UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Matematika - uporabna smer (UNI)

Miha Vuk

DIGITALNI DENAR IN OMEJENI SLEPI PODPISI

Diplomsko delo

Ljubljana, 2001

Kazalo

PROGRAM DIPLOMSKEGA DELA

Digitalni denar in omejeni slepi podpisi

Delo naj predstavi matematične osnove in kriptografske algoritme, potrebne za študij slepih digitalnih podpisov in digitalnega denarja. Glavni cilj je predstaviti osnovne principe in rešitve (npr. sheme Schnorra, Okomota in Brandsa, overjanje s tajnim ključem), nato pa podrobneje predstaviti in analizirati en konkreten sistem digitalnega denarja.

Literatura:

Douglas R. Stinson, Cryptography – Theory and Practice, CRC Press, 1995.

P. Wayner, Digital Cash, Commerce on the Net, Academic Press, 1996.

S. A. Brands, Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy, MIT Press, 2000.

Povzetek

V tem delu bomo spoznali konkreten sistem digitalnega denarja in se poskušali čimbolj približati praktični implementaciji. Najprej bomo spoznali potrebne matematične in kriptografske osnove ter nekaj osnov monetarne ekonomije. Nato bomo izbrali konkreten sistem digitalnega denarja, ki ga je iznašel Brands, in ga poskušali čimbolj prilagoditi praktičnim zahtevam. Pri tem bodo igrali bistveno vlogo certifikati s skrivnim ključem, ki so tudi njegov izum. Za nas bodo najpomembnejši omejeni slepi podpisi certifikatov s skrivnim ključem, ki bodo omogočali učinkovito izdajanje digitalnega denarja. Namen diplome je prikazati trenutno dogajanje na področju digitalnega denarja in podati dovolj informacij za praktično implementacijo.

Ključne besede: digitalni denar, elektronski kovanci, razširitve digitalnega denarja, omejeni slepi podpisi, certifikati s skrivnim ključem.

Abstract

In this thesis a practical off-line digital money system is introduced with a practical implementation in mind. First we develop some mathematical and cryptographical background and also explain some monetary economy. After that we give a description of common properties of different digital money systems and choose one for a detailed study. The chosen system was invented by Brands and is well suited for a practical implementation. The system is based on secret-key certificates that were also invented by Brands. Of our main interest are restrictive blind signatures of secret-key certificates that make possible to construct efficient withdrawal protocols. Our aim is to provide a reader with a general overview of current digital money systems and enough knowledge for a practical implementation.

Keywords: digital money, electronic coins, extensions of off-line cash, restrictive blind signatures, secret-key certificates.

Math. Subj. Class. (2000): 94A60, 94A62.

Poglavje 1

SPLOŠNO O DENARJU

Ljudje so že v pradavnini med seboj trgovali. Najprej je šlo le za blagovno menjavo. Izmenjevali so si dobrine in storitve. Kmalu se je pokazalo, da je takšna menjava precej okorna. Kupčija je bila namreč možna le, če je tudi kupec imel nekaj, kar je prodajalca zanimalo. Da bi se izognili tem problemom, so uvedli splošnega menjalnega posrednika - denar. Sedaj se je lahko vsak kupec zanesel, da bo prodajalec z veseljem vzel njegov denar, saj ga bo lahko kadarkoli brez težav uporabil. Denar ima tri glavne funkcije:

- **menjalni posrednik**: omogoča menjavo
- **nosilec vrednosti**: omogoča vrednostne primerjave
- **hranilec vrednosti**: omogoča varčevanje

Za denar bi bila uporabna katerakoli ekonomska dobrina[1], ki s časom ne spreminja vrednosti. Poljščine recimo ne bi bile primerne, saj se s časom pokvarijo. Tako so se kot denar začele uporabljati predvsem razne redke kovine, najbolj zlato in srebro. Obstajalo je več vrst denarja. Ker vse vrste niso bile po vsem svetu enako redke in enako iskane, so bila spremenljiva tudi menjalna razmerja med njimi.

Denar pa se je dalo tudi ponarejati. Če si zlatu dodal nekaj srebra, je bila vrednost seveda manjša, čeprav se tega na oko ni dalo opaziti. To je majalo zaupanje v denar. To so rešili tako, da so kralji in podobni veljaki začeli kovati kovance. Kovanci so bili narejeni iz fiksne količine določene kovine, nanj pa so v kovnici odtisnili pečat. Ta je zagotavljal, da je kovanec res toliko vreden, za kar je kralj tudi jamčil. Kot garancijo ga je bil pripravljen zamenjati za enako vrednost nečesa drugega (npr. zlata).
S časoma so ljudje nehali gledati na kovance kot na kos kovine, ampak je bil pomemben le še pečat in vrednost, ki jo je nosil kovanec. Začeli so tiskati tudi bankovce, katerih proizvodnja je skoraj zastonj. Navada pa je bila, da je kralj natiskal le toliko bankovcev, kolikor je dobil novega zlata v državno zakladnico. Temu pravimo, da je denar imel kritje. Tako je bilo vedno možno bankovce zamenjati za zlato in to je dajalo ljudem zaupanje v denar. Lahko bi rekli, da je bil denar **terjatev do države** (kralja). Te terjatve sicer ni skoraj nihče vnovčil, imela pa je fiksno vrednost, zato so z njo radi trgovali.

Ta model denarja se je ohranil do današnjih dni. Danes ga ne izdajo več kralji ampak centralna banka in država. Ker se je zaupanje v denar še krepilo, potrebe po njem pa so rasle, danes

[1] Ekonomske dobrine so dobrine, ki so iskane in relativno redke, zato imajo svoje ceno. Zrak recimo vsi potrebujemo, a ga je dovolj, zato nima cene. Obstajajo pa tudi dobrine, ki so redke, a jih nihče ne potrebuje.

države nimajo več kritja za svoj denar[2]. Namesto tega izdajajo denar brez kritja, večinoma tako, da centralna banka izdaja kredite poslovnim bankam. Ker količina denarja ni več vezana na količino kritja, se tudi lažje prilagaja potrebam gospodarstva. Novo je tudi to, da je z zakonom predpisano, da so trgovci dolžni sprejeti domači denar in to v vseh oblikah (različni kovanci in bankovci).

Včasih je ves denar res obstajal v obliki gotovine. Danes temu ni več tako. Fizične in pravne osebe imajo pri različnih bankah odprte račune in na njih denar. Temu pravimo **knjižni denar**. Plačilo je mogoče izvesti s prenosom denarja med dvema računoma. Temu pravimo **brezgotovinsko plačilo**. Tudi emisija lahko poteka v knjižni obliki, če centralna banka da kredit poslovni banki, tako da ji denar le nakaže. Poleg gotovinskega plačevanja poznamo torej tudi razne načine brezgotovinskega plačevanja. Mednje spadajo: čeki, menice in drugi podobni papirji ter bančne in kreditne kartice. Zadnje včasih napačno imenujemo kar **plastični denar**. V resnici to sploh ni denar, ampak le še en način, kako prenesti knjižni denar med dvema računoma. Tudi plačila prek interneta, bodisi prek spletnih poslovalnic bank bodisi s kreditno kartico, spadajo v isto skupino.

V zadnjih dvajsetih letih so se pojavile tudi nove oblike pravega denarja. Gre za oblike, ki so ekvivalentne klasični gotovini. Med bolj primitivne oblike bi lahko šteli telefonske in ostale predplačilne kartice. V zameno za gotovino ali knjižni denar dobiš na kartici določeno število impulzov, ki jih potem po želji porabljaš. To sicer še ni denar, ker ni splošno uporaben, bi pa lahko bil, če bi ga sprejemali vsi. Za kaj takega pa so žal omenjene kartice premalo varne.

Zato so razvili tudi druge oblike, ki jim upravičeno lahko rečemo **digitalni denar ali elektronska gotovina**[3]. V praksi se sicer še nikjer ne uporablja, vendar bodo kmalu ustvarjeni pogoji za njegovo uporabo. Na tem področju ima Slovenija morda celo prednost, saj je majhna država s precej homogenim bančnim sektorjem.

1.1 DIGITALNI DENAR

Poglejmo zdaj podrobneje digitalni denar, ki je tudi glavna tema moje diplome. Namesto z gotovino (kovanci in bankovci) se lahko posluje tudi z elektronskimi kovanci. To je osnovna in tudi najpreprostejša oblika digitalnega denarja.

Če želimo, da bodo temu denarju ljudje zaupali, je prav, da tudi njega izdaja le centralna banka. Teoretično bi ga lahko izdajale tudi poslovne banke, vendar bi jih morala država nadzirati, da bi jim ljudje zaupali. Poleg tega bi tudi večje število izdajateljev denarja verjetno ustvarjalo zmedo. Mi bomo te dileme v prihodnje pustili ob strani in bomo večinoma govorili le o treh ključnih osebah: uporabniku, trgovcu in banki.

Poleg elektronskih kovancev poznamo tudi elektronske čeke. Ti nam omogočajo, da jih ne porabimo v celoti, ampak le delno (seveda le v okviru njihove maksimalne vrednosti). Poleg čekov se da tudi nekatere kovance deliti na manjše dele. Tako imamo v denarnici manj kovancev, pa še vedno lahko plačamo skoraj poljubne zneske. Naj omenim, da je vračanje elektronskega denarja sicer možno, da pa se mu poskušamo izogniti. Je namreč precej nepraktično pričakovati povračilo denarja, ki smo ga recimo poslali po elektronski pošti. Obstajajo tudi kovanci, ki jih lahko večkrat uporabimo, odpirajo pa se še druge nove možnosti, ki jih pri klasični gotovini ne poznamo.

[2]Nekaj kritja države še vedno imajo. Večinoma v tujih valutah, manjši del pa tudi v zlatu.

[3]Pojavlja se dilema ali je primernejši izraz elektronski ali digitalni denar. Noben izraz ni zares ustrezen. V prihodnje bom uporabljal izraze digitalni denar, elektronska gotovina, elektronski kovanci in elektronski čeki. Približno takšno je tudi stanje v tujih virih.

Ena od osnovnih zahtev za digitalni denar je, da mora delovati tudi brez povezave z omrežjem (off-line). To mu omogoča uporabo povsod, čeprav je primarno namenjen poslovanju prek interneta. Bančne kartice tega pogoja recimo ne izpolnjujejo, saj POS[4] avtomat običajno zahteva stalno povezavo z bančnim centrom.

Digitalni denar delimo na anonimen in neanonimen. Slednji je podoben klasičnim čekom. Poleg tega lahko banka s svojim digitalnim podpisom zagotavlja, da ček ima kritje. Za implementacijo takega denarja zadoščajo že s klasičnimi kriptografskimi algoritmi, kot je digitalni podpis. Neanonimnost denarja večine ljudi sploh ne moti. Na veliko uporabljajo čeke in v zadnjem času tudi kreditne in plačilne kartice, ki so poleg tega še precej tvegane. Vendarle bi želeli imeti tudi anonimen digitalni denar, saj to lastnost klasična gotovina že od nekdaj ima in je včasih tudi zaželjena. Da pa bi to dosegli, so potrebni povsem novi, bolj zapleteni protokoli. Tak denar sme biti namreč anonimen le ob pošteni uporabi. Zlorabe naj onemogoči ali pa naj razkrinka goljufa. Predvsem takemu denarju, ki se zdi tudi najbolj perspektiven, se bo posvetila moja diploma.

[4]POS (Point Of Service) avtomati se nahajajo v trgovinah pri blagajnah in omogočajo plačevanje z bančnimi karticami. Običajno zahtevajo stalno povezavo z bančnim centrom, da lahko preverijo, da bo banka zares plačala željeni znesek.

Poglavje 2

MATEMATIČNE OSNOVE

V tem poglavju bomo spoznali matematične strukture in operacije, ki jih bomo potrebovali v nadaljevanju. Najprej si bomo razjasnili nekaj splošnih pojmov in izrekov iz teorije grup. Koristili nam bodo pri razumevanju grupe G_q, kjer bomo izvajali večino operacij. To je splošna ciklična multiplikativna grupa reda q, kjer je q praštevilo. V njej si bomo ogledali operacije, ki jih bomo potrebovali in jih ovrednotili s stališča zahtevnosti. Ogledali si bomo tudi dva praktična primera grup G_q in sicer podgrupo $\mathbb{Z}_p^*$ in podgrupo točk na eliptični krivulji. Nato se bomo posvetili problemu diskretnega logaritma, ki je ključen za kriptografske algoritme, ki jih bomo v nadaljevanju opisovali. To je računsko zelo težak problem, katerega inverz (potenciranje) pa je računsko enostaven. To s pridom izrabljajo mnogi kriptografski algoritmi. Zahtevnost diskretnega logaritma si bomo ogledali tudi na konkretnih primerih.

2.1 SPLOŠNO O GRUPAH

V tem razdelku bomo spoznali nekaj pojmov in za nas pomembne izreke iz teorije grup.

Definicija 2.1. *Grupa je množica G z notranjo operacijo $\circ$, ki slika $G \times G \to G$. Za operacijo morajo veljati še naslednje lastnosti:*

1. *asociativnost: Za vsake $a, b, c \in G$ velja $(a \circ b) \circ c = a \circ (b \circ c)$.*
2. *obstoj enote: Obstaja $e \in G$, tako da za vsak $a \in G$ velja $e \circ a = a \circ e = a$. Pravimo mu nevtralni element ali enota.*
3. *obstoj inverza: Za vsak $a \in G$ obstaja $a^{-1} \in G$, tako da $a \circ a^{-1} = a^{-1} \circ a = e$.*

V praksi pri komutativnih (Abelovih) grupah operacijo običajno pišemo kot seštevanje ($+$), lahko pa tudi kot množenje ($\cdot$), kar pa je v bistvu stvar naše odločitve. Glede na to grupo imenujemo aditivna ali multiplikativna, pa tudi enoto označimo kar z 0 ali 1. Mi bomo večinoma uporabljali multiplikativne grupe (multiplikativen zapis).

Naravno uvedemo operacijo potenciranja. Naj bo $n \in \mathbb{N}$, potem je

$$a^n = \underbrace{a \cdot a \cdot \ldots \cdot a}_{n}, \quad a^0 = e, \quad a^{-n} = (a^{-1})^n.$$

Red elementa končne grupe G z enoto e je najmanjše naravno število r, za katero velja $a^r = e$.

Red grupe je število njenih elementov oziroma moč množice.

Izrek 2.2. ***(Lagrange)*** *Red elementa končne grupe deli moč grupe.* ■

Dokaz tega izreka si lahko ogledate v katerikoli knjigi o grupah, recimo v knjigi [Vid89].

Pravimo, da je grupa G končno **generirana** z množico $\mathcal{M} = \{a_1, a_2, \ldots, a_n\}$, če je G oblike: $G = \{a_{i_1}^{k_{i_1}} \cdot a_{i_2}^{k_{i_2}} \cdot a_{i_3}^{k_{i_3}} \cdot \ldots \cdot a_{i_N}^{k_{i_N}} ; a_{i_j} \in \{a_1, \ldots, a_n\},\ k_{i_j} \in \mathbb{Z}\}$. Grupo generirano z enim samim elementom (**generatorjem**) imenujemo **ciklična grupa**.

Trditev 2.3. *Vse ciklične grupe so komutativne (Abelove).*

Dokaz. Ciklično grupo lahko zapišemo kot $G = \{a^{k_1} \cdot a^{k_2} \cdot \ldots \cdot a^{k_N} ; k_i \in \mathbb{Z}\}$. Ker velja

$$a^i \cdot a^j = \underbrace{a \cdot a \cdot \ldots \cdot a}_{i+j} = a^{i+j} = a^j \cdot a^i,$$

je to enakovredno zapisu $G = \{a^i;\ i \in \mathbb{Z}\}$. Iz tega zapisa in gornje enakosti izhaja, da je ciklična grupa komutativna. ■

Izrek 2.4. *Vsaka končna ciklična grupa moči n je izomorfna grupi števil $(\mathbb{Z}_n, +)$, kjer ima $\mathbb{Z}_n$ elemente $\{0, \ldots, n-1\}$.*

Dokaz. Označimo z G končno ciklično grupo moči n. Definirajmo preslikavo $f : \mathbb{Z} \longrightarrow G$, s predpisom $f(i) = a^i$. Ta preslikava je homomorfizem, saj velja

$$f(i) + f(j) = a^i a^j = a^{i+j} = f(i+j),$$

slika te preslikave pa je ravno ciklična grupa generirana z a, torej grupa G.
Naj bo r red elementa a. Poljuben $i \in \mathbb{Z}$ lahko zapišemo kot $i = kr + s$, kjer je $k \in \mathbb{Z}$ in $0 \leq s < r$. Velja

$$G = \{a^i; i \in \mathbb{Z}\} = \{a^{kr+s}; k \in \mathbb{Z},\ 0 \leq s < r\} = \{a^s; 0 \leq s < r\}.$$

Torej je red generatorja ciklične grupe enak moči grupe (oz. $r = n$). Iz gornje konstrukcije grupe G je očitno sledi $\ker f = \{kr; k \in \mathbb{Z}\} = r\mathbb{Z} = n\mathbb{Z}$. Slika preslikave f je po izreku o dekompoziciji homomorfizma izomorfna $\mathbb{Z}/n\mathbb{Z} = \mathbb{Z}_n$. Torej je grupa G izomorfna grupi $\mathbb{Z}_n$. ■

Posledica 2.5. *Vse končne ciklične grupe moči n so med seboj izomorfne.* ■

2.2 MULTIPLIKATIVNA GRUPA G_q

Grupa G_q je ciklična multiplikativna grupa reda q, kjer je q praštevilo.

Oznaka 2.6. *Naj bo q praštevilo in naj v končni grupi G obstaja element $g \in G$, tako da $G = \{g^i;\ i \in \{0, .., q-1\}\}$. Potem je ta grupa ciklična moči q in jo označimo z G_q.*

Trditev 2.7. *Vsak element grupe G_q, razen enote, je generator.*

Dokaz. Naj bo a poljuben element G_q in naj ima red r. Po izreku 2.2 velja $r|q$. Ker je q praštevilo, velja $r = 1$ ali $r = q$. Prvi primer velja le za enoto, torej za vse ostale velja $r = q$, iz česar sledi, da generirajo celotno grupo. ■

Prej smo pokazali, da so vse končne ciklične grupe enake moči med seboj izomorfne, torej je tudi G_q izomorfna $\mathbb{Z}_q$. Brez težav tudi konstruiramo izomorfizem.

$$f : \mathbb{Z}_q \to G_q$$
$$n \mapsto g^n$$

Da gre res za izomorfizem, pokažemo podobno kot v dokazu izreka 2.4.

2.3 OPERACIJE V GRUPI G_q

Poglejmo sedaj štiri operacije, ki jih bomo izvajali v grupi G_q in zraven še preučimo, kako hitro jih znamo izvesti.

Osnovna operacija je množenje, saj gre za multiplikativno grupo. V praksi se uporabljajo le tiste grupe G_q, kjer je ta operacija dovolj hitra.

Druga operacija je potenciranje. Na prvi pogled se zdi, da izračun a^n ($a \in G_q,\ n \in \mathbb{N}$) zahteva n množenj, vendar se izkaže, da z metodo "kvadriraj in zmnoži" zadostuje že največ $2\lfloor \log_2 n \rfloor$ množenj. Algoritem (v programskem jeziku C) je sledeč:

```
int exp(int a, int n) {
  if (n==0) return 1;
  if (n==1) return a;

  for (int ret=1, int tmp=a; n>0; tmp=tmp*tmp, n=n/2)
    if (n%2!=0)
      ret=ret*tmp;

  return ret;
}
```

Potenciranje z znano osnovo, se da dodatno pospešiti (za trikrat), če imamo zaporedje `tmp` ($a^{2^1}, a^{2^2}, \ldots, a^{2^{\lfloor \log_2 q-1 \rfloor}}$) izračunano že vnaprej. Potem povprečno potenciranje zahteva le še $(\frac{1}{2}\log_2 n) - 1$ množenj.
Za potenciranje tudi v grupi G_q veljajo običajna pravila, ki pa jih bomo še malo dopolnili.

$$a^m a^n = a^{m+n} = a^{m+n \mod q}, \qquad (a^m)^n = a^{mn} = a^{mn \mod q}, \qquad a \in G_q,\ m, n \in \mathbb{Z}$$

Oznaka $\mod q$ označuje ostanek pri deljenju s q. Ker je a reda q, se lahko omejimo na $m, n \in \mathbb{Z}_q$, saj nič ne pridobimo, če bi jemali splošna cela števila. To dejstvo nam pride prav, ko računamo prostorske zahtevnosti algoritmov in protokolov. $\mathbb{Z}_q$ je obseg (celo komutativen obseg oz. polje). V njem znamo učinkovito seštevati, odštevati, množiti, deliti in potencirati, vendar se ne bomo spuščali v podrobnosti.

Tretja operacija je računanje inverza. Za deljenje namreč potrebujemo inverz, saj delimo po pravilu $a/b = ab^{-1}$. V splošnem inverz lahko izračunamo kot $a^{-1} = a^{q-1}$. Za določene konkretne primere grup G_q obstajajo tudi učinkovitejši algoritmi. Recimo za grupe $\mathbb{Z}_p$, kjer je p praštevilo, in njene multiplikativne podgrupe (nekatere spadajo v razred G_q) lahko uporabimo razširjen Evklidov algoritem. V podrobnosti se ne bomo spuščali.

Četrta "operacija" je izbira naključnega elementa. Pri tem tiho privzamemo, da znamo izbrati naključno število poljubne dolžine[1]. Izbira elementa poteka v dveh delih. Najprej izberemo naključno število vsaj tolikšne dolžine kot q in izračunamo ostanek po modulu q. To zapišemo kot $\omega \in_R \mathbb{Z}_q$. Nato izračunamo g^ω in to je naš naključni element. Z g bomo vedno označevali generator grupe G_q. To velja za celotno diplomo.

[1] V praksi je generiranje dovolj naključnih števil pogosto težko izvedljivo, saj je potreben dober vir naključnosti. Najboljši generatorji naključnih števil jemljejo za vir naključnosti kak naključen fizikalni pojav. Za to sicer potrebujejo dodatno strojno opremo, vendar bo to verjetno kmalu postala standardna oprema vsakega računalnika.

2.4 DEFINICIJE ZAHTEVNOSTI ALGORITMOV

Čeprav se še vedno ukvarjamo z operacijami v grupi G_q, bomo ta podrazdelek posvetili vrednotenju zahtevnosti algoritmov. To potrebujemo, da bomo lahko ovrednotili zahtevnosti algoritmov, s katerimi računamo določene operacije.

Definicija 2.8. Algoritem *je računski postopek, ki pri danih podatkih v končnem času reši problem in vrne rezultat.*

Ponavadi nas najbolj zanima, kako dolg je ta čas in koliko prostora potrebuje algoritem, zato definiramo naslednja pojma.

Definicija 2.9. Časovna zahtevnost *algoritma pri danih začetnih podatkih je enaka številu izvedenih osnovnih operacij. Osnovna operacija je ponavadi množenje ali seštevanje števil.*

Definicija 2.10. Prostorska zahtevnost *algoritma pri danih začetnih podatkih je enaka količini potrebnega prostora, ki ga običajno merimo v zlogih (oz. bytih).*

Pogosto je mogoče pridobiti na časovni zahtevnosti, če žrtvujemo nekaj prostorske zahtevnosti in tudi obratno. Mi se bomo ukvarjali predvsem s časovno zahtevnostjo, prostorsko pa bomo omenjali le, kadar bo res problematična.

V večini primerov je težko določiti natančno časovno zahtevnost algoritma, zato si pomagamo z asimptotično oceno. To pomeni, da gledamo, kako se časovna zahtevnost povečuje z večanjem velikosti podatkov. V ta namen bomo definirali $\mathcal{O}$-notacijo in o-notacijo.

Definicija 2.11. *Naj bosta $f(n)$ in $g(n)$ neki funkciji. Če obstaja konstanta $c > 0$ in tako število $n_0 > 0$, da velja $0 \leq f(n) < c \cdot g(n)$ za vse $n \geq n_0$, potem označimo $f(n) = \mathcal{O}(g(n))$. Z drugimi besedami, $f(n)$ ne raste asimptotično hitreje kot $g(n)$ do multiplikativne konstante natančno. Recimo:* $384x^2 + 12x + 232 = \mathcal{O}(x^2)$ *ali* $\sin x = \mathcal{O}(x)$.

Definicija 2.12. *Naj bosta $f(n)$ in $g(n)$ neki funkciji. Če za vsako konstanto $c > 0$ obstaja tako število $n_0 > 0$, da velja $0 \leq f(n) < c \cdot g(n)$ za vse $n \geq n_0$, potem označimo $f(n) = o(g(n))$. Z drugimi besedami* $\lim_{n\to\infty} f(n)/g(n) = 0$. *Recimo:* $19992x = o(x^2)$ *ali* $\log x = o(x)$ *ali* $x^n = o(\mathrm{e}^x)$.

Definicija 2.13. Polinomski algoritem *je algoritem, katerega časovna zahtevnost je enaka $\mathcal{O}(n^k)$, kjer je n velikost podatka in k neka konstanta $k > 0$. Algoritem, katerega časovne zahtevnosti ne moremo tako omejiti, imenujemo* **eksponentni algoritem**.

Velikost podatka se ponavadi meri v številu bitov, ki jih potrebujemo za njegovo reprezentacijo. Npr. velikost pozitivnega števila n je $\log_2 n = 1.44 \ln n$. V tem primeru je algoritem polinomski, če je njegova časovna zahtevnost $\mathcal{O}((\ln n)^k)$ za neko konstanto k in je eksponentni, če je njegova časovna zahtevnost enaka $\mathcal{O}(\mathrm{e}^{(f(\ln n))})$, kjer je f neka funkcija asimptotsko večja ali enaka nekemu polinomu. Pri tej delitvi poznamo tudi vmesni razred podeksponentnih algoritmov (npr. $\mathcal{O}((\ln n)^{\ln \ln n})$).

Poglejmo si za nas zanimive operacije in njihove rede zahtevnosti. Rezultati so zbrani v tabeli 2.1. Rezultati za G_q se nanašajo na pameteno izbrane primere G_q. Če je grupa slabo izbrana, so lahko operacije počasnejše.

Naj omenim, da asimptotična ocena zahtevnosti ni vedno pravi oz. edini kriterij. Na konkretnih podatkih, lahko hitreje deluje algoritem s slabšo asimptotično oceno, poleg tega pa se tudi algoritmi z enako asimptotično oceno lahko razlikujejo za nezanemarljiv konstantni faktor.

Operacija	Red zahtevnosti ($n = \ln q$)
seštevanje ($\mathbb{Z}_q$)	$\mathcal{O}(n)$
odštevanje ($\mathbb{Z}_q$)	$\mathcal{O}(n)$
množenje ($\mathbb{Z}_q$)	$\mathcal{O}(n^2)$
inverz ($\mathbb{Z}_q$)	$\mathcal{O}(n^3)$
množenje (G_q)	$\mathcal{O}(n^2)$
potenciranje (G_q)	$\mathcal{O}(n^3)$
inverz (G_q)	$\mathcal{O}(n^3)$
naključni element (G_q)	$\mathcal{O}(n^3)$

Tabela 2.1: Tabela operacij in njihovih redov časovne zahtevnosti

2.5 PRIMERI GRUP G_q

Multiplikativna grupa $\mathbb{Z}_p^* = \{1, \ldots, p-1\}$ ima red $p-1$. Pokazati se da, da je ciklična. Iz tega sledi, naslednja trditev.

Trditev 2.14. *Za vsak q, ki deli $p-1$, obstaja $g \in \mathbb{Z}_p^*$ reda q.*

Dokaz. Naj bo $qk = p-1$, za nek $k \in \mathbb{Z}$. Označimo z a generator $\mathbb{Z}_p^*$ in naj bo $g = a^k$. Red elementa g je q. ■

Naj bo q praštevilo. Množica $G = \{g^i \in \mathbb{Z}_p^*;\ i \in \{0, \ldots, q-1\}\}$ je očitno podgrupa $\mathbb{Z}_p^*$ in izpolnjuje vse pogoje za G_q. To bo torej naš prvi praktični primer grupe G_q. Najbolje je, če je $p = 2q+1$. Tako je p kar se da majhen pri danem q, s čimer zagotovimo največjo hitrost izvajanja operacij.

Drugi praktični primer bo grupa točk na eliptični krivulji. Podrobnejša razlaga eliptičnih krivulj je preobsežna in izven konteksta te diplome, lahko pa jo bralec poišče v [sciAJ97, ECT, Bar00, Mik01]. Mi bomo le na kratko skicirali glavna dejstva.

Definicija 2.15. *Eliptično krivuljo nad obsegom $\mathbb{R}$ definiramo kot množico točk*

$$E = \{(x, y) \in \mathbb{R} \times \mathbb{R};\ y^2 = x^2 + ax + b,\ a, b \in \mathbb{R}\}.$$

Če dodamo posebno točko $\mathcal{O}$ (točka neskončno), ki bo enota in definiramo še operacijo, dobimo grupo. Operacija je natanko določena, sestavljena iz osnovnih aritmetičnih operacij $(+, \cdot, /)$ in hitro izračunljiva, podrobneje pa je ne bom definiral.

