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

Andrej Tolič

KRIPTOGRAFIJA E–VOLITEV

DIPLOMSKO DELO
NA INTERDISCIPLINARNEM UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Aleksandar Jurišić

Ljubljana, 2011

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00024/2011
Datum: 15.02.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko ter Fakulteta za matematiko in fiziko izdaja naslednjo nalogo:

Kandidat: **ANDREJ TOLIČ**

Naslov: **KRIPTOGRAFIJA E-VOLITEV**
CRYPTOGRAPHY OF E-VOTING

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Kandidat naj predstavi glavne kriptografske gradnike, ki se uporabljajo za učinkovito in varno implementacijo e-volitev: kriptosisteme z javnimi ključi, mešalne mreže, dokaze brez razkritja znanja, deljenje skrivnosti ipd. Navede naj tudi primer papirne sheme za e-volitve ter preuči odprtokodni sistem Helios.

Mentor:

prof. dr. Aleksandar Jurišić

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan Fakultete za računalništvo in informatiko:

prof. dr. Nikolaj Zimic

Dekan Fakultete za matematiko in fiziko:

prof. dr. Andrej Likar

UNIVERZA V LJUBLJANI • Fakulteta za matematiko in fiziko • 6

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Andrej Tolič,

z vpisno številko 63040299,

sem avtor diplomskega dela z naslovom:

KRIPTOGRAFIJA E-VOLITEV

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  prof. dr. Aleksandra Jurišića
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 05.09.2011 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju prof. dr. Aleksandru Jurišiću za številne nasvete in pomoč pri izdelavi diplomske naloge. Hvala Petru Nosetu za koristne pripombe. Hvala tudi vsem bližnjim za podporo v času študija.

Kazalo

Seznam uporabljenih kratic in simbolov

$\mathbb{Z}_n$	grupa celih števil po modulu n
$\mathbb{Z}_n^*$	multiplikativna grupa celih števil po modulu n
$\mathbb{P}$	množica praštevil
$\langle g \rangle$	grupa, ki jo generira g
$\varphi(n)$	Eulerjeva funkcija $\varphi(n)$ – moč množice števil, ki so tuja in ne večja od n
lcm	najmanjši skupni večkratnik (angl. least common multiple)
pk	javni ključ (angl. public key)
sk	zasebni ključ (angl. secret key oz. private key)
DLP	problem diskretnega logaritma (angl. discrete logarithm problem)
DCRA	odločitvena predpostavka o sestavljenih ostankih (angl. decisional composite residuosity assumption)
PKC	kriptosistem z javnim ključem (angl. public key cryptosystem)
ZKP	dokaz brez razkritja znanja (angl. zero-knowledge proof)

Povzetek

Diplomsko delo obravnava kriptografske gradnike in sheme, ki nam omogočajo izvedbo varnih e-volitev. Zaželeno je, da e-glasovnica ohrani tajnost, hkrati pa bi radi, da lahko tretja oseba preveri različne stopnje pravilnosti izvedbe. Kriptografija nam nudi potrebna orodja, s katerimi istočasno zadostimo tema, na videz nasprotujočima si, zahtevama. Zlasti so med omenjenimi orodji pomembni kriptosistemi z javnimi ključi, sheme za deljenje skrivnosti, dokazi brez razkritja znanja in mešalne mreže. Te gradnike predstavimo in obravnavamo v kontekstu e-volitev. V delu opišemo tudi dve shemi za kriptografske e-volitve. Prva je papirna shema, druga pa odprtokodni sistem za internetne volitve, Helios.

Ključne besede:

e-volitve, kriptografija, mešalne mreže, dokazi brez razkritja znanja, deljenje skrivnosti

Abstract

The diploma thesis deals with cryptographic building blocks and schemes that enable us to implement secure e-voting. It is desirable for an e-vote to remain secret, but at the same time we would like the possibility for a third party to verify various levels of correctness of execution. Cryptography provides us the necessary tools to simultaneously realize these two seemingly conflicting requirements. Public-key cryptosystems, secret sharing schemes, zero-knowledge proofs and mixnets are particularly important among aforementioned tools. These building blocks are presented and studied in the context of e-voting. In the thesis we also describe two schemes for cryptographic e-voting. The first one is a paper-based scheme, and the second is an open source system for internet voting, Helios.

Keywords:

e-voting, cryptography, mixnets, zero-knowledge proofs, secret sharing

Poglavje 1

Uvod

Z razmahom elektronskih naprav in računalnikov se je pojavila ideja uporabe teh naprav za potrebe volitev. Ko govorimo o elektronskih volitvah (e-volitve), v resnici govorimo o različnih načinih, kako za potrebe volitev uporabiti elektronske naprave.

Na začetku so se elektronske naprave uporabljale za zajem in štetje fizičnih glasovnic. Sem štejemo sisteme z luknjanimi karticami ter sisteme z optičnim zajemom glasovnic. Elektronske naprave pa lahko uporabimo tudi pri izpolnjevanju in hranjenju glasovnic. Tak primer so t.i. DRE (angl. direct recording by electronics) glasovalne naprave. Pri DRE napravah je glasovnica shranjena v elektronski obliki v pomnilniku naprave. Te naprave ponujajo prednosti, kot so glasovnice v več jezikih, opozarjanje volivca na nepravilno izpolnjeno glasovnico, lažje preštevanje glasov ipd.

Pri pravkar naštetih primerih določene naloge prevzamejo ali olajšajo elektronske naprave, še vedno pa mora volivec zaupati postopkom in napravam povezanim z izvedbo volitev. Pri zaupanju v postopke mislimo na zaupanje, da so postopki povezani s pripravo in izvedbo volitev izpeljani tako, da odkrivajo in onemogočajo zlorabe, zlasti zlorabe s strani oseb, ki so udeležene v te postopke. Pri zaupanju v naprave pa govorimo o potrebi po dokazilih o neoporečnosti elektronskih naprav. V mislih imamo pregled izvorne kode, pregled strojne opreme, pravilno hrambo, omejen dostop do naprav ipd.

Zgoraj navedene težave nas postavijo pred izziv, kako izvesti tajne in hkrati javno preverljive volitve, kjer volivcu ne bi bilo potrebno zaupati napravam in postopkom.

1.1 Kriptografske e-volitve

V tej diplomski nalogi se ukvarjamo z uporabo kriptografije za izvedbo e-volitev, ki bi volivcu omogočale **tajnost** (angl. secrecy) in **preverljivost** (angl. verifiability) brez zanašanja na pravilnost postopkov in neoporečnost naprav. Takim e-volitvam pravimo, da so **preverljive od začetka do konca** (angl. end-to-end auditable, kratica E2E). Takšen pristop se pogosto uporablja v računalništvu, kadar želimo kompleksnost sistema prenesti na višje plasti. V kontekstu e-volitev E2E pristop pomeni, da ne preverjamo več volilnih postopkov in naprav, ampak preverjamo rezultate teh postopkov in naprav (ne nujno samo končne rezultate). Takšen pristop omogoča tudi izvedbo **oddaljenih e-volitev** (angl. remote e-voting) ali, kot jih nekateri imenujejo, internetnih volitev. Vendar, kot bomo kasneje videli, v sklopu E2E e-volitev poznamo sheme za oddaljene e-volitve, kot tudi sheme za papirne e-volitve, kjer volimo na voliščih, vendar so v izvedbo volitev vključene elektronske naprave in kriptografski protokoli. Ključno orodje, s katerim izvedemo E2E preverljive e-volitve, je torej kriptografija, zato bomo od tu dalje, namesto o E2E preverljivih e-volitvah, govorili o **kriptografskih shemah za e-volitve** ali kar kriptografskih e-volitvah. Ker je izvedba kriptografskih e-volitev zahtevno in obsežno opravilo, ni namen te diplomske naloge predstaviti vse podrobnosti takega sistema, ampak se osredotočimo na predstavitev ključnih kriptografskih gradnikov, ki nam omogočajo tajne in hkrati preverljive volitve.

Za začetek naštejmo ključne lastnosti, ki jih morajo imeti kriptografske sheme za e-volitve:

1. **Preverljivost volivca** (angl. voter verifiability) ali tudi **zagotovitev oddaje glasovnice** (angl. ballot casting assurance). Ta lastnost volivcu omogoča, da lahko sam preveri, da je bil njegov glas pravilno štet, brez zanašanja na pravilnost postopkov in neoporečnost naprav. Naprave in osebe, ki obdelujejo glasovnice ali pridejo v stik z njimi, bi jih lahko spremenile ali odstranile. Zato je potrebno volivcu omogočiti, da preveri, ali je bila njegova glasovnica pravilno oddana in zapisana. Lastnost lahko razdelimo na dve podlastnosti:

    - **Oddaj glasovnico, kot je bilo mišljeno** (angl. cast as intended). Tu zahtevamo, da sistem ustvari glasovnico z vsebino, kot jo je izbral volivec in je ne spremeni. Pri tem je najbolj problematično, da volivec odda zašifrirano glasovnico, nima pa možnosti poustvariti iste zašifrirane glasovnice, saj bi to olajšalo prisilo volivcev (o prisili več kasneje).

- **Zapiši glasovnico, kot je bila oddana** (angl. recorded as cast). Tu zahtevamo, da zašifrirane glasovnice prispejo na cilj (oglasno desko) uspešno in nespremenjeno.

Tehnike za zagotavljanje preverljivosti volivca bomo bolj podrobno obravnavali pri posameznih volilnih shemah v 5. in 6. poglavju.

2. **Splošna preverljivost** (angl. universal verifiability). Vsakdo mora imeti možnost preveriti, da je bilo štetje veljavno oddanih glasovnic izvedeno pravilno. To lastnost imenujemo tudi **štetje glasovnic, kot so bile zapisane** (angl. tallied as recorded). Dosežemo jo predvsem z uporabo *dokazov brez razkritja znanja*.

3. **Tajnost** zagotavlja volivcu, da nihče drug (tudi tisti, ki pride v stik z glasovnico) ne pozna vsebine njegove glasovnice oz. da, če je vsebina enkrat znana, se glasovnice ne da povezati z volivcem. Tukaj ločimo dva različna pristopa, ki sta povezana z načinom štetja glasov, in ju bomo opisali kasneje.

Kriptografija nam omogoča, da zagotovimo navidez nasprotujoči si lastnosti, tajnost in preverljivost. Pri tem so pomembna predvsem naslednja področja kriptografije:

- **Kriptografija javnih ključev** (angl. public key cryptography) omogoča tajnost volitev.
- **Deljenje skrivnosti** (angl. secret sharing) omogoča izvedbo pragovnih kriptosistemov (angl. threshold cryptosystem).
- **Dokazi brez razkritja znanja** (angl. zero-knowledge proofs) omogočajo preverljivost.

Več bo o teh področjih kriptografije napisanega v naslednjih poglavjih.

Pri kriptografskih e-volitvah bi, poleg pravkar omenjenih lastnosti, radi dosegli tudi **odpornost na prisilo** (angl. coercion resistance). Z odpornostjo na prisilo želimo preprečiti, da volivec nekomu drugemu dokaže, kako je volil – ali po lastni želji (prodaja glasov) ali pa pod prisilo. Tukaj je potrebno opozoriti, da oddaljene e-volitve same po sebi omogočajo prodajo glasov ali glasovanje pod prisilo, saj praktično ni moč nadzorovati dogajanja na oddaljeni lokaciji. Vseeno pa lahko uvedemo mehanizme, ki tudi pri oddaljenih e-volitvah zmanjšajo možnost glasovanja pod prisilo.

Digitalna oglasna deska (angl. digital bulletin board) je ključen element volilnih shem in ima vlogo komunikacijskega kanala za oddajanje sporočil (angl. broadcast channel). Podatki na njej so volivcem tipično dostopni preko spletne strani. Na njej lahko vidimo imena (ali neko drugo obliko identifikacije) volivcev in pripadajoče zašifrirane glasovnice, ki jih na oglasno desko oddajo volivci. Volilni uradniki na oglasni deski objavljajo delne in končne rezultate štetja skupaj s pripadajočimi dokazi pravilnega delovanja, ki jih lahko vsi preverijo. Odvisno od načina štetja so na oglasni deski lahko objavljene tudi odšifrirane glasovnice, vendar na tak način, da jih ni možno povezati s pripadajočimi volivci. Na oglasno desko lahko zapisujejo zgolj avtenticirani udeleženci, zapisana sporočila pa lahko vidi vsak. Oglasna deska mora biti implementirana tako, da podatkov na njej ni mogoče neavtorizirano spreminjati, brisati in dodajati. Z oglasno desko sta torej povezana *oddaja glasovnice* s strani volivca ter *štetje glasov in objava rezultatov skupaj z dokazi pravilnosti* s strani volilnih uradnikov.

Volivčevo potrdilo (angl. voter receipt). Pri lastnostih kriptografskih shem za e-volitve smo že omenili preverljivost volivca oz. zagotovitev oddaje glasovnice. Volivec pri kriptografskih e-volitvah s pomočjo računalnika pripravi in odda svojo glasovnico. Priprava glasovnice vključuje izpolnitev glasovnice s strani volivca in šifriranje glasovnice s strani računalnika. Pri tem naj opozorimo, da se šifriranje s strani računalnika lahko zgodi pred izpolnjevanjem, kot je to značilno za sheme s papirnimi glasovnicami. Računalnik nato glasovnico odda na oglasno desko, volivec pa ponavadi dobi neke vrste potrdilo, s katerim lahko preveri, da je njegova šifrirana glasovnica uspešno in nespremenjeno prispela na oglasno desko in da je računalnik pravilno zašifriral glasovnico. Vendar pa morajo biti kriptografske sheme za e-volitve zaradi odpornosti na prisilo **brez potrdil** (angl. receipt-free). To ne pomeni, kot bi lahko sklepali iz imena, da potrdil ne sme biti. Pomeni le, da volivec ne sme dobiti takega potrdila, s katerim lahko nekomu drugemu dokaže, kako je volil. Potrdila morajo biti torej taka, da samo volivcu omogočajo, da se prepriča, da je njegov glas pravilno štet. Primer potrdila, ki je odporno na prisilo, bomo videli pri shemi v 5. poglavju.

1.2 Vrste kriptografskih shem in štetje glasov

Način štetja glasov je ključna lastnost, po kateri ločimo kriptografske sheme za e-volitve. Večina avtorjev omenja dva načina. Prvi način uporablja homo-

morfizem določenih kriptosistemov z javnimi ključi. V tem primeru govorimo o shemah s **homomorfnim štetjem glasov** (angl. homomorphic-based tallying). Pri shemah s homomorfnim štetjem glasov uporabimo homomorfne lastnosti kriptosistema, da združimo šifrirane glasovnice v en tajnopis, ki ustreza skupnemu rezultatu. Tako nikoli ne odšifriramo posamezne glasovnice, ampak zgolj končne rezultate.

Drugi način temelji na anonimizaciji glasovnic. Anonimizacijo izvedemo z uporabo **mešalnih mrež** (angl. mixnet). Pri shemah z mešalnimi mrežami posamezne šifrirane glasovnice na kriptografsko varen način permutiramo (premešamo) in ponovno šifriramo, ter na koncu odšifriramo, tako da ni mogoče povezati odšifrirane glasovnice s pripadajočo šifrirano glasovnico. To je pomembno, saj je šifrirana glasovnica na oglasni deski povezana z identiteto volivca.

Nekateri avtorji omenjajo tudi tretjo vrsto shem, ki uporablja slepe digitalne podpise, vendar naj ta vrsta shem ne bi bila popularna, ker ne omogoča splošne preverljivosti [1, str. 65].

Več podrobnosti obeh načinov bo predstavljenih v naslednjih poglavjih. Obema načinoma pa je skupna uporaba **verjetnostnega pragovnega šifriranja z javnimi ključi** (angl. randomized threshold public-key encryption). Glasovnice na oglasni deski so zašifrirane z javnim ključem, ki ga objavijo volilni uradniki. Da bi se preprečile zlorabe, konkretno, odšifriranje posamezne glasovnice na oglasni deski, kriptografske e-volitve uporabljajo verjetnostno pragovno šifriranje z javnim ključem. Pri takem šifriranju se pripadajoči zasebni ključ razdeli med več avtoritet po principu **deljenja skrivnosti**. Za verjetnostno šifriranje z javnimi ključi pa uporabimo kriptosistem, ki ima lastnost, da pri večkratnem šifriranju istega **čistopisa** (angl. plaintext) vrne različne **tajnopise** (angl. ciphertext). To lastnost ima npr. ElGamalov kriptosistem. Lastnost je za kriptografske e-volitve (pa tudi sicer) izredno pomembna, saj šifriramo glasovnice, ki imajo zelo omejen prostor možnih čistopisov. V najpreprostejših primerih (referendum) imamo samo dva možna čistopisa, DA ali NE. Zato je pomembno, da je vsak tajnopis glasovnice drugačen od ostalih, vsaj z izjemno veliko verjetnostjo.

1.3 Organizacija diplomske naloge

V 2. poglavju predstavimo kriptosisteme z javnimi ključi, si ogledamo njihove homomorfne lastnosti in predstavimo izvedbo pragovnih PKC z uporabo deljenja skrivnosti. Dokazi brez razkritja znanja, ki imajo pomembno vlogo pri

kriptografskih e-volitvah, so predstavljeni v 3. poglavju, kjer predstavimo tudi osnove kriptografskih volilnih shem. Mešalne mreže so ključni gradnik shem, ki ne uporabljajo homomorfnega načina štetja glasov. Zaradi tega in pa dolgoletnega razvoja ter številnih tehnik, ki so se pojavile, namenimo 4. poglavje predstavitvi mešalnih mrež. V 5. poglavju predstavimo kriptografsko shemo za e-volitve Scratch & Vote, ki uporablja papirne glasovnice in je namenjena glasovanju na voliščih. Helios, odprtokodni sistem za internetne volitve, je predstavljen v 6. poglavju. Nalogo zaključimo v 7. poglavju.

Poglavje 2

Kriptografija javnih ključev

Kriptografija javnih ključev je glavni temelj kriptografskih e-volitev. V tem poglavju predstavimo kriptosisteme z javnimi ključi (angl. public key cryptosystem, kratica PKC), njihovo varnost ter homomorfizem nekaterih PKC. Homomorfizem je pomemben tako za sheme s homomorfnim štetjem glasov, kjer omogoča štetje glasov brez odšifriranja posameznih glasovnic, kot tudi za sheme na osnovi mešalnih mrež, kjer omogoča eno od ključnih operacij, ponovno šifriranje tajnopisov glasovnic. Na koncu poglavja predstavimo pragovne PKC, ki onemogočajo, da bi nek volilni uradnik (ali majhna skupina le-teh) lahko odšifriral posamezno glasovnico.

2.1 Osnovne definicije in notacija

Za začetek definirajmo **zanemarljivo funkcijo** (angl. negligible function). Naj bodo κ, κ_0 in c naravna števila. Funkcija $\varepsilon(\kappa) : \mathbb{N} \to \mathbb{R}$ je zanemarljiva, če za vsako konstanto c obstaja konstanta κ_0, da je

$$\varepsilon(\kappa) < \frac{1}{\kappa^c}$$

za vsak $\kappa > \kappa_0$.

Naj bo M množica, porazdelitev ali verjetnostni proces (npr. verjetnostni algoritem). Zapis $x \in_R M$ pomeni, da naključno izberemo element x iz M, pri čemer upoštevamo, da gre za enakomerno porazdelitev (verjetnosti, da izberemo določen element, so med seboj enake).

Naj bo $\mathcal{G}$ verjetnostni algoritem za generiranje naključnih bitov. Zapis $x \in_R \mathcal{G}(1^\kappa)$ pomeni, da spremenljivki x priredimo niz naključnih bitov dolžine κ. Namesto x imamo lahko tudi par spremenljivk (x, y), v tem primeru obema

spremenljivkama priredimo različna niza naključnih bitov dolžine κ.

Carmichaelova funkcija [8] za vhod prejme naravno število n in vrne najmanjše naravno število m, tako da je $a^m \equiv 1 \pmod{n}$, za vsako celo število a, ki je tuje in manjše od n. Vrednost Carmichaelove funkcije za n označimo z $\lambda(n)$. Naj bo $n = \prod_{i=1}^{k} p_i^{\alpha_i}$. Carmichaelovo funkcijo računamo po formuli:

$$\lambda(n) = \operatorname{lcm}\big[(p_1 - 1)p_1^{\alpha_1 - 1}, \ldots, (p_k - 1)p_k^{\alpha_k - 1}\big].$$

Lahko pa jo definiramo tudi rekurzivno:

$$\lambda(n) = \operatorname{lcm}(\lambda(p_1^{\alpha_1}), \ldots, \lambda(p_k^{\alpha_k}))$$

pri čemer je $\lambda(p^i) = p^{i-1}(p-1)$ za $p \geq 3$ ali $i \leq 2$. Za $p = 2$ in $i \geq 3$ je $\lambda(2^i) = 2^{i-2}$.

DCRA predpostavlja, da je pri danem sestavljenem številu n in celem številu x, težko ugotoviti, ali obstaja tak y, da je $x \equiv y^n \pmod{n^2}$. Z drugimi besedami, zanima nas, ali je x n-ti ostanek[1] po modulu n^2.

2.2 Kriptografija javnih ključev

Kriptografijo javnih ključev sta javnosti prvič predstavila Diffie in Hellman leta 1976. Osnovna ideja je, da ločimo funkcijo zaklepanja in odklepanja, t.j. šifriranje in odšifriranje. Namesto da bi uporabili isti ključ za obe operaciji, uporabimo par ključev. Preprost opis je sledeč. Anita generira svoj par ključev. **Zasebni ključ** (angl. private key) sk obdrži zase, **javni ključ** (angl. public key) pk pa objavi. Kdorkoli, ki želi poslati Aniti sporočilo, lahko zašifrira čistopis m v tajnopis c z uporabo Anitinega javnega ključa pk. Tako zašifriran tajnopis c lahko odšifrira samo Anita z uporabo svojega zasebnega ključa sk. Zapišimo še formalno definicijo.

Definicija 2.2.1 (**Kriptosistem z javnimi ključi**). *Kriptosistem z javnimi ključi PKC je množica treh algoritmov $\mathcal{G}$, $\mathcal{E}$, $\mathcal{D}$, pri čemer sta prva dva algoritma verjetnostna, zadnji pa je deterministično. Za te algoritme, glede na dani varnostni parameter κ, definiramo naslednje tri operacije:*

[1] n-ti ostanek je posplošitev kvadratnega ostanka. Pravimo, da je x n-ti ostanek po modulu m, če obstaja tak y, da je $x \equiv y^n \pmod{m}$.

- **Generiranje para ključev**. *Naj bo $\mathcal{G}$ algoritem za generiranje nizov naključnih bitov. Par javnih in zasebnih ključev (pk, sk) se generira z uporabo algoritma za generiranje nizov naključnih bitov $\mathcal{G}$:*

$$(pk, sk) \in_R \mathcal{G}(1^\kappa).$$

  Odvisno od aplikacije lahko uporabnik sam s pomočjo programske ali strojne opreme generira svoj par ključev, lahko pa to za uporabnika naredi tretja oseba, vendar mora v tem primeru postopek tretje osebe zagotavljati tajnost uporabnikovega zasebnega ključa.

