KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Paulius Palevičius

ELEKTRONINIŲ PINIGŲ MODELIO REALIZACIJA STANDARTINĖSE IR RIBOTŲ ARITMETINIŲ FUNKCIJŲ SISTEMOSE

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

ELEKTRONINIŲ PINIGŲ MODELIO REALIZACIJA STANDARTINĖSE IR RIBOTŲ ARITMETINIŲ FUNKCIJŲ SISTEMOSE

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
prof. dr. E. Sakalauskas
2010 06 01

Recenzentas
doc.dr. A.Venčkauskas
2010 06 01

Atliko
FMMM 9/1 gr. stud.
P. Palevičius
2010 05 30

KAUNAS, 2011

KVALIFIKACINĖ KOMISIJA

Pirmininkas: Leonas Saulis, profesorius (VGTU)

Sekretorius: Eimutis Valakevičius, docentas (KTU)

Nariai: Algimantas Jonas Aksomaitis, profesorius (KTU)

Vytautas Janilionis, docentas (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Rimantas Rudzkis, habil. dr., vyriausiasis analitikas (DnB NORD Bankas)

Zenonas Navickas, profesorius (KTU)

Arūnas Barauskas, dr., vice-prezidentas projektams (UAB „Baltic Amadeus“)

SUMMARY

Palevičius P. Electronic money model implementation in standard and limited arithmetics systems : 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. – 47 p.

SUMMARY

As mobile phones and technology advance new opportunities for implementation of elektronic money systems become possible. Electronic money is one of the latest methods for paying for goods and there are just a few implementations. In this work implementation of Stefan Brands electronic money model was performed. Stefan Brands protocol was implemented using Java langauge in standard computer and in mobile phone. Efficiency of these implementations was estimated and it was found that implementation on mobile phone is approximetly 100 times slower using Java ME platform. A library for for doing arithmetic operations like addition, subraction, modulus, modular exponentiation, right shift, etc. was implemented using smart card enviroment. As it is not possible to use cryptographic processor directly, RSA encryption scheme was used to perform modular exponention. Results of implementation speed were given and it was concluded that whole client side implementation using Java Card enviroment was two slow so mixed model was suggested. This work also consist of technical and software analysis needed to perform electronic money implementation. Also a brief review of cryptographic ant mathematic methods used in Stefan Brands digital cash system was performed.

TURINYS

ILIUSTRACIJŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

ĮVADAS

Tobulėjant mobiliesiems telefonams ir kitoms technologijoms, atsiranda galimybė pakeisti arba papildyti rinkoje naudojamus grynuosius pinigus jiems ekvivalenčiais elektroniniais pinigais. Kadangi elektroniniai pinigai yra skaitmeninė informacija, kurią nesudėtinga kopijuoti, šiai informacijai apsaugoti yra pasitelkiami kriptografiniai ir matematiniai metodai. Ateities technologijos įgalins žmones atsisakyti identifikacijos dokumentų, kreditinių ir kitų kortelių, jas realizuojant mobiliajame telefone. Šioje infrastruktūroje atsiranda vieta elektroniniams pinigams.

Kadangi elektroniniai pinigai yra viena naujausių elektroninio atsiskaitymo technologijų, šiuo metu jų realizacija rinkoje yra ribota. Europos Komisija jau 2000 metais išleido direktyvą, skatinančią elektroninių pinigų sistemų kūrimą. 2006 metais buvo atliktas šios direktyvos vertinimas ir dėl 2000 metų direktyvos neišsamumo bei lėtos elektroninių pinigų infrastruktūros adaptacijos Europos rinkoje, 2009 metais buvo paruošta antroji direktyva.

Darbe realizuojamos Stephan Brands sistemos pagrindinė saugumo dalis yra diskretaus logaritmo apskaičiavimo problema, todėl sistemos protokoluose dažnai naudojama modulinės eksponentės operacija. Kadangi dirbama su dideliais skaičiais (512 bitų), labai svarbus šios operacijos realizacijos greitis. Šiuolaikiniuose kompiuteriuose darbui su dideliais skaičiais yra sukurtos įvairios bibliotekos. Kadangi mobiliųjų telefonų charakteristikos ženkliai nusileidžia šiuolaikiniams kompiuteriams, svarbu atlikti elektroninių pinigų realizacijos telefone galimybių bei techninės įrangos, reikalingos elektroninių pinigų realizacijai įgyvendinti, analizę.

Darbe surinkta metodologinė medžiaga apie šiame elektroninių pinigų modelyje naudojamas kriptografines schemas ir metodus: Schnorr autentikacijos ir parašo schema, apribojantis aklasis parašas, reprezentacijos uždavinys, nulinio atskleidimo įrodymas ir kita. Taip pat pateikiamos pagrindinės skaičių teorijos sąvokos ir apibrėžimai, reikalingi, norint įsisavinti Stefan Brands elektroninių pinigų modelio veikimo principus.

1 ANALITINĖ DALIS

Tobulėjant tokioms technologijoms, kaip mobilieji telefonai, internetas ir bevieliai tinklai atsiranda naujos komercijos galimybės. Šiais laikais didžioji dalis mokėjimų yra atliekami naudojant mokėjimo korteles, bet jos turi du didelius trūkumus: mokėjimų neanonimiškumas ir privalomas banko dalyvavimas mokėjimo metu, todėl reikalingas tiesioginis interneto ryšys su banku. Elektroniniai pinigai pagal Sakalauskas et al. (2007) apibrėžiami kaip finansinės kriptografijos dalis, turinti užtikrinti patogią ir saugią pinigų bei prekių (paslaugų) cirkuliaciją interneto tinkle tarp trijų pagrindinių subjektų: vartotojo, banko ir pardavėjo. Elektroninių pinigų modeliai yra kuriami taip, kad kuo geriau imituotų paprastų pinigų savybes. Elektroninių pinigų sistemose, subjektai vietoje monetų ir banknotų keičiasi skaitmeniniais duomenimis, kuriuose laikomi piniginį vienetą apibūdinantys parametrai.

1.1 ELEKTRONINIŲ PINIGŲ MODELIAI, SAVYBĖS IR TENDENCIJOS

Anoniminius elektroninius pinigus pirmasis pristatė David Chaum (Chaum, 1998). Jis panaudojo akląjį parašą, kad būtų pasiektas anonimiškumas pinigo paėmimo iš banko ir išleidimo metu. Pirmąjį elektroninių pinigų modelį veikiantį atjungties (offline) rėžimu 1990 m. pristatė Amos Fiat, David Chaum ir Moni Naor. Šios sistemos pagrindą sudarė RSA aklasis parašas. Okamota ir Ohta pristatė modelį, kuriame įgyvendintas pinigų perdavimas kitam vartotojui ir pinigų dalumas, bet šis modelis buvo neefektyvus ir reikalavo daug sistemos resursų (Okamoto, 1995). 1993 metais Stephan Brands straipsnyje „An Efficient Off-line Electronic Cash System Based on the Representation Problem“ pristatė modelį, kuriame buvo ženkliai pagerintas saugumas ir resursų panaudojimas. Šio modelio principai vėliau buvo naudojami kaip pagrindas daugeliui kitų modelių sukurti. Neigiama anoniminių elektroninių pinigų pusė yra ta, kad jie palengvina nusikaltimų, tokių kaip pinigų plovimas, terorizmas, šantažavimas įvykdymą, todėl iki šiol siekiama sukurti sistemas, kuriose būtų įgyvendinta prevencija, neleidžianti elektroninius pinigus naudoti šiems nusikaltimams vykdyti.

Yra sukurta įvairių elektroninių pinigų modelių, kurie tenkinta skirtingus reikalavimus. Galima išskirti pagrindines elektroninių pinigų savybes, kuriomis būtų galima apibūdinti elektroninių pinigų modelius:

1. Saugumas – pinigai kompiuteriniais tinklais turi keliauti saugiai.
2. Klastojimas – pinigai negali būti klastojami.
3. Dvigubo išleidimo prevencija – neturi būti galimybės pinigus išleisti kelis kartus.
4. Anonimiškumas – bankas neturi turėti galimybės susieti pinigą su vartotoju be patikimos šalies sutikimo.
5. Efektyvumas – elektroninių pinigų sistema turi būti efektyvi talpos, skaičiavimo ir komunikacijų atžvilgiu.

6. Atjungties būsena – elektroniniai pinigai turėtų dirbti nereikalaujant tiesioginio ryšio su banku.

Taip pat yra išskiriamos papildomos elektroninių pinigų savybės:

1. Vartotojo sekimas – banko galimybė sekti vartotojo veiksmus su pinigais, gavus patikimos šalies leidimą.
2. Pinigo grąžinimas – paimtas pinigas gali būti grąžintas bankui.
3. Nesusiejimas – neturi būti galimybės susieti du ar daugiau to paties vartotojo išleistus pinigus.
4. Nepagrįstas apkaltinimas – vartotojas arba pardavėjas negali būti nepagrįstai apkaltinti banko, jei nepadarė nusikaltimo.
5. Perdavimas – pinigus turi būti leidžiama perduoti kitiems vartotojams.
6. Dalumas – pinigai turėtų būti dalomi į mažesnes dalis.

Viena svarbiausių savybių skirianti elektroninius pinigus nuo mokėjimo kortelių yra anonimiškumas. Šios savybės dėka elektroniniai pinigai veikia kaip pakaitalas gryniesiems pinigams, bet įvedant anonimiškumą į elektroninių pinigų modelį atveriamas kelias nusikalstamumui, todėl daugelio kuriamų sistemų tikslas yra to išvengti.

1.2 ELEKTRONINIŲ PINIGŲ TEISINIAI AKTAI EUROPOS SĄJUNGOJE

Europos Parlamentas ir Taryba 2000 metų rugsėjo 18 d. išleido direktyvą 2000/46/EB dėl elektroninių pinigų įstaigų steigimosi, veiklos ir riziką ribojančios priežiūros. Europos sąjunga elektroninius pinigus apibrėžia kaip piniginę vertę, išreikštą kaip pretenziją emitentui:

1. kuri yra saugoma elektroninėse laikmenose;
2. kuri išleidžiama gavus lėšų, kurių suma yra ne mažesnės vertės negu išleista piniginė vertė;
3. kurią įmonės, kurios nėra emitentai, priima kaip mokėjimo priemonę.

Dėl lėtos elektroninių pinigų adaptacijos ES valstybėse, 2006 metų vasario 2 buvo pristatyta ataskaita „Elektroninių pinigų direktyvos 2000/46/EB įvertinimas“, kurioje nustatyta, kad 2000/46/EB direktyva buvo neišsami ir pateikti siūlymai jos tobulinimui. Šioje ataskaitoje teigiama, kad elektroninių pinigų potencialas kol kas nebuvo atskleistas ir paskatinta aktyviau vystyti elektroninių pinigų sistemas.

Dėl šių priežasčių išleista nauja ES direktyva 2009/110/EC, kurios pagrindiniai tikslai:

- Įgalinti naujų, inovatyvių ir saugių elektroninių pinigų paslaugų kūrimą.
- Suteikti priėjimą prie rinkos naujoms įmonėms.
- Skatinti efektyvią konkurenciją tarp visų rinkos dalyvių.

1.3 KRIPTOGRAFIJOS IR SKAIČIŲ TEORIJOS TAIKYMAS ELEKTRONINIŲ PINIGŲ MODELIUOSE

Elektroninių pinigų modelių pagrindą sudaro elektroniniu parašu pasirašytos žinutės. Dėl šios priežasties elektroninių pinigų modeliuose naudojamos plačiai žinomos elektroninių parašų schemos. Vienos populiariausių elektroninių parašų schemos yra Rivest et al. (1978) RSA ir 1991 metais pasiūlyta DSA (angl. Digital Signature Algorithm) schemos.

Elektroninių pinigų modelyje didelę svarbą užima subjektų autentikacija. Šiame darbe naudosime Claus P. Schnorr sukurtą autentikacijos ir parašo schemą. Šios schemos saugumas paremtas diskretaus logaritmo apskaičiavimo problema. Schnorr schema taip pat pasitelkiama įgyvendinant apribojantį akląjį parašą ir prevencijai nuo daugkartinio pinigo išleidimo.

Esminė elektroninių pinigų savybė yra anonimiškumas. Norint, kad elektroninis pinigas būtų anonimiškas, kai kurios žinutės turi būti pasirašomos aklai – nematant pačios žinutės turinio, todėl naudojama apribojančio aklojo parašo schema. Šio metodo dėka, bankas pasirašydamas ant elektroninio pinigo, pačio pinigo nemato ir todėl negali susieti vartotojo su pinigu.

Dauguma skaičiavimų viešojo rakto kriptografijoje ir elektroniniuose piniguose yra atliekama lauke $\mathbb{Z}_p$, kur p yra didelis pirminis skaičius, ir grupėje G_q, kur q yra taip pat didelis pirminis skaičius bei tenkinama sąlyga $p = kq + 1$. Šios dvi algebrinės struktūros atlieka svarbų vaidmenį Diffie-Hellman raktų apsikeitimo protokole, Schnorr autentikacijos ir parašo schemoje (Schnorr, 1991), DSA schemoje ir Stephan Brands (Brands, 1993) elektroninių pinigų modelyje.

Elektroninio parašo schemose didelę svarbą turi diskretaus logaritmo apskaičiavimo uždavinys grupėje $\mathbb{Z}_p$, kadangi tiesioginė (kėlimo laipsniu) operacija atliekama labai greitai, o diskrečiojo logaritmo radimas, kai p pakankamai didelis, yra sudėtinga ir kol kas neišspręsta matematinė problema.