Če zamenjamo $\mathbb{R}$ z nekim končnim obsegom dobimo končno množico točk, ki je za nas precej bolj zanimiva. Ta množica je tudi grupa za enako operacijo. Kriptografsko so za to najbolj primerni obsegi $\mathbb{Z}_p$ in $\mathbb{F}_{2^m}$ (druga oznaka je $GF(2^m)$ - Galoisovo polje).

Definicija 2.16. *Naj bo $E(\mathbb{F}_{2^m})$ grupa eliptične krivulje (nad končnim obsegom $\mathbb{F}_{2^m}$) moči n. Naj obstaja neka točka $P \in E(\mathbb{F}_{2^m})$ reda q, kjer je q praštevilo in $q|n$. Potem podgrupa H generirana s točko P ustreza vsem pogojem za G_q.*

Ker so grupe eliptičnih krivulj nad $\mathbb{F}_{2^m}$ od do sedaj znanih primerov grup G_q daleč najbolj primerne za kriptografijo [sciAJ97], jih bomo v prihodnje vzeli za naš model G_q. Model bomo potrebovali predvsem pri kalkulacijah učinkovitosti, sicer pa nam bo zadostovala že sama struktura G_q.

Primerjajmo še predstavljene grupe glede prostorskih in časovnih zahtevnosti. Za zapis elementa iz $\mathbb{Z}_p$ potrebujemo $\lceil \log_2 p \rceil$ bitov in prav toliko za zapis elementa iz podgrupe $\mathbb{Z}_p^*$ moči q. V najboljšem primeru je to $\lceil \log_2 (2q+1) \rceil \approx 1 + \log_2 q$, kar je le za bit slabše od spodnje meje, ki jo določa moč podgrupe q.
Podobno stanje je pri eliptičnih krivuljah. Za zapis elementa iz $\mathbb{F}_{2^m}$ potrebujemo m bitov, zato se zdi, da bomo za zapis točke iz $E(\mathbb{F}_{2^m})$ potrebovali $2m$ bitov (za vsako koordinato posebej). Vendar pa eni x koordinati ustrezata največ dve y koordinati. Tako obstaja postopek, da točko zapišemo le z $m+1$ biti. Ta postopek imenujemo kompresija, njegov obrat pa dekompresija točke [Mik01, Bar00]. Velikost podgrupe q je običajno $q \approx 2^{m-5}$, torej za zapis točke potrebujemo približno $6 + \log_2 q$ bitov, kar je še vedno blizu optimuma.

Časovne zahtevnosti operacij iz tabele 2.1 pri obeh grupah ustrezajo navedenim asimptotskim ocenam. Konkretne hitrosti so odvisne od implementacije, vsekakor pa so pri enakem q-ju primerljive.

2.6 PROBLEM DISKRETNEGA LOGARITMA V GRUPI G_q

Definicija 2.17. *Naj bo $a \in G_q \setminus \{1\}$ in $b \in G_q$. Poišči tako celo število x, da velja $a^x = b$.*

Opomba 2.18. *Zadostuje, da x iščemo na intervalu $0 \leq x \leq q-1$. $a^x = b$ lahko drugače zapišemo kot $x = \log_a b$. Problem ima vedno rešitev, saj po trditvi 2.6 vsak a generira celotno grupo G_q.*

Problem diskretnega logaritma bomo krajše pisali DLP (Discret Logarithm Problem).

Oglejmo si kratek številski primer. Zaradi lažje predstave bomo za model G_q vzeli podgrupo $\mathbb{Z}_p^*$.

Primer 2.19. *Vzemimo $p = 23$, $q = 11$, generator G_q je 2. Naj bo $a = 2$, $b = 3$. Da bi poiskali $x = \log_a b$ nam ne preostane kaj bistveno boljšega, kot da preizkusimo vseh 11 možnosti. Obratno se da dokaj hitro in brez odvečnih preizkušanj izračunati $2^8 = 3$.*

Ko večamo velikost grupe, razlika v zahtevnosti med potenciranjem in logaritmiranjem hitro narašča. Metoda preizkušanja z grobo silo odpove že pri srednje velikih grupah.

Velikost obsega $\mathbb{F}_{2^m}$ (v bitih)	Velikost parametra q (v bitih)	$\sqrt{\pi n/2}$	MIPS let
155	150	2^{75}	$3.8 \cdot 10^{10}$
210	205	2^{108}	$7.1 \cdot 10^{18}$
239	234	2^{117}	$1.6 \cdot 10^{28}$

Tabela 2.2: Čas potreben za rešitev problema DLP v odvisnosti od dolžine parametra q (podatki so za $E(\mathbb{F}_{2^m})$).

Splošno prepričanje je, da je problem DLP v grupi eliptične krivulje (krajše ECDLP) še mnogo težji kot pa v podgrupi $\mathbb{Z}_p^*$ ob podobno velikem q. Trenutno najboljši algoritem za reševanje problema ECDLP je *Pollardova ρ-metoda.* Ta metoda za rešitev posameznega problema potrebuje približno $\sqrt{\pi n/2}$ operacij. Označimo besedno zvezo *milijon ukazov na sekundo* s kratico MIPS (million instructions per second), stroj, ki je zmožen izvesti milijon ukazov na sekundo, pa imenujmo MIPS stroj. V praksi lahko MIPS stroj izvede približno $4 \cdot 10^4$ množenj v grupi

eliptične krivulje na sekundo. Zato je število množenj v grupi eliptične krivulje, ki jih MIPS stroj lahko izvede v enem letu, enako $(4 \cdot 10^4) \cdot (60 \cdot 60 \cdot 24 \cdot 365) \approx 2^{40}$. Tabela 2.2 nam za različne vrednosti q prikazuje potrebni čas za rešitev enega samega problema ECDLP (s Pollardovo ρ-metodo). Z oznako MIPS let smo označili število let, ki jih potrebuje MIPS stroj za rešitev problema ECDLP.
Za primer naj navedem, da bi 10 000 računalnikov, od katerih vsak zmore 1000 MIPS, potrebovalo 3800 let za rešitev enega samega problem ECDLP, kjer ima q le 150 bitov. Torej lahko, glede na današnje izglede razvoja kriptografije, problem ECDLP smatramo za nerešljivega za skoraj poljubno število let (ob izbiri primernega q).

Ostale grupe G_q (recimo podgrupa $\mathbb{Z}_p^*$) potrebujejo za enako zahtevnost DLP približno petkrat daljši q. Temu primerno so tudi počasnejše operacije in večje spominske zahteve. Pri večjih q-jih se to razmerje še slabša.

Poglavje 3

KRIPTOGRAFSKI ALGORITMI

V tem poglavju si bomo ogledali razne kriptografske algoritme, protokole in sheme, ki jih bomo potrebovali v nadaljnjih poglavjih. Najprej si bomo ogledali identifikacijske sheme, s katerimi lahko oseba potrdi svojo identiteto. To je nekako ekvivalentno temu, da nekomu pokažeš osebno izkaznico, ki je izdelana tako, da se je ne da ponarediti. Proučili bomo Schnorrovo in Okamotovo shemo ter razlike med njima. Prva je bolj preprosta, sta pa toliko podobni, da lahko brez težav uporabimo Okamotovo, če bi se Schnorrova izkazala za premalo varno. Identifikacijske sheme bomo potem pretvorili v sheme za digitalni podpis, s čimer bomo lahko podpisovali poljubna sporočila. To so splošni prijemi, ki veljajo za skoraj vse identifikacijske sheme, mi pa bomo ta in tudi vse nadaljne prijeme demonstrirali na Schnorrovi shemi. Nato bomo s tehniko Okamota in Ohte shemo za digitalni podpis prevedli na shemo za slepi podpis. Pri njej podpisnik ne more izvedeti kakšno sporočilo podpisuje, niti če mu pride v roke kdaj kasneje. Ker pa podpisniki običajno nočejo podpisovati povsem poljubnih sporočil, obstajajo tehnike, da lahko preverijo ali sporočilo ustreza določenim pogojem. Prva taka metoda je metoda naključnega preverjanja (Cut&Choose), ki pa zahteva ogromno odvečnega dela. Zato se je danes poskušamo izogibati, čeprav je najbolj splošno uporabna. Modernejše tehnike so učinkovitejše, temeljijo pa na omejenih slepih podpisih. Pri teh podpisnik prejemnika prisili, da lahko genenerira le točno določen tip sporočil, saj bo le tako podpis veljaven. Zato pa je večinoma treba za vsak nov tip sporočil iznajti novo shemo.

3.1 IDENTIFIKACIJSKE SHEME

Vse identifikacijske sheme, ki se jim bomo posvetili, rešujejo sledeči problem. Anita bi rada neko uslugo od Bojana, zato se ta želi najprej prepričati v njeno identiteto. Prav lahko bi se namreč napadalec pretvarjal, da je Anita in od Bojana izsilil uslugo. Anita ima svoj zasebni ključ, Bojan pa njen javni ključ, ki mora biti preverjeno pristen. V ta namen se Anita predstavi s certifikatom, ki ji ga je izdala neka zaupanja vredna certifikatna agencija. Na njem piše Anitina identiteta in njen javni ključ, vse skupaj pa je podpisano s strani certifikatne agencije. Anita bo potrdila svojo identiteto na način, da bo Bojana prepričala, da ima res zahtevan zasebni ključ, poleg tega pa ne bo razkrila nobene uporabne informacije o njem. Bojan tudi ne bo mogel ponoviti protokola in se tretji osebi predstaviti kot Anita.

3.1.1 SCHNORROVA SHEMA

Naj bo G_q dovolj velika grupa, da je problem DLP praktično nerešljiv in g naj bo generator. Anita si izbere $x \in_R \mathbb{Z}_q$, ki bo njen zasebni ključ ter izračuna $h = g^x$, ki bo njen javni ključ. Tega da Bojanu, običajno kot del certifikata. Protokol nato poteka sledeče:

1. Anita izbere $\omega \in_R \mathbb{Z}_q$, izračuna $a = g^\omega$ in ga pošlje Bojanu.

2. Bojan izbere $c \in_R \mathbb{Z}_q$ in ga pošlje Aniti (c lahko izbere tudi iz kake manjše podmnožice obsega $\mathbb{Z}_q$, saj to ne poslabša varnosti).

3. Anita izračuna $r = cx + \omega \mod q$ in ga pošlje Bojanu.

4. Bojan preveri $g^r = h^c a$. Če enakost velja, potem je identiteta Anite potrjena, sicer pa gre za goljufijo ali napako.

Opomba 3.1. *Oznaka $\in_R$ pomeni izbiro naključnega elementa. Podrobnosti izbire so opisane v razdelku 2.3.*

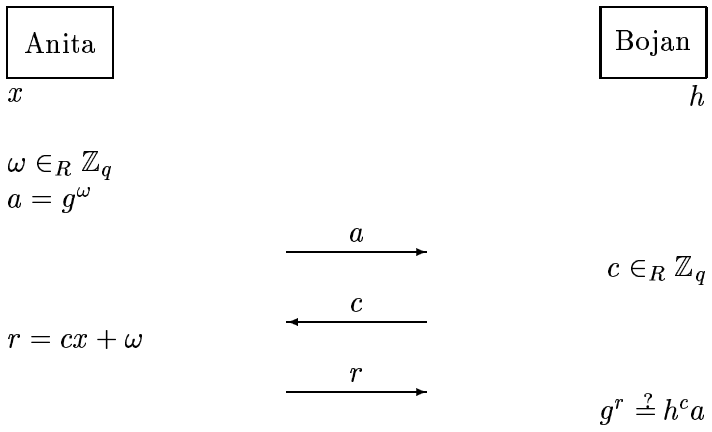

Diagram 3.1: Schnorrova identifikacijska shema

To je klasičen preizkus izziva in odgovora. Če bi se napadalec pretvarjal, da je Anita in bi Bojan izbral c, ki ga napadalec že pozna, bi mu lahko ta pravilno odgovoril, če bi slučajno pred tem izbral njemu pripadajoči a. Torej mora Bojan skrbeti, da izbere zares naključen c, medtem ko je Aniti to vseeno. Podobno mora Anita skrbeti, da izbere zares naključen ω. Če bi namreč dvakrat izbrala isti ω (torej tudi isti a), bi Bojan lahko izračunal njen zasebni ključ po formuli

$$x = \frac{r - r'}{c - c'}.$$

Če bi torej napadalec za določen a znal pravilno odgovoriti za vsaj dva izziva, bi bilo to enakovredno temu, da pozna zasebni ključ x.

Opomba 3.2. *Bojan se ne more nikomur predstaviti kot Anita, niti nikogar prepričati, da se je res pogovarjal z Anito. Prav lahko bi namreč le simuliral protokol, tako da bi si izmislil r in c ter nato izračunal $a = g^r h^{-c}$. Takemu protokolu pravimo zato* ***dokaz brez razkritja znanja*** *[Sti95].*

Ugotovili smo torej, da Anita zna potrditi svojo identiteto (**polnost**), vsak drug, ki zna to storiti z znatno verjetnostjo (z uporabo identifikacijskega protokola), pa bodisi pozna zasebni ključ x, bodisi ga zna izračunati v polinomskem času (**uglašenost**). To pa še ne pomeni, da je Schnorrova shema varna, saj ima protokol, kjer bi se Anita identificirala tako, da bi razkrila svoj privatni ključ, tudi obe zgornji lastnosti.

Definicija 3.3. *Pravimo, da je protokol* ***varen****, če napadalec tudi po polinomskem številu ponovitev identifikacijskega protokola ne izve nobene informacije o zasebnem ključu.*

Nihče še ni dokazal, da je Schnorrova shema res varna, vendar tudi niso znani učinkoviti napadi. Zato se smatra za varno in se jo v praksi uporablja.

3.1.2 OKAMOTOVA SHEMA

Ogledali si bomo še Okamotovo identifikacijsko shemo, ki je zelo podobna Schnorrovi, vendar je dokazljivo varna. Temelji na posplošitvi problema DLP, ki jo imenujemo **problem reprezentacije**. Najprej si oglejmo potrebne pojme in lastnosti.

Definicija 3.4. *Naj bodo fiksni $n \in \mathbb{N}$, G_q in $g_1, .., g_n \in G_q$, ki naj bodo paroma neenaki in različni od 1. Za poljuben $h \in G_q$ najdi tako n-terico eksponentov $(a_1, .., a_n)$, da velja $h = \prod_{i=1}^{n} g_i^{a_i}$. To n-terico imenujemo* ***reprezentacija****.*

Če je $n > 1$, potem za vsak h obstaja več reprezentacij, vendar je že poiskati eno običajno enako zahtevno kot problem DLP.

Trditev 3.5. *Problem reprezentacije je enako težak kot DLP v isti grupi G_q.*

Dokaz.

(problem reprezentacije $\Rightarrow$ DLP):

Če znamo reševati problem reprezentacije za stopnjo n, ga znamo tudi za stopnjo $n' < n$. To storimo tako, da odvečne g_i zamenjamo z g_1 (ali pa z znano potenco g_1, če se povsem držimo definicije). Nato iz rešitve za stopnjo n, z upoštevanjem osnovnih pravil za potenciranje, dobim rešitev za stopnjo n'. Rešitev za stopnjo 1 je tudi rešitev problema DLP.

(DLP $\Rightarrow$ problem reprezentacije):

Izračunamo naslednje logaritme.

$$x = \log_{g_1} h, \qquad x_i = \log_{g_1} g_i, \quad 1 < i \leq n.$$

Če enačbo reprezentacije logaritmiramo ($\log_{g_1}$), dobimo

$$x = a_1 + \sum_{i=2}^{n} a_i x_i.$$

Koeficiente a_i za $1 < i \leq n$ si recimo izberemo poljubne, a_1 pa izračunamo. Tako dobljena n-terica $(a_1, .., a_n)$ ustreza enačbi reprezentacije. ■

Trditev 3.6. *Naj bo $n = 2$ in naj za nek h poznamo dve različni reprezentaciji (a_1, a_2) in (a_1', a_2'). Potem lahko izračunamo $\log_{g_1} g_2$ v grupi G_q.*

Dokaz. Po predpostavki velja

$$g_1^{a_1} g_2^{a_2} = g_1^{a_1'} g_2^{a_2'},$$

iz česar sledi

$$\log_{g_1} g_2 = \frac{a_1 - a_1'}{a_2' - a_2}.$$

■

Opomba 3.7. *Vse operacije v tem poglavju se izvajajo v grupi G_q ali pa v obsegu $\mathbb{Z}_q$. Vedno je iz konteksta razvidno za katero strukturo gre.*

Oglejmo si sedaj Okamotovo shemo. Naj bo G_q dovolj velika grupa, da je problem reprezentacije praktično nerešljiv in (g_1, g_2) naj bo generatorska dvojica ($g_1, g_2 \in G_q$ sta generatorja). Vrednosti $\log_{g_1} g_2$ naj ne pozna nihče. Anita si izbere par $x_1, x_2 \in_R \mathbb{Z}_q$, ki bo njen zasebni ključ ter izračuna $h = g_1^{x_1} g_2^{x_2}$, ki bo njen javni ključ. Tega da Bojanu, običajno kot del certifikata. Protokol nato poteka sledeče:

1. Anita izbere $\omega_1, \omega_2 \in_R \mathbb{Z}_q$, izračuna $a = g_1^{\omega_1} g_2^{\omega_2}$ in ga pošlje Bojanu.
2. Bojan izbere $c \in_R \mathbb{Z}_q$ in ga pošlje Aniti
3. Anita izračuna $r_i = cx_i + \omega_i \mod q$ za $i = 1, 2$ in ju pošlje Bojanu.
4. Bojan preveri $g_1^{r_1} g_2^{r_2} = h^c a$. Če enakost velja, potem je identiteta Anite potrjena, sicer pa gre za goljufijo ali napako.

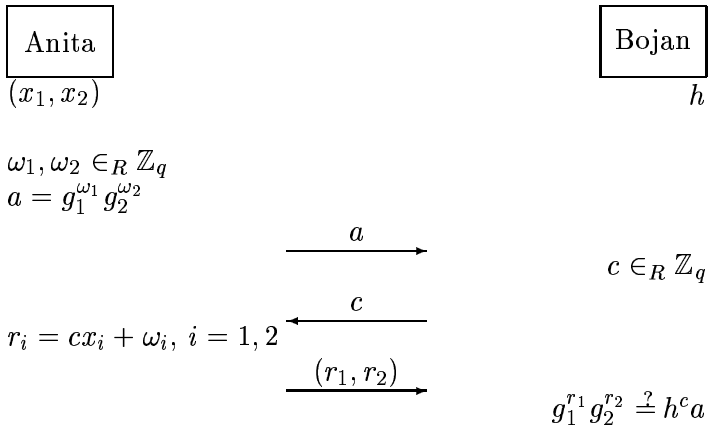

Diagram 3.2: Okamotova identifikacijska shema

Okamotova shema je polna, za razliko od Schnorrove pa zanjo znamo pokazati, da je tudi varna, vsaj kolikor je težak $\log_{g_1} g_2$ v G_q. Brez težav se preveri, da za dve različni reprezentaciji, ki ustrezata istemu javnemu ključu h, nihče ne more dokazati, katera je zasebni ključ, saj obe privedeta do pravilne izvedbe protokola. Če bi napadalka našla neko takšno reprezentacijo, bi ta bila skoraj gotovo (z verjetnostjo $1 - 1/q$) različna od zasebnega ključa Anite. Izbira katerekoli za zasebni ključ je namreč enako verjetna, vseh takih reprezentacij pa je natanko q. Če bi napadalka in Anita nato združili svoje moči, bi po trditvi 3.5 lahko izračunali $\log_{g_1} g_2$ v G_q.

Mi bomo uporabljali Schnorrovo shemo, ki je enostavnejša za uporabo, bolj učinkovita in bolj razumljiva. Če bi se kdaj kasneje pokazalo, da ni varna, je možno brez težav vse algoritme in protokole prilagoditi na uporabo Okamotove sheme.

3.2 PRETVORBA IDENTIFIKACIJSKE SHEME V SHEMO ZA DIGITALNI PODPIS

Vsako identifikacijsko shemo, ki temelji na izzivu in odgovoru, lahko prevedemo na shemo za digitalni podpis. Podpisnik lahko sam izračuna izziv kot enosmerno zgoščevalno funkcijo sporočila in ostalih parametrov, ki si jih je do takrat izbral. Ker gre za enosmerno funkcijo, nihče ne more najti drugačnega sporočila in parametrov, ki bi ustrezali istemu izzivu.

Definicija 3.8. *Enosmerne funkcije so funkcije f, ki so izračunljive v realnem času, medtem ko je računanje inverza računsko prezahtevno in neizvedljivo. Podrobneje: računsko nemogoče je za dani y najti x, tako da $f(x) = y$ razen, če smo y dobili kot sliko x.*

Definicija 3.9. *Pravimo, da je enosmerna zgoščevalna funkcija* ***šibko brez trčenj****, če je računsko prezahtevno za dani $f(x)$ najti tak y, da velja $x \neq y$ in $f(x) = f(y)$.*

Definicija 3.10. *Enosmerna zgoščevalna funkcija f je* ***krepko brez trčenj****, če je računsko prezahtevno najti par x, y, tako da velja $x \neq y$ in $f(x) = f(y)$.*

Za nas so pomembne zgoščevalne funkcije, ki preslikajo poljubno zaporedje poljubnih sporočil in parametrov v $\mathbb{Z}_{2^n}$, kjer je n vnaprej določen.

Opomba 3.11. ***Računsko prezahtevno*** *pri danem n pomeni, da je čas računanja inverza z najboljšim strojem, ki bi si ga lahko napadalec privoščil, daljše kot je napadalec pripravljen čakati (informacija namreč čez čas izgubi vrednost). Računsko prezahtevno pri poljubnem n pomeni, da je časovna zahtevnost računanja inverza eksponentna in da je pri danem zmernem (uporabnem) n inverz računsko prezahteven.*

Včasih zahtevamo od enosmerne funkcije še strožje pogoje, recimo da je **brez korelacij**. To lastnost je sicer težko definirati, pomeni pa, da ne smejo obstajati korelacije, kot je recimo $f(a) + f(b) = f(a + b)$. Za dokaze je pomemben še idealiziran model zgoščevalne funkcije, ki ga imenujemo model naključnostnega oraklja (random oracle model).

Definicija 3.12. ***Model naključnostnega oraklja*** *omogoča obstoj funkcij f, ki za izbrani x vrnejo naključni element $f(x) \in_R G$, kjer je G poljubna množica. Če večkrat izberemo isti x, vedno vrnejo isti $f(x)$, vendar preden prvič izberemo dani x, vrednost $f(x)$ sploh ni določena.*

Pri dokazih običajno za model enosmerne funkcije uporabljamo model naključnostnega oraklja. V praksi potem tako enosmerno funkcijo, zamenjamo s katero od preizkušenih zgoščevalnih funkcij, recimo s SHA-1 ali MD5. Taka shema potem seveda ni več dokazljivo varna, vendar to zadostuje za praktično uporabo.

Oznaka 3.13. *Enosmerno zgoščevalno funkcijo, ki sprejme poljubno število parametrov poljubnega tipa in jih preslika v $\mathbb{Z}_{2^n}$, kjer je $n \in \mathbb{N}$ vnaprej določen, označimo s $\mathcal{H}$. To velja za celotno diplomo.*

Pojem polnosti smo sicer že spoznali, ker pa ga bomo spet potrebovali pri tej in tudi drugih shemah in protokolih, ga definirajmo bolj splošno.

Definicija 3.14. *Shema (oz. protokol) je* ***polna****, kadar pripelje do željenega rezultata, če vsi sodelujoči izpolnjujejo zahtevane pogoje in se držijo sheme.*

V primeru digitalnih podpisov to pomeni, da bo podpis veljaven, če podpisnik pozna svoj zasebni ključ in se drži sheme. V primeru protokolov za digitalni denar (glej 6. poglavje) to pomeni, da bo uporabnik dobil uporaben kovanec, s katerim bo lahko plačeval, če se uporabnik, banka in trgovec držijo protokola.

Poglejmo si sedaj digitalne podpise s Schnorrovo shemo.

1. Anita izbere $\omega \in_R \mathbb{Z}_q$ in izračuna $a = g^\omega$.
2. Nato izračuna izvleček $c = \mathcal{H}(m, a)$, kjer je m sporočilo. Bodimo pozorni, da je $c \in \mathbb{Z}_{2^n}$.
3. Anita izračuna $r = cx + \omega \mod q$ in tvori podpis (c, r).
4. Kdorkoli dobi podpisano sporočilo $(m, (c, r))$ in ima Anitin javni ključ, lahko preveri $c = \mathcal{H}(m, g^r h^{-c})$. Če enakost velja, potem je podpis veljaven, sicer pa gre za goljufijo ali napako.

Dokaz polnosti. Vemo, da velja $a = g^r h^{-c}$. Iz tega sledi $c = \mathcal{H}(m, a) = \mathcal{H}(m, g^r h^{-c})$. Povsem mogoče je, da obstaja $b \in G_q$, $b \neq a$, tako da $c = \mathcal{H}(m, b)$, vendar pa je tak b računsko nemogoče najti, saj je $\mathcal{H}$ enosmerna zgoščevalna funkcija. ■

Opomba 3.15. *Dokaz velja za vsak n, tudi če $c \notin \mathbb{Z}_q$. Je pa večinoma najbolj smiselno, da je $2^n \approx q$.*

3.3 SLEPI PODPISI

V praksi se včasih zgodi, da bi želeli, da nam kdo podpiše sporočilo, ne bi pa želeli, da ga prebere. Problem bi lahko rešili tako, da bi sporočilo najprej zašifrirali z naključnim simetričnim ključem, ga dali podpisat, nato pa bi ključ priložili podpisanemu besedilu. To bi na pogled sicer rešilo zgornji problem, vendar bi si podpisnik lahko zapomnil podpis. Če bi kdaj kasneje naletel na besedilo z enakim podpisom, bi vedel, da gre za isto besedilo ter bi ga tudi lahko prebral. Pri pravih slepih podpisih mora biti ta možnost povezovanja onemogočena. Isti problem nastopi tudi, če podpisniku pošljemo le izvleček (digest)[1] besedila, ki nam ga da enosmerna zgoščevalna funkcija in ne originalnega besedila.

Definicija 3.16. *Shemo za digitalni podpis imenujemo shema za slepi podpis, če ustreza sledečemu opisu:*
Prejemnik si želi, da mu podpisnik podpiše sporočilo m'. Najprej pretvori m' v m in ga pošlje podpisniku. Ta mu v odgovor vrne p, ki je veljaven podpis za m. Prejemnik pretvori p v p', ki je veljaven podpis za m' in tvori podpisano sporočilo (m', p'). Če podpisnik kdaj kasneje naleti na to podpisano sporočilo, ne more ugotoviti, da gre za isto sporočilo (med (m, p) in (m', p') ni korelacije).

3.3.1 PRIMER S SCHNORROVO SHEMO

Okamoto in Ohta [OO89] sta predlagala splošen prijem, kako iz navadne sheme za digitalni podpis, naredimo shemo za slepi podpis. Poglejmo si slepe podpise s Schnorrovo shemo. Anita naj bo podpisnik, Bojan pa prejemnik.

1. Anita izbere $\omega \in_R \mathbb{Z}_q$, izračuna $a = g^\omega$ in ga pošlje Bojanu.
2. Bojan izbere $t_1, t_2 \in_R \mathbb{Z}_q$, izračuna izvleček $c' = \mathcal{H}(m, g^{t_1} h^{t_2} a)$ in pošlje Aniti $c = c' + t_2 \mod q$.

[1] Izvleček (angl. digest) in zgostitev (angl. hash) so enakovredni izrazi, ki vsi označujejo rezultat enosmerne zgoščevalne funkcije.

3. Anita izračuna $r = cx + \omega \mod q$ in ga pošlje Bojanu.

4. Bojan preveri $g^r = h^c a$. Če enakost velja, potem je Anita izdala veljaven podpis. Izračuna $r' = r + t_1 \mod q$ ter tvori podpisano sporočilo $(m, (c', r'))$.

5. Kdorkoli dobi podpisano sporočilo $(m, (c', r'))$ in ima Anitin javni ključ, lahko preveri $c' = \mathcal{H}(m, g^{r'} h^{-c'})$. Če enakost velja, potem je podpis veljaven, sicer pa gre za goljufijo ali napako.

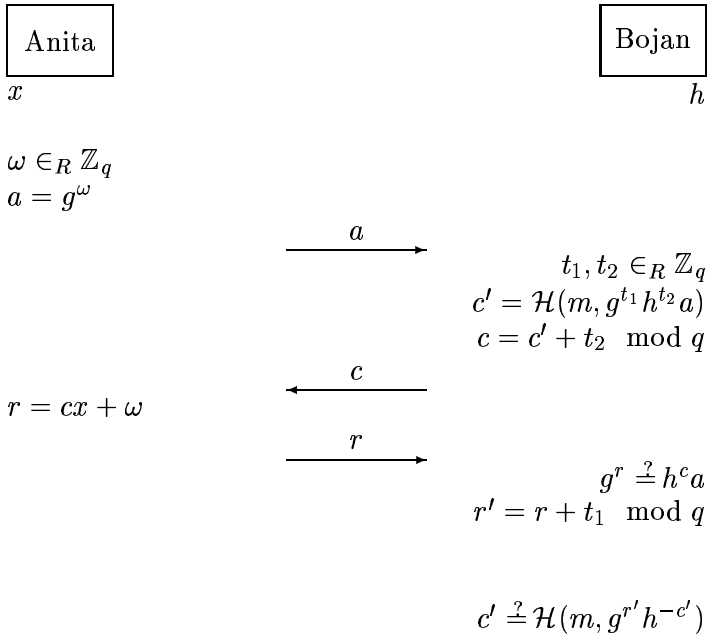

Diagram 3.3: Slepi podpis s Schnorrovo shemo

Dokaz polnosti. Vemo, da velja $a = g^r h^{-c}$ in $c' = \mathcal{H}(m, g^{t_1} h^{t_2} a)$.

$$g^{r'} h^{-c'} = g^{(r+t_1)} h^{-c+t_2} = g^{t_1} h^{t_2} a$$

Iz tega sledi $c' = \mathcal{H}(m, g^{r'} h^{-c'})$. ■

Ker Anita ne pozna t_1 in t_2, ne more po ničemer sklepati, da c in c' ter r in r' izhajajo iz istega podpisa. To velja za vsak n ($c \in \mathbb{Z}_{2^n}$).

