UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Računalništvo in matematika – 2. stopnja

Tim Kalan

VEČSTRANSKI SCHNORROVI PODPISI

Magistrsko delo

Mentor: doc. dr. Tilen Marc

Ljubljana, 2025

Kazalo

Večstranski Schnorrovi podpisi

Povzetek

Tukaj napišemo povzetek vsebine. Sem sodi razlaga vsebine in ne opis tega, kako je delo organizirano.

Schnorr Multisignatures

Abstract

An abstract of the work is written here. This includes a short description of the content and not the structure of your work.

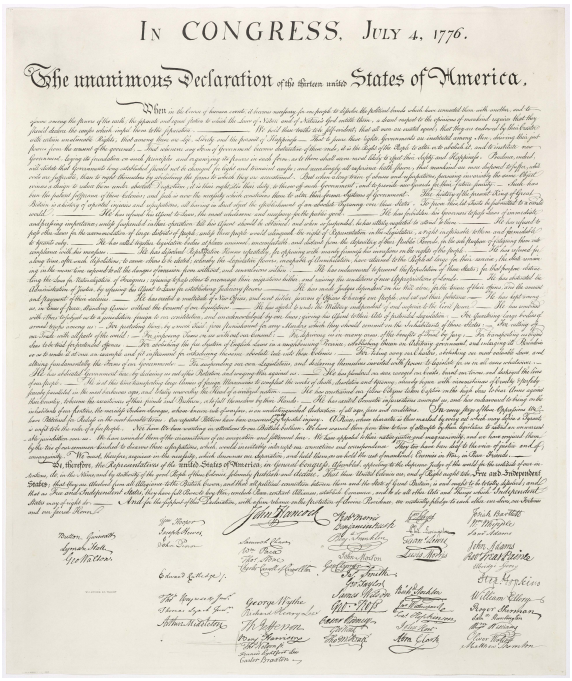
Math. Subj. Class. (2020): 94A60, 11T71

Ključne besede: kriptografija, digitalni podpis, Schnorrov podpis, večstranski podpis

Keywords: cryptography, digital signature, Schnorr signature, multisignature

1 Uvod

Odkar se je na svetu pojavil koncept (ročnega) podpisa, je večina primerov uporabe temeljila na pridobivanju podpisov več deležnikov, saj je bila večina podpisanih dokumentov sporazum ali pogodba med večimi deležniki. Odličen primer je npr. Deklaracija neodvisnosti Združenih držav Amerike, vidna na sliki 1.



Slika 1: Deklaracija neodvisnosti Združenih držav Amerike s podpisi podpornikov spodaj. Vir slike Wikipedia [25].

S pojavom računalnika in napredkom v kriptografiji se je pojavila alternativna oblika podpisovanja. *Digitalni podpisi* so dandanes povsod. Uporabljeni so vsakič, ko dostopamo do spletnih strani, prenašamo podatke ali pa opravljamo kakršnakoli plačila. Poleg avtomatiziranih podpisovanj, ki se zgodijo v ozadju zgoraj omenjenih procesov, pa so digitalni podpisi na voljo tudi kot alternativa ročnemu podpisu človeka. V praktično vseh pogledih so mnogo varnejši od tradicionalnih podpisov, omogočajo bolj sistematično preverjanje (prek računalnikov) in so skorajda enostavnejši za uporabo in prenašanje.

Digitalne podpise sta si prvič zamislila Diffie in Martin Hellman leta 1976 [26], ko sta predvidevala, da lahko take sisteme ustvarimo na podlagi enosmernih funkcij in asimetrične kriptografije. Prvi digitalni podpis, ki je bil dejansko implementiran in široko uporabljen, pa je bil *RSA* podpis, ki so si ga leta 1977 zamislili Ronald Rivest, Adi Shamir in Leonard Adleman [19].

Poleg individualnih podpisov, pa lahko digitalne podpise uporabimo tudi za podpisovanje skupin, npr. če mora skupina deležnikov podpisati pogodbo. Najenostavneje to dosežemo tako, da vsak član pripne svoj digitalni podpis. Če je skupina velika, če je računska moč omejena (npr. če želimo zmanjšati ceno transakcij pri tehnologiji veriženja blokov), ali pa če enostavno želimo preverjevalcu podpisov olajšati delo, lahko poskrbimo, da se skupina podpiše z enim samim, *večstranskim* podpi-

som. Ta podpis vseeno priča o vseh podpisnikih, njegova verifikacija oz. preverjanje pa zajema približno enako dela, kot preverjanje enega samega podpisa.

Taki podpisi so samo en izmed možnih načinov, kako se skupina podpiše. Od ostalih načinov izstopajo po tem, da omogočajo vpogled v sestavo skupine podpisnikov, kar nudi preverjevalcu več informacij pri odločanju, če je podpis sprejemljiv, hkrati pa ohranja odgovornost individualnih članov skupine. Pri tradicionalnih skupinskih podpisih podpis samo priča o podpisu celotne skupine, ustvari pa ga lahko katerkoli član. Take lastnosti so v določenih primerih zelo zaželene, npr. pri tehnologiji veriženja blokov je zelo pomembno, da je znano iz katerih naslovov prihajajo transakcije. To, v kombinaciji s popularizacijo Schnorrovega podpisa, je privedlo mnogo raziskovalcev do želje po večstranskih podpisih, ki vrnejo navadne Schnorrove podpise, in so tako enostavno zamenljivi.

Prva sta si večstranske podpise zamislila Itakura in Nakamura leta 1983 [8], vendar takrat še niso videli veliko uporabe. Velik napredek je bil dosežen, ko si je Schnorr leta 1989 zamislil Schnorrov podpis [3], ki se je izkazal za posebno primernega za večstranske podpise. Še vedno pa so bili takšni podpisi zanimivi bolj iz teoretičnega vidika. Leta 2001 se je pojavil prvi formalni model in dokaz varnosti za večstranske podpise, zahvaljujoč Micaliju, Ohti in Reyzinu [20]. Prvi primer široke uporabe se je pojavil z Bitcoinom [15]. Njegova popularnost je tudi popularizirala raziskovanje večstranskih podpisov, leta 2020 je izšel podpis MuSig2 [16], ki je bil prvi večstranski podpis, ki je bil primerno učinkovit za uporabo v Bitcoin omrežju. Široko je uporabljen še danes.

V poglavju 2 najprej predstavimo kriptografske osnove, ki so potrebne za razumevanje besedila. To zajema modularno aritmetiko in grupe, zgoščevalne funkcije, kriptografijo javnega ključa, nekaj splošnih definicij pri digitalnih podpisih in potem še nekaj besed o varnosti. V poglavju 3 predstavimo Schnorrov podpis, ki je osnova za ostale sheme, ki jih obravnavamo. V poglavju 4 predstavimo nekaj možnih pristopov za podpisovanje skupin. Poglavje 5 predstavi večstranski Schnorrov podpis, tam tudi dokažemo njegovo varnost v modelu slučajnega oraklja. V poglavju 7 primerjamo učinkovitost navadnega in večstranskega Schnorrovega podpisa. Na koncu pa v poglavju 6 omenimo še, kako se je področje razvijalo v zadnjem času.

2 Kriptografske osnove

Preden si lahko natančneje pogledamo, kako lahko skupina ustvari en sam podpis sporočila, si moramo pogledati nekaj kriptografskih osnov. Namen tega poglavja je predstaviti stvari, ki so predpogoj za branje praktično kakršnegakoli kriptografskega besedila s področja digitalnih podpisov.

2.1 Aritmetika v $\mathbb{Z}_p^*$

V kriptografiji imamo pogosto opravka z multiplikativnimi grupami, najenostavnejša med njimi (in tudi tradicionalno največ uporabljena) je *multiplikativna grupa naravnih števil modulo p*, ki jo označimo $\mathbb{Z}_p^*$. Njeni elementi so števila v $\{0, 1, \ldots, p-1\}$, ki so tuja številu p. V posebnem primeru, ko je p praštevilo, so to torej števila

$\{1, 2, \ldots, p-1\}$ in je red grupe (število elementov) $\text{ord}(\mathbb{Z}_p^*) = |\mathbb{Z}_p^*| = p - 1$. Operacija v tej grupi je, kot ime že nakazuje, množenje modulo p.

Spomnimo se, da je red elementa g v poljubni grupi najmanjše naravno število q, da velja $g^q = 1$, kjer je 1 enota za množenje. V primeru, da je p praštevilo, je grupa $\mathbb{Z}_p^*$ *ciklična*, kar pomeni, da v njej obstaja element g, katerega red je enak redu grupe, torej $\text{ord}(g) = p - 1$. V tem primeru se g imenuje *generator*.

Primer 2.1 (Grupa $\mathbb{Z}_{11}^*$)**.** Ker je 11 praštevilo, v grupi $\mathbb{Z}_{11}^*$ obstaja generator, oz. je grupa ciklična z redom $10 = 11 - 1$. Z zaporednim računanjem potenc lahko vidimo, da je $\text{ord}(2) = 10$, torej je 2 generator.

$$\begin{aligned} 2^1 &\equiv 2 \pmod{11} & 2^6 &\equiv 9 \pmod{11} \\ 2^2 &\equiv 4 \pmod{11} & 2^7 &\equiv 7 \pmod{11} \\ 2^3 &\equiv 8 \pmod{11} & 2^8 &\equiv 3 \pmod{11} \\ 2^4 &\equiv 5 \pmod{11} & 2^9 &\equiv 6 \pmod{11} \\ 2^5 &\equiv 10 \pmod{11} & 2^{10} &\equiv 1 \pmod{11} \end{aligned}$$

♢

Opomba 2.2. Spomnimo se *kongruence*: $a \equiv b \pmod{m} \iff m \mid a - b$.

Za konec tega podpoglavja si oglejmo še nekaj trditev, ki bodo pomembne pri dokazovanju pravilnega delovanja Schnorrovega podpisa.

Trditev 2.3. *Naj bosta p in q praštevili, kjer q deli $p - 1$. Naj bo g element grupe $\mathbb{Z}_p^*$ reda q, kar pomeni, da je $g^q \equiv 1 \pmod{p}$. Naj bo k naravno število. Potem velja*

$$g^k \bmod p = g^{k \bmod q} \bmod p.$$

Dokaz. Po osnovnem izreku o deljenju naravnih števil lahko k na en sam način zapišemo kot $k = nq + r$, kjer velja $n \in \mathbb{N}, r < q$.

Leva stran enačbe se potem prepiše

$$\begin{aligned} g^k \bmod p &= g^{nq+r} \bmod p = \\ &= (g^q)^n g^r \bmod p = \\ &= 1^n g^r \bmod p = \\ &= g^r \bmod p, \end{aligned}$$

kjer smo pri prehodu na tretjo vrstico upoštevali, da velja $g^q \equiv 1 \pmod{p}$. Desno stran pa lahko preoblikujemo na naslednji način:

$$\begin{aligned} g^{k \bmod q} \bmod p &= g^{(nq+r) \bmod q} \bmod p = \\ &= g^r \bmod p. \end{aligned}$$

Ker sta obe strani enaki, je trditev dokazana. □

Trditev 2.4. *Naj bodo a, b in p naravna števila. Potem za modularno množenje in potenciranje velja*

$$a \cdot b \bmod p = (a \bmod p) \cdot (b \bmod p) \bmod p, \tag{2.1}$$

$$a^b \bmod p = (a \bmod p)^b \bmod p. \tag{2.2}$$

Dokaz. Uporabimo osnovni izrek o deljenju naravnih števil, da a in b en sam način zapišemo kot

$$\begin{aligned} a &= n_a p + r_a, \\ b &= n_b p + r_b, \end{aligned}$$

kjer velja $r_a < p$ in $r_b < p$.

(2.1): Levo stran preoblikujemo

$$\begin{aligned} a \cdot b \bmod p &= (n_a p + r_a) \cdot (n_b p + r_b) \bmod p = \\ &= (n_a n_b p^2 + n_a p r_b + n_b p r_a + r_a r_b) \bmod p = \\ &= r_a r_b \bmod p, \end{aligned}$$

desno pa

$$\begin{aligned} (a \bmod p) \cdot (b \bmod p) \bmod p &= (n_a p + r_a \bmod p) \cdot (n_b p + r_b \bmod p) \bmod p = \\ &= r_a r_b \bmod p. \end{aligned}$$

Ker se strani ujemata, je trditev dokazana.

(2.2): Ker je potenciranje samo zaporedna uporaba množenj, lahko trditev pokažemo z indukcijo na b in enačbo (2.1):

- $b = 2$: Primer, ko je $b = 1$ (ali $b = 0$) je trivialen, če pa je $b = 2$, pa se problem reducira v
$$a \cdot a \bmod p \stackrel{?}{=} (a \bmod p) \cdot (a \bmod p) \bmod p,$$
kar drži neposredno po enačbi (2.1).
- $n \to n + 1$: Predpostavimo, da enačba (2.2) drži za $b = n$ (I.P.). Ko je $b = n + 1$, dobimo
$$\begin{aligned} a^{n+1} \bmod p = a^n a \bmod p &= \\ &\stackrel{(2.1)}{=} (a^n \bmod p)(a \bmod p) \bmod p = \\ &\stackrel{\text{I.P.}}{=} (a \bmod p)^n (a \bmod p) \bmod p = \\ &= (a \bmod p)^{n+1} \bmod p. \end{aligned}$$

S tem je indukcija končana in trditev dokazana. □

Trditev 2.5. *Naj bo G končna ciklična grupa reda q z generatorjem g. Naj bo n poljubno naravno število. Potem velja*

$$g^n = g^{n \bmod q}.$$

Dokaz. Po definiciji reda elementa grupe in generatorja grupe velja, da je

$$g^q = 1.$$

Po osnovnem izreku o deljenju naravnih števil lahko n na en sam način zapišemo kot $n = mq + r$, kjer velja $m \in \mathbb{N}$ in $0 \leq r < q$. Potem lahko zapišemo

$$g^n = g^{mq+r} = (g^q)^m g^r = 1^m g^r = g^{mq+r \bmod q} = g^{n \bmod q}.$$

□

2.2 Zgoščevalne funkcije

V grobem so (kriptografske) *zgoščevalne funkcije* take funkcije, ki prejmejo poljubno dolg binarni niz (ki lahko predstavlja besede, številke, celotne dokumente, ...), vrnejo pa binarni niz, ki ima vnaprej določeno dolžino. Tem rezultatom pravimo *zgostitve*. Namen zgoščevalnih funkcij je za dokument ustvariti kratek niz, ki unikatno identificira dokument. Želimo si, da je v praksi nemogoče najti dva različna niza z enako zgostitvijo. Natančno zgoščevalne funkcije definiramo:

Definicija 2.6. Kriptografska zgoščevalna funkcija $H : \{0,1\}^* \to \{0,1\}^n$ je funkcija, ki slika binarne nize m poljubne dolžine v njihove **zgostitve** $H(m)$, tj. binarne nize vnaprej določene dolžine n. Zadoščati mora naslednjim lastnostim:

- **Določenost** pomeni, da bo zgoščevanje enakih nizov vedno privedlo do enake zgostitve, tj. H je funkcija in ne naključni algoritem. Zapišemo lahko $\forall m : ((h_1 = H(m) \land h_2 = H(m)) \implies h_1 = h_2)$.
- **Učinkovitost** pomeni, da lahko računalnik izračuna poljubno zgostitev v doglednem času. Izračun zgostitve mora biti računsko učinkovit. Običajno tu zahtevamo, da je časovna zahtevnost funkcije H polinomska v dolžini vhodnega niza.
- **Enosmernost** oz. **odpornost na prasliko** pomeni, da iz predložene zgostitve zelo težko ugotovimo, kateri niz je funkcija prejela kot vhod. Če torej poznamo zgostitev h za nek neznan m, je računsko neizvedljivo najti niz m, da velja $h = H(m)$.
- **Odpornost na drugo prasliko** pomeni, da če poznamo niz in njegovo zgostitev, zelo težko najdemo drug niz z enako zgostitvijo. Če torej poznamo niz m_1 z zgostitvijo h_1, je računsko neizvedljivo najti zgostitev m_2, da velja $H(m_2) = h_1$.
- **Odpornost na trke** pomeni, da je računsko neizvedljivo najti dva niza m_1 in m_2, ki imata enako zgostitev, oz., da velja $H(m_1) = H(m_2)$.
- **Učinek plazu** pomeni, da vsaka sprememba v vhodnem nizu povzroči veliko in nepredvidljivo spremembo v zgostitvi. Če spremenimo en bit v vhodnem nizu, v povprečju pričakujemo, da se spremeni polovica bitov v zgostitvi.

Opomba 2.7. Vse varne kriptografske zgoščevalne funkcije, ki so v uporabi zadoščajo zgornjim lastnostim. To pomeni, da za nobeno od teh funkcij nihče še ni našel trka.

Primer 2.8. Ena izmed najbolj znanih zgoščevalnih funkcij je `SHA-256`. Njeno ime pomeni *Secure Hash Algorithm* (slov. varen zgoščevalni algoritem), 256 pa predstavlja dolžino vrnjene zgostitve. Pogostokrat to ime zasledimo pri nameščanju programske opreme, kjer služi kot avtentikator, da smo res naložili pravi program. Preverja namreč, da se zgostitvi naloženih in prenešenih datotek ujemajo.

Za primer si lahko ogledamo zgostitvi dveh podobnih nizov, *Ljubljana* in *Ljubljena*. Kljub podobnosti bomo videli, da sta rezultata popolnoma drugačna, kar si tudi želimo pri zgoščevalnih funkcijah.

```
SHA-256(Ljubljana) =
b7f147d8b4a6703a951336654355071f9752385f85d0860379e99b484aee7a82

SHA-256(Ljubljena) =
995d2d8ffb40e1838219e65dd2c665701ba34a90e11f7195a4b791838b6787fe
```

Za preglednost nismo prevajali besed v binarne nize, to bi storili npr. z `ASCII` ali `UTF-8` tabelo. Prav tako smo rezultat napisali v šestnajstiškem sistemu, saj je tako krajši. Iz rezultatov pa nazorno vidimo učinek plazu, saj sta popolnoma drugačna. ♢

2.3 Kriptografija javnega ključa

Prve šifre, ki smo jih uporabljali ljudje, so bile *simetrične*, kar pomeni, da sta osebi za komunikacijo obe morali poznati skriven *ključ*, s pomočjo katerega sta tako ustvarjali šifre, kot jih tudi dešifrirali. Ključ je običajno neko dolgo (binarno) število.

Primer 2.9 (Cezarjeva šifra)**.** Ena najbolj znanih šifer, ki izvira iz Antičnega Rima, je *Cezarjeva šifra*. Njen ključ je število, ki je krajše od dolžine naše abecede, v Cezarjevem primeru je bilo to število 3. Šifra potem deluje tako, da vsako črko zamakne za toliko mest v abecedi, kolikor definira ključ. Npr. za slovensko abecedo, bi šifra zamaknila črke:

```
A B C Č D E F G H I J K L M N O P R S Š T U V Z Ž
Č D E F G H I J K L M N O P R S Š T U V Z Ž A B C
```

To bi izraz `JAVNI KLJUČ` preslikalo v `MČARL NOMŽF`. Cezarjeva šifra se imenuje tudi *zamična šifra*. ♢

V prejšnjem stoletju pa se je pojavila alternativa, imenovana *asimetrična kriptografija*, oz. *kriptografija javnega ključa*. Glavna prednost te je, da osebi za komunikacijo ne rabita poznati enakega skrivnega ključa, vendar ima vsak od njiju par ključev, ki ju imenujemo **javni ključ** (angl. *public key*) in **zasebni ključ** (angl. *secret/private key*) in označimo kot par $(\mathrm{pk}, \mathrm{sk})$. Vsaka oseba objavi svoj javni ključ in poskrbi, da nihče ne izve, kaj je njen zasebni ključ.

Šifriranje potem poteka tako, da pridobimo javni ključ od osebe, s katero želimo komunicirati, ga uporabimo za šifriranje in objavimo šifrirano sporočilo. Lastnik ustreznega zasebnega ključa (vsakemu javnemu pripada natanko en zasebni) potem pridobi šifrirano sporočilo in ga z zasebnim ključem dešifrira. Kriptosistemi delujejo na način, da lahko sporočilo, šifrirano z javnim ključem dešifrira samo ustrezen zasebni ključ. Tako zagotovimo varno komunikacijo.

Primer 2.10 (RSA)**.** En prvih algoritmov javnega ključa, ki se uporablja še danes, je *RSA*. Njegova varnost izhaja iz (domnevne) težavnosti problema iskanja prafaktorjev velikega števila. Svoj ključ definiramo tako, da si izberemo dve (zelo veliki) praštevili p in q, ter ju zmnožimo v $n = pq$. Za primer vzemimo $p = 23$ in $q = 17$.

n je potem enak 391. Izbrati si moramo še eksponent e, vzemimo npr. $e = 3$. Naš javni ključ je potem par

$$(n, e) = (391, 3).$$

Postopek šifriranja poteka tako, da oseba, s katero komuniciramo, izbere sporočilo m, npr. $m = 10$, pridobi naš javni ključ, in izračuna šifro c kot

$$c = m^e \bmod n = 10^3 \bmod n = 218.$$

Dogovoriti se moramo še o zasebnem ključu. Za to bomo potrebovali eksponent za dešifriranje d, tako da bo veljalo

$$(m^e)^d \equiv 1 \pmod{\varphi(n)},$$

kjer φ označuje Eulerjevo funkcijo. Iščemo torej multiplikativni inverz eksponenta e, modulo $\varphi(n)$. V našem primeru je to $d = 235$. Zasebni ključ je potem

$$(p, q, d) = (23, 17, 235).$$

Iz zasebnega ključa torej lahko kadarkoli izračunamo javnega, saj enostavno zmnožimo p in q ter izračunamo inverz, v splošnem pa iz n učinkovito ne moremo pridobiti faktorjev p in q, kar nam zagotavlja varnost.

Ko prejmemo šifrirano sporočilo c, ga dešifriramo tako, da izračunamo

$$m = c^d \bmod n = 218^{235} \bmod 391 = 10.$$

♢

Poleg šifriranja, brez da bi si delili ključ, pa je kriptografija javnega ključa omogočila tudi *digitalne podpise*. Ti so uporabljeni vsakič, ko pošljemo transakcijo ali dostopamo do katerekoli spletne strani. Delujejo na podoben način kot šifriranje z javnim ključem, le da najprej uporabimo zasebni ključ na sporočilu, prek javnega ključa pa preverjamo veljavnost podpisa. Velikokrat sta šifrianje in podpisovanje uporabljena hkrati, saj tako pošljemo šifrirano sporočilo, ki ga lahko prebere le oseba, kateri je namenjeno, hkrati pa lahko preveri, da je res prišlo od nas.

2.4 Digitalni podpisi

Ideja *kriptografskih* ali *digitalnih podpisov* (angl. *digital signatures*) je, da služijo kot izboljšava človeškega ročnega podpisa. Za razliko od ročnega podpisa, lahko z digitalnim dosežemo pravo identifikacijo posameznika, ki temelji na njegovem zasebnem ključu. Tako smo lahko za digitalno podpisan dokument prepričani, da ga je res podpisal lastnik točno določenega zasebnega ključa (če predpostavimo, da podpisnik ključa ni posredoval nikomur).

Podpis dokumenta poteka nekoliko drugače, kot pri ročnih podpisih. Pri ročnem podpisu ta postane del dokumenta, digitalni podpis pa je od njega ločen, vseeno pa nastane s pomočjo zgostitve podpisanega dokumenta. Zato bo podpis za dva različna dokumenta vedno drugačen (dokler uporabimo varno zgoščevalno funkcijo).

Ostane še vprašanje preverjanja avtentičnosti podpisa. Pri ročnem podpisu to lahko storimo prek primerjave z znanim, preverjeno avtentičnim podpisom. Ta

postopek je zamuden in nenatančen, veliko večino ročnih podpisov je moč ponarediti z nekaj prakse. Preverjanje digitalnega podpisa pa temelji na kriptografiji javnega ključa. Ker je podpis nastal s pomočjo podpisnikovega zasebnega ključa, lahko s pomočjo ujemajočega javnega ključa preverimo avtentičnost.

Da se lažje pogovarjamo o kriptografskih sistemih, je smotrno definirati, kaj točno so deležniki, kot so podpisnik, preverjevalec in napadalec. V kriptografskih besedilih so pogosto definirani kot verjetnostni Turingovi stroji, mi pa se bomo izognili tej formalizaciji in se pogovarjali enostavno o *naključnostnih algoritmih*. Te lahko definiramo kot navadne, deterministične algoritme, ki imajo dostop do dodatnega parametra, *vira naključnih bitov* ω. Ta vir si lahko predstavljamo kot zelo dolg seznam naključnih bitov, ki ga algoritem lahko bere, ko ga potrebuje (npr. za generiranje naključnih števil). Branje je ponavadi enkratno dejanje; ko algoritem prebere bit, mora pri naslednjem klicu prebrati naslednji bit.

Definicija 2.11. Digitalni ali **kriptografski podpis** $\mathcal{S} = (\mathcal{P}, \mathcal{G}, \mathcal{S}, \mathcal{V})$ je četvorka učinkovitih algoritmov $\mathcal{P}$ za ustvarjanje parametrov podpisa, $\mathcal{G}$ za ustvarjanje ključa, $\mathcal{S}$ za podpisovanje in $\mathcal{V}$ za preverjanje podpisa. Definirana je nad končno množico možnih sporočil $\mathcal{M}$, vrnjeni podpis pa leži v končni množici podpisov Σ.