Elektroninių parašų ir pinigų sistemose labai dažnai naudojamos kriptografinės vienkryptės funkcijos, kitaip dar vadinamos H funkcijomis. H funkcijos naudinga naudoti tuomet, kai norime neriboto ilgio duomenis atvaizduoti į fiksuoto ilgio santrauką. Tokiu būdu galima efektyviau panaudoti skaičiavimo resursus.

1.4 STEFAN BRANDS ELEKTRONINIŲ PINIGŲ MODELIS

Stephan Brands pirmasis pristatė elektroninių pinigų modelį, paremtą reprezentacijos uždaviniu, todėl šios sistemos saugumas ekvivalentus diskretaus logaritmo apskaičiavimo uždaviniui. Reprezentacijos uždavinio panaudojimas leido atsisakyti ankstesnėse sistemose naudojamos iškirpimo ir paėmimo (angl. cut and choose) metodologijos, todėl, palyginus su kitomis sistemomis, Stephan Brands elektroninių pinigų modelis pasižymi didesniu efektyvumu ir saugumu. Stephan Brands modelis leidžia sukurti išplėtimus (angl. exetensions), kuriuos buvo sudėtinga įgyvendinti senesnėse sistemose. Svarbiausi mo-

delio išplėtimai yra stebėtojo (angl. observer) ir piniginės, kurie atlieka didelį vaidmenį elektroninių pinigų dvigubo išleidimo problemai spręsti. Šis modelis taip pat užkerta galimybę bankui nesąžiningai apkaltinti vartotoją arba pardavėją, kas buvo taip pat labai didelis trūkumas iki tol sukurtose sistemose. Modelis taip gali būti išplėstas taip, kad būtų palaikomas pinigų dalumas, bet iki šiol nerastas efektyvus būdas tai padaryti. Dauguma elektroninių pinigų modelių naudoja įvairią metodologiją, kuri užkerta kelią tuos modelius pritaikyti kitoms grupėms. Stephan Brands modelyje naudojami metodai ir principai leidžia ši modulį pritaikyti RSA grupėms arba eliptinių kreivių grupei.

1.5 TECHNINĖ IR PROGRAMINĖ ĮRANGA

Pastaraisiais metais, tobulėjant technologijoms, rinkoje atsiranda vis daugiau įrenginių, įgalinančių elektroninių pinigų ir kitų kriptografinių schemų realizaciją. Prieš keletą metų rinkoje pasirodęs telefonas NOKIA 6212 Classic įgalina kurti programas ne tik telefone, bet ir telefone esančioje mikroprocesorinėje kortelėje. Realizuojant elektroninius pinigus mikroprocesorinėje kortelėje atsiranda naujos galimybės, bet su jomis iškyla ir nauji iššūkiai: sistemos realizacija atliekama ribotų resursų sistemose su ribotu aritmetinių funkcijų palaikymu. Dėl šios priežasties reikalingi metodai, leidžiantys tokias operacijas, kaip daugyba ir modulinė eksponentė atlikti su 512 bitų skaičiais. Sterckx et al. (2009) pasiūlyta metodika, leidžianti įgyvendinti nesudėtingą teisių valdymo sistemą mikroprocesorinėje kortelėje, gali būti pritaikyta ir daug sudėtingesnėje elektroninių pinigų sistemos realizacijoje.

Realizuojant elektroninių pinigų sistemą programavimo kalbų skaičius yra ribotas. Mobiliajame telefone NOKIA 6212 Classic ir mikroprocesorinėje kortelėje naudojamos supaprastintos Java kalbos versijos. Dėl šios priežasties, kompiuterinėje realizacijoje taip pat pasirinkta Java kalba.

Realizacijos metu naudojami įrankiai buvo pasirenkami pagal jų prieinamumą, pirmenybę teikiant atviro kodo programoms. Taip pat daug dėmesio buvo skiriama įrankiams, kurie leistų automatizuoti programų paketų kūrimo ir diegimo procesus. Buvo pasirinktos tokios programos, kaip GPShell, ANT, JCWDE ir kt. kurios leido sumažinti pasikartojančių procedūrų kiekį.

1.6 DARBO UŽDAVINIAI

Pagrindinis darbo tikslas yra elektroninių pinigų modelio realizacija standartinėse ir ribotų aritmetinių funkcijų sistemose. Žemiau pateiktas detalus keliamų uždavinių sąrašas:

- Programinės ir techninės įrangos, susijusios su realizacija mobiliajame telefone ir mikroprocesorinėje kortelėje, analizė.
- Stefan Brands elektroninių pinigų schemoje naudojamų kriptografinių ir matematinių metodų aprašymas.
- Įrankių, įgalinančių greičiau ir efektyviau realizuoti kriptografines sistemas mobiliajame telefone sukūrimas.

- Elektroninių pinigų realizacija kompiuteryje ir mobiliajame telefone, bei šių realizacijų efektyvumo laiko atžvilgiu įvertinimas.
- Stafan Brands metodui reikalingų aritmetinių funkcijų realizacija mikroprocesorinėje kortelėje darbui su dideliais skaičiais.
- Laiko, reikalingo vartotojo protokolams vykdyti naudojant aritmetinių funkcijų realizaciją, įvertinimas.

2 METODOLOGINĖ DALIS

Šiame skyriuje aprašomi kriptografiniai metodai ir modeliai naudojami Stefan Brands elektroninių pinigų schemoje, pateikiamos pagrindinės matematinės sąvokos ir apibrėžimai naudojami konstruojant šiuos modelius. Aprašomos Schnorr autentikacijos ir parašo schemos, kriptografinės vienkryptės funkcijos, Stefan Brands elektroninių pinigų modelis. Atliekama techninės ir aparatinės įrangos, naudojamos elektroninių pinigų realizacijoje, analizė.

2.1 MODULINĖ ARITMETIKA, GRUPIŲ IR LAUKO TEORIJA

Modulinė aritmetika atlieka labai svarbų vaidmenį kriptografinėse parašo schemose ir elektroninių pinigų modeliuose. Žemiau pateikiami pagrindiniai apibrėžimai ir teoremos, naudojamos kriptografinių protokolų sudarymui.

Modulinė aritmetika

Turime natūralųjį skaičių n. Jeigu sveikųjų skaičių a ir b skirtumas $a-b$ dalijasi iš n , sakome, kad a lygsta b moduliu n , ir rašome

$$a \equiv b \pmod{n}.$$

Šį reiškinį vadiname kongruencija (Bulota, 1990).

Pagal Menezes et al. (1996), visiem $a, a_1, b, b_1, c \in Z$, galioja savybės:

1. $a \equiv b \pmod{n}$ jeigu a ir b dalinant iš n gaunama ta pati liekaną.
2. $a \equiv a \pmod{n}$ (refleksyvumas).
3. Jei $a \equiv b \pmod{n}$ tai ir $b \equiv a \pmod{n}$ (simetriškumas).
4. Jei $a \equiv a \pmod{n}$ ir $a \equiv a \pmod{n}$, tai $a \equiv a \pmod{n}$ (tranzityvumas).
5. Jei $a \equiv a_1 \pmod{n}$ ir $b \equiv b_1 \pmod{n}$, tuomet $a + b \equiv a_1 + b_1 \pmod{n}$ ir $ab \equiv a_1 b_1 \pmod{n}$.

Kongruenciją $a \equiv b \pmod{n}$ galime keisti ekvivalenčia lygybe $a = b + tn,\ t \in \mathbb{Z}$. Iš reiškinio $b \pmod{n} = c$ gauname, kad c yra liekana dalijant b iš n. Sąryšis $a \equiv b \pmod{n}$ yra ekvivalentumo sąryšis sveikųjų skaičių aibėje $\mathbb{Z}$, todėl suskaido ją į nesikertančias ekvivalentumo klases $\bar{a}$. Liekanų klasę $\bar{a}$ moduliu n sudaro sveikieji skaičiai:

$$\bar{a} = \{a + nt \,|\, t \in \mathbb{Z}\}$$

Ši liekanų klasių aibė dažnai žymima $\mathbb{Z}_n$. Kitaip tariant, visų sveikųjų skaičių aibė $\mathbb{Z}$ atvaizduojama į baigtinę aibę $\{0, 1, 2, \ldots, n-1\}$ t.y. siurjekcija $\mathbb{Z} \to \mathbb{Z}_n$ (Sakalauskas et al., 2007).

Grupių ir lauko teorija

Grupė $(G, *)$ susideda iš aibės G ir toje aibėje apibrėžtos dvinarės operacijos $*$, tenkinančios sąlygas:

1. Grupės operacija yra asociatyvi: $(a * b) * c = a * (b * c)$ visiems $a, b, c \in G$.
2. Egzistuoja vienintelis neutralus elementas $1 \in G$, tenkinantis $a * 1 = 1 * a = a$ visiems $a \in G$.
3. Kiekvienam $a \in G$ egzistuoja elementas $a^1 \in G$, vadinamas atvirkštiniu a elementu ir tenkinantis sąlygą $a * a^{-1} = a^{-1} * a = 1$.

   Grupė vadinama Abelio (arba komutatyviaja), jeigu:

4. $a * b = b * a$ visiems $a, b \in G$.

Grupė G vadinama cikline jei egzistuoja toks elementas $\alpha \in G$, kur kiekvienam $b \in G$ egzistuoja toks skaičius i su kuriuo $b = \alpha^i$. Toks elementas α vadinamas grupės G generatoriumi.

Aibės $\mathbb{Z}_n$ multiplikativioji grupė yra $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \,|\, gcd\,(a, n) = 1\}$. Atskiru atveju, jeigu n yra pirminis skaičius, tai $\mathbb{Z}_n^* = \{a \,|\, 1 \leq a \leq n - 1\}$.

Tarkime p yra pirminis skaičius, tuomet:

1. Jei $gcd\,(a, p) = 1$, tuomet pagal Ferma teoremą $a^{p-1} \equiv 1 \ \ (mod\ p)$.
2. Jei $r \equiv s \ \ (mod\ p - 1)$, tuomet $a^r \equiv a^s \ \ (mod\ p)$ visiems sveikiesiems skaičiams a t.y. atliekant veiksmus moduliu p, kur p yra pirminis skaičius, eksponentė gali būti sumažinta moduliu $p - 1$.
3. Visiems sveikiesiems skaičiams a galioja $a^p \equiv a \ \ (mod\ p)$.

Tarkime $a \in \mathbb{Z}_n^*$, tuomet a elemento eilė yra mažiausias teigiamas sveikasis skaičius t, su kuriuo $a^t \equiv 1 \ \ (mod\ n)$.

Jeigu elemento $a \in \mathbb{Z}_n^*$ eilė yra t ir $a^s \equiv 1 \ \ (mod\ t)$, tuomet t yra s daliklis.

Tarkime $\alpha \in Z_n^*$. Jeigu α elemento eilė yra $\phi\,(n)$, tada α yra grupės $\mathbb{Z}_n^*$ generatorius. Grupės$\mathbb{Z}_n^*$ generatorių savybės pateiktos žemiau:

1. Grupė Z_n^* turi generatorių tada ir tik tada, kai $n = 2, 4, p^k, 2p^k$, kur p pirminis skaičius ir $k \geq 1$.
2. Jeigu α yra grupės Z_n^* generatorius, tuomet $Z_n^* = \{\alpha^i \, mod\, n \,|\, 0 \leq i \leq \phi\,(n) - 1\}$.
3. Tarkime α yra grupės Z_n^* generatorius, tuomet $b = \alpha^i \, mod\, n$ yra Z_n^* generatorius, tada ir tik tada kai $gcd\,(i, \phi\,(n)) = 1$.
4. α yra grupės Z_n^* generatorius tada ir tik tada, kai $\alpha^{\phi(n)/p} \neq 1 \ \ (mod\ n)$ kiekvienam pirminiam $\phi\,(n)$ dalikliui p.

Didelė dalis darbe nagrinėjamų kriptografinių modelių ir schemų yra konstruojami pirminių skaičių lauke. Žemiau pateikiami žiedo ir lauko apibrėžimai iš (Sakalauskas et al., 2007).

Žiedu vadiname dviveiksmę algebrinę struktūrą, nusakytą netuščioje aibėje $\mathbb{Z}$, t.y. $\langle \mathbb{Z}, +, * \rangle$, jeigu:

1. Struktūra $\langle \mathbb{Z}, + \rangle$ yra adicinė Abelio grupė.
2. $\langle \mathbb{Z}, * \rangle$ yra multiplikacinė asociatyvioji struktūra (pusgrupė).
3. Su visais $a, b, c \in \mathbb{Z}$ tenkinami distributyvumo dėsniai $a * (b + c) = a * b + b * c$ ir $(b + c) * a = b * a + c * a$.

Žiedą vadiname lauku, jeigu struktūra $\langle \mathbb{Z}, * \rangle$ yra multiplikacinė Abelio grupė.

$\mathbb{Z}_n$ su įprastomis sudėties ir daugybos operacijomis moduliu n yra laukas tada ir tik tada, kai n yra pirminis skaičius (Menezes et al., 1996).

2.2 MODULINĖ EKSPONENTĖ IR DISKRETAUS LOGARITMO UŽDAVINYS

Modulinė eksponentė yra viena dažniausiai naudojamų operacijų elektroninio parašo ir elektroninių pinigų modeliuose. Labai svarbu parinkti tokius algoritmus, kad modulinės eksponentės skaičiavimai būtų atliekami kuo sparčiau ir tokius skaičius, su kuriais diskretaus logaritmo uždavinys būtų neišsprendžiamas per tam tikrą nustatytą laiką.