3.3.2 METODA NAKLJUČNEGA PREVERJANJA (CUT & CHOOSE)

Pravzaprav redko se zgodi, da je podpisnik pripravljen podpisati povsem poljubno, njemu skrito sporočilo. Če to že stori, potem uporabi ključ, ki je namenjen le za točno določen namen uporabe (recimo za podpisovanje naključnih sporočil ali pa za podpisovanje kovancev za 10 SIT). Ko nek prejemnik dobi takšno sporočilo, preveri ali se vsebina sporočila ujema z namenom uporabe podpisnikovega ključa. Ta metoda je dokaj preprosta in ne zadostuje vedno. Zato so iznašli boljše metode, da lahko podpisnik preveri ali je sporočilo, ki naj ga podpiše res v pravem formatu in ali vsebuje zahtevane podatke.

Določeni deli sporočila naj bodo strogo predpisani, ostali pa poljubni. Podpisnik pred podpisom želi preveriti ali sporočilo ustreza predpisu, ne sme pa izvedeti ničesar drugega o sporočilu. Naj spomnim, da je treba sporočilo podpisati v enem kosu in da razbitje na več delov ni prava rešitev.

Metoda naključnega preverjanja temelji na preprosti ideji. Če je del sporočila res poljuben, lahko prejemnik zgenerira več enakovrednih različnih sporočil in jih pošlje podpisniku. Ta si enega naključno izbere in ga podpiše, še prej pa od prejemnika zahteva, da ostale razkrije, da podpisnik lahko preveri ali ustrezajo pravilom. Ker je možnost goljufije še vedno znatna, metoda deluje le, če so kazni za goljufe dovolj hude, da se jim ne splača poskušati.

Formalno metoda poteka sledeče:

1. Bojan (prejemnik) želi imeti k enakovrednih podpisanih sporočil. Zgenerira n sporočil in jih zakrije, vsakega s svojim zakrivnim faktorjem (n določi Anita, saj je od njega odvisna varnost sheme). Ta sporočila pošlje Aniti v podpis.

2. Anita izbere $n - k$ sporočil in pošlje Bojanu njihove zaporedne številke.

3. Bojan pošlje za vsako od izbranih sporočil njegov zakrivni faktor[2]. Včasih so deli sporočil še dodatno zakriti. Če jih je potrebno preveriti, mora Bojan poslati tudi njihove zakrivne faktorje, sicer pa lahko ostanejo zakriti.

4. Anita preveri ali vsa izbrana sporočila ustrezajo predpisom. Če to drži, podpiše k preostalih zakritih sporočil in jih pošlje Bojanu, sicer pa ne podpiše ničesar in ga lahko celo prijavi policiji zaradi goljufije.

Ocenimo sedaj učinkovitost te metode. Naj bo izmed n poslanih sporočil ℓ sporočil lažnih. Verjetnost, da Anita izbere kako lažno sporočilo je $1 - \binom{n-\ell}{k}/\binom{n}{k}$. Ta verjetnost je najmanjša, če je $\ell = 1$ (privzamemo, da Bojan goljufa). Potem se izraz poenostavi na $1 - (n-k)/n = k/n$. Ko Bojan izbere k mora Anita izbrati tak n, da je možnost goljufije k/n zanjo še sprejemljiva. Veliki n-ji pomenijo ogromno odvečnega dela in prenosa podatkov, zato je ta metoda uporabna le v primeru, ko je n zmerno velik, kar pa je možno le, če lahko napadalec z goljufivim sporočilom povzroči le zmerno škodo[3]. Možno je tudi, da bi Bojan poskušal prekriti svojo goljufijo in bi simuliral prekinitev zveze zaradi motenj, če bi Anita izbrala nelegalno sporočilo. Metodo bi se sicer dalo popraviti, da bi bila odporna na ta napad, vendar bi to naredilo metodo bolj zapleteno in manj splošno.

Poglejmo to še v praksi. Prvi sistemi za digitalni denar so uporabljali kovance, ki so vsebovali identiteto lastnika, vendar se je ta razkrila le, če je bil kovanec uporabljen dvakrat. Banke so izdajale kovance s podpisovanjem, pri tem pa so z metodo naključnega preverjanja preverjale ali kovanci res vsebujejo identiteto lastnika. Žal pa bi se dalo kovanec, ki bi bil plod uspešne goljufije (kovanec z lažno identiteto), vnovčiti poljubno mnogokrat, tako da le stežka govorimo o zmerni škodi. Banka so bile zato prisiljene uporabljati zelo velike n-je, kar pa je naredilo sistem v praksi neuporaben.

Kot smo videli je metoda naključnega preverjanja sicer res zelo splošna, vendar ogromna količina odvečnega dela in prenešenih podatkov povsem zasenči njene dobre plati. Kriptografi so zato mrzlično iskali boljše rešitve, ki si jih bomo ogledali v naslednjem razdelku.

3.4 OMEJENI SLEPI PODPISI

Pri omejenih slepih podpisih ima prejemnik možnost, da delno zakrije sporočilo, vendar ne more zakriti njegovega bistvenega dela ali lastnosti. Določen del sporočila (ali njegova lastnost) je

[2] Morda je lepši izraz zakrivni ključ. V tujih virih se sicer uporablja izraz blinding factor.

[3] Kaj je zmerno velik n in kaj je zmerno veliko škoda, je seveda odvisno od konkretnega primera.

torej **nezakriven** (blinding invariant). Rečemo tudi, da sporočilo ustreza določenim predpisom. V praksi to ni vedno mogoče, kljub temu pa si poglejmo dva običajna pristopa.

- **običajen slepi podpis + dokaz brez razkritja znanja**: Podpis poteka po običajni shemi za slepi podpis, vendar mora prejemnik potem, ko odda izziv, še z dokazom brez razkritja znanja potrditi, da sporočilo oz. poslani izziv res ustreza zahtevanim predpisom. Dokazi brez razkritja znanja so žal za mnoge tipe sporočil zelo neučinkoviti ali celo ne obstajajo, zato ta pristop za nas ni uporaben in se z njim ne bomo več ukvarjali.

- **del sporočila zgenerira podpisnik**: Pri tem pristopu, na zahtevo prejemnika, del sporočila zgenerira podpisnik. Prejemnik ima nato le še omejene možnosti spreminjanja in zakrivanja sporočila, saj bo podpis podpisnika odvisen od zgeneriranega (nezakrivnega) dela. Podpisano sporočilo bo zato uporabno le, če ni prišlo do nedovoljene manipulacije z nezakrivnim delom. Iz praktičnih razlogov je uporaba sporočila običajno omejena na predpisan **protokol uporabe** (showing protocol). Za drugačne načine uporabe podpisnik ne garantira ničesar (omejena namenska uporaba podpisnikovega ključa). Čeprav je ta pristop precej zoprn za uporabo, bomo vse naše nadaljne omejene slepe podpise gradili na njem.

Poglejmo si kar primer. Anita naj bo podpisnik, Bojan pa prejemnik. Bojan želi generirati sporočilo $m' = I^s$, kjer I označuje njegovo identiteto, Anita pa se želi prepričati, da je I res pravi. Anita ima zasebni ključ x in javni ključ $h = g^x$, ki ga pozna tudi Bojan. Sheme ne bom opisoval, oglejmo si kar diagram 3.4.

Dokaz polnosti. Podpis je veljaven, saj veljata enakosti, ki sta pogoj za veljavnost podpisa.

$$g^{r'} = g^{ru+v} = g^{(xc+\omega)u+v} = g^{(x\frac{c'}{u}+\omega)u+v} = (g^x)^{c'}(g^\omega)^u g^v = h^{c'}a^u g^v = h^{c'}a'$$

$$m'^{r'} = m'^{ru+v} = m'^{(xc+\omega)u+v} = (m^s)^{(x\frac{c'}{u}+\omega)u+v} = ((m^x)^s)^{c'}(m^\omega)^{su}m^{sv} = z'^{c'}b^{su}m'^v = z'^{c'}b'$$

■

Preden se lotimo dokaza omejenosti zakritja, poskusimo ta pojem še definirati.

Definicija 3.17. *Shema za digitalni podpis ima lastnost* ***omejenosti zakritja****, če prejemnik s polinomsko računsko močjo lahko pridobi veljaven par (sporočilo,podpis) le za sporočilo, ki izpolnjuje neke vnaprej predpisane lastnosti. Običajno to pomeni, da prejemnik ne more zakriti podpisniku določenega dela ali lastnosti sporočila ter da tega tudi po podpisu ne more spremeniti.*

Dokaz omejenosti zakritja. Ta del bomo le skicirali. Bojan bi lahko poskusil goljufati in bi poslal izziv, ki bi ustrezal nekemu drugemu sporočilu m_2. Če bi uporabil $c = \mathcal{H}(m_2)$, podpis ne bi bil uporaben, saj so podpisi z Anitinim zasebnim ključem x namenjeni le protokolom uporabe, kjer se preverijo spodnje tri enačbe iz diagrama 3.4. Če bi uporabil $c = \mathcal{H}(m'_2, z'_2, a'_2, b'_2)/u$, bi moral znati izračunati z'_2,a'_2 in b'_2. To pa je enakovredno temu, da pozna x. ■

Bojan je torej dobil sporočilo $m' = m^s$ na način, da Anita ne more povezati podpisanega sporočila m' z osnovnim sporočilom m ali ostalimi podatki iz procesa podpisovanja. Sporočilo $m' = m^s$ se zdi na prvi pogled neuporabno, saj nikomur nič ne pove (čeprav nosi informacijo), toda obstajajo protokoli, ki potrebujejo prav taka sporočila. Lahko bi recimo razvili protokol uporabe, kjer bi se vsebina m razkrila, če bi bilo sporočilo m' dvakrat uporabljeno. To lastnost bi lahko s pridom izkoristili za elektronske kovance.

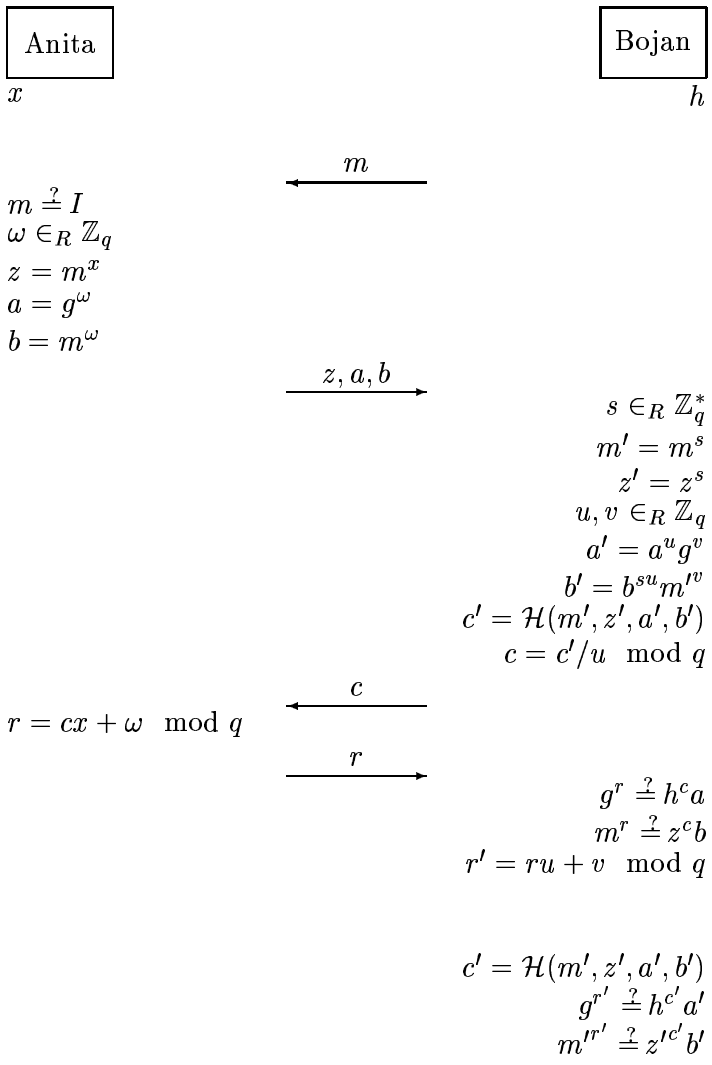

Diagram 3.4: Shema za omejeni slepi podpis sporočila $m' = m^s$

Videli smo, da metoda omejenih slepih podpisov v primerjavi z metodo naključne izbire omogoča bistveno učinkovitejše podpise, vendar imajo podpisana sporočila le zelo ozko področje uporabe. Metoda naključne izbire je pravo nasprotje, saj so podpisi zelo neučinkoviti, podpisana sporočila pa lahko uporabimo kjerkoli. Mi se bomo omejili le na ozka področja uporabe, kjer metoda omejenih slepih podpisov dobro deluje. To sta področji certifikatov s skrivnim ključem in digitalnega denarja.

Poglavje 4

CERTIFIKATI S SKRIVNIM KLJUČEM

V tem poglavju se bomo podrobneje seznanili s certifikati, s posebnim poudarkom na novem pojmu - certifikatih s skrivnim ključem. Najprej si bomo ogledali certifikate z javnim ključem, ki so že v široki uporabi. Pri njih izdajatelj jamči za verodostojnost javnega ključa. Za določene primere uporabe pa bi bilo koristno, da bi bilo jamstvo izdajatelja, za verodostojnost javnega ključa, veljavno le v primeru, da lastnik javnega ključa res pozna tudi pripadajoči skrivni ključ. Takim certifikatom pravimo *certifikati s skrivnim ključem*, saj izdajatelj nekako jamči za verodostojnost skrivnega ključa[1]. Podrobno jih bomo definirali in si ogledali primer, ki temelji na Schnorrovi shemi za digitalne podpise. Navedli bomo možnosti njihove uporabe, posebej pa se bomo posvetili aplikaciji omejenih slepih podpisov, za kar so še posebej primerni. Na uporabi omejenih slepih podpisov pri izdaji certifikatov s skrivnim ključem bo osnovan tudi sistem digitalnega denarja, ki bo osrednja tema 6. poglavja.

4.1 CERTIFIKATI Z JAVNIM KLJUČEM

Certifikat je par (informacija, podpis), kjer izdajatelj certifikata jamči za določeno lastnost informacije. Izdajatelj je običajno certifikatna agencija (CA - Certification Authority). Najbolj pogosti so certifikati, ki vsebujejo identiteto nekoga in razne pripadajoče dodatne informacije, recimo javni ključ te osebe. Tak certifikat bi bil trojica (identiteta osebe, javni ključ, podpis), kjer izdajatelj jamči, da dani osebi res pripada ta javni ključ. Poleg identitete osebe ali pa namesto nje bi lahko nastopali razni atribut (informacije poljubnega tipa). Primera bi bila (identiteta osebe, EMŠO, naslov, ..., javni ključ, podpis) ali pa (pravice dostopa, javni ključ, podpis). Slednji certifikat bi lastniku omogočal dostop do določenih virov ali sredstev. Ker na njem ni identitete lastnika, ampak so le atributi, mu pravimo **atributni certifikat**. To je nekakšen digitalni ekvivalent izkaznici, ki jo moraš pokazati vratarju.

Certifikati z javnim ključem so danes daleč najbolj razširjeni. Poglejmo si njihovo definicijo.

Definicija 4.1. ***Shema certifikatov z javnim ključem*** *je shema za digitalne podpise z dodatno lastnostjo, da mora sporočilo m vsebovati vsaj javni ključ p prejemnika $\mathcal{V}$, za katerega $\mathcal{V}$ pozna pripadajoči zasebni ključ s.*

[1] Izraza skrivni in zasebni ključ pomenita isto. Včasih želimo poudariti lastnost zasebnosti, včasih pa pomen skrivnosti, ki pa jo hkrati lahko javno uporabljamo. Tudi v angleščini se uporabljata izraza secret key in private key.

Par (p, σ) se imenuje ***certificiran javni ključ****, kjer je σ izdajateljev podpis sporočila m, trojica (s, p, σ) pa se imenuje* ***certificiran par ključev****.*

Opomba 4.2. *Sporočilo m običajno vsebuje tudi druge atribute, recimo identiteto lastnika certifikata (in javnega ključa).*

Omenjeni certifikati se danes uporabljajo povsod, kjer je potrebno zaupanje v javne ključe. Če prejemnik zaupa izdajatelju certifikata, potem verjame, da je lastnik javnega ključa res oseba navedena v certifikatu. Če bi se nekdo predstavil s tujim certifikatom, potem ne bi poznal pripadajočega zasebnega ključa in ne bi mogel razumeti prejetih sporočil in primerno odgovarjati. Pri pametnih komunikacijskih protokolih se tovrstno nepoznavanje zasebnega ključa razkrije že pred izmenjavo podatkov.
Certifikat si lahko osebi izmenjata, ko se predstavita, lahko pa jih že prej poiščeta v javno dostopnem imeniku.

4.2 OPIS CERTIFIKATOV S SKRIVNIM KLJUČEM

Včasih bi si želeli, da izdajatelj izda certifikat, s katerim bi garantiral verodostojnost skrivnega (zasebnega) ključa. Pri tem mu ga seveda ne želimo izdati, niti ne želimo, da bi bil vsebovan v certifikatu. K sreči je možno brez razkritja znanja dokazati nekomu, ki pozna naš javni ključ, da poznamo pripadajoči skrivni ključ. Certifikat bo torej vseboval javni ključ, toda izdajatelj ne bo jamčil zanj ampak za skrivni ključ. V praksi to izgleda tako, da lahko kdorkoli ponaredi veljaven certifikat za kak javni ključ, vendar na izbiro javnega ključa pri tem ne more toliko vplivati, da bi lahko izbral takega, za katerega bi poznal pripadajoči skrivni ključ. Certifikat s skrivnim ključem pa ima svojo vrednost le, če lastnik dokaže poznavanje skrivnega ključa. Le v tem primeru podpis izdajatelja res jamči za verodostojnost certifikata.

Definicija 4.3. *Shema certifikatov s skrivnim ključem sestoji iz naslednjih algoritmov:*

1. ***Algoritem za generiranje ključev izdajatelja (CA)****: Ta algoritem zgenerira zasebni ključ izdajatelja x_0 in njemu pripadajoči javni ključ h_0.*

2. ***Algoritem za generiranje ključev uporabnika*** $\mathcal{U}$*: Ta algoritem zgenerira zasebni ključ uporabnika x in njemu pripadajoči javni ključ h. Običajno je to isti algoritem, kot za generiranje ključev izdajatelja.*

3. ***Funkcija za preverjanje certifikata****: Obstaja funkcija T, tako da velja*

$$\textit{certifikat je pravilno generiran} \iff T(((h, I), \sigma), h, h_0) = 1$$

   kjer je $((h, I), \sigma)$ certifikat, I so atributi in σ podpis para (h, I). (Uporaba dodatnih atributov I v certifikatu seveda ni obvezna.)

4. ***Algoritem za izdajo certifikata****: Uporabnik $\mathcal{U}$ pošlje CA svoj javni ključ h in po potrebi dodatne atribute I. Ta mu vrne predpodpis σ_0. $\mathcal{U}$ nato glede na svoj skrivni ključ x pretvori σ_0 v pravi podpis $\sigma = f(\sigma_0, x, ...)$. Tako dobi pravilno generiran certifikat $((h, I), \sigma)$.*

5. ***Algoritem za simulacijo certifikata****: Ta algoritem za dani h_0 zgenerira pravilno generiran certifikat $((h', I), \sigma')$, vendar pa za h', za katere to zmore, le z neznatno verjetnostjo pozna tudi pripadajoči skrivni ključ x'. Verjetnostna porazdelitev (h', σ') se ne razlikuje od (h, σ), zato je nemogoče ugotoviti ali je certifikat pravi ali simuliran.*

Opomba 4.4. *Čeprav je certifikat **pravilno generiran**, to še ne pomeni, da je **veljaven**. Veljavnost je zagotovljena šele, če lastnik tudi dokaže, da pozna pripadajoči skrivni ključ.*

Nadaljne podrobnosti si bomo ogledali kar na primeru, dodatne informacije pa lahko bralec poišče v [Bra95c, Bra95d].

4.2.1 PRIMER UPORABE S SCHNORROVO SHEMO ZA DIGITALNE PODPISE

Poglejmo si kar omenjenih pet točk iz definicije.

1. **Algoritem za generiranje ključev izdajatelja (CA)**: Določi G_q, generator $g \in G_q$ in enosmerno zgoščevalno funkcijo $\mathcal{H}(\cdot)$. $x_0 \in_R \mathbb{Z}_q, \quad h_0 = g^{x_0}$.
2. **Algoritem za generiranje ključev uporabnika $\mathcal{U}$**: $x \in_R \mathbb{Z}_q, \quad h = g^x$.
3. **Funkcija za preverjanje certifikata**: Certifikat je formata $((h, I), (c, r))$, kjer je (c, r) Schnorrov podpis sporočila (h, I) glede na javni ključ $h_0 h$. Torej velja

   $$\text{certifikat je pravilno generiran} \iff c = \mathcal{H}((h, I), g^r (h_0 h)^{-c})$$

   (Uporaba dodatnih atributov I v certifikatu seveda ni obvezna.)
4. **Algoritem za izdajo certifikata**: Za izdajo certifikata $\mathcal{U}$ in CA izvedeta sledeče korake:
   (a) $\mathcal{U}$ izbere $\omega \in_R \mathbb{Z}_q$, izračuna $a = g^\omega$ in ga pošlje CA.
   (b) CA izbere $\omega_0 \in_R \mathbb{Z}_q$. Nato izračuna $c = \mathcal{H}((h, I), g^{\omega_0} a)$ in $r_0 = c x_0 + \omega_0$ ter pošlje (c, r_0) uporabniku $\mathcal{U}$.
   (c) $\mathcal{U}$ preveri $c = \mathcal{H}((h, I), g^{r_0} h_0{}^{-c} a)$. Če enakost velja, potem izračuna $r = r_0 + cx + \omega$ in dobi pravilno generiran certifikat $((h, I), (c, r))$.

   Dokaz polnosti. Pokazati moramo, da je certifikat pravilno generiran.

   $$c = \mathcal{H}((h, I), g^{r_0} h_0{}^{-c} a) = \mathcal{H}((h, I), g^{r_0+\omega} h_0{}^{-c}) = \mathcal{H}((h, I), (g^r h^{-c}) h_0{}^{-c} = \mathcal{H}((h, I), g^r (h_0 h)^{-c})$$

   ■
5. **Algoritem za simulacijo certifikata**: Certifikat se simulira tako, da se za h vzame $h = h_0{}^{-1} g^{t_1}$, za poljubna $t_1, t_2 \in \mathbb{Z}_q$. Potem je $((h, I), (c, ct_1 + t_2 \mod q))$, kjer je $c = \mathcal{H}((h, I), g^{t_2})$, pravilno generiran certifikat.

   Dokaz pravilnosti. Pokazati moramo, da je simuliran certifikat pravilno generiran.

   $$c = \mathcal{H}((h, I), g^{t_2}) = \mathcal{H}((h, I), g^{ct_1+t_2} (g^{t_1})^{-c}) = \mathcal{H}((h, I), g^{ct_1+t_2} (h_0 h)^{-c})$$ ■

   Dokaz enakosti porazdelitve $((h, I), (c, r))$ in $((h', I), (c', r'))$. Če je bila prej porazdelitev x in w enakomerna (v $\mathbb{Z}_q$) in neodvisna, potem je bila enakomerna in neodvisna tudi porazdelitev h in r (v G_q) v izdanem certifikatu. Če sta pri simulaciji t_1 in t_2 porazdeljena enakomerno in neodvisno, potem sta tudi simulirana h in r porazdeljena enakomerno in neodvisno. c sicer ni nujno porazdeljen enakomerno, vendar je v obeh primerih odvisen od h in zadnjega parametra $\mathcal{H}$, ki pa sta enakomerno porazdeljena in neodvisna. Torej je simuliran certifikat (po verjetnostni porazdelitvi) nemogoče ločiti od pravega. ■

   Opomba 4.5. *Dokaz za splošno porazdelitev sicer ne drži, vendar nas v praksi to ne zanima.*

Prikazana shema torej izpolnjuje pogoje iz definicije, toda to še ne zagotavlja varnosti. Če bi recimo pri izdaji certifikata uporabili standarden Schnorrov podpis (oz. $\omega = 0$), bi shema še vedno ustrezala definiciji, CA pa bi lahko izračunala uporabnikov skrivni ključ x, če bi kdaj kasneje dobila v roke izdani certifikat. Izračunala bi ga po formuli $x = c^{-1}(r - r_0)$. Prikazan modificiran Schnorrov podpis to onemogoča.

Privzemimo, da je enako težko ponarediti modificiran Schnorrov podpis kot standardnega. Potem se da dokazati naslednje štiri točke, s katerim definiramo varnost sheme certifikatov s skrivnim ključem:

1. Če $\mathcal{U}$ sprejme r_0, potem je $(x, (h, I), (c, r))$ certificiran par ključev.

2. Niti skupinska zarota ne more v polinomskem času ponarediti certificiranega para ključev (razen z zanemarljivo verjetnostjo).

3. Algoritem za simulacijo certifikatov je popoln. To pomeni, da se simuliranih certifikatov ne da ločiti od pravih.

4. Poznavanje certifikata in pogleda[2] CA (ω_0, r_0) ne omogoča razkriti več informacije o skrivnem ključu x, kot eno samo izvajanje Schnorrove identifikacijske sheme.

4.2.2 UPORABA CERTIFIKATOV S SKRIVNIM KLJUČEM

Certifikati s skrivnim ključem omogočajo nekaj specifičnih uporab, niso pa primerni za splošno uporabo. Tako ni njihov namen, da bi izpodrinili certifikate z javnim ključem. Poglejmo si sedaj področja, kjer se izkažejo njihove prednosti.

Imeniki javnih ključev : Imeniki certifikatov z javnim ključem so velika zbirka podpisanih sporočil. Napadalec lahko z njihovo pomočjo poskuša razbiti zasebni ključ CA (napad z znanimi pari (sporočilo, podpis)). Če bi certifikate zamenjali s certifikati s skrivnim ključem, imenik napadalcu ne bi več koristil. Enakovredno bi bilo, če bi ga s simulacijo zgeneriral kar sam, torej očitno ne nosi nobene informacije o zasebnem ključu CA.

Mehanizmi dostopa : Mnogi mehanizmi dostopa temeljijo na izdanih certificiranih parih ključev. CA npr. izda pametno kartico, ki vsebuje skrivni ključ in certifikat, v katerega so lahko vključeni tudi podatki o lastniku kartice. Uporabnik lahko tako kartico uporabi recimo za vstop v zgradbo ali pa za povezavo z oddaljenim računalnikom. Kartica se predstavi s certifikatom in brez razkritja znanja dokaže poznavanje skrivnega ključa. Če bi se uporabljali certifikati z javnim ključem, bi vratar lahko naknadno dokazal, kdo je vstopal v zgradbo, v našem primeru pa ne more, saj se da oba koraka simulirati.

Varno podpisovanje : Od načinov podpisovanja, ki varujejo zasebnost, so najpomembnejši omejeni slepi podpisi. Za certifikate z javnim ključem učinkovite sheme za omejene slepe podpise niso znane, za certifikate s skrivnim ključem pa so in se jih da učinkovito implementirati.

Digitalni denar : Omenjena področja uporabe se lahko združijo v učinkovit sistem digitalnega denarja, ki si ga bomo ogledali v 6. poglavju.

[2]Pogled je zbirka vseh podatkov, ki jih je lahko podpisnik pridobil (in po možnosti shranil) v procesu podpisovanja.

4.2.3 PROTOKOLI UPORABE IN DELEGIRANJE

Certifikati s skrivnim ključem (oz. certificirani pari ključev) zaživijo šele v povezavi s **protokoli uporabe**. To so protokoli, kjer uporabnik pokaže svoj certifikat in izvede kriptografsko akcijo glede na svoj javni ključ (preverljivo). Kriptografsko akcijo bomo šteli za **varno**, če jo uporabnik lahko izvede le, če pozna tudi pripadajoči skrivni ključ.

Če je kriptografska akcija precej podobna izdaji certifikatov, lahko napadalec to izkoristi. S sprotnimi zahtevami po novih certifikatih za točno določene javne ključe, bi izjemoma lahko izvedel kriptografsko akcijo tudi brez poznavanja skrivnega ključa. Temu napadu rečemo **delegiranje**, saj napadalec izkoristi CA za izvedbo svoje akcije. Pravilno konstruirane sheme so odporne na tovrsten napad in to velja tudi za shemo iz podrazdelka 4.2.1. Podrobnejšo razlago z mnogimi primeri lahko bralec najde v [Bra95d], matematično strožji pristop pa v [Bra00].