- **Šifriranje (angl. encryption)**. *Naj bo pk javni ključ uporabljen za šifriranje, M_{pk} pripadajoči prostor čistopisov, C_{pk} pripadajoči prostor tajnopisov, R_{pk} pripadajoči prostor naključnih vrednosti ter $\mathcal{E}$ šifrirni algoritem. Čistopis $m \in \mathsf{M}_{pk}$ šifriramo v tajnopis $c \in \mathsf{C}_{pk}$, ponavadi z uporabo naključne vrednosti $r \in \mathsf{R}_{pk}$:*

$$c = \mathcal{E}_{pk}(m; r).$$

- **Odšifriranje (angl. decryption)**. *Naj bo sk zasebni ključ, ki pripada javnemu ključu pk, M_{pk} pripadajoči prostor čistopisov, C_{pk} pripadajoči prostor tajnopisov ter $\mathcal{D}$ odšifrirni algoritem. Tajnopis $c \in \mathsf{C}_{pk}$ odšifriramo v čistopis $m \in \mathsf{M}_{pk}$. Odšifriranje je, za razliko od šifriranja, deterministično. Dani tajnopis c se pri danem zasebnem ključu sk vedno odšifrira v enak čistopis m:*

$$m = \mathcal{D}_{sk}(c).$$

Varnost kriptosistemov z javnimi ključi

Preden opišemo primere PKC, si oglejmo nekaj varnostnih pojmov. Neformalno pravimo, da je PKC **semantično varen**, kadar, pri danem tajnopisu c in javnem ključu pk, nasprotnik ne more dognati nobene lastnosti o pripadajočem čistopisu m. Semantično varnost sta leta 1982 definirala Goldwasser in Micali [1, str. 44].

Predstavimo igro med izzivalcem in nasprotnikom, kjer je nasprotnik polinomsko časovno omejen algoritem:

1. Izzivalec generira par ključev $(\mathrm{pk}, \mathrm{sk})$, glede na varnostni parameter κ (npr. dolžina ključa v bitih) in objavi pk.
2. Nasprotnik lahko izzvede poljubno (upoštevajoč, da je nasprotnik polinomsko omejen) število šifriranj in ostalih izračunov.
3. Nasprotnik izbere 2 različna čistopisa m_0 in m_1 ter ju pošlje izzivalcu.
4. Izzivalec naključno izbere bit b, zašifrira čistopis m_b v tajnopis c z uporabo javnega ključa pk in pošlje c nasprotniku.
5. Nasprotnik spet lahko izvede poljubno število šifriranj in ostalih izračunov.
6. Na koncu nasprotnik vrne svojo napoved za b.

Izzivalec torej naključno izbere enega od dveh čistopisov, ki mu jih pošlje nasprotnik, in ga zašifrira. Tajnopis nato pošlje nasprotniku, ta pa mora ugotoviti (razločiti), kateri, od dveh možnih, je pripadajoči čistopis.

PKC je **IND-CPA varen**, kadar ima nasprotnik v zgoraj opisani igri zanemarljivo prednost v primerjavi z nekom, ki naključno ugiba bit b. Z drugimi besedami, verjetnost, da nasprotnik zmaga zgornjo igro, ne sme biti večja od $\frac{1}{2} + \varepsilon(\kappa)$, kjer je $\varepsilon(\cdot)$ zanemarljiva funkcija, ki smo jo definirali na začetku poglavja. Kratica IND-CPA izhaja iz angleškega izraza za **nerazločljivost pri napadu z izbranim čistopisom** (angl.indistinguishability under chosen plaintext attack). Ime izhaja iz dejstva, da nasprotnik lahko sam šifrira poljubne čistopise.

Za **nerazločljivost pri napadu z izbranim tajnopisom** bomo uporabljali kratico IND-CCA, ki izhaja iz angl. imena indistinguishability under chosen ciphertext attack. IND-CCA varnost je po definiciji podobna IND-CPA, le da ima pri IND-CCA nasprotnik dostop do **odšifrirnega oraklja** (angl. decryption oracle). Nasprotnik lahko odšifrirnemu oraklju pošlje poljuben tajnopis, orakelj pa mu vrne pripadajoči čistopis glede na javni ključ pk.

Če gre pri IND-CCA za *prilagodljiv* (angl. adaptive) napad, bomo uporabljali kratico IND-CCA2. Pri IND-CCA ima nasprotnik možnost uporabljati odšifrirnega oraklja, dokler ne prejme izzivalnega tajnopisa, pri prilagodljivi različici IND-CCA2 pa ima dostop tudi po tem, ko že prejme izzivalni tajnopis, le da takrat ne sme oraklju poslati izzivalnega tajnopisa, saj bi bila takšna igra trivialna.

Opišimo IND-CCA ter IND-CCA2 varnosti kot igro med izzivalcem in nasprotnikom skupaj z razlikami med njima:

1. Izzivalec generira par ključev $(\mathrm{pk}, \mathrm{sk})$, glede na varnostni parameter κ (npr. dolžina ključa v bitih) in objavi pk.

2. Nasprotnik lahko izzvede poljubno (upoštevajoč, da je nasprotnik polinomsko omejen) število šifriranj in ostalih izračunov, ter pošilja poljubne tajnopise odšifrirnemu oraklju.

3. Nasprotnik izbere 2 različna čistopisa m_0 in m_1 ter ju pošlje izzivalcu.

4. Izzivalec naključno izbere bit b, zašifrira m_b z uporabo pk v c in pošlje c nasprotniku.

5. Nasprotnik spet lahko izvede poljubno število šifriranj in ostalih izračunov. Poleg tega:

   - Pri IND-CCA (neprilagodljiva različica) ne sme več pošiljati tajnopisov odšifrirnemu oraklju.
   - Pri IND-CCA2 (prilagodljiva različica) lahko pošilja poljubno število tajnopisov odšifrirnemu oraklju, razen tajnopisa c.

6. Na koncu nasprotnik vrne svojo napoved za b.

Ker v 2. in 5. točki igre lahko napadalec odšifira poljubne tajnopise, pri IND-CCA (IND-CCA2) varnosti, za razliko od IND-CPA, govorimo o napadu z izbranim tajnopisom.

IND-CCA2 je močnejša varnost od IND-CCA, slednja pa je močnejša od IND-CPA. Intuitivna razlaga je sledeča. IND-CCA varnost je močnejša od IND-CPA, saj ima nasprotnik pri IND-CCA igri na voljo vse, kar ima na voljo pri IND-CPA, poleg tega pa ima dostop do odšifrirnega oraklja. S tem lahko pridobi čistopise poljubnih tajnopisov in poskuša nekako uporabiti to dodatno znanje.

Podobno je pri razmerju med IND-CCA2 in IND-CCA varnostjo. Nasprotnik ima pri IND-CCA2 igri na voljo vse, kar ima na voljo pri IND-CCA, poleg tega pa ima dostop do odšifrirnega oraklja po tem, ko je že videl izzivalni tajnopis. To mu omogoča, da konstruira tajnopise, ki so na nek način povezani z izzivalnim tajnopisom. Nato za take tajnopise od oraklja pridobi ustrezne čistopise in poskuša uporabiti to dodatno znanje.

Enako kot prej je PKC IND-CCA oz. IND-CCA2 varen, kadar verjetnost, da nasprotnik zmaga zgornjo igro, ni večja od $\frac{1}{2} + \varepsilon(\kappa)$.

2.2.1 Primeri kriptosistemov z javnimi ključi

V nadaljevanju na kratko opišemo nekaj primerov PKC, posebej takih, ki so zanimivi za uporabo pri e-volitvah.

Kriptosistem RSA (Rivest-Shamir-Adleman) [27] je prvi in morda najbolj znan PKC. Definiramo ga takole:

- **Generiranje para ključev**. Izberemo dve veliki praštevili p in q in izračunamo modul $n = pq$. Nato izberemo tak šifrirni eksponent e, da $e \nmid \varphi(n)$. Sledi izračun takega odšifrirnega eksponenta d, da je $ed \equiv 1 \pmod{\varphi(n)}$. Ključa sta potem:
  - $\mathrm{pk} = (n, e)$,
  - $\mathrm{sk} = d$.
- **Šifriranje**. Naj bo $\mathbb{Z}_n$ prostor čistopisov. Čistopis $m \in \mathbb{Z}_n$ šifriramo z uporabo javnega ključa pk in šifrirnega algoritma $\mathcal{E}$ po naslednji formuli:
$$c = \mathcal{E}_{\mathrm{pk}}(m) = m^e \bmod n.$$
- **Odšifriranje**. Naj bo $\mathbb{Z}_n$ prostor tajnopisov. Tajnopis $c \in \mathbb{Z}_n$ odšifriramo z uporabo zasebnega ključa sk in odšifrirnega algoritma $\mathcal{D}$ po naslednji formuli:
$$m = \mathcal{D}_{\mathrm{sk}}(c) = c^d \bmod n.$$

Varnost zasebnega ključa temelji na problemu faktorizacije velikih števil, varnost tajnopisov pa na RSA problemu. Le-ta zahteva, da izvedemo odšifriranje brez poznavanja odšifrirnega eksponenta, torej da izračunamo e-ti koren tajnopisa po modulu n, kjer je e javni ključ. Kriptosistem RSA v osnovni obliki, kot je opisan zgoraj, ni IND-CPA varen, saj je šifriranje deterministično in lahko nasprotnik preprosto šifrira m_0 in m_1 ter nato pogleda, kateri tajnopis se ujema z izzivalnim tajnopisom. Zato v praksi RSA uporabljamo z dopolnilno shemo (angl. padding scheme) OAEP. V tem primeru formule ostanejo enake, le da namesto m šifriramo $m||\mathrm{OAEP}(m)$, pri čemer $\mathrm{OAEP}(m)$ vsebuje naključnost. Kriptosistem RSA-OAEP pa je IND-CPA varen.

ElGamalov kriptosistem [13] iz leta 1984 definiramo takole:

- **Generiranje para ključev**. Izberemo veliko praštevilo p in poiščemo generator g grupe $\mathbb{Z}_p^*$. Nato naključno izberemo $x \in \mathbb{Z}_{p-1}$. Ključa sta potem:

  - $\text{pk} = (p, g, y)$, kjer je $y = g^x \bmod p$,
  - $\text{sk} = x$.

- **Šifriranje**. Naj bo $\mathbb{Z}_p^*$ prostor čistopisov. Čistopis $m \in \mathbb{Z}_p^*$ šifriramo z uporabo javnega ključa pk in šifrirnega algoritma $\mathcal{E}$ po naslednji formuli:

$$c = (\alpha, \beta) = \mathcal{E}_{\text{pk}}(m) = (g^r, m \cdot y^r), \quad r \in_R \mathbb{Z}_{p-1}.$$

- **Odšifriranje**. Naj bo $\mathbb{Z}_p^* \times \mathbb{Z}_p^*$ prostor tajnopisov. Tajnopis $c = (\alpha, \beta) \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ odšifriramo z uporabo zasebnega ključa sk in odšifrirnega algoritma $\mathcal{D}$ po naslednji formuli:

$$m = \mathcal{D}_{\text{sk}}(c) = \beta \cdot \alpha^{-x} \bmod p.$$

Zasebnost tajnopisa temelji na **računskem Diffie-Hellmanovem problemu** (angl. computational Diffie-Hellman, kratica CDH). CDH predpostavlja, da je v ciklični grupi z generatorjem g težko izračunati g^{ab} pri podanih g^a in g^b. Semantična varnost tajnopisov pa temelji na **odločitvenem Diffie-Hellmanovem problemu** (angl. decisional Diffie-Hellman, kratica DDH). DDH predpostavlja, da je v ciklični grupi z generatorjem g, v kateri je problem diskretnega logaritma težak, težko ločiti naključno trojico (g^a, g^b, g^c) od trojice (g^a, g^b, g^{ab}).

Kriptosistem, kot je predstavljen tukaj (in v originalnem članku) ni semantično varen v celotni $\mathbb{Z}_p^*$. Zato ga v praksi uporabimo tako, da izberemo tak p, da ima $p - 1$ velik prafaktor q. Potem izberemo generator $g \in \mathbb{Z}_p^*$, da je $|\langle g \rangle| = q$. Zasebni ključ in naključne vrednosti sedaj izbiramo iz $\mathbb{Z}_q$ namesto $\mathbb{Z}_{p-1}$. Prostor čistopisov je potem $\langle g \rangle$, prostor tajnopisov pa $\langle g \rangle \times \langle g \rangle$.

ElGamalov kriptosistem ni IND-CCA2 varen. Če imamo tajnopis (c_1, c_2), ki pripada čistopisu m, lahko skonstruiramo tajnopis $(c_1, 2c_2)$, ki pripada čistopisu $2m$.

Paillierjev kriptosistem [23] iz leta 1999 definiramo takole:

- **Generiranje para ključev**. Izberemo veliki praštevili p in q ter izračunamo modul $n = pq$. Nato izračunamo Carmichaelovo funkcijo $\lambda(n)$ za n. Pri dani obliki n, je $\lambda = \lambda(n) = \text{lcm}(p-1, q-1)$. Poiščemo še tak generator g za grupo $\mathbb{Z}_{n^2}^*$, da je $g \equiv 1 \pmod{n}$. Ključa sta potem:
  - $\text{pk} = (n, g)$,
  - $\text{sk} = \lambda$.

- **Šifriranje**. Naj bo $\mathbb{Z}_n$ prostor čistopisov. Čistopis $m \in \mathbb{Z}_n$ šifriramo z uporabo javnega ključa pk in šifrirnega algoritma $\mathcal{E}$ po naslednji formuli:

$$c = \mathcal{E}_{\text{pk}}(m) = g^m \cdot r^n \bmod n^2, \quad r \in_R \mathbb{Z}_n^*.$$

- **Odšifriranje**. Naj bo $\mathbb{Z}_{n^2}^*$ prostor tajnopisov. Tajnopis $c \in \mathbb{Z}_{n^2}^*$ odšifriramo z uporabo zasebnega ključa sk in odšifrirnega algoritma $\mathcal{D}$ po naslednji formuli:

$$m = \mathcal{D}_{\text{sk}}(c) = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n,$$

kjer je $L(x) = \frac{x-1}{n}$. Poglejmo, če zgornja funkcija res odšifrira Paillierjev tajnopis.
Velja:

  - Red grupe $\mathbb{Z}_{n^2}^*$ je $\varphi(n^2) = n\varphi(n)$.
  - Za vsak $a \in \mathbb{Z}_{n^2}^*$ je $a^\lambda \equiv 1 \pmod{n}$ in $a^{n\lambda} \equiv 1 \pmod{n^2}$.

Vrednost g lahko zapišemo v obliki $g = nk + 1$ za neko celo število k. Potem je imenovalec v ulomku odšifrirne funkcije enak:

$$\begin{aligned} L(g^\lambda \bmod n^2) &= \frac{((1+nk)^\lambda \bmod n^2) - 1}{n} \\ &= \frac{(nk\lambda) \bmod n^2}{n} \\ &= k\lambda \bmod n^2. \end{aligned}$$

Opazimo, da se potenciranje spremeni v množenje, saj so vsi ostali členi iz $(1 + nk)^\lambda$ večkratniki n^2. Oglejmo si še števec v ulomku odšifrirne funkcije:

$$L(c^\lambda \bmod n^2) = L(g^{m\lambda} \cdot r^{n\lambda} \bmod n^2).$$

Ker je $r^{n\lambda} \equiv 1 \pmod{n^2}$, podobno kot zgoraj dobimo:

$$L(c^\lambda \bmod n^2) = mk\lambda \bmod n^2,$$

iz česar je razvidno, da z uporabo odšifrirne funkcije iz tajnopisa c res dobimo pripadajoči čistopis m. Če pri generiranju javnega ključa izberemo $g = n + 1$, potem je $L(g^\lambda \bmod n^2) = \lambda \bmod n^2$. To pomeni, da operacija odšifriranja vključuje eno modularno potenciranje in en račun inverza, saj je

$$m = \frac{(c^\lambda \bmod n^2) - 1}{n} \cdot \lambda^{-1} \bmod n.$$

Varnost zasebnega ključa λ temelji na problemu faktorizacije velikih števil.

Trditev 2.2.1. *Paillierjev kriptosistem je IND-CPA varen, če predpostavimo resničnost DCRA.*

Dokaz. Imejmo IND-CPA igro med izzivalcem in nasprotnikom (glej razdelek 2.2). Nasprotnik si je izbral čistopisa m_0 in m_1, izzivalec pa je naključno izbral enega izmed njiju in ga šifriral v tajnopis c. Nasprotnik mora potem ugotoviti, kateremu čistopisu pripada c. Opazimo, da je c tajnopis za m_0, kadar je $(c \cdot g^{-m_0} \bmod n^2)$ n-ti ostanek po modulu n^2. Če bi torej nasprotnik imel na voljo učinkovit algoritem za DCRA, potem Paillierjev kriptosistem ne bi bil IND-CPA varen. □

2.3 Homomorfni kriptosistemi z javnimi ključi

Nekateri PKC izkazujejo homomorfizem. To je lastnost, da je določena algebraična operacija nad dvema čistopisoma ekvivalentna določeni (lahko drugačni) algebraični operaciji nad pripadajočima tajnopisoma. Formalno zapišemo, da je PKC $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ **homomorfen** za binarni operaciji $(\oplus, \otimes)$, če za vsak par $(\mathrm{pk}, \mathrm{sk})$ in grupi $(\mathsf{M}_{\mathrm{pk}}, \oplus)$, $(\mathsf{C}_{\mathrm{pk}}, \otimes)$ velja:

$$\mathcal{D}_{\mathrm{sk}}(c_1 \otimes c_2) = \mathcal{D}_{\mathrm{sk}}(c_1) \oplus \mathcal{D}_{\mathrm{sk}}(c_2)$$

za $\forall (c_1, c_2) \in \mathsf{C}^2_{\mathrm{pk}}$.

Za potrebe kriptografskih e-volitev so posebej zanimivi kriptosistemi, ki izkazujejo **aditivni homomorfizem**. Pravimo, da kriptosistem izkazuje aditivni homomorfizem, kadar zmnožek dveh tajnopisov rezultira v tajnopisu, ki je vsota pripadajočih čistopisov.

2.3.1 Ponovno šifriranje

Posledica homomorfizma je zmožnost ponovnega šifriranja (angl. re-encryption), ki igra pomembno vlogo pri mešalnih mrežah. Ponovno šifriranje omogoča vsakomur, ki ima dostop do javnega ključa, da za dani tajnopis c ustvari nov tajnopis c', ki se odšifrira v enak čistopis kot c.

Za homomorfen kriptosistem smo zapisali, da je $(\mathsf{M}_{\mathrm{pk}}, \oplus)$ grupa, se pravi vsebuje element identitete. Naj bo identiteta čistopis m_0. Potem je

$$\forall m \in \mathsf{M}_{\mathrm{pk}} : m \oplus m_o = m.$$

Če sedaj upoštevamo homomorfne lastnosti kriptosistema, definiramo funkcijo **ponovno šifriranje**:

$$\mathcal{RE}_{\mathrm{pk}}(c; r) = c \otimes \mathcal{E}_{\mathrm{pk}}(m_o; r).$$

Če je $\mathcal{D}_{\mathrm{sk}}(c) = m$, potem je tudi $\mathcal{D}_{\mathrm{sk}}(\mathcal{RE}_{\mathrm{pk}}(c; r)) = m$.

Homomorfni kriptosistemi ne morejo biti IND-CCA2 varni, že zaradi zmožnosti ponovnega šifriranja. Po drugi strani pa homomorfizem ne preprečuje IND-CPA varnosti, kako pa homomorfizem vpliva na IND-CCA varnost, pa ni ravno jasno [1, str. 49].

2.3.2 Primeri homomorfnih kriptosistemov z javnimi ključi

Oglejmo si homomorfne lastnosti nekaj znanih kriptosistemov.

RSA. V osnovni različici RSA šifrira po formuli $c = m^e \bmod n$. Posledično je $c_1 \cdot c_2 = (m_1 \cdot m_2)^e \bmod n$. Osnovni RSA je torej homomorfen za operaciji $(\cdot, \cdot)$, vendar zaradi determinističnega šifriranja ni IND-CPA varen in je torej neuporaben. V uporabni izvedbi skupaj z OAEP pa izgubi homomorfizem.

ElGamalov kriptosistem. Naj bo $\otimes$ operacija množenja tajnopisov po komponentah.

Trditev 2.3.1. *ElGamalov kriptosistem je homomorfen za operaciji* $(\cdot, \otimes)$

Dokaz.

$$\begin{aligned}
\mathcal{E}_{\mathrm{pk}}(m_1; r_1) \otimes \mathcal{E}_{\mathrm{pk}}(m_2; r_2) &= (g^{r_1}, m_1 \cdot y^{r_1}) \otimes (g^{r_2}, m_2 \cdot y^{r_2}) \\
&= (g^{r_1+r_2}, (m_1 \cdot m_2) \cdot y^{r_1+r_2}) \\
&= \mathcal{E}_{\mathrm{pk}}(m_1 \cdot m_2; r_1 + r_2).
\end{aligned}$$

□

ElGamalov kriptosistem je zanimiv, saj je homomorfen in IND-CPA varen.

Eksponentni ElGamal. Pri običajnem ElGamalovem kriptosistemu šifriramo tako, da "ključ" y^r pomnožimo s sporočilom m. Če pa sporočilo m postavimo v eksponent in "ključ" y^r pomnožimo z g^m, ne pa z m, je tak kriptosistem aditivno homomorfen. V tem primeru po običajnem ElGamalovem odšifriranju

dobimo vrednost g^m, ki pa nam ne pomeni veliko, zato moramo izračunati še diskretni logaritem, kar pa omeji prostor čistopisov, a to pri mnogih primerih e-volitev ni težava, saj je število možnih glasov dovolj majhno, da lahko diskretni logaritem izračunamo dovolj hitro. Eksponentni ElGamalov kriptosistem zapišimo še formalno:

- **Generiranje para ključev**. Izberemo veliko praštevilo p in tak generator $g \in \mathbb{Z}_p^*$, da je $|\langle g \rangle| = q$. Nato naključno izberemo $x \in \mathbb{Z}_q$. Ključa sta potem:
  - $\mathrm{pk} = (p, g, y)$, kjer je $y = g^x \bmod p$,
  - $\mathrm{sk} = x$.
- **Šifriranje**. Naj bo $\mathbb{Z}_q$ prostor čistopisov. Čistopis $m \in \mathbb{Z}_q$ šifriramo z uporabo javnega ključa pk in šifrirnega algoritma $\mathcal{E}$ po naslednji formuli:

$$c = (\alpha, \beta) = \mathcal{E}_{\mathrm{pk}}(m) = (g^r, g^m \cdot y^r), \quad r \in_R \mathbb{Z}_q.$$

- **Odšifriranje**. Naj bo $\langle g \rangle \times \langle g \rangle$ prostor tajnopisov. Tajnopis $c = (\alpha, \beta) \in \langle g \rangle \times \langle g \rangle$ odšifriramo z uporabo zasebnega ključa sk in odšifrirnega algoritma $\mathcal{D}$ po naslednji formuli:

$$m = \mathcal{D}_{\mathrm{sk}}(c) = \log_g(\beta \cdot \alpha^{-x} \bmod p) \bmod p.$$

Naj bo $\otimes$ operacija množenja tajnopisov po komponentah.

Trditev 2.3.2. *Eksponentni ElGamalov kriptosistem je homomorfen za operaciji* $(+, \otimes)$ *(aditivno homomorfen).*

Dokaz.

$$\begin{aligned} \mathcal{E}_{\mathrm{pk}}(m_1; r_1) \otimes \mathcal{E}_{\mathrm{pk}}(m_2; r_2) &= (g^{r_1}, g^{m_1} \cdot y^{r_1}) \otimes (g^{r_2}, g^{m_2} \cdot y^{r_2}) \\ &= (g^{r_1+r_2}, g^{m_1+m_2} \cdot y^{r_1+r_2}) \\ &= \mathcal{E}_{\mathrm{pk}}(m_1 + m_2; r_1 + r_2). \end{aligned}$$

□

V praksi je število možnih čistopisov približno reda 10^{12}, diskretni logaritem pa pri odšifriranju računamo z učinkovitimi algoritmi, kot je npr. metoda *veliki korak mali korak*.