- $\mathcal{P}$ je algoritem za ustvarjanje javnih parametrov podpisa. Definira množico parametrov Par, ki so na voljo vsem deležnikom pri podpisu. V praksi je ta korak izpuščen, sodelujoči pri podpisu si izberejo shemo in uporabijo dobro poznane varne parametre. Formalno pa je to algoritem, ki prejme varnostni parameter k in vrne parametre Par, oz.

$$\mathrm{Par} = \mathcal{P}(k).$$

- $\mathcal{G}$ je naključnostni algoritem za ustvarjanje para ključev $(\mathrm{pk}, \mathrm{sk})$, ki za svoj vhod prejme parametre Par (javno dostopni ali pa ustvarjeni prek $\mathcal{P}$). Z javnim ključem pk lahko preverjevalec preveri avtentičnost podpisa, z zasebnim ključem sk pa podpisnik podpisuje. Formalno velja

$$(\mathrm{pk}, \mathrm{sk}) = \mathcal{G}(\mathrm{Par}).$$

- $\mathcal{S}$ je naključnostni algoritem, ki za svoja argumenta prejme zasebni ključ sk in sporočilo m, vrne pa podpis σ sporočila m z zasebnim ključem sk oz.

$$\sigma = \mathcal{S}(\mathrm{sk}, m).$$

- $\mathcal{V}$ je deterministični algoritem, ki preverja veljavnost podpisov. Za svoje argumente prejme javni ključ pk, sporočilo m in podpis σ, vrne $veljaven$, če je podpis veljaven, in $neveljaven$, sicer. Velja torej

$$\mathcal{V}(\mathrm{pk}, m, \sigma) = \begin{cases} veljaven, & \text{podpis veljaven,} \\ neveljaven, & \text{sicer.} \end{cases}$$

  Podpis σ sporočila m je veljaven, če je bil ustvarjen z uporabo algoritma $\mathcal{S}$ in zasebnim ključem sk, ki ustreza javnemu ključu pk.

Primer 2.12. Digitalni podpisi, obravnavni v tem besedilu, temeljijo na grupah. Algoritem $\mathcal{P}$ iz definicije 2.11 v tem primeru izbere parametre, ki določajo grupo G. Za varnostni parameter k lahko zapišemo

$$G = \mathcal{P}(k).$$

$\diamondsuit$

2.5 Varnost

Poglavitna lastnost vsakega kriptosistema je njegova *varnost*. Ker je namen digitalnih podpisov zagotoviti sogovorniku, da je sporočilo res poslal lastnik zasebnega ključa, je največja varnostna skrb, da bi *napadalec* lahko ponaredil pošiljateljev podpis in si s tem prisvojil njegovo identiteto. To napadalcu lahko uspe na več nivojih, ki jih lahko razvrstimo od najmanj do najbolj škodljivega:

- **Eksistencialno ponarejanje** (angl. *existential forgery*) pomeni, da obstaja sporočilo, za katerega napadalec lahko ustvari ponarejen podpis (torej podpis, pri katerem ni bil uporabljen zasebni ključ). To pomeni, da lahko najde vsaj en par sporočila in podpisa (m, σ), da velja $\mathcal{V}(\mathrm{pk}, m, \sigma) = veljaven$.

- **Selektivno ponarejanje** (angl. *selective forgery*) pomeni, da lahko napadalec z nezanemarljivo verjetnostjo podpiše poljubno izbrano sporočilo, ki ga lastnik zasebnega ključa še ni podpisal. Torej, če napadalcu nekdo predloži sporočilo m, lahko z nezanemarljivo verjetnostjo najde podpis σ, da velja $\mathcal{V}(\mathrm{pk}, m, \sigma) = veljaven$.

- **Popoln zlom** (angl. *total break*) pomeni, da napadalec pridobi zasebni ključ napadenega in s tem pridobi vse potrebne podatke za podpisovanje poljubnih sporočil v njegovem imenu.

Poleg zgoraj definiranih *ciljev napadalca*, lahko za vsak kriptosistem definiramo tudi *model napada*, ki ga zagotavlja shema. Stinson [21] definira naslednje modele:

- **Napad samo s ključem** je napad, kjer napadalec pozna javni ključ žrtve pk. Predpostavimo tudi, da napadalec vedno pozna delovanje sheme za podpisovanje $\mathcal{S}$ in ima dostop do javnih parametrov podpisa G. Z javnim ključem torej lahko preverja veljavnost podpisov, ni pa prejel nobenega podpisanega sporočila.

- **Napad z znanimi sporočili** je napad, kjer napadalec poseduje seznam parov sporočil in njihovih podpisov $(m_1, \sigma_1), (m_2, \sigma_2), \ldots$, kjer za vsak i velja $\sigma_i = \mathcal{S}(\mathrm{sk}, m_i)$.

- **Napad z izbranimi sporočili** je napad, kjer napadalec podpisniku da seznam sporočil $m_1, m_2, \ldots$, ta pa mu vrne seznam podpisov, da za vsak i velja $\sigma_i = \mathcal{S}(\mathrm{sk}, m_i)$. Napadalčev cilj je iz predloženih parov izvleči zasebni ključ, ali pa na nek drug način podpisati še ne podpisano sporočilo.

Ostane nam še pregled *tipov varnosti*, ki jo lahko pričakujemo oz. zahtevamo od sheme za podpisovanje. Takšna shema ne more biti *brezpogojno varna*, kar bi pomenilo, da je tudi z neomejenimi računskimi zmožnostmi nemogoče ponarediti podpis. To je zato, ker lahko napadalec sistematično preveri vse podpise za neko sporočilo s pomočjo algoritma $\mathcal{V}$, dokler ne najde pravega. Pričakujemo pa lahko *računsko varnost*, kar pomeni, da napadalec ne more najti ponaredka v doglednem času, če ima omejene računske sposobnosti, oziroma *dokazljivo varnost*, kar pomeni, da lahko varnost prevedemo na težavnost nekega matematičnega problema. Varnost večine shem za digitalne podpise temelji prav na (domnevni) težavnosti določenih matematičnih problemov.

2.5.1 Temelji varnosti

V primeru 2.10 smo omenili, da je varnost algoritma RSA odvisna od težavnosti problema iskanja prafaktorjev velikega števila. To je le eden izmed mnogih problemov, ki služijo kot osnova za varnost kriptografskih sistemov. V kriptografiji javnega ključa se pogosto srečamo s cikličnimi grupami, ki podpirajo množenje. V nadaljevanju bomo pogosto uporabili operacijo množenja, za izračune tipa

$$I = g^s,$$

kjer je g generator ciklične grupe, s pa zasebni ključ. Zaradi notacije bi morda kdo hitro pomislil, da lahko zgornjo enačbo obrnemo in s izračunamo kot

$$s = \log_g(I).$$

Taki izračuni v (nekaterih) cikličnih grupah žal (ali pa na srečo) niso tako enostavni, prišli smo do koncepta *diskretnega logaritma*. Pri izračunu logaritmov v množici realnih števil ključno vlogo igra koncept urejenosti. Ker je eksponentna funkcija strogo naraščajoča, realna števila pa urejena, lahko učinkovito izračunamo logaritme do poljubne natančosti (npr. z uporabo bisekcije). V diskretnih cikličih grupah, kot je npr. $\mathbb{Z}_p^*$, ni koncepta urejenosti, kar smo videli tudi v primeru 2.1 z grupo $\mathbb{Z}_{11}^*$. Odnos med g_1^s in g_2^s nam ne pove ničesar o odnosu med s_1 in s_2, zato nam »približki« logaritma ne pomagajo čisto nič. Za izračun takih logaritmov se moramo zanesti na drugačne metode, kot za izračun logaritmov v množici realnih števil.

Definicija 2.13 (Problem diskretnega logaritma [2])**.** Naj bo G ciklična grupa reda q, ki jo generira element g. Naj bo h naključni element iz grupe G. Naj velja $g^x = h$. Eksponent x imenujemo **Diskretni logaritem (DL)**.

Zamislimo si igro, kjer izzivalec in nasprotnik kot vhod prejmeta opis grupe G (torej $q \in \mathbb{N}$ in $g \in G$). Izzivalec potem izbere naključen element $\alpha \in G$ in izračuna $h = g^\alpha$. h pošlje nasprotniku, ta pa mora odgovoriti nazaj z elementom α. To igro imenujemo **problem diskretnega logaritma (PDL)** (angl. *discrete logarithm problem*).

Pri tej igri nas zanima verjetnost pravilnega odgovora nasprotnika, ki je računsko omejen. S tem mislimo, da ima na voljo polinomsko mnogo časa (glede na število bitov q). Če je grupa G takšna, da je verjetnost zanemarljiva, pravimo, da za grupo G drži *predpostavka o težavnosti diskretnega logaritma*.

Če naš kriptosistem torej živi v ciklični grupi, v kateri je problem diskretnega logaritma težek, nam to zagotavlja, da iz javnega ključa ni računsko izvedljivo pridobiti zasebnega. To je osnova za varnost mnogih kriptografskih sistemov, kot sta npr. ElGamalov sistem [7] in Schnorrov podpis [3].

Preden nadaljujemo, natančneje definirajmo, kaj pomeni, da je verjetnost zanemarljiva.

Definicija 2.14 (Zanemarljiva funkcija [2])**.** Zanemarljiva funkcija je taka funkcija $\varepsilon : \mathbb{N} \to \mathbb{R}$, da za vsako število $c > 0$ obstaja naravno število $n_0 \in \mathbb{N}$, da za vsak $n > n_0$ velja

$$|\varepsilon(n)| < \frac{1}{n^c}.$$

V nadaljevanju bomo v ε označevali poljubno zanemaljivo funkcijo.

To so torej funkcije, ki padajo proti nič hitreje, kot inverz kateregakoli polinoma. V kontekstu diskretnega logaritma želimo, da je verjetnost izračuna pravilnega odgovora nasprotnika zanemarljiva.

V besedilu bomo večkrat govorili o »zanemarljivih verjetnostih«, kar pomeni, da se bodo verjetnosti v odvisnosti od določene spremenljivke vedle kot zanemarljive funkcije.

2.5.2 Model slučajnega oraklja

Ko obravnavamo varnost kriptosistemov, se ponavadi pogovarjamo o *standardnemu modelu* kriptografije. Ta model ima samo eno predpostavko: napadalec je omejen samo s časom in količino računske moči, ki mu je na voljo (tu je običajno predpostavljeno, da ima napadalec realno računsko moč). Občasno se znajdemo v primeru, ko moramo za dokaz varnosti sprejeti dodatne predpostavke. V tem primeru govorimo o alternativnih modelih kriptografije.

Ko imamo opravka z zgoščevalnimi funkcijami, je pogosto potrebno sprejeti dodatne predpostavke, da lahko pokažemo varnost. Specifično, ko imamo opravka z zgoščevalno funkcijo $H : A \to B$, predpostavimo, da je bila ta funkcija izbrana naključno med vsemi funkcijami, ki slikajo A v B. To idealizirano verzijo zgoščevalne funkcije imenujemo **slučajni orakelj** (angl. *random oracle*). Za poljuben vhod torej vedno vrne enak odgovor, vendar je bil ta odgovor popolnoma naključno izbran.

Model slučajnega oraklja (angl. *random oracle model*) je model kriptografije, kjer poleg standardnih predpostavk, vsako uporabo zgoščevalne funkcije nadomestimo s slučajnim orakljem [2]. Predpostavimo, da imajo do oraklja dostop vsi vpleteni v kriptosistem, vključno z napadalcem.

Varnostni dokazi, ki temeljijo na uporabi takih slučajnih orakljev, vseeno pričajo o varnosti shem, ki namesto orakljev uporabljajo zgoščevalne funkcije, vsaj dokler je uporabljena zgoščevalna funkcija varna (npr. zanjo nihče še ni našel trka).

Opomba 2.15. V praksi imamo pogosto opravka z elementi grup in zgoščevalnimi funkcijami, ki slikajo en niz bitov v drugega. Zato je potrebno definirati funkcijo $\text{enc} : G \to \{0,1\}^*$, ki slika elemente grupe G v nize bitov. Prek te funkcije lahko pridobimo zgostitve poljubnih elementov grup.

2.5.3 Dokazovanje varnosti

Pogosto je cilj varnostnih dokazov pokazati, da je varnost neke sheme enakovredna težavnosti nekega problema, za katerega domnevamo, da je težek. V takih primerih govorimo o metodi *dokazovanja z redukcijo* (angl. *reduction proof*). Ta metoda temelji na predpostavki, da če bi napadalec lahko učinkovito napadel shemo, bi lahko tudi učinkovito rešil težaven problem.

En način, kako to storimo, je z uporabo *iger*. Potek dokaza varnosti v tem primeru je sledeč:

- Naj bo $\mathcal{S}$ podpisna shema, za katero želimo pokazati varnost (npr. da je verjetnost ponarejanja zanemarljiva). To shemo napada napadalec $\mathcal{F}$, ki je omejen s polinomskim časom. Napadalčevi interakciji s podpisno shemo rečemo *igra*, zanima pa nas verjetnost »zmage« napadalca, torej uspešnega ponarejenja podpisa.

- Definiramo še en algoritem $\mathcal{A}$. Cilj tega algoritma je izvesti redukcijo iz napada na reševanje težavnega problema. Algoritem $\mathcal{A}$ mora v polinomskem času prevesti uspešen napad napadalca $\mathcal{F}$ na shemo $\mathcal{S}$ v rešitev težavnega problema. Ponavadi to pomeni, da algoritem $\mathcal{A}$ sodeluje pri izvajanju podpisne sheme. Za dokaz analiziramo verjetnost uspeha algoritma $\mathcal{A}$, ki je seveda odvisna od verjetnosti uspeha napadalca $\mathcal{F}$.

  Navadno v dokazu predpostavimo, da je napadalec $\mathcal{F}$ uspešen z nezanemarljivo verjetnostjo, kar se prevede v nezanemarljivo verjetnost uspeha algoritma $\mathcal{A}$. Ker bi to pomenilo uspešno rešitev težavnega problema v polinomskem času, zaključimo, da je shema varna.

Za algoritem $\mathcal{A}$ predpostavimo naslednje lastnosti:

- **Nerazločljivost**: Napadalec $\mathcal{F}$ ne ve, ali komunicira s pravim podpisnikom ali z algoritmom $\mathcal{A}$. Če bi lahko napadalec razlikoval, potem bi lahko zaključili, da shema napadalcu razkrije neko skrivnost, saj je prisotnost skrivnosti edina razlika med pravim podpisnikom in algoritmom $\mathcal{A}$.

- **Brez skrivnosti**: Algoritem $\mathcal{A}$ ne pozna zasebnih ključev. Ta lastnost je ključna, saj nam pove, da napadalec prek interakcije z dejanskim podpisnikom ne izve nobene dodatne informacije. Če je napadalec uspešen pri interakciji z algoritmom $\mathcal{A}$ ali pa podpisnikom, ki ima zasebni ključ, potem smo lahko prepričani, da prisotnost zasebnega ključa ne pomaga pri napadu.

- **Prilagodljivost**: Algoritem $\mathcal{A}$ in napadalec med seboj komunicirata. Pogosto je potrebno, da algoritem $\mathcal{A}$ prilagodi svoje odločitve na podlagi napadalčevih odločitev.

  V modelu slučajnega oraklja prilagodljivost dobi dodatno dimenzijo. Algoritem $\mathcal{A}$ poleg celotnega protokola simulira tudi delovanje oraklja. To mu omogoča, da napadalca »previje« na prejšnje stanje, prilagodi odgovor oraklja in ponovno izvede korak sheme.

2.6 Interaktivni protokoli

V tem razdelku bomo govorili o *interaktivnih protokolih* (angl. *interactive protocols*). To so protokoli, kjer si dva sogovornika izmenjujeta sporočila, dokler ne prideta do skupnega zaključka. Pogosto se uporabljajo za namene avtentikacije, kjer ena stran dokazuje svojo identiteto drugi. Zaradi te uporabe, bomo sogovornika poimenovali *dokazovalec* (angl. *prover*) in *preverjevalec* (angl. *verifier*). Taki protokoli so sorodni digitalnim podpisom, pogosto pa so uporabljeni pri konstrukciji bolj zahtevnih kriptografskih shem.

Avtentikacijo lahko vidimo kot poseben primer dokazovanja. Dokazovalec želi dokazati preverjevalcu, da je res tisti, za katerega se izdaja. Protokoli, kjer je cilj dokazati neko trditev, sodijo v skupino *interaktivnih sistemov dokazovanja* (angl. *interactive proof systems*). Formalno morajo taki protokoli zadoščati dvema lastnostma:

- **Celovitost** (angl. *completeness*): Če trditev, ki se dokazuje, drži, potem bo preverjevalec sprejel dokaz dokazovalca (če noben od njiju ne bo goljufal).

- **Zadostnost** (angl. *soundness*): Če trditev, ki se dokazuje, ne drži, potem noben dokazovalec (tudi tak, ki goljufa) ne more predložiti dokaza, ki bi ga preverjevalec sprejel, razen z zanemarljivo verjetnostjo.

Dejanski interaktivni protokol pa potem deluje kot izmenjava sporočil, kjer dokazovalec poskuša prepričati preverjevalca, da trditev drži, preverjevalec pa s serijo vprašanj preverja, če je dokazovalec iskren in če trditev res drži. Pogosto predpostavimo, da ima dokazovalec neomejeno računsko moč, vendar ni iskren, preverjevalec pa je omejen na polinomsko računsko moč in je iskren.

Primer 2.16 (Reševanje sudokuja)**.** Za enostaven primer interaktivnega dokaza si oglejmo reševanje posplošenega Sudokuja. Tradicionalni Sudoku je problem, kjer moramo v 9×9 mrežo zapolniti števila od 1 do 9 tako, da se v vsaki vrstici, stolpcu in 3×3 kvadratu ne ponovijo nobena števila. Problem posplošimo tako, da ga prenesemo na poljubno mrežo velikosti $n^2 \times n^2$ in dovolimo uporabo števil od 1 do n^2.

Znano je, da je ta problem NP-poln [28], kar pomeni, da je težko najti rešitev, vendar je enostavno preveriti, če je rešitev pravilna. Če želi dokazovalec prepričati preverjevalca, da zna rešiti vse Sudokuje, lahko to storita tako, da si preverjevalec izbere naključne Sudokuje, jih pošlje dokazovalcu, ta pa jih reši in vrne rešitve. Preverjevalec nato preveri, če so rešitve pravilne. Po dovolj velikem številu preverjenih Sudokujev, bo preverjevalec lahko prepričan, da dokazovalec res zna reševati Sudokuje. ♢

Opomba 2.17. Interaktivni sistemi dokazovanja ne proizvedejo pravih dokazev v matematičnem smislu. Vedno obstaja določena verjetnost (ki je sicer zanemarljiva), da lahko goljufiv dokazovalec prepriča iskrenega preverjevalca o trditvi, ki ne drži. Ker pa je ta verjetnost zanemarljiva, nas to dejstvo ne moti.

2.6.1 Dokazi brez razkritja znanja

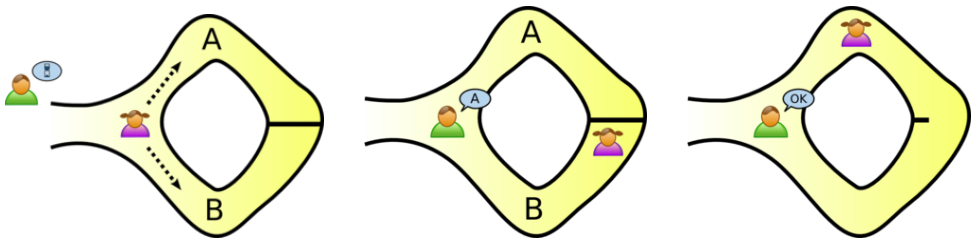
Dokazi brez razkritja znanja (angl. *zero-knowledge proofs*) so posebna vrsta interaktivnih sistemov dokazovanja, ki poleg celovitosti in zadostnosti zadoščajo še tretji lastnosti:

- **Brez razkritja znanja** (angl. *zero-knowledge*): Če trditev, ki se dokazuje, drži, potem noben preverjevalec (tudi tak, ki goljufa) ne bo iz dokaza izvedel ničesar več, kot samo to, da trditev drži.

To je torej orodje, prek katerega lahko dokazovale dokaže, da nekaj ve, brez da bi izdal, kaj to je.

Primer 2.18 (Jama Ali Babe [4])**.** V zgodbi o jami Ali Babe nastopata Ana in Bojan. Živita ob jami Ali Babe, ki je v obliki prstana. Pri vhodu sta na levo in desno vidni dve poti, ki se kasneje združita, vendar prehod preprečujejo vrata, ki jih lahko odpre samo skrivno geslo. Ana je ugotovila, kaj to geslo je, vendar ga ne želi povedati Bojanu, vseeno pa ga želi prepričati, da geslo pozna.

Da Ana Bojanu dokaže svoje znanje, si zamisli igro. Najprej bo ona odšla v jamo po eni izmed poti. Potem bo v jamo vstopil Bojan in povedal, če želi, da se Ana vrne po levi ali desni poti. Če se bo Ana dovolj velikokrat vrnila po poti, ki jo je povedal Bojan (in nikoli po napačni), bo lahko z veliko verjetnostjo prepričan, da Ana res pozna geslo. Primer enega kroga protokola je prikazan na sliki 2. ♢



Slika 2: Igra Ane in Bojana v jami Ali Babe. Vir slike Wikipedia [29].

2.6.2 Dokazi znanja brez razkritja znanja

Za namene kriptografije so pogosto zanimivi dokazi brez razkritja znanja. Da vzpostavimo zaupanje (npr. med dvema strankama, ki se ne poznata), pa ne želimo samo, da ena stranka dokaže obstoj neke skrivnosti, ampak tudi, da to skrivnost dejansko pozna. To je osnovna ideja *dokazov znanja brez razkritja znanja* (angl. *zero-knowledge proofs of knowledge*).

Odličen primer tovrstnega dokaza je dokaz znanja o tem, da poznamo diskretni logaritem dane vrednosti. Ta dokaz ima neposredno uporabo v kriptografiji, saj tako lahko dokažemo, da nek javni ključ res pripada nam. Denimo, da smo ustvarili par ključev (I, s), kjer je $I = g^s$ javni ključ, s pa zasebni ključ. Če nam preverjevalec predloži vrednost $I = g^s$, mi pa ga uspemo prepričati, da poznamo vrednost s, potem je to dovolj, da preverjevalec verjame, da je I res naš javni ključ (razen z zanemarljivo verjetnostjo).

Dokazovanje dejstva, da nek javni ključ res pripada nam prek interaktivnega protokola ima veliko uporabno vrednost. Tovrstne dokaze imenujemo *identifikacijski protokoli* oz. sheme.

Primer 2.19 (Schnorrov identifikacijski protokol)**.** En izmed najenostavnejših protokolov za dokaz znanja brez razkritja znanja je Schnorrov protokol [3]. Tesno je povezan s Schnorrovim podpisom, ki ga bomo spoznali v naslednjem razdelku. Protokol poteka med dokazovalcem, ki želi dokazati preverjevalcu, da pozna diskretni logaritem dane vrednosti, in preverjevalcem, ki želi to preveriti.

Pred začetkom se morata strinjati o ciklični grupi G reda q, ki jo generira element g. Ta grupa predstavlja ogrodje za protokol. Prav tako je obema na voljo javni podatek

$$I = g^s \in G.$$

Cilj dokazovalca je, da preverjevalca prepriča, da pozna vrednost $s \in \mathbb{Z}_q$, ne da bi mu o njej razkril karkoli drugega. Protokol poteka v več korakih:

1. **Zaveza:** Dokazovalec si enakomerno naključno izbere vrednost $r \in \mathbb{Z}_q$ in izračuna
$$X = g^r \in G.$$
To vrednost pošlje preverjevalcu.

2. **Izziv:** Preverjevalec si izbere naključno vrednost $e \in \mathbb{Z}_q$ in jo pošlje dokazovalcu.

3. **Odgovor:** Dokazovalec izračuna odgovor
$$y = r + es$$
in ga pošlje preverjevalcu.

4. **Preverjanje:** Preverjevalec preveri, če velja
$$g^y = X \cdot I^e.$$
Če enakost drži, preverjevalec verjame, da dokazovalec pozna vrednost s, saj je
$$g^y = g^{r+es} = g^r \cdot g^{es} = X \cdot I^e.$$

Varnost protokola temelji na težavnosti problema diskretnega logaritma in pa na dejstvu, da je izziv določen po zavezi. Zato dokazovalec ne more izračunati prepričjivega odgovora, preden dobi izziv. Bolj formalno bomo varnost dokazali pri dokazu varnosti Schnorrovega podpisa v razdelku 3.1. ♢

2.6.3 Fiat-Shamirjeva hevristika

Dokazi znanja brez razkritja znanja so torej odlično splošno orodje, s katerim se lahko prepričamo o identiteti sogovornika, ne da bi ta moral razkriti skrivni ključ. V praksi pa je težava, da so taki protokoli interaktivni, kar pomeni, da zahtevajo neposredno komunikacijo med dokazovalcem in preverjevalcem. Ta komunikacija je zamudna in draga.

