyvendinama, paremta kreditu - debetu.

Sistemoje dalyvauja trys dalyviai: pirkėjas, pardavėjas ir bankas. Banką gali reprezentuoti tam tik-

ras serveris. WebCoin siūlo atjungties režimu veikiančią, maišos funkcijomis paremtą sistemą. Prekybos metu dalyvauja tik dvi šalys - pirkėjas ir pardavėjas. Bankas yra atsakingas už pirkėjų ir pardavėjų registravimą, dvigubo išleidimo nustatymą, pardavėjo neteisėtą kainos didinimą ir pirkėjų, pardavėjų sąskaitų valdymą.

WebCoin sistemoje vartotojas turi atsidaryti banko sąskaitą, o bankas turi jam išduoti pradinę monetą, kurios pagalba vėliau bus atsiskaitoma su pardavėjais. Vėliau pardavėjas perduoda pirkėjų išleistas monetas bankui. Šiame modelyje yra priimama, kad bankas yra sąžiningas ir patikimas pirkėjams ir pardavėjams. Pardavėjai yra mažiau patikimi, o vartotojai gali būti patikimi arba ne.

2.6.1 Protokolas

Siekiant įvesti aiškumo, protokolo apraše bus naudojami šie simboliai:

$$B - bankas$$

$$V - pardavėjas$$

$$U - pirkėjas$$

Atitinkamai jų vieši raktai žymimi PK_B, PK_U, PK_V, o privatūs raktai SK_B, SK_U, SK_V. Maišos funkcijos grandinės moneta $C_i\,(i = 1, 2, \ldots, n)$. Pasirašyta žinutė M žymima $\{M\}\,SK$. Norint, kad monetos tiktų skirtingiems pardavėjams, iš anksto turi būti nustatyta monetos vertė, pavyzdžiui 1 centas. Šią vertę galima pakeisti priklausomai nuo aplinkybių. Taip pat šioje schemoje naudojama dviguba maišos funkcijos moneta. Naudojamos dvi maišos funkcijos H_1 ir H_2, o moneta gali būti vieno vieneto C_i^1 arba dešimties vienetų C_j^{10} atitinkamai. Monetos indeksas nurodys išleistas monetas - šie indeksai bus registruojami ir pas pirkėjus ir pas pardavėjus, o vėliau ir debeto procedūroje.

WebCoin schemą sudaro keturi protokolai: (a) registravimas, (b) inicijavimas, (c) sandoris, (d) atskaitymas. Kai kurie protokolai turės po keletą žingsnių.

Registravimas Šiame etape yra užmezgami ryšiai tarp banko, pirkėjo ir pardavėjo. Bankas atsako už registravimo etapą ir kitų sistemos dalyvių banko sąskaitų valdymą. Pradžioje pirkėjas ir pardavėjas užsiregistruoja banke ir gauną joje paskyrą. Pardavėjas duoda leidimą bankui išduoti prekybai skirtas monetas. Bankas sukuria paskyras pirkėjui ir pardavėjui, o pardavėjas turi įnešti pradinį įnašą į savo sąskaitą. Ši informacija gali būti perduodama tiesiogiai kontaktuojant arba per kokį nors saugų kanalą.

Inicijavimas Šiame etape yra paruošiamas pirmas pirkėjo pirkimas. Bankas nustato išleidimo limitą ir sukuria du monetų modelius C_0^1 ir C_0^{10}. Abi monetos vėliau su pirkėjo pagalba galės sukurti daugiau monetų. Šie modeliai taip pat bus naudojami monetų patikimumui nustatyti. Tuo pačiu bankas išduoda ir pasirašo sertifikatą pirkėjui, kuriame yra abu monetų modeliai, išleidimo limitas, vartotojo viešasis raktas, sertifikato galiojimo laikas. Bankas gali nesutikti išduoti monetos, jeigu pirkėjo sąskaitoje per mažai pinigų, arba jeigu prieš tai vartotojas mėgino du kartus išleisti monetą.

Kai vartotojas pradeda pirkimą, jis turi pasirašyti pirkimo paraišką ir siųsti ją pardavėjui. Žinutėje turi būti banko pasirašytas sertifikatas, kad pardavėjas žinotų jog moneta teisėta.

- **Sertifikato išdavimas**

  Vartotojas banko prašo išduoti sertifikatą. Sertifikatas leidžia pirkėjui susikurti monetų, kurios vėliau bus pervedamos iš jo sąskaitos į pardavėjo sąskaitą.

- **Išleidimo limito nustatymas**

  Atsakydamas į pirkėjo prašymą, bankas patikrina pirkėjo sąskaitą. Jeigu pinigų nėra, pirkėjui pasiūloma įsigyti monetų. Priklausomai nuo pirkėjo banko sąskaitos, banką parenka jam išleidimo limitą. Šis dydis gali būti lygus banko sąskaitoje esančiam pinigų kiekiui arba gali būti mažesnis.

- **Monetų modelių sukūrimas**

  Bankas kuria du monetų modelius - šakninės monetos vartotojui, kuris galės jas naudoti kuriant daugiau monetų. Serveris atsitiktinai parenka du didelius skaičius x ir y, tuomet pritaiko jiems maišos funkciją.

$$C_0^1 = H\left(x\right)$$

$$C_0^{10} = H(y)$$

- **Sertifikato išdavimas**

  Bankas sukuria elektroniniu būdu pasirašytą sertifikatą, pasilieka jo kopiją sau ir perduoda originalą pirkėjui. Sertifikate yra banko identifikacija B_{ID}, pirkėjo identifikacija U_{ID}, pirkėjo viešasis raktas PK_U, galiojimo data E, ir pinigas C, kurį sudaro C_0^1, C_0^{10}, Max.

$$U_{sertifikatas} = \{B_{ID}, U_{ID}, PK_U, E, C\}\, SK_B$$

  Šis sertifikatas turi būti kaskart atnaujinamas, kai kontaktuoja bankas ir pirkėjas. Bankas išduos naują sertifikatą tik tuo atveju, jeigu pirkėjo sąskaita ne tuščia.

- **Sertifikato tikrinimas**

  Pirkėjas, norėdamas įsitikinti, kad sertifikatas yra banko pasirašytas, gali tai padaryti naudodamas

banko viešąjį raktą. Po to pirkėjas iš sertifikato pasiima monetos modelį, o sertifikatą pasilieka vėlesniam etapui.

- **Pirkimo paraišką siuntimas**

  Prieš pervedant monetas, pirkėjas turi pardavėjui nusiųsti pirkimo paraišką.

$$PC = \{U_{ID}, V_{ID}, U_{sertifikatas}, D\}\, SK_U$$

  Pirkimo paraiška yra pasirašoma vartotojo. Jeigu vartotojas norės pirkti kitoje parduotuvėje, jis taip pat turės siųsti pirkimo paraišką.

Sandoris Šiame etape vartotojas kuria monetas generuodamas ir kombinuodamas maišos funkcijų grandines priklausomai nuo prekės kainos. Monetų grandinės yra paremtos šaknine moneta C_0^1 ir C_0^{10} arba išskaičiuotos iš pasikeitusių šakninių monetų - paskutinio pirkimo paskutinių monetų. Pirkėjas gali pirkti prekes iš skirtingų pardavėjų, naudodamas tas pačia grandines. Galimybė tais pačiais e-pinigais atsiskaityti skirtingiems pardavėjams yra labai patogu. Pirkėjui tereikia nusiųsti abiejų grandinių paskutinių monetų reikšmes kartu su jų pradžios ir pabaigos indeksais. Pardavėjas galės nustatyti monetų teisėtumą nuo pradžių taikydamas maišos funkciją.

Tarkime, kad pirkėjas nori pirkti tam tikrą prekę iš kelių pardavėjų $V_1, V_2, \ldots, V_n$, sandorio procedūra turi būti tokia

- **Pirkimo paraiškos tikrinimas**

  Pardavėjas turi patikrinti pirkimo paraišką, kurią atsiuntė pirkėjas. Jis taip pat turi patikrinti vartotojo sertifikatą, kurį pasirašė bankas. Jeigu abu yra teisėti, pardavėjas tikrina ar pirkimo data nėra vėlesnė už sertifikato galiojimo datą. Jeigu taip - pardavėjas patalpina pirkimo paraišką pas save sistemoje.