Paillierjev kriptosistem

Trditev 2.3.3. *Paillierjev kriptosistem je homomorfen za operaciji* $(+, \cdot)$ *(aditivno homomorfen).*

Dokaz.

$$\begin{aligned}\mathcal{E}_{\mathrm{pk}}(m_1; r_1) \cdot \mathcal{E}_{\mathrm{pk}}(m_2; r_2) &= (g^{m_1} \cdot r_1^n) \cdot (g^{m_2} \cdot r_2^n) \\ &= (g^{m_1+m_2} \cdot (r_1 r_2)^n) \\ &= \mathcal{E}_{\mathrm{pk}}(m_1 + m_2; r_1 r_2).\end{aligned}$$

□

Paillierjev kriptosistem nima slabosti, kot je omejeno število sporočil pri eksponentnem ElGamalovem kriptosistemu. Dodatno pa lahko prostor čistopisov povečamo z uporabo posplošenega Paillierjevega kriptosistema, ki je znan tudi kot Damgard-Jurikov kriptosistem [12]. Pri tem kriptosistemu je prostor čistopisov $\mathbb{Z}_{n^s}$, prostor tajnopisov pa $\mathbb{Z}^*_{n^{(s+1)}}$, kjer je s poljubno naravno število, v praksi 2, 3 ali 4.

2.4 Pragovni kriptosistemi z javnimi ključi

Pri kriptografskih e-volitvah so glasovnice šifrirane z javnim ključem volitev pk in objavljene na oglasni deski skupaj z neko obliko identifikacije volivca. Tako javnost lahko preveri, da so glasovali le upravičeni volivci, vsak volivec pa vidi, da je njegova šifrirana glasovnica prispela na cilj nespremenjeno.

Slabost tega sistema pa je, da bi lahko imetnik zasebnega ključa sk odšifriral posamezno glasovnico. Da se to prepreči, se zasebni ključ sk ne zaupa eni sami avtoriteti, ampak se ga razdeli med ℓ zaupnikov (angl. trustee), od katerih zaupnik i prejme delež zasebnega ključa $\mathrm{sk}^{(i)}$. Zasebni ključ se razdeli po principu **deljenja skrivnosti**, z uporabo pragovne sheme (angl. threshold scheme). Bistvo pragovnih shem je v dveh lastnostih:

- Šele ob doseženem pragu k deležev $\mathrm{sk}^{(i)}$, se lahko izračuna skrivnost.
- S $k-1$ ali manj deleži ne moremo izračunati prav nobene informacije o skrivnosti.

Skrivnost je v našem primeru zasebni ključ sk. Dovolj velik prag k onemogoča, da bi koalicija zaupnikov (npr. političnih strank) odšifrirala posamezne glasovnice, dovolj velik ℓ pa onemogoča, da bi koalicija nezadovoljnih zaupnikov preprečila štetje glasov.

2.4.1 Shamirjevo deljenje skrivnosti

Najbrž najbolj zanimivo pragovno shemo za deljenje skrivnosti je predstavil Adi Shamir leta 1979 [30]. Naj bo s skrivnost v končnem obsegu. Skrivnost s se po Shamirjevi shemi razdeli v množico deležev $s^{(1)}, s^{(2)}, \ldots, s^{(\ell)}$, tako da katerakoli podmnožica velikosti k ali več omogoča učinkovit izračun s. Po drugi strani pa katerakoli podmnožica velikosti $k-1$ ali manj ne omogoča izračuna prav nobene informacije o vrednosti s oz. so vse možne vrednosti za s enako verjetne.

Skrivnost s razdelimo tako, da izberemo tak poljuben polinom $P(x)$ stopnje $k-1$ nad primernim končnim obsegom, da je $P(0) = s$. Delež $s^{(i)}$ je potem točka polinoma (x_i, y_i), tako da je $y_i = P(x_i)$. Deleži morajo biti med seboj različni. Ker je polinom P stopnje $k-1$, ga lahko rekonstruiramo s pomočjo k različnih točk, ki pripadajo P, torej k različnih deležev $s^{(i)}$. Za rekonstrukcijo uporabimo Lagrangeve polinome. Naj bo $\{(x_i, y_i)\}$ množica poljubnih k od ℓ točk. Lagrangev polinom za točko i označimo z $\lambda_i(x)$ in ga definiramo kot:

$$\lambda_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{k} \frac{x - x_j}{x_i - x_j}.$$

Posledično je interpolirani polinom enak:

$$P(x) = \sum_{i=1}^{k} \lambda_i(x) y_i.$$

Ker nas zanima samo $s = P(0)$ in ne ostali koeficienti $P(x)$, lahko neposredno izračunamo s na naslednji način:

$$s = P(0) = \sum_{i=1}^{k} \lambda_i(0) y_i = \sum_{i=1}^{k} y_i \left(\prod_{\substack{j=1 \\ j \neq i}}^{k} \frac{-x_j}{x_i - x_j} \right). \tag{2.4.1}$$

Dovoljeno je, da so vsi x_i javni, lahko je kar $x_i = i$. Pripadajoči y_i pa morajo ostati tajni, vsak y_i je znan zgolj zaupniku i.

2.4.2 Izvedba pragovnih kriptosistemov

Shamirjevo deljenje skrivnosti, ki smo ga ravnokar predstavili, je splošna metoda za deljenje skrivnosti s. Skrivnost s je lahko zasebni ključ, geslo, itd. V

našem primeru je skrivnost s zasebni ključ. Najbolj neposredno bi pragovni kriptosistem izvedli tako, da bi imeli nek zaupanja vreden strežnik, ki bi generiral par ključev. Nato bi strežnik zasebni ključ sk po Shamirjevi shemi razdelil na deleže, deleže razdelil med zaupnike, zasebni ključ pa zavrgel. Ko bi zaupniki želeli izvesti odšifriranje, bi v zaupanja vreden strežnik vnesli svoje deleže, strežnik bi iz deležev izračunal zasebni ključ sk in z uporabo sk izvedel odšifriranje. Na koncu bi strežnik zavrgel sk in deleže, zaupnikom pa bi vrnil čistopis.

Čeprav pravkar opisana izvedba pragovnega kriptosistema ni za odpis, bi si vseeno želeli nekaj izboljšav. Prva taka izboljšava je uvedba **porazdeljenega odšifriranja**, pri čemer vsak od k zaupnikov, ki so se zbrali za odšifriranje, izvede delno odšifriranje z uporabo svojega deleža. Tako ni potreben izračun zasebnega ključa sk, prav tako pa se izognemo uporabi zaupanja vrednega strežnika, ki bi bil lahko nepošten.

Druga izboljšava pa je uvedba **porazdeljenega generiranja zasebnega ključa**. Pri tem mora sodelovati vseh ℓ zaupnikov. Na ta način dosežemo, da zasebni ključ sk nikoli (niti pri generiranju) sploh ne izračunamo. Izračunamo zgolj ℓ deležev, tako da lahko s poljubno podmnožico k deležev izračunamo sk. Vendar, če porazdeljeno generiranje zasebnega ključa uporabimo skupaj s porazdeljenim odšifriranjem, nam ključa sk zares ni potrebno nikoli izračunati.

V nadaljevanju opišemo, kako se izvede učinkovit in varen pragovni kriptosistem z uporabo Shamirjeve sheme. Poudarek bo na ElGamalovem kriptosistemu, ki zaradi svoje zgradbe omogoča varno, učinkovito in ne preveč zapleteno izvedbo pragovnega kriptosistema.

ElGamal. Za začetek naj izvedba omogoča zgolj porazdeljeno odšifriranje. Generiranje ključa in deljenje skrivnosti opravlja zaupanja vreden strežnik.

Zaradi nedvoumnosti oznak bomo za potrebe te razlage ElGamalov zasebni ključ označili drugače kot ponavadi. Naj bo skrivnost ElGamalov zasebni ključ a in naj bo $b = g^a$ pripadajoči javni ključ. Zaupanja vreden strežnik z uporabo Shamirjeve sheme razdeli skrivnost a na deleže $a_1, \ldots, a_\ell$. Delež a_i skrivnosti a ni več par, saj smo pri definiciji Shamirjeve sheme zapisali, da so x koordinate lahko javne in lahko kdorkoli z njimi izračuna Lagrangeve koeficiente. Delež a_i ustreza vrednosti y_i iz enačbe 2.4.1. Vsakemu deležu a_i pa pripada javni delež $b_i = g^{a_i}$ javnega ključa b. Delež b_i ustreza vrednosti x_i iz enačbe 2.4.1. Kdorkoli lahko izračuna javni ključ z uporabo javnih deležev b_i poljubnih k zaupnikov. To se naredi tako, da se za vsakega od k zaupnikov izračuna Lagrangev koeficient v točki $x = 0$, t.j. $\lambda_i(0)$. Lagrangevih koeficientov ni možno izračunati vnaprej, saj so odvisni od izbire k zaupnikov. Za majhna k

in ℓ bi sicer bilo možno vnaprej izračunati koeficiente za vse možne podmnožice k zaupnikov, v splošnem pa je to pretežko. Z uporabo $\lambda_i(0)$ in pripadajočih deležev $b_i = g^{a_i}$ javnega ključa b nato izračunamo javni ključ po formuli:

$$b = g^a = g^{\sum_{i=1}^{k} a_i \lambda_i(0)} = \prod_{i=1}^{k} g^{a_i \lambda_i(0)} = \prod_{i=1}^{k} (g^{a_i})^{\lambda_i(0)}.$$

Podobno izvedemo odšifriranje, le da to lahko opravijo samo zaupniki s svojimi deleži zasebnega ključa a_i:

$$m = \beta \cdot \alpha^{-a} = \beta \cdot (\prod_{i=1}^{k} \alpha^{a_i \lambda_i(0)})^{-1}.$$

Prednost tako izvedenega odšifriranja je, da nikoli ne izračunamo zasebnega ključa a. Zaradi še večje varnosti je dobro izvesti tudi porazdeljeno generiranje ključev. Tak protokol za ElGamalov kriptosistem je opisan v [25]. Imejmo spet ℓ zaupnikov in zahtevo, da lahko poljubna skupina k zaupnikov izvede odšifriranje:

1. Vsak zaupnik generira svoj tajni delež a_i ter objavi $b_i = g^{a_i}$. Pri tem je a_i izbran naključno, zasebni ključ a pa je kar vsota vseh a_i.

2. Vsak zaupnik svoj delež a_i po metodi za deljenje skrivnosti razdeli med vse ostale zaupnike. To naredi z uporabo preverljivega deljenja skrivnosti (angl. verifiable secret sharing), s čimer onemogočimo goljufive udeležence. Tudi tukaj je shema pragovna in dovoljuje k od ℓ udeležencem izračun skrivnosti.

3. Poljubna skupina k zaupnikov mora sedaj opraviti ℓk operacij, da izračuna zasebni ključ. Izračunati je potrebno a_i za vsakega od ℓ zaupnikov, kar je možno, saj je bil vsak a_i razdeljen tako, da ga lahko skupina k zaupnikov izračuna. Za vsak tak izračun je potrebnih k operacij, skupno torej ℓk.

Paillier. Paillierjev kriptosistem je zanimiv za e-volitve, ki uporabljajo homomorfizem za štetje glasov. Obstajajo učinkovite metode za porazdeljeno odšifriranje [5] in za porazdeljeno generiranje ključev. Avtorja posplošene različice Paillierjeva kriptosistema sta v [12] opisala tudi metodo za porazdeljeno odšifriranje.

Poglavje 3

Dokazi brez razkritja znanja in osnove e-volitev

V prvem delu poglavja predstavimo dokaze brez razkritja znanja, ki imajo pomembno vlogo pri vseh vrstah kriptografskih shem za e-volitve. V drugem delu poglavja z uporabo do sedaj predstavljenih kriptografskih konceptov predstavimo osnove kriptografskih e-volitev.

3.1 Dokazi brez razkritja znanja

Naj bo za potrebe dokazov v tej nalogi Primož oseba, ki dokazuje, Vera pa tista, ki dokaz preverja. Dokazi brez razkritja znanja (angl. zero-knowledge proof, kratica ZKP) so protokoli, pri katerih poskuša Primož prepričati Vero o resničnosti neke izjave, brez da bi pri tem izdal katerokoli dodatno informacijo, razen da je izjava resnična. Od ZKP protokola želimo da je:

- **Poln** (angl. complete). Če je Vera poštena, bo sprejela dokaz poštenega Primoža z visoko verjetnostjo.
- **Uglašen** (angl. sound). Če je Vera poštena, bo zavrnila dokaz nepoštenega Primoža z visoko verjetnostjo. Tej verjetnosti bomo rekli uglašenost protokola.
- **Brez razkritja znanja** (angl. zero-knowledge). Pri dokazovanju pošteni Primož ne izda nikakršne informacije, le da dokazovana trditev drži.

Tipično izvedbo interaktivnega ZKP prikazuje protokol 3.1.

Protokol 3.1 Interaktivni ZKP protokol

1: **Zaveza**: Primož naključno izbere tajno vrednost in izračuna pripadajočo (javno) zavezo (angl. commitment). Slednjo pošlje Veri. S tem se Primož javno zaveže (angl. commit) k izbrani tajni vrednosti. Tajna vrednost skupaj s skrivnostjo, ki naj bi jo poznal Primož, določa množico Verinih vprašanj, na katere naj bi Primož znal odgovoriti.
2: **Izziv**: Vera sestavi izziv, tako da izbere eno od vprašanj, in ga pošlje Primožu.
3: **Odgovor**: Primož Veri pošlje nazaj odgovor na njen izziv.
4: **Preverjanje**: Če je odgovor pravilen, Vera sprejme dokaz, sicer ga zavrne.

Uglašenost ZKP protokolov navajamo za eno izvajanje zgoraj opisanih treh korakov. Da bi povečali uglašenost pri uporabi protokola, protokol večkrat ponovimo.

Z vidika varnosti v praksi ne govorimo o popolnih ZKP, temveč o računskih ZKP (angl. computational zero-knowledge) in torej predpostavimo, da ima Vera na voljo verjetnostni polinomsko časovno omejen algoritem. To pomeni, da če bi imela Vera na voljo neomejeno računsko moč, bi lahko iz izvajanja protokola dobila dodatne informacije. Znotraj računskih ZKP protokolov ločimo 3 vrste:

- **Dokaz brez razkritja znanja** (angl. zero-knowledge proof) je ZKP protokol, pri katerem predpostavimo, da je Primož računsko neomejen.
- **Argument brez razkritja znanja** (angl. zero-knowledge argument) je ZKP protokol, pri katerem predpostavimo, da ima Primož omejeno računsko moč.
- **Brez razkritja znanja, če je Vera poštena** (angl. honest verifier zero-knowledge, kratica HVZK). Tu predpostavimo, da je Vera poštena. To pomeni, da bi nepoštena Vera morda lahko iz HVZK dokaza izvedela dodatne informacije.

3.1.1 Hevristika Fiat-Shamir

HVZK dokazi so varnostno najšibkejši, saj predpostavljajo, da je Vera poštena, a so zelo uporabni, posebej ker je možno interaktivnost odpraviti s pomočjo hevristike Fiat-Shamir [14]. Le-ta bo poskrbela, da bo Primož lahko izračunal dokaz v enem koraku brez sodelovanja Vere. Avtorja sta pokazala, da je možno interaktivni protokol za identifikacijo pretvoriti v shemo za digitalni podpis.

Ideja hevristike Fiath-Shamir je, da izziv v 2. koraku namesto Vere izračuna Primož sam. To naredi s pomočjo kriptografske zgoščevalne funkcije, ki ji kot vhod poda zavezo, ki jo je izračunal v 1. koraku. Nato izvede še tretji korak, tako da izračuna odgovor na "izziv" in nato zapis vseh treh korakov pošlje Veri oz. ga objavi. Zaradi varnosti je pomembno, da se zgoščevalna funkcija obnaša kot naključni orakelj, se pravi da izhod zgoščevalne funkcije glede na vhode izgledajo naključno, vendar pa funkcija za iste vhode vedno vrne iste izhode.

Hevristika Fiat-Shamir je pomembna, saj pogosto ni praktično (ali je celo nemogoče) izvajati interaktivne ZKP protokole, ampak si želimo, da bi Primož enkrat izračunal dokaz in ga objavil, kdorkoli pa bi ga lahko kasneje preveril. Zato neinteraktivnim dokazom brez razkritja znanja pravimo tudi *podpisi znanja*. V nadaljevanju bomo ZKP protokole opisovali tako, da bomo opisali korake pri interaktivni različici dokaza, pri tem pa upoštevali, da so v praksi pogosto izvedeni neinteraktivno s pomočjo hevristike Fiat-Shamir.

3.1.2 Schnorrov dokaz

Schnorr [28] je razvil identifikacijski protokol, katerega bistvo je, da Primož interaktivno prepriča Vero o poznavanju diskretnega logaritma nekega elementa. V istem članku je predstavil tudi neinteraktivno različico, ki je del sheme za digitalne podpise.

Naj bo $p \in \mathbb{P}$ in naj ima element $g \in \mathbb{Z}_p^*$ praštevilski red q. Predpostavimo, da je DLP računsko neobvladljiv. Za $y \in \langle g \rangle$, tj. $y = g^x$ za nek $x \in \mathbb{Z}_q$, Primož dokaže poznavanje števila x s HVZK protokolom 3.2.

Protokol 3.2 Schnorrov dokaz o poznavanju diskretnega logaritma

1: Primož izbere $r \in_R \mathbb{Z}_q$ in izračuna zavezo $a = g^r \bmod p$ ter jo pošlje Veri.
2: Vera naključno izbere izziv $c \in_R \mathbb{Z}_q$ in ga pošlje Primožu.
3: Primož izračuna odgovor $s = r + cx \bmod p$ in ga pošlje Veri.
4: Vera sprejme dokaz, če je $g^s \equiv ay^c \pmod p$.

V dokazu odpravimo interaktivnost tako, da spremenimo 2. korak. Namesto da izziv izračuna Vera, Primož sam izračuna izziv kot $c = H(a)$, kjer je H varna zgoščevalna funkcija.

Trditev 3.1.1. *Schnorrov dokaz je visoko uglašen ZKP. Verjetnost, da bi nepošteni Primož uspešno prepričal Vero, je* $1/q$*, pri čemer je* q *moč množice, iz katere izbiramo izziv.*

Dokaz. **Protokol je poln**. Če Primož res pozna x, potem bo Vera vedno sprejela dokaz, saj velja:

$$g^s \equiv g^{r+cx} \equiv g^r (g^x)^c \equiv a y^c \pmod{p}.$$

Protokol je uglašen. Recimo, da Primož ne pozna vrednosti x. Primož lahko prelisiči Vero, če mu uspe vnaprej uganiti izziv c. V tem primeru si Primož v prvem koraku protokola vnaprej naključno izbere odgovor s in izračuna zavezo $a = g^s \cdot y^{-c} \bmod p$. Verin izziv nato ignorira in ji kot odgovor vrne prej izbrano vrednost s, ki bo uspešno prestala Verino preverjanje. Ker je izziv izbran naključno iz $\mathbb{Z}_q$, je verjetnost, da bo Primož prelisičil Vero enaka $1/q$.
Protokol je brez razkritja znanja. Preverimo, če Vera iz dokaza res ne prejme nobene informacije o tajni vrednosti x. Vera od Primoža med izvajanjem protokola prejme zavezo a in odgovor s. Izmed slednjih je vrednost x vsebovana zgolj v odgovoru s. Da bi Vera lahko iz s dobila vrednost x, bi morala uganiti naključno vrednost r, ki jo pozna zgolj Primož, ali pa izračunati $r = \log_g a \bmod p$, kar pa je po predpostavki pretežko. □

3.1.3 Chaum-Pedersenov dokaz

Chaum in Pedersen [10] sta uvedla shemo za digitalni podpis, katere del je protokol za dokaz enakosti diskretnih logaritmov. Protokol zelo spominja na Schnorrovega. Ker gre za shemo za digitalne podpise, avtorja interaktivni protokol, ki služi predstavitvi, po zgledu hevristike Fiat-Shamir spremenita v neinteraktivnega.

Naj bo $p \in \mathbb{P}$ in naj ima element $g \in \mathbb{Z}_p^*$ praštevilski red q. Predpostavimo, da je DLP računsko neobvladljiv. Naj bo $m \in \langle g \rangle \setminus \{g\}$. Naj bo $h = g^x \bmod p$ in $z = m^x \bmod p$. Velja torej

$$\log_g h \equiv \log_m z \pmod{p}.$$

Vsi do sedaj našteti parametri so javni, razen seveda x. Primož uporabi HVZK protokol 3.3 za dokaz zgornje enakosti.

Protokol 3.3 Chaum-Pedersenov dokaz o enakosti dveh diskretnih logaritmov

1: Primož izbere $r \in_R \mathbb{Z}_q$ in izračuna zavezo $(a, b) = (g^r, m^r) \bmod p$ ter jo pošlje Veri.
2: Vera naključno izbere izziv $c \in_R \mathbb{Z}_q$ in ga pošlje Primožu.
3: Primož izračuna odgovor $s = r + cx \bmod p$ in ga pošlje Veri.
4: Vera sprejme dokaz, če je $g^s \equiv a h^c \pmod{p}$ in $m^s \equiv b z^c \pmod{p}$.

Tudi pri tem dokazu odpravimo interaktivnost tako, da pri 2. koraku Primož sam izračuna izziv kot $c = H(m||z||a||b)$, kjer je H varna zgoščevalna funkcija, $||$ pa pomeni spoj binarnih nizov, ki predstavljajo m, z, a in b. Chaum-Pedersenov dokaz enakosti diskretnih logaritmov včasih imenujemo tudi dokaz pravilne Diffie-Hellman trojice. Diffie-Hellman trojica je trojica (x, y, z) oblike (g^a, g^b, g^{ab}). Chaum-Pedersenov dokaz torej uporabimo za to, da dokažemo

$$\log_g x \equiv \log_y z \pmod{p}.$$

Trditev 3.1.2. *Chaum-Pedersenov dokaz je visoko uglašen ZKP. Verjetnost, da bi nepošteni Primož uspešno prepričal Vero, je $1/q$, pri čemer je q moč množice, iz katere izbiramo izziv.*

Dokaz. **Protokol je poln**. Če velja enakost in če Primož pozna x, potem bo Vera vedno sprejela dokaz, saj velja:

$$g^s \equiv g^{r+cx} \equiv g^r (g^x)^c \equiv a h^c \pmod{p}$$

in

$$m^s \equiv m^{r+cx} \equiv m^r (m^x)^c \equiv b z^c \pmod{p}.$$

Protokol je uglašen. Recimo, da enakost ne drži, torej $\log_g h \not\equiv \log_m z \pmod{p}$. To pomeni, da je $h = g^{x_1}$ in $z = m^{x_2}$ za različna x_1 in x_2. Slednji vrednosti sta znani Primožu in nista javni. Primož lahko prelisiči Vero, če mu uspe vnaprej uganiti izziv c. V tem primeru v prvem koraku protokola izbere tak par $(r_1, r_2) \in \mathbb{Z}_q^2$, da je $r_1 - r_2 = c(x_2 - x_1) \bmod p$ in izračuna zavezo $(a, b) = (g^{r_1}, m^{r_2}) \bmod p$. V tretjem koraku nato izračuna odgovor $s = r_1 + cx_1 = r_2 + cx_2 \bmod p$, ki bo uspešno prestal Verino preverjanje. Ker je izziv izbran naključno iz $\mathbb{Z}_q$, je verjetnost, da bo Primož prelisičil Vero enaka $1/q$.