Fiat in Shamir sta leta 1987 [18] predlagala način, kako lahko interaktivne dokaze znanja spremenimo v neinteraktivne. Osnovna ideja je, da namesto preverjevalca, breme izbire izziva prenesemo na slučajnega oraklja, do katerega imata dostop tako dokazovalec kot preverjevalec. Hevristika torej deluje samo v modelu slučajnega oraklja.

Primer 2.20 (Schnorrova shema in Fiat-Shamirjeva hevristika)**.** S pomočjo Fiat-Shamirjeve hevristike lahko Schnorrovo identifikacijsko shemo spremenimo v neinteraktivno, kar praktično neposredno privede do Schnorrovega podpisa, ki ga bomo spoznali v naslednjem razdelku.

Glavna ideja je, da se znebimo preverjevalčevega izziva. Namesto tega dokazovalec uporabi oraklja H, da pridobi zgostitev iz zaveze X in sporočila m, ki ga želi podpisati (ta korak tudi veže sporočilo na podpis):

$$e = H(\mathrm{enc}(X)||m),$$

kjer »||« označuje stikanje nizov. Na podlagi te zgostitve dokazovalec izračuna odgovor $y = r + es$ in ga pošlje preverjevalcu skupaj z vrednostjo X in e. Preverjevalec ima sedaj dovolj podatkov, da tudi sam izračuna vrednost zgostitve in preveri enakost.

Fiat-Shamirjeva hevristika tu ohranja varnost, saj dokazovalec še vedno ne more izračunati prepričljivega odgovora, preden izračuna zavezo (razen, če najde trk v H). Nekaj bitov varnosti se vseeno izgubi, ker napadalcu ni potrebno več čakati na odgovore, pač pa lahko izračuna poljubno mnogo zgostitev. Ob dovolj velikih parametrih to dejstvo ne predstavlja težav. Varnost bomo dokazali v razdelku 3.1. ♢

3 Schnorrov podpis

Eden izmed najenostavnejših, dokazano varnih podpisov je *Schnorrov podpis* [3]. Kot vsi podpisi, tudi ta potrebuje štiri algoritme: za ustvarjanje javnih parametrov, ustvarjanje ključa, podpisovanje sporočil in preverjanje podpisa.

Čeprav je Schnorr originalno [3] opisal podpis v multiplikativnih grupah naravnih števil modulo p, v shemi ni nič, kar bi preprečilo delovanje v poljubnih grupah. Zares je Schnorrov podpis mogoče posplošiti na katerekoli končne grupe, kjer obstaja učinkovit algoritem za množenje in je problem diskretnega logaritma 2.13 težek.

- **Parametri**: Naj bo G končna grupa reda p. V njej si izberemo element g reda q, pri čemer mora biti q dovolj veliko praštevilo (njegova velikost je odvisna od varnostnega parametra k). V splošnem pa sta lahko p in q tudi enaka. Ker računanje potega v podgrupi, ki jo določa g, se običajno privzame, da je G kar grupa reda q, ki jo generira g.

  Poleg grupe G si morata podpisnik in preverjevalec izbrati še varno kriptografsko zgoščevalno funkcijo $H : \{0,1\}^* \to \mathbb{Z}_q$. Funkcijo v dokazu varnosti modeliramo kot naključno, v praksi mora zadoščati vsaj lastnostim iz definicije 2.6. Velikost kodomene te funkcije definira velikost končnega podpisa. Iz zgostitve, dolge $\log_2 q$ bitov, dobimo podpis, dolg $2\log_2 q$ bitov [21].

- **Ključ**: Za ustvarjanje ključa si izberemo naključno število $s \in \mathbb{Z}_q$ in izračunamo

  $$I = g^s$$

  z uporabo učinkovitega algoritma za množenje. Javni ključ I je torej element grupe G, zasebni ključ s pa je element $\mathbb{Z}_q$. Ker smo predpostavili, da je v

grupi G problem diskretnega logaritma težek, iz javnega ključa I ni mogoče pridobiti zasebnega ključa s.

- **Podpis**: Za podpis sporočila $m \in \{0,1\}^*$ si najprej izberemo naključno število $r \in \mathbb{Z}_q$ in izračunamo *zavezo* (angl. *commitment*)

$$X = g^r.$$

Ta korak je popolnoma enak kot pri ustvarjanju ključa, vendar ima pomembno razliko. Zasebni ključ se ne spreminja, pri izbiri r pa je potrebno paziti, da je ta res naključna, in da se r ne ponovi (glej opombo 3.1).

Potem z uporabo zgoščevalne funkcije $H : \{0,1\}^* \to \mathbb{Z}_q$ izračunamo *izziv* (angl. *challenge*)

$$e = H(\text{enc}(X)||m).$$

Za konec je potrebno izračunati še

$$y = es + r \bmod q,$$

podpis sporočila m pa je potem par (X, y) oz.

$$\mathcal{S}(s, m) = (X, y).$$

- **Preverjanje**: Za preverjanje veljavnosti podpisa (X', y') sporočila m je potrebno najprej izračunati

$$e' = H(\text{enc}(X')||m)$$

in nato preveriti, če velja

$$g^{y'} \stackrel{?}{=} X' \cdot I^{e'}. \tag{3.1}$$

Ta enačba res preveri veljavnost Schnorrovega podpisa, saj lahko za veljaven podpis (X, y) sporočila m zapišemo

$$g^y \stackrel{2.5}{=} g^{es+r \bmod q} = g^{es+r} = g^r \cdot (g^s)^e = X \cdot I^e,$$

kjer smo uporabili trditev 2.5.

Ker Schnorrov podpis deluje v skoraj poljubnih končnih grupah, je zelo prilagodljiv in uporaben. V zadnjem času je precej popularna uporaba Schnorrovih podpisov v *eliptičnih grupah*. Te omogočajo izbiro manjših parametrov, kar naredi podpis bolj časovno in prostorsko učinkovit.

Opomba 3.1. V primeru, da je enak r uporabljen večkrat, podpisnik tvega, da lahko napadalec iz dveh njegovih podpisov izračuna njegov zasebni ključ s. Naj bosta (X_1, y_1) in (X_2, y_2) podpisa sporočil m_1 in m_2. Potem velja

$$y_1 - y_2 = e_1 s + r_1 - e_2 s - r_2 = (e_1 - e_2)s + (r_1 - r_2).$$

V primeru, da sta r_1 in r_2 enaka, lahko napadalec z enostavnim izračunom inverza $(e_1 - e_2)$ pridobi zasebni ključ s (za izračun e_1 in e_2 ima napadalec dovolj informacij, saj je izbrana zgoščevalna funkcija javna).

Izkaže se, da je lahko problematična že uporaba generatorja naključnih števil za pridobivanje r, ki ne vrača enakomerno porazdeljenih števil. Če napadalec dobi dovolj veliko količino sporočil in podpisov, lahko v tem primeru reši *problem skritega števila* in pridobi zasebni ključ [23].

3.1 Varnost Schnorrovega podpisa

Ko govorimo o varnosti Schnorrovega podpisa, imamo v mislih odpornost sheme proti eksistencialnem ponarejanju, kjer napadalec lahko za katerokoli sporočilo dobi veljaven podpis. Želimo torej, da napadalcu ne uspe ponarediti podpisa za nobeno sporočilo, ki še ni bilo podpisano.

Kot omenjeno na začetku poglavja, je varnost Schnorrovega podpisa odvisna od težavnosti problema diskretnega logaritma. Varnost Schnorrovega podpisa je zato odvisna od varnosti grupe, v kateri deluje. Cilj tega razdelka je pokazati, da je Schnorrov podpis varen, če je problem diskretnega logaritma težek.

Definicija 3.2 (Varnost podpisne sheme)**.** Naj bo k varnostni parameter. Naj bo F napadalec, ki deluje v polinomskem času. Definirajmo *podpisovalni eksperiment* $\mathrm{Sign}_F(k)$:

1. Na podlagi varnostnega parametra se definira par ključev (I, s), kjer je I javni ključ, s pa zasebni ključ.

2. Napadalec F prejme javni ključ I. Od nekega podpisnika P lahko zahteva podpise poljubno izbranih sporočil.

3. Napadalec F podpisniku P lahko pošlje sporočilo m, v odgovor pa dobi podpis σ. To lahko stori poljubno mnogokrat.

4. Napadalec F si izbere sporočilo m' in zanj izračuna podpis σ'. Eksperiment vrne 1, če je podpis σ' veljaven postup za sporočilo m', sicer vrne 0.

Cilj napadalca je torej ponarediti podpis za sporočilo, ki še ni bilo podpisano, brez da bi poznal zasebni ključ podpisnika. Podpisna shema je varna , če je verjetnost, da izvedba eksperimenta $\mathrm{Sign}_F(k)$ vrne 1, zanemarljiva v varnostnem parametru k.

Varnost Schnorrovega podpisa je mogoče dokazati tako, da pokažemo varnost Schnorrove identifikacijske sheme, opazimo, da nam uporaba Fiat-Shamirjeve hevristike vrne ravno Schnorrov podpis, in pokažemo, da je ta transformacija varna. Najprej pa moramo seveda definirati, kaj pomeni, da je identifikacijska shema varna in kako povežemo identifikacijsko shemo s sporočilom. V osnovi bomo rekli, da je shema varna, če napadalec ne more prepričati preverjevalca, da je on pravi dokazovalec, brez da bi poznal zasebni ključ. To mora veljati tudi v primeru, da napadalec lahko »posluša« (torej vidi sporočila) več pogovorov med dokazovalcem in preverjevalcem.

Definicija 3.3 (Varnost identifikacijske sheme)**.** Naj bo k varnostni parameter in T_{sk} orakelj, ki ne prejme nobenega vhodnega podatka, ob klicu pa vrne *transkript* ene izvedbe identifikacijske sheme. Transkript predstavlja eno izvedbo identifikacijske sheme z izbranimi javnimi parametri. Formalizira napadalčevo sposobnost »prisluškovanja« in mu omogoča, da pridobi vsa izmenjana sporočila med dokazovalcem in preverjevalcem. Naj bo F napadalec, omejen s polinomskim časom. Definirajmo *identifikacijski eksperiment* $\mathrm{Id}_F(k)$:

1. Na podlagi varnostnega parametra se definira par ključev (I, s), kjer je I javni ključ, s pa zasebni ključ.

2. Napadalec F prejme javni ključ I in neomejen dostop do oraklja T_{sk}.

3. Napadalec F na poljubni točki pošlje zavezo X, v odgovor pa dobi izziv e. Tudi na tej točki lahko napadalec kliče oraklja T_{sk}.

4. Napadalec F izračuna odgovor y. Eksperiment vrne 1, če odgovor y prepriča preverjevalca, da komunicira s pravim dokazovalcem (torej lastnikom zasebnega ključa), sicer vrne 0. V primeru Schnorrove identifikacijske sheme to ustreza preverbi, če velja $g^y = X \cdot I^e$.

Cilj napadalca je torej pridobiti diskretni logaritem javnega ključa I, s čimer se lahko izdaja za dokazovalca. Identifikacijska shema je varna pred pasivnim napadom (ali samo varna), če je verjetnost, da izvedba eksperimenta $\mathrm{Id}_F(k)$ vrne 1, zanemarljiva v varnostnem parametru k.

Izrek 3.4 (Varnost Schnorrove identifikacijske sheme [10])**.** *Naj bo G ciklična grupa, v kateri je problem diskretnega logaritma težek. Potem je Schnorrova identifikacijska shema v grupi G varna.*

Dokaz. Dokaz poteka z redukcijo uspešnega napada na rešitev problema diskretnega logaritma. Naj bo napadalec F naključnostni algoritem, ki teče v polinomskem času. Cilj napadalca je, da iz javnega ključa dobi zasebnega (izračuna diskretni logaritem javnega ključa). Napad poteka podobno, kot opisano v identifikacijskem eksperimentu iz definicije 3.3.

Na podlagi napadalca F konstruirajmo algoritem A, ki iz uspešnega napada napadalca F na Schnorrovo identifikacijsko shemo pridobi rešitev problema diskretnega logaritma. Algoritem A prejme vse javne parametre sheme (G, g, q, I) kot vhod. I je javni ključ, za katerega želi napadalec izračunati diskretni logaritem. Algoritem deluje v naslednjih korakih:

1. Algoritem A zažene napadalca F. Ta med svojim delovanjem kliče oraklja T_{sk}, ki napadalcu vrača transkripte izvedb identifikacijske sheme. Algoritem A mora te odgovore simulirati sam, to pa lahko stori tako, da izvede identifikacijsko shemo v obratnem vrstnem redu:

   - Najprej si izbere naključen odgovor $y \in \mathbb{Z}_q$ in izziv $e \in \mathbb{Z}_q$.
   - Preuredi enačbo za preverjanje

     $$g^y = X \cdot I^e$$

     v enačbo za izračun zaveze

     $$X = g^y \cdot I^{-e}.$$

     Tako lahko vrne veljaven transkript (y, e, X) napadalcu F.

   Ker računa v obratnem vrstnem redu, se izogne izračunu diskretnega logaritma, opraviti mora le eno eksponenciranje.

2. Ko napadalec pošlje zavezo X, algoritem A izbere enakomerno naključno število $e_1 \in \mathbb{Z}_q$, jo pošlje napadalcu F, ta pa odgovori z odgovorom y_1.

3. Algoritem A ponovno zažene napadalca F z enakimi parametri (torej tudi enakim virom naključnih bitov ω) a z različnim izzivom $e_2 \in \mathbb{Z}_q$. Napadalec F v tem primeru vrne odgovor y_2.

4. Če sta oba odgovora veljavna, torej velja

$$g^{y_1} = X \cdot I^{e_1} \text{ in}$$
$$g^{y_2} = X \cdot I^{e_2},$$

potem lahko algoritem A izračuna rešitev problema diskretnega logaritma kot

$$\begin{aligned} X = g^{y_1} \cdot I^{-e_1} &= g^{y_2} \cdot I^{-e_2} \\ g^{y_1} \cdot g^{-y_2} &= I^{e_1} \cdot I^{-e_2} \\ g^{y_1 - y_2} &= (g^s)^{e_1 - e_2} \\ s &= (y_1 - y_2)(e_1 - e_2)^{-1} \end{aligned}$$

Da dokažemo varnost, moramo sedaj obravnavati verjetnost uspeha napadalca F in algoritma A. Naj bo $V(\omega, e)$ indikator, da napadalec F vrne pravilen odgovor pri napadu, če je e izziv in ω naključni vir bitov. Označimo z δ_ω verjetnost uspeha po izzivu e pri fiksni vrednosti ω

$$\delta_\omega = \Pr_e(V(\omega, e) = 1).$$

S to vrednostjo lahko izrazimo verjetnost uspeha identifikacijskega eksperimenta

$$\Pr(\mathrm{Id}_F(k) = 1) = \Pr_{\omega,e}(V(\omega, e) = 1) = \sum_\omega \Pr(\omega) \cdot \delta_\omega.$$

Spomnimo se, da algoritem A uspešno izračuna rešitev problema diskretnega logaritma, če napadalec F uspe dvakrat pri različnih izzivih e_1 in e_2. Potem lahko izrazimo

$$\begin{aligned} \Pr(A \text{ uspe}) &= \Pr_{\omega,e_1,e_2}(V(\omega, e_1) = 1 \wedge V(\omega, e_2) = 1 \wedge e_1 \neq e_2) \\ &\geq \Pr_{\omega,e_1,e_2}(V(\omega, e_1) = 1 \wedge V(\omega, e_2) = 1) - \Pr(e_1 = e_2) \\ &= \sum_\omega \Pr(\omega) \cdot \delta_\omega^2 - \frac{1}{q} \\ &\geq (\sum_\omega \Pr(\omega) \cdot \delta_\omega)^2 - \frac{1}{q} \\ &= \Pr(\mathrm{Id}_F(k) = 1)^2 - \frac{1}{q}, \end{aligned}$$

kjer smo pri prehodu na predzadnjo vrstico uporabili Jensenovo neenakost [9]. Ker pa je uspeh algoritma A ekvivalenten rešitvi problema diskretnega logaritma v polinomskem času, problem pa po predpostavki obravnavamo v grupi, kjer je problem diskretnega logaritma težek, lahko zaključimo, da je verjetnost uspeha algoritma A zanemarljiva. Ker je velikost grupe q odvisna od varnostnega parametra k, je tudi člen $1/q$ zanemarljiv. Iz tega lahko zaključimo tudi, da je verjetnost uspeha napadalca F $\Pr(\mathrm{Id}_F(k) = 1)$ zanemarljiva, kar pomeni, da je Schnorrov identifikacijski protokol varen. □

Kot smo videli v primeru 2.20, lahko Fiat-Shamirjevo hevristiko uporabimo, da iz Schnorrove identifikacijske sheme pridobimo neinteraktivno verzijo. Če torej lahko pokažemo, da je ta transformacija varna, smo dokazali varnost Schnorrovega podpisa.

Izrek 3.5 (Varnost Fiat-Shamirjeve hevristike [10]). *Naj bo $\mathcal{S}'$ Schnorrova identifikacijska shema in $\mathcal{S}$ Schnorrova podpisna shema, pridobljena iz $\mathcal{S}'$ z uporabo Fiat-Shamirjeve hevristike. Če je identifikacijska shema $\mathcal{S}'$ varna, potem je tudi podpisna shema $\mathcal{S}$ varna, če varnost obravnavamo v modelu slučajnega oraklja.*

Dokaz. Naj bo napadalec F na podpisno shemo $\mathcal{S}$ naključnostni algoritem, ki teče v polinomskem času. Naj bo število napadalčevih klicev slučajnega oraklja polinomsko omejeno v varnostnem parametru k. Zgornjo mejo označimo s q.

Brez škode za splošnost predpostavimo, da bo napadalec F vsak klic slučajnega oraklja na nekem vhodu opravil največ enkrat. Predpostavimo tudi, da če napadalec F uspešno ponaredi podpis (X, y) sporočila m, potem je gotovo poklical oraklja H s podatki $\text{enc}(X)||m$.

Na podlagi napadalca F na podpisno shemo $\mathcal{S}$ želimo konstruirati napadalca F' na identifikacijsko shemo $\mathcal{S}'$, ki ima dostop do transkripcijskega oraklja T_{sk} in javnega ključa I. Napadalec F' deluje v naslednjih korakih:

1. Izbere si enakomerno naključno število $j \in \{1, 2, \ldots, q\}$.

2. Požene napadalca F in zanj simulira delovanje podpisne sheme $\mathcal{S}$. Ko napadalec F opravi i-ti klic oraklja H z vhodom $\text{enc}(X_i)||m_i$, napadalec F' odgovori na podlagi vrednosti i in j:

   - Če je $i = j$, napadalec F' uporabi X_i kot zavezo v identifikacijski shemi, v odgovor dobi izziv e_i in ga pošlje napadalcu F kot odgovor na klic oraklja H.
   - V nasprotnem primeru is izbere enakomerno naključno število $e_i \in \mathbb{Z}_q$ in ga pošlje napadalcu F kot odgovor na klic oraklja H.

   Ko napadalec F zahteva podpis sporočila m, napadalec F' pokliče transkripcijskega oraklja T_{sk}, ki mu vrne transkript (X, e, y). Par (X, y) vrne kot podpis napadalcu F.

3. Če napadalec F uspe ponarediti podpis (X, y) sporočila m, napadalec F' preveri, ali velja $(X, m) = (X_j, m_j)$. Če velja, napadalec F' vrne y kot odgovor.

V koraku 2 je simulacija podpisne sheme uspešna, saj je izziv e v vsakem primeru izbran enakomerno naključno (le iz drugega vira), prav tako je par (X, y) iz transkripcijskega oraklja izračunan na enak način kot veljaven podpis. Napadalec F' torej ne loči med simulacijo in pravim podpisom. Problem nastane le, če je napadalec F' že definiral orakljev odgovor s podatkoma X in m, ko dobi transkript (X, e', y) od transkripcijskega oraklja T_{sk}. Če v tem primeru e ni enak e', simulacija ne uspe. Verjetnost, da se to zgodi, je zanemarljiva, saj je e izbran enakomerno naključno iz $\mathbb{Z}_q$ in je q polinomsko omejen v varnostnem parametru k.

Potrdimo še, da je rezultat napada na identifikacijsko shemo smiseln. Če v zadnjem koraku velja $(X, m) = (X_j, m_j)$ in napadalec F uspe ponarediti podpis

(X, y), potem napadalec F' res lahko pošlje zavezo X_j, v odgovor pa dobi izziv e_j, na podlagi katerega napadalec na podpis F izračuna odgovor y. Ta odgovor je zaradi ujemanja izziva tudi veljaven odgovor v zadnjem koraku identifikacijske sheme, kar pomeni, da je napadalec F' uspešno napadel.

Zadnji korak je, da pogledamo verjetnost uspeha napadalca F'. Naj δ_k označuje verjetnost, da pride do neujemanja pri odgovorih oraklja (zgoraj smo videli, da je zanemarljiva). Ker je j izbran enakomerno naključno, je verjetnost, da velja $i = j$, enaka $1/q$. Verjetnost uspešnega napada lahko izrazimo kot

$$\Pr(F' \text{ uspe}) = \frac{1}{q} \cdot (\Pr(F \text{ uspe}) - \delta_k),$$

oz.

$$\Pr(F \text{ uspe}) = q \cdot \Pr(F' \text{ uspe}) + \delta_k.$$

Ker je po predpostavki identifikacijska shema varna, je verjetnost uspeha napadalca F' zanemarljiva. Prav tako je tudi verjetnost δ_k zanemarljiva, q pa je polinomska zgornja meja, kar pomeni, da je tudi verjetnost uspeha napadalca F zanemarljiva. To pomeni, da je Schnorrov podpis varen, če je identifikacijska shema varna. □

Izrek 3.6 (Varnost Schnorrovega podpisa)**.** *Naj bo G ciklična grupa, v kateri je problem diskretnega logaritma težek. Naj bo H slučajni orakelj, ki je dostopen vsem deležnikom. Potem je Schnorrov podpis v grupi G varen.*

Dokaz. Po izreku 3.4 je Schnorrov identifikacijski protokol v grupi G varen. Prav tako je po izreku 3.5 varna tudi pretvorba iz Schnorrove identifikacijske sheme v Schnorrov podpis, če upoštevamo model slučajnega oraklja. Zaključimo torej, da je Schnorrov podpis varen. □

3.2 Primer: Schnorrov podpis v $\mathbb{Z}_p^*$

Za ilustrativni primer si poglejmo, kako je bil Schnorrov podpis prvotno opisan [3]. Predstavljeni podpis je poseben primer podpisa, opisanega zgoraj, kjer je grupa G multiplikativna grupa naravnih števil modulo praštevila p. Za namene tega dela si ga je posebej koristno pogledati, saj bo enaka grupa uporabljena tudi pri njegovi večstranski različici v poglavju 5.

- **Parametri**: Najprej je potrebno generirati par praštevil p in q, tako da q deli $p - 1$. Praštevilo p definira grupo $\mathbb{Z}_p^*$, ki je multiplikativna grupa celih števil modulo p. V tej grupi je potem potrebno izbrati element g, ki je reda q. Za varnost je potrebno, da ima p vsaj 2048 bitov, q pa vsaj 224 bitov.

  Kot v splošni verziji Schnorrovega podpisa, si morata podpisnik in preverjevalec izbrati še varno kriptografsko zgoščevalno funkcijo $H : \{0,1\}^* \to \mathbb{Z}_q$.

- **Ključ**: Za ustvarjanje para ključev je potreben izbor naključnega števila $s \in \mathbb{Z}_q$ in izračun

  $$I = g^s \bmod p.$$

  Ta izračun nam da par ključev

  $$\text{pk} = (p, q, g, I),$$
  $$\text{sk} = s.$$

- **Podpis**: Za podpis enega sporočila mora podpisnik generirati naključno število $r \in \mathbb{Z}_q$ in izračunati *zavezo*

$$X = g^r \bmod p.$$

Potem z uporabo funkcije H izračunamo *izziv*

$$e = H(\text{enc}(X)||m),$$

Za konec je potrebno izračunati še

$$y = es + r \bmod q,$$

podpis sporočila m pa je potem par (X, y) oz.

$$S(s, m) = (X, y).$$

Postopek je torej enak splošni verziji Schnorrovega podpisa, le da je v tem primeru uporabljeno modularno množenje.

- **Preverjanje**: Za preverjanje veljavnosti podpisa (X', y') sporočila m, je potrebno izračunati

$$e' = H(\text{enc}(X')||m)$$

in preveriti, če velja

$$g^{y'} \stackrel{?}{\equiv} X' \cdot I^{e'} \pmod p. \tag{3.2}$$

Z nekaj modularne aritmetike lahko pokažemo, da enačba (3.2) preverja veljavnost Schnorrovega podpisa.

Po trditvi 2.3 lahko levo stran enačbe za preverjanje Schnorrovega podpisa (3.2) prepišemo

$$\begin{aligned} g^{y'} \bmod p = g^{es+r \bmod q} \bmod p = \\ = g^{es+r} \bmod p. \end{aligned}$$

Desno stran enačbe (3.2) pa po trditvi 2.4 lahko prepišemo

$$\begin{aligned} X' \cdot I^{e'} \bmod p = g^r \bmod p \cdot (g^s \bmod p)^e \bmod p = \\ \stackrel{(2.2)}{=} (g^r \bmod p) \cdot (g^{es} \bmod p) \bmod p = \\ \stackrel{(2.1)}{=} g^{es+r} \bmod p, \end{aligned}$$

kjer smo pri prehodu v drugo vrstico uporabili lastnost (2.2), pri prehodu v tretjo pa lastnost (2.1). Ker se obe strani ujemata za veljavne podpisne vrednosti, ta enačba res preverja Schnorrov podpis.