Modulinė eksponentė

Modulinė eksponentė - tai funkcija $y = g^x \pmod p$. Jei g, x ir p yra neneigiami skaičiai ir $g < p$, $g \neq 0$, tai egzistuoja vienintelis sprendinys y, turintis savybę $0 < y < p$ (Sakalauskas et al., 2007) .

Kriptografiniuose modeliuose dažniausiai naudojami skaičiai yra 256 ir daugiau bitų, todėl labai svarbus modulinės eksponentės apskaičiavimo laikas, panaudojant minimalius resursus. Skaičiuoti modulinę eksponentę tiesioginiu būdu, reikia pirmiausia apskaičiuoti g^x, o paskui rasti dalybos iš p liekaną. Jei kėlimas laipsniu atliekamas kaip nuosekli daugyba, reikalaujama $O(x)$ laiko (Sakalauskas et al., 2007). Šis būdas yra labai neefektyvus ir ribotų resursų sistemose jo realizacija būtų neįmanoma dėl riboto atminties kiekio.

Atmintį taupantis metodas reikalauja daugiau operacijų negu pirmasis, bet naudojant šį metodą reikalaujama mažiau atminties, o tuo pačiu operacijos vykdomos greičiau, todėl pats algoritmas yra greitesnis.

Iš kairės į dešinę dvejetainis metodas gerokai sumažina veiksmų skaičių ir naudojamos atminties kiekį, reikalingą moduliniam kėlimui laipsniu. Šis metodas - tai antrojo metodo ir bendresnio principo, vadinamo dvejetainiu kėlimu laipsniu, derinys. Kitaip jis dar vadinimas kvadratinimo metodu.

Diskretaus logaritmo uždavinys

Žemiau pateiktas modulinės eksponentės atvirkštinis uždavinys - diskretaus logaritmo apskaičiavimas $\mathbb{Z}_p^*$ pogrupiuose.

Tarkime p yra pirminis skaičius ir q yra skaičiaus $p-1$ daliklis. Tegul G yra q eilės ciklinis $\mathbb{Z}_p^*$ pogrupis ir α yra G generatorius. Tuomet diskretaus logaritmo problema grupėje G apibrėžiama taip: turint $q, p, \alpha, \beta \in G$ reikia rasti sveikąjį skaičių $x,\ 0 \leq x \leq q-1$, su kuriuo butų tenkinama lygybė $\alpha^x \equiv \beta\ \ (mod\ p)$ (Menezes et al., 1996).

Tinkamai parinkus p, šis uždavinys gali būti labai sudėtingas. Parenkant p svarbu, kad $p-1$ turėtų bent vieną labai didelį pirminį daugiklį, nes Pohling-Hellman algoritmas yra efektyvus skaičiuojant diskretų logaritmą, kuomet $p-1$ sudarytas iš mažų pirminių dauginamųjų. 2.1 lentelėje pateikti diskretaus logaritmo skaičiavimo įverčiai.

2.1 lentelė: Diskretaus logaritmo skaičiavimo algoritmai ir sudėtingumo įverčiai

Algoritmas	Sudėtingumo įvertis
Pollard's Rho	$O\left(\sqrt{n}\right)$
Pohling-Hellman	$O\left(\sum_{i=1}^{r} e_i \left(\log n + \sqrt{p_i}\right)\right),\ n = p_1^{e_1} p_2^{e_2} \ldots p_r^{e_r}$
Paprastas perrinkimas	$O\left(n\right)$
Mažo - didelio žingsnio algoritmas	$O\left(\sqrt{n}\right)$
Indeksų skaičiavimo algoritmas	$O\left(e^{2\sqrt{\ln p}\sqrt{\ln \ln p}}\right)$

2.3 KRIPTOGRAFINĖS VIENKRYPTĖS FUNKCIJOS

Elektroninių parašų ir pinigų modeliuose svarbi sudedamoji dalis yra kriptografinės vienkryptės funkcijos, kitaip dar vadinamos H funkcijomis. Stipri H funkcija yra tokia funkcija, kuri neriboto ilgio tekstogramą M pakeičia į fiksuoto bitų skaičiaus eilutę $H\left(M\right)$ ir tenkina tokias sąlygas (Wagstaff, 2003):

1. Turint M turi būti lengva apskaičiuoti $H\left(M\right)$.
2. Turi veikti viena kryptimi – turint $H\left(M\right)$, turi būti sunku rasti tekstogramą M.
3. Turint tekstogramą M_1 turi būti sunku rasti tokią tekstogramą M_2, kad $H\left(M_1\right) = H\left(M_2\right)$.
4. Turi būti sunku rasti tokius M_1 ir M_2 , kad $H\left(M_1\right) = H\left(M_2\right)$.

Vienas dažniausiai pasitaikančių H funkcijų panaudojimo atvejų yra elektroninio parašo schemose. Ilgas pranešimas arba dokumentas yra suspaudžiamas į fiksuoto ilgio santrauką ir tuomet pasirašomas. Tikrintojas gavęs pranešimą suspaudžia jį į tą pačią santrauką ir patikriną parašą. Tokiu būdu sutaupomi techninės įrangos, naudojamos pasirašymui, resursai ir sumažinamas elektroninio parašo dydis. H funkcijos taip pat naudojamos duomenų integralumui patikrinti, kuomet tam tikru laiko momentu sukuriama duomenų santrauka ir vėliau galima patikrinti ar duomenys buvo modifikuoti (Menezes et al., 1996).

Žinomiausi H funkcijų algoritmai yra MD5, RIPEMD ir SHA. Darbe buvo pasirinktas SHA (angl. Secure Hash Algorythm) algoritmų klasės variantas SHA-1. SHA-1 algoritmas yra sukurtas NSA (angl. Nacional Security Agency) ir plačiai naudojamas daugelyje saugumo programų ir protokolų.

2.4 SCHNORR AUTENTIKACIJOS IR PARAŠO SCHEMOS

Pasirinktas Stefan Brands elektroninių pinigų modelis naudoja Schnorr autentikacijos schemą. Žemiau pateikiamas Schnorr sistemos parametrų inicijavimas bei autentikacijos ir elektroninio parašo schemos (Schnorr, 1991).

Sistemos inicijavimas

Tarkime turime jau apibrėžtą struktūrą $\langle \mathbb{Z}_p, +, * \rangle$ - sveikųjų skaičių moduliu p lauką. Sistemos inicijavimo metu, raktų autentikacijos centras pasirenka ir apskaičiuoja sistemos parametrus, kurie bus naudojami tolimesniuose skaičiavimuose. Raktų autentikacijos centro veiksmų seka pateikta žemiau:

1. Parenkami tokie pirminiai skaičiai p ir q, kad būtų tenkinama lygybė $p = 2q + 1$.
2. Randamas $g \in \mathbb{Z}_p$ su eile q, tenkinantis sąlygas $g^q \equiv 1 \pmod p$ ir $g \neq 1$.
3. Pasirenkama kriptografinė vienkryptė funkcija H.
4. Apskaičiuojamas viešasis bei privatusis raktai.

Parametrai p, g, q, H ir viešasis raktai paviešinami.

Vartotojų registracijos protokolas

Vartotojų registracijos metu vartotojas pasirenka privatųjį raktą $x \in \{1, 2, \ldots, q\}$ ir apskaičiuoja viešąjį raktą $h \equiv g^x \pmod p$. Raktų autentikacijos centras išsaugo vartotojo identifikacijos numeri I kartu su informacija apie vartotoją bei uždeda parašą S ant (I, h).

Vartotojų autentikacijos protokolas

Vartotojų autentikacijos protokolo dalyviai yra vartotojas A ir tikrintojas B. Autentikacijos protokolo veiksmų seka pateikta žemiau:

1. A pasirenka atsitiktinį skaičių $w \in \{1, \ldots, q-1\}$ ir apskaičiuoja $a \equiv g^w \pmod p$.
2. A nusiunčia B savo identifikacijos numerį I, viešąjį raktą h, parašą S bei pasirinktą atsitiktinį skaičių w.
3. B patikrina parašą S ir nusiunčia atsitiktinį skaičių $c \in \{1, \ldots, q-1\}$ vartotojui A.
4. A apskaičiuoja $y \equiv w + xc \pmod q$ ir nusiunčia vartotojui B.
5. B patikrina ar tenkinama lygybė $g^x \equiv ah^c \pmod p$.

Elektroninio parašo protokolas

Elektroninio parašo protokolo dalyviai yra A ir parašo tikrintojas B. Elektroninio parašo protokolo veiksmų seka pateikta žemiau:

1. A pasirenka atsitiktinį skaičių $w \in \{1, \ldots, q-1\}$ ir apskaičiuoja $a = g^w \pmod p$.
2. A sujungia tekstogramą M ir a bei apskaičiuoja $c = H(M, a)$.
3. A apskaičiuoja $y \equiv w + xc \pmod q$ ir nusiunčia (c, y) kartu su a ir M tikrintojui B.
4. B patikrina ar tenkinama lygybė $g^y \equiv ah^c \pmod p$.
5. B patikrina ar tenkinama lygybė $c = H(M, a)$.
6. B priima parašą jei tenkinamos abi lygybės.

2.5 KITI KRIPTOGRAFINIAI METODAI IR SCHEMOS

Reprezentacijos uždavinys

Sudėtingesnėms sistemoms apibrėžiamas bendresnis diskretaus logaritmo uždavinio atvejis - daugiamatis diskretaus logaritmo uždavinys arba kitaip dar vadinamas reprezentacijos uždavinys, kuris leidžia praplėsti diskretaus logaritmo uždavinį įvedant daugiau komponenčių. Tai suteikia didesnį lankstumą, kuriant sudėtingas kriptografines sistemas, tarp jų ir elektroninių pinigų sistemas. Pirmą kartą šį apibendrinimą pateikė Brands (1993).

Tarkime $k > 1$ ir $1 \leq a_i \leq q$, kiekvienam $i = 1 \ldots k$. Reprezentacijos uždavinys - rasti indeksų vektorių $\{a_1, \ldots, a_k\}$, turint generatorių vektorių $\{g_1, \ldots, g_k\} \in G_q$, kad būtų tenkinama lygybė

$$g_1^{a_1} g_2^{a_2} \ldots g_k^{a_k} \equiv h \pmod p$$

Indeksų vektorius $\{a_1, \ldots, a_k\}$ yra vadinamas reprezentacija. Priimama sąlygą $a_i \neq 0$, nes priešingu atveju $g_i^0 \equiv 1 \pmod p$. Indeksų vektoriaus apskaičiavimas suvedamas į tokį uždavinį:

$$a_1 \log g_1 + a_2 \log g_2 + \ldots + a_k \log g_k \equiv \log h \pmod p$$

Tarkime turime k elementų reprezentacijoje $\{a_1, \ldots, a_k\}$. Tuomet egzistuoja q^{k-1} skaičiaus h reprezentacijos.

Nulinio atskleidimo įrodymas

Reprezentacijos $\{a_1, a_2, \ldots, a_k\}$ neįmanoma apskaičiuoti jei parametrai pakankamai dideli. Tarkime vartotojas A žino $h \equiv g_1^{a_1} g_2^{a_2} \ldots g_k^{a_k} \pmod p$ reprezentaciją $\{a_1, a_2, \ldots, a_k\}$, kai duota $\{g_1, g_2, \ldots, g_k\}$

ir nori tai įrodyti vartotojui B, neatskleidžiant žinomos reprezentacijos. Žemiau pateiktas nulinio atskleidimo įrodymo (angl. zero knowledge proof) protokolas pagal Brands (1993).

1. A žino $h \equiv g_1^{a_1} g_2^{a_2} \dots g_k^{a_k} \pmod p$ ir $\{a_1, a_2, \dots, a_k\}$, tikrintojas B žino h ir $\{a_1, a_2, \dots, a_k\}$.
2. A sugeneruoja k elementų atsitiktini vektorių $\{w_1, w_2, \dots, w_k\} \in G_q$.
3. A apskaičiuoja $z \equiv g_1^{w_1} g_2^{w_2} \dots g_k^{w_k} \pmod p$ ir nusiunčia šį skaičių B.
4. B sugeneruoja atsitiktinį skaičių c ir nusiunčia vartotojui A.
5. A apskaičiuoja $\{r_1, r_2, \dots, r_k\}$, kur $r_i \equiv a_i + cw_i \pmod q$ ir nusiunčia šį vektorių tikrintojui B.
6. B patikrina ar tenkinama lygybė $zh^c \equiv g_1^{r_1} g_2^{r_2} \dots g_k^{r_k} \pmod p$.

Apribojantis aklasis parašas

Aklasis parašas yra naudojamas tuomet, kai norima pasirašyti dokumentą jo neatskleidžiant pasirašančiam asmeniui. Apribotas aklasis parašas (angl. restrictive blind signature) skiriasi nuo paprasto aklojo parašo tuo, kad leidžia pasirašančiam asmeniui prijungti papildomą informaciją, pvz. galiojimo laiką. Šią schemą pateikė Brands (1993).

Kaip ir Schnorr autentikacijos ir elektroninio parašo schemoje veiksmai atliekami lauke$\langle \mathbb{Z}_p, +, * \rangle$. Parenkame tokius pirminius skaičius p ir q, kad būtų tenkinama $p = kq + 1$, randame G_q generatorių g. Vartotojas sugeneruoja privatųjį raktą $x \in \{1, 2, \dots, q\}$ ir apskaičiuoja viešąjį raktą $h \equiv g^x \pmod p$. Parametrai h, g, q, p yra paviešinami. Žemiau pateikiamas pasirašymo protokolas, kuomet pasirašoma tekstograma M.