Protokol je brez razkritja znanja. Preverimo, če Vera iz dokaza res ne prejme nobene informacije o tajni vrednosti x. Vera od Primoža med izvajanjem protokola prejme zavezo (a, b) in odgovor s. Izmed slednjih je vrednost x vsebovana zgolj v odgovoru s. Da bi Vera lahko iz s dobila vrednost x, bi morala uganiti naključno vrednost r, ki jo pozna zgolj Primož, ali pa izračunati $r = \log_g a \bmod p$ oz. $r = \log_m b \bmod p$, kar pa je po predpostavki pretežko. □

3.2 Osnove kriptografskih shem za e-volitve

Nekaj osnov o kriptografskih e-volitvah smo podali že v uvodnem poglavju, z uporabo novega znanja pa sedaj dopolnimo opis. Volilnih shem za e-volitve je

veliko. Praktično vsak raziskovalni članek, ki govori o kriptografskih gradnikih, ki se lahko uporabijo tudi za e-volitve, predlaga svojo shemo. V tem razdelku zato poskušamo podati splošen opis kriptografske sheme za e-volitve, ki velja za večino predlaganih shem.

Za začetek opišimo korake pri izvedbi kriptografskih e-volitev:

(a) **Priprava volitev**. Parametri volitev se generirajo in tisti, ki so javni, se tudi objavijo.

(b) **Priprava in zapisovanje glasovnice**. Volivec pripravi svojo glasovnico s pomočjo elektronske naprave ali pa izpolni že pripravljeno papirno glasovnico. V prvem primeru se elektronska naprava uporabi tudi za šifriranje glasovnice, v drugem primeru pa je glasovnica že predhodno šifrirana. V obeh primerih glasovnica vsebuje tajnopis, ki je šifriran z uporabo javnega ključa volitev. Tako šifrirana glasovnica se nato objavi na vsem dostopni oglasni deski skupaj z identiteto volivca. Identiteta volivca ni šifrirana, se pa lahko uporabi oblika, ki zagotavlja neke vrste anonimnost, npr. davčna številka.

(c) **Anonimizacija in združevanje glasovnic**. Na oddanih šifriranih glasovnicah se izvede operacija združevanja in operacija anonimizacije. Pri tem koraku gre torej za štetje glasov in pa zakrivanje povezave med glasovnico in volivcem, vse na kriptografsko varen in dokazljiv način.

(d) **Izračun rezultatov**. Volilni uradniki sodelujejo pri izračunu končnih rezultatov v nešifrirani obliki. Pri tem tudi objavijo dokaze o pravilnosti delovanja.

Kot smo že omenili, poznamo dve vrsti shem za splošno preverljive kriptografske e-volitve. V prvo vrsto spadajo sheme s homomorfnim štetjem glasov ali tudi **združevalne sheme** (angl. aggregate schemes), pri katerih v čistopisni obliki dobimo zgolj končni rezultat. V drugo vrsto spadajo sheme na osnovi mešalnih mrež ali tudi **sheme, ki ohranjajo glasovnice** (angl. ballot-preserving scheme). Pri teh po končani anonimizaciji dobimo čistopise glasovnic, torej je vsaka posamezna glasovnica ohranjena, vendar ne moremo nobene odšifrirane glasovnice povezati s pripadajočo oddano šifrirano glasovnico. V obeh primerih pa izvedemo pragovni PKC z uporabo pragovne sheme za deljenje skrivnosti, s čimer preprečimo odšifriranje posameznih glasovnic in bojkotiranje štetja glasovnic. V nadaljevanju opišemo zgoraj naštete korake bolj podrobno, s tem da točkama (c) in (d) namenimo svoj podrazdelek.

(a) **Priprava volitev**. Naj bo ℓ število volilnih uradnikov, pri čemer uradnik ni nujno ena sama oseba, ampak je lahko politična stranka, državni organ ipd. Označimo z U_i i-tega uradnika. Naj bo N število volivcev, j-tega volivca označimo z V_j. Naj bo s število volilnih vprašanj (angl. race), kjer k-to vprašanje označimo z R_k. Vprašanje R_k ima o_k možnih odgovorov, lahko pa shema dovoljuje tudi vpis vrednosti s strani volivca. Izbiro pri vprašanju R_k s strani volivca V_j označimo s $t_j^{(k)}$. Glasovnica za volivca V_j je potem

$$m_j = (t_j^{(1)}, t_j^{(2)}, \ldots, t_j^{(s)}).$$

Pri tem je način kodiranja m_j odvisen tudi od načina štetja glasov. Najbolj pomembni parametri za posamezne volitve so javni ključ pk, ki je javno objavljen in s katerim se šifrirajo glasovnice, ter ℓ deležev pripadajočega zasebnega ključa, ki jih označimo z $\mathrm{sk}^{(1)}, \ldots, \mathrm{sk}^{(\ell)}$, kjer $\mathrm{sk}^{(i)}$ pripada uradniku U_i. Pri generiranju parametrov volitev uporabljamo tehnike iz razdelka 2.4.

(b) **Priprava in zapisovanje glasovnice**. Volivec V_j zašifrira čistopis glasovnice m_j v tajnopis c_j z uporabo javnega ključa pk in ustrezne naključne vrednosti r_j. Konkreten kriptosistem, ki ga pri tem uporabimo, je odvisen od volilne sheme, mora pa biti semantično oz. IND-CPA varen.

V uvodnem poglavju smo govorili o **zagotovitvi oddaje glasovnice**, kjer je pomembno, da V_j dobi zagotovilo, da čistopisu glasovnice m_j res ustreza tajnopis c_j. Poleg tega pa je pomembna lastnost tudi **odpornost na prisilo**. Le-ta prepreči, da bi volivec nekomu tretjemu dokazal, kako je volil, tudi če bi sam tako želel. Prepričati volivca o pravilnosti šifriranja (zagotovitev oddaje) bi bilo preprosto, če ne bi bilo potrebno paziti na odpornost na prisilo. Volivcu bi preprosto izdali naključno vrednost r_j, uporabljeno pri šifriranju, in volivec bi lahko preveril, da se m_j res zašifrira v c_j. Vendar bi s tem seveda izgubili odpornost na prisilo. Obstajajo različni načini, kako hkrati zagotoviti obe lastnosti. Posebej jih ne bomo obravnavali, bomo pa jih spoznali pri konkretnih shemah, ki jih obravnavamo v 5. in 6. poglavju.

Pri oddaji glasovnice je pomembna tudi *avtentikacija volivca*, da ugotovimo identiteto volivca in upravičenost do oddaje glasu. Pri tem se za internetne sheme uporabljajo uveljavljeni načini digitalne avtentikacije, predvsem z uporabo digitalnih potrdil. Pri shemah, kjer volimo na voliščih, pa se lahko uporabljajo bolj klasični pristopi, npr. predložitev osebnega dokumenta. Pri zapisu šifrirane glasovnice na oglasno desko objavimo tudi identiteto volivca. Zaradi splošne preverljivosti je pomembno, da vsak lahko preveri, da so identitete volivcev na oglasni deski z uradno objavljenega seznama upravičenih volivcev. Vseeno pa zaradi dodatne anonimnosti raje uporabimo tako obliko identitete,

ki je nekdo, ki nima dostopa do npr. registra prebivalcev, ne more povezati z osebo. To je pomembno zaradi dveh razlogov. Objavljene glasovnice so šifrirane s kriptosistemi, ki so dolgoročno varni, vendar nikoli ne moremo biti povsem prepričani, da nekdo ne bo imel nekoč možnost glasovnice odšifrirati. Drugi razlog pa je podatek o udeležbi na volitvah. Če bi bile identitete npr. kar imena, bi vsi lahko videli, kdo je volil in kdo ne.

3.2.1 Anonimizacija in združevanje glasovnic ter izračun rezultatov

Ko se glasovnica zašifrira, jo na overjen način pošljemo na oglasno desko oz. na nek strežnik za zbiranje glasovnic. Glasovnice je nato potrebno ločiti od volivca in prešteti glasove. Pri tem ločimo dva načina, po katerih tudi delimo kriptografske sheme za e-volitve.

Sheme na osnovi mešalnih mrež (angl. mixnet-based scheme). Poznamo jih tudi kot sheme, ki ohranjajo glasovnice (angl. ballot-preserving scheme). Pri tem načinu se šifrirane glasovnice, ki so objavljene na oglasni deski oz. zbrane nekje drugje, pošlje skozi t.i. anonimni komunikacijski kanal, ki glasovnice permutira, ponovno šifrira ter na koncu odšifrira. Bistvo tega je, da odšifriranih glasovnic na izhodu kanala ni možno povezati s pripadajočimi šifriranimi glasovnicami na vhodu in posledično z volivci. Obenem pa mora tak kanal zagotoviti pravilnost delovanja, torej dokazati, da ni prišlo do spremembe glasovnic, dodajanja novih glasovnic, brisanja obstoječih glasovnic in da je bila res izvedena anonimizacija. Tak anonimni komunikacijski kanal realiziramo z uporabo mešalnih mrež.

Mešalne mreže so zanimive, saj omogočajo izvedbo kompleksnejših volitev kot sistemi s homomorfnim štetjem glasov, npr. volitve, kjer volivec sam vpiše neke podatke v glasovnico. Težava pri sistemih s homomorfnim štetjem glasov je namreč v tem, da morajo biti tajnopisi pripravljeni tako, da se nad njimi lahko izvede homomorfna operacija. Po drugi strani pa tajnopisi pri shemah z mešalnimi mrežami nimajo omejitev oblike. Imajo pa tajnopisi pri klasičnih mešalnih mrežah omejitev dolžine, saj so omejeni z velikostjo prostora čistopisov pri PKC. Vendar obstajajo tudi hibridne mešalne mreže, kjer vhode v mrežo šifriramo s simetričnimi kriptosistemi, a jih ne bomo obravnavali. Prednost mešalnih mrež je tudi, kot že alternativno ime pove, ohranjanje glasovnic. Za razliko od homomorfnih sistemov, kjer se združevanje izvaja nad tajnopisi glasovnic in na koncu dobimo odšifriran zgolj končni rezultat, pri shemah z mešalnimi mrežami dobimo odšifrirane posamezne glasovnice.

Seveda pa imajo mešalne mreže tudi slabosti. Glavna slabost je učinkovitost, saj so dokazi pravilnosti delovanja računsko precej zahtevni. Druga slabost, povezana s prvo, pa je, da je teorija pri dokazih najbolj učinkovitih mešalnih mrež sorazmerno zapletena, medtem ko je teorija pri homomorfnih shemah sorazmerno preprosta. Kljub temu so mešalne mreže zanimive, saj edine omogočajo izvedbo določenih vrst volitev, privlačne pa so tudi zaradi možnosti nadaljnega razvoja, medtem ko je teorija pri homomorfnih shemah bolj ali manj dokončna. Zaradi tega se mešalnim mrežam bolj podrobno posvetimo v 4. poglavju. Mešalne mreže je v prvotni različici uporabljal sistem Helios, ki ga spoznamo v 6. poglavju. Helios v zadnjih različicah ponovno vpeljuje možnost uporabe mešalnih mrež.

Sheme s homomorfnim štetjem glasov (angl. homomorphic-based scheme). Pri teh shemah uporabimo homomorfizem kriptosistemov, ki smo ga spoznali v razdelku 2.3. Homomorfizem sicer uporabljamo tudi pri shemah na osnovi mešalnih mrež, kjer je ponovno šifriranje, ki ga omogočajo homomorfni kriptosistemi, bistveno za delovanje mešalnih mrež. Tukaj pa izkoriščamo homomorfizem za to, da lahko preštevamo glasove, ne da bi odšifrirali posamezne glasovnice. Z izrazom **števec** bomo označevali tajnopis, ki nastane z množenjem glasovnic med seboj. Števec, zaradi uporabe homomorfnih kriptosistemov, odraža število glasov za neko izbiro. Kako točno kodiramo števce v glasovnicah, je odvisno od posameznih shem. Na vsak način se izvede nad tajnopisi glasovnic homomorfna operacija (ponavadi množenje), ki rezultira v seštevanju pripadajočih čistopisov. Za ta namen potrebujemo aditivni homomorfizem, ki omogoča, da se z množenjem tajnopisov pripadajoči čistopisi seštevajo. Ko imamo enkrat tajnopis, ki predstavlja seštevek posameznih tajnopisov, ga volilni uradniki pragovno odšifrirajo.

Osnovno različico z dvema možnima odgovoroma sta predlagala Benaloh in Fischer [7]. Imejmo DA odgovor, ki ga predstavlja čistopis 1, in NE odgovor, ki ga predstavlja čistopis 0. Vsaka šifrirana glasovnica je torej oblike $\mathcal{E}_{\mathrm{pk}}(0)$ ali $\mathcal{E}_{\mathrm{pk}}(1)$. Naj bo izmed N glasovnic k takih, ki šifrirajo 1, preostalih $N-k$ pa šifrira 0. Ko vseh N glasovnic pomnožimo med seboj, dobimo

$$\overbrace{\mathcal{E}_{\mathrm{pk}}(1)\cdot\mathcal{E}_{\mathrm{pk}}(1)\cdots\mathcal{E}_{\mathrm{pk}}(1)}^{k}\cdot\overbrace{\mathcal{E}_{\mathrm{pk}}(0)\cdot\mathcal{E}_{\mathrm{pk}}(0)\cdots\mathcal{E}_{\mathrm{pk}}(0)}^{N-k} = \mathcal{E}_{\mathrm{pk}}(\overbrace{1+\cdots+1}^{k}+\overbrace{0+\cdots+0}^{N-k}) = \mathcal{E}_{\mathrm{pk}}(k).$$

Zbrane šifrirane glasovnice se torej pomnožijo, tako zmnožen tajnopis pa nato

volilni uradniki pragovno odšifrirajo in dobijo število DA glasov. Preostanek $N - k$ je število NE glasov.

Primer kriptosistema, ki izkazuje aditivni homomorfizem, je eksponentni ElGamalov kriptosistem, ki smo ga spoznali v razdelku 2.3.2. Slabost tega kriptosistema je, da je pri odšifriranju potrebno računati diskretni logaritem, kar pa ni težava za preprostejše volitve z enim števcem. V tem primeru je tudi pri velikem številu glasov računanje diskretnega logaritma hitro. Če pa bi želeli uporabljati tehniko večih števcev (za npr. več kandidatov ali vprašanj), ki so jo predlagali Baudron et al. [5], potem potrebujemo kriptosistem, ki izkazuje aditivni homomorfizem in ima učinkovito operacijo odšifriranja. Tak primer je Paillierjev kriptosistem, ki ga uporabijo avtorji v [5]. Na protokolu avtorjev Baudron et al. temelji tudi shema Scratch & Vote [4], ki jo predstavimo v 5. poglavju. Homomorfno štetje glasov z eksponentnim ElGamalovim kriptosistemom pa uporablja sistem Helios. Tukaj velja opozoriti, da pri tehniki večih števcev avtorjev Baudron et al. govorimo o kodiranju večih števcev znotraj enega tajnopisa. Tudi sistem Helios, ki ne uporablja tega načina, omogoča vprašanja z več odgovori in večje število vprašanj, le da to reši na preprost način, in sicer z uporabo večjega števila tajnopisov. Podrobnosti bomo videli pri opisu Heliosa.

Pri homomorfnih sistemih je potrebno poskrbeti tudi za pravilno oblikovane glasovnice. Volivec bi lahko za svojega kandidata preprosto šifriral vrednost 1000 in mu s tem namenil 1000 glasov namesto enega. Zaradi tega morajo pri homomorfnih shemah volivci skupaj z glasovnico oddati tudi dokaz, da oddani tajnopis glasovnice pripada pravilno oblikovanemu čistopisu. Seveda dokaz ne sme razkriti nobene informacije o tem, katera vrednost je šifrirana v tajnopisu glasovnice, kar nas ponovno pripelje do dokazov brez razkritja znanja.

Poglavje 4

Mešalne mreže

Prvo mešalno mrežo je predstavil David Chaum leta 1981 [9]. Mešalne mreže služijo izvedbi anonimnega komunikacijskega kanala, tipično tako, da je množica tajnopisov poslana na vhod mešalne mreže, na izhodu pa dobimo množico čistopisov, ki jih ni mogoče povezati s pripadajočimi tajnopisi. Za kriptografske sheme za e-volitve so mešalne mreže zanimive, saj omogočajo ohranitev posameznih glasovnic in s tem bolj kompleksne glasovnice, kot tiste, ki so pripravljene za homomorfno štetje glasov. Da pa bi bile mešalne mreže zanimive za volitve, mora kdorkoli imeti možnost preveriti, da je bilo mešanje izvedeno korektno. V tem primeru govorimo o **splošno preverljivih mešalnih mrežah** (angl. universally verifiable mixnet).

4.1 Osnovni pojmi in zapis

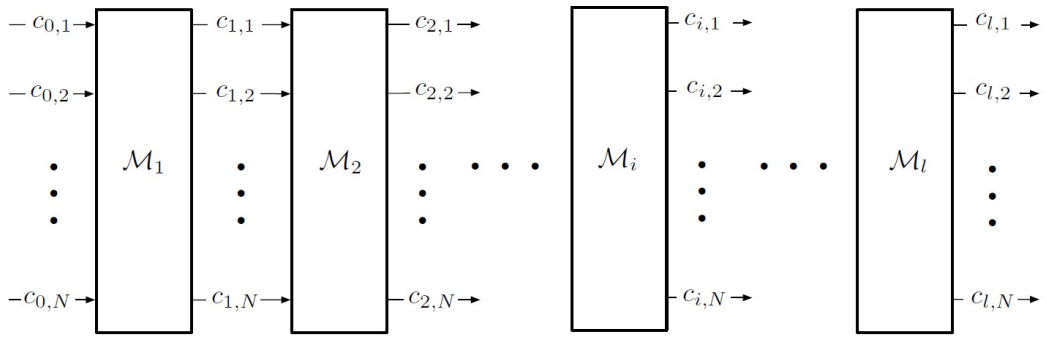
Mešalno mrežo označimo z $\mathcal{M}$. Sestavljena je iz ℓ mešalnih strežnikov (angl. mix server), kjer i-ti mešalni strežnik označimo z $\mathcal{M}_i$, $i \in [1, \ell]$. Naj bo N število volivcev oz. sporočil, ki jih obravnava mešalna mreža $\mathcal{M}$. Za $j \in [1, N]$ strežnik $\mathcal{M}_i$ sprejme tajnopis $c_{i-1,j}$ in na izhodu vrne tajnopis $c_{i,j}$. Ko bomo v kontekstu mešalnih mrež govorili o tajnopisih in čistopisih, bomo imeli v mislih tajnopise in čistopise glasovnic. Včasih bomo govorili tudi o vhodih in izhodih mešalnih strežnikov, kar bo ponavadi pomenilo različne oblike tajnopisov glasovnic, včasih pa tudi čistopise. Primer, ko so izhodi mešalnega strežnika čistopisi, je zadnji mešalni strežnik v mreži, kjer strežniki sproti izvajajo delno odšifriranje.

Tipično mešalni strežnik $\mathcal{M}_i$ tajnopise $c_{i-1,j}$ ponovno šifrira in permutira. Permutacijo, ki jo izvede strežnik $\mathcal{M}_i$, označimo s π_i, naključno vrednost, ki jo uporabi za ponovno šifriranje tajnopisa $c_{i-1,j}$, pa z $r_{i,j}$. Vsak mešalni strežnik

$\mathcal{M}_i$ potem izračuna svoje izhode tako, da velja

$$\mathcal{D}(c_{i,\pi_i(j)}) = \mathcal{D}(c_{i-1,j}), \quad \forall i \in [1,\ell], \forall j \in [1,N],$$

kjer je $\mathcal{D}(c)$ čistopis, ki pripada tajnopisu c. Naj opozorimo, da tajnopisa $c_{i-1,j}$ in $c_{i,\pi_i(j)}$ nista enaka, čeprav jima pripada enak čistopis. Tajnopis $c_{i,\pi_i(j)}$ je ponovno šifriran tajnopis $c_{i-1,j}$, pri čemer se je za ponovno šifriranje uporabila naključna vrednost $r_{i,j}$. Delovanje prikazuje slika 4.1. Poleg ponovnega



Slika 4.1: Mešalna mreža z ℓ mešalnimi strežniki in N vhodi.

šifriranja in permutiranja, mešalni strežniki izvajajo tudi odšifriranje tajnopisov glasovnic. Odvisno od mreže, lahko to dela vsak mešalni strežnik sproti, lahko pa po končanem ponovnem šifriranju in permutiranju vsi strežniki skupaj izvedejo odšifriranje.

Tajnost. Mešalne mreže poskušajo zakriti povezavo med vhodi in izhodi mreže. Zato že v osnovi govorimo o računski varnosti, saj bi računsko neomejen nasprotnik preprosto odšifriral tajnopise na vhodu in jih primerjal s čistopisi na izhodu. Ker splošno preverljive mešalne mreže podajo dokaze o pravilnosti delovanja, se je potrebno vprašati, koliko, če kaj, informacije uide s takim dokazom, in ali lahko mešalna mreža prelisiči protokol za dokazovanje. Pri tajnosti ločimo tri stopnje polnosti:

- **Polna in neodvisna** tajnost pomeni, da so vse permutacije med vhodi in izhodi možne, računsko omejen nasprotnik pa ne more dobiti nobene informacije o odvisnosti med vhodi in izhodi.
- **Polna toda odvisna** tajnost pomeni, da pri vseh možnih mešanjih kateremukoli vhodu lahko ustreza katerikoli izhod, niso pa možne vse permutacije v enem mešanju. To pomeni, da če poznamo določene povezave med vhodi in izhodi, nam dokaz pravilnosti lahko izda dodatne povezave.

- **Nepolna** tajnost pomeni, da dokaz pravilnosti omeji možne povezave med vhodi in izhodi.

Seveda je najboljša prva možnost, vendar se v praksi lahko zadovoljimo tudi s preostalimi možnostmi, če le ni prevelike odvisnosti in nepolnosti.

Oglasna deska. Omenjali smo jo že v uvodnem poglavju. S podrobnostmi izvedbe se ne bomo ukvarjali, zgolj predpostavimo, da je oglasna deska avtenticiran oddajni kanal (angl. authenticated broadcast channel). To pomeni, da avtenticirani uporabniki na njej objavijo sporočila, kdorkoli pa potem lahko vidi ta sporočila. Mešalne mreže so sestavljene iz mešalnih strežnikov, ki izvajajo različne operacije nad tajnopisi z namenom, da se zakrije povezava med vhodi in izhodi. Na začetku volivci objavijo tajnopise glasovnic na oglasni deski, od koder jih prebere prvi mešalni strežnik, opravi primerne operacije nad njimi in rezultate objavi nazaj na oglasno desko. Vsak mešalni strežnik nato, ko je na vrsti, prebere z oglasne deske tajnopise, ki jih je tja objavil strežnik, ki je mešal pred njim. Zadnji mešalni strežnik objavi (včasih ob pomoči ostalih strežnikov pri zadnjem koraku odšifriranja) odšifrirana sporočila na oglasni deski.

V nadaljevanju predstavimo različne vrste mešalnih mrež, ideje, na katerih temeljijo, napade na njih in protiukrepe.