4 Pregled skupinskih podpisov

Ko pridemo do podpisovanja skupin, si lahko zamislimo več različnih rešitev. Micali [20] definira dve lastnosti oz. spektra, ki jim lahko zadošča podpis skupine:

- **Prilagodljivost** (angl. *flexibility*): Popolnoma prilagodljiv podpis skupine je takšen, ki ga lahko proizvede katerakoli podskupina originalne skupine podpisnikov. Ko je podpis preverjen, se mora tisti, ki ga je preveril, odločiti, če je ustrezen del skupine podal svoj podpis. Popolnoma neprilagodljiv podpis bi bil takšen, ki ga lahko ustvari le celotna skupina ali pa katerkoli član v imenu celotne skupine.

- **Odgovornost** (angl. *accountability*): Če lahko iz podpisa ugotovimo, kateri člani so sodelovali pri ustvarjanju, nam podpis omogoča odgovornost. Ta lastnost je lahko zaželena, če se želimo prepričati, ali je ustrezen del skupine sodeloval pri podpisu (npr. ali je pri podpisovanju sodeloval generalni direktor podjetja). V drugih primerih pa si želimo anonimnost posameznih članov (npr. če bi generiranje podpisa predstavljalo nekakšno tajno glasovanje, bi želeli vedeti samo, koliko članov je sodelovalo).

V nadaljevanju bomo skupino potencialnih podpisnikov (torej podpisnikov, ki imajo možnost sodelovati pri podpisovanju) označili s $P = P_1, \ldots, P_L$, kjer ima skupina L članov. Dejanski podpis pa bo ustvaril samo del skupine $S \subseteq P$.

4.1 Skupinski podpisi

Skupinski podpis (angl. *group signature*) v imenu celotne skupine P ustvari en anonimen član. To torej pomeni, da je podpis popolnoma neprilagodljiv, saj ni možno prisiliti skupine, da bi podpis ustvaril več kot en član. Prav tako v splošnem noben član, niti tisti, ki preverja podpis, ne more ugotoviti, kdo je podpis ustvaril. Da skupinski podpisi omogočijo vsaj delno odgovornost, skupina določi *vodjo skupine*, ki ima možnost razkriti identiteto podpisnika, če pride do težav. V tem primeru seveda vodja predstavlja atraktivno tarčo za napad. Skupinske podpise sta si zamislila Chaum in van Heyst [5].

Primer 4.1. Skupinski podpisi so zelo uporabni v primerih, ko moramo vedeti samo, da neka oseba pripada skupini. Primer je recimo dostop do varovanih območij, kjer je neprimerno, da bi natančno sledili vsem posameznikom, vseeno pa mora biti dostop omejen samo zaposlenim. ♢

4.2 Pragovni podpisi

Če želimo zagotoviti, da se s podpisom strinja zadosten delež skupine, lahko uporabimo **pragovni podpis** (angl. *threshold signature*). Ta nam omogoča določeno mero prilagodljivosti, saj lahko katerikoli zadosten delež skupine ustvari podpis. Še vedno je nemogoče upoštevati morebitno hierarhično strukturo skupine. Po definiciji pragovnih podpisov, ti ne omogočajo odgovornosti, nekateri celo zagotavljajo popolno anonimnost podpisnikov.

Večina pragovnih podpisov temelji na interpolaciji polinoma $(l-1)$-stopnje z l točkami. Podpis je potem ustvarjen s pomočjo vrednosti polinoma v neki točki. Po interpolaciji se informacija o tem, točno katere točke smo uporabili, izgubi. Take podpise imenujemo tudi *l-od-L sheme*.

Primer 4.2. Denimo, da ima banka sef, v katerem hrani vse pomembne dokumente in denar. Zaradi izjemne pomembnosti sefa si ne želimo, da bi ga lahko odprla katerakoli posamezna oseba. Če osebje banke uporabi pragovni podpis, lahko zagotovi, da je pri odpiranju sefa vedno prisotnih več ljudi, vseeno katerih. Vsak zaposleni dobi točko interpolacije, ko se jih zbere dovolj, lahko skupaj ugotovijo vrednost polinoma v vnaprej določeni točki in odklenejo sef. ♢

4.3 Večstranski podpisi

Za nekatere uporabe podpisov, bi si od njih želeli podobne lastnosti, kot jih ima večstranski ročen podpis. Pri njem lahko hitro preberemo podpisnike, torej imamo popolno prilagodljivost. Vidimo lahko seznam podpisnikov, torej lahko presodimo, če so med njimi tisti, ki smo jih želeli. Prav tako podpisniki nosijo popolno odgovornost, saj na papirju piše njihovo ime.

Podoben učinek bi z digitalnimi podpisi lahko dosegli, če bi namesto enega podpisa skupine, od članov zbrali individualne podpise in jih nanizali v seznam. Dobili bi torej digitalni podpis skupine, ki ponuja popolno prilagodljivost in odgovornost. Težava je samo, da je dolžina podpisa (in s tem čas preverjanja) proporcionalna številu podpisnikov. **Večstranski podpisi** (angl.*multisignatures*) ohranijo lastnosti seznama podpisov, rezultat sheme je pa en sam podpis, ki je enako dolg ne glede na število podpisnikov, prav tako je od števila neodvisen čas preverjanja. Tega s seznamom podpisov ni mogoče doseči, saj tako dolžina podpisa, kot čas preverjanja podpisa rasteta linearno s številom podpisnikov (vsak doda en podpis seznamu, ki ga je potrebno preveriti). Večstranski podpisi so torej odlična posplošitev ročnih podpisov skupin, ki vseeno ohrani učinkovito preverjanje.

Še ena pomembna lastnost večstranskih podpisov je, da nekatere sheme vračajo podpise, ki so kar se preverjanja tiče, popolnoma zamenljivi z običajnimi digitalnimi podpisi. To omogoča enostavno nadgradnjo obstoječih sistemov, ki že uporabljajo digitalne podpise.

Primer 4.3. Recimo, da imamo nek organ, ki izdaja certifikate avtentičnosti uporabnikov (npr. potrjuje avtentičnost javnih ključev). Za večjo robustnost in varnost, je lahko ta organ razporejen na več strežnikov. Tako preprečimo razpad sistema v primeru izpada enega strežnika. Zato je torej tudi pomembno, da certifikacijo uporabnika potrdi nekaj strežnikov, ne pa nujno vsi. Tu lahko torej neka podskupina vseh strežnikov organa skupaj izda en večstranski podpis, ki potrjuje avtentičnost uporabnika. ♢

4.4 Agregirani podpisi

Agregirani podpisi imajo zelo podobne lastnosti kot večstranski. Poleg standardnih algoritmov pri digitalnih podpisih za ustvarjanje parametrov $\mathcal{P}$, ustvarjanje ključev

$\mathcal{G}$, podpisovanje $\mathcal{S}$ in preverjanje $\mathcal{V}$, imajo agregirani podpisi (angl. *aggregate signatures*) še dodaten algoritem za združevanje podpisov $\mathcal{C}$. Ta algoritem prejme seznam trojic javnih ključev, sporočil in podpisov, vrne pa en sam podpis. Od večstranskih podpisov se razlikuje po tem, da sporočila niso nujno vsa enaka [24].

Primer 4.4. Recimo, da avtentičnost ključev posameznih podpisnikov preverja niz centrov za certificiranje podpisov, kjer vsak naslednji center jamči za avtentičnost centra pod njem. Prvi center zajamči avtentičnost našega ključa, naslednji zajamči avtentičnost centra in tako naprej. Ko bi želel nekdo prejeti naš certificiran javni ključ, bi moral preveriti veljavnost podpisov vseh centrov, s pomočjo agregiranih podpisov bi to lahko storil z enim samim preverjanjem. ♢

5 Večstranski Schnorrov podpis

Micali et al. [20] so prvi definirali formalni model za večstranske podpise in podali formalni dokaz varnosti za podpisno shemo v tem modelu. Zamislili so si formalni model za večstranske podpise in ga poimenovali **večstranski podpis podskupine z odgovornostjo** (angl. *Accountable-Subgroup Multisignature (ASM)*). V tem razdelku predstavimo njihov model, primer večstranskega podpisa, ki temelji na Schnorrovem podpisu, in argumentiramo varnost podpisa oz. modela na sploh.

5.1 Večstranski podpis podskupine z odgovornostjo

Kljub daljšemu imenu, večstranski podpis podskupine z odgovornostjo le predstavlja formalni model za večstranske pospise, predstavljene v razdelku 4.3. Ideja oz. cilj večstranskega podpisa je torej, da lahko katerakoli podskupina podpisnikov S, neke skupine P, brez potrebe po **centru zaupanja** (angl. *trusted third party (TTP)*) ustvari podpis. Generiranje ključev je torej popolnoma v domeni skupine P. Podpis, ki ga ustvari S predstavlja splošno preverljiv dokaz strukture S in dejstva, da vsak član skupine stoji za (torej podpisuje) sporočilom m. V tem razdelku predstavimo osnovno idejo ASM.

Definicija 5.1 (Večstranski podpis podskupine z odgovornostjo)**.** Skupina P je sestavljena iz L podpisnikov, torej

$$P = P_1, P_2, \ldots, P_L.$$

Podpisnik predstavlja naključnostni polinomski algoritem. Vsak podpisnik pozna svojo identifikacijsko številko (eno od števil $1, \ldots, L$) in pa *varnostni parameter* k, ki je enak za vse podpisnike.

Kot vsi digitalni podpisi iz definicije 2.11, ima tudi ta štiri glavne komponente:

- $\mathcal{P}$ je algoritem za ustvarjanje parametrov. Pognan je samo enkrat, na začetku sodelovanja. Vrne javne parametre sheme Par. Velikost parametrov je odvisna od varnostnega parametra k. Velja torej

$$\mathcal{P}(k) = \text{Par}.$$

- $\mathcal{G}$ je algoritem za ustvarjanje ključev. Za neko skupino podpisnikov P je pognan samo enkrat, na začetku sodelovanja. Vsak podpisnik P_i dobi kot vhod seznam vseh podpisnikov v P in parametre podpisne sheme Par ter požene algoritem $\mathcal{G}$, ki vrne par ključev $(\mathrm{pk}_i, \mathrm{sk}_i)$. Zapišemo lahko torej

  $$\mathcal{G}_i(L, \mathrm{Par}) = (\mathrm{pk}_i, \mathrm{sk}_i),$$

  kjer smo brez škode za splošnost predpostavili, da je vsak podpisnik enolično predstavljen s svojim indeksom $i \in \{1, 2, \ldots, L\}$ v skupini P.

- $\mathcal{S}$ je protokol za podpisovanje. Pognan je vsakič, ko neka podskupina S želi ustvariti podpis. Vsak podpisnik požene algoritem $\mathcal{S}$ s seznamom vseh podpisnikov S, njihovimi javnimi ključi pk_j, kjer j teče po identifikacijskih številkah vseh članov S, sporočilom m in lastnim zasebnim ključem sk_i kot vhodnimi podatki. Algoritem $\mathcal{S}$ je porazdeljen protokol, pri izvedbi morajo sodelovati vsi člani S, ki med seboj komunicirajo. Po uspešni izvedbi lahko en od članov objavi podpis σ.

- $\mathcal{V}$ je algoritem za preverjanje veljavnosti podpisa. Požene ga tisti, ki želi preveriti veljavnost večstranskega podpisa. Ni nujno, da je to eden izmed podpisnikov iz P. Kot vhod algoritem dobi seznam podpisnikov S, pripadajoče javne ključe, sporočilo m in morebiten podpis σ. Algoritem potem vrne

  $$\mathcal{V}((id_1, \ldots, id_l), (\mathrm{pk}_{id_1}, \ldots, \mathrm{pk}_{id_l}), m, \sigma) = \begin{cases} veljaven, & \text{podpis veljaven,} \\ neveljaven, & \text{sicer.} \end{cases}$$

5.1.1 Robustnost, varnost in napadalec

Večstranski podpis iz definicije 5.1 ni *robusten*. To pomeni, da v primeru izpada enega od podpisnikov P_i, ki je del podskupine S, ta ne more ustvariti večstranskega podpisa. Podpis še vedno lahko ustvari $S \setminus \{P_i\}$. Ker pri končni shemi ne iščemo robustnosti, lahko definiramo zelo močnega napadalca, ki ima velik vpliv na celoten sistem.

Definicija 5.2 (Napadalec pri ASM)**.** Napadalca v modelu večstranskih podpisov podskupine z odgovornostjo bomo označili z F. Ima naslednje zmožnosti:

- Ima popoln nadzor nad vsemi komunikacijskimi kanali med člani skupine P. Lahko bere, spreminja in preprečuje dostavo vseh sporočil. Prav tako lahko v imenu kateregakoli podpisnika pošlje sporočilo.

- Kadarkoli lahko *pokvari* kateregakoli podpisnika. Ko je podpisnik pokvarjen, napadalec izve njegovo celotno notranje stanje, vključno z vsemi skrivnostmi.

- Nadzira lahko vhod algoritma za ustvarjanje ključev $\mathcal{G}$ za vse podpisnike. Vsakemu lahko poda drugačno skupino P.

- Od kateregakoli nepokvarjenega podpisnika lahko kadarkoli zahteva podpis nekega sporočila skupaj s podskupino, ki jo določi napadalec. To sposobnost imenujemo *napad z izbranim sporočilom in podskupino.*

Zaradi obširnih zmožnosti napadalca, ta lahko vedno prepreči podpis sporočila. Naš cilj, kar se tiče varnosti, je, da preprečimo eksistencialno ponarejanje podpisa. Želimo torej, da napadalec ni zmožen ponarediti podpisa za katerokoli sporočilo v imenu katerekoli podskupine.

Definicija 5.3 (Varnost pri ASM)**.** Naj bo k varnostni parameter (ki si ga delijo vsi podpisniki). Naj bo $c > 0$ poljubna konstanta. Naj bo F napadalec, ki ustreza definiciji 5.2, torej lahko nadzoruje in spreminja dejanja podpisnikov skupine S. Naj bo njegova računska moč omejena s polinomskim časom v parametru k. Naj bo p verjetnost, da napadalec F vrne trojico (σ, m, S), za katero velja:

- σ je veljaven večstranski podpis sporočila m s strani skupine S.
- Obstaja nepokvarjen podpisnik P_j iz skupine S, od katerega napadalec F ni zahteval podpisa sporočila m s strani skupine S.

Shemi za večstranske podpise podskupine z odgovornostjo bomo rekli, da je **varna**, če je verjetnost p zanemarljiva, oz. če za vsakega polinomsko omejenega napadalca F obstaja zanemarljiva funkcija ε, da velja

$$p < \varepsilon(k).$$

Kot je standardno pri varnostni obravnavi digitalnih podpisov, tudi tu predpostavimo, da preverjevalec kljub močnemu napadalcu lahko vedno pridobi prave javne ključe podpisnikov iz S. To pomeni, da v času preverjanja pozna prave identitete podpisnikov, tudi če napadalec lahko doseže, da podpisniki v času podpisovanja ne vedo zares, s kom podpisujejo (ne morejo zaupati, da res sodelujejo s člani podskupine S, saj napadalec nadzira komunikacijo).

5.1.2 Slučajni orakelj pri ASM

Varnostna obravnava shem za večstranske podpise podskupine z odgovornostjo zahteva model slučajnega oraklja. Zato predpostavimo, da je k_2 še en varnostni parameter. Vsi člani skupine P in napadalec imajo dostop do slučajnega oraklja $H : \{0,1\}^* \to \{0,1\}^{k_2}$, ki je naključno izbrana funkcija med vsemi funkcijami, ki slikajo med $\{0,1\}^*$ in $\{0,1\}^{k_2}$.

5.2 Konstrukcija večstranskega Schnorrovega podpisa

V nadaljevanju bomo konstruirali večstransko verzijo Schnorrovega podpisa, predstavljenega v poglavju 3. Končna shema spada med večstranske podpise podskupin z odgovornostjo. Ideja konstrukcije je, da začnemo z naivno verzijo, nato pa rešimo njene probleme, kar nas privede do formalne definicije v razdelku 5.3 in dokaza varnosti v razdelku 5.4.

5.2.1 Naivna verzija

Vsi podpisniki v skupini P se strinjajo o izboru končne grupe G reda q, ki jo generira generator g. Vsak podpisnik P_i si neodvisno in naključno izbere skrivno število $s_i \in \mathbb{Z}_q$ in izračuna

$$I_i = g^{s_i}.$$

Tako vsak podpisnik ustvari svoj par ključev

$$\begin{aligned} \mathrm{pk}_i &= (q, g, I_i), \\ \mathrm{sk}_i &= s_i. \end{aligned}$$

Poljubna podskupina $S = \{P_{id_1}, \ldots, P_{id_l}\}$ skupine P podpiše sporočilo m s tremi krogi komunikacije:

1. Vsak podpisnik P_i iz podskupine S si izbere naključen element $r_i \in \mathbb{Z}_q$ in izračuna zavezo

   $$X_i = g^{r_i}.$$

   Podpisniki potem pošljejo svoje zaveze izbranemu podpisniku D.

2. D izračuna *skupno zavezo*

   $$\tilde{X} = X_{id_1} \cdot X_{id_2} \cdot \cdots \cdot X_{id_l}.$$

   in jo pošlje vsem podpisnikom.

3. Vsak podpisnik s pomočjo slučajnega oraklja izračuna izziv

   $$e = H(\mathrm{enc}(\tilde{X})||m||S)$$

   in svoje *individualne podpise*

   $$y_{id_i} = es_{id_i} + r_{id_i} \bmod q.$$

   Individualni podpisi so potem spet poslani podpisniku D, ta pa sedaj lahko izračuna

   $$\tilde{y} = (y_{id_1} + y_{id_2} + \cdots + y_{id_l}) \bmod q$$

   in vrne končen večstranski podpis

   $$\sigma = (\tilde{X}, \tilde{y}).$$

Preverjanje veljavnosti podpisa je podobno kot pri navadnemu Schnorrovemu podpisu. Da preverimo podpis $(\tilde{X}', \tilde{y}')$ sporočila m najprej izračunamo

$$e' = H(\mathrm{enc}(\tilde{X}')||m||S)$$

in preverimo, če velja

$$g^{\tilde{y}'} \stackrel{?}{=} \tilde{X}' \cdot \left(\prod_{P_i \in S} I_i \right)^{e'}. \tag{5.1}$$

5.2.2 Generiranje parametrov in Predpostavka DL

Micali et al. [20] so izpostavili in rešili več problemov z zgornjo naivno idejo. Prvi problem se pojavi pri generiranju skupnih parametrov javnega ključa. Ker si podpisniki delijo samo varnostni parameter k in pa slučajnega oraklja H, morajo za parametre uporabiti H na vnaprej dogovorjen način. Napadalec bo tako poznal točen postopek ustvarjanja skupnih parametrov, kar mu potencialno da prednost. Izkaže se, da lahko s primerno uporabo H poskrbimo, da to ne predstavlja nevarnosti.

Ker pa je pri uporabi kriptografije standardno, da se uporabi znane, vnaprej definirane grupe, se s problemom generiranja parametrov ne bomo ukvarjali. Enostavno predpostavimo, da imajo podpisniki dostop do skupnih, varnih parametrov.

Ker podpis temelji na Schnorrovem, varnost temelji na težavnosti diskretnega logaritma. Predpostavka o varnosti je tako standardna predpostavka o težavnosti diskretnega logaritma, predstavljena v definiciji 2.13. Pri obravnavi sheme torej predpostavimo, da je G grupa, v kateri je izračun diskretnega logaritma težaven.

Primer 5.4 (Generianje parametrov v grupi $\mathbb{Z}_p^*$)**.** V najosnovnejši obliki podpisa uporabimo grupo $\mathbb{Z}_p^*$, kjer je p praštevilo. Za delovanje potrebujemo še praštevilo q, ki deli $p-1$. Za generiranje lahko uporabimo standarden algoritem za pridobivanje praštevil 1.

Algoritem 1 Algoritem $GenPrimes(k)$ za generiranje praštevil.

```
q ← naključno izbrano k-bitno število
p ← 2q + 1
while q ni praštevilo in p ni praštevilo do
    q ← naključno izbrano k-bitno število
    p ← 2q + 1
end while
return p, q
```

Kot primer predstavimo še točno predpostavko o težavnosti diskretnega logaritma v grupi $\mathbb{Z}_p^*$.

Definicija 5.5 (Predpostavka o težavnosti diskretnega logaritma pri ASM v grupi $\mathbb{Z}_p^*$)**.** Naj bo A poljuben naključnostni algoritem, ki teče v polinomskem času in svoje parametre sprejme

- praštevili p in q, da velja $p = 2q + 1$ in q je dolg k-bitov,
- naključni element $g \in \mathbb{Z}_p^*$ reda q,
- naključni element I iz podgrupe $\mathbb{Z}_p^*$, ki jo generira g.

Naj p_k^A označuje verjetnost, da A vrne $s \in \mathbb{Z}_q$, tako da velja

$$I \equiv g^s \pmod{p}.$$

Potem za vsako konstanto $c > 0$ in za vsak dovolj velik k velja, da je verjetnost p_k^A zanemarljiva, oz. da obstaja zanemarljiva funkcija ε, za katero velja

$$p_k^A < \varepsilon(k).$$

♢

5.2.3 Napad na generiranje ključev

Naslednja težava nastopi v obliki napada v fazi generiranja ključev. Ker napadalec nadzoruje vso komunikacijo, lahko v tej fazi enostavno počaka, da vsi podpisniki ustvarijo svoje ključe. Na tej točki napadalec stopi v vlogo (torej pokvari) zadnjega podpisnika P_L in generira nov par ključev. Če si torej izbere naključen $s \in \mathbb{Z}_q$, lahko izračuna svoj javni ključ kot

$$I_L = \left(\prod_{i=1}^{L-1} I_i\right)^{-1} \cdot g^s.$$

To mu omogoča, da se podpisuje v imenu celotne podskupine S, saj se bodo pri množenju v enačbi (5.1) za preverjanje tako izničil vsi ostali javni ključi:

$$\begin{aligned} g^{\tilde{y}} &\stackrel{?}{=} \tilde{X} \cdot \left(\prod_{i=1}^{L} I_i\right)^e. \\ &\stackrel{?}{=} \tilde{X} \cdot \left(\left(\prod_{i=1}^{L-1} I_i\right) \cdot I_L\right)^e. \\ &\stackrel{?}{=} \tilde{X} \cdot \left(\left(\prod_{i=1}^{L-1} I_i\right) \cdot \left(\prod_{i=1}^{L-1} I_i\right)^{-1} \cdot g^s\right)^e. \\ &\stackrel{?}{=} \tilde{X} \cdot (g^s)^e. \end{aligned}$$

To pomeni, da na veljavnost podpisa vpliva samo napadalec, torej lahko ponaredi katerikoli podpis.

Za rešitev tega problema lahko uporabimo na dokaze o znanju brez razkritja znanja. Torej, da preprečimo napad na generiranje ključev, od vsakega podpisnika P_i zahtevamo, da poleg svojega javnega ključa I_i objavi tudi dokaz o znanju brez razkritja znanja za svoj zasebni ključ glede na javni ključ. Objaviti mora torej dokaz, da pozna diskretni logaritem I_i z bazo g. Ta dokaz je neinteraktiven, saj obravnavmo model slučajnega oraklja in lahko uporabimo Fiat-Shamirjevo hevristiko 2.6.3.

Ker tovrstni dokazi brez preverjevalca nimajo smisla, se je na tej točki vredno vprašati, kdo bo preverjal veljavnost dokazov. Izkaže se, da je najbolj enostavno, da se dokaze doda v posamezne javne ključe. Tako pade breme preverjanja na tistega, ki bo preverjal podpis. Problematično je dejstvo, da taka sprememba podaljša javne ključe in privede do izgube učinkovitosti. Preverjanje veljavnosti podpisa bo sedaj poleg dveh potenciranj, ki sta v enačbi (3.1) potrebni za preverjanje standarnega Schnorrovega podpisa, potrebovalo še dodatnih $2|S|$ potenciranj, saj je potrebno preveriti vse dokaze javnih ključev, kar ustreza preverjanju $|S|$ Schnorrovih podpisov. Posamezni preverjevalec podpisov lahko sicer dodatno delo opravi le enkrat za vsako skupino, če natančno beleži rezultate preverjanja dokazov.

5.2.4 Učinkovitost podpisovanja: Merklova drevesa

Kot bomo videli, je za dokaz varnosti potrebno, da lahko algoritem $\mathcal{A}$, ki bo simuliral proces podpisovanja napadalcu, v polinomskem času za vsakega pokvarjenega

podpisnika P_j pridobi diskretni logaritem I_j iz predloženega dokaza o znanju brez razkritja znanja. Izkaže pa se, da če bo P_j dokaz izračunal s q klici slučajnega oraklja, potem bo simulator uspešno pridobil diskretni logaritem z verjetnostjo največ $1/q$. Še več, če je b pokvarjenih podpisnikov, bo algoritem $\mathcal{A}$ uspešen z verjetnostjo največ $1/q^b$. Z drugimi besedami, če želimo, da algoritem $\mathcal{A}$ teče v polinomskem času, je podpisnikov lahko največ logaritemsko mnogo v številu klicev oraklja q. Ker pa je teh klicev lahko največ polinomsko mnogo v varnostnem parametru k, je torej število podpisnikov lahko največ logaritemsko v k.