1. B nusiunčia tekstograma M vartotojui A.
2. A sugeneruoja atsitiktini skaičių w ir apskaičiuoja parametrus $z \equiv M^x \pmod p$, $a \equiv g^w \pmod p$, $b = M^w \pmod p$.
3. B sugeneruoja skaičių c ir nusiunčia vartotojui A. Vartotojas B sugeneruoja keturis atsitiktinius skaičius s, t ir u, v. Naudodamas s ir t vartotojas užmaskuoja tekstogramą M ir parašą z:

$$M' \equiv M^s g^t \pmod p$$

$$z' \equiv z^s h^t \equiv (M^x)^s (g^x)^t \equiv (M^s g^t)^x \equiv M'^x \pmod p$$

Tokiu būdu vartotojas B gauna vartotojo A parašą ant pakeistos žinutės, kurios vartotojas A nežino.

Naudojant u ir v apskaičiuojami pakeitimai parametrams a ir b.

$$a' \equiv a^u g^v \equiv (g^w)^u \, g^v \equiv g^{w'} \pmod p$$

$$b' \equiv (a^{ut}) \, (b^{us}) \, M'^v \equiv \left(g^{w^{ut}}\right) \left(M^{w^{us}}\right) M'^v \equiv \left(g^{t^{uw}}\right) \left(M^{s^{uw}}\right) M'^v \pmod p \equiv \ldots$$

$$\ldots \equiv M'^{uw} M'^v M'^{w'} \pmod p$$

B apskaičiuoja komponentą $c' = H\,(M', z', a', b')$ ir $c \equiv c' u^{-1} \pmod q$ ir nusiunčia vartotojui A.

4. A apskaičiuoja atsakymą $r \equiv w + cx \pmod q$ ir nusiunčia vartotojui B.
5. B turėdamas c ir r patikrina $ah^c \equiv g^r \pmod p$ ir $bz^c \equiv M^r \pmod p$.

Šios lygybės patvirtina, kad parašas padėtas ant tekstogramos M priklauso vartotojui A. Protokolas labai svarbus Stephan Brands elektroninių pinigų modelyje. Šio protokolo pagalba įgyvendinamas pinigų anonimiškumas, nes bankas pasirašo ant elektroninio pinigo nematydamas galutinio pinigo, kuris cirkuliuos elektroninių pinigų sistemoje.

2.6 STEFAN BRANDS ELEKTRONINIŲ PINIGŲ MODELIS

Stephan Brands elektroninių pinigų modelis yra sudarytas naudojant kriptografines schemas ir modelius aptartus ankstesniuose skyriuose. Šių metodų pagalba konstruojami paėmimo, mokėjimo, inkasavimo protokolai, taip pat užtikrinamas anonimiškumas ir dvigubo išleidimo prevencija. Elektroninių pinigų cirkuliacijos dalyviai yra bankas B, pirkėjas P ir verslininkas V. Taip pat sistemos inicijavimo metu gali dalyvauti ir neutrali šalis, kuri nustato sistemos parametrus. Žemiau pateiktas elektroninių pinigų modelis, pagal Brands (1993).

Sistemos inicijavimas

Veiksmai atliekami lauke $\langle \mathbb{Z}_p, +, * \rangle$, o parametrai p ir q parenkami taip kaip Schnorr protokole. Apskaičiuojami generatoriai g, g_1, g_2. Taip pat parenkamos dvi vienkryptės funkcijos: 5-ių kintamųjų H ir 4-ių kintamųjų H_0.

Bankas atsitiktinai pasirenka savo privatųjį raktą $x \in G_q$ ir apskaičiuoja viešąjį raktą

$$h \equiv g^x \pmod p \tag{2.1}$$

Banko viešasis raktas h paviešinamas. Banko išduodamas pinigas yra tam tikra, pasirašyta, skaitmeninė informacija. Šioje sistemoje darysime prielaidą, kad visi pinigai yra vienodos vertės. Pinigas yra aprašomas skaičiais A, B, o parašas ant šių skaičių $\{z', a', b', r'\}$.

Pirkėjo sąskaitos atidarymo protokolas

Pirkėjas, norėdamas atsidaryti sąskaitą banke, turi pasirinkti savo privatųjį raktą u_1 ir apskaičiuoti identifikacijos numerį h_{u_1}.

$$h_{u_1} \equiv g_1^{h_{u_1}} \ (mod\ p) \tag{2.2}$$

Šį identifikacijos numerį vartotojas siunčia bankui ir bankas jį išsaugoja duomenų bazėje kartu su kita vartotojo informacija (vardu, pavarde, adresu, telefono numeriu ir kita). Bankas apskaičiuoja ir nusiunčia pirkėjui parametrą z.

$$z \equiv (h_{u_1} g_2)^x \ (mod\ p) \tag{2.3}$$

Verslininkui užtenka užregistruoti identifikacijos numerį banke.

Pinigo paėmimo protokolas

Pinigo paėmimo protokolo pradžioje, vartotojas turi nusiųsti užklausą bankui, prašydamas elektroninio pinigo. Užklausa turi būti siunčiama su parašu, kad bankas būtų tikras vartotojo autentiškumu. Šio protokolo metu sugeneruojami skaičiai, aprašantys elektroninį pinigą.

Bankas pasirenka atsitiktinį skaičių w, apskaičiuoja ir nusiunčia pirkėjui skaičius

$$a \equiv g^w \ (mod\ p)\,,\ b \equiv (h_{u_1} g_2)^w \ (mod\ p)\,. \tag{2.4}$$

Pirkėjas sugeneruoja penkis atsitiktinius skaičius $s, x_1, x_2, u, v \in G_q$. Naudodamas parametrus s, x_1, x_2 pirkėjas apskaičiuoja elementus

$$A \equiv (Ig_2)^s \ (mod\ p)\,,\ B \equiv g_1^{x_1} g_2^{x_2} \ (mod\ p)\,,\ z' \equiv z^s \ (mod\ p)\,. \tag{2.5}$$

Šiuo atveju z' yra parašas ant A, nes $z' \equiv z^s \equiv (Ig_2)^{x^s} \equiv A^x \ (mod\ p)$. Naudodamas u ir v pirkėjas apskaičiuoja elementus

$$a' \equiv a^u g^v \ (mod\ p)\,,\ b' \equiv b^{su} A^v \ (mod\ p) \tag{2.6}$$

Pirkėjas taip pat apskaičiuoja c', jį užmaskuoja ir nusiunčia bankui.

$$c' = H\left(A, B, z', a', b'\right),\ c \equiv c' u^{-1} \ (mod\ p) \tag{2.7}$$

Bankas apskaičiuoja atsakymą r ir nusiunčia jį pirkėjui. Bankas taip pat nurašo vieną pinigą iš pirkėjo sąskaitos.

$$r \equiv w + cx \ \ (mod\ q) \tag{2.8}$$

Vartotojas sutinka su pinigų nuskaitymu tik tada, jei tenkinamos lygybės $g^r \equiv ah^c \ \ (mod\ p)$ ir $(h_{u_1} g_2)^r \equiv bz^c$. Vartotojas taip pat apskaičiuoja

$$r' \equiv v + ru \ \ (mod\ p) \tag{2.9}$$

Tokiu būdu sukonstruojamas pinigas $\{A, B, z', a', b', r'\}$

Pinigo mokėjimo protokolas

Mokėjimo protokolas vyksta tarp pirkėjo P ir verslininko V.

Pirkėjas nusiunčia elektroninį pinigą $\{A, B, z', a', b', r'\}$ verslininkui.

Verslininkas nusiunčia išbandymą

$$d \equiv H_0\left(A, B, M, t\right) \ \ (mod\ p) \tag{2.10}$$

pirkėjui. Čia M yra verslininko identifikacijos numeris, o t yra skaičius nurodantis laiką ir datą.

Pirkėjas apskaičiuoja atsakymus

$$r_1 \equiv du_1 s + x_1 \ \ (mod\ q)\,,\ r_2 = ds + x_2 \ \ (mod\ q) \tag{2.11}$$

ir nusiunčia verslininkui. Parametras u_1 yra vartotojo slaptasis raktas, o s, x_1, x_2 yra paėmimo protokolo metu pasirinkti atsitiktiniai skaičiai.

Verslininkas patikrina lygybes

$$A^d B \equiv g_1^{r_1} g_2^{r_2} \ \ (mod\ p)\,,\ g^{r'} \equiv a' h^{c'} \ \ (mod\ p)\,,\ A^{r'} \equiv b' z'^{c'} \ \ (mod\ p) \tag{2.12}$$

ir, jeigu lygybės tenkinamos, verslininkas priima elektroninį pinigą.

Pinigo inkasavimo protokolas

Verslininkas nusiunčia bankui elektroninį pinigą $\{A, B, z', a', b', r'\}$ kartu su $\{r_1, r_2, d\}$ ir pinigų pervedimo laiku t.

1. Bankas patikrina ar tenkinamos lygybės 2.12.

2. Bankas patikrina ar duomenų bazėje jau egzistuoja pinigas $\{A, B, z', a', b', r'\}$. Jeigu pinigas randamas duomenų bazėje, tuomet bankas imasi veiksmų, kad atskleistų sukčiaujantį asmenį.

Jei pirma ir antra sąlygos tenkinamos, į verslininko sąskaitą įskaičiuojama pinigo vertė, o duomenys apie pinigą įrašomi duomenų bazėje.

Jeigu pinigas jau yra duomenų bazėje, tuomet bankas tikrina:

1. Jei elektroninio pinigo pervedimo laikas sutampa, tuomet verslininkas bando pinigą pervesti antrą kartą.
2. Priešingu atveju pirkėjas elektroninį pinigą išleido daugiau negu vieną kartą, tuomet bankas gali identifikuoti pirkėją.

Bankas žino suklastotų pinigų parametrus $\{A, B, t, r_1, r_2\}$ ir $\{A, B, t, r_1', r_2'\}$. Naudodamas šiuos parametrus bankas gali atskleisti vartotojo tapatybę tokiu būdu:

$$(r_1 - r_1')\,(r_2 - r_2')^{-1} \equiv (d\,(u_1 s) - d'\,(u_1 s))\,(ds - d's)^{-1} \equiv u_1 \pmod q \tag{2.13}$$

Turėdamas vartotojo privatųjį raktą, bankas gali apskaičiuoti pirkėjo viešąjį raktą h_u ir susieti jį su pirkėju duomenų bazėje.

2.7 NOKIA 6212 MOBILUSIS TELEFONAS

Šiame skyriuje aprašytas programų mobiliesiems telefonams kūrimas naudojant Java ME platformą ir nagrinėjama Nokia 6212 Classic telefono architektūra bei galimybės. Skyriuje bus aprašomi pagrindiniai šio telefono moduliai, ryšio priemonės, duomenų talpinimo galimybės, palaikomos bibliotekos. Taip pat pateikiamas trumpas saugaus elemento aprašymas. Nokia 6212 Classic yra S40 šeimos telefonas išleistas 2008 metais. Be įprasto funkcionalumo, kaip Bluetooth ryšys, Java ME programinė įranga, 3G ryšys ir kt., telefone taip pat palaikoma artimo lauko komunikacija NFC (angl. Near Field Communication) - vienas naujausių ir perspektyviausių ryšio standartų.

Java ME platforma

Java šiuo metu yra plačiai naudojama programavimo kalba. Viena didžiausių šios kalbos populiarumo priežasčių yra tai, kad galima kurti programinę įrangą skirtingoms operacinėms sistemoms ir kompiuterių architektūroms. Šiuo metu pagrindinės Java kalbos platformos yra šios:

- Java Standard Edition (SE)
- Java Enterprise Edition (EE)
- Java Micro Edition (ME)

2.1 pav.: Mobilusis telefonas Nokia 6212 Classic

- Java Card

Nokia 6212 Classic telefone yra įdiegta Java Micro Edition platforma. Ši platforma palaiko tik dalį Java Standard Edition platformos aplikacijų programavimo interfeiso API (angl. application programing interface), bei naudoja skirtingą programų diegimo procedūrą. Norint įdiegti programą į mobilųjį telefoną, reikia sukurti specialų MIDLET paketą, kuriame talpinama pati programa, bei jai reikalingi resursai.

Telefone yra realizuoti standartinis Java ME API - CLDC (angl. Connected Limited Device Configuration) bei MIDP (angl. Mobile Information Device Profile). Be šių bibliotekų Nokia 6212 Classic palaiko PIM (angl. Personal Information Management), FileConnection API, Contactless Communication API, Contactless Communication Extension API, Record Management Store. CLDC įgyvendina minimalią Java sistemą:

- Java kalba
- Java kalbos pagrindinės bibliotekos
- Įvedimo ir išvedimo bibliotekos
- Saugumas
- Tinklo bibliotekos

MIDP yra CLDC papildymas, skirtas mobiliesiems telefonams, kuriuose realizuotos bibliotekos ir funkcijos skirtos programos paruošimui ir diegimui, išplėstos tinklo bibliotekų galimybės, pastovioji laikmena (angl. Record Management System), garso, vartotojo sąsajos ir kitos bibliotekos.