4.2 Prvotne mešalne mreže

4.2.1 Chaumova mešalna mreža

Kot smo že omenili, je prvo mešalno mrežo predstavil David Chaum leta 1981 [9]. Uporabil je **RSA lupine** (angl. RSA onions) z naključnim dopolnilom. Ideja RSA lupin je, da se čistopis "ovije" z večimi plastmi (lupinami) šifriranja. Za vsako plast uporabimo različen javni ključ in naključno dopolnilo, tako da eni plasti šifriranja ustreza en mešalni strežnik. Vsak mešalni strežnik $\mathcal{M}_i$ ima svoj javni ključ pk_i in zasebni ključ sk_i. Vhod v mešalno mrežo je čistopis m_j šifriran na naslednji način:

$$c_{0,j} = \mathcal{E}_{\mathrm{pk}_1}\,(r_{1,j}, \mathcal{E}_{\mathrm{pk}_2}\,(r_{2,j}, \ldots, \mathcal{E}_{\mathrm{pk}_\ell}(r_{\ell,j}, m_j)\ldots)).$$

Mešalni strežniki mešajo po vrsti, prvi je $\mathcal{M}_1$, zadnji pa $\mathcal{M}_\ell$. Mešalni strežnik $\mathcal{M}_i$ odšifrira lupino, šifrirano z javnim ključem pk_i, odstrani naključno dopolnilo $r_{i,j}$ in nato na izhod vrne "olupljene" tajnopise po leksikografskem vrstnem redu. Zadnji mešalni strežnik $\mathcal{M}_\ell$ odstrani še zadnjo lupino in na svoj izhod,

ki je tudi izhod celotne mešalne mreže, vrne permutirane čistopise. Chaumova mešalna mreža ne nudi splošne preverljivosti.

Varnost. Napad na Chaumovo mrežo izkorišča homomorfizem kriptosistema RSA. Gre za napad s povezanim vhodom (angl. related input attack), kjer napadalec uporabi dve zaporedni mešanji, in sicer tako, da v drugo mešanje pošlje vhod, ki je povezan z vhodom iz prvega mešanja, ki bi ga rad izsledil. Če napadalca zanima $c_{0,j}$ iz prvega mešanja, ustvari vhod $c' = c_{0,j} \cdot \mathcal{E}(f)$. Tako obstaja algebraična povezava med c' in $c_{0,j}$, ki se izkazuje tudi v algebraični povezavi med pripadajočima čistopisoma. Napadalec tako po končanem drugem mešanju lahko z veliko verjetnostjo zazna par povezanih čistopisov in s tem najde čistopis, ki ustreza $c_{0,j}$. Ker Chaumova mreža ni več aktualna, se s podrobnostmi napada (kako upoštevati naključno dopolnilo pri oblikovanju f) ne ukvarjamo in jih bralec lahko pogleda v [26].

4.2.2 Mešalna mreža s ponovnim šifriranjem

Leta 1993 so Park et al. [24] predlagali prvo mešalno mrežo s ponovnim šifriranjem (angl. re-encryption mixnet). Motiviralo jih je dejstvo, da je pri mrežah na osnovi Chaumove dolžina tajnopisa sorazmerna s številom mešalnih strežnikov, saj se na vsaki plasti nova naključna vrednost spoji z obstoječim tajnopisom. Pri mešalnih mrežah s ponovnim šifriranjem se naključne vrednosti algebraično dodajo v obstoječi tajnopis. Primerna izbira je ElGamalov kriptosistem, ki, kot smo že videli, zaradi naključnosti in homomorfizma omogoča ponovno šifriranje.

Naj bo $\mathcal{M}$ mešalna mreža. Parametri mreže naj bodo veliko praštevilo p, generator g za $\mathbb{Z}_p^*$ ter faktorizacija $\varphi(p)$, s pomočjo katere lahko vsak preveri, da je g veljaven generator za $\mathbb{Z}_p^*$. Poleg tega naj vsak mešalni strežnik $\mathcal{M}_i$ generira svoj zasebni ključ

$$\mathrm{sk}_i = x_i \in_R \mathbb{Z}_{p-1}$$

in pripadajoči javni ključ

$$\mathrm{pk}_i = y_i = g^{x_i} \bmod p.$$

ElGamalov tajnopis definiramo običajno kot:

$$c = \mathcal{E}_{\mathrm{pk}}(m; r) = (\alpha, \beta) = (g^r, m \cdot y^r).$$

Skupni javni ključ mešalne mreže naj bo

$$\mathsf{PK} = \prod_{i=1}^{\ell} \mathrm{pk}_i = g^{\sum_{i=1}^{\ell} x_i}.$$

Ponovno šifriranje ElGamalovih tajnopisov izvedemo (upoštevajoč podrazdelek 2.3.1) na naslednji način:

$$\begin{aligned} \mathcal{RE}_{\mathrm{pk}}(c; r') &= c \cdot \mathcal{E}_{\mathrm{pk}}(1; r') \\ &= (\alpha \cdot g^{r'}, \beta \cdot y^{r'}) \\ &= (g^r \cdot g^{r'}, y^r \cdot y^{r'}) \\ &= (g^{r+r'}, y^{r+r'}) \\ &= \mathcal{E}_{\mathrm{pk}}(m; r + r'). \end{aligned}$$

Volivec V_j pripravi tajnopis glasovnice za mešalno mrežo tako, da zašifrira svoj glas m_j z uporabo skupnega javnega ključa PK:

$$c_{0,j} = \mathcal{E}_{\mathsf{PK}}(m_j; r_j),\ r_j \in_R \mathbb{Z}_{p-1}.$$

Mešalni strežnik $\mathcal{M}_i$ ponovno šifrira vsak tajnopis z novo naključno vrednostjo po formuli:

$$c_{i,j} = \mathcal{RE}_{\mathsf{PK}}(c_{i-1,j}; r_{i,j}).$$

Mešalni strežnik $\mathcal{M}_i$ izvede tudi tajno in naključno permutacijo π_i. Le-to izvede tako, da po opravljenem ponovnem šifriranju dobljene tajnopise leksikografsko uredi. Izhodi $c_{\ell,j}$ zadnjega mešalnega strežnika se nato odšifrirajo, pri čemer vsak mešalni strežnik izvede svoj del porazdeljenega odšifriranja s svojim zasebnim ključem sk_i.

Delno odšifriranje predlagajo Park et al. [24] kot osnovni protokol, pri katerem se hkrati s ponovnim šifriranjem izvaja tudi delno odšifriranje. Tukaj ga omenjamo kasneje, zaradi lažje razlage oz. kot nadgradnjo prejšnje različice. Definirajmo naslednji javni ključ

$$\mathsf{PK}_i = \prod_{i'=i}^{\ell} \mathrm{pk}_{i'} = g^{\sum_{i'=i}^{\ell} x_{i'}}$$

za strežnik $\mathcal{M}_i$. Ključ je enak produktu $g^{x_{i'}}$ za $i' = i, i+1, \ldots, \ell$. Potem je PK_1 kar PK in $\mathsf{PK}_\ell = \mathrm{pk}_\ell$. Vhod v mešalno mrežo je enak kot prej, operacije mešalnega strežnika pa se spremenijo. Po novem je tajnopis, ki pride

do strežnika $\mathcal{M}_i$, oblike $\mathcal{E}_{\mathsf{PK}_i}(m; r_i)$. Strežnik $\mathcal{M}_i$ lahko z uporabo svojega zasebnega ključa sk_i izvede delno odšifriranje in tako pretvori tajnopis v obliko $\mathcal{E}_{\mathsf{PK}_{i+1}}(m; r_{i+1})$. To naredi na naslednji način:

$$\mathsf{PartialDec}_{\mathrm{sk}_i}(c) = (\alpha, \beta \cdot \alpha^{-x_i}).$$

Strežniki torej izvajajo delno odšifriranje tako, da velja:

$$\mathsf{PartialDec}_{\mathrm{sk}_i}(\mathcal{E}_{\mathsf{PK}_i}(m)) = \mathcal{E}_{\mathsf{PK}_{i+1}}(m).$$

Posledično mešalni strežnik $\mathcal{M}_i$ izvede na tajnopisu $c_{i-1,j}$ naslednjo operacijo:

$$c_{i,j} = \mathcal{RE}_{\mathsf{PK}_{i+1}}(\mathsf{PartialDec}_{\mathrm{sk}_i}(c_{i-1,j}); r_{i,j}).$$

Na koncu tako dobljene tajnopise še leksikografsko uredi, kar služi kot tajna in naključna permutacija π_i.

Varnost. ElGamalov kriptosistem, kot je opisan v 2. poglavju (pred navedbo popravkov) in v [24] ni semantično varen. Iz tega tudi izhaja možen napad, ki ga preprečimo tako, da uporabljamo varno različico ElGamalovih parametrov (glej podrazdelek 2.2.1). Druga nevarnost, ki ostaja tudi pri popravljeni različici ElGamalovega kriptosistema, pa je njegov homomorfizem (ki ga sicer s pridom uporabljamo), saj je možno izvesti napad s povezanimi vhodi, ki smo ga opisali že pri Chaumovi mreži. Protiukrep za te vrste napadov so tehnike, ki se tudi sicer uporabljajo za zaščito ElGamalovih tajnopisov pred napadi z izbranim tajnopisom.

Avtorji mreže s ponovnim šifriranjem (in delnim odšifriranjem) za preverjanje s strani volivca predlagajo enak postopek, kot ga je predlagal že Chaum za svojo mrežo [9], opišemo pa ga tukaj. Oddaja glasovnice poteka v dveh korakih. V prvem koraku volivec V_j pošlje svoj "javni ključ" pk_j (seveda primerno zašifriran za mešalno mrežo) v mešalno mrežo in preveri, da se ključ nespremenjen pojavi na oglasni deski. Pri tem nihče razen volivca V_j ne ve, komu ta javni ključ pripada. V drugem koraku pa volivec V_j najprej s svojim zasebnim ključem sk_j zašifrira glasovnico m_j, ki vsebuje vnaprej določeno število ničel na koncu sporočila, in nato pošlje $(\mathrm{pk}_j, \mathcal{E}_{\mathrm{sk}_j}(m_j))$ v mešalno mrežo. Dopolnitev z ničlami je pomembna, saj prepreči, da bi nekdo ustvaril ponarejeno sporočilo, ki bi pri odšifriranju z nekim javnim ključem z oglasne deske, kot rezultat vrnilo glasovnico s točno določenim številom ničel na koncu. Vendar, kot rečeno, ta način omogoča zgolj preverljivost volivca in ne tudi splošno preverljivost.

4.3 Splošno preverljiva mešalna mreža Sako-Kilian

Za kriptografske e-volitve so posebej zanimive splošno preverljive mešalne mreže, pri katerih lahko kdorkoli preveri pravilnost delovanja. Prvo splošno preverljivo mešalno mrežo sta objavila Sako in Kilian [29].

Mešalna mreža Sako-Kilian temelji na pravkar predstavljeni mešalni mreži z delnim odšifriranjem avtorjev Park et al. [24]. Še vedno je x_i zasebni ključ mešalnega strežnika $\mathcal{M}_i$, pripadajoči delež javnega ključa je $y_i = g^{x_i}$, skupni javni ključ pa je $g^{\sum_{i=1}^{\ell} x_i}$. Ponovno šifriranje in delno odšifriranje izvajamo enako kot pri mreži Park et al. Naštejmo še glavne razlike med mrežo Park et al. in mrežo Sako-Kilian:

- Mreža Sako-Kilian uporablja varno različico ElGamalovih parametrov v $\mathbb{Z}_p$. In sicer, $p = 2q + 1$, kjer sta p in q praštevili, ter $g \in \mathbb{Z}_p^*$, tako da je $|\langle g \rangle| = q$.

- Vsak mešalni strežnik $\mathcal{M}_i$ takoj po sprejemu tajnopisov izvede delno odšifriranje in *objavi delno odšifrirane tajnopise.*

- Po delnem odšifriranju mešalni strežnik izvede še ponovno šifriranje in permutiranje tajnopisov ter objavi **dokaz pravilnega delnega odšifriranja** in **dokaz pravilnega ponovnega šifriranja in permutiranja**.

Sedaj opišimo omenjena dokaza.

Dokaz pravilnega delnega odšifriranja. Naj bo $\mathcal{M}_i$ mešalni strežnik in naj bo $c = (\alpha, \beta)$ nek vhodni tajnopis za ta strežnik. Naj bo $\mathsf{PartialDec}(c) = (\alpha', \beta')$ delno odšifriran c. Oba tajnopisa sta javna, zato lahko kdorkoli izračuna $\beta/\beta' \equiv \alpha^{x_i} \equiv g^{rx_i} \pmod{p}$, kjer je x_i zasebni ključ strežnika $\mathcal{M}_i$.

Mešalni strežnik $\mathcal{M}_i$ mora potem dokazati, da je pri delnem odšifriranju res uporabil svoj zasebni ključ x_i. Z drugimi besedami, $(y_i, \alpha, \beta/\beta') = (g^{x_i}, g^r, g^{rx_i})$ mora biti veljavna Diffie-Hellmanova trojica oz. veljati mora

$$\log_g(y_i) \equiv \log_\alpha(\beta/\beta') \pmod{p}.$$

Sako in Kilian predlagata dokaz, ki ga prikazuje protokol 4.1.

Protokol 4.1 Sako-Kilianov dokaz pravilega delnega odšifriranja

1: Primož izbere $r \in_R \mathbb{Z}_q$ in izračuna $y_r = g^r$ ter $\alpha_r = \alpha^r$. Nato pošlje par (y_r, α_r) Veri.
2: Vera naključno izbere eno od naslednjih dveh možnosti:

- Izzove Primoža, da ji odgovori z r. Če r ustreza (y_r, α_r), Vera dokaz sprejme, sicer ga zavrne.
- Izzove Primoža, da ji odgovori z $s = r - x_i \bmod p$. Če je $y_r \equiv g^s \cdot y_i \pmod p$ in $\alpha_r \equiv (\beta/\beta') \cdot \alpha^s \pmod p$, Vera dokaz sprejme, sicer ga zavrne.

Trditev 4.3.1. *Sako-Kilianov dokaz pravilnega delnega odšifriranja je ZKP. Verjetnost, da bi nepošteni Primož uspešno prepričal Vero, je* $1/2$.

Dokaz. **Protokol je poln**. Če velja enakost $\log_g(y_i) \equiv \log_\alpha(\beta/\beta') \pmod p$ in če Primož pozna x_i ter izvaja protokol pošteno, potem bo Vera vedno sprejela dokaz. Če Vera v drugem koraku protokola zahteva, da Primož razkrije r, potem bo uspešno preverila, da je bila vrednost r uporabljena pri izračunu para (y_r, α_r), saj je Primož pošteno izvajal protokol. Če pa bo Vera v drugem koraku zahtevala, da Primož razkrije $r' = r - x_i$, potem bo uspešno preverila, da je

$$y_r \equiv g^r \equiv g^{r-x_i} \cdot g^{x_i} \equiv g^{r'} \cdot y \pmod p$$

in

$$\alpha_r \equiv \alpha^r \equiv \alpha^{x_i} \cdot \alpha^{r-x_i} \equiv (\beta/\beta') \cdot \alpha^{r'} \pmod p.$$

Protokol je uglašen. Recimo, da strežnik $\mathcal{M}_i$ pri delnem odšifriranju ni uporabil ključa x_i. To pomeni, da enakost ne drži, torej $\log_g(y_i) \not\equiv \log_\alpha(\beta/\beta') \pmod p$. Primož lahko prelisiči Vero, če mu uspe vnaprej uganiti, za katero od dveh možnosti se bo Vera odločila pri drugem koraku protokola.

Če misli, da ga bo Vera izzvala, da ji razkrije r, potem mora zgolj pravilno izračunati par (y_r, α_r) za poljuben r. V tem primeru bo Vera sprejela dokaz, saj bo vrednost r ustrezala paru (y_r, α_r).

Če pa Primož misli, da ga bo Vera izzvala, da ji razkrije s, potem izračuna $y_r = g^s \cdot y_i \bmod p$ in $\alpha_r = (\beta/\beta') \cdot \alpha^s \bmod p$. Vera bo sprejela dokaz, saj bosta vrednosti (y_r, α_r) očitno ustrezali zahtevani enakosti.

Ker Vera naključno izzove Primoža z eno od dveh možnosti, je verjetnost, da bo Primož uganil, katero možnost bo izbrala Vera, in jo prelisičil, enaka $1/2$.

Protokol je brez razkritja znanja. Preverimo, če Vera iz dokaza res ne

prejme nobene informacije o tajni vrednosti x_i. Vera od Primoža med izvajanjem protokola prejme r ali $s = r - x_i \bmod p$. Izmed slednjih je vrednost x_i vsebovana zgolj v odgovoru s. Da bi Vera lahko iz s dobila vrednost x_i, bi morala uganiti naključno vrednost r, ki jo pozna zgolj Primož, ali pa izračunati $r = \log_g y_r \bmod p$ oz. $r = \log_\alpha \alpha_r \bmod p$, kar pa je po predpostavki pretežko. □

Če protokol 4.1 ponovimo t-krat, verjetnost, da bi nepošteni Primož prepričal Vero, zmanjšamo na 2^{-t}. Dokaz je zapisan v interaktivni obliki, z uporabo hevristike Fiat-Shamir pa odpravimo interaktivnost. Namesto tega dokaza lahko uporabimo tudi Chaum-Pedersenov dokaz, ki smo ga opisali v 3. poglavju in je visoko uglašen.

Sako in Kilian pokažeta, da se lahko dokaze posameznih tajnopisov združi v enega. Primož potencira vse α_j in β_j/β'_j na naključno potenco e_j, ki jo prejme od Vere. Nato pomnoži skupaj vse $\alpha_j^{e_j}$ in $(\beta_j/\beta'_j)^{e_j}$, ter dokaže enakost

$$\prod_{j=1}^{N} \alpha_j^{e_j} = \prod_{j=1}^{N} (\beta_j/\beta'_j)^{e_j},$$

kjer je N število volivcev oz. število vhodov v mešalno mrežo.

Dokaz pravilnega ponovnega šifriranja in permutiranja. Naj bo π permutacija in $(r_1, \ldots, r_N)$ vektor naključnih vrednosti, ki jih je strežnik $\mathcal{M}_i$ uporabil pri izvajanju, katerega pravilnost dokazujemo. Vrednost r_j, za $j = 1, \ldots, N$, je naključna vrednost uporabljena za ponovno šifriranje j-tega vhoda. Mešalni strežnik (Primož) dokaže pravilno ponovno šifriranje in permutiranje s pomočjo protokola 4.2.

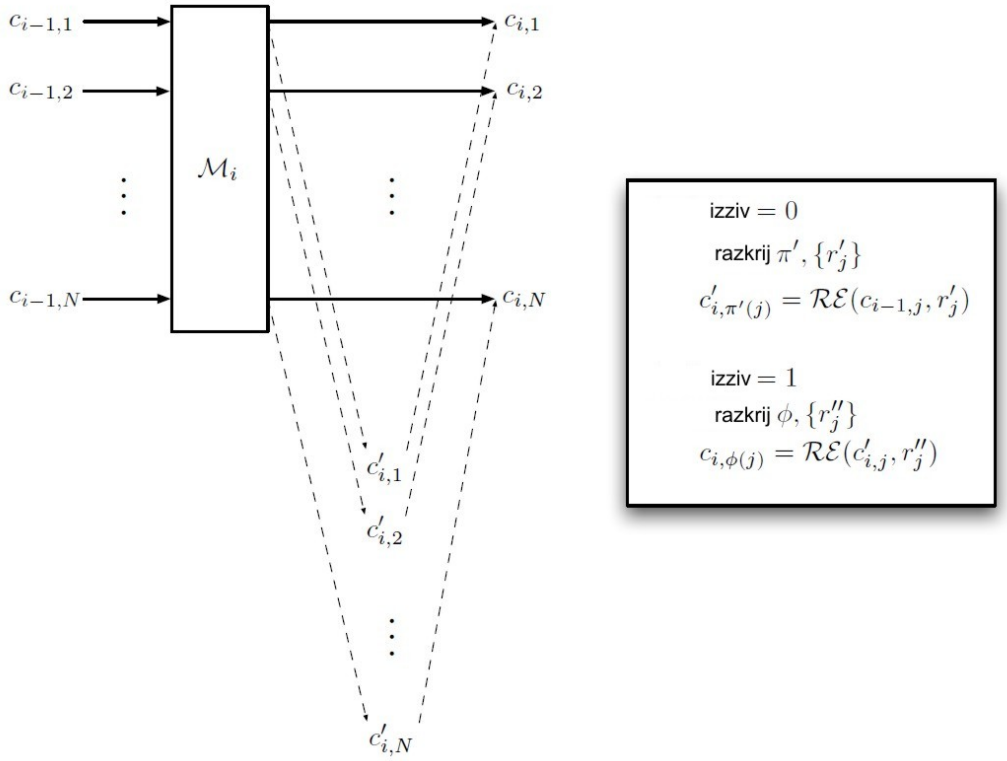
Protokol 4.2 Sako-Kilianov dokaz o pravilnem ponovnem šifriranju in permutiranju

1: Primož naključno generira novo permutacijo λ in nov vektor naključnih vrednosti $(t_1, \ldots, t_N)$. Nato s temi novimi vrednostmi izvede permutiranje in ponovno šifriranje.
2: Vera naključno izbere eno od dveh možnosti:

- Zahteva vrednosti $(\lambda, (t_1, \ldots, t_N))$ in preveri pravilnost drugega permutiranja in ponovnega šifriranja.
- Zahteva vrednosti $(\lambda \circ \pi^{-1}, (r_1 - t_1, \ldots, r_N - t_N))$ in preveri, da po permutiranju in ponovnem šifriranju izhodov drugega mešanja z uporabo $(\lambda \circ \pi^{-1}, (r_1 - t_1, \ldots, r_N - t_N))$ dobi izhode prvega mešanja.

Trditev 4.3.2. *Sako-Kilianov dokaz pravilnega ponovnega šifriranja in mešanja je ZKP. Verjetnost, da bi nepošteni Primož uspešno prepričal Vero, je* $1/2$.

Pri dokazovanju postopamo podobno kot pri prejšnjih ZKP. Delovanje protokola 4.2 prikazuje slika 4.2. Tudi pri tem dokazu interaktivnost odpravimo s pomočjo hevristike Fiat-Shamir.



Slika 4.2: Prikaz dokaza pravilnega permutiranja in ponovnega šifriranja pri mreži Sako-Kilian. Permutacija ϕ označuje permutacijo $\lambda \circ \pi^{-1}$, naključna vrednost r''_j pa označuje naključno vrednost $r_j - t_j$.

Če želimo doseči uglašenost $1 - 2^{-160}$, dokaza pravilnosti zahtevata $640N$ modularnih potenciranj za posamezen mešalni strežnik. Kot smo videli, vsak od obeh dokazov zahteva 2 modularni potenciranji. Uglašenost enega izvajanja protokola je $1/2$, torej protokol izvedemo 160-krat za vsak vhod v mrežo, skupaj torej $640N$ modularnih potenciranj na strežnik.

Varnost. Sako-Kilian mešalna mreža temelji na mreži avtorjev Park et al. in tudi uporablja ElGamalov kriptosistem, le da v različici z varnejšimi parametri. Zato je še vedno podvržena napadom s povezanim vhodom, ki smo

jih že opisali. Kot protiukrep smo navedli tehnike za doseganje varnosti pred napadi z izbranim tajnopisom. Sako in Kilian pa dodatno predlagata, da bi volivci na oglasni deski skupaj s tajnopisi glasovnic objavili dokaze o poznavanju čistopisa. Napadalec, ki kreira povezan tajnopis, ne pozna pripadajočega čistopisa, pozna le povezavo, ki jo potem išče v parih čistopisov, zato napadalec ne bi mogel izračunati dokaza o poznavanju čistopisa. Primer takega dokaza za ElGamalov kriptosistem je Schnorrov dokaz iz 3. poglavja.

Z vidika napadov nepoštenih mešalnih strežnikov si ponavadi želimo, da je tajnost ohranjena, če je vsaj en strežnik pošten.