- **Monetų sukūrimas**

  Pirkėjas skaičiuoja maišos funkcijas su argumentais C_0^1 ir C_0^{10}iš savo sertifikato. Monetų jis gali pasigaminti kiek nori. Pavyzdžiui jis pirmą kartą apsiperka per sertifikato galiojimo laiką, o prekių vertė yra n centų. Tuomet jis nustato kiek kokių centų jam reikia

$$i = n \bmod 10$$

$$j = floor(n/10)$$

  Monetų grandinės

$$C_1^1 = H\left(C_0^1\right), C_2^1 = H\left(C_1^1\right), \ldots, C_i^1 = H\left(C_{i-1}^1\right)$$

$$C_1^{10} = H\left(C_0^{10}\right), C_2^{10} = H\left(C_1^{10}\right), \ldots, C_j^{10} = H\left(C_{j-1}^{10}\right)$$

Jeigu tai nėra pirmasis dienos pirkimas, reiškia pirkėjas anksčiau jau sukūrė kažkiek monetų. Tarkim, kad prieš tai paskutinė išleista vieno cento moneta yra C_g, o paskutinė išleista dešimties centų moneta yra C_k, tuomet monetų grandinės bus tokios

$$C_{g+1}^1 = H\left(C_g^1\right), C_{g+2}^1 = H\left(C_{g+1}^1\right), \ldots, C_{g+i}^1 = H\left(C_{g+i-1}^1\right)$$

$$C_{k+1}^{10} = H\left(C_k^{10}\right), C_{k+2}^{10} = H\left(C_{k+1}^{10}\right), \ldots, C_{k+j}^{10} = H\left(C_{k+j-1}^{10}\right)$$

$$g = \overline{0, Max - 1}$$

$$k = \overline{0, Max/10 - 1}$$

Tuomet C_{g+i}^1 ir C_{k+j}^{10} yra paskutinės išleistos monetos. Prieš sukuriant monetas, kiekvienos grandinės ilgis yra nustatomas priklausomai nuo Max ir nuo paskutinių išleistų monetų indeksų.

- **Pirkimo žinutės siuntimas**

  Kai pirkėjas nutaria pirkti prekes iš pirkėjo, jis savo slaptu raktu pasirašo pirkimo žinutę ir siunčią ją pardavėjui. Pirkimo žinutė sudaryta iš sekančių dalių

$$PM = \left\{g, k, C_{g+i}^1, C_{k+j}^{10}, i, j, T\right\} SK_U$$

  g ir k yra šakninių monetų indeksai vieno ir dešimties centų monetoms atitinkamai. Tuomet $(g + i)$ ir $(g + k)$ yra paskutinių monetų indeksai. C_{g+i}^1 ir C_{k+j}^{10} taps šakninėmis monetomis sekančiame pirkime. T nurodo laiko momentą, kai buvo atsiųsta pirkimo žinutė. Tai apsaugo nuo pakartotinio tos pačios žinutės siuntimo. Pardavėjas nesunkiai gali nustatyti prieš tai išleista monetų kiekį

$$PSA = g + k \cdot 10$$

  Todėl paskutinės išleistos monetos C_{g+i}^1 ir C_{k+j}^{10} ir atitinkami indeksai $g+i, k+j$ bus išsaugoti pirkėjo kaip šakninių monetų reikšmės. Jos taip pat yra išsaugomos pardavėjo duomenų bazėje, jeigu parduotuvėje apsilankytų tas pats pirkėjas. Pažymėtina, kad pirkėjas monetas išleidžia didėjimo tvarka nepraleidžiant nė vieno žingsnio.

- **Pirkimo žinutės tikrinimas**

  Pardavėjas patikrina pirkimo žinutę su pirkėjo viešu raktu, kurį galima rasti prieš tai siųstos pirkimo paraiškoje. Be to jis iš pirkimo paraiškos gali sužinoti galiojimo datą $E, Max, U_{ID}, C_0^1, C_0^{10}$. Vėliau atliekami keli mažesni žingsniai:

– Pardavėjas suskaičiuoja prieš tai išleistą sumą
– Pardavėjas palygina Max su PSA ir reikalinga pirkimui suma $SDA = i + j \cdot 10$. Jeigu Max yra didesnis už PSA ir SDA sumą, tuomet i ir j yra teisėti. Kitu atveju pirkėjas pereikvoja savo limitą - pardavėjas neišduoda prekės.
– Pardavėjas pradeda tikrinti monetas taikydamas joms maišos funkciją. Pagal pirkėjo ID U_{ID} pardavėjas tikrina ar pas jį lankomasi pirmą kartą per sertifikato galiojimo laiką.

* Jeigu taip:
pardavėjas paima monetas C_0^1 ir C_0^{10}iš sertifikato, ir taiko maišos funkciją joms tol, kol gaunamos monetos $\hat{C}_{g+i}^1$ ir $\hat{C}_{k+j}^{10}$. Tai panašus procesas, kaip pirkėjo monetų kūrimas.

$$\hat{C}_{g+i}^1 = H\left(H\left(...H\left(C_0^1\right)\right)\right), g+i$$

$$\hat{C}_{k+j}^{10} = H\left(H\left(...H\left(C_0^{10}\right)\right)\right), k+j$$

* Jeigu ne:
Pardavėjas savo duomenų bazėje ieško įrašo prie U_{ID}, iš jo paima paskutines išleistas monetas C_p^1 ir C_q^{10} ir taiko joms maišos funkciją kol gauna $\hat{C}_{g+i}^1$ ir $\hat{C}_{k+j}^{10}$

$$\hat{C}_{g+i}^1 = H\left(H\left(...H\left(C_p^1\right)\right)\right), g+i-p$$

$$\hat{C}_{k+j}^{10} = H\left(H\left(...H\left(C_q^{10}\right)\right)\right), k+j-q$$

– Pardavėjas palygina maišos funkcijų rezultatus $\hat{C}_{g+i}^1$ ir $\hat{C}_{k+j}^{10}$ su C_{g+i}^1 ir C_{k+j}^{10}. Jeigu jie sutampa - monetos tikros.

- **Prekės išdavimas ir monetų išsaugojimas**
Po sėkmingo monetų patikrinimo pardavėjas perduoda pirkėjui prekę. Po to jis patalpina paskutinę išleistą ir atitinkamus indeksus $g+i$, $k+j$ į duomenų bazę šalia U_{ID}.

Atskaitymas Šiame etape pirkėjų ir pardavėjų sąskaitos yra atnaujinamos. Pardavėjas kontaktuoja su banku ir surenka visas pirkimo žinutes, surūšiuoja ir sugrupuoja jas pagal U_{ID} ir nusiunčia bankui. Kiekvieno pirkėjo atveju pardavėjas turi siųsti bankui paskutinių monetų reikšmes ir atitinkamus indeksus kartu su U_{ID}. Jis taip pat turi pareikalauti atskaitymo prie pasibaigiant sertifikatų galiojimo laikui. Bankas patikrina atskaitymo žinutes, pirkėjų sertifikatus ir sutikrina išleidimo limitus. Naudodamas monetų modelius C_{g+i}^1 ir C_{k+j}^{10}, bankas patvirtina paskutinės monetos reikšmę taikydamas maišos funkciją panašiai kaip pardavėjas tikrina monetas. Kiekvieno tikrinimo atveju jeigu viskas tvarkoje, bankas per-

veda pinigus iš atitinkamo vartotojo sąskaitos į pardavėjo sąskaitą. Šį protokolą galima suskirstyti į kelis etapus:

- **Pirkėjo debetavimas**

  Kas kart pirkėjui baigus apsipirkimus, jis turėtų nusiųsti paskutines išleistas monetas ir C^1_{g+i} ir C^{10}_{k+j} ir atitinkamus indeksus $g+i, k+j$ bankui. Tuomet bankas suskaičiuoja bendras pirkėjo išlaidas

  $$T_1 = (g+i) + 10 \cdot (k+j)$$

  ir atnaujina pirkėjo banko sąskaitą atitinkamai.