Duomenų saugojimo laikmenos

Elektroninių pinigų sistemos realizacijoje svarbu parinkti laikmeną telefone pinigams ir slaptiems parametrams (elektroniniam parašui) saugoti. NOKIA 6212 Classic telefone duomenis galima saugoti:

- telefono vidinėje dokumentų sistemoje
- mobilioje atminties kortelėje telefone
- įrašų valdymo sistemoje (RMS)
- saugiajame elemente (angl. Secure Elemente)

Kadangi elektroninis pinigas, nežinant vartotojo privataus rakto, negali būti išleistas, nekeliame reikalavimo elektroninius pinigus saugoti saugiąjame elemente. Elektroninių pinigų saugojimui buvo pasirinkta telefono vidinė failų sistema. Šį pasirinkimą nulėmė tai, kad talpos vidinėje atmintyje užtenka dideliam pinigų kiekiui.

Tam, kad pametus arba vagystės atveju, užtikrinti privataus rakto saugumą, raktas kartu su visa slapta informacija talpinamas saugiąjame elemente.

Saugusis elementas

Telefone, kartu su NFC moduliu, yra įdiegtas saugusis elementas (angl. Secure Element), kuris gali būti panaudotas, kaip:

- NFC mikroprocesorinių kortelių skaitytuvas
- Mikroprocesorinė kortelė

Realizuojant elektroninių pinigų sistemą bus reikalingas saugaus elemento, kaip mikroprocesorinė kortelė, funkcionalumas. Šioje kortelėje yra įdiegta Java Card platforma. Platesnė informacija apie mikroprocesorinę kortelę bei Java Card platformą pateikta kituose skyriuose.

Patikimi paslaugų valdytojai

Mikroprocesorinėse kortelėse talpinami duomenys būna slapti, todėl yra apribotas priėjimas prie šių kortelių. Norint įdiegti papildomas programas, tai galima padaryti:

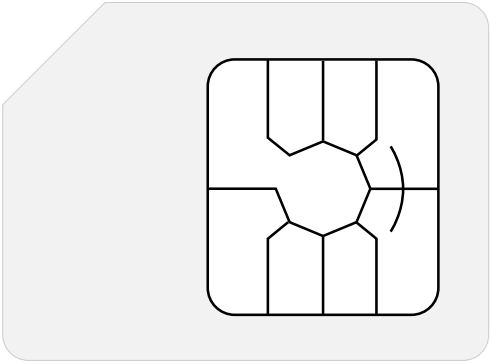
1. per patikimą paslaugų valdytoją (angl. Trusted Service Manager)
2. atrakinus kortelę ir gavus gamyklinius raktus, su kuriais galima prieiti prie mikroprocesorinės kortelės.

Patikimi paslaugų valdytojai yra organizacijos, kurių tikslas padėti paslaugų tiekėjams saugiau paskirstyti ir valdyti programas naudojant mobiliojo ryšio operatorių tinklus. Šiam darbui pasirinktas antras variantas, kurio pagalba, galima diegti ir testuoti programas savarankiškai. Atrakinimo trūkumas yra tai, kad atrakinus mikroprocesorinę kortelę, ji tampa nepatikima ir ateityje patikimi paslaugų valdytojai į ją negalės įdiegti naujų programų.

2.8 MIKROPROCESORINĖS KORTELĖS

Šiame skyriuje pateikta bendra informacija apie mikroprocesorines korteles (angl. Smart Card), jų funkcionalumą ir parametrus. Mikroprocesorinės kortelės yra nedideli įrenginiai, atsparūs klastojimui ir neautorizuotam modifikavimui, todėl gali būti panaudoti identifikacijai, autentikacijai, duomenų talpinimui ir programoms vykdyti, kur reikalaujamas didelis saugumas.

Šiuo metu mikroprocesorinės kortelės daugiausiai naudojamos bankiniame sektoriuje, pvz. kreditinės arba debetinės kortelės, ir mobiliuosiuose telefonuose, pvz. SIM (angl. Subscriber Identity Modules). Daugelis identifikacijos ir autentikacijos sistemų taip pat naudoja mikroprocesorines korteles. Vienas naujausių šio pritaikymo pavyzdžių - Lietuvoje ir daugelyje kitų šalių pradėtos diegti asmens tapatybės kortelės su mikroprocesorine kortele.



2.2 pav.: Kontaktinės mikroprocesorinės kortelės pavyzdys

Techninė įranga

Standartinėse mikroprocesorinės kortelėse dažniausiai būna įdiegtas nuo 4 iki 6 MHz dažnio procesorius. Dauguma modernių kortelių taip pat turi papildomą procesorių, skirtą kriptografinėms funkcijoms atlikti.

Mikroprocesorinės kortelės atmintis būna trijų tipų:

1. Pastovioji atmintis (ROM)
2. Elektroniškai ištrinama programuojama pastovioji atmintis (EEPROM)
3. Operatyvinė atmintis (RAM)

Tam, kad būtų išsaugoti duomenys, ROM atminties tipui nėra reikalingas energijos šaltinis. Šiame atminties modulyje laikomi gamykloje įdiegti duomenys ir modifikuoti jos turinį nėra galimybės. EEPROM atminties moduliui, kaip ir ROM, nereikalingas papildomas energijos šaltinis. Šis atminties modelis nuo ROM modelio skiriasi tuo, kad galima modifikuoti jos turinį įdiegiant naujas programas ir talpinant duomenis. EEPROM atminties trūkumas yra egzistuojantis ribotas rašymo ciklų skaičius. RAM yra nepastovioji atmintis, skirta laikiniems duomenims saugoti, programos vykdymo metu.

Komunikacijos modelis (APDU)

Mikroprocesorinės kortelės nefunkcionuoja savarankiškai ir visada yra didesnės sistemos dalis, todėl didelę svarbą turi duomenų apsikeitimas tarp mikroprocesorinių kortelių ir jų skaitytuvų. Mikroprocesorinės kortelės dirba kliento - serverio režimu, kur duomenų apsikeitimą visada inicijuoja serveris, o kortelė atlieka kliento vaidmenį. Tokio duomenų apsikeitimo pavyzdys galėtų būti banko kortelė ir bankomatas.

Duomenų apsikeitimui naudojami APDU (angl. Application Protocol Data Unit) duomenų paketai, apibrėžti ISO/IEC 7816-4 standarte. APDU paketai gali būti dviejų tipų - instrukcija ir atsakas. Instrukcija yra siunčiama serverio programos į mikroprocesorinę kortelę, kortelėje instrukcija apdorojama ir grąžinamas atsakas. APDU komandas sudaro baitų masyvai, kurie pateikti žemiau.

2.2 lentelė: Duomenų apsikeitimo APDU instrukcijos struktūra

Antraštė				Turinys		
CLA	INS	P1	P2	Lc	Duomenys	Le

2.3 lentelė: Duomenų apsikeitimo APDU atsako struktūra

Turinys	Priesaga	
Duomenys	SW1	SW2

APDU duomenų paketų blokų reikšmės pateiktos žemiau:

- Duomenys - apsikeitimo metu perduodami duomenys.
- CLA - baitas, nurodantis klasę.
- INS - baitas, nurodantis instrukciją.
- P1,P2 - parametrų baitai.
- Lc - duomenų ilgis instrukcijos komandoje.
- Le - duomenų ilgis, kurio tikimasi iš atsako komandos.
- SW1 - atsako komandos būsenos baitai.

Mikroprocesorinių kortelių skaitytuvas

Duomenų apsikeitimas naudojant APDU duomenų blokus realizuojamas naudojant bevielį mikroprocesorinių kortelių skaitytuvą ACR ACS 122. Šio skaitytuvo pagalba galima diegti programas ir atlikti duomenų apsikeitimus su telefone esančia mikroprocesorine kortele. Atrakinus telefone esantį saugųjį elementą gaunami gamykliniai telefono raktai, kurie reikalingi norint įdiegti naujas programas. Naujų programų diegimas atliekamas naudojant GlobalPlatform programinį įrankį GPShell, kurio pagalba galima siųsti APDU duomenų paketus.

2.3 pav.: Bevielis mikroprocesorinių kortelių skaitytuvas ACR ACS 122

2.9 JAVA CARD STANDARTAS

Java Card yra Java kalbos versija, kuri leidžia Java kalba parašytas programas vykdyti ribotų resursų sistemose, tokiose kaip mikroprocesorinės kortelės. Ši platforma plačiai naudojama mobiliųjų telefonų SIM kortelėse ir banko kreditinėse bei debetinėse kortelėse. Java Card platforma leidžia rašyti objektiškai orientuotas programas, kurios palaikomos įvairiose mikroprocesorinėse kortelėse, todėl nereikia kiekvienai kortelei rašyti atskiros programos.

Programa veikianti Java Card platformoje yra vadinama apletu (angl. applet), o tokių apletų rinkinys yra vadinamas paketu. Kiekvienas apletas ir paketas turi unikalius identifikacijos numerius AID (angl. Application Identifier). Šio numerio pagalba galima trinti, diegti ir pasirinkti darbui atskirus apletus ir paketus. Java Card technologija nuo pirmos dienos buvo kuriama suteikiant didelę svarbą saugumui ir duomenų apsaugai. Kiekvienas apletas ir jam priklausantys duomenys yra atskirtas ugniasiene nuo kitų apletų ir izoliuotas nuo sistemos branduolio. Duomenys gali būti nuskaitomi ir perduodami tik per tam

tikslui sukurtus specializuotus metodus t.y. nėra tiesioginio priėjimo prie duomenų Java Card sistemoje.

Kad galėtų veikti ant ribotų resursų sistemose, Java Card palaiko tik dalį Java kalbos:

- Palaikomi tik boolean, byte, short kintamųjų tipai.
- Nepalaikomi daugiamačiai masyvai.
- Nepalaikomas dinaminis klasių užkrovimas, šiukšlių surinkėjas (angl. Garbage Collector), objektų serializacija ir klonavimas.
- Palaikomi paketai, klasės, interfeisai, paveldėjimas, virtualūs metodai, metodų perdengimas, išimčių valdymas.

Java Card platformos programos yra pirmiausia sukompiliuojamos ir sukuriami 'class' failai, kaip ir naudojant standartinę Java SE platformą. Tuomet klasių failai yra konvertuojami ir sukuriamas .CAP failas, kuris apjungia visas klases. Konvertavimo tikslas yra patikrinti klasių tinkamumą ir optimizavus kodą, sukurti paketą, tinkantį Java Card platformai. Tuomet .CAP failas yra patalpinamas į Java Card kortelę ir gali būti naudojamas Java Card interpretatoriaus.

Dėl apribotų resursų sudėtingi kriptografiniai algoritmai yra realizuoti specialiai tam paruoštame kriptografiniame procesoriuje. Java Card yra realizuotas API skirtas darbui su kriptografinėmis funkcijomis ir modeliais. Darbui su kriptografiniais metodais klases galima rasti javacard.security ir javacardx.crypto paketuose.

- KeyBuilder - klasė skirta kriptografinių raktų generavimui.
- KeyPair - klasė skirta privataus ir viešo rakto poroms saugoti.
- MessageDigest - kriptografinės vienkryptės funkcijos.
- RandomData - klasė skirta atsitiktiniams skaičiams generuoti.
- Signature - DSA, RSA ir kitus algoritmus palakanti klasė, leidžianti pasirašyti duomenis.
- Cipher - RSA, AES ir kitus algoritmus palaikanti šifravimo klasė.

Java Card platformos trūkumas yra tai, kad nėra galimybės prieiti prie kriptografinio procesoriaus tiesiogiai, norint atlikti tokius veiksmus kaip modulinė daugyba ir modulinis eksponentiavimas. Dėl to labai apsunkinamas naujų kriptografinių modelių ir schemų diegimas į Java Card mikroprocesorines korteles.

2.10 ELEKTRONINIŲ PINIGŲ REALIZACIJOS IR TYRIMO REIKALAVIMAI

Tiriamojo darbo metu, Stefan Brands elektroninių pinigų modelio realizacija bus atliekama trimis etapais:

1. Modelio realizacija Java SE platformoje.
2. Modelio realizacija Jave ME platformoje.
3. Kritinio saugumo funkcijų realizacija Java Card platformoje.

Modelio realizacija Java SE platformoje

Šioje realizacijoje vartotojo, banko ir verslininko dalys bus realizuotos naudojant Java SE platformą. Darbui su dideliais skaičiais bus naudojama BigInteger biblioteka.

Modelio realizacija Jave ME platformoje

Šioje realizacijoje, tariama, kad vartotojo privatus ir viešas raktai laikomi laisvai prieinamoje atminties dalyje, todėl pametus telefoną, elektroniniai pinigai gali būti išleisti neautorizuotų asmenų. Darbui su dideliais skaičiais bus naudojama Bouncy Castle BigInteger realizacija. Svarbu patikrinti šios realizacijos greitį, kuomet operuojami skaičiai bus 256 arba 512 bitų, kaip yra rekomenduojama Schnorr autentikacijos ir parašo schemose. Svarbus dėmesys bus skiriamas vartotojo ir verslininko dalies realizacijos greičiui, nes banko dalis visuomet bus atliekama specialiai tam dedikuotose sistemose, kaip bankomatai arba interneto serveriai, todėl bus naudojama Java SE platforma ir našesnė techninė įranga.

Kritinio saugumo funkcijų realizacija Java Card platformoje

Šio etapo metu bus atliekama kritinio saugumo funkcijų realizacija Java Card platformoje. Šioje dalyje daugiausiai dėmesio bus skiriama vartotojo protokolo daliai, dėl:

- Elektroninių pinigų realizacijos greičio.
- Elektroninių pinigų realizacijos saugumo.