Trditev 4.3.3. *Mešalna mreža Sako-Kilian ne ohranja tajnosti, če je zgolj en mešalni strežnik pošten.*

Dokaz. Težava je v tem, da so delno odšifrirani tajnopisi objavljeni na oglasni deski, preden so premešani in ponovno šifrirani. Če so potem vsi strežniki razen enega nepošteni, ti nepošteni vzamejo delno odšifrirane tajnopise poštenega strežnika in potem skupaj odšifrirajo glasovnice, saj je edino, do česar nimajo dostopa, zasebni ključ poštenega strežnika. Dostopa do permutacijske funkcije in naključnih vrednosti poštenega strežnika pa ne potrebujejo, saj pošteni strežnik še ni izvedel permutiranja in ponovnega šifriranja na objavljenih delno odšifriranih tajnopisih. □

Trditev 4.3.4. *Mešalna mreža Sako-Kilian, pri kateri mešalni strežniki najprej izvedejo permutiranje in ponovno šifriranje ter nato delno odšifriranje, ohranja tajnost, če je vsaj en mešalni strežnik pošten.*

Dokaz. V tem primeru nepošteni mešalni strežniki še vedno lahko odšifrirajo glasovnice, ne poznajo pa permutacije poštenega strežnika in zato ne morejo povezati čistopisov s tajnopisi. □

4.4 Mešalne mreže z učinkovitimi dokazi

Do sedaj opisane mešalne mreže nudijo splošno preverljivost in so tako z vidika funkcionalnosti primerne za uporabo pri e-volitvah. Težavo pa predstavlja učinkovitost teh mrež. Splošna preverljivost zahteva dokaze pravilnosti, ki so računsko zelo zahtevni za večje število vhodov. Primer iz [2] nam pove, da je na sodobnem prenosniku dokaz pravilnega mešanja za 500 glasovnic trajal 3 ure. Praštevilo p je bilo reda 1024 bitov.

Zato so se raziskave na področju mešalnih mrež usmerile v iskanje učinkovitih dokazov pravilnosti. Ker so te mreže precej bolj zapletene, v nalogi pa jim namenjamo le eno poglavje, bomo podali kratke opise najbolj zanimivih.

Na začetku na kratko navedemo in opišemo ideje pri dveh najhitrejših mešalnih mrežah z visoko uglašenostjo dokaza. Zadnja opisana mreža oz. tehnika dokazovanja za mešalne mreže pa utemelji pravilnost mešanja s sorazmerno visoko, vendar ne najvišjo verjetnostjo. Avtorji so pri tem načinu dokazovanja naredili kompromis med hitrostjo na eni, ter uglašenostjo in tajnostjo na drugi strani.

4.4.1 Hitre mešalne mreže z visoko uglašenostjo

Leta 2001 sta Furukawa in Sako [15] predstavila mešalno mrežo, pri kateri postopek mešanja obravnavata kot matrično množenje, dokaz pa brez rakritja znanja dokaže poznavanje matrike in naključnih vrednosti za ponovno šifriranje ter dejstvo, da je, matrika uporabljena za mešanje, res permutacijska matrika. Dokaz pravilnosti zahteva $18n$ modularnih potenciranj [15] za posamezen mešalni strežnik, pri čemer je n število vhodov v mrežo.

Neff [22] je leta 2001 predstavil splošno preverljivo in popolnoma tajno mešalno mrežo, ki uporablja do sedaj najhitrejši dokaz pravilnosti. Opišimo način mešanja in idejo dokaza.

Naj bo $\mathcal{M}_i$ posamezen mešalni strežnik z vhodi (α_j, β_j) in pripadajočimi izhodi (α'_j, β'_j) ter permutacijo π. Strežnik meša in ponovno šifrira vhode po naslednji formuli:

$$(\alpha'_j, \beta'_j) = (g^{r_{\pi(j)}}\alpha_j, y^{s_{\pi(j)}}\beta_j).$$

Bistvo dokaza iz [22] je, da je mešanje izvedeno pravilno, kadar je $(r_1, \ldots, r_N) = (s_1, \ldots, s_N)$, pri čemer je N število vhodov v mešalno mrežo. Ko imamo enkrat znane vhode in izhode posameznega mešalnega strežnika, preverjamo enakost omenjenih vektorjev tako, da izberemo naključni vektor $(t_1, \ldots, t_N)$ preverimo enakost $(r_1, \ldots, r_N)\cdot(t_1, \ldots, t_N) = (s_1, \ldots, s_N)\cdot(t_1, \ldots, t_N)$. Če slednja enakost velja, potem velja tudi $(r_1, \ldots, r_N) = (r_1, \ldots, r_N)$ z visoko verjetnostjo. Dokaz pravilnosti zahteva reda $8N$ modularnih potenciranj [22] za posamezen mešalni strežnik.

4.4.2 RPC preverjanje

Leta 2002 so Jakobsson, Juels in Rivest [21] predstavili **naključno delno preverjanje** (angl. randomized partial checking, kratica RPC). V tem primeru ne gre za mešalno mrežo, temveč dokazovalni sistem za mešalne mreže. Posledično RPC preverjanje ni odvisno od uporabljenega kriptosistema ali načina delova-

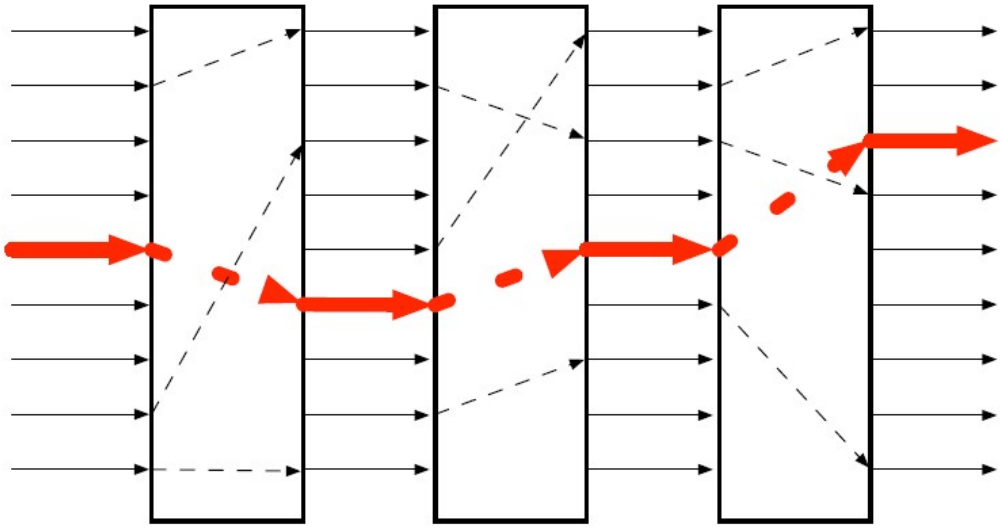
nja mešalne mreže. RPC odlikuje tudi učinkovitost, saj dokazovanje pravilnosti zahteva manj procesiranja kot samo mešanje, vendar na račun uglašenosti.

RPC dokaže pravilnost delovanja mešalne mreže s sorazmerno visoko, vendar ne najvišjo, verjetnostjo. Ideja je, da vsak mešalni strežnik razkrije naključno polovico povezav med vhodi in izhodi. Ob primerni izvedbi, tak način ohranja zasebnost volivcev, obenem pa nam da visoko zaupanje, da je bilo mešanje izvedeno pravilno. Opišimo RPC malo podrobneje.

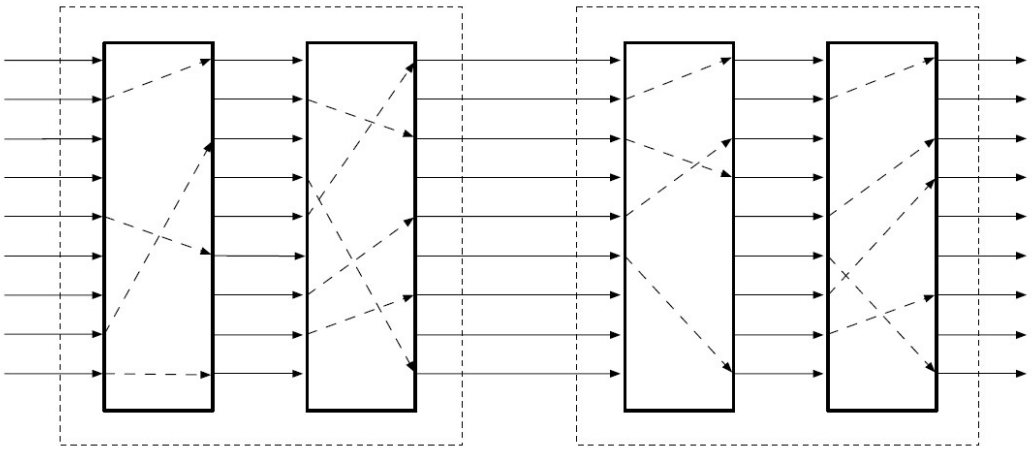
Naj bo $\mathcal{M}$ mešalna mreža, lahko kar mreža avtorjev Park et al. (glej podrazdelek 4.2.2). Ko mešalni strežnik $\mathcal{M}_i$ vrne izhode $c_{i,j}$, izračuna in objavi zaveze k vrednostim $(j, \pi_i(j), r_{i,j})$ za vsak izhod. Zavezo za $(j, \pi_i(j), r_{i,j})$ izračuna preprosto tako, da naključno izbere niz bitov w in izračuna, $H(w||j||\pi_i(j)||r_{i,j})$, kjer je H kriptografska zgoščevalna funkcija, $||$ pa pomeni spoj binarnih nizov. Z vrednostmi $(j, \pi_i(j), r_{i,j})$ se lahko preveri, ali je strežnik $\mathcal{M}_i$ res preslikal $c_{i-1,j}$ v $c_{i,\pi_i(j)}$, vendar na tej točki te vrednosti niso javne, javne so zgolj zaveze k tem vrednostim. Ko mešalni strežnik objavi izhode in zaveze, Vera pošlje strežniku naključno izbrano polovico izzivalnih vhodov, strežnik pa vrne vrednosti $(j, \pi_i(j), r_{i,j})$ za te vhode, s katerimi Vera preveri, ali je bilo mešanje izvedeno korektno. Da je strežnik vrnil pravilne vrednosti $(j, \pi_i(j), r_{i,j})$ za izzvane vhode, nam zagotavljajo prej objavljene zaveze k vrednostim $(j, \pi_i(j), r_{i,j})$ za vse vhode.

Pri majhnem številu strežnikov in naključnem izboru preverjanih vhod-izhod povezav se lahko zgodi, da se razkrije celotna pot glasovnice in s tem popolnoma izgubi anonimnost. Primer prikazuje slika 4.3. Rešitev je v parjenju zaporednih mešalnih strežnikov. Prvi par tvorita $(\mathcal{M}_1, \mathcal{M}_2)$, drugi par $(\mathcal{M}_3, \mathcal{M}_4)$ itd. Nato naključno razdelimo sporočila med obema strežnikoma znotraj para na 2 enako veliki množici. S prvo množico sporočil izzovemo prvi strežnik znotraj para, z drugo množico pa drugega. S tem preprečimo, da bi se za katerokoli sporočilo razkrila pot od vhoda v prvi strežnik para do izhoda drugega strežnika v paru. To prikazuje slika 4.4. Če predpostavimo, da je več kot polovica mešalnih strežnikov poštenih, potem je vsaj en par strežnikov pošten, s tem pa je ohranjena anonimnost vsake glasovnice.

Preverjanje zahteva n modularnih potenciranj (2 potenciranji za $n/2$ tajnopisov) za posamezen mešalni strežnik, pri čemer je n število vhodov v mrežo. Vsak mešalni strežnik preveri pravilnost mešanja za neko glasovnico z verjetnostjo $\frac{1}{2}$, saj za preverjanje naključno izberemo polovico glasovnic. Verjetnost, da RPC pri posameznem mešalnem strežniku ne zazna v goljufivih glasovnic, je 2^{-v}. Predlagano je, da bi pri volitvah en par mešalnih strežnikov pripadal eni organizaciji. S tem bi bila anonimnost zagotovljena, če bi bila vsaj ena taka organizacija poštena.



Slika 4.3: Prikaz razkritja celotne poti, kadar je število strežnikov majhno, izbira preverjanih vhod-izhod povezav pa naključna.



Slika 4.4: Prikaz RPC preverjanja s parjenjem zaporednih mešalnih strežnikov, s čimer preprečimo razkritje celotnih poti.

Poglavje 5

Scratch & Vote

5.1 Uvod

V tem poglavju opišemo shemo za kriptografske e-volitve Scratch & Vote (s kratico SV), ki sta jo leta 2006 predstavila Adida in Rivest [4]. Shema SV je namenjena glasovanju na voliščih in ima naslednje lastnosti:

- **Papirne glasovnice**. Glasovnice so natisnjene na papirju, izpolnimo jih s pisalom.
- **Samostojno preverjanje glasovnice**. Glasovnice vsebujejo vse potrebne informacije za preverjanje pravilnosti.
- **Homomorfno štetje**. Ne uporabljamo mešalnih mrež temveč homomorfno štetje glasov, ki ga kdorkoli lahko ponovi, saj je postopek sorazmerno preprost.

Slika 5.1: Primer neizpolnjene (levo) in izpolnjene (desno) glasovnice pri Scratch & Vote.

Volitve pri SV z vidika volivca potekajo po naslednjih korakih:

1. **Prijava**. Volivec se prijavi na volišču in si naključno izbere glasovnico, ki naj je volilni uradnik ne vidi. Glasovnica je naluknjana navpično po sredini, na levi strani je seznam kandidatov, na desni strani pa okenca, kjer volivec označi izbiro. Takoj pod okenci se nahaja 2D črtna koda v formatu PDF417. Pod črtno kodo pa se nahaja odstranljiva površina, ki je od preostanka desnega dela ločena s še eno naluknjano črto. Na površini je opozorilo, da glasovnica postane neveljavna, če se površino podrgne stran. Glasovnico prikazuje slika 5.1.

2. **Preverjanje (opcijsko)**. Volivec se lahko odloči za preverjanje glasovnice. Naključno izbere še eno glasovnico, odstrani površino na dnu in glasovnico preda organizaciji, ki ji zaupa in ki ima opremo, da preveri pravilnost te glasovnice. Podrobnosti bomo opisali kasneje, pomembno je, da se pri 1 oddani in 1 preverjeni glasovnici volivec prepriča z verjetnostjo $\frac{1}{2}$, da je bila tudi oddana glasovnica pravilna. Če vsak volivec preveri eno glasovnico, potem je bila preverjena naključna polovica vseh glasovnic, kar pomeni, da so ostale glasovnice pravilne z visoko verjetnostjo.

3. **Oddaja glasu**. Volivec z glasovnico stopi v kabino, označi svojo izbiro in odtrga levo polovico glasovnice, na kateri so zgolj imena kandidatov v naključnem vrstnem redu. V kabini je škatla, kamor lahko anonimno odvrže levo polovico. Nato volivec stopi do volilnega uradnika in mu izroči desno polovico glasovnice. Uradnik preveri, da je odstranljiva površina nedotaknjena in nato ta spodnji del glasovnice odtrga in zavrže v pričo volivca in ostalih prisotnih. Volivec nato preostanek glasovnice (okenca z oznako in črtna koda) vstavi v napravo za zajem glasovnice, ki prebere črtno kodo in zapiše številko okenca, kjer je bil označen izbor. Nato volivec glasovnico kot potrdilo vzame s seboj.

4. **Preverjanje**. Volivec se doma prijavi na spletno stran volitev in poišče svojo glasovnico, ter preveri, da se ujema s potrdilom, ki ga ima doma. Za preverjanje črtne kode potrebuje primerno opremo, v praksi pa bi verjetno neodvisne organizacije ponujale preverjanje glasovnic.

5.2 Kriptografski gradniki

SV shema uporablja Paillierjev kriptosistem, ki smo ga spoznali v 2. poglavju. Kriptosistem izkazuje aditivni homomorfizem, torej velja

$$\mathcal{E}_{\text{pk}}(m_1) \cdot \mathcal{E}_{\text{pk}}(m_2) = \mathcal{E}_{\text{pk}}(m_1 + m_2).$$

Za razliko od eksponentnega ElGamalovega kriptosistema, ki ima zaradi računanja diskretnega logaritma omejen prostor čistopisov, Paillierjev kriptosistem nima te omejitve, kar je pomembno zaradi vrste homomorfnega števca, ki ga SV uporablja.

Pragovno odšifriranje. Kot smo omenili že v 2. poglavju, Paillierjev kriptosistem omogoča učinkovito izvedbo pragovnega odšifriranja in tudi učinkovito izvedbo porazdeljenega generiranja ključev.

Homomorfno štetje. Homomorfno štetje glasov sta v osnovni obliki uvedla Benaloh in Fischer [7] in smo ga predstavili že v 2. poglavju. Če pa bi lahko imeli v čistopisu ene glasovnice npr. 20 števcev, potem tako glasovnico lahko uporabimo za 1 vprašanje z 20 možnimi odgovori, 2 vprašanji z 10 odgovori, 10 vprašanj z 2 odgovoroma itd. Tako tehniko so predstavili Baudron et al. [5] in jo uporabimo tudi pri SV shemi. Baudron et al. uporabijo Paillierjev kriptosistem, lahko pa bi uporabili tudi kak drug kriptosistem, če je le aditivno homomorfen in pri tem omogoča učinkovito odšifriranje.

Naj bo prostor čistopisov velikosti b bitov, torej vsak čistopis je dolg b bitov. Prostor tajnopisov je prav tako velikosti b bitov. Naj bo T tajnopis dolg b bitov, ki predstavlja skupni števec. Naj bo k število števcev, ki jih želimo imeti znotraj skupnega števca T, pri čemer želimo, da vsak števec lahko šteje do vrednosti $2^M - 1$. Bistvo tehnike je, da znotraj b bitov skupnega števca T izberemo k blokov dolžine M bitov. Da je to možno, mora biti $b \geq kM$. Pomembno je, da je $2^M - 1$ dovolj veliko število, sicer pride do preliva bitov v naslednji števec. Če želimo števec $j \in [1, k]$ povečati za vrednost t_j, potem skupnemu števcu T prištejemo vrednost $t_j \cdot 2^{(j-1)M}$. Pri volitvah želimo števcem prištevati vrednost 1. Da bi torej za 1 povečali vrednost števca j znotraj skupnega števca T, izvedemo naslednjo operacijo:

$$T' = T \cdot \mathcal{E}_{\text{pk}}(2^{(j-1)M}),$$

kjer je T' nov tajnopis skupnega števca, v katerem je števec j povečan za 1 glede na vrednost v T. Pri tem je ključna IND-CPA varnost Paillierjevega

kriptosistema, saj iz $\mathcal{E}_{\text{pk}}(2^{(j-1)M})$ ne moremo razbrati vrednosti j oz. ugotoviti, kateri števec je bil povečan.

Dokazi pravilnosti. Pri shemah s homomorfnim štetjem je potrebno imeti zagotovilo, da glasovnica prispeva natanko en glas za vsakega možnega izbranega kandidata. Pri SV to pomeni, da mora tajnopisu c_j, ki poveča števec kandidata $j \in [1, k]$ za 1, ustrezati čistopis $2^{(j-1)M}$. Označimo tak čistopis z m_j. Kot bomo videli, ima SV glasovnica k šifriranih čistopisov m_j, po enega za vsak $j \in [1, k]$. Dokazati moramo torej, da množica tajnopisov $(c_1, \ldots, c_k)$ šifrira neko permutacijo (vrstni red je zaradi varnosti naključen) množice $(m_1, m_2, \ldots, m_k)$. Pri tem dokaz seveda ne sme izdati, katero od k vrednosti v resnici šifrira določen tajnopis. Za ta namen uporabimo uporabimo dokaz iz [5]. Ta Primožu omogoča, da Vero prepriča, da Paillierjevemu tajnopisu c ustreza natanko eden od čistopisov iz množice $\mathcal{S}$. Omenjeni dokaz je disjunktivna različica dokaza, da Paillierjevemu tajnopisu c ustreza čistopis m. Bistvo disjunktivnih dokazov je, da Primož simulira dokaze za tiste vrednosti čistopisov, ki ne ustrezajo tajnopisu, in da pravilno izvede dokaz za tisto vrednost, ki ustreza tajnopisu.

Naj bo $\mathcal{S} = \{m_1, \ldots, m_k\}$ in $i \in \{1, \ldots, k\}$. Dokaz, da je $c = \mathcal{E}_{\text{pk}}(m_i)$, prikazuje protokol 5.1. Protokol je zapisan v interaktivni obliki, ki jo lahko odpravimo s hevristiko Fiat-Shamir. Operacija $a \div b$ pomeni celoštevilski količnik pri deljenju a z b.

Protokol 5.1 Dokaz, da Paillierjev tajnopis šifrira čistopis iz množice $\mathcal{S}$

1: Primož izbere $\rho \in_R \mathbb{Z}_n^*$. Naključno izbere tudi $k-1$ vrednosti $e_1, \ldots, e_{i-1}, e_{i+1}, \ldots, e_k$ iz $\mathbb{Z}_n$ in $k-1$ vrednosti $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k$ iz $\mathbb{Z}_n^*$. Primož izračuna $u_i = \rho^n \bmod n^2$ in $u_j = v_j^n (g^{m_j}/c)^{e_j} \bmod n^2$, $j \neq i$. Na koncu Primož pošlje zaveze $u_1, \ldots, u_k$ Veri.

2: Vera naključno izbere izziv $e \in_R \mathbb{Z}_n$ in ga pošlje Primožu.

3: Primož izračuna $e_i = e - \sum_{j \neq i} e_j \bmod n$. Nato izračuna

$$v_i = \rho r^{e_i} g^{e - \sum_{j \neq i} e_j \div n} \bmod n$$

in pošlje odgovor $\{v_j, e_j\}_{j \in \{1,\ldots,k\}}$ Veri.

4: Vera sprejme dokaz, če je $e \equiv \sum_j e_j \pmod{n}$ in

$$v_j^n \equiv u_j (c/g^{m_j})^{e_j} \pmod{n^2}$$

za $j \in \{1, \ldots, k\}$. Sicer, Vera dokaz zavrne.

Opisani dokaz je visoko uglašen. Verjetnost, da bi nepošteni Primož uspešno prepričal Vero, je $1/n$.

Z uporabo omenjenega dokaza zagotovimo, da se števec za nekega kandidata lahko poveča zgolj za 1. Še vedno pa s tem nismo dokazali, da vsak tajnopis šifrira natanko eno izmed vrednosti $2^{(j-1)M}$. To naredimo tako, da dokažemo, da zmnožek tajnopisov c_j šifrira vsoto čistopisov m_j za $j \in [1, k]$. Za ta namen uporabimo že omenjeni osnovni dokaz, da Paillierjevemu tajnopisu c ustreza čistopis m. Podobno kot protokol 6.1 iz 6. poglavja prikazuje disjunktivno različico Chaum-Pedersenovega dokaza, protokol 5.1 prikazuje disjunktivno različico dokaza, da Paillierjevemu tajnopisu c ustreza čistopis m. Zato slednjega ne navajamo, saj se ga da razbrati iz protokola 5.1.

5.3 Opis sheme

Kot smo že omenili, bi lahko poljubno prirejali uporabo števcev, zaenkrat pa predpostavimo, da imamo eno vprašanje s k možnimi kandidati (odgovori), torej k števcev znotraj skupnega števca. Volilni uradniki objavijo seznam kandidatov in jim priredijo oznake od 1 do k, torej priredijo števec $j \in [1, k]$ kandidatu. Nato uradniki porazdeljeno generirajo par ključev in shranijo vsak svoj delež zasebnega ključa. Izbere se še dolžina posameznega števca v bitih, torej tak M, da je 2^M več kot je število volilnih upravičencev. Poskrbi se tudi za zadostno dolžino čistopisov v bitih, tako da je $b \geq kM$. Kot javne parametre volitev se objavi

$$(\mathrm{pk}, M, (\mathrm{kandidat}_1, \ldots, \mathrm{kandidat}_k)).$$

Primer. Imejmo 4 kandidate: Ana, Bojan, Ciril in Dejan z oznakami od 1 do 4 po abecedi. Recimo, da je volilnih upravičencev 200 milijonov, kar je dovolj za skoraj vse države, in se odločimo za $M = 28$, saj je $28 > \log_2(2 \times 10^8)$. Potrebujemo torej čistopise dolžine $28 \times 4 = 112$ bitov, kar pa ni težava, saj so čistopisi pri PKC tipično še daljši.