Ta problem lahko rešimo tako, da vsak podpisnik P_i najprej izračuna svoj par ključev (s_i, I_i) in na podlagi naključno izbrane vrednosti r_i tudi zavezo $X_i = g^{r_i} \pmod p$, ki bi jo uporabil pri dokazu znanja brez razkritja znanja. Potem si vsi podpisniki izmenjajo javne ključe I_i in zaveze X_i. Vsak podpisnik lahko sedaj izračuna *skupni izziv*

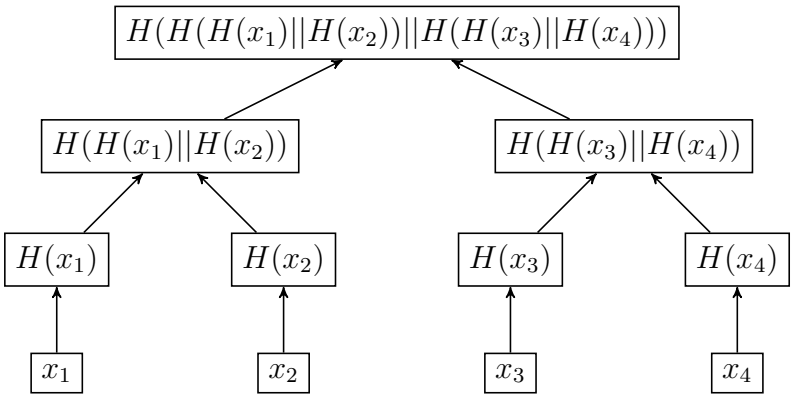
$$e = H(\text{enc}(X_1)||\text{enc}(I_1)||\text{enc}(X_2)||\text{enc}(I_2)||\ldots||\text{enc}(X_L)||\text{enc}(I_L)).$$

Podpisniki nato dokončajo svoj dokaz na podlagi zgostitve e in svojega para ključev.

Kot bomo videli, bo to pomenilo, da za generiranje dokazov b pokvarjenih podpisnikov sedaj rabi le bq klicev oraklja, torej je polinomski algoritem $\mathcal{A}$ uspešen z verjetnostjo $1/bq$. To omogoča, da je podpisnikov več kot logaritemsko mnogo. Pojavi pa se drugačen problem z učinkovitostjo, saj mora sedaj vsak podpisnik v svojem javnem ključu hraniti še Schnorrov podpis sporočila e in pa tudi vse javne ključe sopodpisnikov in njihove zaveze. Brez teh podatkov namreč dokaz znanja (Schnorrov podpis) ni preverljiv. To pomeni, da je za katerokoli podskupino S velikost ključa proporcionalna velikosti celotne skupine P. Poleg tega mora sedaj preverjevalec preveriti, da se vsi seznami javnih ključev ujemajo. V primerjavi z osnovno naivno idejo, kjer je preverjevalec potreboval le $|S|$ navadnih Schnorrovih javnih ključev, je v tej izvedbi velikost javnega ključa nedopustno velika.

Da nazaj pridobimo učinkovit podpis, lahko uporabimo slučajnega oraklja in kriptografsko orodje, imenovano **Merklovo drevo** (angl. *Merkle tree*). V osnovi je to *binarno drevo*, torej drevo, kjer ima vsako vozlišče največ dva otroka. Ideja je, da v liste shranimo zgostitve katerekolih podatkov, pridobljene s pomočjo izbranega slučajnega oraklja ali zgoščevalne funkcije. Drevo potem gradimo navzgor tako, da v vsako vozlišče shranimo zgostitev stika levega in desnega otroka. Tako nadaljujemo do korena. Pomembno je, da je za pridobivanje zgostitev uporabljena zgoščevalna funkcija, za katero je računsko neizvedljivo najti trk, ali pa slučajni orakelj. Osnovna struktura je prikazana na sliki 3.

Pomembno je opaziti, da če zgoščevalna funkcija ustvarja zgostitve, dolge k bitov, bodo toliko dolgi vsi podatki, ki jih hranijo vsa vozlišča drevesa. Za Merklova drevesa je ključno, da je v primeru varne zgoščevalne funkcije računsko zelo težko spremeniti katerokoli vrednost v drevesu, brez da bi se spremenila vrednost v korenu. Enako velja za spremembe vhodnih podatkov. Merklova drevesa nam torej omogočajo, da se zavežemo L vrednostim $x_1, x_2, \ldots, x_L$ tako, da enostavno vrnemo en sam k-bitni niz. To storimo tako, da shranimo zgostitve vrednosti $x_1, x_2, \ldots, x_L$ v liste drevesa in izračunamo koren. Vrednostim se zavežemo tako, da preverjevalcu damo vrednost korena. Ko mu kasneje želimo razkriti vrednosti, mu jih enostavno povemo, on pa lahko z izračunom drevesa preveri, da so vrednosti prave. Povzeto po [14].



Slika 3: Primer Merklovega drevesa s štirimi listi. Imamo torej 4 vhodne podatke $x_1, x_2, \ldots, x_4$, na vsakem nivoju drevesa se izračuna stik in zgostitev. Ustvarjeno s pomočjo [17].

Trditev 5.6 (Overjevalna pot [14])**.** *Naj bo A zavezovalec, ki se želi zavezati vrednostim $x_1, x_2, \ldots, x_L$ preverjevalcu B. Če se A zaveže tako, da B pove vrednost korena Merklovega drevesa, izračunanega na podlagi vrednosti $x_1, x_2, \ldots, x_L$, potem lahko dokaže zavezo eni vrednosti x_i ($1 \leq i \leq L$) tako, da preverjevalcu B pove $1 + \lceil \log L \rceil$ vrednosti: x_i in pa **overjevalno pot**, torej vrednosti, shranjene v sestrskih vozliščih vseh vozlišč na poti med vključno x_i in korenom (brez korena, ki ga B že pozna).*

Dokaz. Trditev lahko dokažemo z indukcijo na globino Merklovega drevesa d. Naj bo x vrednost, za katero želimo dokazati zavezo in $w_1, w_2, \ldots, w_d$ overjevalna pot.

- $d = 2$: Če je globina drevesa 2, želimo pokazati, da moramo poleg x povemo samo eno dodatno vrednost. Ta vrednost je torej zgostitev x-ove sosednje vrednosti y. Ko B izračuna $H(x||y)$, je to že koren, in lahko vrednost primerja z zabeleženo.

- $n \to n + 1$: Predpostavimo torej, da za globino drevesa n lahko preverimo zavezo s podanimi $1 + d$ vrednostmi. To nam omogoča izračuna vrednosti v enem od korenovih otrok v drevesu globine $n+1$, po indukcijski predpostavki. Če poznamo še vrednost drugega otroka, je možen izračun korena.

□

Varnost Merklovih dreves temelji na odpornosti zgoščevalne funkcije proti trkom. Ker je vsako vozlišče drevesa zgostitev kombinacije otrok, se kakršnakoli sprememba v drevesu prenese v spremembo korena. Torej javna objava korena zaveže objavitelja k vrednostim v listih. Da bi lahko spremenil vrednost lista brez spremembe korena, bi moral najdi trk v zgoščevalni funkciji, in zamenjati vrednost lista z vrednostjo, ki daje enako zgostitev. Če so uporabljene varne zgoščevalne funkcije, je to računsko neizvedljivo.

Merklova drevesa lahko sedaj uporabimo, da naredimo večstranski podpis bolj učinkovit. Podpisniki še vedno predložijo dokaze znanja prek Schnorrovega podpisa, po izmenjavi dokazov pa vsak izračuna Merklovo drevo z listi $I_1, I_2, \ldots, I_L$. To drevo bo torej globine $\log L$. Za svoj javni ključ potem vsak podpisnik P_i nastavi I_i in overjevalno pot za I_i. Ker je ponavadi I_i veliko daljši od dolžine zgostitev, in ker je v

overjevalni poti le $\log L$ zgostitev, se ključ ne podaljša veliko (npr. če je I_i dolg 2000 bitov in so zgostitve 200-bitne, bo 1000 podpisnikov ustvarilo ključe, dolge manj od 4000 bitov, torej manj kot dvakrat daljše).

Ko sedaj preverjevalec preverja podpis sporočila m s strani S, lahko za vsakega podpisnika izračuna vrednost korena Merklovega drevesa in preveri, da koreni za vse podpisnike ujemajo. Obstoj enega nepokvarjenega podpisnika tako prisili vse pokvarjene, da uporabljajo prave ključe (ali pa najdejo trk v zgoščevalni funkciji, kar je izjemno težko). S to spremembo smo tako uspešno ohranili podpise varne in časovno učinkovite.

5.3 Definicija večstranskega Schnorrovega podpisa

Sedaj smo pripravili vse potrebno, da podamo natančno definicijo končne sheme.

5.3.1 Skupni parametri

Vsi podpisniki se strinjajo o skupnem varnostnem parametru k in sekundarnem varnostnem parametru k', ki je deterministično izračunan iz k (v praksi zadostuje kar $k' = 100$). Za velikost L skupine podpisnikov P predpostavimo, da je polinomska v k.

Shemo obravnavamo v modelu slučajnega oraklja, tako da imajo vsi podpisniki dostop do slučajnega oraklja H. Na dogovorjen način ga uporabijo, da ustvarijo pet neodvisnih orakljev

$$\begin{aligned} H_1, H_2 &: \{0,1\}^* \to \{0,1\}, \\ H_3, H_5 &: \{0,1\}^* \to \{0,1\}^{k'}, \\ H_4 &: \{0,1\}^* \to \{0,1\}^{2k'}. \end{aligned}$$

Za lažjo obravnavo si izberemo še enega izmed podpisnikov, in ga označimo z D. Ta podpisnik bo vrnil končni podpis, ne ve pa nobenih dodatnih skrivnosti. Vlogo D si lahko razdelijo vsi podpisniki, a je opis sheme malce lažji, če si izberemo enega – v tem primeru je on prejemnik vseh sporočil ostalih podpisnikov, svoje odgovore pa potem oddaja (angl. *broadcast*) ostalim podpisnikom. Če si ne izberemo podpisnika D, si podpisniki sporočila pošiljajo v krogu (vsak pošlje svoje sporočilo desnemu sosedu, ko prejme sporočilo levega, spet pošlje desnemu. Vsak torej pošlje L sporočil, na koncu imajo vsi vse potrebne podatke za izračune).

Kot že omenjeno, v praksi podpisniki uporabijo dobro poznano grupo in zato ne rabijo generirati parametrov, kar se pa tiče teoretične obravnave, pa morajo sami generirati parametre podpisne sheme. V nadaljevanju bomo predpostavili, imajo podpisniki na voljo skupno grupo G reda q, ki jo generira generator g.

Primer 5.7 (Generiranje parametrov v grupi $\mathbb{Z}_p^*$)**.** Generiranje parametrov je seveda odvisno od izbrane grupe. V primeru $\mathbb{Z}_p^*$ morajo vsi podpisniki generirati parametre p, q in g. Za prva dva uporabijo algoritem $GenPrimes(k)$ 1. Za izvor naključnosti uporabijo zaporedje $H_1(2^k), H_1(2^k+1), \ldots$, Za generiranje elementa g reda q uporabijo izvor naključnosti $H_2(2^k), H_2(2^k+1), \ldots$ ♢

5.3.2 Generiranje ključev

Ko ima vsak podpisnik P_i $(1 \leq i \leq L)$ na voljo skupne parametre, si enakomerno naključno izbere svoj zasebni ključ $s_i \in \mathbb{Z}_q$ in izračuna

$$I_i = g^{s_i}.$$

Do tu je postopek enak, kot pri naivni verziji. Da preprečimo napad na generiranje ključev, mora vsak podpisnik na tem mestu izbrati naključen $u_i \in \mathbb{Z}_q$ in izračunati zavezo

$$X_i = g^{u_i}.$$

Potem si podpisniki med seboj izmenjajo vrednosti (X_i, I_i). Vsak na tem mestu lahko izračuna

$$\begin{aligned} e &= H_3(X_1||I_1||X_2||I_2||\ldots||X_L||I_L), \\ y_i &= es_i + u_i \bmod q, \end{aligned}$$

in pošlje y_i vsem ostalim podpisnikom.

Na tem mestu ima vsak podpisnik P_i na voljo par (X_j, y_j) vseh ostalih podpisnikov $(1 \leq j \leq L)$. Ta par predstavlja Schnorrov podpis in s tem dokaz znanja brez razkritja znanja zasebnega ključa s_j glede na javni ključ I_j. Ker je izziv e izračunan iz podatkov vseh podpisnikov, onemogoča naknadne spremembe ključev.

Vsak podpisnik P_i mora preveriti veljavnost vseh podpisov (X_j, y_j) vseh ostalih podpisnikov P_j $(1 \leq j \leq L)$. To stori s standardnim izračunom za preverjanje Schnorrovega podpisa (3.1):

$$g^{y_j} \stackrel{?}{=} X_j \cdot I_j^e.$$

Ko podpisnik P_i ugotovi, da so vsi podpisi veljavni, lahko izračuna Merklovo drevo, ki ima v listih po vrsti razporejene $I_1, I_2, \ldots, I_L$ in za svojo zgoščevalno funkcijo uporablja H_4. Ko je drevo izračunano, P_i prebere overjevalno pot pot_i svojega ključa I_i in vrne javni ključ

$$\text{pk}_i = (q, g, I_i, \text{pot}_i).$$

5.3.3 Podpisovanje

Naj bo $S = \{P_{id_1}, P_{id_2}, \ldots, P_{id_l}\}$ podskupina velikosti l skupine P, ki želi večstransko podpisati sporočilo m. To lahko storijo s tremi krogi komunikacije:

1. Vsak podpisnik $P_{id_j} (1 \leq j \leq l)$ si izbere naključno število $r_j \in \mathbb{Z}_q$ in izračuna zavezo

   $$X_j = g^{r_j}.$$

   Zavezo X_j nato pošlje podpisniku D.

2. D izračuna skupno zavezo

   $$\tilde{X} = (X_1 \cdot X_2 \cdot \cdots \cdot X_l)$$

   in jo pošlje vsem podpisnikom $P_{id_j} (1 \leq j \leq l)$.

3. Vsak podpisnik $P_{id_j} (1 \leq j \leq l)$ izračuna

$$e = H_5(\tilde{X}||m||S),$$
$$y_j = es_j + r_j \bmod q,$$

in pošlje y_j podpisniku D.

Na tem mestu ima D vse, kar potrebuje, da dokonča podpis. Najprej izračuna

$$\tilde{y} = (y_1 + y_2 + \cdots + y_l) \bmod q$$

in nato vrne končni podpis

$$\sigma = (\tilde{X}, \tilde{y}).$$

5.3.4 Preverjanje

Preverjanje je podobno naivni verziji, le da je treba dodatno preveriti ustreznost Merklovih korenov. Če torej preverjevalec želi preveriti veljavnost podpisa $\sigma = (\tilde{X}, \tilde{y})$ sporočila m, moramo predpostaviti, da lahko dostopa do vseh javnih ključev $\{\mathrm{pk}_{id_1}, \mathrm{pk}_{id_2}, \ldots, \mathrm{pk}_{id_l}\}$ vseh podpisnikov $S = \{P_{id_1}, P_{id_2}, \ldots, P_{id_l}\}$.

Najprej mora preverjevalec preveriti, da se vseh l izvodov skupnih parametrov q in g ujema (preveri torej, da so vsi podpisniki uporabljali enako grupo in generator). Da dokončno potrdi veljavnost in ustreznost javnih ključev, za vsakega podpisnika $P_{id_j} (1 \leq j \leq l)$ s pomočjo njegovega javnega ključa I_{id_j} in overjevalne poti pot_{id_j} izračuna vrednost v korenu Merklovega drevesa V_{id_j}, kot prikazano v trditvi 5.6. Ko izračuna vse vrednosti korenov V_{id_j}, mora preveriti, da so res enake.

Ko se preverjevalec prepriča o veljavnosti ključev, postopa podobno kot pri naivni verziji. Najprej mora izračunati

$$\tilde{I}_S = (I_{id_1} \cdot I_{id_2} \cdot \cdots \cdot I_{id_l}).$$

Če si dosledno beleži izračune, lahko preverjevalec ta del preverjanja podpisa opravi le enkrat za vsako podskupino podpisnikov S.

Potem mora preverjevalec izračunati izziv e s pomočjo slučajnega oraklja

$$e = H_5(\tilde{X}||m||S)$$

in preveriti, če velja

$$g^{\tilde{y}} \stackrel{?}{=} \tilde{X}\tilde{I}_S^e. \tag{5.2}$$

Če je torej podpis $\sigma = (\tilde{X}, \tilde{y})$ veljaven podpis sporočila m, lahko levo stran enačbe (5.2) zapišemo kot

$$\begin{aligned} g^{\tilde{y}} &= g^{(y_1+y_2+\cdots+y_l) \bmod q} \\ &\stackrel{2.5}{=} g^{y_1+y_2+\cdots+y_l} \\ &= g^{es_1+r_1+es_2+r_2+\cdots+es_l+r_l} \\ &= g^{r_1+r_2+\cdots r_l} g^{e(s_1+s_2+\cdots s_l)}, \end{aligned}$$

kjer smo pri prehodu iz prve v drugo vrstico uporabili lastnost 2.5.

Desno stran enačbe (5.2) pa lahko zapišemo kot

$$\begin{aligned}\tilde{X}\tilde{I}_S^e &= (X_1 \cdot X_2 \cdot \cdots \cdot X_l)(I_{id_1} \cdot I_{id_2} \cdot \cdots \cdot I_{id_l})^e \\ &= (g^{r_1} \cdot g^{r_2} \cdot \cdots \cdot g^{r_l})(g^{s_1} \cdot g^{s_2} \cdot \cdots \cdot g^{s_l})^e \\ &= g^{r_1+r_2+\cdots r_l} g^{e(s_1+s_2+\cdots s_l)}.\end{aligned}$$

Ker se leva in desna stran ujemata, enačba (5.2) pravilno preveri pravilnost večstranskega Schnorrovega podpisa.

5.4 Varnost večstranskega Schnorrovega podpisa

Dokazati si želimo, da je shema, definirana v zgornjem razdelku 5.3, *varna*, torej ustreza pogojem definicije varnosti 5.3, kjer ima napadalec F sposobnosti, definirane v 5.2. Varnost torej pomeni, da je preprečimo eksistencialno ponarejanje, zaradi napadalčevega nadzora nad komunikacijskim omrežjem pa si lahko privošči napad z izbranimi sporočili. Še več, ker se ukvarjamo z večstranskimi podpisi, si napadalec lahko izbere še podskupino, ki bo sporočilo podpisovala. Napadalec lahko torej na katerikoli točki od katerekoli podskupine zahteva podpis kateregakoli izbranega sporočila.

Zaradi kompleksnosti napadalca, dokaz varnosti poteka tako, da najprej dokažemo ekvivalentnost med šibkejšim napadalcem F' in prvotnim napadalcem F, potem pa dokažemo varnost v primeru šibkega napadalca F'. Še pred tem pa moramo dokazati lemo, ki je ključna za večino dokazov varnosti podpisnih shem v modelu slučajnega oraklja.

5.4.1 Lema o razcepu

Lema o razcepu (angl. *Forking lemma*) je ključen rezultat pri dokazovanju varnosti z redukcijo, ki omogoča pretvorbo ponarejenega podpisa v rešitev problema diskretnega logaritma. Je formalno orodje, prek katerega pridobimo več povezanih ponarejenih podpisov, ki jih lahko združimo v rešitev problema diskretnega logaritma. Temelji na izvedbi dveh napadov z enakimi vhodnimi podatki in parametri, a drugačnim slučajnim orakljem.

Pred dokazom leme o razcepu moramo najprej pokazati enostavnejšo lemo iz verjetnosti.

Lema 5.8. *Naj bosta $\mathcal{X}$ in $\mathcal{Y}$ množici. Naj bo (X, Y) par slučajnih spremenljivk z vrednostmi iz $\mathcal{X}$ in $\mathcal{Y}$, ki ima poljubno verjetnostno porazdelitev. Naj bo $0 < \epsilon < 1$ in naj bo $A \subseteq \mathcal{X} \times \mathcal{Y}$ podmnožica, za katero velja*

$$\Pr((X, Y) \in A) \geq \epsilon.$$

Potem obstaja podmnožica $\Omega \subset \mathcal{X}$, tako da velja:

- $\Pr(X \in \Omega) > \epsilon/2$ *(posebej to pomeni, da $\Omega \neq \emptyset$) in*
- *za vsak a iz Ω velja, da* $\Pr((a, Y) \in A) \geq \epsilon/2$.

Dokaz. Za vsak x iz $\mathcal{X}$ definirajmo $p(x) : \mathcal{X} \to [0,1]$ kot

$$p(x) = \Pr((x, Y) \in A \mid X = x).$$

Zakon o popolni verjetnosti nam pove, da velja

$$\Pr((X, Y) \in A) = \sum_{x \in \mathcal{X}} \Pr(X = x) \cdot p(x) = \mathrm{E}[p(X)].$$

Ker je po predpostavki velja $\Pr((X, Y) \in A) \geq \epsilon$, velja tudi

$$\mathrm{E}[p(X)] \geq \epsilon.$$

Sedaj lahko definiramo množico Ω kot

$$\Omega = \{x \in \mathcal{X} \mid p(x) \geq \epsilon/2\} \subseteq \mathcal{X}.$$

Neposredno iz definicije množice sledi drugi del leme: če je a v Ω, mora zanj veljati

$$p(a) = \Pr((a, Y) \in A \mid X = a) \geq \epsilon/2.$$

Za prvi del leme pa najprej predpostavimo obratno, da je $\Pr(X \in \Omega) \leq \epsilon/2$. Ker za x, ki ni v Ω velja $p(x) < \epsilon/2$ (zunaj Ω je $p(x)$ strogo manjši od $\epsilon/2$), lahko pričakovano vrednost ocenimo kot

$$\begin{aligned}
\mathrm{E}[p(X)] &= \mathrm{E}[p(X) \mid X \in \Omega] \Pr(X \in \Omega) + \mathrm{E}[p(X) \mid X \notin \Omega] \Pr(X \notin \Omega) \\
&\leq 1 \cdot \Pr(x \in \Omega) + \frac{\epsilon}{2} \cdot \Pr(x \notin \Omega) \\
&= \Pr(x \in \Omega) + \frac{\epsilon}{2} \cdot (1 - \Pr(x \in \Omega)) \\
&\leq \frac{\epsilon}{2} + \frac{\epsilon}{2} \cdot \left(1 - \frac{\epsilon}{2}\right) \\
&= \epsilon - \frac{\epsilon^2}{4} \\
&< \epsilon.
\end{aligned}$$

To je v nasprotju z našo predpostavko, da velja $\mathrm{E}[p(X)] \geq \epsilon$. Tako smo pokazali, da mora veljati $\Pr(X \in \Omega) > \epsilon/2$, s čimer smo dokazali prvi del leme. □

Sedaj imamo vse potrebno, da dokažemo lemo o razcepu.

Izrek 5.9 (Lema o razcepu [12])**.** *Naj bo $\mathcal{S}$ podpisna shema, ki vrača dvodelne podpise $\sigma = (\sigma_1, \sigma_2)$, kjer σ_2 temelji na zgostitvi vhodnega sporočila m in σ_1 (npr. pri Schnorrovem podpisu je ta zgostitev izziv $e = H(enc(X)||m)$). Naj slučajni orakelj H vrača zgostitve dolžine n. Varnostni parameter naj bo k, tako da velja $n > k$.*

Naj bo napadalec F naključnostni polinomski algoritem, ki ima na voljo vse javne podatke pri podpisni shemi $\mathcal{S}$ in dostop do oraklja H. Če lahko F z nezanemarljivo verjetnostjo vrne par (m, σ), kjer je σ veljaven podpis sporočila m, potem lahko z nezanemarljivo verjetnostjo z enakimi vhodnimi podatki in uporabo drugega oraklja, F vrne dva para (m, σ) in $(m.\sigma')$, kjer podpisa temeljita na drugačnih zgostitvah.

Dokaz leme o razcepu 5.9. Naj bo napadalec F napadalec naključnostni algoritem, ki teče v polinomskem času. Z ω označimo vir naključnih bitov, do katerega ima F dostop. Denimo, da med svojim napadom napadalec F slučajnemu oraklju H pošlje največ polinomsko mnogo poizvedb v varnostnem parametru k. Označimo poizvedbe, ki jih F stori $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_Q$ (brez škode za splošnost lahko predpostavimo, da so poizvedbe med seboj različne). Pripadajoče odgovore oraklja na poizvedbe označimo $\rho_1, \rho_2, \ldots, \rho_Q$.

Predpostavimo, da lahko za naključno izbran vir bitov ω in naključno izbran orakelj H, napadalec F z nezanemarljivo verjetnostjo vrne par (m, σ), kjer je σ veljaven podpis sporočila m. Vir naključnosti Verjetnost tu jemljemo nad vsemi viri bitov ω in nad vsemi orakljevimi odgovori $\rho_1, \rho_2, \ldots, \rho_Q$. To verjetnost označimo z $\Pr_{\omega,\rho_1,\rho_2,\ldots,\rho_Q}$.