4.3 OMEJENI SLEPI PODPISI CERTIFIKATOV S SKRIVNIM KLJUČEM

Tokrat bomo predstavili precej drugačno vrsto omejenih slepih podpisov kot v razdelku 3.4. Uporabnik $\mathcal{U}$ bo dobil certificiran par ključev, kjer je $h' = {g_0}^{s_0}{g_1}^{s_1}$ javni ključ, (s_0, s_1) pa skrivni ključ. Podpisnik CA bo poznal in preveril nezakrivni del s_1, medtem ko o s_0 in h' ne bo vedel ničesar. V nezakrivni del s_1 so lahko zapisane tudi poljubne informacije. $\mathcal{U}$ dobi na koncu certifikat $(h', (c', r'))$, ki ga CA ne more povezati s svojim pogledom iz procesa izdaje certifikata.

Oglejmo si sedaj primer, ki temelji na Schnorrovi shemi. Čeprav si bomo ogledali le eno shemo, so vse ostale sheme v grobem zelo podobne (lahko pa za osnovo vzamejo kako drugo shemo, npr. Guillou-Quisquater-jovo).

Priprava : CA izbere G_q, generator $g_0 \in_R G_q \setminus \{1\}$, enosmerno zgoščevalno funkcijo $\mathcal{H}(\cdot)$ in zasebna ključa $x_0, x_1 \in_R \mathbb{Z}_q$. Nato izračuna $h_0 = {g_0}^{x_0}$ in $g_1 = {g_0}^{x_1}$. Podatke $(G_q, g_0, g_1, h_0, \mathcal{H}(\cdot))$ da v javnost kot parametre sistema.

Izdaja certifikata :

1. CA izbere $\omega \in_R \mathbb{Z}_q$, izračuna $a = g_0^w$ in ga pošlje CA.
2. $\mathcal{U}$ izbere $s_0, t_1, t_2 \in_R \mathbb{Z}_q$. Izračuna $h' = h{g_0}^{s_0}$, kjer je $h = {g_1}^{s_1}$ in $c' = \mathcal{H}(h', {g_0}^{t_1}(h_0 h)^{t_2} a)$. Nato zakrije $c = c' + t_2 \mod q$ in pošlje c CA.
3. CA podpiše $r = c(x_0 + x_1 s_1) + \omega \mod q$ in pošlje r.
4. $\mathcal{U}$ preveri ${g_0}^r (h_0 h)^{-c} = a$. Če enakost velja, izračuna $r' = r + t_1 + c' s_0 \mod q$ in dobi pravilno generiran certifikat $(h', (c', r'))$.
5. Kdorkoli lahko preveri, da drži $c' = \mathcal{H}(h', {g_0}^{r'}((h_0 h')^{-c'}))$.

Dokaz polnosti. Pokazati moramo enakost iz točke 5.

$$\begin{aligned} c' &= \mathcal{H}(h', {g_0}^{t_1}(h_0 h)^{t_2} a) \\ &= \mathcal{H}(h', {g_0}^{t_1}(h_0 h)^{t_2} {g_0}^{r}(h_0 h)^{-c}) \\ &= \mathcal{H}(h', {g_0}^{r+t_1+c's_0-c's_0}(h_0 h)^{t_2-c}) \\ &= \mathcal{H}(h', {g_0}^{r'}(h_0 h)^{-c'}({g_0}^{s_0})^{-c'}) \\ &= \mathcal{H}(h', {g_0}^{r'}(h_0 h')^{-c'}) \end{aligned}$$

■

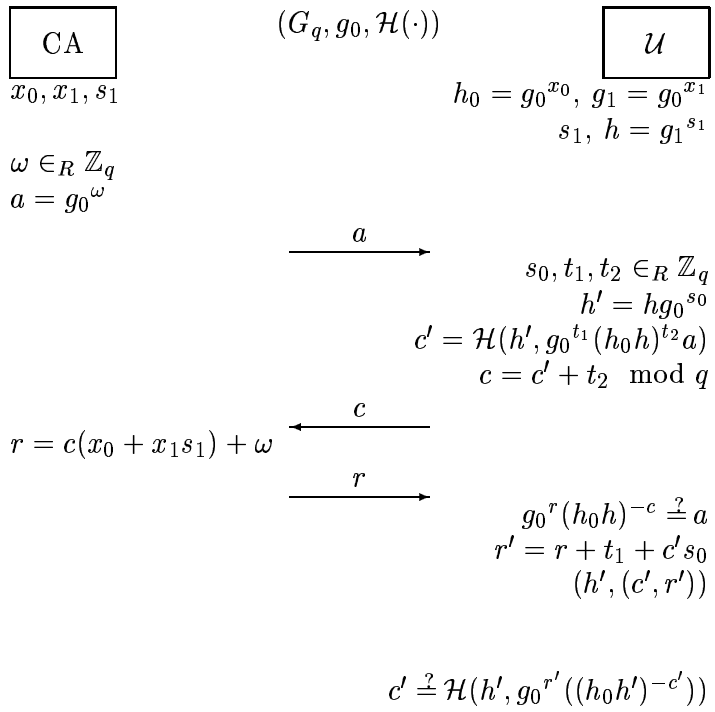

Diagram 4.1: Shema za omejeni slepi podpis certificiranega para ključev

S tem smo dokazali polnost sheme. Da bi dokazali, da je gornja shema res varna shema za omejene slepe podpise, nam manjkajo še sledeči dokazi:

- Shema varuje zasebnost uporabnika (s_0, t_1, t_2).
- Shema varuje zasebnost CA (x_0, x_1).
- Shema je algoritem za izdajo certifikata iz definicije 4.3 in če mu dodamo še manjkajoče algoritme, skupaj izpolnjujejo štiri pogoje za varno shemo certifikatov s skrivnim ključem.
- Omejenost in slepost : Uporabnik $\widehat{\mathcal{U}}$ nikakor ne more zakriti nezakrivnega dela s_1 v končnem certificiranem paru ključev, $\widehat{CA}$ pa ne more razkriti zakritega dela s_0. Zato $\widehat{CA}$ ne more povezati svojega pogleda in končnega certifikata.

Opomba 4.6. *Z $\widehat{\mathcal{Z}}$ označimo osebo, ki zmore izračunati največ polinomsko mnogo operacij. Ima recimo polinomsko mnogo časa in stroj fiksne moči.*

Dokaze omenjenih lastnosti za podobne sheme si lahko bralec ogleda v [Bra00, poglavje 4].

Izkaže se, da gornja shema še ni povsem varna, saj lahko dva napadalca, ki sočasno dvigneta certifikat, s skupnimi močmi ponaredita certificiran par ključev. Temu so lahko izognemo, če certifikate izdajamo le zaporedno (nikoli dva istočasno) ali pa z manjšo modifikacijo gornje sheme. Tu ne bomo šli v detajle, podoben primer pa si lahko ogledate v [Bra00, razdelek 4.4].

Gornja shema trenutno ne vsebuje nobenih dodatnih atributov I. Brez težav bi jih lahko dodali, vendar na način, da bi jih poznala oba ($\mathcal{U}$ in CA). Toda če bi atributi vsaj približno opisovali lastnika certifikata, bi CA lahko prek njih povezala svoj pogled s končnim certifikatom. S tem bi izgubili bistveno lastnost slepih podpisov.

Poglavje 5

DIGITALNI DENAR

Sedaj smo prišli do dveh bistvenih poglavij, kjer bomo spoznali digitalni denar. V tem poglavju bomo predstavili osnove, ki so skupne vsem sistemom digitalnega denarja. Najprej bomo pregledali zahteve in jih primerjali z lastnostmi, ki jih imajo različni sistemi. Nato bomo malo podrobneje opisali splošen sistem za anonimen digitalni denar. Na koncu bomo spoznali še razne razširitve, ki jih nekateri sistemi omogočajo.
Za uvod si oglejmo še kratek primer. Jože je len in se mu ne ljubi hoditi v banko, zato kar od doma (prek interneta ali telefona) dvigne nekaj tisočakov in jih da v svojo denarnico. Nato se odpravi v javno hišo, kjer precej denarja tudi porabi. Seveda si ne želi, da bi banka lahko izvedela za ta njegov skrivni izdatek in če bo pošteno ravnal z denarjem, se mu tega tudi ni treba bati. Ne glede na to, da Jože želi biti anonimen, pa banka lahko odkrije goljufa, če bi nekdo poskusil z istim denarjem dvakrat plačati. Čeprav se zdi to na prvi pogled nemogoče, bomo videli, da nam digitalni denar omogoča vse kar smo opisali in še marsikaj drugega.

5.1 OSNOVNE ZAHTEVE IN PRIJEMI

Poglejmo si najprej, kakšne so trenutne plačilne metode, da bomo lažje realno ovrednotili sisteme digitalnega denarja. Danes se pretežno uporabljajo trije načini plačevanja. Prvi in najstarejši je gotovinsko plačevanje. Z gotovino naj bi sicer lahko plačevali povsod in poljubne zneske, vendar gre skoraj vedno za manjše zneske. Gotovina je anonimna, zato pa je tudi primerna za ponarejanje. Pretvorbi nelegalno pridobljene gotovine v knjižni denar pravimo pranje denarja. Gotovina je tudi popolnoma prenosljiva. Takoj, ko jo nekomu damo, jo lahko on že uporabi. Ob izgubi ali kraji si škode ne moremo povrniti, vendar gre običajno za manjše zneske.
Drugi način plačevanja je knjižni denar. Nekomu denar nakažemo in dobimo potrdilo, ki mu ga po potrebi pokažemo. Pogosto tega niti ne zahteva, saj lahko sam preveri ali je nakazani denar že na računu. Nakazila so navadno neanonimna, v vsakem primeru pa ima banka vse podatke in celoten pregled nad prometom. Običajno to nikogar ne moti, saj smejo banke te podatke izdajati le na zahtevo sodišča. Knjižni denar ni popolnoma prenosljiv, saj prihaja do manjših časovnih zamikov pri nakazilih. Je dokaj varen, saj lahko uporabnik ob izgubi ali kraji bančno kartico takoj prekliče, da prepreči nadaljno škodo. Pravzaprav škode sploh ne bi smelo biti, če banke preverjajo osebne dokumente vsake stranke.
Tretji način plačevanje so kreditne in bančne[1] kartice, ki jih včasih napačno imenujemo plastični denar. To je še en način nakazovanja knjižnega denarja, ki pa je bistveno manj varen. Zaradi

[1] Bančne kartice se uporabljajo za poslovanje z banko, pa tudi za plačila pri trgovcih, ki so opremljeni s POS avtomati. Zato jih omenjamo pri dveh plačilnih metodah.

praktičnosti vseeno zamenjuje gotovino pri fizičnih plačilih in nakazila pri plačilih na daljavo. Varnost temelji le na posedovanju informacij, ki so zapisane na kartici. Kdorkoli ima te informacije, lahko z računa kreditne kartice potegne skoraj poljubno vsoto denarja. Pogosto se dogaja, da trgovci (posebej v manj razvitih državah) isti (ali podobni) račun vnovčijo po večkrat. Žal ni mehanizma, da bi dokazali ali goljufa trgovec ali uporabnik. Če se na račun nekega trgovca le nabere preveč pritožb, ga izključijo iz mreže kreditnih kartic. Tega si trgovci ne želijo in to je tudi VSE kar zagotavlja, da sistem deluje.

Gornje tri plačilne metode smo si pogledali zato, da vidimo, da lahko tudi sistemi, ki niso 100% varni, v praksi zadovoljivo delujejo. Sistem digitalnega denarja, ki ga bom predstavil, je bistveno bolj varen od plačilni kartic, praksa pa bo pokazala ali je celo bolj varen od ostalih dveh plačilnih metod. Zato lahko pričakujemo, da bi njegova uporaba v praksi potekala brez težav.

Vrnimo se sedaj spet k sistemom za digitalni denar in si poglejmo splošne osnove. Na splošno velja, da naj bi dober digitalni denar izpolnjeval naslednje zahteve:

1. **Neodvisnost**: Varnost digitalnega denarja ni odvisna od fizične lokacije. Hrani se lahko na poljubnem mediju in se prenaša preko računalniškega omrežja ali prek kake druge povezave.

2. **Varnost**: Digitalnega denarja ni mogoče ponarejati, niti ga nihče ne more večkrat uporabiti.

3. **Zasebnost**: Ob pošteni uporabi je zasebnost uporabnika zagotovljena. Nihče ne more povezati uporabnika in njegovega plačila, saj se uporabniku ni potrebno predstaviti niti trgovcu.

4. **Plačila brez povezave s centrom (off-line payments)**: Nihče ne potrebuje povezave z bančnim centrom, da bi lahko izvedel plačilo.

5. **Prenosljivost**: Digitalni denar se lahko prenaša med uporabniki. Denar, ki si ga plačal nekomu, lahko ta uporabi, brez da bi se moral za to povezati z banko.

6. **Deljivost**: Enoto denarja (elektronski kovanec) je možno deliti na manjše dele in vsak del porabiti ločeno, če vsota vrednosti delov ne preseže osnovne vrednosti enote.

Polega omenjenih šestih zahtev, ki se pogosto pojavljajo v literaturi, si želimo še praktičnost in učinkovitost. Obstajajo sistemi digitalnega denarja, ki izpolnjujejo vseh šest zahtev, vendar so običajno manj praktični in učinkoviti kot ostali, ki jih večinoma izpolnjujejo le pet. Tudi klasična gotovina izpolnjuje le pet zahtev, saj odpove pri deljivosti. Te pomanjkljivosti sicer ni imelo zlato, ampak kot vidimo to ni bil zadosten razlog, da ne bi njegova uporaba zamrla. Omenjene zahteve veljajo za običajno uporabo, pri določenih posebnih transakcijah pa so lahko nekatere celo nezaželjene. Včasih se recimo zahtevajo neanonimna plačila, kar je v nasprotju s tretjo zahtevo.

Glede na tretjo zahtevo delimo sisteme na anonimen in neanonimen denar. Če pogledamo tri klasične plačilne metode le gotovinska plačila zagotavljajo anonimnost. Ljudem se to očitno ne zdi preveč pomembno, v določenih primerih pa se celo zahteva neanonimno plačilo. Kljub temu bi večini ljudi ustrezal denar, ki bi omogočal anonimna plačila, vendar bi v primeru spora vsaka stran lahko dokazala, da ni goljufala.

5.1.1 NEANONIMEN DIGITALNI DENAR

Neanonimen digitalni denar, ki zahteva povezavo s centrom, pravzaprav že obstaja. Povežemo se na spletno poslovalnico svoje banke in nekomu nakažemo denar. Banka nam izda potrdilo, s katerim lahko po potrebi prejemniku dokažemo, da smo plačilo res izvedli. Naj še enkrat opomnim, da tu ne gre za pravi denar ampak le za plačilno metodo.

Pravi neanonimen digitalni denar pa bi lahko izgledal takole:

1. Banka $\mathcal{B}$ izda uporabniku $\mathcal{U}$ zahtevano število elektronskih kovancev[2]. Elektronski kovanec je sporočilo, ki vsebuje identiteto uporabnika, vrednost kovanca in njegovo serijsko številko ter ga je banka podpisala. Pred tem je seveda preverila identiteto uporabnika in odštela zahtevan znesek z njegovega računa.

2. Ko želi uporabnik $\mathcal{U}$ trgovcu $\mathcal{T}$ plačati dogovorjen znesek, vzame iz svoje elektronske denarnice potrebno število kovancev[3], jim doda trenutni čas in datum, identiteto trgovca ter naključno število, ki ga zgenerira trgovec in vse skupaj podpiše. Takšne kovance nato preda trgovcu v plačilo. Ta preveri podpis banke in uporabnika in če je vse v redu plačilo sprejme.

3. Trgovec prinese kovance v banko in jih želi položiti na svoj račun. Banka preveri ali je kovance res sama izdala ali je uporabnikov podpis pravi in ali je identiteta trgovca enaka tisti na kovancih. Če je vse v redu, si banka v svojo bazo zabeleži serijske številke kovancev in pripadajoče kovance. Če teh serijskih številk še ni v bazi, potem ti kovanci še niso bili vnovčeni in jih izplača. Ko bo kdaj kasneje kdo hotel vnovčiti kovanec z isto serijsko številko, bo vedela, da gre za ponaredbo. Če bo povsem enak tistemu v bazi, bo vedela, da goljufa trgovec. Če pa se bo razlikoval vsaj v naključnem številu, ki ga je izbral trgovec, potem je krivda na strani uporabnika. V obeh primerih bo seveda goljufa odškodninsko in kazensko preganjala.

Prva in četrta zahteva digitalnega denarja sta izpolnjeni, tretja pa očitno ne. Kar se tiče varnosti, je sistem toliko varen, kolikor so varni zasebni ključi, ki jih osebe uporabljajo za digitalne podpise in identifikacijo. Sistem s to stopnjo varnosti bomo v nadaljevanju šteli za varen sistem. Prenosljivost je mogoče doseči tako, da trgovec dobljene kovance uporablja naprej na enak način, kot jih je pri plačilu uporabnik. Žal pri tem prijemu kovanci rastejo pri vsakem plačilu, zato jih je treba kmalu nesti v banko, ki jih zamenja za nove. Temu se ne da izogniti pri nobenem sistemu [CP93a].

Deljivost bi lahko dodali, če bi uporabnik pri plačilu navedel tudi kateri del kovanca želi porabiti (to bi dodal v del, ki ga podpiše). Interval $[0, 1]$ bi recimo pomenil celoten kovanec, interval $[0.2, 0.6]$ pri kovancu za 100 SIT pa le 40 SIT. Banka bi lahko zaznala, če bi prišlo do prekrivanja intervalov in goljufa preganjanja.

Omenjenih lastnosti sistema ne bomo dokazovali, saj ta sistem služi le kot primer. Ker se bomo v nadaljevanju precej podrobneje posvetili podobnim problemom, bodo bralcu gornje navedbe verjetno postale očitne.

[2] V elektronskem svetu ni razlike med bankovci in kovanci. Zato bomo vedno govorili le o elektronskih kovancih.

[3] Kadar bo iz konteksta razvidno, da gre za elektronske kovance, bomo izpuščali oznako elektronski. Enako velja tudi za čeke in gotovino.

5.1.2 ANONIMEN DIGITALNI DENAR

Neanonimen digitalni denar nam je uspelo enostavno implementirati s klasičnimi kriptografskimi algoritmi, predvsem z digitalnim podpisom. Da dodamo anonimnost, moramo shemo bistveno spremeniti in uporabiti novejše tehnike, predvsem omejene slepe podpise. Večina shem za anonimen digitalni denar je v grobem podobna Brandsovi shemi, ki jo bomo obravnavali v 6. poglavju. Zato si bomo v tem razdelku najprej pogledali lastnosti, ki so skupne vsem. Nehali bomo tudi govoriti o anonimnih in neanonimnih shemah, ampak se bomo omejili le na anonimne, zato tega ne bomo več izrecno navajali.

V grobem izgledajo vse uporabne sheme tako:

1. **Priprava sistema (setup)** • Banka $\mathcal{B}$ izbere določene skupne parametre in svoj zasebni ključ. Teh je lahko tudi več, recimo za vsako vrednost kovancev svoj zasebni ključ. To sicer delno zmanjšuje škodo ob kraji ali razbitju ključa, kljub temu pa so zasebni ključi banke izjemno dragocena stvar, ki jo je treba maksimalno varovati.

2. **Odprtje (digitalnega) računa** • Uporabnik $\mathcal{U}_i$ si izbere svoj zasebni ključ, izračuna javni ključ in ga nese v banko. Ta preveri uporabnikovo identiteto, mu odpre račun in poleg ostalih podatkov zabeleži tudi njegov javni ključ. Običajno mu dodeli tudi identifikacijsko številko u_i, ki je lahko kar številka računa. Če uporabnik že ima pri banki odprt klasičen račun, odprtje novega računa seveda ni potrebno. Odprtje digitalnega računa zahteva od uporabnika, da se fizično oglasi v banki, izjema je le, če je banka že prej preverila uporabnikovo identiteto ter lahko z njim komunicira po varnem kanalu. Ob odprtju računa si uporabnik običajno priskrbi tudi potrebne javne ključe banke, čeprav bi lahko to storil tudi kdaj kasneje.

3. **Dvig denarja** • Uporabnik $\mathcal{U}_i$ se banki najprej predstavi in potrdi svojo identiteto, nato pa zahteva dvig določene količine denarja. Točno specificira, koliko katerih kovancev bi želel. Lahko zahteva tudi čeke ali kako drugo obliko. Banka preveri ali je uporabnik upravičen do zahtevanega dviga in če je, mu ga izplača. To stori tako, da z uporabo sheme za omejene slepe podpise podpiše kovance, ki jih je uporabnik zgeneriral. Pri tem ne dobi nobene informacije o tem, kako bodo izgledali kovanci, ko jih bo uporabnik uporabljal, zagotovi pa si, da bo podpis veljaven le, če so kovanci pravilno konstruirani. To med drugim pomeni, da bodo vsebovali podatek o identiteti uporabnika (npr. u_i), ki je nujno potreben za razkritje identitete goljufa, če je isti kovanec večkrat uporabljen. Izplačane kovance uporabnik shrani, poleg njih pa še nekatere informacije, ki jih je uporabljal pri konstrukciji kovancev. Le s pomočjo teh dodatnih informacij je možno kovance uporabiti kot plačilno sredstvo. Nekatere sheme zahtevajo pri dvigu uporabo varnega kanala, druge pa so varne tudi brez njega. Kljub temu se običajno, zaradi varovanja zasebnosti, z banko komunicira prek varnega kanala.

4. **Plačilo** • Uporabnik $\mathcal{U}_i$ želi trgovcu $\mathcal{T}_j$ plačati določen znesek. Najprej določi s katerimi kovanci bo plačal, nato za vsak kovanec prosi trgovca za izziv. To je psevdonaključno število, ki ga zgenerira trgovec. Pomembno je, da se nikoli ne zgodi, da bi isti kovanec (če bi goljuf večkrat uporabljal isti kovanec) dobil isti izziv. Zato je najbolje, da trgovec nikoli ne izbere dveh enakih izzivov, kar običajno zahteva tudi banka. Izziv bi bil lahko naključno število, običajno pa je zgostitev (hash oz. digest) kovanca, identitete trgovca in nekega števca, ki se spremeni za vsak prejeti kovanec, lahko pa bi dodali še kaj (npr. čas in datum). Informacijo, ki je parameter zgoščevalni funkciji, označimo s $\texttt{spec}$ ($\texttt{spec} = (\textit{kovanec}, \mathcal{T}_j, \textit{unikatni števec})$, $d = \mathcal{H}(\texttt{spec})$). Teoretično bi bilo možno,

da bi zgostitvena funkcija za dve različni vrednosti spec proizvedla enaka izziva, vendar je pri uporabi standardnih zgoščevalnih funkcij ta verjetnost zanemarljiva[4].

Ko uporabnik dobi izzive, iz vsakega kovanca, izziva in dodatne informacije o generiranju kovanca izračuna **plačilni dodatek**. Nato pošlje trgovcu vse kovance in pripadajoče plačilne dodatke. Ta lahko preveri veljavnost kovancev in ali so plačilni dodatki pravilno zgenerirani glede na izziv. To lahko stori ne glede na to, da mu dodatne informacije o generiranju kovanca niso znane. Če ne ugotovi nobenih nepravilnosti, sprejme plačilo in uporabniku izda račun oz. potrdilo. Naj omenim še, da uporabniku nikjer pri plačilu ni treba razkriti svoje identitete, trgovec pa jo običajno mora, vendar mu banka dovoljuje uporabo psevdonima.

5. **Polog** • Trgovec $\mathcal{T}_j$ se poveže z banko $\mathcal{B}$ in želi položiti prejet denar. Če z banko še ni posloval, se mora najprej registrirati, pri tem pa sme uporabiti psevdonim. Trgovec nato banki pošlje banki vse kovance, pripadajoče plačilne dodatke in pripadajoče vrednosti spec. Banka podatke preveri ter pogleda v bazo ali je bil kak kovanec že vnovčen. Če so podatki pravilni in ni sledu goljufije, banka vstavi kovance in ostale informacije v bazo izplačanih kovancev ter nakaže denar na račun, ki ga izbere trgovec. Običajno je to njegov digitalni račun pri isti banki, lahko pa bi si izbral tudi kak drug način plačila.

V primeru goljufije je postopek podoben kot pri neanonimnem denarju. Če ima sporen kovanec enak spec kot tisti v bazi, potem goljufa trgovec in banka mu ne izplača denarja. V nasprotnem primeru goljufa uporabnik in banka lahko iz dveh različnih plačilnih dodatkov izračuna njegovo identiteto u_i. Škodo pokrije z njegovega računa (če ni že prazen) in ga še sodno preganja. Seveda je možno, da mu je kdo ukradel zasebni ključ in s tem identiteto ter je goljufal v njegovem imenu. Za varovanje zasebnega ključa obstajajo posebni postopki in naprave, ki jih bomo tudi omenili v nadaljevanju, sicer pa je naloga sodišča, da presodi to možnost.

5.1.3 VARNOST

Glavne varnostne prijeme smo uvedli že v prejšnjem podrazdelku. Uporabnik mora paziti, da ne uporabi dvakrat istega kovanca, trgovec pa, da pravilno izbere (oz. generira) izziv. Oba sta motivirana, da se pravilno vedeta, sicer bi bila obtožena goljufije. Zato sistem deluje in to varno. Za konkreten sistem je treba seveda še dokazati, da ne obstajajo druge poti za ponarejanje ali večkratno uporabo (double spending).

Uporabnik se je dolžan držati pravil in imeti nadzor nad kovanci, kar je za človeka malo prezahtevno. Lahko bi imel kovance spravljene na osebnem računalniku, kjer bi poseben program skrbel za nadzor in manipulacijo z denarjem. To bi bilo sicer uporabno za plačila prek spleta, fizičnih plačil pa ne bi omogočalo. Poleg tega je tudi varnost osebnih računalnikov včasih vprašljiva, saj na njih teče veliko nepreverjenih programov. Zato je bolje, da denar hranimo v posebej za to namenjeni napravi imenovani digitalna denarnica ali kratko **denarnica**. To je majhen računalnik, ki hrani kovance in druge oblike denarja, zna plačevati in izvajati druge potrebne operacije. Ves čas skrbi, da ne pride do nepravilnosti pri poslovanju, pa tudi, da ne goljufa kdo drug. Uporabnik ji mora zaupati, lahko pa izbira med večimi proizvajalci ali si celo naredi svojo.

[4]Če ima zgoščevalna funkcija zgostitev (oz. izvleček) dolgo n bitov, potem je zaradi paradoksa rojstnih dni [Sti95] verjetnost za gornji dogodek $2^{-n/2}$. Pri uporabi standardne zgoščevalne funkcije SHA-1, ki ima zgostitev dolgo 160 bitov, pomeni to verjetnost 2^{-80}, kar je zanemarljivo.

Po velikosti je ponavadi malo večja od plačilnih kartic. Ima tipkovnico in infrardečo komunikacijo. Denarnico odklenemo s PIN[5] kodo, nato pa lahko nekaj minut z njo plačujemo zneske, ki jih vtipkamo in potrdimo. Denarnica komunicira s plačilno napravo prek infrardeče povezave, zato nam je ni potrebno dajati trgovcu, kot je to v navadi pri kreditnih karticah. Dvig denarja poteka na bankomatih ali prek spleta. Zanj potrebujemo drugo, daljšo in pomembnejšo PIN kodo, tako da se nam pri plačevanje ni treba preveč bati, da nas kdo opazuje pri vnašanju kode. Opisan model denarnice je bil razvit leta 1995 pri projektu CAFE [BBC+94, CAFa]. V prihodnje se bodo gotovo pojavili še bolj izpopolnjeni modeli, recimo pametne kartice, ki jim bo za avtentikacijo[6] uporabnika zadoščal npr. prstni odtis.

Zlorabe pa so še vedno možne, saj si lahko goljuf naredi svojo goljufivo denarnico. Banka ga bo sicer naknadno odkrila, vendar včasih denarja za povzročeno škodo ni mogoče izterjati. Lahko je goljuf pobegnil v kako daljno deželo ali pa je denar preprosto zapravil.

Da se banke izognejo temu tveganju, pridejo v poštev **denarnice z nadzornikom** (wallets with observers). Nadzornik je majhna pametna kartica, ki jo vstavimo v našo izbrano denarnico. Če je sistem zasnovan za uporabo nadzornikov, potem denarnica brez njega sploh ni uporabna. Nadzornik mora sodelovati pri vsakem dvigu in plačilu. Če bomo hoteli isti kovanec porabiti drugič, bo nadzornik odklonil sodelovanje in plačila ne bomo mogli izvesti. Nadzornik mora biti sicer odporen na vdore (tamper resistant), vendar se na to ne smemo 100% zanesti. Je le majhna elektronska naprava in če bi se komu vendarle posrečilo, da bi vdrl vanj in znal izvajati večkratna plačila z istim kovancem, mora biti banka vseeno sposobna identificirati goljufa. Nadzornik je le nadgradnja sistema, ki mora tudi brez njega ustrezati prej opisanim zahtevam.