- **Atskaitymo žinučių tikrinimas**

  Dienos (ar valandos) pabaigoje pardavėjas siunčia atskaitymo žinutes RM bankui. Jos yra pasirašomos pardavėjo slaptu raktu SK_V. RM sudaro pardavėjo identifikacija V_{ID} ir pirkėjo mokėjimo informacija. Kiekvieno pirkėjo atveju pirkimo paraiškos, pirkimo žinutės ir visų pirkimų suma SUM yra patalpinamos atskaitymo žinutėje.

  $$RM = \left\{ V_{ID}, \sum \left(U_{ID}, PC, \sum (PM), \sum (SUM) \right) \right\} SK_V$$

  Bankas patikrina pardavėjo parašą su jo viešu raktu PK_V ir patikrina sertifikatų galiojimą kiekvienam PC. Bankas lengvai atskiria sertifikatus, nes pats juos pasirašė. Po to bankas tikrina paskutines monetas kiekviename PM taikydamas maišos funkciją, kaip minėta anksčiau. Jeigu monetos teisėtos, skaičiuojama bendra išleidimo suma $\sum (SDA)$ visų pirkėjų. Ši suma palyginama su pardavėjo atsiųsta $\sum (SUM)$. Jeigu sutampa, bankas papildo pardavėjo sąskaitą $\sum (SUM)$ suma, kitu atveju pervedama suma $\sum (SDA)$. Taip yra dėl to, kad vartotojas pasirašo $\sum (SDA)$ o ne $\sum (SUM)$.

- **Dvigubo išleidimo patikrinimas**

  Bankas surenka visus RM, sugrupuoja pirkimo žinutes pagal pirkėją ir susumuoja visus SDA iš skirtingų RM su tuo pačiu vartotojo ID ir tuomet skaičiuoja

  $$T_2 = \sum \left(\sum (SDA) \right)$$

  T_2 yra lyginamas su T_1. Jeigu T_1 yra didesnis už T_2 tai yra priimtina, nes kai kurie pardavėjai gali kompensuoti monetos praradimą arba grąžinti monetą po jos galiojimo laiko pabaigos. Kitu atveju nustatoma, kad vartotojas du kart išleido pinigą arba jį padirbo.

- **Sąskaitų atnaujinimas**

Bankas patvirtina visas monetas gautas iš pardavėjo. Jeigu monetos teisėtos, bankas papildo jų sąskaitas ir paima atitinkamas sumas iš pirkėjų sąskaitų.

2.7 Pilnai hierarchinis įgaliojantysis aklasis parašas

Realiame pasaulyje bankų sistema susideda iš daugybės lygių, padalinių, skyrių. Bankų yra tiesiog per daug. Net esant nacionalinio centrinio banko priežiūrai, kai kurie bankai gali būti tokie maži, kad kai kuriems klientams jei gali būti visiškai nežinomi. Jeigu klientas norės apsipirkti parduotuvėje su mažo banko išduotais elektroniniais pinigai, gali taip nutikti, kad pardavėjas nepatikės tų pinigų kilme. Todėl esant atjungties režimui e-pinigų sistema būtų nepatogi.

Remiantis įgaliojančiu parašu, aklu parašu, diskretaus logaritmo problema ir bitiesiniu poravimu buvo sukurtas pilnai hierarchinis įgaliojantis aklasis parašas. Tokiu atveju bankas gali pasirašyti kliento monetas, o klientui nereiks atskleisti savo tapatybės atsiskaitant su pardavėju.

Šioje schemoje dalyvauja įgalioti $l+1$ lygio pasirašytojai $\{B_1, B_2, \ldots, B_l, B_0\}$. Pažymėsiu, kad tikrieji pasirašytojai $\{B_2, \ldots, B_l, B_0\}$ yra taip pat ir atitinkamai 1 lygio, 2 lygio, ..., l-1-ojo lygio įgalioti pasirašytojai. Tiesą sakant įgaliotas pasirašytojas B_0 yra l lygio įgaliotas pasirašytojas. Taip pat i lygio tikrasis pasirašytojas $B_i\,(i = 2, \ldots, l)$ gali perduoti įgaliojimus $B_{i+1}\,(i = 2, \ldots, l)$ tikrajam pasirašytojui. Schemą sudaro keturi algoritmai (HG, HD, HS, HV).

PP sudėtingumo klasės algoritmas HG išduoda pasirašytojo B_i viešo/slapto raktų poras $(pk_i, sk_i)\,(i = 1, 2$ ir įgalioto pasirašytojo B_0 viešą/slaptą raktus (sk, pk).

HD yra eilė įgaliojimo algoritmų $HD_i\,(i = 1, 2, \ldots, l)$. Kai vykdomas HD_i algoritmas, i-tojo lygio pasirašytojas B_i perduoda savo pasirašymo teises i-tojo lygio įgaliotam pasirašytojui B_{i+1}. Toliau seka algoritmo HD_{i+1} vykdymas. Taip vykdoma, kol l lygio tikrasis pasirašytojas perduoda teises įgaliotam pasirašytojui B_0. Kiekvienas HD_i algoritmo vykdymas reikalauja raktų porų (pk_i, sk_i) ir (pk_{i+1}, sk_{i+1}) ir delegavimų $m_{\omega 1}, \sigma_1, m_{\omega 1}, \sigma_1, \ldots, m_{\omega,i-1}, \sigma_{i-1}$ - visų aukštesnių lygių. Čia $m_{\omega i}$ yra garantas, kurį sudaro i - tojo ir i+1 lygio tikrųjų pasirašytojų tapatybės, delegavimo laikas ir kitos savybės.

PP sudėtingumo klasės pasirašymo algoritmas HS reikalauja visų delegavimų $m_{\omega 1}, \sigma_1, m_{\omega 1}, \sigma_1, \ldots, m_{\omega l},$ žinutės m ir sukurią aklą parašą (m, σ).

NP sudėtingumo klasės parašo tikrinimo algoritmas reikalauja (m, σ) ir sukuria rezultatą 0 arba 1.

Toliau bus suformuotas pilnai hierarchinis įgaliojantis aklasis parašas, kuriame dalyvauja du bankų lygiai.

1. Pagrindinė organizacija ar pasaulinis bankas inicijuoja HG algoritmą ir sukuria sisteminius parametrus p, q ir viešo/slapto raktų poras pasirašytojams. Čia p yra didelis pirminis skaičius, o q yra didelis pirminis skaičiaus $p-1$ daliklis, g yra q eilės grupės Z_q^* generatorius. Įgaliotas pasirašytojas

B_0 gauna savo viešo/privataus raktų porą

$$y = g^x mod\, p$$

$$x \in Z_q^*$$

Tikrieji pasirašytojai $B_i\ (i = 1, 2)$ gauna atitinkamai

$$y_i = g^{x_i} mod\, p$$

$$x_i \in Z_q^*$$

Sistemoje taip pat turi būti aprašytos trys viešos kriptografinės maišos funkcijos $H_1(), H_2(), H_3()$.

2. Šiame žingsnyje bus įvykdyti HD_1 ir HD_2 delegavimo algoritmai.

   (a) Delegavimo algoritmas HD_1 veikia taip

      i. B_i parenka atsitiktinį sveikąjį skaičių $k_i \in Z_q^*$ ir suskaičiuoja $r_i = g^{k_i} mod\, p$.
      ii. B_1 ir B_2 bendradarbiaudami suskaičiuoja $r_{12} = r_1 \cdot r_2$
      iii. B_1 sugeneruoja ar kitaip parenka garantą $m_{\omega 1}$, apskaičiuoja s_1 ir siunčia jį B_2

$$s_1 = k_1 r_{12} + x_1 H_1\left(m_{\omega 1} \| r_{12}\right) mod\, q$$

      iv. B_2 skaičiuoja

$$s_2 = k_2 r_{12} + x_2 H_1\left(m_{\omega 1} \| r_{12}\right) mod\, q$$

      ir gauna įgaliojantį raktą (x_{12}, y_{12})

$$x_{12} = s_1 + s_2 mod\, q$$

$$y_{12} = g^{x_{12}} = r_{12}^{r_{12}} \left(y_1 y_2\right)^{H_1(m_{\omega 1} \| r_{12})} mod\, p$$