Ker so odgovori oraklja izbrani iz naključne funkcije, je verjetnost, da je bila ena izmed poizvedb oraklju prav tista, ki je vsebovala sporočilo m in delni podpis σ_1, nezanemarljiva. Sicer bi moral napadalec F uganiti odgovor na to poizvedbo, da bi lahko ustvaril drugi del podpisa σ_2 (uganiti bi torej moral naključno vrednost, kar lahko stori z največ zanemarljivo verjetnostjo). Označimo poizvedbo s temi podatki $\mathcal{Q}_\beta = (m, \sigma_1)$ $(1 < \beta < Q)$. Ker je poizvedb polinomsko mnogo v varnostnem parametru k in je verjetnost uspeha nezanemarljiva, morata obstajati polinom $P(k)$ in število β, za katera je verjetnost uspeha

$$\Pr_{\omega,\rho_1,\rho_2,\ldots,\rho_Q}(F \text{ vrne veljaven ponaredek in } \mathcal{Q}_\beta = (m, \sigma_1)) \geq \frac{1}{P(k)},$$

kjer je poizvedba $\mathcal{Q}_\beta$ res temeljila na sporočilu in delnem podpisu, slučajnost pa izvira iz vira bitov ω in orakljevih odgovorov $\rho_1, \rho_2, \ldots, \rho_Q$.

Če označimo z Ω množico vseh naključnih virov bitov ω in z R množico vseh zaporedij $\rho_1, \rho_2, \ldots, \rho_Q$ orakljevih odgovorov, lahko uporabimo lemo 5.8, kjer je $X = \Omega$, $Y = R$, A je dogodek, da je F uspešen in $\epsilon = 1/P(k)$. Lema nam zagotavlja obstoj množice Ω_β, ki vsebuje vse vire bitov ω, za katere velja, da je verjetnost uspeha po orakljevih odgovorih $\rho_1, \ldots, \rho_Q$ navzdol omejena z $1/2P(k)$, oz., da za vir bitov ω iz množice Ω_β velja

$$\Pr_{\rho_1,\rho_2,\ldots,\rho_Q}(F \text{ vrne veljaven ponaredek in } \mathcal{Q}_\beta = (m, \sigma_1)) \geq \frac{1}{2P(k)}.$$

Sedaj lahko poleg β fiksiramo še vir bitov ω iz množice Ω_β. To nam da deterministično zaporedje poizvedb $Q_1, Q_2, \ldots, Q_Q$, kjer velja $\mathcal{Q}_\beta = (m, \sigma_1)$, napadalec pa uspe z nezanemaljivo verjetnostjo. Na tem mestu lahko ponovno uporabimo lemo 5.8, kjer je X množica odgovoro na prvih $\beta - 1$ poizvedb, Y množica odgovorov na preostalih $Q - \beta + 1$ poizvedb, A je dogodek, da je F uspešen in $\epsilon = 1/2P(k)$. To nam zagotavlja obstoj množice $R_{\beta,\omega}$, ki vsebuje vsa zaporedja $(\rho_1, \ldots, \rho_{\beta-1})$, za katera velja, da je verjetnost uspeha po preostanku odgovorov $(\rho_\beta, \ldots, \rho_Q)$ večja od $1/4P(k)$. To pomeni, da za zaporedje $(\rho_1, \ldots, \rho_{\beta-1})$ iz množice $R_{\beta,\omega}$, velja

$$\Pr_{\rho_\beta,\ldots,\rho_Q}(F \text{ vrne veljaven ponaredek in } \mathcal{Q}_\beta = (m, \sigma_1)) \geq \frac{1}{4P(k)}.$$

Množici Ω_β in $R_{\beta,\omega}$ določata možne β, ω in $(\rho_1, \ldots, \rho_{\beta-1})$, za katere si lahko izberemo dve množici odgovorov $(\rho_\beta, \ldots, \rho_Q)$ in $(\rho'_\beta, \ldots, \rho'_Q)$, za kateri napadalec F

z nezanemarljivo verjetnostjo vrne dva para (m, σ) in (m, σ'), kjer sta σ in σ' veljavna podpisa sporočila m, ki temeljita na drugačnih zgostitvah. Tu smo uporabili pogoj $n \gg \log(k)$ oz., da je dolžina orakljevih odgovorov n veliko večja od varnostnega parametra k. Ta pogoj nam zagotavlja, da je veliko različnih oraljevih odgovorov, oz. da je verjetnost, da se zgodi trk, zanemarljiva.

Torej, z naključno izbiro β, ω, $(\rho_1, \ldots, \rho_{\beta-1}), (\rho_\beta, \ldots, \rho_Q)$ in $(\rho'_\beta, \ldots, \rho'_Q)$ lahko z nezanemarljivo verjetnostjo dobimo dva veljavna podpisa σ in σ' sporočila m, ki temeljita na drugačnih zgostitvah. □

5.4.2 Šibek napadalec in šibka varnost

Nasledji korak pri dokazu varnosti je definicija **šibkega napadalca** (ali nasprotnika) (angl. *weak adversary*) F'. Njegov cilj je enak izvornemu napadalcu F. Želi si torej ponarediti podpis s pomočjo napada z izbranim sporočilom in podskupino. Izbere si lahko podpisnika, in od njega zahteva podpis izbranega sporočila skupaj z izbrano podskupino S. Napadalca se razlikujeta v tem, da je F' bistveno šibkejši, oz. ima manj nadzora nad potekom podpisovanja in komunikacijo med podpisniki.

Definicija 5.10 (Šibek napadalec pri ASM)**.** Šibek napadalec F' ima v modelu večstranskega podpisa podskupine z odgovornostjo naslednje zmožnosti:

- Preden skupina P generira svoje ključe, izbere podpisnika P_i, ki ga bo napadel. Ta podpisnik bo edini, ki bo pri izvedbi podpisne sheme pošten.
- Ko je podpisnik P_i napaden, mora F' zanj priskrbeti vse vhode za algoritme in vse vhodna sporočila, ki jih prejme. Prav tako lahko vidi vsa poslana sporočila podpisnika. Efektivno pri podpisovanju sodelujeta le F' in P_i.
- Po generiranju ključev, lahko šibek napadalec F' od podpisnika P_i zahteva podpis izbranega sporočila skupaj z izbrano podskupino S (izvede napad z izbranim sporočilom in podskupino).

Ker mora šibek napadalec za izbranega podpisnika predložiti vse vhode in komunikacijo, lahko vidimo delovanje šibkega napadalca F' kot delovanje napadalca F, ki pokvari vse podpisnike, razen podpisnika P_i. V obeh primerih bo podpisnik P_i v podpisu sodeloval le z napadalcem, ki zanj nadzoruje celoten potek podpisovanja. To povezavo tipov napadalcev formalno opišemo v izreku 5.12. Pred dokazom izreka pa moramo, tako kot za napadalca F, tudi za šibkega napadalca definirati, kaj pomeni varnost sheme.

Definicija 5.11 (Šibka varnost pri ASM)**.** Naj bo k varnostni parameter (ki si ga delijo vsi podpisniki). Naj bo $c > 0$ poljubna konstanta. Naj bo F' šibek napadalec, ki je omejen s polinomskim časom v parametru k in ustreza definiciji 5.10. Naj bo P_i podpisnik, ki ga šibek napadalec napada. Naj bo p verjetnost, da šibek napadalec F' vrne trojico (σ, m, S), za katero velja:

- σ je veljaven večstranski podpis sporočila m s strani skupine S.
- Podpisnik P_i je v skupini S, vendar šibek napadalec F' od njega ni zahteval podpisa sporočila m s strani skupine S.

Shemi za večstranske podpise podskupine z odgovornostjo bomo rekli, da je **šibko varna**, če je verjetnost p zanemarljiva, oz. če za vsakega polinomsko omejenega šibkega napadalca F' obstaja zanemarljiva funkcija ε, da velja

$$p < \varepsilon(k).$$

Sedej imamo vse potrebno, da dokažemo ekvivalentnost šibke varnosti in varnosti, in s tem ekvivalentnost šibkega napadalca in napadalca.

Izrek 5.12. *Naj bo velikost L skupine podpisnikov P polinomsko omejena v varnostnem parametru k. Potem je večstranski Schnorrov podpis skupine velikosti L iz razdelka 5.3 varen natanko tedaj, ko je šibko varen.*

Dokaz. Ker se definicija varnosti in šibke varnosti razlikuje le v uporabi napadalca F ali šibkega napadalca F', je potrebno pokazati le, da sta F in F' ekvivalentna (ob ustrezni velikost skupine podpisnikov).

($\Rightarrow$): Predpostavimo, da podpis ni šibko varen. Kot smo že prej smo omenili, je delovanje šibkega napadalca F' enako delovanju napadalca F, ki pokvari vse podpisnike razen enega. Napadalec F ima torej vse zmožnosti, ki jih ima šibek napadalec F'. Če podpis ni šibko varen, lahko F s simulacijo F', opisano zgoraj, z nezanemarljivo verjetnostjo vrne ponarejen podpis, torej shema tudi ni varna.

($\Leftarrow$): Predpostavimo, da podpis ni varen. Obstaja torej napadalec F, ki je omejen s polinomskim časom v varnostnem parametru k in uspešno izvede napad z izbranim sporočilom in podskupino.

Konstruirajmo šibkega napadalca F': skladno z definicijo si pred generiranjem ključev enakomerno naključno izbere podpisnika P_i ($1 \leq i \leq L$), ki ga bo napadel. Od tega trenutka naprej pri podpisovanju sodelujeta samo podpisnik P_i in šibek napadalec F', ki ima vlogo vseh ostalih podpisnikov.

Ko napadalec F izvede napad, mora torej šibek napadalec F' simulirati delovanje vseh podpisnikov razen P_i. Vedenje šibkega napadalca F' je tekom napada odvisno od dejanj napadalca F:

- Če se napadalec F odloči pokvariti kateregakoli podpisnika, ki ni P_i, mu mora šibek napadalec F' (v vlogi podpisnika) posredovati potrebne zasebne informacije. To lahko stori, saj simulira delovanje podpisnika.
- Če napadalec F pokvari podpisnika P_i, je šibek napadalec pri svojem napadu neuspešen.
- Če si napadalec F izbere podpisnika (razen P_i) za tarčo napada z izbranim sporočilom in podskupino, mora šibek napadalec F' simulirati odziv podpisnika. To ponovno lahko stori, saj simulira delovanje podpisnika.
- Če napadalec F izbere podpisnika P_i za tarčo napada z izbranim sporočilom in podskupino, šibek napadalec F' prejme zahtevek za napad od F in ga posreduje podpisniku P_i. Ta podpisnik normalno pošlje svoj odgovor napadalcu F.

Ko F konča svoj napad, vrne trojico (σ, m, S). Če je to veljaven ponaredek, jo vrne tudi šibek napadalec F'. Če ponaredek ni veljaven, če P_i ni eden izmed podpisnikov v S, ali pa če je P_i tekom podpisovanja pokvarjen s strani F, je šibek

napadalec neuspešen pri svojem napadu. Poglejmo si verjetnost, da F' uspe pri svojem napadu:

$$\begin{aligned}
&\Pr(F' \text{ vrne veljaven ponaredek}) = \\
=&\Pr(F \text{ vrne veljaven ponaredek } \cap\ P_i \in S \text{ in ni pokvarjen}) \\
=&\sum_{i=1}^{L} \Pr(F' \text{ izbere } P_i) \cdot \Pr(F \text{ vrne veljaven ponaredek } \cap\ P_i \in S \text{ in ni pokvarjen}) \\
=&\sum_{i=1}^{L} \frac{1}{L} \cdot \Pr(P_i \in S \text{ in ni pokvarjen} \mid F \text{ vrne veljaven ponaredek}) \\
&\cdot \Pr(F \text{ vrne veljaven ponaredek}) \\
\geq&\frac{1}{L} \cdot \Pr(F \text{ vrne veljaven ponaredek}),
\end{aligned}$$

kjer smo pri prehodu iz tretje v četrto vrstico uporabili osnovno lastnost verjetnosti, ki pravi

$$\Pr(A \cap B) = \Pr(A \mid B) \cdot \Pr(B),$$

pri prehodu v zadnjo vrstico pa smo upoštevali, da velja

$$\sum_{i=1}^{L} \Pr(P_i \in S \text{ in ni pokvarjen} \mid F \text{ vrne veljaven ponaredek}) \geq 1. \tag{5.3}$$

Neenačba (5.3) drži, saj mora pri uspešnem ponarejanju sodelovati vsaj en nepokvarjen podpisnik. Če torej F vrne veljaven ponaredek, bo vsaj ena od verjetnosti v zgornji vsoti enaka 1.

Pri ponarejanju je torej F uspešen največ L-krat več kot F', kar pomeni, da shema ni šibko varna, saj je velikost skupine L polinomsko omejena. □

5.4.3 Dokaz varnosti

Ker sta napadalec in šibek napadalec ekvivalentna, je dovolj pokazati varnost v primeru šibkega napadalca, shema pa je potem tudi varna po izreku 5.12.

Izrek 5.13 (Varnost večstranskega Schnorrovega podpisa)**.** *Če drži predpostavka o težavnosti diskretnega logaritma pri ASM 2.13, je večstranski Schnorrov podpis, definiran v razdelku 5.3, varen.*

Dokaz. Želimo torej pokazati, da če obstaja šibek napadalec, ki je omejen s polinomskim časom v varnostnem parametru k, potem obstaja nek algoritem A, ki prav tako teče v polinomskem času, s katerim lahko prekršimo predpostavko o diskretnem logaritmu pri ASM.

Naj bo F' šibek napadalec, ki uspešno izvede napad z izbranim sporočilom in podskupino, P_i pa podpisnik, ki ga napada (in je torej edini pošten podpisnik). Javni ključ podpisnika P_i označimo I_i.

Konstruirali bomo algoritem A, ki za vhod prejme javne parametre sheme in vrednost za katero želi izračunati diskretni logaritem, torej q, g in I, in vrne $s \in \mathbb{Z}_q$, da velja $g^s = I$. Ideja je, da A simulira proces podpisovanja za šibkega napadalca F'.

Cilj algoritma A je, da iz vrnjenega ponarejenega podpisa, pridobljenega ob uspešnem napadu, razbere diskretni logaritem, ki ga išče. Za doseganje tega cilja mora algoritem A biti sposoben simulirati delovanje vseh aspektov procesa podpisovanja (od slučajnega oraklja do podpisnika P_i, ki ga na začetku izbere šibek napadalec F' za svojo tarčo).

Algoritem A mora najprej doseči, da sta parametra q in g iz predloženega problema diskretnega logaritma točno deljeni javni parametri pri večstranskem Schnorrovem podpisu. To doseže prek manipulacije izvora naključnosti za generiranje parametrov, torej orakljev H_1 in H_2. Ker algoritem A simulira delovanje vseh slučajnih orakljev, lahko nastavi H_1 in H_2 tako, da bodo rezultati algoritma za generiranje ključev točno q in g. Za konec nastavi še javni ključ I_i podpisnika P_i na I.

Okvirno lahko delovanje šibkega napadalca F' opišemo prek dveh vrst poizvedb, ki jih opravlja pri slučajnem oraklju:

- *zgoščevalna poizvedba*, kjer F' pošlje slučajnemu oraklju nek niz, ta pa mu odgovori z njegovo zgostitvijo. Algoritem A lahko simulira odgovor na tovrstne poizvedbe z vračanjem naključnih odgovorov. Največje število zgoščevalnih poizvedb, ki jih opravi F', označimo $q_{\text{zgoščevalna}}$.

- *podpisovalna poizvedba*, kjer šibek napadalec F' podpisniku P_i (ki ga je izbral na začetku) pošlje sporočilo m in skupino S, ta pa mu nazaj pošlje zavezo X_i. Potem mora šibek napadalec F' igrati vlogo podpisnika D, ki agregira informacije (tekom podpisovanja zbere individualne zaveze in podpise, vrača pa skupne). Podpisniku P_i pošlje skupno zavezo $\tilde{X}$, nazaj pa dobi y_i. Podpisovalne poizvedbe torej potekajo v dveh krogih in predstavljajo izvedbo napada z izbranim sporočilom in podskupino nad podpisnikom P_i, katerega delovanje mora uspešno simulirati algoritem A. Največje število teh poizvedb, ki jih opravi F', označimo $q_{\text{podpisovalna}}$.

Za lažjo konstrukcijo algoritma A, je smotrno šibkega napadalca F' še dodatno poenostaviti. Naj $\mathcal{F}$ označuje *poenostavljenega napadalca*, ki deluje popolnoma enako kot šibek napadalec F', le da pred drugim krogom podpisovalne poizvedbe $\mathcal{F}$ najprej pošlje zgoščevalno poizvedbo slučajnemu oraklju H_5 z vhodom $(\tilde{X}, m, S)$. Podobno, preden $\mathcal{F}$ vrne ponarejen podpis $(\tilde{X}, \tilde{y})$ sporočila m_F, najprej opravi zgoščevalno poizedbo pri slučajnemu oraklju H_5 z vhodom $(\tilde{X}, m_F, S)$. Tako $\mathcal{F}$ vsaki podpisovalni poizvedbi in ponaredku dodeli zgostitev.

Dodatno predpostavimo še, da poenostavljen napadalec $\mathcal{F}$ po vsaki zgoščevalni poizvedbi shrani odgovor, zato vsako zgoščevalno poizvedbo za določen vhod opravi le enkrat.

Poenostavljen napadalec $\mathcal{F}$ torej opravi $q_{\text{podpisovalna}}$ podpisovalnih poizvedb in $q_{\text{zgoščevalna}} + q_{\text{podpisovalna}} + 1$ zgoščevalnih poizvedb. Skupno število klicev oraklja H_5 s strani $\mathcal{F}$ označimo q_H. Shranjevanje odgovorov na poizvedbe je računsko zanemarljivo, zato $\mathcal{F}$ opravi toliko dela, kot šibek napadalec F' skupaj s stroškom poizvedb. Če torej predpostavimo učinkovitost slučajnega oraklja, teče $\mathcal{F}$ v polinomskem času, saj tudi šibek napadalec F' teče v polinomskem času.

Algoritem A bo za delovanje uporabljal poenostavljenega napadalca $\mathcal{F}$, saj bo tako za vsak potencialni ponaredek in vsak napad prejel predogled rezultata prek zgostitve, ki jo vrne H_5.

Algoritem A mora torej znati odgovoriti na vse poizvedbe poenostavljenega napadalca $\mathcal{F}$. Na zgoščevalne poizvedbe odgovarja naključno, s pomočjo vnaprej izbranih simuliranih odgovorov $e_1, e_2, \ldots, e_{q_H}$ oraklja H_5. Odgovor na podpisovalno poizvedbo za podpisnika P_i na vhodu m in S je malce bolj zapleten. Algoritem A stori sledeče:

1. Shrani konfiguracijo poenostavljenega napadalca $\mathcal{F}$.
2. Si izbere naključen odgovor e_j $(1 \leq j \leq q_H)$ oraklja H_5, v upanju, da bo to zgostitev, ki bo identificirala to poizvedbo.
3. Si izbere naključen $y \in \mathbb{Z}_q$.
4. Izračuna
$$X_i = I_i^{-e_j} g^y \bmod p$$
in pošlje X_i v odgovor prvega kroga poizvedbe poenostavljenemu napadalcu $\mathcal{F}$.
5. Po prejetju $\tilde{X}$ od $\mathcal{F}$ preveri, če e_j ustreza poizvedbi na vhodu $(\tilde{X}, m, S)$. Če da, vrne y, sicer se vrne na korak 2.

Zgornje je primer splošne metode za analiziranje varnosti v kriptografiji, ki se imenuje **previjanje nazaj** (angl. *rewinding*). Popularna je pri analizi dokazov brez razkritja znanja in pri dokazih varnosti v modelu slučajnega oraklja. Metoda dovoli, da prek naključnih izbir algoritem odgovori na poizvedbo, preden je postavljena, za ceno največ toliko previjanj, kot ima poizvedba odgovorov.

Opomba 5.14. V kontekstu dokaza varnosti je napadalec naključnostni algoritem, omejen s polinomskim časom. Simulator (algoritem A), ki ga konstruiramo, pa je algoritem, ki tekom svojega delovanja uporablja napadalca kot podprogram. To simulatorju omogoča, da pozna celotno interno stanje napadalca, vključno z vsemi naključnimi izbirami, ki jih je napadalec opravil. Zato lahko simulator shranjuje stanja napadalca, in shranjena stanja uporabi, da ga previje nazaj na prejšnje stanje.

V tem primeru je previjanje nazaj nujno potrebno za simuliranje iskrenega podpisnika P_i. To je zato, ker poenostavljen napadalec $\mathcal{F}$ igra vlogo vseh ostalih podpisnikov, torej ima odločilno vlogo pri izbiri skupne zaveze $\tilde{X}$. Na podlagi te zaveze je izračunan skupni izziv $e = H_5(\tilde{X}||m||S)$, ki ga P_i uporabi za izračun y_i.

Ko algoritem A simulira podpisni proces za poenostavljenega napadalca $\mathcal{F}$, mora torej A izračunati zavezo X_i in jo poslati v prvem krogu poizvedbe (namesto podpisnika P_i). Problem je, da na tej točki še ni znan izziv e. Za uspešno simulacijo mora algoritem A uganiti, kateri odgovor oraklja H_5 bo ustrezal vhodu. Ker je možnih odgovorov oraklja q_H in je vsaka enako verjetna, je verjetnost, da algoritem A izbere pravilen odgovor, $1/q_H$. Pričakovano število potrebnih previjanj je torej q_H.

Sedaj lahko konstruiramo algoritem A na podlagi poenostavljenega napadalca $\mathcal{F}$. A torej prejme instanco problema diskretnega logaritma s podatki p, q, g in I (kot predstavljeno v definiciji 5.5).

Naslednji korak je dokaz znanja diskretnega logaritma $g^s \equiv I \pmod{q}$. Ta diskretni logaritem je točno ta, ki ga algoritem A išče, torej ga na tem mestu še ne pozna. Dokaz znanja lahko zaobidemo na podoben način, kot pri odgovarjanju na

podpisovalne poizvedbe. V jedru problema je ponovno dejstvo, da mora A poslati zavezo X_i, preden pozna izziv e. Algoritem A mora torej ponovno uganiti pravilen odgovor, tokrat oraklja H_3. Ponovno to lahko doseže s postopkom previjanja poenostavljenega napadalca $\mathcal{F}$. Če označimo število poizvedb, ki jih $\mathcal{F}$ pošlje oraklju H_3 s q_{H_3}, lahko izračunamo, da je pričakovano število previjanj, ki jih opravi algoritem A, enako q_{H_3}.

Na tem mestu je vse pripravljeno, da lahko algoritem A začne simulirati postopek podpisovanja in požene poenostavljenega napadalca $\mathcal{F}$ ter odgovarja na njegove poizvedbe kot opisano zgoraj. Ideja dokaza od tu naprej je, da ko poenostavljen napadalec $\mathcal{F}$ vrne ponarejen podpis, algoritem A uporabi previjanje in **lemo o razcepu** 5.9, da pridobi še en ponarejen podpis. Iz teh dveh ponaredkov potem lahko izlušči odgovor na dani problem diskretnega logaritma.

Denimo, da $\mathcal{F}$ vrne ponarejen podpis $(\tilde{X}_0, \tilde{y}_0)$ sporočila m_0 s strani podskupine S_0. Še več, ponaredek je nastal na podlagi j_0-te poizvedbe oraklju H_5 z vhodom $(\tilde{X}_0, m_0, S_0)$ in odgovorom e_{j_0}. Ker A pozna celotno notranje stanje poenostavljenega napadalca $\mathcal{F}$ in si je shranjeval njegova interna stanja, lahko simulator A previje poenostavljenega napadalca $\mathcal{F}$ nazaj na začetek podpisovnja z enakim virom naključnih bitov ω. Na njegove poizvedbe odgovarja enako kot prej, le na j_0-to poizvedbo odgovori z novo naključno vrednostjo e'_{j_0}. Delovanje napadalca $\mathcal{F}$ pred previjanjem imenujemo prvi pogon, delovanje po previjanju pa drugi pogon.

Ker so podpisovalne poizvedbe razdeljene na dva kroga in temeljijo na zgoščevalnih poizvedbah (za pridobitev izziva e), se lahko zgodi, da je j_0-ta zgoščevalna poizvedba opravljena sredi ene od podpisovalnih. Označimo z r število podpisovalnih poizvedb v prvem pogonu, katerih prvi krog poenostavljen napadalec $\mathcal{F}$ opravi pred j_0-to zgoščevalno poizvedbo. Ker mora legitimen ponarejen podpis podpisovati sporočilo s strani skupine, ki tega sporočila še ni podpisala, lahko sklepamo, da nobena od podpisovalnih poizvedb v prvem pogonu ni temeljila na j_0-ti zgoščevalni poizvedbi.

Ker je pri drugem pogonu uporabljen enak vir naključnih bitov ω in so odgovori na poizvedbe do j_0-te enaki, nobena od prvih $r-1$ podpisovalnih poizvedb v drugem pogonu ni temeljila na j_0-ti zgoščevalni. V drugem pogonu napadalca lahko algoritem A torej vrne shranjene odgovore na prvih $r-1$ podpisovalnih poizvedb, kar med drugim pomeni, da za te poizvedbe ni potrebno previjanje nazaj. To pa je lahko potrebno pri r-ti podpisovalni poizvedbi, saj se drugi krog lahko zgodi po j_0-ti podpisovalni poizvedbi. Verjetnost previjanja je $1 - 1/q_H$, kjer je $1/q_H$ verjetnost, da je drugi krog r-te podpisovalne poizvedbe izveden na podlagi enake zgoščevalne poizvedbe, kot pri prvem pogonu.