Kadangi mokėjimai atliekami parduotuvėse ir panašiose įstaigose, kur susidaro eilės, labai svarbu, kad mokėjimai būtų atliekami kuo sparčiau. Taip pat labai svarbu, kad elektroninių pinigų realizacija būtų saugi. Pametus arba telefono vagystės atveju, turi nebūti būdų išleisti elektroninius pinigus. Tai galima užtikrinti tik vienu būdu - vartotojo kriptografiniai privatieji raktai turi būti saugomi saugioje atmintyje. Reikia ištirti Stefan Brands modelio vartotojo pusės realizacijos galimybes mikroprocesorinėje kortelėje.

Šio etapo metu bus panaudotas mikroprocesorinių kortelių skaitytuvas, leisiantis kritinio saugumo funkcijas įdiegti mikroprocesorinėje kortelėje. Java Card apletų diegimui mikroprocesorinėje kortelėje bus naudojama GPShell programa, kurios pagalba bus automatizuotas diegimas ir testavimas.

Pradiniam realizacijos testavimui bus pasitelkiamas JCWDE (angl. Java Card Workstation Development Environment) įrankis, leidžiantis emuliuoti telefone esančią mikroprocesorinę kortelę ir tokiu būdu pagreitinti testavimą.

Aritmetinių operacijų realizacija Java Card platformoje

Kadangi Java Card platforma neturi bibliotekos darbui su dideliais skaičiais, ši biblioteka turi būti realizuota savarankiškai. Stefan Brands elektroninių pinigų sistemoje naudojamos aritmetinės operacijos:

- Sudėtis (+)
- Skirtumas (-)
- Modulis (mod)
- Modulinė eksponentė (modPow)
- Sandauga (*)
- Vieno bito postūmis į dešinę (>>)

Skaičiai turi būti saugomi baitų masyvuose. Ankščiau išvardintas operacijas, kaip ir operacijos rezultatą, reikia realizuoti tarp baitų masyvų. Visos operacijos turi būti atliekamos baigtiniame lauke $\mathbb{Z}_p$. Sukurtos operacijos turi būti testuojamos su įvairiais skaičiais.

3 ELEKTRONINIŲ PINIGŲ REALIZACIJA

3.1 AUTOMATINIS MODELIO PARAMETRŲ GENERAVIMAS

Kadangi modelis bus testuojamas su įvairaus dydžio skaitmenimis, sukurta programa, kurios pagalba turint tam tikrą skaičių e, automatiškai sugeneruojami Stefan Brands elektroninių pinigų sistemos parametrai p, q, g, g_1, g_2, x, h, u_1, h_{u_1}, z.

findPQ(BigInteger e)

Metodas ieško didesnių už e pirminių skaičių p, kol tenkinama lygybė $p = 2q + 1$ ir q yra pirminis skaičius. Metodas gražina modelio parametrus p ir q.

findG()

Metodas atsitiktinai parenka skaičius ir skaičių priima kaip generatorių, jeigu tenkinamos lygybės $g^2 \equiv 1 \ (mod\ p)$ ir $g^q \equiv 1 \ (mod\ p)$. Metodas gražina generatorių g.

generateBankKeys()

Atsitiktinai sugeneruojamas skaičius $x \in G_q$ ir pagal 2.1 apskaičiuojamas banko viešasis raktas h.

generateUserKeys()

Atsitiktinai sugeneruojamas skaičius $u_1 \in G_q$ ir pagal 2.2 bei 2.3 apskaičiuojami vartotojo privatus raktas h_{u_1} ir komponentė z.

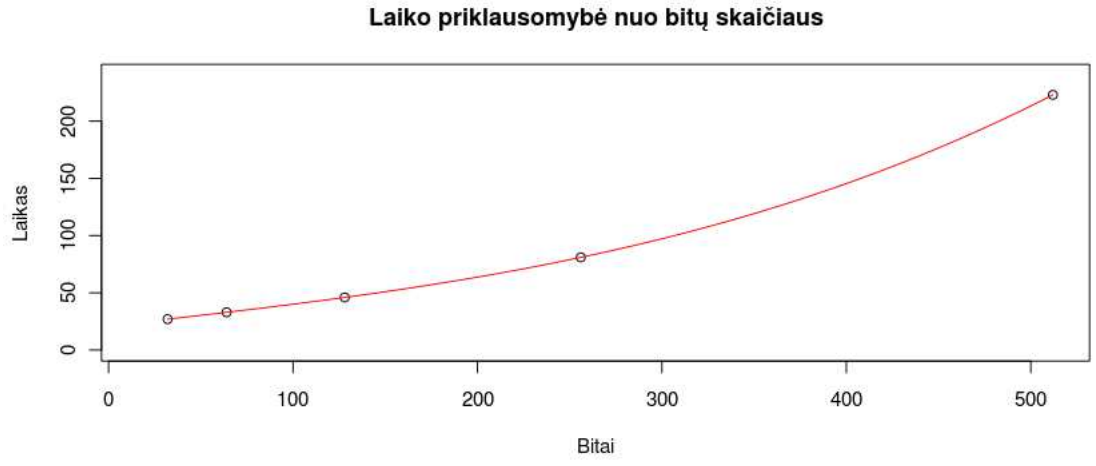
Kadangi skaičius Java ME ir Java Card platformose saugosime skirtingu formatu, tai sugeneruoti parametrai išvedami BigInteger ir baitų masyvų formatais. Žemiau pateiktas sistemos parametrų generavimo programos išvedamas rezultatas, kuomet parenkami 32 bitų parametrai:

```
BigInteger p = new BigInteger("4294967387");
BigInteger q = new BigInteger("2147483693");
BigInteger g = new BigInteger("2148750335");
BigInteger g1 = new BigInteger("884991541");
BigInteger g2 = new BigInteger("2094192099");
BigInteger x = new BigInteger("79223638");
BigInteger h = new BigInteger("4122967090");
BigInteger u1 = new BigInteger("250511266");
BigInteger hu1 = new BigInteger("3755315433");
BigInteger z = new BigInteger("1897390270");
```

```
byte[] p = { (byte) 0x01, (byte) 0x00, (byte) 0x00, (byte) 0x00, (byte) 0
   x5b, };
byte[] q = { (byte) 0x80, (byte) 0x00, (byte) 0x00, (byte) 0x2d, };
byte[] g = { (byte) 0x80, (byte) 0x13, (byte) 0x53, (byte) 0xff, };
byte[] g1 = { (byte) 0x34, (byte) 0xbf, (byte) 0xe6, (byte) 0x35, };
byte[] g2 = { (byte) 0x7c, (byte) 0xd2, (byte) 0xd5, (byte) 0xe3, };
byte[] x = { (byte) 0x04, (byte) 0xb8, (byte) 0xdb, (byte) 0x56, };
byte[] h = { (byte) 0xf5, (byte) 0xbf, (byte) 0x7c, (byte) 0x32, };
byte[] u1 = { (byte) 0x0e, (byte) 0xee, (byte) 0x7f, (byte) 0xa2, };
byte[] hu1 = { (byte) 0xdf, (byte) 0xd5, (byte) 0x90, (byte) 0xe9, };
byte[] z = { (byte) 0x71, (byte) 0x17, (byte) 0xe0, (byte) 0xbe, };
```

3.2 MODELIO REALIZACIJA JAVA SE PLATFORMOJE

Stefan Brands elektroninių pinigų modelis Java SE platformoje realizuotas Bouncy Castle bibliotekos pagalba. Sukurti metodai visiems elektroninių pinigų modelio cirkuliacijos subjektams (bankui, vartotojui, verslininkui) ir apskaičiuotas laikas, reikalingas visai sistemai bei atskiriems žingsniams vykdyti. Skaičiavimai atlikti prie 32, 64, 128, 256 ir 512 bitų parametrų. Darbui su dideliais skaičiais naudojama BigInteger biblioteka. Žemiau pateikti rezultatai, gauti testuojant bendrą sistemos laiką ir atskirai vartotojo bei pardavėjo laikus. Galima pastebėti, kad laikas, reikalingas protokolų įvykdymui Java SE platformoje, testavimui naudojant AMD Athlon 3000+ kompiuterį, yra trumpas visų protkolų atžvilgiu.



3.1 pav.: Bendra realizacijos laiko priklausomybė nuo bitų skaičiaus J2SE platformoje

3.2 pav.: Verslininko realizacijos laiko priklausomybė nuo bitų skaičiaus J2SE platformoje

3.3 pav.: Vartotojo realizacijos laiko priklausomybė nuo bitų skaičiaus J2SE platformoje

3.3 PROGRAMŲ JAVA ME IR JAVA CARD PLATFORMOSE KONSTRAVIMAS IR DIEGIMAS

Norint sistemą testuoti Jave ME ir Java Card platformose, reikia automatizuoti dažnai pasikartojančius procesus, kaip programos kompiliavimas, tikrinimas, paketo kūrimas, pasirašymas ir įkėlimas į telefono atmintį arba į mikroprocesorinę kortelę. Šiam tikslui buvo sukurti arba panaudoti įvairūs įrankiai, leidžiantys daugelį procesų atlikti automatiškai, be vartotojo įsikišimo.

Paketų Java ME platformai konstravimas ir diegimas

Automatiniam programų kompiliavimui ir galutiniam paketo sukūrimui buvo pasitelkta ANT sistema. Buvo sukurtas programos kompiliavimo, patikrinimo ir pasirašymo instrukcijų failas, kuris vykdomas naudojant ANT sistemą. Galimos instrukcijos yra: init, compile, dist, preverify, deploy, sign, upload. Šios sistemos pagalba sutaupomas brangus laikas, reikalingas pasikartojančioms procedūroms

atlikti. Sukuriamas paketas į telefoną įkeliamas naudojant Bluetooth ryšį.

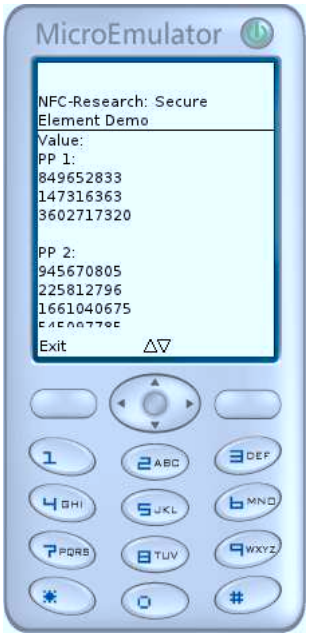
Norint iš telefono pasiekti mikroprocesorinę kortelę, J2ME apletas turi būti pasirašytas įgaliotos organizacijos. Norint pasirašyti reikalingas sertifikatas. Šių sertifikatų kaina rinkoje siekia daugiau nei 1500 Lt, bet buvo rastas metodas telefone įdiegti savo sertifikatą ir vėliau jį naudoti pasirašinėjant programas. Daugiau informacijos apie sertifikato sukūrimą ir diegimą telefone rasite Dwarf (2006).

Testuojant sistemos realizacijos korektiškumą buvo naudojamas telefono emuliatorius 3.4. Žemiau pateiktas automatinės paketo paruošimo sistemos išvedamas tekstas:

```
Buildfile: /home/paulius/Dropbox/Mokslai/Magistrinis/Code/me/build.xml
init:
compile:
dist:
[jar] Building jar: /home/..dist/brandsMeTest-0.1.jar
preverify:
[proguard] ProGuard, version 4.3
[proguard] Reading program jar [/home/..lib/bc.jar]
...
[proguard] Copying resources from program jar [/home/../lib/bc.jar]
...
deploy:
[copy] Copying 1 file to /home/paulius/Dropbox/Mokslai/Magistrinis/Code/me/
   deployed
[jad] Writing the JAD file...
sign:
[sign] Certificate Issued By: Paulius Palevicius
[sign] Certificate Issued To: Paulius Palevicius
[sign] Writing the JAD file...
BUILD SUCCESSFUL
```

Paketų Java ME platformai konstravimas ir diegimas

Kadangi diegimas į mikroprocesorines korteles yra sudėtingas, be to rašymų skaičius į EEPROM atmintį yra ribotas, testavimui pasitelktas JCWDE įrankis, leidžiantis emuliuoti telefone esančią mikroprocesorinę kortelę. Tam tikslui sukurtas instrukcijų failas, kurio pagalba emuliuojama programa. Tuomet apdutool programos pagalba, perduodamos APDU komandos į emuliuojamą programą. Testavimui skirto apdutool instrukcijų failo pavyzdys:



3.4 pav.: Elektroninių pinigų testavimas mobilaus telefono emuliatoriuje

```
powerup;
// Pasirenkamas diegimo apletas
0x00 0xA4 0x04 0x00 0x09 0xa0 0x00 0x00 0x00 0x62 0x03 0x01 0x08 0x01 0x7F;
0x80 0xB8 0x00 0x00 0x0c 0x0a 0xA0 0x00 0x00 0x00 0x62 0x03 0x01 0x0C 0x03
   0x01 0x00 0x7F;
// Pasirenkamas testuojamas apletas
0x00 0xA4 0x04 0x00 0x0a 0xA0 0x00 0x00 0x00 0x62 0x03 0x01 0x0C 0x03 0x01
   0x7F;
// čSiuniama instrukcija
0x00 0xAA 0x11 0x00 0x15 0xa9 0x34 0x38 0x1c 0x1f 0x22 0xf6 0xd2 0x78 0xd7
   0x1b 0x2c 0x80 ...;
powerdown;
```

Java Card apletai į mikroprocesorinę kortelę diegiami naudojant programą GPShell ir mikroprocesorinių kortelių skaitytuvą. Šiai programai taip pat buvo sukurtas instrukcijų failas, leidžiantis automatizuoti procesą. Taip pat buvo sukurtas instrukcijų failas, leidžiantis atlikti testavimą mikroprocesorinėje kortelėje. Buvo pastebėta, kad naudojant mikroprocesorinių kortelių skaitytuvą su ilgai trunkančiais skaičiavimais, ryšys nutrūksta, todėl dalis testavimo buvo atliekama tiesiai iš telefono, naudojant J2ME apletą.