   (b) Delegavimo algoritmas HD_2 vykdomas taip

      i. B_2 ir B_0 susigeneruoja atitinkamai r_3 ir $r_4$tokiu pačiu būdu kaip l lygio delegavime.
      ii. B_2 ir B_0 skaičiuoja $r_{34} = r_3 r_4$
      iii. B_2 siunčia aukštesnio lygio įgaliojimą $(y_1, y_2, r_{12}, m_{\omega 1})\ B_0$
      iv. B_2 sugeneruoja ar kitaip parenka garantą $m_{\omega 2}$, apskaičiuoja s_3 ir siunčia juos B_0

$$t = H_2\left(m_{\omega 1} \parallel m_{\omega 2} \parallel r_{12} \parallel r_{34}\right)$$

$$s_3 = k_3 r_{34} + x_{12} t \, mod \, q$$

v. B_0 skaičiuoja

$$s_4 = k_4 r_{34} + xt \, mod \, q$$

ir gauna įgaliojimo raktų porą (x_p, y_p)

$$x_p = s_3 + s_4 \, mod \, q$$

$$y_p = g^{x_p} = r_{34}^{r_{34}} h^t mod \, p$$

kur $h = y r_{12}^{r_{12}} \left(y_1 y_2\right)^{H_1(m_{\omega 1} \| r_{12})}$

3. Aklojo parašo algoritmas HS vykdomas taip.

   (a) B_0 parenka atsitiktinį sveikąjį skaičių $k \in Z_q^*$ir apskaičiuoja $\overline{r} = g^k mod \, p$. B_0 siunčia įsipareigojimą $\overline{r}$ ir viešą informaciją $(y_1, y_2, y, r_{12}, r_{34}, m_{\omega 1}, m_{\omega 2})$ klientui C.

   (b) C laisvai pasirenka du atsitiktinius skaičius $\alpha, \beta \in Z_q^*$ir apskaičiuoja

$$r = \overline{r} g^{\alpha} y_p^{\beta} mod \, p$$

$$\overline{c} = H_3 \left(m \parallel m_{\omega 1} \parallel m_{\omega 2} \parallel r_{12} \parallel r_{34} \parallel r\right) + \beta$$

   (c) B_0 skaičiuoja

$$\overline{s} = k + x_p \overline{c} \, mod \, q$$

   (d) C skaičiuoja

$$s = \overline{s} + \alpha$$

$$c = \beta - \overline{c}$$

4. Aklojo parašo algoritmas HS pateikia hierarchinį įgaliojantį akląjį parašą $(r_{12}, r_{34}, m_{\omega 1}, m_{\omega 2}, r, m, s, c)$ žinutei m. Parašo tikrumą galima nustatyti skaičiuojant

$$r = g^s y_p^c mod \, p$$

jeigu ši lygybė galioja, tuomet tikrinimo algoritmas turėtų gražinti 1.

Teorema 1. Pilnai hierarchinio įgaliojančio aklojo parašo schema yra saugi.

Įrodymas. Pirma parodysime, kad schema yra **nesuklastojama** remiantis atsitiktinio orakulo modeliu ir diskretaus logaritmo prielaida.

Hierarchiniai įgaliojimai, tai yra pirmo lygio tarp B_1 ir B_2, ir antro lygio tarp B_2 ir B_0 yra nesuklastojami. Įgaliojimo perdavimo raktai $s_i\,(i = 1, 2, 3)$ yra Schnorr schemos parašai.

Be to pasirašytojas B_0 negali padirbti įgaliojančio parašo be C pagalbos. Tarkim, kad tikrasis pasirašytojas gali padirbti įgaliojantį parašą $(r_{12}, r_{34}, m_{\omega 1}m_{\omega 2}, r, m, s', c')$. Tuomet parašas gali pereiti tikrinimo lygtį

$$r = g^{s'} y_p^{c'} mod\, p$$

tuomet gauname

$$g^s y_p^c = r = g^{s'} y_p^{c'} mod\, p$$

$$g^{s-s'} = g^{x_p(c'-c)} mod\, p$$

taigi įgaliojimo slaptasis raktas gali būti apskaičiuojamas taip

$$x_p = \frac{s - s'}{c' - c} mod\, q$$

Ši lygybė parodo, kad y_p diskretus logaritmas x_p gali būti apskaičiuotas, kas prieštarauja diskretaus logaritmo prielaidai.

Akivaizdu, kad trečia šalis negali padirbti galiojančio hierarchinio įgaliojančio aklojo parašo, nes turi mažiau informacijos negu B_0.

Toliau parodysime, kad parašo schema tenkinta **neatsekamumo** savybę.

Tarkime, kad tikrasis pasirašytojas B_0 mėgina susieti save su akluoju parašu žinutei m kaip kenkėjas. B_0 atpažįstamas, kaip $\{k, \overline{r}, \overline{s}, \overline{c}\}$. Esant aklajam parašui $(r_{12}, r_{34}, m_{\omega 1}, m_{\omega 2}, r', m, s', c')$ B_0 bandys nustatyti ar tai galima sieti su jo parametrais $\{k, \overline{r}, \overline{s}, \overline{c}\}$.

Žinome, kad galioja šios lygtys kažkokiems $\alpha', \beta' \in Z_q^*$

$$\overline{r} = g^k mod\, p$$

$$r' = \overline{r} g^{\alpha'} y_p^{\beta'} mod\, p$$

$$\overline{s} = k + x_p \overline{c}\, mod\, q$$

$$s' = \overline{s} + \alpha'$$

$$c' = \beta' - \overline{c}$$

Todėl galime suskaičiuoti

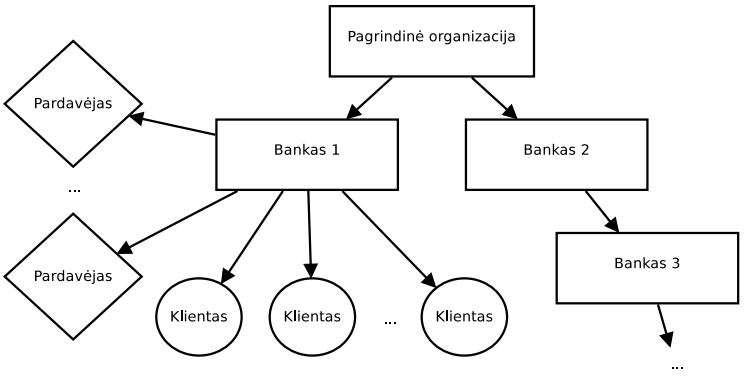
$$r' = g^{k+\alpha'} y_p^{\beta'} = g^{k+s-\overline{s}} y_p^{\beta'} = g^s y_p^{\beta'-\overline{c}} = g^s y_p^c mod\, p$$

Taigi bet kieno parametrai gali atspindėti bet kurį akląjį parašą. Trumpai tariant tikrasis pasirašytojas B_0 niekada savęs nesusieja su savo akluoju įgaliojančiuoju parašu žinutei m.

3 TIRIAMOJI DALIS

3.1 Hierarchinė bankų schema

Šiame skyriuje kaip mano siūlomame modelyje turi būti išdėliota bankų hierarchinė sistema. Kalbant apie bankus turėsiu omenyje ne tik bankus, bet ir tam tikras elektroninių pinigų organizacijas, kredito bendroves ir kitas patikimas šalis. Bet kuriuo atveju hierarchinės sistemos viršūnėje turi būti tam tikra pagrindinė organizacija, kuri parenka sisteminius parametrus ir išduoda įgaliojimus dukterinėms organizacijoms ar bankams. Visos šalys turi būti patikimos, nes jos gali išduoti įgaliojimus pasirašyti pagrindinio banko vardu šiam nedalyvaujant ir nežinant.



3.1 pav.: Bankų hierarchinė schema

3.2 Modeliavimas ir realizavimas

WebCoin elektroninių pinigų schemoje siūlomas monetos modelis yra kuriamas banko ir patalpinamas į sertifikatą. Sertifikatas šiuo atveju pasirašomas banko. Tačiau tokiu atveju pirkėjas nėra anonimiškas prieš pardavėją. Nors skaidrumas didina pasitikėjimą, pirkėjas dažnai nenorės, kad pardavėjas žinotų jo tapatybę ar kažkokius asmeninius duomenis. Pirkėjo anonimiškumą prieš pardavėją geriausiai užtikrintų jo parašo nenaudojimas.

Mano modelio specifikacijoje dalyvauja trys šalys. Bankas, klientas, pardavėjas, kuriuos atitinkamai žymėsiu B, K, P.