Dobri sistemi za anonimen digitalni denar pa omogočajo stopnjo anonimnosti, ki nekatere skrbi. Bojijo se namreč, da bi to olajšalo pranje denarja ter omogočalo varnejše izsiljevanje in ilegalno trgovino. Osebno mislim, da to ni resna grožnja in da bi kriminalci pač uporabljali klasične preizkušene metode, če bi jim preprečili uporabo digitalnih. Kljub temu so se pojavili nekateri sistemi [CPS96, FTY97, SPC95], ki naj bi omogočali **pošten digitalni denar**. Ideja je v tem, da banka sama ne more povezovati dvigov in pologov, da pa lahko to stori sodišče - seveda ob pomoči banke. Sodišče bi zagotavljalo anonimnost v normalnih pogojih, ob sumu kaznivega dejanja pa bi lahko izrabilo svoj priviligiran položaj. Osebno mislim, da ti sistemi niso perspektivni in da zmanjšujejo zaupanje v denar, zato se z njimi ne bomo ukvarjali. Je pa verjetno možno večim obstoječim sistem dodati to funkcionalnost.

Povračilo ob izgubi je lastnost, ki se mi zdi precej bolj pomembna. Uporabniki bi si močno želeli, da bi ob kraji ali izgubi denarnice bilo možno v doglednem času denar dobiti nazaj. Podoben problem lahko nastane, če zaradi težav pri prenosu nismo prejeli kovancev, za katere banka trdi, da smo jih. Nekateri sistemi omogočajo, da nam banka povrne neporabljene kovance, če ji priskrbimo določene informacije, ki smo ji uporabljali pri konstrukciji kovancev. Te informacije moramo imeti spravljene seveda na nekem drugem, varnem mestu, da jih ne izgubimo skupaj z denarnico. Lahko jih recimo zašifriramo z nekim izbranim simetričnim ključem in jih spravimo kar v banki ali pri kaki drugi instituciji. Potem moramo hraniti le kopijo ključa (original je v denarnici), ki je lahko dolgotrajen. Tako kopijo je najbolje shraniti kar v sef. Izbrani podatki morajo imeti lastnost, da le banka in uporabnik skupaj lahko rekonstruirata kovance in jih položita na račun. Brez pomoči banke so podatki nekoristni (banka nudi pomoč seveda le lastniku kovancev).

[5]Personal Identification Number - osebna koda, ki služi za avtentikacijo uporabnika. Uporabljamo jih na primer pri bankomatih in GSM aparatih.

[6]Avtentikacija pomeni postopek preverjanja ali je identiteta določene osebe res pristna (ali je sogovornik res oseba za katero se izdaja). Avtorizacija pomeni postopek preverjanja ali določena oseba ima zahtevane pravice. Običajno se pred avtorizacijo izvede še avtentikacija.

Največji problem pri povračilu kovancev je, da uporabnik običajno ne ve katere kovance je že porabil. Prav lahko bi pomotoma ali namenoma zahteval povračilo kovancev, ki jih je že porabil, kar seveda ni dovoljeno. Zato se običajno počaka nekaj časa, da trgovci vnovčijo večino prejetih kovancev. Nato banka uporabniku vrne denar za zahtevane izgubljene kovance, razen za tiste, ki so že bili vnovčeni. Če kasneje kdo vnovči katerega izmed izgubljenih kovancev, ga banka izplača z računa uporabnika.

Ta postopek zahteva določeno stopnjo zaupanja, ki jo mora banka imeti do uporabnika. Če postopek primerjamo s stopnjo zaupanja pri uporabi današnjih kreditnih kartic, pa se zdi, da bi vendarle moral delovati.

5.1.4 ZASEBNOST

Dejali smo, da z digitalnim denarjem želimo ohraniti zasebnost. Podrobneje ločimo dve lastnosti:

- **nesledljivost** (untracebility): Kovancem ni mogoče slediti v preteklost. To pomeni da nihče ne more izvedeti, kdo je z njim plačal (ali plačeval). Identiteta uporabnika ostane skrita.
- **nepovezljivost** (unlinkability): Plačil ni mogoče povezovati med seboj. To pomeni, da ni mogoče ugotoviti, da bi dve plačili, ob različnih časih ali krajih, izvedla ista oseba ali pa izvedeti kake podobne informacije.

Nepovezljivost je očitno močnejša zahteva kot nesledljivost. Kdor bi namreč lahko izsledil identitete uporabnikov kovancev, bi lahko brez težav poiskal povezave med plačili. Naj še omenim, da gornji definiciji veljata le za poštene uporabnike. Ob goljufiji se identiteta goljufa razkrije, da ga lahko izsledimo in tudi povežemo z drugimi goljufijami.

Sodobni sistemi za elektronske kovance nam vsi zagotavljajo nesledljivost. Navadni elektronski kovanci nam nudijo tudi nepovezljivost. Tej lastnosti pa se običajno odpovemo, če želimo dodati prenosljivost in/ali deljivost, saj bi sicer sistem postal preveč neučinkovit (razlaga v naslednjih podrazdelkih). Kogar to moti, lahko še vedno uporablja navadne kovance.

5.1.5 PRENOSLJIVOST

Želeli bi prenosljivost denarja, zraven pa bi radi ohranili anonimnost in varnost. To pomeni, da bo moral tak kovanec vsebovati zakrite identitete vseh uporabnikov, ki jim je šel "skozi roke". Le tako bo lahko banka v primeru večkratne uporabe razkrila goljufa.

Hans van Antwerpen [vA90] je predlagal prijem, s katerim lahko prenosljivost dodamo skoraj vsakemu sistemu. Zamisel temelji na uporabi tako imenovanih **praznih kovancev**. Uporabnik pri banki dvigne večjo količino praznih kovancev in jih ima na zalogi. Prazni kovanci so skoraj identični kot običajni elektronski kovanci, le da sami po sebi ne nosijo vrednosti temveč le informacijo o uporabniku. Banka zato ob dvigu tudi ne trga denarja z uporabnikovega računa.

Prazen kovanec dobi svojo vrednost šele, če ga povežemo s kovancem z vrednostjo. Trgovec, ki ima prazen kovanec, to stori tako, da pri sprejemu kovanca (plačila) za izziv izbere zgostitev (izvleček) praznega kovanca. Nato si shrani osnovni kovanec in plačilni dodatek, ki bi ga potreboval za vnovčenje, ter zraven še prazni kovanec. Vsemu skupaj rečemo razširjen kovanec, saj lahko z njim povsem običajno plačujemo. Plačamo v bistvu s praznim kovancem, vendar moramo priložiti še osnovni kovanec in plačilni dodatek, da lahko prejemnik preveri veljavnost kovanca.

Ker vsak uporabnik v plačilni verigi doda svoj prazni kovanec, nato pa še plačilni dodatek, velikost razširjenega kovanca hitro naraste. Očitno je, da se temu ne da izogniti, saj mora razširjen kovanec nositi informacijo o vseh uporabnikih, ki jim je šel "skozi roke", Chaum in Pedersen [CP93a] pa sta to celo dokazala. Rast je linearna, kar pomeni, da je še sprejemljiva. V praksi se ocenjuje, da je lahko taka plačilna veriga dolga do 20 členov (stroge omejitve ni), potem pa je že skrajni čas, da tak razširjen kovanec nekdo nese v banko, kjer ga lahko zamenja za navadnega. V primeru goljufije banka (podobno kot pri navadnih kovancih) razkrije goljufa, čeprav se je kasneje s goljufivim kovancem še naprej poslovalo. Podrobnosti bomo opisali v podrazdelku 6.7.3.

Ker je osnovni kovanec v razširjenem kovancu ves čas ostaja isti, bi ga vsak uporabnik lahko spoznal, če bi kovanec kasneje še kdaj prejel. Torej je prenosljiv denar povezljiv. Identitete ostalih uporabnikov kovanca sicer ne bi mogel razbrati, morda pa bi mu že ta informacija kaj koristila. Če koga to moti, lahko prejete kovance vedno nese v banko in jih zamenja za nove navadne kovance. Povezljivost je torej izbirna lastnost.

Možno je, ni pa nujno, da bi že prazni kovanci imeli specificirano vrednost. Verjetno je najbolj smiselno, da se specificira maksimalna dovoljena vrednost, da se omejijo morebitne škode.

5.1.6 DELJIVOST

Deljivost fizičnih kovancev ali bankovec je lastnost, ki je danes ne poznamo in ne uporabljamo. Če dobro pomislimo, pa bi bilo vsekakor koristno imeti namesto polne denarnice drobiža le en večji bankovec. Nekateri sistemi za digitalni denar nam to omogočajo, vendar so zato bolj zapleteni.

Okamoto [OO92, EO94, Oka95] se je s tem največ ukvarjal in je razvil več sistemov. Njegovi sistemi temeljijo na kvadratnih ostankih v grupi RSA. So precej zapleteni in praktično le pogojno uporabni, omogočajo pa **poljubno deljivost**. To pomeni, da je mogoče en kovanec razdeliti na poljubno število kosov skoraj poljubne velikosti.

Brandsovi sistemi, ki se jim bomo posvetili v 6. poglavju, omogočajo preprostejšo različico deljivosti. Recimo ji **deljivost s podkovanci**. En kovanec vsebuje več podkovancev, ki imajo točno določene vrednosti. Pri prvem plačilu aktiviramo le željene podkovance. To večinoma pomeni, da le za njih priskrbimo plačilni dodatek. Pri naslednjem plačilu lahko aktiviramo željene preostale podkovance. Seveda bi lahko tudi dvakrat aktivirali isti podkovanec, vendar bi potem banka lahko razkrila našo identiteto in nas obtožila goljufije.

Deljivost s podkovanci se izplača le, kadar je opazno bolj učinkovita kot enako število navadnih kovancev. To se nanaša tako na velikost kovanca, kot tudi na velikost plačilnega dodatka ter na računsko zahtevnost. Ta pogoj je običajno izpolnjen, predvsem če je v enem kovancu večje število podkovancev, recimo vsaj pet. Žal pa so vsi podkovanci znotraj istega kovanca med seboj povezljivi. Pri vsakem plačilu moramo namreč dati celoten kovanec (ne pa tudi vseh plačilnih dodatkov). Če recimo večkrat plačujemo v isti veleblagovnici, bi lahko le ta odkrila povezave med našimi nakupi. Kogar to moti, lahko še vedno uporablja klasične (nedeljive) kovance, ki so le za konstanten faktor (približno trikrat) manj učinkoviti.

5.2 OBLIKE DIGITALNEGA DENARJA

Že do sedaj smo spoznali nekatere oblike digitalnega denarja, predvsem elektronske kovance. V tem razdelku si bomo te in tudi druge oblike podrobneje ogledali. Nekateri sistemi ne omogo-

čajo učinkovitih deljivih kovancev, zato ta problem rešujejo z večkratno uporabnimi kovanci in povračljivimi čeki. Pri sistemih, kjer je deljivost kovancev možno učinkovito implementirati (kot npr. Brandsov sistem v 6. poglavju), te oblike nimajo posebnega smisla.

a) KOVANCI

To je osnovna in še vedno najbolj uporabna oblika. Ker jo že dobro poznamo, je ne bom podrobneje opisoval.

b) VEČKRATNO UPORABNI KOVANCI [Bra93]

Ta razširitev delno nadomešča deljive kovance. Ideja je v tem, da je identiteta uporabnika v kovancu zakrita na način, da lahko uporabnik kovanec uporabi n-krat, če pa ga tudi $(n+1)$-krat, banka lahko razkrije njegovo identiteto in ga obtoži goljufije. Iz konstrukcijskih razlogov so kovanci te vrste običajno večji od navadnih, vendar se kljub temu izplačajo. Ker ob vsakem plačilu plačamo z istim kovancem, so plačila med seboj seveda povezljiva.

c) ČEKI

Poznamo tudi digitalni ekvivalent klasičnih čekov (neanonimi elektronski čeki z ali brez predplačila), vendar si bomo tu ogledali drugačne vrste čekov.

- **čeki s povračilom** [Bra93]: Tej vrsti čekov bi lahko rekli tudi kovanci z enkratno deljivostjo. Pri prvem plačilu plačamo s čekom željeni znesek, ki ne sme presegati vrednosti čeka. Običajno omogočajo le deljivost s podkovanci. Nadaljna plačila s tem čekom niso več mogoča, pač pa ga lahko nesemo v banko, kjer po posebnem povračilnem protokolu dobimo nazaj neporabljeni del. Deljivi kovanci v vsem prekašajo to obliko digitalnega denarja.
- **čeki s števcem** [Bra94a, Bra94b]: Ti čeki so namenjeni le za uporabo v denarnicah z nadzornikom. Pri dvigu čekov nam banka ne trga denarja (pri prejšnji vrsti ga), pač pa nam trga denar, kadar napolnimo števec denarja, ki ga vodi nadzornik. Ko želimo ček uporabiti, nadzornik pogleda ali stanje števca omogoča željeni izdatek, nato napiše na ček vrednost in odobri njegovo uporabo. Problem nastane, kadar neko vdre v nadzornika in lahko nastavlja stanje števca. Banka zato določi čekom maksimalno vrednost, toda kljub temu se goljuf lahko okoristi za razliko med maksimalno vrednostjo čekov v denarnici in stanjem števca. Če bi nato še večkrat dvignil čeke, bi banka kmalu ugotovila, da preveč dviguje čeke in premalo polni števec.

  Zaradi opisane slabosti je ta vrsta čekov primerna le za minimalne zneske, recimo tja do 100 SIT. Je pa od vseh implementacij deljivosti denarja najbolj učinkovita.

d) PRAZNI KOVANCI IN ČEKI

Prazne kovance smo že obravnavali, ko smo govorili o prenosljivosti denarja, zato bomo opis izpustili. Naj tu omenim le še eno njihovo dobro lastnost. Ker imajo prazni kovanci običajno določeno le maksimalno vrednost ali pa niti te ne, to pomeni, da bi jih lahko imenovali tudi prazni čeki. To omenjam zato, ker je tako splošna tudi njihova uporaba. Lahko služijo za ponovno uporabnost denarja, ki nam ga je nekdo plačal s kovanci ali pa s čeki. Isti prazni kovanec je torej splošno uporaben. Lahko bi služil celo, če je osnovi kovanec izdala druga banka ali je nominiran v tuji valuti. To je zelo pomembno za trgovanje na svetovnem tržišču, kjer nastopa večje število izdajateljev denarja.

Poglavje 6

BRANDSOV DIGITALNI DENAR TEMELJEČ NA CERTIFIKATIH S SKRIVNIM KLJUČEM

To poglavje bo kriptografsko najbolj zanimivo. V njem si bomo ogledali konkreten sistem digitalnega denarja. Najprej bomo naredili hiter pregled zgodovinskega razvoja in obstoječih sistemov ter za nadaljno obravnavo izbrali najboljšega. Nato se bomo spomnili znanja iz petega poglavja in se posebej posvetili denarnicam z nadzornikom. Na njih je namreč zasnovan naš izbrani sistem. Spet se bomo spomnili certifikatov s skrivnim ključem, nakar bomo podrobno, korak za korakom, opisali naš sistem digitalnega denarja, ki temelji na njih. Ogledali si bomo vse protokole in nato poskušali dokazati pravilnost, varnost in ostale željene lastnosti sistema. Posebej bomo obravnavali napad s sočasnim dvigom in pokazali, kako ga onemogočimo. Obravnavali bomo možne težave, ki bi lahko ogrozile delovanje sistema ali omogočile zlorabe, ter prikazali možne rešitve. Nato si bomo ogledali, kako je možno obstoječi sistem še razširiti. Prikazali bomo uporabo čekov s števcem ter kako dodati prenosljivost in deljivost. Na koncu, ko bomo sistem že dodobra poznali, bo pravi čas, da ga ovrednotimo še s praktičnega stališča. Poračunali bomo razne prostorske in časovne zahtevnosti ter ocenili na kakšni strojni opremi in s kakšnimi obremenitvami, bi lahko tekel. Čisto za konec se bomo poigrali z idejo, da bi tak sistem res uvedli v Sloveniji kot splošno plačilno sredstvo. Videli bomo, da koristi varnega elektronskega poslovanja daleč presežejo potencialne težave, zato bi lahko sistem že danes uporabljali tudi v praksi.

6.1 PREGLED PREJŠNJIH SISTEMOV

CWI (Nizozemski nacionalni institut za matematiko in računalniško znanost) je zadnjih 15 let center dogajanja na tem področju. Tam je nastala večina pomembnih sistemov.

Pionir na tem področju je bil David Chaum [Cha83, DN88, CP93a]. Kasneje se mu je pridružil še Hans van Antwerpen, ki je uvedel prenosljivost denarja [vA90] in skupaj s Chaumom sta odkrila podpise brez možnosti zanikanja [CvA90].

Njuno delo je izvrstno nadaljeval Stefan A. Brands. Njegovi sistemi so dosegli stopnjo učinkovitosti in praktičnosti, ki je že omogočala implementacijo v praksi. Prvi njegovi sistemi [Bra93, 1993] temeljijo na problemu reprezentacije (glej definicijo 3.3). Bili so še rahlo kompliciran, vendar dovolj prilagodljivi, da so izpolnjevali vse zahteve in omogočali vse razširitve. Pri njih je

Brands prvič uporabil nekatere prijeme, ki so se kasneje izkazali za koristne. Nato se je Brands posvetil certifikatom s skrivnim ključem [Bra95c, Bra95d, Bra95b], ki jih je sam iznašel in že leta 1995 predstavil enostavnejši in še učinkovitejši sistem [Bra95a], ki bo tema tega poglavja. Že prej je uvidel, da sistem za praktično uporabo potrebuje denarnice z nadzornikom [Bra94c], kar je potrebno upoštevati že pri snovanju sistema.
Brands je napisal tudi nekaj bolj splošnih del [Bra98, Bra94a, Bra94b], nedavno pa je izdal tudi knjigo [Bra00], kjer predstavi svojo doktorsko disertacijo in združi svoje poglede na kriptografske varnostne mehanizme.

Od ostalih kriptografov, aktivnih na tem področju, bi omenil še Okamota [OO90, OO92, EO94, Oka95] in Fergusona [Fer93, Fer94], ki sta tudi, vsak po svoje, razvila zanimive in uporabne sisteme.
Kljub temu je izmed omenjenih Brandsov sistem [Bra95a] matematično najlepši, najučinkovitejši in zdi se, tudi najboljši. Ker je v svojem bistvu preprost, je primeren za opis, omogoča mnoge dokaze varnosti in tudi sicer zaenkrat vliva največ zaupanja. Zato bomo izmed omenjenih sistemov izbrali prav tega, da ga v tem poglavju podrobno preučimo.

6.2 DENARNICA Z NADZORNIKOM

Denarnice z nadzornikom smo spoznali v prejšnjem poglavju. Uvedel jih je že Chaum [CP93b, CP], zares pa so zaživele v Brandsovih sistemih. Izkazalo se je namreč, da brez njih sistem za banko ne more biti sprejemljiv, pa tudi sicer se na ta način poveča praktičnost sistema.

Nadzornik je majhna pametna kartica, ki jo izda banka in v izjemnih primerih jo lahko dobi celo nazaj. Če želimo, da bo sistem, ki uporablja nadzornika, nudil anonimnost uporabniku, morata biti izpolnjeni naslednji zahtevi:

- **preprečitev sledljivosti**: Nadzornik nikoli direktno na komunicira z banko, ampak je vedno vmes še denarnica. Ta mora sporočila toliko spremeniti, da banka ne more povezati svojega pogleda s pogledom nadzornika (če ga kdaj dobi nazaj).

- **preprečitev prikritega kanala**: Onemogočeno mora biti, da bi si banka in nadzornik izmenjala kakršno koli informacijo. Primer takega prikritega kanala bi bila kaka lastnost naključnega števila, ki bi se ohranjala, tudi ko denarnica podatke spremeni. Obstajajo vhodni in izhodni prikriti kanali in treba je onemogočiti oboje.

6.3 SISTEM DIGITALNEGA DENARJA TEMELJEČ NA CERTIFIKATIH S SKRIVNIM KLJUČEM

V grobem sistem ustreza splošnemu opisu anonimnih sistemov digitalnega denarja iz prejšnjega poglavja. Kot v podrazdelku 5.1.2 ga bomo razdelili na naslednje dele: priprava, odprtje računa, dvig denarja, plačilo in polog denarja. Omejene slepe podpise certifikatov s skrivnim ključem že poznamo, prav tako razna področja njihove uporabe. To nam bo zelo koristilo pri razumevanju sistema.
Sistem temelji na uporabi denarnice z nadzornikom, vendar deluje tudi, če nekdo vdre v nadzornika in ga simulira. Zato je možno sistem poenostaviti, če se odpovemo nadzorniku. Čeprav bi bil tak sistem primeren za uvod, bom ta del izpustil, saj smo si dovolj predznanja nabrali že v četrtem poglavju. Če želi, naj ga bralec, po končanem branju diplome, konstruira za vajo.

Osnovni sistem je namenjen izdajanju in uporabi preprostih kovancev (le za eno vrednost). Naknadno bomo pokazali, kako ga razširiti za kovance različnih vrednosti. Sistem bomo najprej opisali, nato pa sledi dokaz pravilnosti in reševanje možnih težav.

6.3.1 PRIPRAVA

Preden banka $\mathcal{B}$ začne izdajati denar in odpirati račune, mora izbrati splošne parametre sistema in svoje ključe. To obsega naslednje korake:

- izbere model grupe G_q.
- izbere zasebna ključa $x_{00}, x_{10} \in_R \mathbb{Z}_q$.
- izbere generator $g_0 \in_R G_q \setminus \{1\}$.
- izračuna $h_0 = g_0^{x_{00}}$ in $g_1 = {g_0}^{x_{10}}$.
- izbere enosmerno zgoščevalno funkcijo $\mathcal{H}(\cdot)$, ki je krepko brez trčenj in brez korelacij. Zgoščevalna funkcija $\mathcal{H}(\cdot)$ slika poljubno dolgo zaporedje bitov v $\mathbb{Z}_{2^n}$, kjer je n neko naravno število. V praksi MD5 in SHA-1 izpolnjujeta zahtevane pogoje, čeprav to še ni dokazano.

Opomba 6.1. *$\in_R$ pomeni izbiro naključnega elementa. Verjetnostna porazdelitev izbire je enakomerno porazdeljena in neodvisna od vseh ostalih spremenljivk in parametrov.*

Parametri $(G_q, g_0, g_1, \mathcal{H}(\cdot))$ so javni parametri, (x_{00}, x_{10}) pa sta zasebna ključa banke, ki jih je treba najstrožje čuvati. Javni parametri so javno objavljeni in jih mora imeti vsak uporabnik sistema. Običajno jih dobi ob odprtju računa. Parametra G_q in g_0 sta lahko tudi standardizirana, kar omogoča razvoj prilagojene strojne opreme.

6.3.2 ODPRTJE RAČUNA

Odprtje računa se razlikuje glede na to ali ga odpre uporabnik $\mathcal{U}_i$ ali trgovec $\mathcal{T}_j$.

odprtje računa trgovca $\mathcal{T}_j$:

Trgovec $\mathcal{T}_j$ lahko odpre račun pod psevdonimom. S tem istim psevdonimom se bo tudi predstavljal strankam. Banka spravi v bazo psevdonim, stanje števca (0) in javni ključ trgovca, ki ga bo ta uporabljal za avtentikacijo. Tak račun je možno odpreti tudi na daljavo, ko se pojavi potreba.

Če bo $\mathcal{T}_j$ želel z računa tudi dvigovati digitalni denar, bo moral izvesti še postopek za odprtje računa uporabnika. Kljub temu bo kot trgovec še vedno lahko uporabljal psevdonim. Pravo identiteto bo moral dokazati le pri dvigu denarja.

odprtje računa uporabnika $\mathcal{U}_i$:

Uporabnik $\mathcal{U}_i$ se mora banki $\mathcal{B}$ najprej predstaviti in potrditi svojo identiteto. Da ji tudi kopijo svojega certifikata, tako da bo lahko preverjala njegove podpise in njegovo identiteto pri komunikaciji na daljavo. Certifikat je lahko namenjen za poljuben asimetričen sistem (običajno RSA). Banka da uporabniku majhno pametno kartico - nadzornika $\mathcal{N}_i$, ki jo ta vstavi v svojo denarnico. Še pred tem $\mathcal{B}$ izbere zasebni (identifikacijski) ključ nadzornika $x_{1i} \in_R \mathbb{Z}_q$, ga vpiše v nadzornika, pa tudi v svojo bazo poleg identitete $\mathcal{U}_i$. Za vsak slučaj preveri, da nima enakega ključa morda že kak drugi nadzornik. (Zato verjetnostna porazdelitev ni čisto enakomerna, vendar dovolj podobna.) Banka izračuna še $h_i = g_1^{x_{1i}}$ in

ga da $\mathcal{U}_i$, ki ga bo kasneje potreboval. (Banka lahko vpiše v nadzornika tudi javni ključ $\mathcal{U}_i$, tako da bo nadzornik za uporabo zahteval avtentikacijo. To sicer ni nujno potrebno za varnost sistema, vendar verjetno poveča stopnjo zaupanja pri ljudeh.)

Pri obeh načinih odprtja računa, banka priskrbi tudi javne parametre sistema in svoj certifikat, ki bo kasneje omogočal vzpostavitev varne povezave.

6.3.3 DVIG DENARJA

Za dvig denarja $\mathcal{B}$ in $\mathcal{U}_i$ najprej vzpostavita varno povezavo, nakar $\mathcal{U}_i$ pošlje podpisano zahtevo po dvigu določenega števila kovancev. S podpisano zahtevo banka preveri identiteto $\mathcal{U}_i$. Nato za vsak kovanec izvedeta sledeče korake:

1. Nadzornik $\mathcal{N}_i$ izbere $\omega_i \in_R \mathbb{Z}_q$, izračuna $a_i = g_1^{\omega_i}$ in ga pošlje $\mathcal{U}_i$. Število ω_i (in tudi a_i) shrani za kasnejšo uporabo pri plačilu.
2. Banka $\mathcal{B}$ izbere $\omega_0 \in_R \mathbb{Z}_q$, izračuna $a_0 = g_0^{\omega_0}$ in ga pošlje $\mathcal{U}_i$.
3. Uporabnik $\mathcal{U}_i$ izbere šest naključnih števil $x_{0i}, \omega_{0i}, \omega_{1i}, s_1, s_2, s_3 \in_R \mathbb{Z}_q$. Izračuna $h_i' = g_0^{x_{0i}} h_i$, $a_i' = g_0^{\omega_0} g_1^{\omega_i} h_i{}^{s_3} a_i$ in zgostitev $c_0' = \mathcal{H}(h_0 h_i', a_i', g_0^{s_1}(h_0 h_i)^{s_2} a_0)$ ter pošlje $c_0 = c_0' + s_2 \mod q$ banki $\mathcal{B}$.
4. Banka $\mathcal{B}$ odgovori z $r_0 = c_0(x_{00} + x_{10}x_{1i}) + \omega_0 \mod q$ in odšteje vrednost kovanca z računa $\mathcal{U}_i$.
5. Uporabnik $\mathcal{U}_i$ preveri $g_0^{r_0}(h_0 h_i)^{-c_0} = a_0$. Če enakost velja, izračuna $r_0' = r_0 + c_0' x_{0i} + s_1 \mod q$. Za plačilo spravi kovanec $(h_i', a_i'), (c_0', r_0')$ in pa tudi $(x_{0i}, \omega_{0i}, \omega_{1i}, s_3), a_i$.

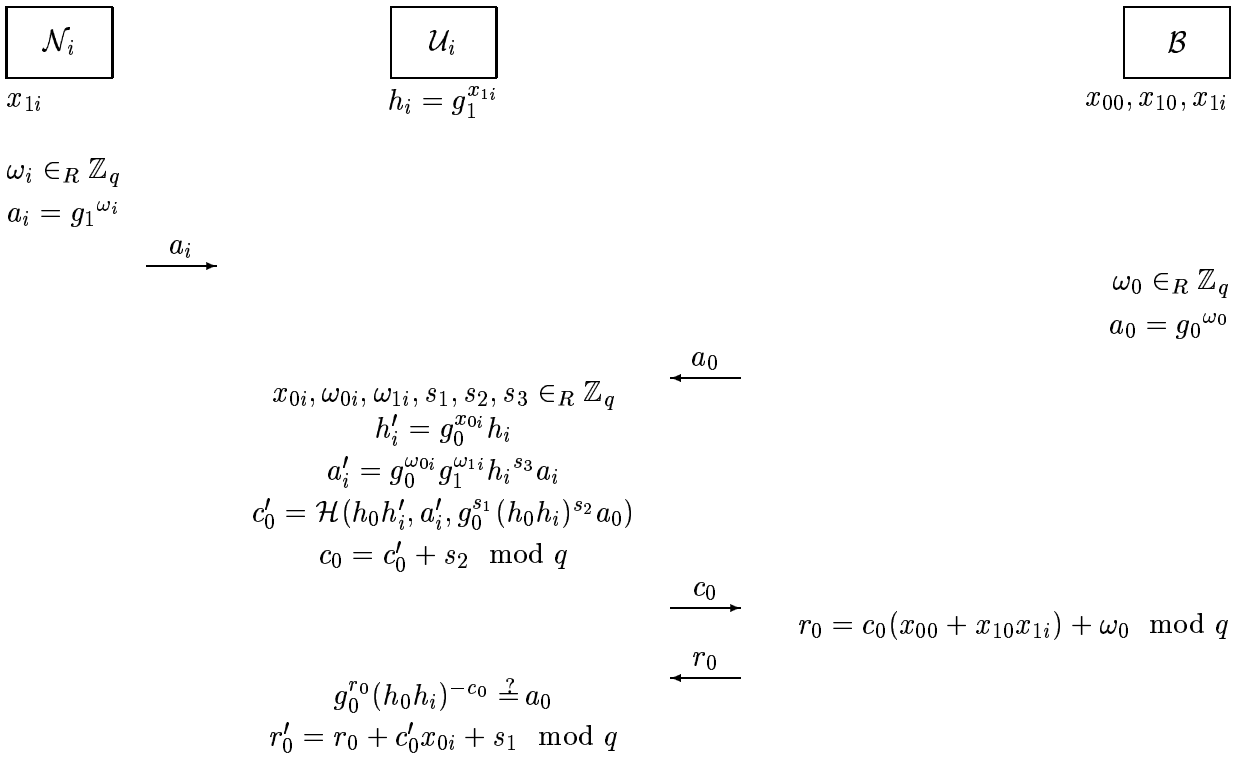

Diagram 6.1: Protokol za dvig denarja

Dvig večih kovancev je mogoče izvesti zaporedno ali vzporedno. Bolj varen je zaporedni dvig, kjer je v primeru napak sporen le en kovanec.