3.4 MODELIO REALIZACIJA JAVA ME PLATFORMOJE

Stefan Brands elektroninių pinigų modelis Java ME platformoje kaip ir Java SE platformoje realizuotas Bouncy Castle bibliotekos pagalba. Skaičiavimams naudojama ta pati klasė, kaip ir Java SE

platformoje, todėl adaptuojant reikėjo pakeisti tik sąsają duomenų įvedimui ir išvedimui. Skaičiavimai, kaip ir Java SE platformoje, atlikti prie 32, 64, 128, 256 ir 512 bitų parametrų. Žemiau pateikti rezultatai, gauti testuojant bendrą sistemos laiką ir atskirai vartotojo bei pardavėjo laikus.

3.5 pav.: Bendra realizacijos laiko priklausomybė nuo bitų skaičiaus J2ME platformoje

3.6 pav.: Verslininko realizacijos laiko priklausomybė nuo bitų skaičiaus J2ME platformoje

Kaip matome iš grafikų, realizacijos greitis Java ME platformoje, palyginus su Java SE platforma, skiriasi daugiau nei 100 kartų. Toks didelis skirtumas atsiranda dėl techninių NOKIA 6212 Classic parametrų. Laikas reikalingas vartotojo protokolų daliai atlikti siekia ~13 sekundžių. Pridėjus laiką, reikalingą duomenų apsikeitimui tarp banko (verslininko) terminalo ir vartotojo, bendras vartotojo pinigo paėmimo ir išleidimo protokolų laikas gali išaugti iki 15-20 sekundžių, priklausomai nuo grafinės sąsajos realizacijos ir duomenų apsikeitimo kanalo greičio.

3.7 pav.: Vartotojo realizacijos laiko priklausomybė nuo bitų skaičiaus J2ME platformoje

3.5 ARITMETINIŲ FUNKCIJŲ REALIZACIJA JAVA CARD PLATFORMOJE

Šiame skyriuje aptarsime metodus, panaudotus realizuojant biblioteką, skirtą darbui su dideliais skaičiais, Java Card platformoje.

Skaičių reprezentacija

Sukurtoje sistemoje skaičiai reprezentuojami baitų masyve. Baitų masyvo dydis priklauso nuo pasirinkto baigtinio lauko parametro p. Nepriklausomai nuo skaitmenų skaičiaus, masyvo dydis lieka toks pat. 3.1 lentelėje pateikta dvejų skaičių reprezentacija baitų masyve, kuomet p yra 64 bitų.

3.1 lentelė: Skaičių reprezentacija baitų masyve, kai p yra 64 bitų

18446744073709551616							
e4	3e	93	24	65	cb	31	f0
281474976710656							
00	01	5b	0a	55	5d	b8	6b

Skaičių palyginimas

Dvejų skaičių palyginimas realizuotas dviem etapais. Pirmo etapo metu patikrinama ar baitų masyvai yra vienodo ilgio. Jeigu ilgiai skiriasi, masyvai lyginami pagal jų dydį. Jeigu ilgiai sutampa, tuomet tikrinama einant iš kairės į dešinę, lyginant atitinkamus baitus.

Skirtumas moduliu p

Realizuojant dviejų skaičių a ir b skirtumą s moduliu p, galimi du variantai:

- Jeigu $a > b$, tuomet $s = a - b$.
- Jeigu $a < b$, tuomet $s = p - (b - a)$

Didesnio ir mažesnio skaičių skirtumas realizuotas imant masyvo elementus iš kairės į dešinę ir atliekant atitinkamų narių skirtumą re.

- Jei $re > 0$, tuomet į naujo masyvo atitinkamą elementą įrašoma re reikšmė
- Jei $re < 0$, įrašoma reikšmė $256 + re$ ir sekančio žingsnio metu prie narių skirtumo pridedamas -1.

Modulinė operacija

Modulinės operacijos apskaičiavimui pateikiamas skaičius a ir modulis m. Tuomet, kol $a > m$ perskaičiuojama reikšmė $a = m - a$. Atsakymas gaunamas kai $a < m$. Metodo realizacijoje naudojamos skaičių palyginimo ir skirtumo operacijos.

Sudėtis moduliu p

Realizuojant dviejų skaičių a ir b sudėtį, atliekama pilna šių skaičių sudėtis, o tada, jeigu $a > m$ pritaikoma modulinė operacija. Didesnio ir mažesnio skaičių sudėtis realizuota imant masyvo elementus iš dešinės į kairę ir apskaičiuojant atitinkamų narių sumą re.

- Jeigu, gauta suma $re < 256$, tuomet į naujo masyvo atitinkamą elementą įrašoma re reikšmė.
- Jeigu, gauta suma $re > 255$, tuomet įrašoma $256 - re$ reikšmė ir sekančio žingsnio metu prie sumos pridedamas 1.

Postūmis į dešinę arba dalyba iš dviejų

Dvejetainio skaičiaus a postūmis į dešinę per vieną bitą yra ekvivalenti operacija dalybai iš dviejų. Jeigu a yra nelyginis skaičius, t.y. mažiausiai reikšmingas bitas $(\ldots 01000110\mathbf{1})$ yra lygus vienetui, tuomet apskaičiuojama $a \equiv a + p \pmod p$. Šios operacijos pagalba gaunamas lyginis skaičius, ekvivalentus duotajam.

Masyvo elementai imami iš kairės į dešinę ir kiekvienam pritaikoma postūmio į dešinę operacija. Jeigu operacijos metu masyvo elemento mažiausiai reikšmingas bitas panaikinamas, tuomet jis perkeliamas sekančiam elementui.

Modulinė eksponentė

Standartiniai modulinės eksponentės algoritmai Java Card platformoje yra neefektyvūs, todėl modulinės eksponentės operacija buvo realizuota naudojant RSA šifravimo schemą. Java Card platformoje RSA šifravimo ir dešifravimo operacijos yra atliekamos naudojant specialiai tam skirtą kriptografinį procesorių.

- Žinutės m šifravimas: $c = m^e \ (mod\ m)$.
- Šifruoto teksto c dešifravimas: $m = e^d \ (mod\ m)$.

Matome, kad žinutės šifravimas ir šifruoto teksto dešifravimas yra modulinei eksponentei ekvivalenčios operacijos. Žemiau pateikta modulinės eksponentės realizacijos ištrauka naudojant RSA šifravimo metodą.

```
RSAPublicKey rpk;
rpk = (RSAPublicKey) KeyBuilder.buildKey(KeyBuilder.TYPE_RSA_PUBLIC,
   KeyBuilder.LENGTH_RSA_512, false);
rpk.setModulus(p, (short)0, (short)64);
rpk.setExponent(e, (short)0, (short)64);
cipher.init(rpk, Cipher.MODE_ENCRYPT);
cipher.doFinal(m, (short)0, (short)(64), buf, (short)ISO7816.OFFSET_CDATA);
```

Sandauga moduliu p

Kadangi dviejų skaičių a ir b sandaugos moduliu p realizacija mikroprocesorinėje kortelėje yra neefektyvi, problema išspręsta naudojant 3.1 lygybę.

$$(a+b)^2 \equiv a^2 + 2ab + b^2 \ (mod\ m) \tag{3.1}$$

Iš šios lygybės galima nesunkiai išreikšti $(a+b)^2 - a^2 - b^2 \equiv 2ab \ (mod\ m)$. Apskaičiavus kairėje lygybės pusėje esantį reiškinį galimi du atvejai:

1. $l = (a+b)^2 - a^2 - b^2$ reikšmė lyginė.
2. $l = (a+b)^2 - a^2 - b^2$ reikšmė nelyginė.

Pirmu atveju reikšmė pastumiama į dešinę per vieną bitą. Kadangi $l \equiv l + m \ (mod\ m)$, antru atveju pridedamas skaičius m. Kadangi m yra nelyginis skaičius, gaunama reikšmė $l + m$ yra lyginė. Tuomet ji pastumiama į dešinę per vieną bitą.

Realizacijos greitis

Naudojant sukurtą biblioteką, buvo realizuota Stefan Brands elektroninių pinigų modelio vartotojo dalis mikroprocesorinėje kortelėje. Pilnas vartotojo dalies protokolų vykdymo laikas - 52 sekundės. Nors realizacija mikroprocesorinėje kortelėje užtikrina didžiausią saugumo lygį, toks laikas nėra priimtinas, todėl siūlomas mišrus variantas:

- Protokolo dalys, kuriose naudojamas vartotojo privatus raktas, atliekamos mikroprocesorinėje kortelėje.
- Likusios protokolo dalys realizuojamos Java ME platformoje.

Lentelėje 3.2 pateikiami skirtingų operacijų vykdymo laikas:

3.2 lentelė: Operacijų vykdymo laikas

Operacija	Laikas (s)
Suma	0.49
Skirtumas	0.33
Postūmis į dešinę	0.79
Modulinė eksponentė	0.39
Sandauga	3.1

Iš lentelės matome, kad neefektyviausia operacija yra daugyba. Taip yra todėl, kad ši operacija yra kitų operacijų kompozicija. Patobulinus sumos, skirtumo ir postūmio realizacijas, efektyviau veiktų ir sandaugos operacija.

IŠVADOS

- Pasiūlyta ribotos aritmetikos funkcijų klasė, darbui su dideliais skaičiais, kuri reikalinga Brands elektroninių pinigų schemos realizacijai, ribotų resursų skaičiavimo aplinkoje.
- Parodyta, kad ribotų aritmetinių funkcijų klasė gali būti realizuojama Java Card platformoje, panaudojant standartines Java Card funkcijas.
- Nustatytas operacijų vykdymo laikas mikroprocesorinėje kortelėje. Nustatyta, kad operacijos atliekamos pakankamai lėtai, todėl siūloma realizuoti tik tą vartotojo protokolo dalį, kurioje naudojamas privatus raktas.
- Atlikta elektroninių pinigų realizacija kompiuteryje ir mobiliajame telefone. Vertinant šioms realizacijoms reikalingą laiką, buvo įvertinta, kad realizacija kompiuteryje veikia 100 kartų greičiau negu telefone.
- Darbe atlikta programinės ir techninės įrangos, susijusios su elektroninių pinigų realizacija mobiliajame telefone ir mikroprocesorinėje kortelėje, analizė. Ši analizė galės būti panaudota ne tik realizuojant elektroninių pinigų sistemas, bet ir kitas kriptografines schemas, tokias kaip mobilus elektroninis parašas.
- Atliekant programinės įrangos realizaciją, didžioji dalis procedūrų, susijusių su programinės įrangos konstravimu ir diegimu į mobilųjį telefoną buvo automatizuotos, sukuriant specializuotus įrankius ir instrukcijų failus. Šių įrankių dėka sutaupomas laikas ir palengvinama kriptografinių ir matematinių metodų realizacija ateityje.
- Darbe surinkti ir aprašyti kriptografiniai protokolai, metodai ir schemos, naudotos realizuojant Stephan Brands elektroninių pinigų sistemą.

REKOMENDACIJOS

Žemiau pateikiamos problemos ir pasiūlymai tolimesniems tyrimams.

- Elektroninių pinigų modelio realizacijos galimybės, naudojant Mircro SD korteles su integruotu mikroprocesoriumi.
- Stafan Brands modelio su dalumo papildymu arba kito, pinigų dalumą palaikančio modelio realizacija.
- Aritmetinių operacijų realizacijos Java Card 2.2.1 aplinkoje adaptavimas naujausiai 3.0 versijai.
- Stafan Brands elektroninių pinigų modelio cirkuliacijos sistemos realizacija naudojant Bluetooth arba NFC ryšio kanalus.

PADĖKOS

Baigiant darbą, dėkoju savo darbo vadovui, profesoriui Eligijui Sakalauskui, už vadovavimą, pagalbą ir patarimus vykdant tyrimus, rašant magistro baigiamąjį darbą bei aprūpinimą technine įranga, be kurios šis darbas būtų neįmanomas.

LITERATŪRA

Brands, S. A. An Efficient Off-line Electronic Cash System Based On The Representation Problem. Technical report, Amsterdam, The Netherlands, The Netherlands, 1993.

Bulota, K. Algebra ir skaičių teorija. Vilnius, 1990.

Chaum, D. Blind signatures for untraceable payments. *Advances in Cryptology*, 1998. pp. 199–203.

Dwarf, B. HOWTO Build, Sign and Install MIDlets. Website, 2006. `http://browndrf.blogspot.com/`.

EC. E-Money Directive (2009/110/EC). Website, 2009. `http://ec.europa.eu/internal_market/payments/emoney/index_en.htm`.

Menezes, A. J., P. C. Oorschot, S. A. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996.

Okamoto, T. An Efficient Divisible Electronic Cash Scheme. In Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology. CRYPTO '95, Springer-Verlag, London, UK, 1995 pp. 438–451.

Rivest, R. L., A. Shamir, L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 1978. 21, 120–126.

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

Schnorr, C.-P. Efficient Signature Generation by Smart Cards. *J. Cryptology*, 1991. 4, 161–174.