Modelio realizaciją pateiksiu matematinio paketo Wolfram Alpha sintakse. Šis paketas turi visas būtinas bibliotekas dirbti su dideliais skaičiais. Dėka jau sukurtų patogių funkcijų, programos kodas yra glaustas ir aiškus. Pilnas programos kodas pateikiamas šio darbo priede.

3.2.1 Banko sertifikato perdavimas klientui

Klientas K kreipiasi į banką, norėdamas gauti sertifikatą. Kaip ir WebCoin schemoje prieš tai klientas turi atsidaryti sąskaitą banke ir įnešti pradinį įnašą. Bankas klientui pateikia tokius sisteminius

parametrus p, q, g.

Parametrai skaičiuojami pagal teoremą (Sakalauskas; Listopadskis; Dosinas; Lukšys; Katvickis: 2007).

Teorema. Jeigu $p = 2q + 1$ yra pirminis ir q – pirminis, tuomet $g \in Z_p^*$ yra generatorius tada ir tik tada kai $g^q \neq 1 \, mod \, p$ ir $g^2 \neq 1 \, mod \, p$.

```
(*  Kuriami sisteminiai parametrai *)
For[i = RandomPrime[5000], PrimeQ[2*i + 1] == False, i = RandomPrime[5000]];
q = i
p = 2*i + 1
For[

g = RandomInteger[{1, p}];,
(PowerMod[g, q, p] != 1 && PowerMod[g, 2, p] != 1) == False,
g = RandomInteger[{1, p}];

];
(* Sukuriamos raktu poros

x[[1]] - pagr organizacijos
x[[2]] - banko
x[[3]] - kliento

*)
x = Table[RandomInteger[{1, q}] , {i, 1, 3}]
y = Table[PowerMod[g, x[[i]], p], {i, 1, 3}]
k = Table[RandomInteger[{1, q}], {i, 1, 4}];
r = Table[PowerMod[g, k[[i]], p], {i, 1, 4}];
mwO = "Pagr organizacija";
(* HD1 - pagr organizacijoa deleguoja bankui *)
rOB =   r[[1]]*r[[2]] ;(* skaiciuoja abu *)
h1 =   Mod[Hash[{mwO, Mod[rOB, q]}], q];
s1 = Mod[k[[1]]*rOB + x[[1]]*h1,q]; (* Skaiciuoja pagr organizacija, siuncia bankui *) s2 = Mod[
(* delegavimo raktas x,y, kuri skaiciuojasi bankas *)
xOB = Mod[s1 + s2, q]
yOB = PowerMod[g, xOB, p]
```

Paprastumo dėlei tarsiu, kad šis bankas įgaliojimus gavo tiesiai iš pagrindinės organizacijos. Tokią prielaidą taikau dėl paprastesnių skaičiavimo išraiškų. Klientas susikuria savo slaptą ir viešą raktus, viešą raktą perduoda bankui. Bankas savu ruožtu patalpina jį prie kitos informacijos apie klientą.

$$y_K = g^{x_K} mod \, p$$

$$x_K \in Z_q^*$$

Tuomet bankas klientui perduoda savo aukštesnių lygių įgaliojimus, taip įrodydamas savo teisėtumą

$$(y_O, y_B, r_{OB}, m_{\omega 1})$$

Šiuo atveju y_O yra pagrindinės organizacijos, išdavusios įgaliojimu viešasis raktas, y_B - yra banko viešasis raktas, r_{OB} - pagrindinės organizacijos įgaliojimų perdavimo bankui raktas, o $m_{\omega O}$ - garantas. Garante turėtų būti patalpinama informacija apie įgaliojimą išdavusią organizaciją, įgaliojimo galiojimo laikas, kiti teisėtumui pagrįsti skirti atributai.

Bankas klientui sukuria sertifikatą. Šiame sertifikate turi būti patalpintos šakninės elektroninių pinigų monetos, pinigų išleidimo limitas, įgaliojimo galiojimo laikas.

$$cert_K = \{U_{ID}, B_{ID}, C, Max, E, y_K\}$$

$$C = \left\{C_0^1, C_0^{10}\right\}$$

$$C_0^1 = H\left(x\right)$$

$$C_0^{10} = H(y)$$

$$C_0^{100} = H(z)$$

Čia B_{ID} - banką identifikuojantys atributai, Max - išleidimo limitas, E - galiojimo terminas, y_K - kliento viešasis raktas. Įvedami trys monetų nominalai - 1ct,10ct,100ct=1lt. Monetas sukuria bankas atsitiktiniu būdu parinkdamas maišos funkcijoms argumentus. Taip pat bankas išduotas monetas išsaugo savo duomenų bazėje kartu su sertifikato kopija. Pažymėsiu, kad sertifikato galiojimo terminas turi būti pakankamai trumpas - kelios dienos ar savaitė. Pardavėjas P visada matys šį galiojimo laiką ir nepriims monetų iš kliento, kurio sertifikatas pasibaigęs.

Bankas suskaičiuoja parametrą r

$$r_1 = g^{k_B}$$

Čia k_B yra atsitiktiniu būdu parinktas skaičius mažesnis už q. Skaičiuojamas parašas ant sertifikato:

$$h_1 = H\left(cert_K \parallel m_{\omega O} \parallel r_{OB} \parallel r\right)$$

$$s_1 = k_B + h_1 \cdot x_B \, mod \, q$$

$$\sigma_1 = (s_1, h_1)$$

```
(* bankas siuncia klientui siuos: yO,yB,rOB,mwO
```

```
Bankas isduoda sertifikata ir pasiraso

*)
c1 = Hash[RandomInteger[5000]]; (*Vieno cento moneta*)
c10 = Hash[RandomInteger[5000]]; (*Dešimt centų moneta*)
c100 = Hash[RandomInteger[5000]]; (*Dešimt centų moneta*)
max = 1000;
c = {c1, c10, c100};
Kid = 101; (* kliento id *)
Bid = 202; (* banko id *)
Expr = 1305884062;(* Fri,20 May 2011 \ 09:34:22 GMT *)
certK = {Kid, Bid, c, max, Expr, y[[3]]};
k[[2]] = RandomInteger[{1, q}];
r[[2]] = PowerMod[g, k[[2]], p];
h = Hash[{certK, mwO, rOB, r[[2]]}]
s = Mod[k[[2]] + h*x[[2]], q];
```

Gautas sertifikatas, kartu su parašu σ.

3.2.2 Pirkimo paraiškos siuntimas

Klientas apsilankydamas pas pardavėją pirmiausia siunčia jam pirkimo paraišką. Tai yra tarsi santykių tarp pardavėjo ir kliento užmezgimas. Pirkimo paraiška turi atrodyti taip

$$PP = \{K_{ID}, P_{ID}, cert_K, data\}$$

Klientas skaičiuoja parašą, pagal Schnorr elektroninio parašo schemą ir siunčia ją pardavėjui.

$$r_2 = g^{k_K}$$

$$h_2 = H\left(PP, r_2\right)$$

$$s_2 = k_K + h_2 \cdot x_K \, mod \, q$$

$$\sigma_2 = (s_2, h_2)$$

```
(* Pirkimo paraiskos siuntimas *)
Pid = 303;(* pardavejo id *)
timePP = 1305804648;
PP = {Kid, Pid, certK, timePP}
```

Kartu su pirkimo paraišką PP, parašu σ, yra siunčiami vieši duomenys $(m_{\omega O}, r_{OB}, y_O, y_B)$Pardavėjas

gavęs paraišką turi įsitikinti jos teisėtumu. Sertifikato tikrumą tikrina tokiu būdu:

$$r_1' = g^{s_1} y_B^{h_1} mod\, p$$

$$h_1' = H\left(cert_K, r_1'\right)$$

Jeigu h_1' ir h_2 sutampa, vadinasi sertifikatas galioja. Analogišku būdu tikrinamas vartotojo parašas ant pirkimo žinutės PP. Jeigu abu parašai teisėti, sertifikato galiojimo data nepasibaigus, užmezgamas kontaktas ir klientas gali pirkti prekę.

3.2.3 Pirkimo žinutės siuntimas

Prieš aprašant pirkimo žinutės schemą, pažymėsiu, kad klientas savo elektroninių pinigų sąsajoje visada fiksuoja kokios yra jo šakninės monetos, kokios yra pradinės monetos, išduotos banko, ir kokios monetos yra išleistos paskutinės. Kartu su paskutinėmis išleistomis monetomis turi būti saugomi ir jų indeksai.