6.3.4 PLAČILO

Če je uporabnik $\mathcal{U}_i$ uspešno izvedel protokol za dvig kovanca, lahko s tem kovancem plača trgovcu $\mathcal{T}_j$ po sledečem protokolu:

1. Uporabnik $\mathcal{U}_i$ izračuna $d = \mathcal{H}(h_i', \texttt{spec}, a_i')$. Nato pošlje $d' = d + s_3 \mod q$ in a_i nadzorniku $\mathcal{N}_i$. Parameter spec je zaporedje večih podatkov v točno določenem formatu, ki ga je določila banka $\mathcal{B}$. En podatek običajno enolično določa $\mathcal{T}_j$, drugi pa zaporedno številko njegovega plačila. V tem primeru banka ne bi sprejela dveh plačil z enakimi zaporednimi številkami. Če $\mathcal{U}_i$ ne more spec-a sam ugotoviti, mu ga mora priskrbeti $\mathcal{T}_j$. (Namesto spec-a lahko priskrbi že izračunan d).

2. Nadzornik $\mathcal{N}_i$ pogleda ali ima v spominu a_i. Če ga ima, vzame pripadajoči ω_i, izračuna $r_i = d' x_{1i} + \omega_i \mod q$ in ga pošlje $\mathcal{U}_i$. Nato izbriše ω_i in a_i iz spomina. Če a_i ni v spominu, to pomeni, da izbrani kovanec ni na razpolago. V tem primeru $\mathcal{N}_i$ zavrne sodelovanje pri plačilu.

3. Uporabnik $\mathcal{U}_i$ preveri $g_1^{r_i} h_i^{-d'} = a_i$. Če enakost velja, potem izračuna $r_{0i} = d x_{0i} + \omega_{0i} \mod q$ in $r_{1i} = r_i + \omega_{1i} \mod q$ in pošlje trojico $(h_i', a_i'), (c_0', r_0'), (r_{0i}, r_{1i})$ trgovcu $\mathcal{T}_j$.

4. Trgovec $\mathcal{T}_j$ izračuna $d = \mathcal{H}(h_i', \texttt{spec}, a_i')$ in preveri

$$g_0^{r_{0i}} g_1^{r_{1i}} (h_i')^{-d} = a_i' \quad \text{in} \quad c_0' = \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'} (h_0 h_i')^{-c_0'}).$$

   Če obe enakosti veljata, sprejme plačilo.

6.3.5 POLOG DENARJA

Če trgovec $\mathcal{T}_j$ sprejme plačilo, se lahko kadarkoli kasneje poveže z banko in po sledečem protokolu položi denar na svoj račun:

1. Trgovec $\mathcal{T}_j$ pošlje $\mathcal{B}$ kovanec $(h_i', a_i'), (c_0', r_0')$, plačilni dodatek (r_{0i}, r_{1i}) in spec. Vsemu skupaj pravimo **plačilni zapis**.

2. Banka $\mathcal{B}$ izračuna $d = \mathcal{H}(h_i', \texttt{spec}, a_i')$ in sprejme plačilni zapis, če velja

$$g_0^{r_{0i}} g_1^{r_{1i}} (h_i')^{-d} = a_i' \quad \text{in} \quad c_0' = \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'} (h_0 h_i')^{-c_0'}). \tag{6.1}$$

   Dodatno preveri, da kovanec $(h_i', a_i'), (c_0', r_0')$ še ni bil izplačan ter da je spec pravilno zgeneriran. Če je vse v redu, shrani podatke $(h_i', a_i'), (c_0', r_0'), (\texttt{spec}, r_{0i}, r_{1i})$ v bazo plačil. Denar nakaže na račun trgovca navedenega v spec-u. Če želi $\mathcal{T}_j$ denar nakazati kam drugam ali ga dvigniti, se mora še avtenticirati.

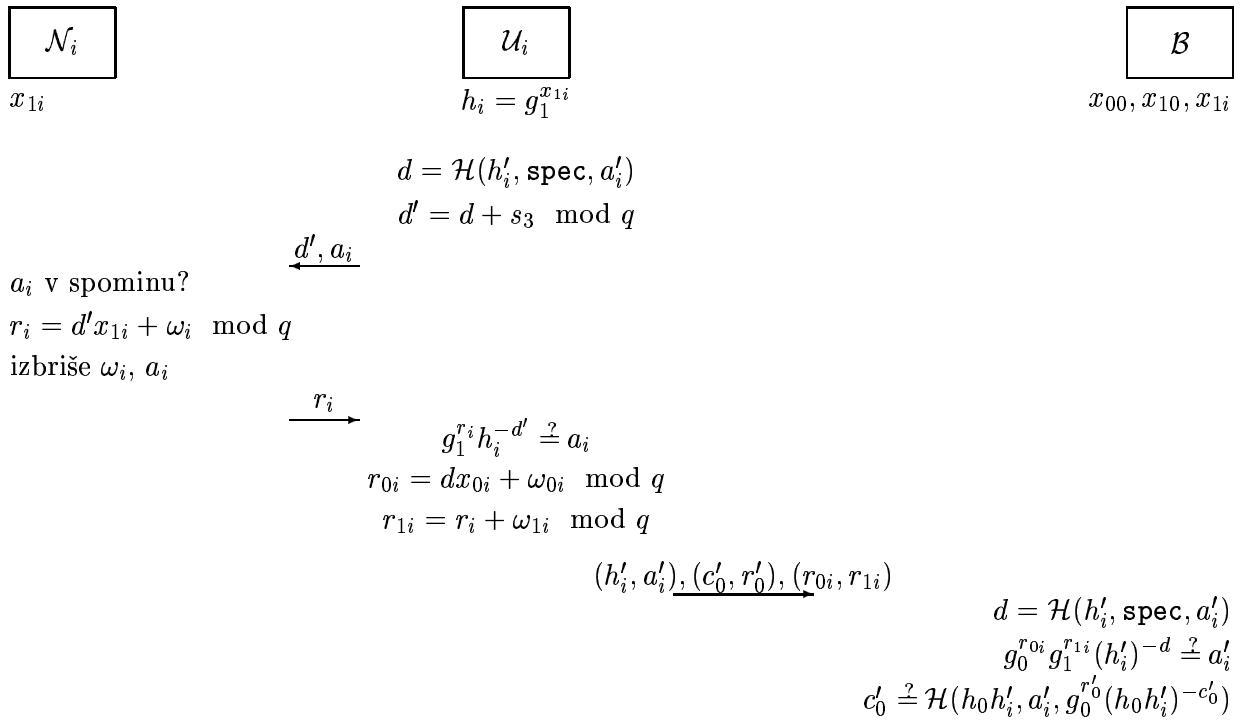

Diagram 6.2: Plačilni protokol

6.3.6 ISKANJE GOLJUFA

Če $\mathcal{B}$ ugotovi, da sta plačilni zapis in spec veljavna, vendar je isti kovanec že enkrat bil vnovčen, potem poišče v bazi plačil par $(\mathtt{spec}^*, r_{1i}^*)$, ki je bil uporabljen pri prvem plačilu s tem kovancem. Izračuna $d = \mathcal{H}(h_i', \mathtt{spec}, a_i')$ in $d^* = \mathcal{H}(h_i', \mathtt{spec}^*, a_i')$. Nadalje ločimo tri možnosti:

- $d \neq d^*$: Potem je goljufal $\mathcal{U}_i$, saj je dvakrat plačal z istim kovancem. $\mathcal{B}$ izračuna $x_{1i} = (r_{1i} - r_{1i}^*)/(d - d^*) \mod q$ in poišče uporabnika $\mathcal{U}_i$, ki ima nadzornika $\mathcal{N}_i$ z zasebnim ključem x_{1i}. Seveda je ta vrsta goljufije možna le, če $\mathcal{U}_i$ vdre v nadzornika.

- $d = d^*$, $\mathtt{spec} = \mathtt{spec}^*$: Tokrat je očitno goljufal $\mathcal{T}_j$. Banka mu zato ne izplača denarja. Običajno $\mathcal{B}$ sploh zahteva, da so vsi spec-i med seboj različni (mora paziti $\mathcal{T}_j$).

- $d = d^*$, $\mathtt{spec} \neq \mathtt{spec}^*$: Verjetnost za ta primer je zanemarljivo majhna. Goljufal je $\mathcal{U}_i$, vendar se njegove identitete ne da razkriti. Škodo pokrije banka, toda to ni problem, saj je verjetnost tega dogodka res zanemarljiva.

6.3.7 POVEZAVA S CERTIFIKATI S SKRIVNIM KLJUČEM

Certifikati, ki jih uporabljamo pri našem sistemu, so zelo podobni tistim iz podrazdelka 4.2.1. Certificiran javni ključ je par $((h_i', a_i'), (c_0', r_0'))$. Pravilno generiran je natanko tedaj, kadar velja $c_0' = \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'} (h_0 h_i')^{-c_0'})$. Certificiran par ključev je trojica $(((x_{0i}, x_{1i}), (\omega_{0i}, \omega_{1i}')), (h_i', a_i'), (c_0', r_0'))$, tako da je $((h_i', a_i'), (c_0', r_0'))$ certificiran javni ključ ter da dodatno velja $h_i' = g_0^{x_{0i}} g_1^{x_{1i}}$ in $a_i' = g_0^{\omega_{0i}} g_1^{\omega_{1i}'}$, kjer je $\omega_{1i}' = \omega_{1i} + x_{1i} s_3 + \omega_i$.

Simulacija certifikata poteka kot v podrazdelku 4.2.1. Najprej izberemo $t_1, t_2 \in_R \mathbb{Z}_q$ in $a_i' \in_R G_q$.

Nato izračunamo

$$\begin{aligned} h'_i &= h_0^{-1} g_0^{t_1} \mod q, \\ c'_0 &= \mathcal{H}(g_0^{t_1}, a'_i, g_0^{t_2}), \\ r'_0 &= c'_0 t_1 + t_2 \mod q. \end{aligned}$$

Tak certifikat je pravilno generiran, saj velja

$$c'_0 = \mathcal{H}(h_0 h'_i, a'_i, g_0^{r'_0}(h_0 h'_i)^{-c'_0}). \tag{6.2}$$

Dokaz (enakosti 6.2)**.**

$$c'_0 = \mathcal{H}(g_0^{t_1}, a'_i, g_0^{t_2}) = \mathcal{H}(h_0 h'_i, a'_i, g_0^{c'_0 t_1 + t_2}(g_0^{t_1})^{-c'_0}) = \mathcal{H}(h_0 h'_i, a'_i, g_0^{r'_0}(h_0 h'_i)^{-c'_0})$$ ■

6.4 DOKAZ PRAVILNOSTI

Označimo z $\overline{\mathcal{Z}}$ osebo, ki je poštena in se drži protokola, z $\widehat{\mathcal{Z}}$ osebo, ki lahko goljufa in ima polinomsko računsko moč, ter z $\widetilde{\mathcal{Z}}$ osebo, ki lahko goljufa in ima neomejeno računsko moč. Z $\mathcal{Z}$ označimo poljubno izmed teh treh oseb.

6.4.1 DOKAZ POLNOSTI

Trditev 6.2. *Če se $\overline{\mathcal{U}_i}$ in $\overline{\mathcal{B}}$ držita protokola za dvig denarja, ima $\overline{\mathcal{U}_i}$ na koncu veljaven kovanec $(h'_i, a'_i), (c'_0, r'_0)$, za katerega velja $c'_0 = \mathcal{H}(h_0 h'_i, a'_i, g_0^{r'_0}(h_0 h'_i)^{-c'_0})$.*

Dokaz. Najprej pokažimo enakost iz 5. koraka protokola za dvig:

$$g_0^{r_0}(h_0 h_i)^{-c_0} = g_0^{c_0(x_{00}+x_{10}x_{1i})+\omega_0}(h_0 h_i)^{-c_0} = h_0^{-c_0}(g_1^{x_{1i}})^{-c_0} a_0 (h_0 h_i)^{-c_0} = a_0,$$

nato pa še veljavnost kovanca:

$$\begin{aligned} c'_0 &= \mathcal{H}(h_0 h'_i, a'_i, g_0^{s_1}(h_0 h_i)^{s_2} a_0) \\ &= \mathcal{H}(h_0 h'_i, a'_i, g_0^{s_1}(h_0 h_i)^{s_2} g_0^{r_0}(h_0 h_i)^{-c_0}) \\ &= \mathcal{H}(h_0 h'_i, a'_i, g_0^{r'_0 - c'_0 x_{0i}}(h_0 h_i)^{-c'_0}) \\ &= \mathcal{H}(h_0 h'_i, a'_i, g_0^{r'_0}(h_0 g_0^{x_{0i}} h_i)^{-c'_0}) \\ &= \mathcal{H}(h_0 h'_i, a'_i, g_0^{r'_0}(h_0 h'_i)^{-c'_0}). \end{aligned}$$

Torej ima $\overline{\mathcal{U}_i}$ na koncu veljaven kovanec. ■

Trditev 6.3. *Če je $\overline{\mathcal{U}_i}$ uspešno opravil protokol za dvig denarja ter se $\overline{\mathcal{U}_i}$ in $\overline{\mathcal{N}_i}$ držita plačilnega protokola, potem bo $\mathcal{T}_j$ sprejel plačilo.*

Dokaz. Najprej pokažimo enakost iz 3. koraka plačilnega protokola, ki je povsem običajna Schnorrova identifikacija:

$$g_1^{r_i} h_i^{-d'} = g_1^{d' x_{1i} + \omega_i} g_1^{x_{1i}(-d')} = g_1^{\omega_i} = a_i,$$

nato pa še veljavnost plačilnega zapisa:

$$\begin{aligned} g_0^{r_{0i}} g_1^{r_{1i}} (h_i')^{-d} &= g_0^{r_{0i}} g_1^{r_{1i}} (g_0^{x_{0i}} h_i)^{-d} \\ &= g_0^{r_{0i} - d x_{0i}} g_1^{r_{1i}} h_i^{-d} \\ &= g_0^{\omega_{0i}} g_1^{r_{1i}} h_i^{-(d' - s_3)} \\ &= g_0^{\omega_{0i}} g_1^{r_i + \omega_{1i}} h_i^{s_3 - d'} \\ &= g_0^{\omega_{0i}} g_1^{\omega_{1i}} h_i^{s_3} g_1^{r_i} h_i^{-d'} \\ &= g_0^{\omega_{0i}} g_1^{\omega_{1i}} h_i^{s_3} a_i \\ &= a_i'. \end{aligned}$$

Drugo enakost (iz 6.1) smo dokazali že pri prejšnji trditvi. Iz omenjenih enakosti sledi, da bo $\mathcal{T}_j$ sprejel plačilni zapis (in s tem plačilo), saj lahko kdorkoli preveri njegovo pravilnost. ■

Trditev 6.4. *Če je $\overline{\mathcal{T}_j}$ uspešno opravil plačilni protokol ter se drži protokola za polog denarja, potem bo $\mathcal{B}$ sprejela polog.*

Dokaz. Enakosti, ki jih $\mathcal{B}$ preveri, smo že dokazali (glej zadnja dva dokaza). Unikatnost spec-a sledi iz tega, da je en del enolično vezan na $\mathcal{T}_j$, ta pa mora skrbeti, da se zaporedne številke plačil ne ponavljajo. S tem smo dokazali, da bo $\mathcal{B}$ sprejela polog. ■

Pri dvigu in plačilu se uporabniku ni treba zanašati na poštenost ostalih strank, ampak je dovolj, da preveri zahtevane enakosti (v diagramih so označene z $\stackrel{?}{=}$). Enako velja za trgovca pri pologu.

S tem smo zaključili dokaz polnosti sistema.

6.4.2 DOKAZ VAROVANJA ZASEBNOSTI

Trditev 6.5. *Za vsakega poštenega uporabnika $\overline{\mathcal{U}_i}$ ter za (vse hkrati):*
- vsak pogled $\widetilde{\mathcal{B}}$ v protokolu za dvig, ki bi ga $\overline{\mathcal{U}_i}$ sprejel
- vsak pogled $\widetilde{\mathcal{T}_j}$ v plačilnem protokolu, če se $\overline{\mathcal{U}_i}$ drži protokola
- vsak pogled $\widetilde{\mathcal{N}_i}$ v protokolu za dvig in plačilnem protokolu, ki bi ga $\overline{\mathcal{U}_i}$ sprejel
obstaja natanko ena izbira naključnih vrednosti $(x_{0i}, \omega_{0i}, \omega_{1i}, s_1, s_2, s_3)$, tako da se pogledi $\widetilde{\mathcal{B}}$, $\widetilde{\mathcal{T}_j}$ in $\widetilde{\mathcal{N}_i}$ nanašajo na isti kovanec.

Dokaz. Definirajmo najprej množice pogledov.

$$\begin{aligned} Pogled(\widetilde{\mathcal{B}}) &= \{(a_0, c_0, r_0);\ a_0 \in G_q,\ c_0, r_0 \in \mathbb{Z}_q,\ g_0^{r_0}(h_0 h_i)^{-c_0} = a_0\} \\ Pogled(\widetilde{\mathcal{T}_j}) &= \{(h_i', a_i', c_0', r_0', r_{0i}, r_{1i}, d);\ h_i', a_i' \in G_q,\ r_0', r_{0i}, r_{1i} \in \mathbb{Z}_q,\ c_0', d \in \mathbb{Z}_{2^n}, \\ &\quad g_0^{r_{0i}} g_1^{r_{1i}} (h_i')^{-d} = a_i',\ c_0' = \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'}(h_0 h_i')^{-c_0'})\} \\ Pogled(\widetilde{\mathcal{N}_i}) &= \{(a_i, d', r_i);\ a_i \in G_q,\ d', r_i \in \mathbb{Z}_q,\ g_1^{r_i} h_i^{-d'} = a_i\} \end{aligned}$$

S temi pogledi so izbire $\overline{\mathcal{U}_i}$ enolično določene.

$$\begin{aligned} x_{0i} &= \log_{g_0}(h_i'/h_i) \\ \omega_{0i} &= r_{0i} - d x_{0i} \mod q \\ \omega_{1i} &= r_{1i} - r_i \mod q \\ s_1 &= r_0' - r_0 - c_0' x_{0i} \mod q \\ s_2 &= c_0 - c_0' \mod q \\ s_3 &= d' - d \mod q \end{aligned}$$

Da se pokazati, da te vrednosti izbir ustrezajo vsem zahtevanim enakostim v protokolih. ■

Trditev 6.6. *Če se $\mathcal{U}_i$ drži protokolov in ne goljufa, potem tudi $\widetilde{\mathcal{B}}$, $\widetilde{\mathcal{N}_i}$ in vsi trgovci $\widetilde{\mathcal{T}_j}$ skupaj ne morejo povezati dviga s plačilom.*

Dokaz. To je direktna posledica prejšnje trditve in dejstva, da je verjetnostna porazdelitev $(x_{0i}, \omega_{0i}, \omega_{1i}, s_1, s_2, s_3)$ enakomerna in neodvisna. Za vsak dvig je torej enako verjetno, da je povezan s katerimkoli plačilom. ■

6.4.3 VARNOST SISTEMA

Da lahko rečemo, da je sistem **varen za banko**, mora izpolnjevati naslednje zahteve:

1. Nobena skupina napadalcev (**zarota**) ne more ponarediti plačilnega zapisa.
2. Če banka ugotovi, da je bil kovanec večkrat uporabljen, potem lahko ugotovi identifikacijski ključ goljufa in s tem njegovo identiteto.
3. Nobena zarota ne more večkrat uporabiti istega kovanca, brez da bi prej vdrla v vsaj enega nadzornika.

Trenutno sistem ne izpolnjuje druge zahteve, saj lahko zarota pri sočasnem dvigu pridobi kovanec za poljuben identifikacijski ključ (torej drugačen od vseh članov zarote). To bomo rešili v naslednjem razdelku.

Da lahko rečemo, da je sistem **varen za uporabnika**, mora izpolnjevati naslednje zahteve:

1. Pravilno pridobljen kovanec bo vedno veljavno plačilno sredstvo.
2. Nihče mu ne more podtakniti goljufije.
3. Ob pravilni uporabi je anonimnost zagotovljena.

Zaenkrat druga zahteva še ni izpolnjena, saj banka lahko v imenu kogarkoli dviga kovance in jih nato večkrat uporabi. Temu se bomo posvetili v podrazdelku 6.6.1.

Da lahko rečemo, da je sistem **varen za trgovca**, mora izpolnjevati naslednje zahteve:

1. Pravilno sprejet kovanec bo banka vedno izplačala.
2. Nihče mu ne more podtakniti goljufije.

Dokazi, da naš sistem izpolnjuje navedene zahteve, so dolgi in zapleteni. Večinoma privzamejo varnost problema DLP (neobstoj polinomske rešitve) in Schnorrovih shem. Bralec si jih večino lahko ogleda v [Bra95a, razdelek 5.3].

6.5 NAPAD S SOČASNIM DVIGOM IN KAKO GA PREPREČITI

Poglejmo si, kako bi lahko zarota dveh napadalcev ($\mathcal{U}_i$ in $\mathcal{U}_j$) dobila kovanec za nek tretji identifikacijski ključ x'_{1i}, če lahko povsem sočasno izvaja protokol za dvig denarja. Torej bi lahko tak kovanec brez strahu poljubnokrat uporabila, saj njune identitete ne bi mogli razkriti.

Naj $\mathcal{U}_i$ in $\mathcal{U}_j$ oba vdreta v svoja nadzornika in izvesta njuna zasebna (identifikacijska) ključa. Sedaj lahko rečemo, da imata onadva zasebna ključa x_{1i}, x_{1j} in pripadajoča javna ključa h_i, h_j. Takole bi izgledal hkratni protokol za dvig:

1. Običajen korak.
2. i) Banka $\mathcal{B}$ izbere $\omega_0 \in_R \mathbb{Z}_q$, izračuna $a_0 = g_0^{\omega_0}$ in ga pošlje $\mathcal{U}_i$.
   j) Banka $\mathcal{B}$ izbere $\omega_0'' \in_R \mathbb{Z}_q$, izračuna $a_0'' = g_0^{\omega_0''}$ in ga pošlje $\mathcal{U}_j$.

2′. (Priprava) Napadalca $\mathcal{U}_i$ in $\mathcal{U}_j$ skupaj izračunata $h_i' = g_1^{x_{1i}'}$ za poljuben identifikacijski ključ x_{1i}'. Nato izbereta $\omega_{0i}', \omega_{1i}' \in_R \mathbb{Z}_q$ ter izračunata $a_i' = g_0^{\omega_{0i}'} g_1^{\omega_{1i}'}$ in $c_0' = \mathcal{H}(h_0 h_i', a_i', a_0 a_0'')$.

3. i) Napadalec $\mathcal{U}_i$ pošlje banki $c_0 = (c_0' x_{1j} - c_0' x_{1i}')/(x_{1j} - x_{1i})$.
   j) Napadalec $\mathcal{U}_j$ pošlje banki $c_0'' = (c_0' x_{1i}' - c_0' x_{1i})/(x_{1j} - x_{1i})$.
4. i) Banka $\mathcal{B}$ pošlje $r_0 = c_0(x_{00} + x_{10}x_{1i})$ uporabniku $\mathcal{U}_i$.
   j) Banka $\mathcal{B}$ pošlje $r_0'' = c_0''(x_{00} + x_{10}x_{1j})$ uporabniku $\mathcal{U}_j$.
5. Napadalca $\mathcal{U}_i$ in $\mathcal{U}_j$ najprej preverita

$$g_0^{r_0}(h_0 h_i)^{-c_0} = a_0 \quad \text{in} \quad g_0^{r_0''}(h_0 h_j)^{-c_0''} = a_0'',$$

nato pa izračunata $r_0' = r_0 + r_0''$.

Trditev 6.7. *Po gornjem postopku $\mathcal{U}_i$ in $\mathcal{U}_j$ skupaj dobita trojico*

$$((0, x_{1i}'), (\omega_{0i}', \omega_{1i}')), (h_i', a_i'), (c_0', r_0'),$$

ki je certificiran par ključev.

Dokaz. Očitno je $h_i' = g_0^0 g_1^{x_{1i}'}$ in $a_i' = g_0^{\omega_0'} g_1^{\omega_i'}$. Dokažimo še $c_0' = \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'}(h_0 h_i')^{-c_0'})$.

$$\begin{aligned}
c_0' &= \mathcal{H}(h_0 h_i', a_i', a_0 a_0'') \\
&= \mathcal{H}(h_0 h_i', a_i', g_0^{r_0}(h_0 h_i)^{-c_0} g_0^{r_0''}(h_0 h_j)^{-c_0''}) \\
&= \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'} h_0^{-(c_0 + c_0'')} h_i^{-c_0} h_j^{-c_0''}) \\
&= \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'} h_0^{-c_0'} g_1^{-(x_{1i}c_0 + x_{1j}c_0'')}) \\
&= \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'} h_0^{-c_0'} g_1^{-c_0' x_{1i}'}) \\
&= \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'}(h_0 h_i')^{-c_0'})
\end{aligned}$$

Še razlaga dveh substitucij:

$$c_0 + c_0'' = (c_0' x_{1j} - c_0' x_{1i}' + c_0' x_{1i}' - c_0' x_{1i})/(x_{1j} - x_{1i}) = (c_0' x_{1j} - c_0' x_{1i})/(x_{1j} - x_{1i}) = c_0'$$

in

$$\begin{aligned}
c_0 x_{1i} + c_0'' x_{1j} &= (c_0' x_{1j} x_{1i} - c_0' x_{1i}' x_{1i} + c_0' x_{1i}' x_{1j} - c_0' x_{1i} x_{1j})/(x_{1j} - x_{1i}) \\
&= (-c_0' x_{1i}' x_{1i} + c_0' x_{1i}' x_{1j})/(x_{1j} - x_{1i}) \\
&= c_0' x_{1i}'.
\end{aligned}$$

(Vse operacije so seveda v obsegu $\mathbb{Z}_q$.) ■

Opomba 6.8. *Iz opisa napada smo (zaradi preglednosti) izpustili korake, ki bi jih morala napadalca izvesti, da bi dobila certificiran par ključev na popolnoma slep način. Po zdajšnjem protokolu bi $\mathcal{B}$ namreč lahko izračunala x_{1i}', če bi vedela, kdaj poteka napad. Naj še omenim, da identifikacijske ključe x_{1i} drugih uporabnikov pozna le $\mathcal{B}$. Torej je zanemarljiva možnost, da bi napadalca komu podtaknila goljufijo (saj ne poznata njegovega identifikacijskega ključa), razen če mu ukradeta denarnico in vdreta v njegovega nadzornika.*

Poglejmo si sedaj, kako tovrstne napade preprečiti.

- Enostavna možnost je, da ne dovolimo vzporednih dvigov. Zadostuje že, da $\mathcal{B}$, ko izda a_0, zahteva odgovor c_0, preden nekomu drugemu spet izda a_0. En uporabnik bi verjetno še vedno lahko hkrati dvignil več kovancev, čeprav to ni očitno. Pri tem prijemu zagotovimo varnost brez večanja zahtevnosti protokolov, vendar deluje le, kadar je čas od vprašanja do odgovora zelo kratek ali pa so majhne potrebe po dvigu denarja.
- Kadar ta dva pogoja nista izpolnjena, naj bi preprosta modifikacija protokola za dvig in plačilnega protokola preprečila tovrstne napade. Podrobnosti ne bom opisoval, oglejmo si kar diagrama 6.3 in 6.4. Javni ključ $\mathcal{N}_i$ tokrat označimo s $f_i = g_1^{x_{1i}}$. Ideja modifikacije je v tem, da x_{1i} "premaknemo" za y, ta premik pa sporočimo šele skupaj z r_0. Tako je ključ $x_{1i} + y$, glede na katerega je konstruiran r_0, znan šele po oddaji c_0, zato opisan tip napadov ni mogoč.