Opomba 5.15. V dejstvo, da nobena od prvih $r-1$ podpisovalnih poizvedb v drugem pogonu ni temeljila na j_0-ti zgoščevalni poizvedbi, temelji na predpostavki, da ne dovoljujemo hkratnih podpisovalnih poizvedb, ki je standardna pri dokazih z lemo o razcepu.

Če previjanje pri r-ti podpisovalni poizvedbi ni potrebno, potem so pogoji pri j_0-ti zgoščevalni poizvedbi popolnoma enaki, kot pri prvem pogonu (enak vir naključnih bitov napadalca in enake zgoščevalne poizvedbe). Od tu lahko zaključimo, da bo tudi j_0-ta poizdveba imela enak vhod: $(\tilde{X}_0, m_0, S_0)$, odgovor pa bo seveda drugačen. Če tudi v tem pogonu poenostavljen napadalec $\mathcal{F}$ uspešno vrne ponaredek $(\tilde{X}_1, \tilde{y}_1)$

podpisa sporočila m_1 s strani podskupine S_1, ki temelji na poizvebi j_0, potem smo lahko prepričani, da velja $m_1 = m_0, S_1 = S_0$ in $\tilde{X}_1, = \tilde{X}_0$.

Ključno, enačbi za preverjanje podpisov potem lahko zapišemo kot

$$g^{\tilde{y}_0} \equiv \tilde{X}_0 \tilde{I}_{S_0}^{e_{j_0}} \pmod{p},$$
$$g^{\tilde{y}_1} \equiv \tilde{X}_1 \tilde{I}_{S_1}^{e'_{j_0}} \equiv \tilde{X}_0 \tilde{I}_{S_0}^{e'_{j_0}} \pmod{p}.$$

Opazimo, da lahko izrazimo $\tilde{X}_0$ na dva načina, ki ju enačimo:

$$\tilde{X}_0 \equiv g^{\tilde{y}_0} (\tilde{I}_{S_0}^{e_{j_0}})^{-1} \equiv g^{\tilde{y}_1} (\tilde{I}_{S_0}^{e'_{j_0}})^{-1} \pmod{p}.$$

Če premečemo dobljeno enačbo, dobimo

$$\tilde{I}_{S_0}^{-e'_{j_0}} \tilde{I}_{S_0}^{e_{j_0}} \equiv g^{\tilde{y}_0} (g^{\tilde{y}_1})^{-1} \pmod{p},$$

kar nam omogoča izraziti $\tilde{I}_{S_0}$ kot

$$\tilde{I}_{S_0} \equiv g^{(\tilde{y}_0 - \tilde{y}_1)(e_{j_0} - e'_{j_0})^{-1}} \pmod{p}.$$

To torej pomeni, da A lahko izračuna diskretni logaritem $\tilde{I}_{S_0}$ kot $(\tilde{y}_0 - \tilde{y}_1)(e_{j_0} - e'_{j_0})^{-1}$. Še vedno pa nismo pri koncu, saj to ni diskretni logaritem, ki nas zanima. Če podpisnike iz S_0 označimo $S_0 = \{P_{i_1}, \ldots, P_{i_l}\}$, potem je

$$\tilde{I}_{S_0} = \prod_{j=1}^{l} I_{i_j},$$

kjer je vrednost, katere diskretni logaritem želimo izračunati, I_i, nujno ena izmed I_{i_j}, saj mora podpisnik P_i biti del skupine S_0, če želimo, da je ponaredek veljaven. A mora sedaj pridobiti diskretne logaritme vseh ostalih I_{i_j} in jih odšteti iz diskretnega logaritma $\tilde{I}_{S_0}$. Tu uporabimo dejstvo, da mora P_i med generiranjem ključev pridobiti in preveriti dokaze znanja brez razkritja znanja o diskretnih logaritmih vseh I_{i_j}, razen za $i_j = i$. Spomnimo se, da poenostavljen napadalec $\mathcal{F}$ deluje tako, da si izbere tarčo (podpisnika P_i) in se nato postavi v vlogo vseh ostalih podpisnikov. To pomeni, da mora dokaze znanja generirati prav napadalec $\mathcal{F}$ (ki torej pozna vrednosti s_{i_j}).

Ker so dokazi znanja brez razkritja znanja o diskretnih logaritmih samo Schnorrovi podpisi, lahko uporabimo lemo o razcepu 5.9, prek katere algoritem A lahko previje napadalca $\mathcal{F}$ in prejme diskretni logaritem s_{i_j}, katerega znanje dokazuje $\mathcal{F}$. Ker vsi podpisniki za dokaz uporabijo enak izziv $H_3(X_1, I_1, \ldots, X_L, I_L)$, se mora zgoditi samo en razcep, na podlagi katerega lahko algoritem A izračuna vse diskretne logaritme s_{i_j}.

Vprašanje, ki ostane, je časovna zahtevnost algoritma A. Standardne operacije podpisovanja so polinomske v varnostnem parametru k, zato so polinomske tudi za A. Prav tako so učinkoviti odgovori na zgoščevalne poizvedbe. Najbolj časovno zahteven del so previjanja, ki jih je lahko q_H za vsak podpis in generiranje ključev. Ker je poenostavljen napadalec $\mathcal{F}$ polinomski algoritem, lahko opravi največ polinomsko poizvedb, torej je q_H polinomsko število v k. Časovna zahtevnost algoritma A pa je proporcionalna q_H, torej tudi polinomska v k.

Verjetnost uspeha algoritma A temelji na verjetnosti uspeha poenostavljenega napadalca $\mathcal{F}$. Ta verjetnost je pomnožena še s faktorjem, ki je obratno sorazmeren polinomu v q_H. Ker smo začeli z nezanemarljivo verjetnostjo, in jo skalirali samo s polinomskimi faktorji, je verjetnost uspeha algoritma A še vedno nezanemarljiva. A je torej res polinomski algoritem, ki z nezanemarljivo verjetnostjo vrne odgovor na problem diskretnega algoritma. Ker predpostavljamo, da tak algoritem ne obstaja, je večstranski Schnorrov podpis varen. □

6 Sodobni pristopi k večstranskim podpisom

Razvoj večstranskih podpisov se je začel v osemdesetih letih prejšnjega stoletja, ko sta Nakamura in Itakura [8] predstavila prvi večstranski podpis. Do leta 2001 nihče ni uspel dokazati varnosti večstranskega podpisa, niti ni nihče predstavil formalnega modela za tovrstne podpise. Takrat so Micali, Ohta in Reyzin [20] predstavili večstranske podpise podskupine z odgovornostjo in pokazali njihovo varnost v modelu slučajnega oraklja.

Glavna pomanjkljivost njihovega modela je bila, da so podpisi zahtevali tri kroge komunikacije, kar se je za praktično uporabo izkazalo za nedopustno. Glavna težnja razvoja večstranskih podpisov je tako postala zmanjšanje števila potrebnih krogov komunikacije. V zadnjih letih se je pojavilo več shem, ki so obljubljale prav to [6, 16].

Cena zmanjšanja količine potrebne komunikacije pa je očitno bila opustitev lastnosti *odgovornosti* in *prilagodljivosti*. Novejši večstranski podpisi še vedno predstavljajo dokaz, da je neka skupina skupno podpisala sporočilo, vendar ne razkrivajo, kdo je bil član skupine. V zameno za to pa so sposobni vrniti kratke podpise, ki so v nekaterih primerih celo popolnoma enaki navadnim Schnorrovim podpisom [16].

V nadaljevanju poglavja si najprej ogledamo varnostne probleme večstranskih podpisov z dvema krogoma komunikacije, nato pa predstavimo MuSig2, moderno, varno in učinkovito shemo, ki je v uporabi danes.

6.1 Varnost večstranskih podpisov v splošnem

Tekom let je bilo predstavljenih več shem večstranskih podpisov, ki so za podpisovanje potrebovale le dva kroga komunikacije med podpisniki in so temeljile na Schnorrovem podpisu [1, 11, 22, 13]. Večina je vsebovala tudi dokaze varnosti, katerih podlaga je prav formalni model, ki smo ga predstavili v poglavju 5, dejanski dokazi pa so bili mnogo kompleksnejši in daljši od predstavljenega.

Glavni cilj modernih večstranskih Schnorrovih podpisov je eliminirati prvi krog komunikacije pri podpisovanju, torej izračuna in izmenjave zavez X_i. Pristopov k temu je bilo več, prav vsi pa so se izkazali za problematične, vsaj dokler ostajamo pri Schnorrovih podpisih in modelu slučajnega oraklja.

Uporabo katerekoli novejše sheme v praksi pa so preprečili Drijvers et al. [6], ko so pokazali, da so imeli prav vsi večstranski podpisi z dvema krogoma komunikacije varnostno luknjo. Dokazali niso samo, da so bili dokazi varnosti pomanjkljivi, temveč tudi, da predstavljene sheme niso varne (vsaj ne z uporabo poznanih metod dokazovanja). Še več, predstavili so tudi napad na te sheme, ki ima polinomsko ča-

sovno zahtevnost. Temeljna pomanjkljivost predstavljenih dokazov varnosti je bila, da so zanemarili možnost, da napadalec izvede več napadov hkrati. Če je napadalec previt, medtem ko izvaja še en napad, lahko to izkoristimo, da pridobimo še en podpis, ki temelji na isti zavezi, kar privede do razkritja zasebnega ključa poštenega podpisnika [6].

Seveda pa je dokaz nevarnosti zajel samo do tedaj poznane metode. Kasneje je raziskovalcem uspelo ustvariti podpis, ki zaobide težave prejšnjih shem in je (zaenkrat) dokazano varen. Podpis MuSig2 je predstavljal velik preboj, saj vrača podpise, ki so popolnoma enaki Schnorrovim. Med drugim to pomeni, da lahko njihovo veljavnost preverimo z uporabo Schnorrovee enačbe za preverjanje podpisov (3.1).

6.2 MuSig2

Zaradi popularnosti in enostavnosti Schnorrovih podpisov, se je pojavila težnja po razvoju večstranskega podpisa, ki je učinkovit, varen in vrača Schnorrove podpise. Obstoj takega podpisa pomeni takojšnjo priložnost za uporabo v vseh aplikacijah, ki temeljijo na Schnorrovih podpisih (recimo Bitcoin po nadgradnji Taproot [27]). Prvi podpis, ki je izpolnjeval vse te pogoje, je bil MuSig2, ki so ga leta 2020 predstavili Jonas, Ruffing in Seurin [16]. Pred pojavom MuSig2 so obstajali drugi večstranski podpisi, ki so vračali Schnorrove podpise, vendar so zahtevali tri kroge komunikacije, ali pa so vsaj preprečevali hkratne podpise. Eden izmed njih je tudi originalni MuSig [13], čigar nevarnost je pokazal Drijvers [6].

V tem razdelku predstavimo glavne komponente in ideje MuSig2. Podpis je dokazano varen v modelu slučajnega oraklja, vendar bomo dokaz varnosti izpustili, saj je zelo kompleksen. Osredotočili se bomo na razlike med predstavljenim večstranskim Schnorrovim podpisom in MuSig2.

Za začetek predstavimo glavne lastnosti MuSig2:

- Omogoča hkratno podpisovanje več sporočil s strani iste skupine podpisnikov. To je zelo uporabno v praksi, kjer lahko hkratno podpisovanje predstavlja velik prihranek časa.

- Podpira *agregacijo ključev* (angl. *key aggregation*), kar pomeni, da je mogoče seznam javnih ključev podpisnikov združiti v en sam javni ključ, ki je lahko potem uporabljen za preverjanje veljavnosti podpisov.

- Vrnjeni podpisi so popolnoma enaki Schnorrovim podpisom. To pomeni, da preverjevalec lahko ne ve, ali preverja Schnorrov podpis ali MuSig2. To lastnost omogoča sposobnost agregacije ključev, saj lahko preverjevalec uporabi agregirani javni ključ in enačbo za preverjanje Schnorrovih podpisov (3.1) za preverjanje veljavnosti podpisov MuSig2.

- Podpisovanje zahteva le dva kroga komunikacije med podpisniki. Še več, prvi krog podpisovanja se lahko zgodi preden je sploh znano sporočilo, kar še dodatno pohitri dejanski proces podpisa.

- Časovna zahtevnost podpisovanja in preverjanja je primerljiva z navadnim Schnorrovim podpisom.

- V primerjavi z opisanim večstranskim Schnorrovim podpisom, MuSig2 ne omogoča prilagodljivosti in odgovornosti. Torej za dano skupino podpisnikov se lahko podpišejo le vsi hkrati, njihov podpis priča o skupnem strinjanju, preverjrevalec ne more izvedeti, kdo je dejansko podpisoval. MuSig2 torej omogoča večjo zasebnost podpisnikov, kar je za nekatere aplikaicje zaželena lastnost.

6.2.1 Skupni parametri

Ta del podpisa je praktično enak Schnorrovemu večstranskemu podpisu. Podpisniki si izberejo varnostni parameter k. Na podlagi tega parametra morajo si izberejo grupo G reda q z generatorjem g, v kateri je problem diskretnega logaritma težak.

Poleg grupe podpisniki potrebujejo še tri zgoščevalne funkcije

$$H_{agg}, H_{non}, H_{sig} : \{0,1\}^* \to \mathbb{Z}_q,$$

kjer s pomočjo funkcije H_{agg} agregirajo javne, funkcija H_{non} služi za agregiranje enkratnih vrednosti (angl. *nonces*), ki jih potrebujejo tekom podpisa, funkcija H_{sig} pa služi za generiranje podpisov.

Krajše lahko zapišemo

$$((G, q, g), H_{agg}, H_{non}, H_{sig}) = \mathcal{P}(k).$$

6.2.2 Generiranje ključev

Na tem mestu se pojavi prva velika prednost MuSig2. Večstranski Schnorrov podpis je preprečil napad na generiranje ključev tako, da je od podpisnikov zahteval dokaze znanja brez razkritja znanja in konstrukcijo Merklovega drevesa. To je zahtevalo veliko komunikacje med podpisniki, celo linearno v njihovem številu. MuSig2 se s tem napadom sooči kasneje, med podpisovanjem. To omogoča bistveno enostavnejšo fazo generiranja ključev, ki je povsem enaka generiranju ključev pri navadnem Schnorrovem podpisu.

Vsak podpisnik P_i $(1 \leq i \leq L)$ si izbere naključen zasebni ključ $s_i \in \mathbb{Z}_q$ in izračuna svoj javni ključ

$$I_i = g^{s_i}.$$

Zapišemo lahko

$$(I_i, s_i) = \mathcal{G}(G, q, g).$$

Da je MuSig2 dejansko lahko uporabljen kot navaden Schnorrov podpis, je potrebna še *agregacija* javnih ključev. Naj $\{I_1, I_2, \ldots, I_L\}$ označuje množico vseh javnih ključev vseh podpisnikov v skupini P. Za agregacijo ključev najprej vsak podpisnik P_i z javnim ključem I_i $(1 \leq i \leq L)$ izračuna *koeficient agregacije* c_i:

$$c_i = H_{agg}(\{I_1, I_2, \ldots, I_L\} || I_i).$$

Agregirani javni ključ skupine P je nato

$$\tilde{I} = \prod_{i=1}^{L} I_i^{c_i}.$$

Agregacijo lahko izvede torej kdorkoli, ki pozna vse javne ključe podpisnikov. Če to delajo podpisniki sami, je tu potreben en krog komunikacije. Funkcijo agregacije označimo z $\mathcal{A}$, torej lahko zapišemo

$$\tilde{I} = \mathcal{A}(I_1, I_2, \ldots, I_L) = \prod_{i=1}^{L} I_i^{c_i}.$$

6.2.3 Podpisovanje

Ker vsa varnost MuSig2 sloni na podpisovanju, je ta del malo kompleksnejši, a vseeno zelo učinkovit. Kot omenjeno, podpisovanje poteka v dveh krogih. Vsak krog je sestavljen iz korakov podpisovanja in agregaicje, ki ju označimo $\mathcal{S}_i$ in $\mathcal{SA}_i$ (za $i = 1, 2$). Po komunikaciji je potreben še zadnji korak podpisovanja, ki ga označimo z $\mathcal{S}_3$.

Velika varnostna pomanjkljivost preteklih večstranskih podpisov z dvema krogoma komunikacije je bila izbira ene same naključne vrednosti tekom podpisovanja. MuSig2 uporablja več takšnih vrednosti, pri sami količini pa dopušča prosto izbiro. Število naključnih vrednosti bomo označili z $\nu \geq 2$.

- **Prvi krog**: Vsak podpisnik P_i $(1 \leq i \leq L)$ si izbere ν naključnih vrednosti $r_{1,i,j} \in \mathbb{Z}_q$ $(1 \leq j \leq \nu)$ in izračuna zaveze

  $$X_{1,i,j} = g^{r_{1,i,j}},$$

  ki jih pošlje podpisniku D. Ta torej od vsakega podpisnika P_i prejme množico zavez $\{X_{1,i,j} | 1 \leq j \leq \nu\}$, iz katerih za vsak j $(1 \leq j \leq \nu)$ izračuna

  $$X_j = \prod_{i=1}^{L} X_{1,i,j}.$$

  Končni rezultat prvega kroga so zaveze $\{X_j | 1 \leq j \leq \nu\}$.

  Zapišemo lahko torej

  $$\begin{aligned}(X_{1,i,1}, X_{1,i,2}, \ldots, X_{1,i,\nu}) &= \mathcal{S}_1(P_i),\\ (X_1, X_2, \ldots, X_\nu) &= \mathcal{SA}_1(X_{1,i,j} | 1 \leq i \leq L, 1 \leq j \leq \nu).\end{aligned}$$

  Uporabna lastnost tega koraka je, da lahko podpisniki izračunajo zaveze $X_{1,i,j}$, preden je sploh znano s kom bodo podpisovali in preden se izbere sporočilo m. Ko je enkrat skupina ustvarjena, lahko izvedejo še korak $\mathcal{SA}_1$, ponovno brez vednosti o sporočilu m.

- **Drugi krog**: Vsak podpisnik na te mestu potrebuje svoj zasebni ključ s_i, agregirani javni ključ $\tilde{I}$, ki ga lahko izračuna z algoritmom $\mathcal{A}$ in svoj koeficient agregacije c_i. Prav tako potrebujejo rezulat prvega kroga, torej zaveze $\{X_j | 1 \leq$

$j \leq \nu\}$ in sporočilo m. Vsak podpisnik P_i $(1 \leq i \leq L)$ izračuna

$$\begin{aligned} b &= H_{non}(\tilde{I}||X_1||X_2||\ldots||X_\nu||m), \\ \tilde{X} &= \prod_{j=1}^{\nu} X_j^{b^{j-1}}, \\ e &= H_{sig}(\tilde{I}||\tilde{X}||m), \\ y_i &= es_ic_i + \sum_{j=1}^{\nu} b^{j-1} r_{1,i,j} \bmod q. \end{aligned}$$

Krajše zapišemo

$$y_i = \mathcal{S}_2(P_i, s_i, \tilde{I}, X_1, X_2, \ldots, X_\nu, m).$$

Vrednosti y_i so poslane podpisniku D, ki izračuna vsoto

$$\tilde{y} = \sum_{i=1}^{L} y_i \bmod q = \mathcal{SA}_2(y_i | 1 \leq i \leq L).$$

- **Končni korak**: Če je podpisnik D, tisti, ki bo vrnil podpis, ima na tem mestu vse potrebne podatke. Enostavno vrne podpis

$$\sigma = (\tilde{X}, \tilde{y}) = \mathcal{S}_3(\tilde{X}, \tilde{y}).$$

sporočila m.

6.2.4 Preverjanje

Z uporabo agregacije javnih ključev, postane preverjanje podpisa MuSig2 zelo enostavno, saj je povsem enako preverjanju navadnega Schnorrovega podpisa. Preverjevalec prejme agregiran javni ključ $\tilde{I}_S$ skupine podpisnikov S, sporočilo m in podpis $\sigma = (\tilde{X}, \tilde{y})$. Izračuna izziv

$$e = H_{sig}(\tilde{I}_S||\tilde{X}||m)$$

in uporabi enačbo (3.1), ki jo tu zapišemo

$$g^{\tilde{y}} \stackrel{?}{=} \tilde{X}\tilde{I}_S^e,$$

da preveri veljavnost podpisa. Enačba res preveri veljavnost podpisa, saj lahko za veljaven podpis zapišemo

$$\begin{aligned} g^{\tilde{y}} &= g^{\sum_{i=1}^{L} y_i \bmod q} \\ &\stackrel{2.5}{=} g^{\sum_{i=1}^{L}(es_ic_i + \sum_{j=1}^{\nu} b^{j-1} r_{1,i,j})} \\ &= g^{\sum_{i=1}^{L} es_ic_i} \cdot g^{\sum_{i=1}^{L}(\sum_{j=1}^{\nu} b^{j-1} r_{1,i,j})} \\ &= \prod_{i=1}^{L} g^{es_ic_i} \cdot \prod_{j=1}^{\nu}(\prod_{i=1}^{L} g^{r_{1,i,j}})^{b^{j-1}} \\ &= \prod_{i=1}^{L} I_i^{ec_i} \cdot \prod_{j=1}^{\nu} X_j^{b^{j-1}} \\ &= \tilde{I}_S^e \cdot \tilde{X} \\ &= \tilde{X}\tilde{I}_S^e. \end{aligned}$$

6.2.5 Varnost

Varnost pri podpisu MuSig2 ponovno pomeni nemoč napadalca, da ponaredi katerikoli podpis, pri katerem sodeluje vsaj en nepokvarjen podpisnik. Varnost dokažemo na podoben način kot pri večstranskem Schnorrovem podpisu, le da je dokaz še bolj kompliciran. Na tem mestu bomo zato predstavili le osnovne ideje, na katerih dokaz sloni.

Podlago za varnost Schnorrovega podpisa predstavlja težavnost problema diskretnega logaritma. Pri MuSig2 se zanesemo na podoben problem, ki ga imenujemo *problem še enega diskretnega logaritma* (angl. *one-more discrete logarithm problem*). Zasnova je enaka kot pri PDL, imamo grupo z operacijo množenja, generator g in element grupe I, zanima pa nas diskretni logaritem $s = \log_g I$. Razlika pri problemu še enega diskretnega logaritma je, da imamo na voljo oraklja, ki nam lahko pove q rešitev problema diskretnega logaritma, mi pa želimo izračunati $q + 1$ problemov. Predpostavljamo, da ne obstaja polinomski naključnostni algoritem, ki to lahko stori z nezanemarljivo verjetnostjo.

Pod to predpostavko lahko v modelu slučajnega oraklja dokažemo, da je MuSig2 varen, če vsak podpisnik pri podpisovanju uporabi vsaj štiri naključne vrednosti ($\nu \geq 4$). Če dodatno predpostavimo še *model algebraičnih grup* (angl. *algebraic group model*), ki predpostavlja, da so napadalci algebraični, torej za vsak element grupe, ki ga izračunajo, podajo njegov algebraični opis (na podlagi do sedaj znanih elementov), potem lahko dokažemo, da je MuSig2 varen tudi za $\nu = 2$.

Jedro dokaza je ponovno lema o razcepu 5.9 v kombinaciji s previjanjem. Dokaz pa je še dodatno kompleksnejši, ker dokazuje tudi varnost v primeru hkratnih podpisov, česar osnovna lema o razcepu ne omogoča. V tem primeru varnost omogoča dejstvo, da podpisniki izberejo več naključnih vrednosti, ki jih združijo s pomočjo zgoščevalne funkcije H_{non}.

7 Učinkovitost Schnorrovega, večstranskega Schnorrovega in MuSig2 podpisa

Naravno vprašanje, ki se pojavi ob omembi večstranskega Schnorrovega podpisa podskupine z odgovornostjo ali pa MuSig2, je, kakšne prednosti ponujata pred uporabo več navadnih Schnorrovih podpisov. Najočitnejša prednost je, da večstranski podpis vrne en sam podpis, ki je primerljivo dolg s Schnorrovim, medtem ko uporaba individualnih podpisov pomeni, da mora preverjevalec preveriti toliko podpisov, kot je velika skupina. To prednost večstranskega podpisa je posebej pomembna v primerih, ko je preverjevalčev čas zelo omejen ali drag. Odličen primer je pri vseh aplikacijah, ki temeljijo na veriženju blokov, kjer preverjanje podpisov poteka »na verigi«, torej je za vsako operacijo potrebno plačati.

Seveda pa je potrebno omeniti tudi slabosti večstranskih podpisov v primerjavi z navadnimi. Najbolj očitna je, da morajo podpisniki med seboj komunicirati, kar lahko pomeni velike izgube časa. Uporabna vrednost večstranskih podpisov torej sloni na primerih, kjer je komunikacija enostavna, čas in računska moč preverjevalca pa izjemno dragocena (ponovno so odličen primer aplikacije, ki slonijo na veriženju blokov).

7.1 Empirična analiza

Za empirično potrditev zgornjih trditev smo naredili program, ki implementira vse podpise v nekoliko poenostavljenem okolju. Zanemarili smo čas komunikacije pri vseh verzijah podpisov, saj je to nezanesljiv podatek, ki je odvisen od okolja. Cilj primerjave je videti, kako se razlikuje čas izvedbe posameznih komponent podpisov pri različnem številu podpisnikov.