Sterckx, M., B. Gierlichs, B. Preneel, I. Verbauwhede. Efficient Implementation of Anonymous Credentials on Java Card Smart Cards. In 1st IEEE International Workshop on Information Forensics and Security (WIFS 2009). IEEE, London,UK, 2009 pp. 106–110.

Wagstaff, S. Cryptanalysis of Number Theoretic Ciphers. Computational Mathematics Series, Chapman & Hall/CRC Press, Boca Raton, FL, USA, 2003.

1 PRIEDAS: ARITMETINIŲ FUNKCIJŲ REALIZACIJA MIKROPROCESORINĖJE KORTELĖJE

```
class FF
{
  public static byte[] shortToByteArray(short value) {
    return new byte[] {
      (byte)(value >>> 8),
      (byte)value
    };
  }

  public static short byteArrayToShort(byte [] b) {
    return (short) (((b[2] & 0xFF) << 8)+(b[3] & 0xFF));
  }

  public static byte[] lsubtract(byte[] a, byte[] b) {
    short re, e, f, use;
    byte sol[] = new byte[a.length];

    use = 0;
    for (short i= (short)(a.length-1); i >= 0; i--) {
      e = (short) (a[i] & 0xff);
      f = (short) (b[i] & 0xff);
      re = (short) (e-f);

      if ((short) (re-use) < 0) {
        sol[i] = (byte) (256+re-use);
        use = 1;
      }
      else {
        sol[i] = (byte) (re-use);
        use = 0;
      }
    }
```

```
  return sol;
}

public static byte[] ladd(byte[] a, byte[] b) {
  short re, e, f, keep, use, usec;
  short length = (short) ((a.length > b.length) ? a.length : b.length);
  byte bsum[];
  bsum = new byte[length];
  use = usec = 0;
  for (short i= (short) (a.length-1); i >= 0; i--) {
    e = (short) (a[i] & 0xff);
    f = (short) (b[i] & 0xff);
    re =  (short) (e+f+use);

    usec = (short) (re / 255);

    if (usec > 0) bsum[i] = (byte) (re-256);
    else  bsum[i] = (byte) re;

    if (usec > 0) use = 1;
    else use = 0;
  }

  return bsum;
}

public static short compare(byte[] a, byte[] b) {
  if (a.length != b.length) return (short) (a.length > b.length ? 1 : -1)
     ;

  for (short i=0; i<a.length; i++) {
    if (a[i] != b[i]) return (short) (a[i] > b[i] ? 1 : -1);
  }

  return 0;
}
```

```
public static byte[] subtract(byte[] a, byte[] b, byte[] m) {
  if (compare(a,b) == 1) {
    return modulus(lsubtract(a,b),m);
  } else {
    return modulus(lsubtract(b,a),m);
  }
}

public static byte[] add(byte[] a, byte[] b, byte[] m) {
  return modulus(ladd(a,b),m);
}

public static byte[] modulus(byte[] a, byte[] m) {
  while (compare(a,m) == 1) {
    a = lsubtract(a,m);
  }

  return a;
}

public static byte[] exp(byte[] a, byte[] e, byte[] m) {
  byte [] r = new byte[(short) m.length];
  RSAPublicKey rpk;
  rpk = (RSAPublicKey) KeyBuilder.buildKey(KeyBuilder.TYPE_RSA_PUBLIC,
     KeyBuilder.LENGTH_RSA_512, false);

  rpk.setModulus(m, (short)0, (short)m.length);
  rpk.setExponent(e, (short)0, (short)e.length);

  Cipher cipher = Cipher.getInstance(Cipher.ALG_RSA_NOPAD, false);
  cipher.init(rpk, Cipher.MODE_ENCRYPT);
  cipher.doFinal(a, (short)0, (short)(a.length), r, (short)0);
  return r;
  //return add(a,e,m);
}
```

```
public static byte[] pad(byte[] a, byte[] m) {
  if (a.length < m.length) {
    byte [] b = new byte[(short)m.length];
    Util.arrayCopy(a, (short)0, b, (short) (m.length-a.length), (short) a
       .length);
    for (short i = 0; i< (short) (m.length-a.length); i++) {
      b[i] = (byte) 0x00;
    }

    return b;
  }

  return a;
}

public static byte[] multiply(byte[] a, byte[] b, byte[] m) {
  byte[] b2 = {(byte) 0x02};

  return modulus(rightShift(subtract(exp(add(a,b,m),b2,m),add(exp(a,b2,m)
     ,exp(b,b2,m),m),m),m),m),m);
}

public static byte[] leftShift(byte[] a, byte[] m) {
  byte yourbytearray[] = new byte[a.length];
  Util.arrayCopy(a, (short)0, yourbytearray, (short)0, (short) a.length);

  byte overf = 0;
  byte lastoverf = 0;

  for (short i = (short) (yourbytearray.length-1); i >= 0; --i) {
    if ((yourbytearray[i] & 0x80) == 0x80) {
      overf = 0x01;
    } else {
      overf = 0x00;
    }
```

```
    yourbytearray[i] = (byte) ((yourbytearray[i] << 1) + lastoverf);
    lastoverf = overf;
  }

  return yourbytearray;
}

public static byte[] rightShift(byte[] a, byte[] m) {
  byte yourbytearray[] = new byte[a.length];
  Util.arrayCopy(a, (short)0, yourbytearray, (short)0, (short) a.length);

  boolean overf = false;
  boolean lastoverf = false;

  if ((yourbytearray[(short)(yourbytearray.length-1)] & 0x01) == 0x01) {
    yourbytearray = ladd(a,m);
  }

  for (short i = 0; i < yourbytearray.length; ++i) {
    if ((yourbytearray[i] & 0x01) == 0x01) {
      overf = true;
    } else {
      overf = false;
    }
    if (lastoverf) {
      yourbytearray[i] = (byte) ((yourbytearray[i] >>> 1) | ((byte) 0x80)
          );
    } else {
      short test = (short) (yourbytearray[i] & 0xff);
      test = (byte) (test >>> 1);
      yourbytearray[i] = (byte) test;
    }

    lastoverf = overf;
  }
  return yourbytearray;
```

```
    }
}
```

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.math.BigInteger;
import org.bouncycastle.crypto.*;
import org.bouncycastle.crypto.digests.*;

public class brandsProcess {
  public static String output(String label, BigInteger[] data) {
    String out;
    out = label+":\n";

    for (int i=0; i<data.length; i++) {
      out += data[i]+"\n";
    }

    out += "\n";

    return out;
  }

  public static BigInteger hash (String m)
  {
    BigInteger mn = BigInteger.ZERO;
    SHA256Digest md = new SHA256Digest();
    byte[] result = new byte[md.getDigestSize()];

    md.update(m.getBytes(),0,m.getBytes().length);
    md.doFinal(result,0);
    mn = new BigInteger(result);
    return mn;
  }

  public static BigInteger hashmod (BigInteger p, String m)
  {
```

```
  BigInteger mh = hash(m);
  return mh.mod(p);
}

public static BigInteger hashC (BigInteger p, BigInteger[] data)
{
  String sdata = "";

  for (int i=0; i<data.length; i++) {
    sdata = sdata+data[i];
  }

  return hashmod (p, sdata);
}

public static short modCompare (BigInteger p, BigInteger a, BigInteger b)
{
  if (a.compareTo(b) == 0) return 1;
  if (p.compareTo(a.add(b)) == 0) return 1;

  return 0;
}

public static String process () {
  BigInteger p    = new BigInteger("4294967387");
  BigInteger q    = new BigInteger("2147483693");
  BigInteger g    = new BigInteger("2148750335");
  BigInteger g1   = new BigInteger("1216004768");
  BigInteger g2   = new BigInteger("549585894");
  BigInteger x    = new BigInteger("1638570863");
  BigInteger h    = new BigInteger("4006305935");
  BigInteger u1   = new BigInteger("1703966116");
  BigInteger hu1  = new BigInteger("2702159174");
  BigInteger z    = new BigInteger("1762983208");

  int userStartTime=0;
```

```
int userEstimatedTime=0;
int merchantStartTime=0;
int merchantEstimatedTime=0;

int ssize = 31;

BigInteger vi = BigInteger.valueOf(32);
BigInteger t = BigInteger.valueOf(123456);
Random random = new Random();

/* Withdrawal step 1 */

BigInteger w = new BigInteger(ssize, random); w = w.mod(q);
BigInteger a = g.modPow(w,p);
BigInteger b = hu1.multiply(g2).mod(p).modPow(w,p);

out += output("PP 1", new BigInteger[] {w,a,b});

/* Withdrawal step 2 */

userStartTime = (int)System.currentTimeMillis();

BigInteger s = new BigInteger(ssize, random); s = s.mod(q);
BigInteger x1 = new BigInteger(ssize, random); x1 = x1.mod(q);
BigInteger x2 = new BigInteger(ssize, random); x2 = x2.mod(q);
BigInteger u = new BigInteger(ssize, random); u = u.mod(q);
BigInteger v = new BigInteger(ssize, random); v = v.mod(q);


BigInteger A = hu1.multiply(g2).mod(p).modPow(s,p);
BigInteger B = g1.modPow(x1,p).multiply(g2.modPow(x2,p)).mod(p);
BigInteger z_ = z.modPow(s,p);

BigInteger a_ = a.modPow(u,p).multiply(g.modPow(v,p)).mod(p);
BigInteger b_ = b.modPow(s.multiply(u).mod(q),p).multiply(A.modPow(v,p)
   ).mod(p);
```

```
BigInteger c_ = hashC(q,new BigInteger[] {A,B,z_,a_,b_});

BigInteger c = c_.multiply(u.modInverse(q)).mod(q);

userEstimatedTime += (int)System.currentTimeMillis() - userStartTime;

out += output("PP 2", new BigInteger[] {s,x1,x2,u,v,A,B,z_,a_,b_,c,c_})
   ;

/* Withdrawal step 3 */

BigInteger r = w.add(c.multiply(x)).mod(q);

out += output("PP 3", new BigInteger[] {r});

/* Withdrawal step 4 */

userStartTime = (int)System.currentTimeMillis();

BigInteger pass = BigInteger.ONE;
BigInteger r_ = v.add(r.multiply(u)).mod(q);

if (modCompare(
  p,
  g.modPow(r,p),
  a.multiply(h.modPow(c,p)).mod(p)) != 1)
{
  pass = BigInteger.ZERO;
}

if (modCompare(
  p,
  hu1.multiply(g2).mod(p).modPow(r,p),
  b.multiply(z.modPow(c,p)).mod(p)) != 1)
{
  pass = BigInteger.ZERO;
```

```
}

userEstimatedTime += (int)System.currentTimeMillis() - userStartTime;

out += output("PP 4", new BigInteger[] {r_,pass});

/* Payment 1 */

merchantStartTime = (int)System.currentTimeMillis();

BigInteger d = hashC(q, new BigInteger[] {A,B,vi,t});

merchantEstimatedTime += (int)System.currentTimeMillis() -
   merchantStartTime;

out += output("MP 1", new BigInteger[] {d});

/* Payment 2 */

userStartTime = (int)System.currentTimeMillis();

BigInteger r1 = u1.multiply(s).mod(q).multiply(d).mod(q).add(x1).mod(q)
   ;
BigInteger r2 = d.multiply(s).mod(q).add(x2).mod(q);

userEstimatedTime += (int)System.currentTimeMillis() - userStartTime;

out += output("MP 2", new BigInteger[] {r1,r2});

/* Payment 3 */

c_ = hashC(q, new BigInteger[] {A,B,z_,a_,b_});
pass = BigInteger.ONE;

merchantStartTime = (int)System.currentTimeMillis();
```

```
if (modCompare(
  p,
  B.multiply(A.modPow(d,p)).mod(p),
  g1.modPow(r1,p).multiply(g2.modPow(r2,p)).mod(p)) != 1)
{
  pass = new BigInteger("11");
}

if (modCompare(
  p,
  g.modPow(r_,p),
  a_.multiply(h.modPow(c_,p)).mod(p)) != 1)
{
  pass = new BigInteger("12");
}

if (modCompare(
  p,
  A.modPow(r_,p),
  b_.multiply(z_.modPow(c_,p)).mod(p)) != 1)
{
  pass = new BigInteger("13");
}

merchantEstimatedTime += (int)System.currentTimeMillis() -
   merchantStartTime;

out += output("MP 3", new BigInteger[] {pass});

/* Deposit 1 */

d = hashC(q,new BigInteger[] {A,B,vi,t});
c_ = hashC(q,new BigInteger[] {A,B,z_,a_,b_});

pass = BigInteger.ONE;
```

```
    if (modCompare(
      p,
      B.multiply(A.modPow(d,p)).mod(p),
      g1.modPow(r1,p).multiply(g2.modPow(r2,p)).mod(p)) != 1)
    {
      pass = new BigInteger("11");
    }

    if (modCompare(
      p,
      g.modPow(r_,p),
      a_.multiply(h.modPow(c_,p)).mod(p)) != 1)
    {
      pass = new BigInteger("12");
    }

    if (modCompare(
      p,
      g.modPow(r_,p),
      a_.multiply(h.modPow(c_,p)).mod(p)) != 1)
    {
      pass = new BigInteger("13");
    }

    out += output("IP 1", new BigInteger[] {d,c_,pass});

    out += "UT: "+Integer.toString(userEstimatedTime)+"\n";
    out += "MT: "+Integer.toString(merchantEstimatedTime)+"\n";

    return out;
  }
}
```