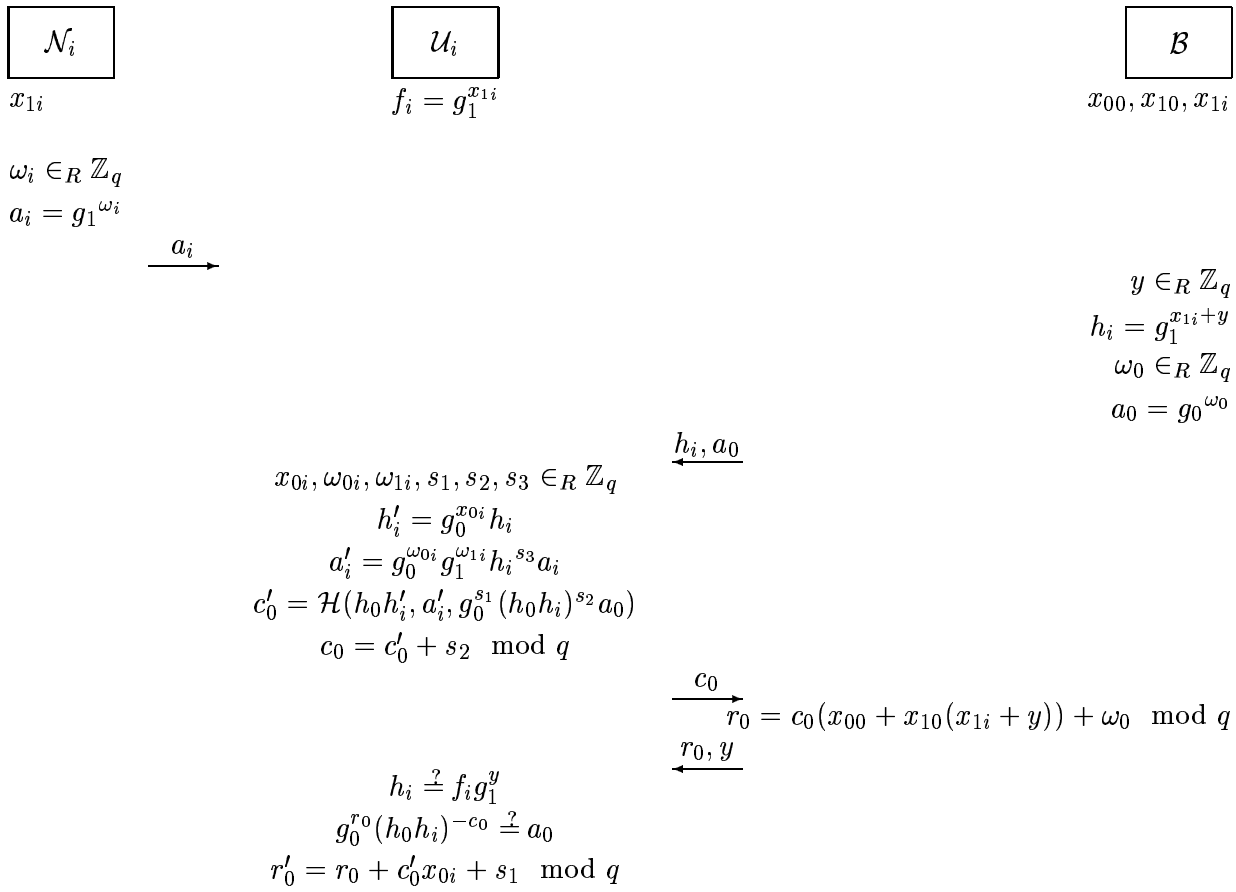

Diagram 6.3: Modificiran protokol za dvig denarja

6.6 REŠEVANJE MOŽNIH TEŽAV

V tem razdelku se bomo posvetili raznim težavam, ki lahko nastanejo pri delovanju sistema. Na nekatere smo že naleteli, ostale pa bomo še spoznali. V podrazdelku 6.3.6 smo si že ogledali, kaj stori banka, če je isti kovanec večkrat vnovčen. To si še enkrat preberimo, saj je pomembno za nadaljevanje.

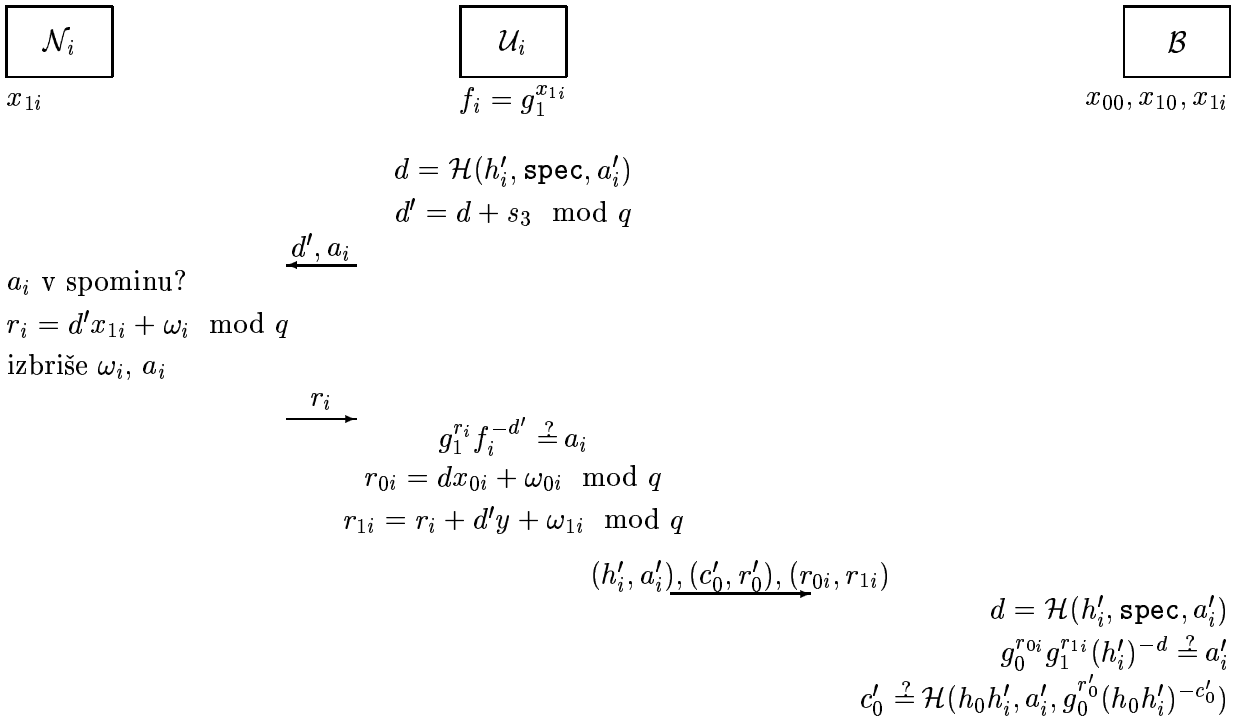

Diagram 6.4: Modificiran plačilni protokol

6.6.1 GOLJUFIJA BANKE

Banka lahko goljufa tako, da dviga kovance z računa uporabnika in jih poljubno porablja. Ker pozna identifikacijski ključ, lahko z njimi plača tudi večkrat in tako uporabniku naprti goljufijo. Večji problem je, da se lahko tudi pravi goljuf sklicuje na ta scenarij in trdi, da on večkrat porabljenih kovancev sploh ni dvignil.

Rešitev ponuja sledeči prijem. Banka mora za vsak izdani kovanec imeti podpisano zahtevo uporabnika. Ker ne pozna njegovega zasebnega ključa, te zahteve ne more ponarediti. Kadar banka trdi, da je uporabnik večkrat vnovčil isti kovanec, mora biti sposobna izračunati identifikacijski ključ. Ideja je v tem, da bi lahko bil identifikacijski ključ za vsak kovanec drugačen. $\mathcal{U}_i$ bi izbral $y \in_R \mathbb{Z}_q$ in bi ga sporočil v svoji podpisani zahtevi. Banka bi izdala r_0 glede na $x_{1i} + y$ (podobno kot v diagramu 6.3), v plačilnem protokolu pa bi lahko $\mathcal{U}_i$ sam pretvoril $r_i \to r_i + d'y$ (podobno kot v diagramu 6.4). Tako bi vsi protokoli še vedno delovali. V primeru goljufije bi banka razkrila identifikacijski ključ $x_{1i} + y$. Ob primerni porazdelitvi x_{1i} in y bi lahko oba izračunala. (Primer: če je q dolg 160 bitov, bi lahko bil $0 \leq y < 2^{40}$ in $x_{1i} \equiv 0 \mod 2^{40}$. Potem bi vsak identifikacijski ključ enolično določal x_{1i} in y.)
Banka tako ne bi mogla nelegalno izdajati kovancev, saj bi za določen $x_{1i} + y$ lahko $\mathcal{U}_i$ vedno dokazal, da je on legalno prejel drugačen kovanec. Moral bi razkriti kovanec $(h_i', a_i'), (c_0', r_0')$, s čimer bi za en kovanec izgubil nesledljivost, vendar je to še sprejemljivo. Banka gotovo ne bi tvegala, da jo ujamejo na goljufiji. Da se zagotovi avtentičnost kovanca, bi moral $\mathcal{U}_i$ priskrbeti še $(x_{0i}, \omega_{0i}, \omega_{1i}, s_1, s_2, s_3, r_0)$. Da se ne bi pojavil dvom o njihovi avtentičnosti, bi lahko $\mathcal{U}_i$ dodal $\mathcal{H}(x_{0i}, \omega_{0i}, \omega_{1i}, s_1, s_2, s_3)$ že v svojo podpisano zahtevo. Sodišče bi preverilo ali ti parametri res privedejo do navedenega kovanca. Če to drži, potem očitno goljufa banka, saj $\mathcal{U}_i$ drugače ne bi mogel dobiti kovanca.

Omenjena rešitev zahteva od uporabnika, da hrani podatke za vse doslej dvignjene kovance. Kadar to ni možno, lahko $\mathcal{U}_i$ podatke za en kovanec zašifrira, doda y in vse skupaj podpiše ter

pošlje banki, ki jih shrani poleg svojih podatkov o izdanem kovancu. Banka je v tem primeru obvezana, da za vsak sporen kovanec priskrbi pripadajoče podatke. Goljufati ne more, saj so podatki podpisani.

Naj omenim, da banka sicer lahko ponaredi plačilni dodatek za kovanec, ki je že bil vnovčen, saj pozna $\log_{g_0} g_1$, vendar z njim ne more podtakniti goljufije uporabniku. Da bi to storila, bi morala drugače generirati plačilni dodatek, za kar bi morala poznati x_{0i} in x_{1i}. Vrednosti x_{0i} ne pozna, izračunala pa bi jo lahko le, če bi znala izračunati diskretni logaritem v grupi G_q.

6.6.2 NEPOPOLNO IZVEDEN PROTOKOL

Vedno je možno, da je protokol nepopolno izveden. Lahko pride do napake pri prenosu ali do prekinitve zveze, oboje pa je možno tudi namerno. Zato je pomembno, da so naši protokoli kljub temu še vedno varni.

Vsaka izvedba vsakega protokola ima serijsko številko. Če se torej protokol predčasno prekine, ga je možno ponoviti z istimi vprašanji in odgovori. Morebitne napake se večinoma kmalu pokažejo, saj vsaka stran preverja logičnost odgovorov. V tem primeru se še enkrat ponovi isti dialog (ista vprašanja in odgovori), tokrat po možnosti brez napak. Izjema je le komunikacija z nadzornikom pri dvigu, kjer se napake ne pokažejo, zato je treba dodati mehanizem za detekcijo in odpravo napak.

Vsaka stran je dolžna na željo druge ponoviti protokol. Če tega ne stori, jo lahko k temu prisili sodišče.
Primer 1: Banka $\mathcal{B}$ lahko prekine protokol preden pošlje r_0, vendar ko je že trgala denar z računa $\mathcal{U}_i$. Če noče nadaljevati, jo lahko v to prisili sodišče, saj mora odliv denarja z računa opravičiti s podpisano zahtevo po dvigu. Lahko sicer trdi, da je že izdala kovanec, zato je najbolje, da $\mathcal{U}_i$ podpiše tudi c_0. Potem bo morala "ponovno" izdati tudi pripadajoči r_0.
Primer 2: Uporabnik $\mathcal{U}_i$ trdi, da je že plačal, trgovec $\mathcal{T}_j$ pa noče izdati računa. Uporabnik $\mathcal{U}_i$ lahko isto plačilo vedno ponovi. Morda $\mathcal{T}_j$ plačila res ni prejel, ampak kdorkoli lahko preveri, da je plačilo res namenjeno njemu. Torej bo moral vsaj naknadno potrditi prejem denarja (lahko izda račun ali pa vrne denar). Podobno je z izdajo računa. Da ohranimo anonimnost, račun ne vsebuje identitete plačnika, ampak (h'_i, a'_i) kovancev, s katerimi je bil plačan.
Tudi za polog velja, da lahko veljavnost plačilnega zapisa preveri vsakdo, zato mora banka prej ali slej veljaven polog sprejeti.

6.6.3 POVRNITEV IZGUBLJENEGA DENARJA

Lahko se zgodi, da nam denarnico ukradejo, da jo izgubimo ali pa da enostavno preneha delovati. V tem primeru se še vedno da povrniti izgubljeni denar. Ker denarnice zahtevajo avtorizacijo uporabnika, običajno ni nevarnosti, da bi kdo denar v njej porabil.

Večja težava je ugotoviti, kateri denar zahtevati nazaj, saj evidenco o porabljenih kovancih običajno vodi denarnica, ki pa je izgubljena. Priporočljivo je zato evidenco porabljenih kovancev voditi še na domačem računalniku. Če želimo povračilo kovancev, moramo imeti kopijo vseh podatkov, ki smo jih uporabljali pri dvigu. Te podatke lahko hrani tudi banka na način opisan v podrazdelku 6.6.1. Kljub temu moramo še vedno voditi vsaj evidenco dvignjenih in porabljenih kovancev. Ko zaprosimo za povračilo kovanca in priskrbimo potrebne podatke, banka preveri ali je bil ta kovanec že vnovčen. Če ni bil, nam ga izplača, vendar smo dolžni poravnati stroške, če ga kdo naknadno vnovči. Če stroške poravnamo, se to ne šteje za goljufijo. Če pa je bil kovanec že vnovčen, se napaka tolerira, vendar s tem za ta kovanec izgubimo nesledljivost.

Opisan sistem dobro deluje pri manjših zneskih, kjer banka še lahko zaupa uporabniku. K sreči običajno v denarnici ne nosimo veliko denarja. Pri večjih zneskih lahko banka počaka nekaj časa na morebitno vnovčenje, lahko pa tudi zahteva dodatno garancijo ali celo odpove vračilo. Lažje je, če imajo kovanci omejeno življenjsko dobo.

6.7 MOŽNE RAZŠIRITVE SISTEMA

Do sedaj smo opisovali sistem, ki omogoča uporabo le ene vrste navadnih kovancev. Sedaj si bomo ogledali, kako z minimalnimi spremembami sistem razširiti, da omogoča še kaj drugega.

6.7.1 RAZLIČNI KOVANCI

V praksi ne zadostuje le ena vrsta kovancev, ampak želimo uporabljati kovance različnih vrednosti, pa tudi prazne kovance in še kaj drugega. Klasičen prijem je, da za vsako vrsto kovancev $\mathcal{B}$ uporablja drugi zasebni ključ. Možnih vrst kovancev je lahko zelo veliko, tako da je koristno uporabiti naslednji prijem, da zmanjšamo število potrebnih ključev.

Pri našem sistemu to izgleda takole. Za vsako vrsto kovancev se uporablja drugačen x_{00} in pripadajoči h_0, medtem ko je x_{10} (in g_1) vedno enak. Naj bodo osnovne vrednosti kovancev 1,2,4,8, 16 SIT,...,2^{i-1},... ter njim pripadajoči zasebni ključi x_{00i} in javni ključi h_{0i} ($i > 0$). Zasebni ključ x_{00i} ustreza kovancu za 2^{i-1} SIT in prav tako pripadajoči javni ključ $h_{0i} = g_0^{x_{00i}}$. Če pa bi želeli kovanec za 11 SIT, bi uporabili zasebni ključ $x_{00} = x_{001} + x_{002} + x_{004}$ in pripadajoči javni ključ $h_0 = h_{01}h_{02}h_{04}$. Na ta način lahko preprosto dobimo ključe za vsako vrednost kovancev. Lahko dodamo tudi dodatne ključe, ki označujejo posebne tipe kovancev. Recimo x_{00p} in h_{0p} za prazne kovance ter x_{00c} in h_{0c} za čeke. Potem bi prazen kovanec za 5 SIT uporabljal javni ključ $h_{0p}h_{01}h_{03}$, ček za 6 SIT pa $h_{0c}h_{02}h_{03}$. Včasih iz samega kovanca ni očitno za kakšno vrsto kovanca gre. Da nam ni treba preverjati vseh možnosti za h_0, pomaga, če je kovancu priložen kratek opis. Potem je potrebno preveriti le ali kovanec ustreza opisu.

6.7.2 ČEKI

Kot smo že opisali v prejšnjem poglavju, poznamo dve vrsti anonimnih čekov: čeke s povračilom in čeke s števcem. Pri obeh lahko porabimo celotno vrednost čeka ali pa le del. Čeki s povračilom v našem sistemu niso uporabni, saj jih učinkoviti deljivi kovanci (opisali jih bomo v podrazdelku 6.7.4) v vsem prekašajo.

Čeki s števcem so za nas uporabni in si jih bomo ogledali. Opisan pristop žal omogoča vzpostavitev prikritega kanala in tudi sicer delno omogoča sledljivost. Problem je sicer mogoče rešiti, vendar mi je znana le precej manj učinkovita rešitev. Zato si bomo kljub slabostim ogledali enostaven pristop.

Oglejmo si protokole za dvig denarja, dvig čekov in plačilo.

Dvig denarja :

Banka $\mathcal{B}$ na zahtevo uporabnika $\mathcal{U}_i$ izda podpisan par $(vrednost, seq)$, kjer je seq zaporedna številka dviga z danega računa. Poleg tega odšteje $vrednost$ z računa uporabnika $\mathcal{U}_i$. Uporabnik $\mathcal{U}_i$ preveri ali je izdani zapis pravilno zgeneriran in ga pošlje naprej nadzorniku $\mathcal{N}_i$, ki ga preveri in prišteje $vrednost$ svojemu števcu. Ker se zapis posreduje brez zaslepitve, je treba format sporočila tako specificirati, da je možnost uporabe prikritega kanala čim manjša.

Dvig čekov :

Dvig čekov poteka tako kot dvig običajnih kovancev. Razlika je le v tem, da h_0 specificira, da gre za ček in določa njegovo maksimalno vrednost (primer: $h_0 = h_{0c}h_{02}h_{03}$).

Plačilo :

Pri plačilu sme nadzornik $\mathcal{N}_i$ odobriti uporabo čeka le, če je znesek plačila manjši od stanja števca. Torej mora $\mathcal{N}_i$ vedeti znesek plačila in izdati tako dovoljenje za uporabo čeka, da lahko ček uporabimo le za ta znesek. Enostaven pristop je, da znesek plačila dodamo v `spec`. Izziv d bo izračunal kar $\mathcal{N}_i$, ki mu bo moral $\mathcal{U}_i$ sporočiti tudi vrednosti (h'_i, a'_i) (v tem primeru je $s_3 = 0$).

Očitno je, da je pri takem sistemu mogoče povezati pogleda $\mathcal{N}_i$ in $\mathcal{T}_j$, torej je mogoče ugotoviti identiteto plačnika. Vse omenjene slabosti sistema veljajo le, če $\mathcal{N}_i$ pride v roke banki. Če pazimo, da se to ne zgodi, potem je ta sistem za uporabnika enako varen kot osnovni sistem. Za banko je tveganje predvsem, če nekdo vdre v nadzornika. Potem bo lahko izdajal čeke za maksimalno vrednost ne glede na stanje števca. Vendar je možnost goljufije omejena, saj bo banka kmalu zaznala, da $\mathcal{U}_i$ pogosto dviguje čeke in skoraj nikoli denarja. Da preprečimo večje škode, se zato tovrstni čeki uporabljajo le za minimalne zneske (namesto drobiža).

6.7.3 PRENOSLJIVOST

Ideje prenosljivosti denarja smo dokaj podrobno opisali že v podrazdelku 5.1.5.
Banka izda trgovcu $\mathcal{T}_j$ (in hkrati uporabniku) prazne kovance po običajnem protokolu za dvig. Razlika je le v javnemu ključu h_0 (in pripadajočemu zasebnemu ključu). Če maksimalna vrednost praznih kovancev ni omejena, velja $h_0 = h_{0p}$, če pa je vrednost omejena na recimo 20 SIT, pa velja $h_0 = h_{0p}h_{03}h_{05}$. Lahko bi imeli tudi prazne kovance s fiksno vrednostjo.

V plačilnem protokolu $\mathcal{T}_j$ zgenerira izziv d kot zgostitev praznega kovanca ($d = \mathcal{H}(h'_{ip}, a'_{ip})$) in ga pošlje uporabniku $\mathcal{U}_i$. Nadalje poteka protokol kot običajno. Ko hoče $\mathcal{T}_j$ kdaj kasneje s praznim kovancem plačati, stori to kot z običajnim kovancem, le da mora priložiti še osnovni kovanec in plačilni dodatek, saj šele to daje praznemu kovancu vrednost. Postopek lahko verižno teče naprej, vendar je treba pri vsakem plačilu priložiti celotno verigo kovancev (in dodatkov). Ko ta veriga postane nepraktično velika, je koristno, da nekdo kovanec (z verigo vred) pošlje v banko. Banka pri pologu vstavi v bazo vnovčenih kovancev vse kovance iz verige. Če je bil kateri že vnovčen, po običajnem postopku poišče goljufa.

Ta sistem prenosljivosti omogoča povezljivost, saj vsak lahko v verigi spozna svoj kovanec, predhodnikov in naslednikov. V praksi se zdi ta slabost sprejemljiva.

6.7.4 DELJIVOST

Kot vemo iz podrazdelka 5.1.6 poznamo poljubno deljivost in deljivost s podkovanci. Poljubno deljivost omogočajo čeki s števcem, saj lahko porabimo poljuben znesek do maksimalne vrednosti čeka. Mi se bomo na tem mestu posvetili deljivosti s podkovanci, s katero lahko učinkovito razširimo naš osnovni sistem za elektronske kovance.

Naj bo en kovanec sestavljen iz n podkovancev. (n je lahko za vsak kovance drugačen in podkovanci so lahko različni.) Pri konstrukciji kovancev sestavo določimo s tem, da uporabimo $h_0 = \prod_{k=1}^{n} h_{0v(k)}$, kjer je $v(k)$ indeks ključa z enako vrednostjo kot podkovanec št. k. Kovanec sestavljen iz dveh podkovancev z vrednostjo 1 SIT in dveh podkovancev z vrednostjo 4 SIT, bi

uporabljal $h_0 = h_{01}h_{01}h_{03}h_{03}$. Podkovanci so urejeni po vrednosti ($i < j \Longrightarrow v(i) < v(j)$), tako da točno vemo, kakšno vrednost ima podkovanec na določenem mestu.

Ideja je, da nadzornik $\mathcal{N}_i$ za vsak podkovanec izbere svoj "ω_i", ki ga označimo z ω_{ik}. V plačilnem protokolu potem za vsak podkovanec izračuna svoj r_{ik}. Uporabnik $\mathcal{U}_i$ mu pošlje še množico številk podkovancev, ki jih želi porabiti. Označimo jo s $\mathcal{K}$. Za tiste podkovance, ki jih $\mathcal{U}_i$ želi porabiti ($k \in \mathcal{K}$), nadzornik $\mathcal{N}_i$ pošlje r_{ik} za ostale pa le $g_1^{r_{1ik}}$, ki jih še zmnoži skupaj. Tako bosta trgovec $\mathcal{T}_j$ in banka $\mathcal{B}$ lahko še vedno preverila veljavnost kovanca, vedela pa bosta tudi, kateri podkovanci so **aktivirani** (oz. vnovčljivi). Banka sprejme polog namreč le za tiste podkovance, za katere je znan r_{1ik}. Preverjanje veljavnosti je bolj zapleteno, da ohranimo nesledljivost, čeprav se določeni podatki ponovijo pri vsakem plačilu. Poglejmo si kar diagrama 6.5 in 6.6.

Opomba 6.9. *Pri preverjanju veljavnosti sem uporabil potenciranje z eksponentom iz G_q, kar sicer v splošni grupi G_q ne obstaja. Z notranjo operacijo potenciranja lahko razširimo grupe G_q, ki so podgrupe grupe $\mathbb{Z}_p^*$, za ostale modele grupe G_q pa mi tovrstne razširitve niso znane. Morda bi se dalo preverjanje izvesti tudi brez uvedbe nove operacije, vendar je to vprašanje še odprto. Ker banki zadostuje že $g_0^{x_{0i}}$, da lahko poveže dvig s plačilom, se zdi, da brez dodatne operacije anonimnega plačila ni mogoče izvesti, če je par (h_i', a_i') za več plačil enak.*

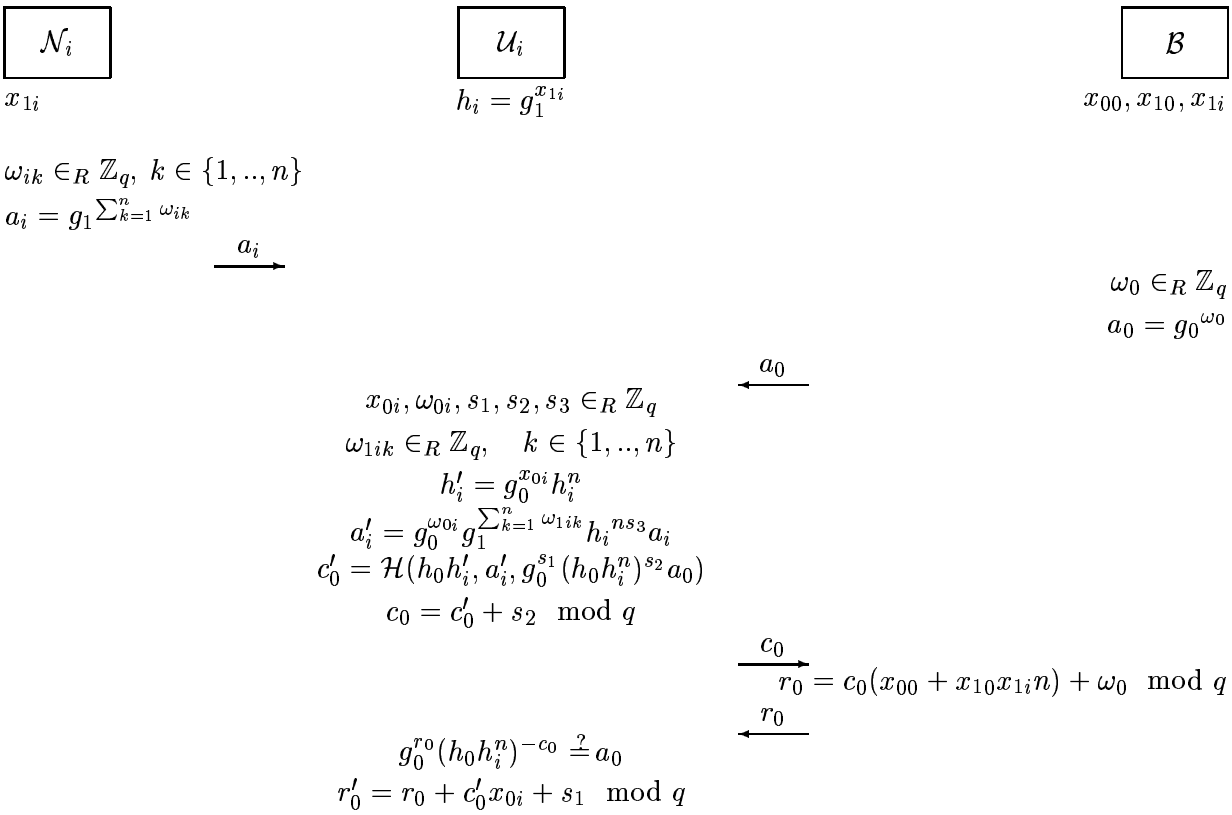

Diagram 6.5: Protokol za dvig deljivih kovancev

Polog potek kot običajno. Trgovec $\mathcal{T}_j$ pošlje celoten plačilni zapis v banko. Ta ga preveri in vnovči tiste podkovance, za katere ima v plačilnem zapisu r_{1ik}. Če je isti podkovanec že bil vnovčen, kar je možno le, če nekdo vdre v nadzornika, potem $\mathcal{B}$ iz r_{1ik} in r_{1ik}^* lahko izračuna identifikacijski ključ x_{1i}.

$$d = \mathcal{H}(h_i', \texttt{spec}, a_i')$$
$$d' = d + s_3 \mod q$$
$$d', a_i, \mathcal{K} \leftarrow$$

a_i v spominu in
$\mathcal{K}$ legalna zahteva?
$r_{ik} = d' x_{1i} + \omega_{ik} \mod q$
$r_i = g_1^{\sum_{k \notin \mathcal{K}} r_{ik}}$
označi porabljene podkovance
ali izbriše vse ω_{ik} in a_i

$$r_i, \{r_{ik}; k \in \mathcal{K}\} \rightarrow$$
$$r_i g_1^{\sum_{k \in \mathcal{K}} r_{ik}} h_i^{-d'n} \stackrel{?}{=} a_i$$
$$r_{0i} = d x_{0i} + \omega_{0i} \mod q$$
$$r_{1ik} = r_{ik} + \omega_{1ik} \mod q,\ k \in \mathcal{K}$$
$$r_{1i} = r_i g_1^{\sum_{k \notin \mathcal{K}} \omega_{1ik}} \mod q$$
$$R = g_0^{g_0^{r_{0i}} r_{1i}}$$
$$(h_i', a_i'), (c_0', r_0'), (R, \{r_{1ik}; k \in \mathcal{K}\}, \mathcal{K}) \rightarrow$$
$$d = \mathcal{H}(h_i', \texttt{spec}, a_i')$$
$$P = g_1^{\sum_{k \in \mathcal{K}} r_{1ik}} (h_i')^{-d}$$
$$R^P \stackrel{?}{=} g_0^{a_i'}$$
$$c_0' \stackrel{?}{=} \mathcal{H}(h_0 h_i', a_i', g_0^{r_0'} (h_0 h_i')^{-c_0'})$$

Diagram 6.6: Plačilni protokol za deljive kovance

Dokažimo še nekaj manj očitnih enakosti. (Oznaka $\sum_k$ pomeni isto kot $\sum_{k=1}^n$.)

$$\begin{aligned} r_i g_1^{\sum_{k \in \mathcal{K}} r_{ik}} h_i^{-d'n} &= g_1^{\sum_{k \notin \mathcal{K}} r_{ik}} g_1^{\sum_{k \in \mathcal{K}} r_{ik}} h_i^{-d'n} \\ &= g_1^{\sum_k r_{ik}} h_i^{-d'n} \\ &= g_1^{d' x_{1i} n \sum_k \omega_{ik}} h_i^{-d'n} \\ &= h_i^{d'n} a_i h_i^{-d'n} \\ &= a_i \end{aligned}$$

$$R^P = (g_0^{(g_0^{r_{0i}} r_{1i})})^P = g_0^{(g_0^{r_{0i}} r_{1i} P)} = g_0^{a_i'},$$

saj velja

$$\begin{aligned} g_0^{r_{0i}} r_{1i} P &= g_0^{r_{0i}} r_i g_1^{\sum_{k \notin \mathcal{K}} \omega_{1ik}} g_1^{\sum_{k \in \mathcal{K}} r_{1ik}} (h_i')^{-d} \\ &= g_0^{r_{0i}} g_1^{\sum_{k \notin \mathcal{K}} \omega_{1ik}} r_i g_1^{\sum_{k \in \mathcal{K}} r_{ik}} g_1^{\sum_{k \in \mathcal{K}} \omega_{1ik}} (h_i)^{-(d'-s_3)n} g_0^{-d x_{0i}} \\ &= g_0^{d x_{0i} + \omega_{0i}} g_1^{\sum_k \omega_{1ik}} h_i^{n s_3} g_0^{-d x_{0i}} r_i g_1^{\sum_{k \in \mathcal{K}} r_{ik}} h_i^{-d'n} \\ &= g_0^{\omega_{0i}} g_1^{\sum_k \omega_{1ik}} h_i^{n s_3} a_i \\ &= a_i'. \end{aligned}$$

Ostale enakosti si lahko bralec za vajo dokaže sam. Nisem dokazal, vendar se zdi, da je opisan sistem praktično enako varen kot osnovni sistem za navadne kovance. Vsekakor smo onemogočili, da bi iz podatkov, ki nastopajo pri vsakem plačilu (npr. x_{0i}), banka lahko ugotovila povezavo z dvigom denarja. Sistem je kljub dodatnim operacijam še vedno zelo učinkovit, vendar smo omejeni pri izbiri modela grupe G_q. Uporaba se izplača že, če sta v enem kovancu le dva podkovanca. Plačila z istim kovancem so seveda povezljiva.