Zaradi zanemarjanja komunikaicje pričakujemo, da bodo časi podpisovanja in generiranja ključev pri večstranskih podpisih nekoliko krajši, kot v resnici, čas preverjanja pa bo odražal realno stanje.

Podpise smo implementirali v programskem jeziku Go, vsa izvorna koda je na voljo na https://github.com/timkalan/magisterij. Poskusi so bili izvedeni na računalniku z Apple M1 Pro procesorjem in 16 GB RAM-a. Vsak poskus smo ponovili šestkrat, da smo dobili stabilne rezultate, ki smo jih nato povprečili. Za dodatno nazornost, smo izračunali še standardne odklone vseh poskusov. Vsi grafi prikazujejo povprečne vrednosti skupaj s standardnim odklonom.

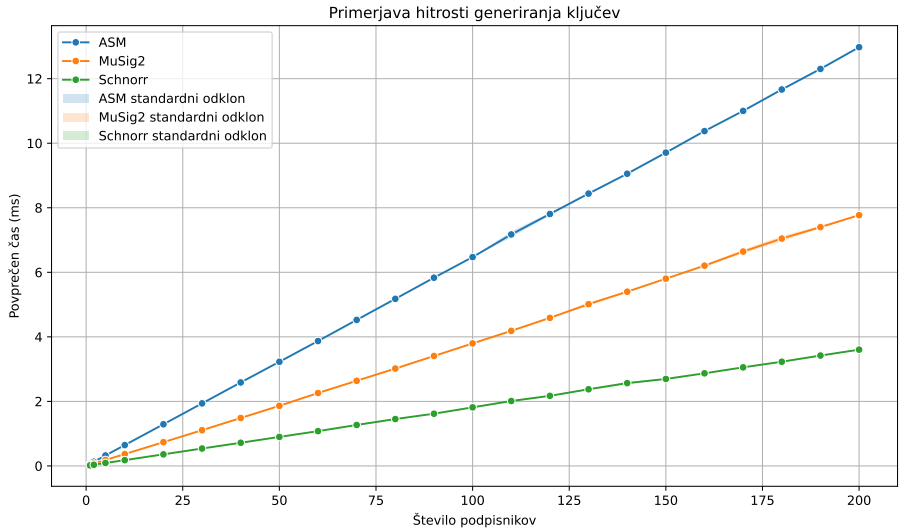
Poskusi so bili narejeni z $1, 2, 5, 10, 20, 30, \ldots, 200$ podpisniki. Velikost parametra q, ki določa grupo iz katere je generator, je bila 256 bitov. Za grupo smo izbrali multiplikativno grupo naravnih števil modulo p, kjer je $p - 1$ večkratnik parametra q. Za zgoščevalno funkcijo smo uporabili `SHA-256`, ki je bila izbrana zaradi velike razširjenosti.

7.1.1 Generiranje ključev

Generiranje ključev se pri vseh podpisih začne enako. Vsi podpisniki morajo izbrati naključne zasebne ključe in generirati javne ključe kot $X_i = g^{s_i} \pmod p$. Tu se generiranje ključev za navaden Schnorrov podpis in MuSig2 konča, pri večstranskem podpisu pa je potrebno še ustvariti dokaze znanja brez razkritja znanja, jih poslati vsem ostalim podpisnikom in preveriti njihovo veljavnost. To je seveda zelo veliko dodatnega dela, ki pa ga je za določeno skupino potrebno opraviti samo enkrat (kot je tudi potrebno samo enkrat generirati par ključev pri navadnem Schnorrovem podpisu).

Kot je razvidno iz slike 4, je čas generiranja ključev pri večstranskem Schnorrovem podpisu mnogo daljši, ne glede na količino podpisnikov. MuSig2 porabi enako časa kot Schnorrov podpis, vendar smo vključili še čas agregacije ključev, zato je potrebno še nekaj dodantega časa. Rast zahtevnosti v vseh primerih je linearna s številom podpisnikov, a je vseeno hitrejša pri večstranskih podpisu. V praksi je za večstranski Schnorrov podpis potrebno še več časa, saj smo tu zanemarili čas komunikacije, ki je v tem primeru linearen s številom podpisnikov. Ker pa se mora ta komunikacija zgoditi samo enkrat, jo tu zanemarimo (ključe skupina ustvari le ob začetku sodelovanja). Za predstavo razlike, pri 100 podpisnikih se za generiranje ključev pri navadnem podpisu porabita približno 2 milisekundi, pri MuSig2 6 milisekund, pri večstranskem Schnorrovem pa 6 milisekund.

Generiranje ključev predstavlja veliko prednost za uporabo navadnega Schnorrovega podpisa, vendar si moramo zapomniti, da je to postopek, ki ga mora skupina opraviti le enkrat, hkrati pa ne obremenjuje preverjevalca. Če smo torej v situaciji, kjer mora ista skupina podpisnikov velikokrat podpisati sporočila, je čas generiranje



Slika 4: Primerjava časov generiranja ključev pri Schnorrovem, večstranskem Schnorrovem in MuSig2 podpisu. Prikazan je skupen čas generiranja ključev za vse podpisnike.

ključev zanemarljiv, v primeru enkratnih skupin pa je morda smiselno razmišljati o uporabi navadnega Schnorrovga podpisa.

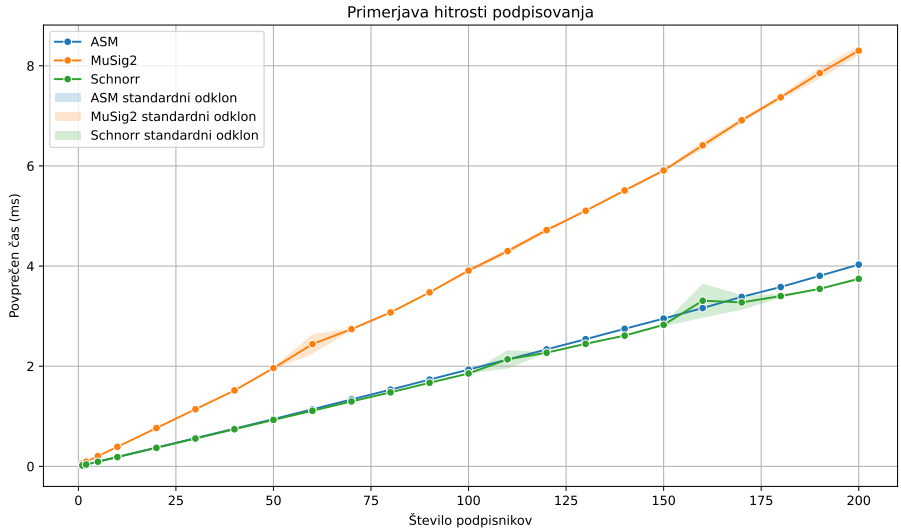
7.1.2 Podpisovanje

Pri podpisovanju največjo razliko med podpisi naredi potreba po komunikaciji med podpisniki pri večstranskih podpisih. Zahtevno je predvsem to, da so za ustvarjanje enega podpisa potrebni trije krogi komunikacije za večstranski Schnorrov podpis:

- izmenjava zavez X_i,
- izmenjava skupne zaveze $\tilde{X}$
- in izmenjava izziva e, in dva kroga pri MuSig2:
- izmenjava zavez $X_{1,i,j}$,
- izmenjava delnih podpisov y_i.

Če pa komunikacijo zanemarimo, pa je čas podpisovanja enak za Schnorrov in večstranski Schnorrov podpis, kot je razvidno tudi iz naše analize na sliki 5 (kjer smo zanemarili čas komunikacije). Vsa računska kompleksnost pri MuSig2 pa se zgodi ravno v času podpisovanja, zato v povprečju porabi dvakrat več časa kot ostala podpisa. Večstranski Schnorrov podpis je v tem eksperimentu dobil nepravičnp prednost pred MuSig2, saj je zanemarjena razlika v času komunikacije.

Graf torej odraža čas, ki ga porabi računalnik za računanje, zanemari pa čas komunikacije. Teoretično je čas za komunikacijo konstanten, če lahko podpisniki hkrati pošljajo in prejemajo več sporočil. V praksi pa čas zavisi od mnogih pogojev, kot so hitrost povezave, število podpisnikov in stabilnost povezave.



Slika 5: Primerjava časov podpisovanja pri Schnorrovem, večstranskem Schnorrovem in MuSig2 podpisu. Prikazan je skupen čas podpisovanja za vse podpisnike.

7.1.3 Preverjanje

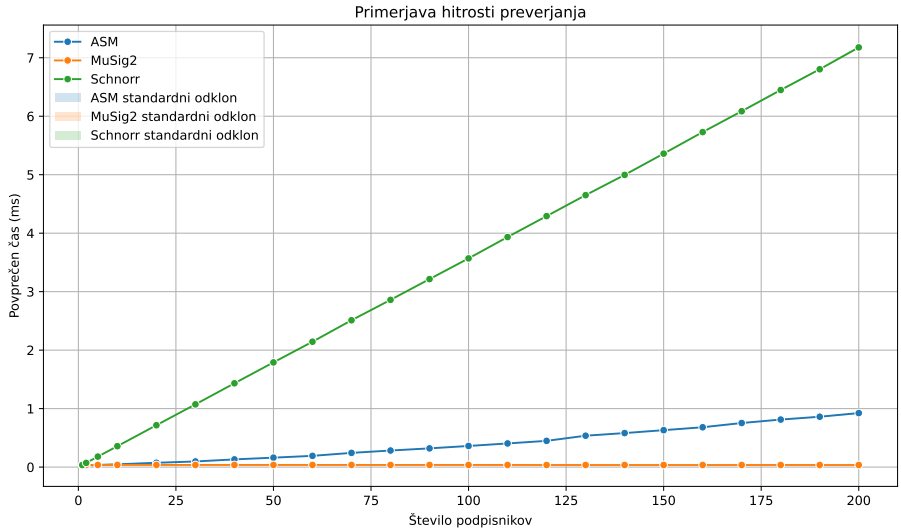
Ena največjih prednosti večstranskih podpisov je, da podpis vedno ostane enako dolg kot navaden podpis – ne glede na število podpisnikov. Prednost je sploh očitna pri MuSig2, saj mora preverjevalec preveriti samo en navaden Schnorrov podpis. Pri večstranskem Schnorrovem podpisu ima še dodano nalogo preverjanja ustreznosti Merklovih korenov, kar pa je relativno hitro in učinkovito.

Navaden Schnorrov podpis zahteva preverjanje vsakega podpisa posebej, kar pomeni, da efektivno dolžina podpisa raste linearno s številom podpisnikov. To pomeni, da tako raste tudi čas preverjanja. Prednost večstranskih podpisov lahko vidimo na sliki 6. Zahtevnost preverjanja navadnega Schnorrovega podpisa narašča mnogo hitreje, kot zahtevnost večstranskega. Kljub veliki prednosti večstranskega Schnorrovega podpisa, vidimo, da zahtevnost vseeno ni konstantna, temveč tudi linearno narašča. To je zato, ker implementacija podpisa vsakič, ko se podpis preveri, ponovno konstruira Merklovo drevo. V praksi to za zaporedne podpise enakih skupin ne bi bilo potrebno, saj bi preverjevalec enostavno lahko shranil Merklovo drevo, kar pomeni, da bi bilo preverjanje še toliko bolj učinkovito.

Tu se odlično prikaše prednost MuSig2, saj je čas preverjanja konstanten, ne glede na število podpisnikov. Kazen, ki jo plačamo za to v primerjavi z večstranskim Schnorrovim podpisom, je pomanjkanje odgovornosti, ki jo omogoča prav Merklovo drevo.

7.1.4 Celotni podpis

Če združimo vse operacije – torej generiranje ključev, podpisovanje in preverjanje – je enostranski podpis vseeno učinkovitejši, ker večstranska podpisa zahtevata veliko računanja za generiranje ključev oz. podpisovanje. To se jasno vidi na sliki 7. Seveda pa je to nekoliko nerealistična primerjava, saj nas redko zanima celoten čas, ki ga bodo porabili različni deležniki. Kljub temu nam da dobro predstavo o tem, kako



Slika 6: Primerjava časov preverjanja veljavnosti podpisov pri Schnorrovem, večstranskem Schnorrovem in MuSig2 podpisu. Prikazan je skupen čas preverjanja za vse podpisnike.

se čas podpisovanja različnih podpisov spreminja s številom podpisnikov in kako na ta čas vplivajo posamezne operacije podpisa.

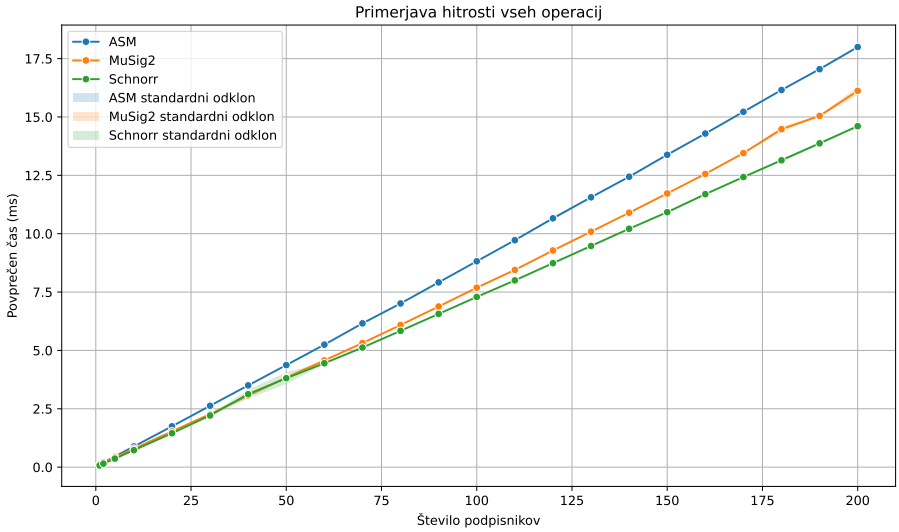
Za konec lahko analiziramo še situacijo, kjer se ista skupina podpisuje večkrat, torej lahko zanemarimo čas generiranja ključev. Slika 8 jasno prikaže prednost večstranskega Schnorrovega podpisa v tem primeru. Ta prednost pa je seveda spet varljiva, saj MuSig2 res porabi več časa za računanje, vendar prihrani eno tretjino komunikacije, kar je v realnem svetu glavni faktor, ki odloča o učinkovitosti.

7.2 Razprava

Na podlagi rezultatov vidimo, da so večstranski podpisi lahko zelo uporabno orodje, če so izpolnjeni določeni pogoji: omejiti želimo delo za preverjevalca in imamo veliko skupino ljudi, ki se večkrat skupaj podpisuje. Pri vseh vrstah podpisov je vredno omembe, da časovna zahtevnost narašča linearno s številom podpisnikov za vse operacije, z izjemo preverjanja pri MuSig2, ki ima konstantno časovno zahtevnost.

Kljub učinkovitosti pri preverjanju, ima večstranski Schnorrov podpis tudi nekatere pomanjkljivosti. Prva je kompleksnost generiranja ključev, ki zahteva interaktivno izmenjavo dokazov znanja in preverjanje Merklovega drevesa, kar je zamudno in težko optimizirati za zelo velike skupine. Druga težava je potreba po usklajeni komunikaciji pri podpisovanju. Ker so za vsak podpis potrebni trije komunikacijski krogi, je to lahko velik strošek v omrežjih z visoko zakasnitvijo ali nizko zanesljivostjo. Te težave vsaj delno odpravi MuSig2, kjer je generiranje ključev enako zahtevno kot pri navadnem Schnorrovem podopisu, podpisovanje pa poteka v dveh krogih komunikacije.

Izboljšali bi lahko tudi našo implementacijo: trenutno ne upoštevamo časa, ki se porabi za komunikacijo med podpisniki, kar je seveda lahko velik faktor. Tako smo se odločili, ker je glavna prednost večstranskih podpisov preverjanje, ki pa ne zahteva komunikacije. Prav tako bi lahko natančneje analizirali majhne spremembe



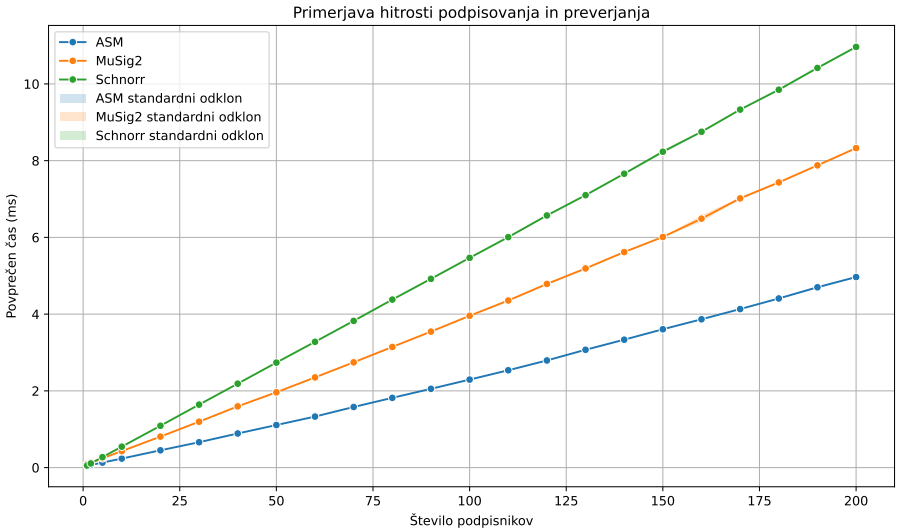
Slika 7: Primerjava časov celotnega postopka izvedbe podpisne sheme pri Schnorrovem, večstranskem Schnorrovem in MuSig2 podpisu. Prikazan je skupen čas celotnega postopka za vse podpisnike.

pri postopku, kot je na primer shranjevanje Merklovega drevesa med zaporednimi podpisi pri večstranskem Schnorrovem podpisu.

7.3 Sklepne misli

Podpis MuSig2 predstavlja pomemben korak v razvoju večstranskih podpisov. V primerih, ko želimo, da se celotna skupina dokazljivo strinja s sporočilom, a ne želimo razkriti članov, je MuSig2 odlična izbira. Poleg učinkovitosti in varnosti, je za praktično uporabo pomembno tudi, da vrača Schnorrove podpise.

Če pa želimo podpisno shemo, ki omogoča popolno prilagodljivost in odgovornost, pa je (vsaj zaenkrat) potrebno uporabiti tri kroge komunikacije in podpis kot je recimo večstranski Schnorrov podpis. Čeprav ni najbolj učinkovit, je predstavljal prvi korak k razvoju MuSig2, saj so njegovi avtorji prvi, ki so predstavili formalni model in dokazali varnost večstranskih podpisov.



Slika 8: Primerjava časov podpisovanja in preverjanja veljavnosti podpisov pri Schnorrovem, večstranskem Schnorrovem in MuSig2 podpisu. Prikazan je skupen čas podpisovanja in preverjanja za vse podpisnike.

Literatura

[1] A. Bagherzandi, J.-H. Cheon in S. Jarecki, *Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma*, v: Proceedings of the 15th ACM Conference on Computer and Communications Security, CCS '08, Association for Computing Machinery, Alexandria, Virginia, USA, 2008, str. 449–458, DOI: 10.1145/1455770.1455827, dostopno na https://doi.org/10.1145/1455770.1455827.

[2] D. Boneh in V. Shoup, *A graduate course in applied cryptography*, Stanford University, Stanford, 2023.

[3] C. P. Schnorr, *Efficient identification and signatures for smart cards*, v: Advances in Cryptology — CRYPTO' 89 Proceedings (ur. G. Brassard), Springer New York, New York, NY, 1990, str. 239–252.

[4] J.-J. Quisquater in dr. *How to explain zero-knowledge protocols to your children*, v: Advances in Cryptology — CRYPTO' 89 Proceedings (ur. G. Brassard), Springer New York, New York, NY, 1990, str. 628–631.

[5] D. Chaum in E. van Heyst, *Group signatures*, v: Advances in Cryptology — EUROCRYPT '91 (ur. D. W. Davies), Springer Berlin Heidelberg, Berlin, Heidelberg, 1991, str. 257–265.

[6] M. Drijvers in dr. *On the security of two-round multi-signatures*, v: 2019 IEEE Symposium on Security and Privacy (SP), 2019, str. 1084–1101, DOI: 10.1109/SP.2019.00050.

[7] T. Elgamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory **31**(4) (1985) 469–472, DOI: 10.1109/TIT.1985.1057074.

[8] K. Itakura in K. Nakamura, *A public-key cryptosystem suitable for digital multisignatures*, NEC Research & Development **71** (1983) 1–8.

[9] S. V. GmbH in E. M. Society, *Encyclopedia of Mathematics: Jensen inequality*, [ogled 25. 4. 2025], dostopno na http://encyclopediaofmath.org/index.php?title=Jensen_inequality&oldid=47465.

[10] J. Katz in Y. Lindell, *Introduction to modern cryptography, second edition*, 2nd, Chapman & Hall/CRC, 2014.

[11] C. Ma in dr. *Efficient discrete logarithm based multi-signature scheme in the plain public key model*, Designs, Codes and Cryptography **54**(2) (2010) 121–133, DOI: 10.1007/s10623-009-9313-z, dostopno na https://doi.org/10.1007/s10623-009-9313-z.

[12] D. Pointcheval in J. Stern, *Security proofs for signature schemes*, v: Advances in Cryptology — EUROCRYPT '96 (ur. U. Maurer), Springer Berlin Heidelberg, Berlin, Heidelberg, 1996, str. 387–398.

[13] G. Maxwell in dr. *Simple schnorr multi-signatures with applications to bitcoin*, Des. Codes Cryptography **87**(9) (2019) 2139–2164, DOI: 10.1007/s10623-019-00608-x, dostopno na https://doi.org/10.1007/s10623-019-00608-x.

[14] S. Micali, *Computationally sound proofs*, SIAM Journal on Computing **30**(4) (2000) 1253–1298, DOI: 10.1137/S0097539795284959, eprint: https://doi.org/10.1137/S0097539795284959, dostopno na https://doi.org/10.1137/S0097539795284959.

[15] S. Nakamoto, *Bitcoin: a peer-to-peer electronic cash system* (2009), dostopno na http://www.bitcoin.org/bitcoin.pdf.

[16] J. Nick, T. Ruffing in Y. Seurin, *MuSig2: Simple Two-Round Schnorr Multi-Signatures*, Cryptology ePrint Archive, Paper 2020/1261, 2020, DOI: 10.1007/978-3-030-84242-0_8, dostopno na https://eprint.iacr.org/2020/1261.

[17] Nikhil, P. Mortensen in Jesse, *Specifying the depth of a binary tree on the side*, [ogled 25. 10. 2024], dostopno na https://tex.stackexchange.com/questions/176513/specifying-the-depth-of-a-binary-tree-on-the-side.

[18] A. Fiat in A. Shamir, *How to prove yourself: practical solutions to identification and signature problems*, v: Advances in Cryptology — CRYPTO' 86 (ur. A. M. Odlyzko), Springer Berlin Heidelberg, Berlin, Heidelberg, 1987, str. 186–194.

[19] R. L. Rivest, A. Shamir in L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Commun. ACM **21**(2) (1978) 120–126, DOI: 10.1145/359340.359342, dostopno na https://doi.org/10.1145/359340.359342.

[20] S. Micali, K. Ohta in L. Reyzin, *Accountable-subgroup multisignatures*, v: Proceedings of the 8th ACM conference on Computer and Communications Security (ur. P. Samarati), ACM, Philadelphia, PA, USA, 2001, str. 245–254, DOI: 10.1145/501983.502017, dostopno na https://doi.org/10.1145/501983.502017.

[21] D. R. Stinson in M. B. Paterson, *Cryptography: theory and practice*, Textbooks in Mathematics, CRC Press, 2018.

[22] E. Syta in dr. *Keeping authorities "honest or bust"with decentralized witness cosigning*, v: 2016 IEEE Symposium on Security and Privacy (SP), 2016, str. 526–545, DOI: 10.1109/SP.2016.38.

[23] M. Tibouchi, *Attacks on schnorr signatures with biased nonces*, v: NTT Secure Platform Laboratories, ECC Workshop, 2017.

[24] D. Boneh, *Aggregate signatures*, v: Encyclopedia of Cryptography and Security (ur. H. C. A. van Tilborg in S. Jajodia), Springer US, Boston, MA, 2011, str. 27–27, DOI: 10.1007/978-1-4419-5906-5_139, dostopno na https://doi.org/10.1007/978-1-4419-5906-5_139.

[25] *United States Declaration of Independence*, [ogled 30. 10. 2024], dostopno na https://en.wikipedia.org/wiki/United_States_Declaration_of_Independence.

[26] D. Whitfield in H. Martin, *New directions in cryptography*, IEEE Transactions on Information Theory **22**(6) (1976) 644–654, DOI: 10.1109/TIT.1976.1055638.

[27] P. Wuille, J. Nick in A. Towns, *Taproot: SegWit version 1 spending rules*, Bitcoin Improvement Proposal 341, Accessed: 2025-07-18, 2020, dostopno na https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki.

[28] T. Yato in T. Seta, *Complexity and completeness of finding another solution and its application to puzzles*, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **86-A** (2003) 1052–1060, dostopno na https://api.semanticscholar.org/CorpusID:17154424.

[29] *Zero-knowledge proof*, [ogled 9. 9. 2024], dostopno na https://en.wikipedia.org/wiki/Zero-knowledge_proof.