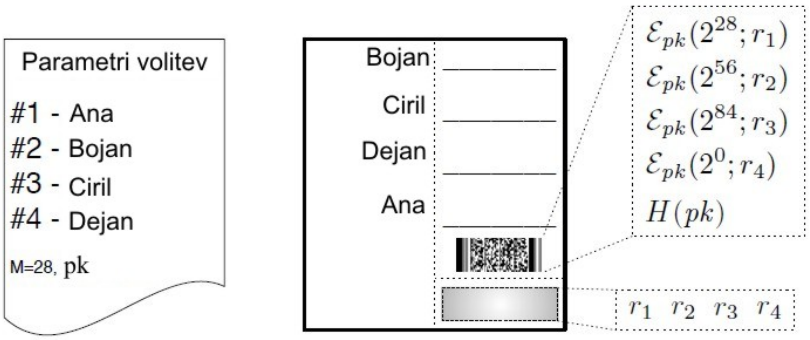
Glasovnica. Glasovnico smo opisali že na začetku, vendar brez razlage. Oglejmo si glasovnico na sliki 5.2, ki prikazuje primer, ki smo ga ravnokar navedli. Naključna razvrstitev kandidatov na sliki je $2, 3, 4, 1$, saj so številke kandidatom prirejene po abecedi. 2D črtna koda po vrsti zapiše tajnopise, tako kot so prikazani na sliki. Pomembno je, da tajnopis na prvem mestu ustreza povečanju števca kandidata, ki je naveden na glasovnici na prvem mestu. V tem primeru je to kandidat 2, zato je na prvem mestu tajnopis $\mathcal{E}_{\mathrm{pk}}(2^{(2-1)28}; r_1)$,

ki poveča 2. števec. Na drugem mestu je kandidat 3, zato je drugi tajnopis $\mathcal{E}_{\mathrm{pk}}(2^{(3-1)28}; r_2)$ itd.

Preverjanje. Kot smo že omenili uradniki poskrbijo za objavo dokazov brez razkritja znanja, ki zagotavljajo, da so tajnopisi na glasovnici pravilno oblikovani. Ker so ti dokazi predolgi, da bi bili natisnjeni na glasovnici, se jih objavi na oglasni deski, skupaj z neko identifikacijsko številko glasovnice in tajnopisi, ki so na glasovnici.

Vendar pa ti dokazi ne dokazujejo, da vrstni red kandidatov na glasovnici res ustreza vrstnemu redu tajnopisov, ki so zapisani v črtni kodi. Zaradi tega SV omogoča še dodatno preverjanje volivca. Le-to je odgovor sheme SV na problem *zagotovitve oddaje*.

Pod odstranljivo površino na dnu glasovnice so zapisane naključne vrednosti, kot prikazuje slika 5.2. S tem lahko preverimo, da je glasovnica pravilno oblikovana. SV predlaga, da vsak volivec izbere 2 glasovnici naključno in nato izmed obeh naključno izbere tisto, ki jo bo preveril. Na izbrani glasovnici volivec odstrani površino na dnu in tako dobi naključne vrednosti, ki so se uporabile pri šifriranju tajnopisov. Vrednost r_i pripada tajnopisu, ki poveča števec kandidata na i-tem mestu na glasovnici in ne kandidata z oznako i (razen, kadar se vrednosti slučajno ujemata). Volivec nato po vrstnem redu iz glasovnice, za vsakega kandidata pripravi primerno vrednost, ki poveča njegov števec (upoštevajoč kandidatovo oznako j), in jo zašifrira z uporabo pravkar razodete naključne vrednosti. Dobljeni tajnopis se mora ujemati s tajnopisom v črtni kodi. Ker seveda volivec v praksi nima pri sebi naprave za šifriranje in branje črtnih kod, to izvajajo npr. politične stranke ali neki drugi aktivisti, ki bi ponujali preverjanje na voliščih. Volivec da lahko glasovnico preverjati



Slika 5.2: Parametri volitev in glasovnica pri shemi SV.

večim organizacijam, pri tem pa nobena ne dobi informacije o tem, kako je volivec volil.

Kot smo omenili že v uvodu, če vsak volivec izbere dve glasovnici in eno preveri, potem je bila preverjena naključna polovica glasovnic. Verjetnost, da bi bilo med oddanimi glasovnicami t goljufivih, je manjša od 2^{-t}.

Oddaja glasovnice. Volivec v kabini označi izbiro ter odstrani in anonimno zavrže levi del glasovnice. S preostankom pride do volilnega uradnika, ki preveri nedotaknjenost odstranljive površine, ki skriva naključne vrednosti, uporabljene pri šifriranju. To je pomembno, saj bi imel v nasprotnem primeru volivec možnost nekomu dokazati, kako je glasoval, s tem da bi pogledal naključno vrednost, ki ustreza njegovi izbiri, z njo izračunal tajnopis za ustreznega kandidata in pokazal, da se ujema s tajnopisom iz črtne kode. Po tem, ko uradnik preveri nedotaknjenost odstranljive površine, ta del glasovnice odstrani in anonimno zavrže v pričo ostalih in volivca. Volivec nato vstavi preostanek glasovnice v napravo za zajem, ki prebere črtno kodo in pozicijo izbora ter to, skupaj z identiteto volivca, objavi na oglasni deski. Pri tem je pomembno, da naprava za zajem glasovnice, ne ve, komu gre glas. Naprava zgolj ugotovi, da je označen kandidat na j-tem mestu, v našem primeru je $j \in [1, 4]$. Volivec obdrži ta preostanek glasovnice kot potrdilo, s katerim lahko preveri, da je njegova glasovnica pravilno prispela na oglasno desko.

Štetje glasov. Za vsako glasovnico na oglasni deski volilni uradniki in ostali opazovalci preverijo dokaze o pravilnosti glasovnice. Če je dokaz v redu, se iz črtne kode prebere tajnopis, ki ustreza izboru – j-ti tajnopis za izbrano j-to mesto. Tajnopis se nato pomnoži k homomorfnemu skupnemu števcu in tako povzroči povečanje števca za izbranega kandidata. Ko so vse glasovnice tako obdelane, se zbere zadostno število volilnih uradnikov, ki pragovno odšifrirajo skupni števec.

5.4 Varnost sheme

Napadi glasovnice. Varnost glasovnice lahko ogrozijo volilni uradniki, tako da ustvarijo glasovnice, ki izgledajo veljavne, vendar razvrstitev kandidatov ne ustreza razvrstitvi čistopisov v črtni kodi, ali pa ustvarijo glasovnice, ki so popolnoma napačne in bi povzročile neveljaven glas. Obe vrsti zaznava naključno preverjanje. Dodatno pa popolnoma napačne glasovnice odkrijemo s pomočjo dokazov brez razkritja znanja. Ker pa dokazi zagotavljajo zgolj, da vsi

tajnopisi v črtni kodi šifrirajo vsako možno izbiro, z njimi ne moremo ugotoviti, da razvrstitev kandidatov na glasovnici ne ustreza razvrstitvi tajnopisov.

Zunanji napadalci bi lahko poskušali vnesti večje število napačnih glasovnic z namenom oviranja volitev. Te poskuse zaznajo uradniki in ostale organizacije, saj vnešene napačne glasovnice nimajo veljavnih dokazov in zato niso na seznamu potrjenih glasovnic.

Koalicija volivcev in uradnikov, bi lahko poskušala oddati glasovnico, ki npr. poveča števec nekega kandidata za veliko vrednost. Enako kot prej, take poskuse preprečijo dokazi brez razkritja znanja.

Tajnost glasovnic. Volilni uradniki bi lahko povzročili odtok informacij z uporabo ne dovolj varnih naključnih vrednosti pri šifriranju. Tukaj je predvsem pomembno, kako se določa seme naključnosti. Avtorja predlagata, da se najprej izbere semena za naključnost posameznih glasovnic, nato pa še neko javno izbrano seme, ki se vključi v že izbrana semena. Vendar avtorja opozorita, da bi bilo potrebno to tematiko bolje raziskati.

Ena od skrbi je tudi odstranljiva površina, ki mora biti izdelana tako, da prepreči, da bi nekdo lahko brez zaznave prebral naključne vrednosti. To bi omogočilo glasovanje pod prisilo oz. prodajo glasov.

Težava je tudi zajem vsebine glasovnic, preden te pridejo do volivca. Volilni uradniki, kurirji ipd. bi lahko posneli črtno kodo in razvrstitev kandidatov. Kasneje, ko sta izbor in črtna koda skupaj z identiteto volivca objavljena na oglasni deski, bi napadalci preprosto poiskali pripadajočo črtno kodo in ugotovili, kateri kandidat ustreza mestu izbora. Možna rešitev je, da bi se glasovnice tiskale sproti na volitvah, kar pa nekoliko oteži izvedbo. Ostale rešitve pa gredo v smeri fizičnega varovanja kot npr. zapečatene glasovnice, prekrita črtna koda ipd.

Pri oddaji glasovnice lahko ponovno pride do razkritja naključnih vrednosti. Volilni uradnik bi težko sam neopazno odstranil površino, večja težava pa je sodelovanje med volivcem in uradnikom. Če bi postopek opazovala samo onadva, bi uradnik volivcu preprosto predal naključne vrednosti. Temu se izognemo z zadostnim številom neodvisnih opazovalcev postopka oddaje.

Posebna težava pa je lahko prisiljeno naključno glasovanje. Določene volilne enote izkazujejo večje število glasov za nekega kandidata. Napadalec bi lahko volivce prisilil, da označijo npr. prvo mesto na glasovnici. To lahko sicer ustreza izbiri volivca, vendar pa napadalec dobi zagotovitev, da so vsi volivci pod njegovo prisilo izbrali kandidata na prvem mestu. Ker so razvrstitve naključne, se rezultati pomaknejo proti enakomerni porazdelitvi glasov, s tem pa se prednost statistično preferiranega kandidata zmanjša. Možna rešitev je, da

volivcem omogočimo dovolj velik izbor glasovnic, tako da prvo mesto ustreza njihovi siceršnji izbiri.

Napadi na oglasno desko in štetje. Oglasna deska je očitna tarča napadalca, ki bi želel spremeniti oddane glasovnice, jih izbrisati, vstaviti nove ipd. Rešitev je uporaba digitalnih potrdil s strani volilnih uradnikov. Da preprečimo napade samega strežnika, na katerem je oglasna deska, izvedemo oglasno desko na večih strežnikih s posebnimi protokoli, ki skrbijo za konsistentno delovanje vseh strežnikov.

Ob varni izvedbi oglasne deske, napadi na postopek štetja niso možni, saj ima vsaka glasovnica objavljen dokaz o pravilni obliki, vsak lahko preveri izvedbo homomorfnega združevanja glasov, pravilnost odšifriranja končnega skupnega števca pa je tudi dokazana.

Poglavje 6

Helios

6.1 Uvod

Helios [18, 2] je odprtokodni sistem za internetne volitve, ki ga je leta 2008 razvil Ben Adida. Glavni namen Heliosa je, da javnosti približa kriptografsko varne e-volitve. Vsakdo lahko na spletni strani sistema ustvari volitve, vnese volivce in jih povabi k oddaji glasovnic. Za volitve z večjim številom volivcev pa lahko postavimo Helios na lastnem strežniku in ga tudi dopolnimo, npr. vpeljemo digitalna potrdila za avtentikacijo. Helios ni celovita in kompleksna rešitev, ampak predvsem prosto dostopen *splošno preverljiv* sistem, ki volivcem omogoča *zagotovitev oddaje glasovnice*. Zaradi navedenega in pa dejstva, da gre za internetni sistem, Helios ni namenjen volitvam z visoko verjetnostjo napadov, ampak volitvam v organe raznih klubov, študentskih svetov ipd. Helios je bil med drugim uporabljen za volitve organov IACR, mednarodnega združenja za kriptološko raziskovanje. Marca 2009 pa je belgijska Universite catholique de Louvain s pomočjo Heliosa implementirala e-volitve za izbor rektorja [3].

V različici v1 je Helios uporabljal shemo na osnovi mešalne mreže Sako-Kilian, v različicah v2, v3 in v4 pa so avtorji prešli na uporabo sheme s homomorfnim štetjem glasov. So pa pri različici v4 ponovno napovedali možnost mešalnih mrež.

V nadaljevanju poglavja bomo predstavili ključne kriptografske gradnike v Heliosu. Poudarek bo na opisu kriptografske sheme za e-volitve, pregledali pa bomo tudi varnost sistema. Če ne bo drugače navedeno, bomo opisovali sistem iz zadnje različice.

6.2 Kriptografski gradniki

V tem razdelku predstavimo kriptografske gradnike, ki jih uporablja Helios in podrobneje opišemo tiste, ki niso bili opisani že v predhodnih poglavjih. Gradniki bodo v celoto smiselno umeščeni v naslednjem razdelku, kjer bomo opisali sistem.

Kot smo že omenili, je Helios v prvi različici uporabljal shemo, ki je temeljila na mešalni mreži Sako-Kilian. Le-to smo opisali v 4. poglavju. V vseh kasnejših različicah pa Helios temelji na kriptografski shemi s homomorfnim štetjem glasov. Ta uporablja sledeče kriptografske gradnike:

- eksponentni ElGamalov kriptosistem (glej 2.3.2), saj izkazuje aditivni homomorfizem in je sorazmerno preprost glede na ostale aditivno homomorfne PKC;
- odšifriranje porazdeljeno med več zaupnikov;
- Chaum-Pedersenov dokaz o enakosti diskretnih logaritmov (glej 3.1) za dokazovanje pravilne oblike glasovnice in pravilnega odšifriranja;
- Schnorrov dokaz o poznavanju diskretnega logaritma nekega elementa (glej 3.1) za dokazovanje poznavanja zasebnega ključa, ki pripada deležu javnega ključa;
- disjunktivni dokaz o enakosti diskretnih logaritmov, ki uporabi Chaum-Pedersenov dokaz kot sestavni del dokaza o pravilni obliki glasovnice.

Naj bo ℓ število volilnih uradnikov (zaupnikov), pri čemer i-tega označimo z U_i. Naj bo N število volivcev (enako številu glasovnic), pri čemer j-tega označimo z V_j. Naj bo $p \in \mathbb{P}$ in naj ima element $g \in \mathbb{Z}_p^*$ praštevilski red q. Torej, $\langle g \rangle$ je podgrupa grupe $\mathbb{Z}_p^*$ in $|\langle g \rangle| = q$. Javni parametri volitev so trojica (p, q, g).

Za računanje zgostitev se trenutno uporablja SHA-256, lahko pa bi uporabili katerokoli drugo kriptografsko zgoščevalno funkcijo.

Eksponentni ElGamal. Čistopis m šifriramo z uporabo javnega ključa y in naključne vrednosti $r \in_R \mathbb{Z}_q$ v tajnopis $(\alpha, \beta) = (g^r, g^m \cdot y^r)$. Pri Heliosu so čistopisi (vrednosti m) zgolj 0 ali 1. Vprašanje z več možnimi odgovori izvedemo z uporabo večjega števila tajnopisov.

Porazdeljeno odšifriranje. Helios od različice v2 naprej podpira porazdelitev odšifriranja končnega rezultata med več zaupnikov. V podrazdelku 2.4.2

smo opisali učinkovito in varno shemo za izvedbo pragovnega ElGamalovega kriptosistema. Avtorji so razmišljali o implementacije te možnosti, vendar so se odločili za način, ki ne uporablja Shamirjevega deljenja skrivnosti in ne omogoča pragovnega odšifriranja v pravem pomenu, je pa zato preprostejši za implementacijo in razumevanje.

Uradnik U_i izbere delež zasebnega ključa $x_i \in_R \mathbb{Z}_q$ in izračuna $y_i = g^{x_i} \bmod p$, kar je pripadajoči delež javnega ključa y. Nato izračuna in objavi zapis Schnorrovega dokaza o poznavanju diskretnega logaritma $\log_g y_i \bmod p$, torej o poznavanju pripadajočega zasebnega ključa x_i. Javni ključ je potem

$$y = y_1 \cdot y_2 \cdot \ldots \cdot y_\ell \bmod p.$$

Pri odšifriranju tajnopisa (α, β) vsak U_i izvede svoj delež odšifriranja, tako da izračuna t.i. odšifrirni količnik $k_i = \alpha^{x_i} \bmod p$. Nato sistem izračuna

$$M = \frac{\beta}{k_1 \cdot \ldots \cdot k_\ell} \bmod p.$$

Čistopis dobimo z računanjem diskretnega logaritma $m = \log_g M \bmod p$, pri čemer uporabimo učinkovit algoritem, npr. metodo veliki korak - mali korak.

Dokazi pravilnega porazdeljenega odšifriranja. Pri vseh opisanih dokazih gre za neinteraktivne različice. S pomočjo Chaum-Pedersenovega dokaza uradniki dokažejo pravilnost delnega odšifriranja. Pokažejo namreč, da je $\log_g y_i \equiv \log_\alpha k_i \pmod p$.

Dokazi pravilne oblike glasovnice. Helios omogoča večje število vprašanj in več možnih odgovorov pri posameznem vprašanju. Ker sistem stremi k čim bolj preprostim rešitvam, za ta namen ne uporablja kompleksnejših števcev, kot je npr. tisti iz sheme Scratch & Vote. Raje se za vsako vprašanje uporabi večje število tajnopisov, konkretno en tajnopis za vsak možen odgovor. Če smo izbrali nek odgovor, šifriramo vrednost 1, sicer šifriramo 0. Pri štetju glasov se tajnopisi, ki pripadajo posameznemu odgovoru, pomnožijo med seboj in tako dobimo tajnopis, ki predstavlja števec glasov za dani odgovor. Helios omogoča tudi nastavitev minimalnega in maksimalnega števila izbranih odgovorov pri posameznem vprašanju. Zaradi navedenega je pomembno, da so glasovnice pravilno oblikovane. Sicer bi nekdo lahko za svojega kandidata namesto 1 šifriral vrednost 1000 in mu tako namenil 1000 glasov. Ali pa bi imeli glasovnice, kjer bi bilo število izbranih odgovorov izven nastavljenih meja.

Zato se dokazuje pravilnost množice tajnopisov, ki pripadajo posameznemu vprašanju. Potrebno je dokazati dve stvari. Najprej dokažemo, da vsak posamezen tajnopis, ki ustreza posameznemu odgovoru, res šifrira zgolj vrednost

0 ali 1. S tem preprečimo, da bi nekdo npr. oddal 1000 glasov za določen odgovor oz. oddal negativne glasove za nek drug odgovor. Potrebno pa je tudi dokazati, da je v skupini tajnopisov, ki pripada danemu vprašanju, število izbranih odgovorov znotraj nastavljenih meja. Konkretno, dokazati je treba, da zmnožek tajnopisov posameznih odgovorov rezultira v tajnopisu, ki šifrira vrednost med minimalnim in maksimalnim število možnih odgovorov.

Za oba namena uporabimo **disjunktivni dokaz o enakosti diskretnih logaritmov** [11], s katerim dokažemo, da eksponentni ElGamalov tajnopis šifrira natanko eno izmed vrednosti $\{\mathsf{min}, \mathsf{min}+1, \ldots, \mathsf{max}\}$. V prvem primeru izvajamo dokaz na tajnopisih, ki pripadajo posameznim odgovorom, pri čemer je $\mathsf{min} = 0$ in $\mathsf{max} = 1$. V drugem primeru pa izvajamo dokaz na tajnopisu, ki je zmnožek tajnopisov posameznih odgovorov, vrednosti min in max pa sta odvisni od posameznega vprašanja.

Dokaz za osnovo uporablja Chaum-Pedersenov dokaz, je neinteraktiven in uporablja zgoščevalno funkcijo H, trenutno je to SHA-256. Ime izhaja iz dejstva, da moramo za dani tajnopis pokazati, da velja enakost $\log_g \alpha \equiv \log_y(\beta/g^m) \pmod p$ za natanko eno vrednost $m \in \{\mathsf{min}, \ldots, \mathsf{max}\}$. Enakost torej velja za tisto vrednost m, ki smo jo dejansko šifrirali. Ideja dokaza je, da za ostale vrednosti m simuliramo Chaum-Pedersenov dokaz relacije $\log_g \alpha \equiv \log_y \frac{\beta}{g^m} \pmod p$, za pravo vrednost m pa izračunamo pravi dokaz. Naj bo $(\alpha, \beta) = (g^r, g^m \cdot y^r)$ tajnopis, za katerega dokazujemo, da je $m \in \{\mathsf{min}, \ldots, \mathsf{max}\}$. Dokaz in preverjanje prikazuje protokol 6.1.

6.3 Opis sistema

V nadaljevanju podamo splošen opis sistema, nato pa podrobneje opišemo podatkovne tipe, ki jih uporablja Helios. Slednji nam pomagajo razumeti zgradbo sistema. Razdelek zaključimo z opisom izvedbe volitev, kjer zaokrožimo predstavitev sistema.

6.3.1 Splošen opis in programske komponente

Kriptografska shema v prvi različici Heliosa je temeljila na mešalni mreži Sako-Kilian (glej 4.3) in **Benalohovem protokolu** [6]. Ključna lastnost slednjega protokola, ki je pomembna za Helios, je ločitev priprave glasovnice od oddaje glasovnice. Od različice v2 dalje se namesto mešalne mreže Sako-Kilian uporablja **homomorfno štetje glasov na osnovi eksponentnega ElGamalovega kriptosistema**. Se pa v različici v4 spet uvaja podpora mešalnim

Protokol 6.1 Disjunktivni Chaum-Pedersenov dokaz o enakosti dveh diskretnih logaritmov

1: Primož za vsako vrednost $i \in \{\mathsf{min}, \ldots, m-1, m+1, \ldots, \mathsf{max}\}$ simulira Chaum-Pedersenov dokaz. To naredi tako, da si izbere izziv $c_i \in_R \mathbb{Z}_q$ ter odgovor $s_i \in_R \mathbb{Z}_q$ ter nato za ti vrednosti izračuna primerni zavezi, in sicer $a_i = g_i^s/\alpha^{c_i} \bmod p$ ter $b_i = y^{s_i}(\beta/g^i)^{-c_i} \bmod p$.

2: Primož za vrednost m izvede pravi Chaum-Pedersenov dokaz. Naključno izbere zavezo $w \in_R \mathbb{Z}_q$ ter izračuna zavezi $a_m = g^w \bmod p$ ter $b_m = y^w \bmod p$. Nato izračuna še pravi izziv kot

$$c_m = H(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}) - \sum_{i \in \{\mathsf{min}, \ldots, m-1, m+1, \ldots, \mathsf{max}\}} (c_i \bmod q)$$

ter pripadajoči odgovor $s_m = w + r \cdot c_m \bmod q$.