6.7.5 UPORABA RAZLIČNIH VALUT

Nekateri kriptografi so se veliko ukvarjali s sistemi, ki bi omogočali uporabo različnih valut. Če želimo valute tudi menjati med seboj, potem moramo imeti še sistem ažuriranja tečajev in stvari se precej zapletejo. Precej lažje je na območjih, kjer so valute med seboj zamenljive po fiksnem tečaju in brez provizij, kot je to na primer na območju Eura.

Naš sistem lahko hrani različne tipe kovancev, torej tudi različne valute. Vrsta valute je lahko vsebovana v javnem ključu h_0. Kovanec za 10 SIT bi uporabljal $h_0 = h_{00SIT}h_{002}h_{004}$, za 20 DEM pa $h_0 = h_{00DEM}h_{003}h_{005}$.
Menjavo valute je najbolje prepustiti trgovcu $\mathcal{T}_j$. On naj torej izstavi uporabniku $\mathcal{U}_i$ račun v valuti, ki si jo ta želi. Dogovor o vrsti valute lahko opravi tudi sama denarnica in hkrati še preveri, da je bila menjava poštena. Tako lahko $\mathcal{U}_i$ vse izdatke primerja v valuti, ki je je navajen.

6.7.6 ZDRUŽITEV VSEH RAZŠIRITEV

Najprej smo opisali osnovni sistem, za tem prilagoditev za vzporedne dvige in podobne izboljšave, nato pa posebej vsako razširitev. Tako smo poskrbeli, da je bil vsak element in prijem karseda razumljiv. Brez posebnih omejitev lahko vse prijeme in razširitve tudi združujemo. Tako dobimo sistem, ki je v vsakem primeru varen, hkrati pa omogoča deljive kovance različnih valut in vrednosti. Omogoča tudi prenosljivost, za tiste, ki so pripravljeni tvegati nekaj anonimnosti, pa so na voljo tudi čeki s številcem. Sistem dobro prenese motnje in namerne napake ter omogoča tudi povračilo denarja v primeru izgube denarnice.

Seveda je tak kombiniran sistem še bolj kompleksen, zato ga ne bom opisoval. Kdor je do sem pozorno bral diplomo, bi ga verjetno znal celo sestaviti sam. Kljub kvalitetam velja poudariti, da kompleksen sistem zahteva tudi nekaj dodatnega dela, zato to ni vedno najboljša rešitev. Vedno moramo pretehtati zahteve in se odločiti za kombinacijo, ki je najbolj učinkovita, a še vedno izpolnjuje zahteve.

6.8 OCENA UČINKOVITOSTI

Najprej določimo model grupe G_q. Ker za denar želimo res dolgoročno varnost, bomo izbrali grupo eliptične krivulje $E(\mathbb{F}_{2^m})$, kjer je $m = 239$ in q je dolg 234 bitov (glej tabelo 2.2). Problem DLP v taki grupi po najhitrejši danes znani metodi zahteva $1.6 \cdot 10^{28}$ MIPS let, kar je več kot faktorizacija 4096-bitnega števila (ekvivalent pri sistemu RSA). Množenje v tej grupi in seštevanje ter množenje v $\mathbb{Z}_q$ bomo šteli za osnovne operacije. Za hranjenje elementa iz $\mathbb{Z}_q$ potrebujemo 234 bitov (30 bytov), za hranjenje elementa iz G_q pa 240 bitov (še vedno 30 bytov) (razlaga v razdelku 2.5). Rezultat standardne zgoščevalne funkcije SHA-1 je dolg 160 bitov (20 bytov).

Poglejmo sedaj kar v tabelah, koliko operacij potrebujemo pri raznih protokolih.

Protokol za dvig:

	$\mathcal{N}_i$	$\mathcal{U}_i$	$\mathcal{B}$
+/- v $\mathbb{Z}_q$	0	4	2
$\cdot$ v $\mathbb{Z}_q$	0	1	2
$\cdot$ v G_q	0	6	0
g^x v G_q	1	8	1
$\mathcal{H}(\cdot)$	0	1	0

Plačilni protokol:

	$\mathcal{N}_i$	$\mathcal{U}_i$	$\mathcal{T}_j$
+/- v $\mathbb{Z}_q$	1	4	2
$\cdot$ v $\mathbb{Z}_q$	1	1	0
$\cdot$ v G_q	0	1	4
g^x v G_q	0	2	5
$\mathcal{H}(\cdot)$	0	1	1

Protokol za polog:

	$\mathcal{T}_j$	$\mathcal{B}$
+/- v $\mathbb{Z}_q$	0	2
$\cdot$ v $\mathbb{Z}_q$	0	0
$\cdot$ v G_q	0	4
g^x v G_q	0	5
$\mathcal{H}(\cdot)$	0	1

Vidimo torej, da vsi protokoli potrebujejo le zmerno število operacij. Vendar vse operacije niso enako zahtevne. Potenciranje v našem modelu G_q zahteva v povprečju dobrih 100 množenj, če je osnova vnaprej znana (če osnova ni vnaprej znana, pa trikrat toliko). Torej je smiselno, da štejemo le potenciranja in zgoščevanja ($\mathcal{H}(\cdot)$). V ciklusu enega kovanca vsi skupaj opravijo 22 potenciranj in 4 zgoščevanja. Najmanj je obremenjen nadzornik $\mathcal{N}_i$. Njegovo vlogo lahko prevzame že navadni 8-bitni procesor na pametni kartici. Tudi vlogo uporabniškega modula $\mathcal{U}_i$ bi lahko prevzela pametna kartica, vendar bi morala biti precej zmogljivejša, saj je zaželjeno spodobno hitro izvajanje protokolov.

Glede prenosa podatkov so zahteve skromne. (Promet med $\mathcal{U}_i$ in $\mathcal{N}_i$ bomo kar zanemarili, saj znotraj ene naprave prenos ni problematičen.) Pri dvigu $\mathcal{U}_i$ in $\mathcal{B}$ izmenjata le $2*234+240 = 708$ bitov (89 bytov). Pri plačilu pa $\mathcal{U}_i$ in $\mathcal{T}_j$ izmenjata $3 * 234 + 2 * 240 + 160 = 1342$ bitov (168 bytov). Polog zahteva podoben prenos kot plačilo (podrobnosti so odvisne od `spec`-a).

Denarnica uporabnika $\mathcal{U}_i$ mora do plačila hraniti kovanec ter nekaj dodatnih informacij, kar znaša $5 * 234 + 2 * 240 + 160 = 1810$ bitov (227 bytov). Ostali podatki za primer spora so lahko shranjeni kje drugje. Banka $\mathcal{B}$ za vsak vnovčen kovanec shrani v bazo plačilni zapis $(h'_i, a'_i), (c'_0, r'_0), (\texttt{spec}, r_{0i}, r_{1i})$, kar znaša 1642 bitov (206 bytov), če je `spec` dolg 300 bitov. Banka $\mathcal{B}$ mora v praksi hraniti še podpisano zahtevo po dvigu in podatke, ki jih uporabnik $\mathcal{U}_i$ shrani zraven kovanca za primer goljufije, tako da bi vse skupaj lahko ocenili na 1 Kb/kovanec. Ker se večji del teh informacij zelo redko uporablja, so lahko shranjene tudi na trakovih, ki so poceni, a imajo dolg dostopni čas.

S tem smo zaključili z oceno osnovnega sistema in ocenimo še ostale rešitve in razširitve. Modificiran protokol za (vzporedni) dvig zahteva 2 dodatni potenciranji in 474 bitov (60 bytov) dodatnega prenosa. To ni ravno veliko, modificiran plačilni protokol pa sploh ne zahteva omembe vrednega dodatnega dela.

Uporaba različnih kovancev je mogoča brez dodatnega dela, saj se množenja ne štejejo. Isto velja za uporabo različni valut.

Prenosljivost pri prvem plačilu ne zahteva dodatnega dela. Pri nadaljnjih plačilih število operacij za preverjanja raste linearno z dolžino verige, saj je treba preveriti vsak kovanec v verigi. Tudi prenos raste linearno, saj je za vsak kovanec v verigi treba prenesti kovanec in plačilni dodatek. Z dolžino verige rastejo linearno tudi zahteve za hranjenje plačilnega zapisa.

Deljivost zahteva nekaj dodatnega dela, vendar še vedno precej manj, kot če bi namesto podkovancev imeli enako število kovancev. Žal ta razširitev ne deluje na grupi eliptične krivulje, zato moramo izbrati manj učinkoviti model, podgrupo grupe $\mathbb{Z}_p^*$. Dvig zahteva le 1 dodatno potenciranje in 0 dodatnega prenosa. Dodatno sicer obremeni nadzornika $\mathcal{N}_i$ in sicer po 1 potenciranje na vsak podkovanec, vendar jih je možno izračunati na zalogo, tako da to ni problem. Slabša situacija je pri plačilu. Uporabnik $\mathcal{U}_i$ in trgovec $\mathcal{T}_j$ izvedeta skupaj 11 potenciranj (potenciranje brez znane osnove sem štel trojno) namesto 7 pri običajnem sistemu. Ker to število ni odvisno od števila podkovancev, se izplača že pri dveh podkovancih. Prenos za en sam aktiviran podkovanec je skoraj enak kot pri običajnem sistemu, za vsak dodatni aktiviran podkovanec pa je potrebno prenesti le en dodaten element $\mathbb{Z}_q$ (dolžina je odvisno od modela grupe G_q). Vidimo, da je sistem uporaben celo, če vsakič aktiviramo le en podkovanec. V tem primeru namreč profitiramo pri dvigu ter s tem, da imamo možnost porabiti poljubno število podkovancev. Če vedno porabimo celoten kovanec, sistem nima smisla, saj se v tem primeru bolje izkaže običajni sistem.

Čeki s števcem zahtevajo malo dodatnega dela le pri polnjenju števca (dvigu denarja), sicer pa je vse kot pri običajnem sistemu. Le $\mathcal{N}_i$ je pri plačilu bolj obremenjen, saj mora sam izračunati zgostitev.

Če pregledamo dobljene rezultate, vidimo, da vsi omogočajo ceneno praktično uporabo. Možne so raznorazne vrste implementacij od priročnih denarnic za vsakdanjo rabo do plačil prek interneta. V zadnjem času se odpira tudi možnost integracije z GSM aparati. Zaradi majhnih zahtev po prenosu jim je sistem pisan na kožo. Če bi le malenkostno zmanjšali grupo G_q, bi lahko plačali že z enim samim SMS[1] sporočilom.

Za konec si poglejmo še kalkulacije v velikem merilu za primer, da bi sistem implementirali v Sloveniji. Recimo, da bi za celo Slovenijo (2 milijona uporabnikov) denar izdajala centralna banka. Vsak uporabnik bi v povprečju na dan dvignil 100 kovancev in jih prav toliko porabil. Povprečno bi pri sebi imel največ 1000 kovancev, saj je že to kar velik znesek. Življenjska doba sistema naj bo 100 let, življenjska doba kovancev pa naj ne bo dodatno omejena. Da omogočimo vse kovance med 0.01 SIT in 100,000 SIT bomo potrebovali $\lceil \log_2 10^7 \rceil = 24$ javnih ključev h_{0i} in pripadajočih zasebnih ključev x_{00i}. Če želimo omogočiti še razne tipe kovancev (in čekov), potrebujemo še nekaj dodatnih ključev. Vse skupaj je torej manj kot 30 javnih ključev, ki jih mora imeti vsak uporabnik $\mathcal{U}_i$ in še posebej trgovec $\mathcal{T}_j$. Če so ključi dobro varovani, jim lahko brezpogojno zaupamo, sicer pa moramo imeti še mehanizem za preklic ključev.

[1] SMS sporočila so kratka sporočila, ki si jih lahko izmenjujejo uporabniki GSM telefonov. Dolžina je omejena na 160 bytov.

Ocenimo sedaj predvsem prostorske zahteve za vsako "osebo". Nadzornik $\mathcal{N}_i$ mora hraniti ω_i in a_i za vsak kovanec, kar za 1000 kovancev nanese 60 Kb. To zmorejo boljše pametne kartice, cenene pa bi zmogle kakih 100 kovancev. Uporabnik $\mathcal{U}_i$ mora hraniti kovance in nekaj dodatnih informacij, kar znaša za 1000 kovancev 230 Kb. To za žepno napravo danes ni več problem. Podatke o porabljenih kovancih lahko redno odlaga na domači računalnik, tako da hrani le še neporabljene kovance.
Trgovec $\mathcal{T}_j$ mora hraniti plačilne zapise. Recimo, da na dan dobi 100,000 kovancev, potem to znese 21 Mb, kar zmore vsak računalnik. Če bi se več kot enkrat dnevno povezal z banko, bi bila ta številka še manjša. Kadar si med seboj plačujejo navadni uporabniki, gre za precej manjše število kovancev, zato to zmore že običajna denarnica.
Banka je izmed vseh daleč najbolj obremenjena, vendar si lahko privošči močne računalnike in veliko prostora za podatke. V 100 letih bi banka izdala $100*365*2*10^6*100 \approx 7*10^{12}$ kovancev in prav toliko bi jih bilo vnovčenih. Za to bi potrebovala 7,000 Tb prostora, kar je že danes povsem sprejemljivo. Njeni specializirani računalniki bi na dan opravili $1.2*10^6$ potenciranj in $4*10^5$ zgoščevanj za izdajo in sprejem 200 milijonov kovancev. To zmore že današnji osebni računalnik. Še največ težav bi povzročala baza za iskanje že vnovčenih kovancev. Za vsako vrsto kovancev je baza lahko ločena, toda tudi kovancev iste vrste je lahko do 10^{12}. Če vsak kovanec v bazi zasede le 80 bitov (10 bytov), to skupaj znese 10 Tb. Tako velike baze za redno iskanje so danes še na meji tehnologije, toda tudi ocenjeno število kovancev se nanaša na prihodnost.

Kot vidimo je sistem že danes praktično izvedljiv, zato je torej le še vprašanje časa in navad, kdaj bo prešel v prakso. Čeprav smo kalkulacijo izvedli predvsem za osnovni sistem, velja podobno tudi za vse razširitve. Vsekakor koristi presežejo potencialne težave.

ZAKLJUČEK

Spoznali smo konkreten sistem digitalnega denarja, možnosti, ki jih ponuja in tudi razne pasti, na katere je treba paziti. Dokazali smo tudi tiste trditve in enakosti, ki so bistvene za razumevanje. Bralcu, ki je do sem pozorno bral in ima nekaj znanja računalništva, bi to, upajmo, zadostovalo, da se lahko loti implementacije. Kriptografske probleme smo, upam, dokaj dobro obdelali, kot vedno pa se pri implementaciji pojavijo razni računalniški problemi. Treba je določiti format in način prenosa podatkov, mesto in način shranjevanja podatkov, do zadnje podrobnosti definirati razne zahteve in postopke, itd. Poseben povdarek je treba posvetiti združljivosti s čim večjim številom naprav, saj lahko le tako omogočimo splošno uporabo.

Področje implementacije digitalnega denarja se šele razvija, zato standardov še ni. Glede na trenutni obseg plačil prek spleta, pa tudi glede na druge koristi, ki jih prinaša, lahko sklepamo, da se bo s to tržno nišo v prihodnosti ukvarjalo veliko ljudi. Razvili se bodo standardi, ki bodo omogočili vključitev širokega kroga uporabnikov in integracijo v že obstoječe delovne in plačilne procese.

Dokler vsega tega še ni, je verjetno težko prepričati banke v smiselnost in praktično varnost takega sistema. Zato predvidevam, da se bodo tovrstni sistemi najprej uporabili na področjih, ki so denarju zgolj podobna. Taki primeri so recimo nakupni boni, žetoni ali pa karte za kino. Vsekakor je precej lažje uvesti sistem, če so vsi "trgovci" pod nadzorom "banke", kot je to recimo pri žetonih za avtobus. Avtobusno podjetje bi začelo prodajati elektronske žetone, na avtobuse pa bi vgradilo potrebna plačilna mesta. (Za avtobusni promet že obstaja rešitev z uporabo pametnih kartic, ki pa je mnogo enostavnejša in ne temelji na sistemih za digitalni denar.)

Uvajanje digitalnega denarja bo torej postopen proces. Najprej se bo pojavil na zgoraj omenjenih ozkih področjih, kjer varnost ni tako bistvenega pomena, nato pa bo postopno začel nadomeščati klasične oblike denarja. Vsekakor bo verjetno preteklo še precej vode, preden bo zaupanje v digitalni denar enako kot v klasično gotovino. Čeprav gotovina gotovo še dolgo ne bo "izumrla", pa je že danes koristno, da se seznanimo z digitalnim denarjem in pripomoremo k njegovi uveljavitvi v informacijski družbi.

Za konec naj omenim še nekaj novosti, ki jih nisem zasledil v literaturi. Skoraj vse sheme in protokole v diplomi sem osnoval na Schnorrovi shemi. Zato sem moral prilagoditi že obstoječe sheme za slepi in omejeni slepi podpis, ki niso temeljile na Schnorrovi shemi. Shemo za omejeni slepi podpis sem tudi sestavil iz dveh sorodnih, da se je bolje vklopila v koncept diplome. V 6. poglavju sem osnovno shemo za digitalni denar sicer res povzel po Brandsu, toda vse ostale prilagoditve in razširitve sem sestavil sam. Posebej sta zanimivi učinkovita izvedba deljivosti in sistem shranjevanja uporabnikovih zaupnih podatkov pri banki. Viri, ki sem jih uporabljal, se niso ukvarjali z oceno praktičnosti sistemov, ki so jih opisovali. Jaz sem želel narediti še korak več proti praktični implementaciji, zato sem se lotil tudi tega.

Literatura

[Bar00] J. Barbič, "Schoofov algoritem", Diplomsko delo, Fakulteta za matematiko in fiziko v Ljubljani, 2000.

[BBC+94] J.-P. Boly, A. Bosselaers, R. Cramer, R. Michelsen, S. Mjølsnes, F. Muller, T. Pedersen, B. Pfitzmann, P. de Rooij, B. Schoenmakers, M. Schunter, L. Vallée, and M. Waidner, "The ESPRIT Project CAFE - High Security Digital Payment Systems", ESORICS 94, LNCS 875 (Berlin), Springer-Verlag, 1994, pp. 217–230.

[Ble99] G. Bleumer, "Many-Time Restrictive Blind Signatures", 1999, `http://citeseer.nj.nec.com/bleumer99manytime.html`.

[Bra] "Personal homepage of Stefan Brands", `http://www.xs4all.nl/~brands`.

[Bra93] S. Brands, "An Efficient Off-line Electronic Cash System Based On The Representation Problem.", 246, Centrum voor Wiskunde en Informatica (CWI), ISSN 0169-118X, 1993, p. 77.

[Bra94a] S. Brands, "Electronic Cash on the Internet", 1994, `http://citeseer.nj.nec.com/brands95electronic.html`.

[Bra94b] S. Brands, "Off-Line Cash Transfer by Smart Cards", Tech. Report CS-R9455, CWI, 1994.

[Bra94c] S. Brands, "Untraceable Off-line Cash in Wallets with Observers", Proc. CRYPTO '93 (Douglas R. Stinson, ed.), Springer-Verlag, 1994, pp. 302–318.

[Bra95a] S. Brands, "Off-Line Electronic Cash Based on Secret-Key Certificates", Proceedings of the Second International Symposium of Latin American Theoretical Informatics (LATIN '95) (Valparaiso, Chili), vol. 911, Springer-Verlag, 1995, pp. 131–166.

[Bra95b] S. Brands, "Restrictive Blinding of Secret-Key Certificates", 144, Centrum voor Wiskunde en Informatica (CWI), ISSN 0169-118X, 1995, p. 35.

[Bra95c] S. Brands, "Secret-Key Certificates", 103, Centrum voor Wiskunde en Informatica (CWI), ISSN 0169-118X, 1995, p. 16.

[Bra95d] S. Brands, "Secret-Key Certificates (continued)", 99, Centrum voor Wiskunde en Informatica (CWI), ISSN 0169-118X, 1995, p. 16.

[Bra97] S. Brands, "Rapid Demonstration of Linear Relations Connected by Boolean Operators", Theory and Application of Cryptographic Techniques, 1997, pp. 318–333.

[Bra98] S. Brands, "Electronic Cash", Handbook on Algorithms and Theory of Computation, CRC Press, November 1998.

[Bra00] S. Brands, "Rethinking Public Key Infrastructure and Digital Signatures", MIT Press, 2000.

[CAFa] "CAFE - Conditional Access For Europe", `http://www.semper.org/sirene/projects/cafe`.

[CAFb] "CAFE - OPERA", `http://www.ercim.org/publication/Ercim_News/enw30/hirschfeld.html`.

[Can97] R. Canneti, "Toward realizing random oracles: Hash functions that hide all partial information", Proc. of CRYPTO '97, 1997, pp. 455–469.

[CFT98] A. H. Chan, Y. Frankel, and Y. Tsiounis, "Easy Come - Easy Go Divisible Cash", Theory and Application of Cryptographic Techniques, 1998, pp. 561–575.

[Cha83] D. Chaum, "Blind Signatures for Untraceable Payments", Proc. CRYPTO '82 (New York) (R. L. Rivest, A. Sherman, and D. Chaum, eds.), Plenum Press, 1983, pp. 199–203.

[Cha90] D. Chaum, "Zero-Knowledge Undeniable Signatures", Theory and Application of Cryptographic Techniques, 1990, pp. 458–464.

[CP] R. Cramer and T. Pedersen, "Improved Privacy in Wallets with Observers", `http://citeseer.nj.nec.com/366607.html`.

[CP93a] D. Chaum and T. Pedersen, "Transferred cash grows in size", Proc. EUROCRYPT '92, LNCS 658 (New York), Springer-Verlag, 1993, pp. 390–407.

[CP93b] D. Chaum and T. Pedersen, "Wallet Databases with Observers", Advances in Cryptology: Proc. of CRYPTO '92, LCNS 740, Springer-Verlag, 1993, pp. 89–105.

[CPS96] J. Camenisch, J.-M. Piveteau, and M. Stadler, "An Efficient Fair Payment System", ACM Conference on Computer and Communications Security, 1996, pp. 88–94.

[CvA90] D. Chaum and H. van Antwerpen, "Undeniable signatures", Proc. CRYPTO 89, Springer-Verlag, 1990, pp. 212–217.

[DN88] A. Fiat D. Chaum and M. Naor, "Untraceable Electronic Cash", Advances in Cryptology - CRYPTO '88, 1988, pp. 319–327.

[eCa] "eCash Technologies, Inc.", `http://www.digicash.com`.

[ECT] "Eliptic Curve Online Tutorial", `http://www.certicom.ca`.

[EO94] T. Eng and T. Okamoto, "Single-Term Divisible Electronic Coins", Theory and Application of Cryptographic Techniques, 1994, pp. 306–319.

[Esp] "Esprit, the EU information technologies programme", `http://www.cordis.lu/esprit/home.html`.

[Fer93] N. Ferguson, "Single Term Off-Line Coins", Theory and Application of Cryptographic Techniques, 1993, pp. 318–328.

[Fer94] N. Ferguson, "Extensions of single-term coins", Advances in Cryptology — CRYPTO '93 Proceedings, Springer-Verlag, 1994, pp. 292–301.

[FS87] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems", Advances in Cryptology — CRYPTO '86 (New York), Springer-Verlag, 1987, pp. 186–194.

[FTY97] Y. Frankel, Y. Tsiounis, and M. Yung, "Indirect Discourse Proofs: Achieving Efficient Fair Off-Line E- cash", Asiacrypt '96, LNCS 1163 (Berlin), Springer-Verlag, 1997, pp. 287–300.

[Mik01] M. Mikac, "Evidenca poštnih plačil v digitalni dobi", Diplomsko delo, Fakulteta za matematiko in fiziko v Ljubljani, 2001.

[Oka95] T. Okamoto, "An Efficient Divisible Electronic Cash Scheme", Proc. of CRYPTO '95, 1995, pp. 438–451.

[OO88] K. Ohta and T. Okamoto, "A modification to the Fiat-Shamir scheme", Proc. of CRYPTO '88, Springer-Verlag, 1988.

[OO89] T. Okamoto and K. Ohta, "Divertible Zero-Knowledge Interactive Proofs and Commutative Random Self-Reducibility", Advances in Cryptology – EUROCRYPT '89, LNCS 434, Springer-Verlag, 1989, pp. 481–496.

[OO90] T. Okamoto and K. Ohta, "Disposable Zero-Knowledge Authentication and their Applications to Untraceable Electronic Cash", Advances in Cryptology — CRYPTO '89 Proceedings, Springer-Verlag, 1990, pp. 134–149.

[OO92] T. Okamoto and K. Ohta, "Universal Electronic Cash", Proc. of CRYPTO '91, LNCS 576, Springer-Verlag, 1992, pp. 324–337.

[Pes00] L. Pesonen, "A Comparison of Chaum's and Brands' Electronic Cash Schemes", 2000, `http://citeseer.nj.nec.com/401417.html`.

[PS96] D. Pointcheval and J. Stern, "Security Proofs for Signature Schemes", Theory and Application of Cryptographic Techniques, 1996, pp. 387–398.

[sciAJ97] A. Jurišić in A. J.Menezes, "Elliptic Curves and Cryptography", Dr. Dobbs Journal (1997), 26–37.

[SPC95] M. Stadler, J.-M. Piveteau, and J. Camenisch, "Fair Blind Signatures", Theory and Application of Cryptographic Techniques, 1995, pp. 209–219.

[Sti95] Douglas R. Stinson, "Cryptography: Theory and Practice", CRC Press, 1995.

[vA90] H. van Antwerpen, "Electronic cash", Magistersko delo, CWI, 1990.

[Vid89] I. Vidav, "Alegebra", Mladinska knjiga, 1989.

[Way96] Peter Wayner, "Digital Cash: Commerce on the Net", Academic Press, 1996.