Perkant prekę iš pardavėjo, klientas skaičiuoja reikiamą sumą pirkiniui ir siunčia pasirašytą žinutę, kaip aprašyta 2.6.1 skyriuje.

$$PZ = \left\{g, k, l, C_{g+i}^1, C_{k+j}^{10}, C_{l+m}^1, i, j, m, T\right\}$$

$$r_3 = g^{k_K'}; h_3 = H\left(PZ, r_3\right); s_2 = k_K' + h_3 \cdot x_K mod\, q; \sigma_3 = (s_3, h_3)$$

Tipinė pirma pirkimo žinutė už 1lt 23ct galėtų atrodyti taip

$$PZ^* = \left\{0, 0, 0, C_0^1, C_0^{10}, C_0^{100}, 3, 2, 1, 1305763061\right\}$$

```
(* Pirkimo zinutes siuntimas *)
g1 = 0; k1 = 0; l1 = 0;
sakc1 = c1;
sakc10 = c10;
sakc100 = c100;
kaina1 = 1.23;
(* skaiciuojama, kiek kokiu monetu reikia *)
i1 = IntegerPart[Mod[kaina1*100, 10]];
j1 = (IntegerPart[Mod[kaina1*100, 100]] - i1)/10;
m1 = (IntegerPart[Mod[kaina1*100, 1000]] - i1 - 10*j1)/100;
timePZ1 = 1305804423;
mint[c_] := Hash[{c}];(* funkcija monetoms kurti *)
```

```
PZ1 = {g1, k1, l1,

Nest[mint, sakc1, g1 + i1],
Nest[mint, sakc10, k1 + j1],
Nest[mint, sakc100, l1 + m1],

i1, j1, m1, timePZ1}
```

O sekanti po šios už 1lt 11ct

$$PZ^{**} = \left\{3, 2, 1, H\left(H\left(H\left(H\left(C_0^1\right)\right)\right)\right), H\left(H\left(H\left(C_0^{10}\right)\right)\right), H\left(H\left(H\left(C_0^{100}\right)\right)\right), 1, 1, 1, 1305763073\right\}$$

Analogiškai, kaip 3.2.2 skyriuje, pardavėjas patikrina kliento žinutę.

3.2.4 Pirkimo žinutės tikrinimas

Tikrinimo protokolas vykdomas taip pat, kaip aprašyta 2.6.1 skyriuje. Pardavėjas tikrina kliento atsiųstą pirkimo žinutę PZ naudodamas jo viešąjį raktą. Viešas raktas yra kliento sertifikate.

Skaičiuojama kliento prieš tai išleista suma ir reikalinga pinigų suma pirkiniui.

$$IPS = g * 1 + k * 10 + l * 100$$

$$RPS = i * 1 + j * 10 + m * 100$$

Suma turi neviršyti sertifikate nurodytos išleidimo ribos

$$IPS + RPS < Max$$

Jeigu žinutė pasirašyta, parašas sutampa ir pirkėjui pinigų užtenka, galima tikrinti monetas. Pardavėjas paima pradines monetas ir joms iteraciniu būdu taiko maišos funkciją. Gautą rezultatą sulygina su pirkėjo atsiųstomis monetomis.

```
(* pirkimo zinutes tikrinimas *)
IPS = PZ1[[1]]*1 + PZ1[[2]]*10 + PZ1[[3]]*100; (* isleista pinigu suma *)
RPS = PZ1[[7]]*1 + PZ1[[8]]*10 + PZ1[[9]]*100; (* reikalinga pinigu suma *)
If[

RPS + IPS < certK[[4]],
"Pinigu pakanka",
"Pinigu nepakanka"

]
(* tikrinamos monetos *)
```

```
c1' = Nest[mint, certK[[3]][[1]], PZ1[[1]] + PZ1[[7]]]
c2' = Nest[mint, certK[[3]][[2]], PZ1[[2]] + PZ1[[8]]]
c3' = Nest[mint, certK[[3]][[3]], PZ1[[2]] + PZ1[[9]]]
If[PZ1[[4]] == c1', "1ct monetos teisetos", "1ct monetos neteisetos"]
If[PZ1[[5]] == c2', "10ct monetos teisetos", "10ct monetos neteisetos"]
If[PZ1[[6]] == c3', "1lt monetos teisetos", "1lt monetos neteisetos"]
```

3.2.5 Atskaitymo žinutės siuntimas

Pardavėjas yra suinteresuotas kuo dažniau kreiptis į banką ir gauti pinigus į savo sąskaitą. Kas kart kreipdamasis pardavėjas siunčia bankui atskaitymo žinutę, kur formuojama taip.

$$AZ = \left\{ P_{ID}, \sum \left(K_{ID}, PP, \sum (PZ), \sum (SUM) \right) \right\}$$

Atskaitymo žinutė siunčiama bankui pasirašant pardavėjo slaptu raktu pagal Schnorr algoritmo schemą.

V_{ID}	K_1	PP_{K_1}	$PZ, PZ_2, \ldots, PZ_m$	$SUM = \sum RPS$
	K_2	PP_{K_2}	$PZ_1, PZ_2, \ldots, PZ_n$	$SUM = \sum RPS$
	...			
	K_n	PP_{K_n}	$PZ_1, PZ_2, \ldots, PZ_o$	$SUM = \sum RPS$

3.1 lentelė: Atskaitymo žinutė

3.3 Modelio savybės

Šiame modelyje klientas sukuria monetas ir iškeičia jas į prekes. Kertiniai šio modelio akmenys yra maišos funkcijos ir elektroninio parašo schema. Reiškia kiekviena operacija turi būti patvirtinama elektroniniu parašu. Didelis modelio privalumas yra galimybė naudoti skirtingo nominalo monetas. WebCoin modelyje naudojamos dvi maišos grandinės, o aš modelį papildžiau dar viena grandine. Tokia modelio struktūra leidžia sumažinti skaičiavimų kiekį.

3.3.1 Efektyvumas

Norint pasiekti didesnį skaičiavimo efektyvumą galimos trys kryptys: mažinti sudėtingų skaičiavimų kiekį; kuo daugiau skaičiavimų atlikti atjungties režimu; mažinti kaupiamų duomenų kiekį visoms modelyje dalyvaujančioms šalims.

Mažai skaičiavimo operacijų Faktas, kad parašo schemos skaičiavimai reikalauja žymiai daugiau resursų, negu maišos funkcijų skaičiavimai. Todėl modelyje maišos funkcija taikoma kuo dažniau, o parašo schema kuo rečiau. Pirmą kartą klientui bendraujant su pardavėju yra vykdomi du elektroninio parašo skaičiavimo ciklai. Vienas skirtas kliento sertifikatui, o kitas pirkimo paraiškos parašui. Antrą kartą bendraujant tam pačiam klientui ir pardavėjui nebūtina tikrinti vartotojo sertifikato, nes pardavėjas jau žino klientą.

Naudojant tris maišos funkcijų grandines, žymiai sumažinama maišos funkcijų skaičiavimų kiekis. Ypač kai pirkinio suma yra didesnė. Dar labiau skaičiavimų kiekį sumažina monetų indeksų naudojimas. Kuriant monetą nereikia taikyti maišos funkcijos nuo pradžių, užtenka paimti paskutinę monetą ir skaičiuoti nuo jos. Pardavėjui tikrinti monetą taip pat paprasčiau, nes skaičiavimai prasideda nuo paskutinės monetos. Ta pati savybė galioja ir kuomet bankui reikia sutikrinti monetas.

Atjungties režimas Modelis veikia pilnu atjungties režimu. Klientas kontaktuoja su banku tik kai reikia gauti sertifikatą. Sandorio metu ryšys su banku nėra reikalingas.

Mažai saugojamų duomenų Dėl monetų indeksų naudojimo, visos schemos šalys sutaupo resursų, nes modelis gali būti įgyvendinamas ribotų resursų aplinkoje - mobiliame telefone. Klientas savo sistemoje saugoja tik sertifikatą, išlaidų sumą, paskutinių monetų reikšmes ir indeksus. Pardavėjui tereikia saugoti kiekvieną pirkimo žinutę ir klientų pirkimo paraiškas.

