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

Mitja Trampuš

MISELNI POKER

Diplomska naloga
na univerzitetnem interdisciplinarnem študiju
matematike in računalništva

Mentor: prof. dr. Aleksandar Jurišić

Ljubljana, 2008

Original izdane teme

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

Zahvala

Hvala mentorju za nasvete in popravke pri izdelavi celotne diplomske naloge ter Arjani Žitnik za komentarje in predloge v četrtem poglavju.

Hvala družini, še posebej staršem, za podporo v celotnem času šolanja.
Hvala tudi prijateljem, s katerimi je bilo prijetno iti skozi ta leta.

Kazalo

Povzetek **1**

Summary **3**

1 Uvod **5**

2 Problem miselnega pokra **8**
2.1 Pravila pokra 9
2.2 Igra kot zaporedje operacij 10
2.3 Zahtevane formalne lastnosti 10

3 Kriptografska orodja **14**
3.1 Problem faktorizacije 14
3.1.1 Pollardova metoda $p-1$ 15
3.1.2 Pollardova metoda ρ 16
3.1.3 Dixonova metoda, kvadratno sito in sito v številskih poljih . 19
3.1.4 Pomen za miselni poker 20
3.2 Problem diskretnega logaritma 20
3.2.1 Shanksova metoda 21
3.2.2 Pollardova metoda ρ za diskretni logaritem 23
3.2.3 Index calculus 25
3.2.4 Pomen za miselni poker 26
3.3 Problem kvadratnih ostankov in korenjenje v $\mathbb{Z}_p^*$ 27
3.3.1 Povezava med korenjenjem in faktorizacijo 30
3.3.2 Pomen za miselni poker 31
3.4 Zapriseženi biti 31
3.4.1 Implementacija z zgoščevalno funkcijo 32
3.4.2 Goldwasser-Micalijev protokol 33
3.4.3 Pomen za miselni poker 35
3.5 Dokazi brez razkritja znanja (ZKP) 35

3.5.1 ZKP za Hamiltonov cikel 36
3.5.2 Chaum-Pedersenov ZKP 37
3.5.3 Pomen za miselni poker 38

4 Primeri shem za miselni poker **40**
4.1 Sheme s TTP in Fortune-Merritova shema 41
4.1.1 Osnovne operacije 42
4.1.2 Varnostna analiza 44
4.1.3 Časovna zahtevnost 46
4.2 Shamir-Rivest-Adlemanova shema 47
4.2.1 Časovna in varnostna analiza 49
4.3 Castellàjeva ZKP-shema 49
4.3.1 Implementacija operacij 50
4.3.2 Varnostna analiza 55
4.3.3 Časovna zahtevnost 59

5 Miselni poker v vsakdanu **62**

6 Zaključek **66**

A Primer poročila o reviziji spletne igralnice **67**

Seznam algoritmov in protokolov **70**
Literatura **71**

Izjava **74**

Seznam uporabljenih kratic in simbolov

CDH - Computational Diffie-Hellmann, slovensko računski Diffie-Hellmann

DDH - Decisional Diffie-Hellmann, slovensko odločitveni Diffie-Hellmann

DLP - Discrete Logarithm Problem, slovensko problem diskretnega logaritma

RSA - Šifrirna shema, ki so jo razvili Rivest, Shamir in Adleman.

TTP - Trusted Third Party, slovensko center zaupanja.

$A \leq_P B$ - problem A je v polinomskem času Turingovo prevedljiv na problem B

ZKP - Zero-Knowledge Proof, slovensko dokaz brez razkritja znanja

Povzetek

Ogledamo si problematiko miselnega pokra, to je pokra brez fizičnih kart, ki ga najpogosteje igramo preko interneta. Posebna veja kriptografije se ukvarja z vprašanjem, kako in s kakšnim protokolom pri miselnem pokru zagotoviti poštenost igranja, četudi nekateri igralci želijo goljufati. Še posebej zanimivo to vprašanje postane, če prepovemo uporabo centralnega, zaupanja vrednega strežnika, ki bi vodil igro (delil karte in podobno), ne bi pa se je udeležil kot igralec. Strežnik, ki nadzoruje celotno igro, je namreč očitna tarča za napade.