3: Vera za tajnopis (α, β) in zapis dokaza

$$(a_{\mathsf{min}}, b_{\mathsf{min}}, c_{\mathsf{min}}, s_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}, c_{\mathsf{max}}, s_{\mathsf{max}})$$

preveri pravilnost na sledeč način. Najprej se prepriča, da je $g_i^s \equiv a_i \cdot \alpha^{c_i} \pmod p$ in $y^{s_i} \equiv b_i \cdot (\beta/g^i)^{-c_i} \pmod p$ za $\mathsf{min} \leq i \leq \mathsf{max}$. Nato še preveri, če je

$$H(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}) = \sum_{\mathsf{min} \leq i \leq \mathsf{max}} c_i \bmod q.$$

Če vse pravkar zapisane enakosti veljajo, Vera dokaz sprejme, sicer ga zavrne.

mrežam, vendar je zaenkrat zgolj na ravni abstraktnih podatkovnih modelov, nobena mreža še ni implementirana. Tukaj bomo opisali Helios, kakršen je v zadnji različici v4. Z vidika ključnih elementov kriptografske sheme je nespremenjen že od različice v2. Izboljšave od različice v2 dalje so šle v smeri modularnosti Heliosa, splošnosti podatkovnih struktur zaradi možnosti uporabe drugih kriptosistemov, boljšega uporabniškega vmesnika ipd.

Helios je razvit v programskih jezikih Python (na strežniški strani), JavaScript (koda v brskalniku volivca) in Java. Slednja se uporablja za kriptografske operacije na strani volivca, ker je izvajanje potrebnih matematičnih operacij nad velikimi števili bistveno hitrejše v Javi kot pa v JavaScriptu. Komunikacijo med JavaScriptom in Javo omogoča vmesnik LiveConnect, ki ga podpira večina sodobnih brskalnikov. Koda za delo s podatkovnimi strukturami v brskalniku volivca uporablja JavaScript knjižnico jQuery, same podatkovne strukture pa so zapisane v formatu JSON (angl. JavaScript Object Notation). Od različice v2 dalje Helios teče na spletnem ogrodju Django, za podatkovno bazo pa uporablja PostgreSQL.

6.3.2 Podatkovni tipi in parametri

Vse podatkovne strukture imajo od različice v4 dalje tudi element namenjen opisu tipa, ki ga bomo pri posameznih opisih izpuščali. To omogoča razširljivost, saj tako lažje uvedemo nov kriptosistem, način mešanja ipd. Za kodiranje števil se uporablja format base64.

Naštejmo nekaj glavnih podatkovnih struktur:

- `Javni ključ` je sestavljen iz elementov g, p, q in y, katerih pomen je jasen že iz razdelka o kriptografskih gradnikih.

- `Tajnopis` predstavlja splošen tajnopis (ne tajnopis glasovnice) in v primeru ElGamalovega kriptosistema vsebuje elementa `alpha` in `beta`, ki sta komponenti tajnopisa.

- `Vprašanje` predstavlja posamezno vprašanje na volitvah. Struktura vsebuje seznam možnih odgovorov, minimalno in maksimalno možno število izbranih odgovorov in še nekaj podrobnosti.

- `Volitve` so struktura, ki zapisuje informacije o volitvah kot so: naziv, unikatna številka, ustvarjalec, opis, način štetja glasov, zgostitev seznama volivcev, seznam vprašanj in javni ključ. Zaradi integritete obstaja še posebna podatkovna struktura, *prstni odtis volitev*. Le-ta vsebuje zgostitev vseh podatkov za dane volitve.

- `ZKP` predstavlja zapis neinteraktivnega dokaza brez razkritja znanja. Trenutno Helios uporablja zgolj eksponentni ElGamalov kriptosistem, zato se na tem mestu uporablja Chaum-Pedersenov dokaz, ki se uporablja tako za dokazovanje pravilnega odšifriranja, kot tudi za dokaz pravilne oblike glasovnice – glej 6.2. Zapis tega dokaza sestoji iz zaveze, izziva in odgovora, pri čemer je zaveza par števil (a, b) (glej 3.1.3).

- `Disjunktivni ZKP` predstavlja zapis dokaza, s katerim dokažemo, da nek tajnopis šifrira natanko eno od vrednosti med min in max. Tu Helios uporablja disjunktivni Chaum-Pedersenov dokaz, ki smo ga opisali na koncu razdelka 6.2. Zapis tega dokaza je seznam zapisov posameznih ZKP. Slednjih je $\mathsf{max} - \mathsf{min} + 1$, pri čemer so vsi razen enega simulirani, kot smo povedali že pri opisu dokaza.

- `Odgovor` predstavlja odgovor na posamezno vprašanje volitev. Naj bo n število možnih odgovorov na vprašanje. Odgovor potem sestoji iz n tajnopisov in $n+1$ disjunktivnih ZKP. Vsak tajnopis šifrira vrednost 1, če je volivec označil pripadajoči odgovor in 0, če ga ni. Vsakemu tajnopisu pripada po en disjunktivni ZKP, ki dokazuje, da tajnopis res šifrira eno od vrednosti 0 ali 1. Dodatni disjunktivni ZKP pa dokazuje, da zmnožku vseh tajnopisov, ki šifrira vsoto pripadajočih čistopisov, pripada čistopis z vrednostjo med min in max.

- `Glasovnica` je struktura, v kateri so zbrani pravkar opisani odgovori. Z volitvami je povezana preko prstnega odtisa in preko unikatne številke volitev.

- `Glasovnica in odgovori s čistopisi` je struktura, ki služi preverjanju pravilnega šifriranja. Če se volivec odloči, da bo namesto oddaje glasovnico raje pregledal, mu sistem vrne glasovnico, ki vsebuje prirejene odgovore na vprašanja. In sicer, posamezen odgovor ima poleg že omenjenih podatkov še n pripadajočih čistopisov ter n naključnih vrednosti. Z njimi volivec lahko preveri, da vsakemu čistopisu, šifriranemu z zapisano naključno vrednostjo, res ustreza pripadajoči tajnopis. Take glasovnice ni več možno oddati.

- `Volivec` je struktura, ki vsebuje osnovne podatke o volivcu – ime, naslov e-pošte in unikatno identifikacijsko številko. S to številko je lahko povezan tudi volivčev psevdonim. Vsi volivci so zbrani na seznamu volivcev, zgostitev tega seznama pa je eden od parametrov volitev, kot smo že omenili.

- `Oddana glasovnica` je struktura, ki predstavlja že oddano glasovnico. Vsebuje čas oddaje, zgoraj opisano glasovnico in pripadajočo zgostitev, unikatno številko volivca ter zgostitev volivca.

- `Rezultat` je podatkovna struktura, ki hrani števce volitev v obliki seznama seznamov. Vsakemu vprašanju pripada seznam števcev glasov za posamezen odgovor, vsak tak seznam pa je združen v seznam vseh vprašanj na volitvah.

- `Zaupnik` je struktura, ki hrani podatke povezane z volilnimi uradniki oz. zaupniki. Prvi in privzeti zaupnik je vedno sistem Helios sam, lahko pa dodamo nove zaupnike. Podatki v strukturi so: unikatna številka zaupnika, delež javnega ključa zaupnika (g, p, q, y_i), zgostitev deleža javnega ključa, zapis dokaza o poznavanju zasebnega ključa x_i, seznam seznamov odšifrirnih količnikov ter seznam seznamov dokazov o pravilnem odšifriranju. Slednja seznama imata toliko elementov, kolikor je vprašanj na volitvah, vsakemu vprašanju pa pripada seznam števcev za posamezen odgovor. Dokazi o pravilnem odšifriranju so zapisi Chaum-Pedersenovega dokaza, da je $\log_g y_i \equiv \log_\alpha k_i \pmod p$ (glej 6.2), kar nas prepriča, da je zaupnik U_i pri izračunu odšifrirnih količnikov uporabil svoj zasebni ključ x_i. Dokaz o poznavanju zasebnega ključa pa je zapis Schnorrovega dokaza o poznavanju diskretnega logaritma nekega elementa (glej 3.1.2). Ta zapis ne uporablja podatkovne strukture ZKP, ampak ima svojo strukturo prav za ta namen.

6.3.3 Izvedba volitev

Predstavitev sistema zaokrožimo v tem podrazdelku z opisom vseh korakov pri izvedbi volitev. Parametri so enaki kot v 6.2, razen če zapišemo drugače.

Priprava volitev. Upravljalec volitev je oseba, ki na Helios strežniku ustvari volitve, tako da vnese ime, opis, zaupnike, volivce, vprašanja, čas začetka in trajanja volitev ipd. Vsak vnešen volivec po e-pošti prejme podatke za prijavo v sistem. Upravljalec vnese tudi zaupnike, razen če želi, da je Helios edini zaupnik. Pri tem upravljalec vnese zgolj osnovne podatke o zaupniku – ime, e-pošta. Helios potem zaupniku pošlje e-pošto s podatki za prijavo v sistem. Vsak zaupnik U_i nato na svojem računalniku naključno generira svoj zasebni ključ x_i in pripadajoči javni ključ $y_i = g^{x_i} \bmod p$. Pri tem se uporablja JavaScript in Java koda, ki se na zaupnikov računalnik naloži s Helios strežnika. Zaupnik izračuna tudi Schnorrov dokaz o poznavanju diskretnega logaritma

$\log_g y_i \bmod p$, s čimer se prepreči kreiranje neveljavnega javnega ključa. Zaupnik zasebni ključ obdrži zase, javni ključ pa skupaj z dokazom pošlje upravljalcu. Ta nato na **oglasni deski** objavi javne ključe zaupnikov skupaj z dokazi, seznam volivcev, seznam vprašanj z odgovori, čas začetka volitev ipd. Skupni javni ključ volitev y je kar zmnožek javnih ključev zaupnikov – glej 6.2. Izračuna in objavi se tudi prstni odtis skupnega javnega ključa. Upravljalec objavi še prstni odtis volitev.

Priprava glasovnice. Volivec s Heliosa na računalnik prenese parametre volitev, na katerih želi glasovati. Na tej točki se še ne prijavlja v sistem. Poleg volivca v resnici lahko omenjeni prenos izvede kdorkoli. Volivec lahko preveri, da je prstni odtis prejetega javnega ključa y enak objavljenemu prstnemu odtisu. Poleg parametrov se na volivčev računalnik prenese tudi skripta za pripravo in oddajo glasovnice. Ta se do oddaje glasovnice ne poveže v internet. Volivec označi izbire pri posameznih vprašanjih in nato s pomočjo skripte pripravi šifrirano glasovnico (glej podatkovno strukturo `Glasovnica` in `Odgovor`, ki smo ju opisali v podrazdelku 6.3.2). Skripta pri vsakem vprašanju za vsak možen odgovor šifrira 1, če je bil odgovor izbran, in 0, če ni bil. Nato izračuna še disjunktivne dokaze o pravilnem šifriranju, kot smo jih opisali v 6.2. Na tej točki je šifrirana glasovnica shranjena v pomnilniku računalnika v obliki, kot smo jo opisali v 6.3.1. Volivcu skripta prikaže tudi prstni odtis pripravljene glasovnice.

Preverjanje glasovnice. Po pripravi glasovnice ima volivec dve možnosti. Lahko se odloči za preverjanje glasovnice ali pa za oddajo. Če se odloči za preverjanje, mu skripta razkrije pripravljeno glasovnico skupaj s čistopisi in naključnimi vrednostmi, ki so bile uporabljene za šifriranje tajnopisov (podatkovna struktura je opisana v 6.3.1). Tako lahko volivec preveri, da je sistem res pravilno šifriral izbrane odgovore. Na ta način Helios doseže *zagotovitev oddaje* glasovnice. Zaradi *odpornosti na prisilo* sistem volivcu ne dovoli oddati razkrite glasovnice.

Oddaja glasovnice. Če se je volivec po pripravi glasovnice odločil za oddajo, skripta od volivca zahteva vnos prijavnih podatkov (uporabniško ime, geslo) in se prvič po začetku priprave glasovnice poveže v internet. Volivec pri tem shrani prstni odtis svoje glasovnice za kasnejše preverjanje. Skripta Heliosu pošlje prijavne podatke volivca in pripravljeno glasovnico. Helios preveri prijavne podatke volivca in dokaze o pravilni obliki glasovnice. Če je vse v redu, Helios na oglasni deski volitev objavi glasovnico skupaj z imenom (ali psevdo-

nimom) volivca in prstnim odtisom oddane glasovnice.

Splošno preverjanje. Ko preteče možnost oddaje glasovnic, Helios določen čas dovoli preverjanje oddanih glasovnic, preden se izvede štetje in odšifriranje. Na tej točki volivci lahko preverijo, da so njihove glasovnice prispele na oglasno desko nespremenjeno, saj je na oglasni deski tabela z imeni volivcev (ali njihovimi psevdonimi) in prstnimi odtisi oddane glasovnice. Volivec pa lahko tudi vidi celotno oddano glasovnico. Prav tako lahko kdorkoli (splošna preverljivost) preveri, da so vse oddane glasovnice pravilne oblike.

Štetje in odšifriranje. Ko preteče čas za preverjanje oddanih glasovnic, Helios začne štetje in odšifriranje glasovnic. V prvi fazi Helios sam izvede množenje tajnopisov glasovnic in na oglasni deski objavi tajnopise, ki predstavljajo števce glasov za določen odgovor pri določenem vprašanju. Ta korak lahko preveri kdorkoli preprosto tako, da ponovi množenje tajnopisov in primerja dobljene rezultate.

V drugi fazi Helios pozove zaupnike, naj vsak od njih izvede svoj del odšifriranja. Zaupniki izračunajo odšifrirne količnike, po enega za vsak števec (tajnopis (α, β)) skupaj z dokazi pravilnega odšifriranja – glej 6.2. Vsak zaupnik na oglasni deski objavi odšifrirne količnike in pripadajoče dokaze, ki jih lahko kdorkoli preveri. Helios za vsak števec (α, β) izračuna vrednost

$$M = \frac{\beta}{k_1 \cdot \ldots \cdot k_\ell} \bmod p.$$

Iz M izračunamo čistopis $m = \log_g M \bmod p$ z uporabo učinkovitega algoritma za izračun diskretnih logaritmov. Primer takega algoritma je metoda veliki korak – mali korak, s katero na sodobnih računalnikih brez težav računamo logaritme reda 32 bitov. To pomeni, da števci pri Heliosu lahko hranijo vrednosti do $2^32 - 1$ volivcev, kar je dovolj za večino volitev.

Na koncu sistem objavi rezultate. Kdorkoli lahko ponovi zadnja dva izračuna in tako preveri pravilnost.

Računska zahtevnost. Računsko najbolj zahteven del izvedbe volitev je postopek štetja glasov (zaradi računanja diskretnih logaritmov) in preverjanja dokazov o pravilni obliki glasovnice, ki jih volivci oddajo skupaj s šifriranimi odgovori. Avtorji iz [3] so navedli naslednje rezultate.

Štetje so implementirali v programskem jeziku C z uporabo knjižnice GMP [17]. Na tipičnem prenosniku jim je uspelo obdelati 1300 glasovnic na minuto. Glasovnica je vsebovala eno vprašanje s tremi možnimi odgovori.

Za izračun diskretnih logaritmov so implementirali metodo veliki korak – mali korak. Na tipičnem prenosniku so izračunali diskretni logaritem reda 42 bitov v manj kot 10 sekundah. Pri tem so uporabili tabelo vnaprej izračunanih vrednosti velikosti 12.5 MB.

6.4 Varnost

Ključni element varnosti pri Heliosu so dokazi pravilnosti ter možnost preverjanja glasovnice namesto oddaje.

Nepravilno odšifriranje s strani uradnikov preprečimo z dokazi o pravilnem odšifriranju. Vstavljanje neprimernih glasovnic v sistem s strani volivcev preprečimo z dokazi o pravilni obliki glasovnice.

Največ možnosti za napad je na strani odjemalca, in sicer kompromitiran računalnik volivca. Ta lahko šifrira povsem drugačno izbiro glasov in tudi zakrije to dejanje, če se volivec odloči za pregled šifrirane glasovnice. Uspešno je bil prikazan napad, kjer napadalec prevzame nadzor nad volivčevim brskalnikom, tako da izkoristi varnostno luknjo pri določenih različicah programa Adobe Reader [19]. Manjši protiukrep temu je uvedba možnosti oddaje pregledane glasovnice na oglasno desko, kjer potem lahko ostali nekompromitirani računalniki zaznajo napako. Sicer pa so avtorji eksplicitno navedli, da Helios ne bo odporen na take vrste napadov.

Možen je tudi napad s kopiranjem glasovnice [19]. V tem primeru napadalec podvoji objavljeno glasovnico in jo odda kot svojo. Če to naredi dovolj velika skupina napadalcev, se vzorec izbranih odgovorov napadene glasovnice pozna na rezultatih. Iz teh napadalci nato poskušajo razbrati izbrane odgovore pri napadeni glasovnici.

Ena od težav bi bil tudi kompromitiran Helios strežnik [2]. Ta bi lahko oddajal glasovnice za volivce, ki niso volili, saj pozna vsa uporabniška imena in gesla. To bi preprečili s prenosom avtentikacije na bolj varen strežnik. Prav tako bi Helios strežnik lahko ob oddaji spremenil glasovnico in vnesel drugačen tajnopis. To lahko preprečimo, če le dovolj volivcev izvede preverjanje zgostitve glasovnice, ki je objavljena na oglasni deski.

Poglavje 7

Zaključek

Ogledali smo si kriptografske gradnike, ki nam omogočajo implementacijo varnih e-volitev. Opisali smo dva različna pristopa, ki se ločita po načinu štetja glasov in si ogledali dve konkretni shemi, od katerih je ena že implementirana.

Posebno poglavje smo namenili mešalnim mrežam. Prednost slednjih je, da so čistopisi lahko poljubne oblike, kar pomeni, da volivcem omogočajo vpisovanje v glasovnico. To je zlasti zanimivo za anonimne ankete ali pa tiste volitve, kjer volivci sami vpišejo določene vrednosti. Kljub temu, pa je velikost glasovnice omejena z velikostjo prostora čistopisov uporabljenega PKC. To težavo odpravljajo hibridne mešalne mreže, kjer za šifriranje glasovnice uporabimo simetrične kriptosisteme. Leta 2007 je bila objavljena prva splošno preverljiva hibridna mešalna mreža [16]. Zaključimo lahko, da je področje mešalnih mrež še vedno v razvoju, čeprav je bilo že veliko storjenega od Chaumove mreže iz leta 1981. Pričakujemo lahko, da se bo razvoj na tem področju usmeril v čim bolj preproste, a hkrati učinkovite mreže, ki bodo omogočale izvedbo glasovnic s čim manj omejitvami.

Ugotovimo lahko, da nam obstoječa kriptografska orodja že omogočajo izvedbo e-volitev. Vseeno pa bo najbrž minilo še nekaj časa, preden se bodo e-volitve razširile in se uporabljale tudi pri volitvah z višjo stopnjo tveganja, kot so referendumi, volitve predsednikov, poslancev ipd. Težave niso zgolj tehnične narave, ampak tudi, kako odgovorne in volivce prepričati, da bodo zaupali e-volitvam. Primer resnejše uporabe e-volitev bodo lokalne volitve na Norveškem [20] septembra 2011, kjer bodo dovolili glasovanje preko interneta z možnostjo kasnejšega preglasovanja na volišču. V načrtu so zapisali, da je cilj izvedba parlamentarnih e-volitev leta 2017.

Slike

Seznam protokolov

Literatura

[1] B. Adida, *Advances in Cryptographic Voting Systems*, doktorska disertacija, MIT, 2006.

[2] B. Adida, *Helios: Web-based Open-Audit Voting*, Proceedings of the 17th Usenix Security Symposium, 2008.

[3] B. Adida, O. de Marneffe, O. Pereira, J. Quisquater, *Electing a university president using open-audit voting: analysis of real-world use of Helios*, Proceedings of the 18th Usenix Security Symposium, 2009.

[4] B. Adida, R.L. Rivest, *Scratch & Vote – Self-contained Paper-based Cryptographic Voting*, ACM Workshop on Privacy in the Electronic Society, 2006.

[5] O. Baudron, P.A. Fouque, D. Pointcheval, G. Poupard, J. Stern, *Practical Multi-Candidate Election System*, Proceedings of the 20th ACM Symposium on Principles of Distributed Computing, ACM Press, 2001.

[6] J. Benaloh, *Simple verifiable elections*, Proceedings of the USENIX/ACCURATE Electronic Voting Technology Workshop 2006, 2006.

[7] J. Benaloh (Cohen), M.J. Fischer, *A Robust and Verifiable Cryptographically Secure Election Scheme*, Proceedings of 26th Symposium on Foundations of Computer Science, 1985.

[8] Carmichaelova funkcija, `http://mathworld.wolfram.com/CarmichaelFunction.html`, prebrano 12.08.2011.

[9] D. Chaum, *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*, Communications of the ACM, Volume 24, 1981.

[10] D. Chaum, T. Pedersen, *Wallet Databases with Observers*, Crypto '92, Volume 740 of Lecture Notes in Computer Science, Springer, 1993.

[11] V. Cortier, B. Smyth, *Attacking and fixing Helios: An analysis of ballot secrecy*, Proceedings of the 24th IEEE Computer Security Foundations Symposium (CSF'11).

[12] I. Damgard, M. Jurik, *A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System*, Public Key Cryptography 2001, Springer, 2001.

[13] T. ElGamal, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, Advances in Cryptology — Crypto 84, Volume 196 of Lecture Notes in Computer Science, Springer, 1985.

[14] A. Fiat, A. Shamir, *How to prove yourself: practical solutions to identification and signature problems*, Advances in Cryptology — Crypto 1986, Volume 263 of Lecture Notes in Computer Science, Springer, 1986.

[15] J. Furukawa, K. Sako, *An Efficient Scheme for Proving a Shuffle*, Advances in Cryptology — Crypto 2001, Volume 2139 of Lecture Notes in Computer Science, Springer, 2001.

[16] J. Furukawa, K. Sako, *An Efficient Publicly Verifiable Mix-Net for Long Inputs*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Volume E90-A Issue 1, 2007.

[17] Spletna stran knjižnice GMP, `http://gmplib.org/`, prebrano 29.08.2011.

[18] Spletna stran sistema Helios, `http://heliosvoting.org`, prebrano 11.08.2011.

[19] Spletna stran sistema Helios – podstran o napadih, `http://documentation.heliosvoting.org/attacks-and-defenses`, prebrano 28.08.2011.

[20] Internetne volitve na Norveškem 2011, `http://evalg.dep.no`, prebrano 28.08.2011.

[21] M. Jakobsson, A. Juels, R.L. Rivest, *Making mix nets robust for electronic voting by randomized partial checking*, USENIX Security Symposium, 2002.

[22] C.A. Neff, *A verifiable secret shuffle and its application to e-voting*, ACM 8th Conference on Computer and Communications Security, ACM, 2001.

[23] P. Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, Advances in Cryptology - Eurocrypt 99, 1999.

[24] C. Park, K. Itoh, K. Kurosawa, *Efficient Anonymous Channel and All/Nothing Election Scheme*, Advances in Cryptology - Eurocrypt '93, Volume 765 of Lecture Notes in Computer Science, Springer, 1994.

[25] T.P. Pedersen, *A Threshold Cryptosystem without a Trusted Party*, Eurocrypt, Volume 547 of Lecture Notes in Computer Science, Springer, 1991.

[26] B. Pfitzmann, A. Pfitzmann, *How To Break The Direct RSA Implementation Of Mixes*, Advances in Cryptology - Eurocrypt '89 Proceedings, 1990.

[27] R. L. Rivest, A. Shamir, L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, Volume 21 Issue 2, 1978.

[28] C.P. Schnorr, *Efficient identification and signatures for smart cards*, Advances in Cryptology - Crypto '89, Volume 435 of Lecture Notes in Computer Science, Springer, 1990.

[29] K. Sako, J. Kilian, *Receipt-free Mix-type Voting Scheme - a practical solution to the implementation of a voting booth*, Eurocrypt, Volume 921 of Lecture Notes in Computer Science, Springer, 1995.

[30] A. Shamir, *How to Share a Secret*, Communications of the ACM, Volume 22, 1979.