3.3.2 Saugumas

Pasiūlyta schema suteikia tiek saugumo, kiek suteikia maišos funkcijos ir elektroninio parašo schema. Vartotojo sertifikatas užtikrinamas banko parašu, o bankui įgaliojimus pasirašyti suteikia aukštesnio lygio bankas. Piramidės viršūnėje yra patikima pagrindinė organizacija. Tai leidžia pardavėjui pasitikėti klientu, nekontaktuojant su banku. Nors monetų kūrimo atsakomybė daugiausiai tenka klientui, bankas taip pat prisideda sukurdamas pradines monetas ir jas pasirašydamas.

Klastojimo prevencija Esminė modelio savybė, kad monetų indeksai yra lygiai taip pat nepadirbami, kaip ir pačios monetos. Vienkryptės, atsparios kolizijoms maišos funkcijos užtikrina monetų nepadirbamumą. Niekas kitas be vartotojo negali sukurti tokios pačios monetos, ar numatyti sekančios, net jeigu žinomos praėjusios monetos. Nuo monetų padirbimo apsaugoja vienkryptės, atsparios kolizijoms maišos funkcijos. Be to kuriant netikras monetas vis tiek reikės banko parašo.

Kredito viršijimo prevencija Sistema paremta debetu, vartotojas per tam tikrą laiką negali išleisti didesnės sumos, nei ją nustatė bankas. Šis apribojimas apsaugo nuo apgaulės, nes apsipirkti neišeina, jeigu pasiektas išlaidų limitas. Limitas yra pridedamas prie sertifikato ir pasirašomas banko. Galioja sertifikato galiojimo laiką, o vėliau yra nustatomas iš naujo.

Kiekvieno sandorio metu pardavėjas tikrina, ar pirkėjas nepasiekė šio limito ir išduoda ar neišduoda prekių.

Dvigubo išleidimo prevencija ir perkainojimo prevencija Sistemos ciklo pabaigoje bankas surenka pirkimo žinutes iš pardavėjų ir patikrina ar monetos teisėtos. O kadangi bankas saugoja sertifikatų kopijas, du kart panaudotas monetas gali atpažinti nesunkiai. Bankas imasi kitų prevencinių priemonių. Klientams yra nustatomos taisyklės. Jeigu klientas du kartus išleidžia pinigą, jo ID yra patalpinamas į juodąjį sąrašą, kuris yra pasiekiamas visiems pardavėjams, kai šie perduoda pirkimo žinutes. Be to bankas gali atsisakyti atnaujinti sertifikatą sukčiaujančiam klientui.

Jeigu pardavėjas du kartus perduoda tą pačią pirkimo žinutę, parašas yra pakartojamas, nes jo padirbti neįmanoma. O kadangi pardavėjas nėra anonimiškas prieš banką, jį demaskuoti labai lengva.

Neatsekamumas Ši savybė galioja tarp bankų, perduodančių įgaliojimus kitiems bankams. Šios savybės pagrįstumas yra įrodytas 2.7 skyriuje.

Diskretaus logaritmo uždavinys Modelis yra nesaugus, jeigu mokame suskaičiuoti diskretų logaritmą. Realaus pasaulio modelio realizacijoje turi būti naudojami pakankamai dideli skaičiai, kad būtų neįmanoma suskaičiuoti diskretaus logaritmo per priimtiną laiką. Lentelėje (3.2) pateikiami pavyzdžiai apie pasiektus rekordus, skaičiuojant diskretų logaritmą. Šie duomenys svarbūs pasirenkant, kokio dydžio skaičius naudoti realiame modelyje. Pažymėsiu, kad egzistuojant kvantiniam kompiuteriui, ši problema būtų lengvai sprendžiama per priimtiną laiką.

Data	Įranga	Laikas	Algoritmas	Skaičius
2005 09 22	$4 \cdot 16$ procesorių masyvų po $1.3Ghz$	17 dienų	Adleman lauko rėtis	$GF\left(2^{613}\right)$
2007 02 05	Himalaya klasteris	neskelbiama	klasikinis rėtis	$GF\left(2^{530}\right)$

3.2 lentelė: Diskretaus logaritmo skaičiavimo rekordai

Maišos funkcijos Modelis yra nesaugus, jeigu iš maišos funkcijos rezultato galime sužinoti maišos argumentą. Arba jeigu galime rasti kitą argumentą, kuris duotų tą patį maišos rezultatą. Lentelėje (3.3) pateikti keli maišos funkcijų pavyzdžiai kartu su operacijų skaičiumi, reikalingu įveikti algoritmą.

Algoritmas	Rezultatas (bitai)	Argumentas (simboliai)	Kolizijų ataka (operacijų sk.)	Žinutės paieškos ataka (operacijų sk.)
MD2	128	32	2^{63}	2^{73}
MD5	128	32	2^{21}	2^{123}
SHA-0	160	32	2^{33}	
SHA-1	160	32	2^{51}	

3.3 lentelė: Maišos funkcijos ir jų patikimumas

3.4 Modelio parametrai

Siūlomi modelio parametrai ne tik užtikrina sistemos saugumą, bet ir leidžia atlikti operacijas per priimtiną laiką. Modelio realizacija orientuojama į mobiliųjų telefonų saugius mikroprocesorius. Bandymų metu nustatytas 512 bitų skaičių operacijų skaičiavimo laikas pateikiamas lentelėje (3.4). Ne visų operacijų greitis yra aktualus. Greitos operacijos svarbios tik tarp kliento ir pardavėjo. Pirkimo paraiškos siuntimo operacijoje didžiąją laiko dalį užima parašo skaičiavimas, nes visi kiti duomenys jau suskaičiuoti. Parašo schemą sudaro šios operacijos:

- modulinė eksponentė (0.39 s.),
- maišos funkcija (0.029 s.),
- sandauga (3.1 s.),
- suma (0.49 s.).

Tuomet iš viso parašui formuoti reikia 4 sekundžių. Pirkimo žinutės siuntime dar reikia apskaičiuoti monetų grandines. Jeigu tarsime, kad didžiausia galima mokėjimo suma yra 100 litų, tuomet su trimis maišos grandinėmis ilgiausiai truktų apmokėti 99Lt 99ct. Ši suma reikalautų $99 + 9 + 9$ skaičiavimo operacijų. Tai truktų papildomai 3 sekundes.

Operacija	laikas, s.
Suma	0.49
Skirtumas	0.33
Postūmis į dešinę	0.79
Modulinė eksponentė	0.39
Sandauga	3.1
100 SHA1 operacijų	2.9

3.4 lentelė: Operacijų laikas su 512 bitų ilgio skaičiais mikroprocesorinėje kortelėje

IŠVADOS

- Pasiūlyta vidutinių mokėjimų e. pinigų sistema, kuri turi įrodomo pinigų dalumo ir ID paremto anonimiškumo sąvybes
- Sukurtas hierarchinis e. pinigų cirkuliacijos modelis, panaudojant įgaliojančiuosius parašus, pasižyminčius įrodomu neatsekamumu
- Parodyta, kad sistema pasižymi tokiomis sąvybėmis:
  - atjungties režimas
  - mažai skaičiavimo operacijų
  - mažai saugojamų duomenų
  - klastojimo, kredito viršijimo, perkainojimo, dvigubo išleidimo prevencija
- Ištirtos modelio saugumo charakteristikos ir parinkti saugūs sistemos kriptografiniai parametrai. Pakankamo saugumo maišos funkcija SHA1 (160 bitų). Kiti skaičiavimai atliekami su 512 bitų ilgio skaičiais.
- Nustatyta mokėjimo trukmė tarp pirkėjo ir pardavėjo mikroprocesorinėje kortelėje yra ne ilgesnė nei 7 sekundės. Tai pagrindžia sistemos efektyvumą

REKOMENDACIJOS

- Norint realizuoti didesnių sumų mokėjimus, reikia įvesti papildomus saugiklius schemoje.
- Įvesti papildomus pinigų nominalus

LITERATŪRA

Canetti, R., O. Goldreich, S. Halevi. The Random Oracle Methodology, Revisited, 1998.

Chaum, D. Blind signatures for untraceable payments, 1983.

Mambo, M., K. Usuda, E. Okamoto. Proxy Signatures: Delegation of the Power to Sign Messages. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 1996. E79-A(9), 1338–1354.

Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008.

Rivest, R. L. PayWord and MicroMint: Two simple micropayment schemes, 1996.

Rosenberg, B. Handbook of Financial Cryptography and Security. CRC Press, 2010.

Sakalauskas, E., N. Listopadskis, G. S. Dosinas, K. Lukšys, A. Katvickis. Kriptografinės sistemos. Vitae Litera, Kaunas, 2007.

Schmeh, K. Cryptography and public key infrastructure on the Internet. John Wiley and Sons, 2003.

Tan, Z. Hierarchical Proxy Blind Signature: A Solution to E-cash in the Real World. In The 9th International Conference for Young Computer Scientists. CWI (Centre for Mathematics and Computer Science), Zhang Jia Jie, Hunan, China, 2008 .

PRIEDAS

Wolfram Alpha programos kodas

```
(*  Kuriami sisteminiai parametrai *)
For[i=RandomPrime[5000],PrimeQ[2*i+1]==False,i=RandomPrime[5000]];
q=i
p=2*i+1
For[

g=RandomInteger[{1,p}];,
(PowerMod[g,q,p]!=1&&PowerMod[g,2,p]!=1)==False,
g=RandomInteger[{1,p}];

];
1901 3803 1099
(* Sukuriamos raktu poros

x[[1]] - pagr organizacijos
x[[2]] - banko
x[[3]] - kliento

*)
x=Table[RandomInteger[{1,q}] ,{i,1,3}]
y=Table[PowerMod[g,x[[i]],p],{i,1,3}]
k=Table[RandomInteger[{1,q}],{i,1,4}];
r=Table[PowerMod[g,k[[i]],p],{i,1,4}];
mwO="Pagr organizacija";
{46,163,265} {3392,3492,3219}
(*  Perduodami igaliojimai  *)
(* HD1 - pagr organizacijoa deleguoja bankui *)
rOB=r[[1]]*r[[2]] ;(* skaiciuoja abu *)
h1 = Mod[Hash[{mwO,Mod[rOB,q]}],q];
s1 = Mod[k[[1]]*rOB+x[[1]]*h1,q]; (* Skaiciuoja pagr organizacija, siuncia bankui *)
s2 = Mod[k[[2]]*rOB+x[[2]]*h1,q]; (* Skaiciuoja bankas *)
(* delegavimo raktas x,y, kuri skaiciuojasi bankas *)
xOB=Mod[s1+s2,q]
yOB=PowerMod[g,xOB,p]
118 517
(* bankas siuncia klientui siuos:

yO,yB,rOB,mwO
Bankas isduoda sertifikata ir pasiraso

*)
c1=Hash[RandomInteger[5000]]; (*Vieno cento moneta*)
c10=Hash[RandomInteger[5000]] ;(*Desimt centu moneta*)
```

```
c100=Hash[RandomInteger[5000]] ;(*Simto centu moneta*)
max=1000;
c={c1,c10,c100};
Kid=101; (* kliento id *)
Bid=202; (* banko id *)
Expr=1305884062; (* Fri,20 May 2011 09:34:22 GMT *)
certK={Kid,Bid,c,max,Expr,y[[3]]};
k[[2]]=RandomInteger[{1,q}];
r[[2]]=PowerMod[g,k[[2]],p];
h=Hash[{certK,mwO,rOB,r[[2]]}]
s=Mod[k[[2]]+h*x[[2]],q];
(* bankas siuncia klientui siuos: certK,h,s *)
2064294
(* Klientas pasitikrina gauta sertifikata *)
rr=Mod[PowerMod[g,s,p]*PowerMod[y[[2]],Mod[-h,q],p],p];
hh=Hash[{certK,mwO,rOB,rr}]
2064294
(* Pirkimo paraiskos siuntimas *)
Pid =303;(* pardavejo id *)
timePP=1305804648;
PP={Kid,Pid,certK,timePP}
{101,303,{101,202,{769175894,1760298541,569033542},1000,1305884062,3219},1305804648}
(* Pirkimo zinutes siuntimas *)
g1=0;
k1=0;
l1=0;
sakc1=c1;
sakc10=c10;
sakc100=c100;
kaina1=1.23;
(* skaiciuojama, kiek kokiu monetu reikia *)
i1=IntegerPart[Mod[kaina1*100,10]];
j1=(IntegerPart[Mod[kaina1*100,100]]-i1)/10;
m1=(IntegerPart[Mod[kaina1*100,1000]]-i1-10*j1)/100;
timePZ1=1305804423;
mint[c_]:=Hash[{c}];(* funkcija monetoms kurti *)
PZ1={g1,k1,l1,

Nest[mint,sakc1,g1+i1],
Nest[mint,sakc10,k1+j1],
Nest[mint, sakc100,l1+m1],

i1,j1,m1,timePZ1}
{0,0,0,1119775539,1344273271,1386200163,3,2,1,1305804423}
(* pirkimo zinutes tikrinimas *)
```

```
IPS=PZ1[[1]]*1+PZ1[[2]]*10+PZ1[[3]]*100; (* isleista pinigu suma *)
RPS=PZ1[[7]]*1+PZ1[[8]]*10+PZ1[[9]]*100; (* reikalinga pinigu suma *)
If[RPS+IPS<certK[[4]],"Pinigu pakanka","Pinigu nepakanka"] (* tikrinamos monetos *)
c1'=Nest[mint,certK[[3]][[1]],PZ1[[1]]+PZ1[[7]]]
c2'=Nest[mint,certK[[3]][[2]],PZ1[[2]]+PZ1[[8]]]
c3'=Nest[mint,certK[[3]][[3]],PZ1[[2]]+PZ1[[9]]]
If[PZ1[[4]] == c1',"1ct monetos teisetos","1ct monetos neteisetos"]
If[PZ1[[5]] == c2',"10ct monetos teisetos","10ct monetos neteisetos"]
If[PZ1[[6]] == c3',"1lt monetos teisetos","1lt monetos neteisetos"]
Pinigu pakanka
1119775539
1344273271
1386200163
1ct monetos teisetos
10ct monetos teisetos
1lt monetos teisetos
(* Pirkimo zinutes siuntimas *)
kaina2=1.11;
(* saknines monetos pakeiciamos paskutinemis isleistomis *)
g2=g1+i1;
k2=k1+j1;
l2=l1+m1;
sakc1=1119775539;
sakc10=1344273271;
sakc100= 1386200163;
(* skaiciuojama, kiek kokiu monetu reikia *)
i2=IntegerPart[Mod[kaina2*100,10]];
j2=(IntegerPart[Mod[kaina2*100,100]]-i2)/10;
m2=(IntegerPart[Mod[kaina2*100,1000]]-i2-10*j2)/100;
timePZ1=1305806423;
mint[c_]:=Hash[{c}];(* funkcija monetoms kurti *)
PZ2={g1+i1,k1+j1,l1+m1,

Nest[mint,sakc1,i2],
Nest[mint,sakc10,j2],
Nest[mint, sakc100,m2],

g2+i2,k2+j2,l2+m2,timePZ1}
{3,2,1,473083565,662468616,1339743239,4,3,2,1305806423}
(* pirkimo zinutes tikrinimas *)
IPS=PZ2[[1]]*1+PZ2[[2]]*10+PZ2[[3]]*100 (* isleista pinigu suma *)
RPS=PZ2[[7]]*1+PZ2[[8]]*10+PZ2[[9]]*100 (* reikalinga pinigu suma *)
If[RPS+IPS<certK[[4]],"Pinigu pakanka","Pinigu nepakanka"] (* tikrinamos monetos *)
c1'=Nest[mint,certK[[3]][[1]],PZ2[[7]]];
c2'=Nest[mint,certK[[3]][[2]],PZ2[[8]]];
```

```
c3'=Nest[mint,certK[[3]][[3]],PZ2[[9]]];
If[PZ2[[4]] == c1',"1ct monetos teisetos","1ct monetos neteisetos"]
If[PZ2[[5]] == c2',"10ct monetos teisetos","10ct monetos neteisetos"]
If[PZ2[[6]] == c3',"1lt monetos teisetos","1lt monetos neteisetos"]
Pinigu pakanka
1ct monetos teisetos
10ct monetos teisetos
1lt monetos teisetos
```