Z izvedbo sheme za igranje miselnega pokra brez takšnega strežnika se ukvarja večji del te diplomske naloge. Najprej natančneje predstavimo ozadje problema, formaliziramo varnostne zahteve za ustrezno rešitev in opišemo že znana matematična in kriptografska orodja, ki tipično služijo kot gradniki pri konstrukciji takšnih protokolov. Nato predstavimo tri rešitve, ki podane varnostne zahteve izpolnjujejo v različni meri. Najzanimivejša od teh rešitev jim zadosti v celoti in je hkrati tudi časovno razmeroma učinkovita.

V zaključku si ogledamo položaj spletnega pokra v današnji družbi ter premislimo o potencialu uporabe naprednih kriptografskih rešitev, kot je pravkar opisana, v vsakdanjih partijah spletnega pokra.

Ključne besede:

miselni poker, mentalni poker, poker, kriptografija, varnost, dokaz brez razkritja znanja, internet, igralništvo, kartanje, igre s kartami

Summary

We consider mental poker, i.e. a game of poker played without the aid of physical cards, most commonly on the internet. A special branch of cryptography deals with ensuring fairness and security in such online games even if not all the players are well-intentioned. The problem becomes particularly interesting if we wish to dispose of a centralized trusted server which would conduct the game (deal the cards and similar) without itself participating as an active player. Such a server, if present in the game, has complete control over its proceedings and therefore represents a natural and vulnerable target for various attacks.

A greater part of this thesis deals with steps involved in constructing a cryptographic scheme for playing poker without a central server of this kind. We first formally identify the background of the problem and its security requirements. We then proceed to describe those mathematical and cryptographic tools that typically play an important role in constructing mental poker protocols. Building on that, we present three solutions which satisfy the identified security constraints to varying degrees. The most interesting of these solutions satisfies all the constraints while maintaining reasonable performance.

Lastly, we comment on the role of online poker in today's society and consider the potential of using advanced decentralized solutions like the ones described in everyday games of online poker.

Keywords:

mental poker, poker, cryptography, security, zero-knowledge proof, internet, gambling, card games

Poglavje 1

Uvod

Mnoge družabne igre je težko ali celo nemogoče igrati, če nimamo ustreznih fizičnih pripomočkov. *Človek, ne jezi se* brez kocke, *Črni Peter* brez kart, *Scrabble* brez ploščic s črkami, *kamen, škarje, papir* brez gest z rokami, *met kovanca* brez kovanca – takšne različice iger se zdijo povsem nesmiselne.

Pa vendar danes ljudje počnemo ravno to: kartamo brez kart in kockamo brez kock. Naštete igre namreč igramo tudi preko interneta, in tam ni kock, kart in kovancev, pa tudi soigralcev nimamo pred očmi. Brez teh pripomočkov smo vsaj navidez primorani nenehno verjeti soigralcem na besedo. Ko na primer prek spleta igramo poker in neznanec na "mizo" vrže pikovo damo, imamo le njegovo zagotovilo, da jo je nekoč res potegnil z vrha navideznega kupčka kart. S tem slepo zaupamo v njegovo poštenost, čeprav je bolj realno pričakovati, da bo vsaj občasno uporabil tudi kakšno karto, ki je v resnici ni nikoli potegnil, bi pa bila zanj zelo ugodna. Še posebej hitro lahko do takega goljufanja pride, kadar je v igri denar.

Težavo v praksi dandanes omilimo tako, da tovrstne igre igramo s pomočjo namenskega, neodvisnega strežnika, na katerega nihče od igralcev nima vpliva. Strežnik je nato tisti, ki meče kocko in kovanec ali meša in deli karte. Ne le to, strežnik lahko nadzoruje igralce, da ne goljufajo: prav dobro na primer ve, ali je imel naš soigralec pikovo damo res v rokah, preden jo je uporabil. Mnogo lažje je zaupati v nepristranskost strežnika kot pa neposredno soigralcem, saj strežniku ni v očitnem interesu, da bi goljufal v dobro kateregakoli od igralcev.

Met kovanca, nadvse enostavna igra, še jasneje prikaže prednosti uporabe strežnika. Zamislite si, da s prijateljem prek interneta stavim zaboj piva, da ne ugane, na katero stran (grb ali cifra) bo ob metu padel kovanec. Denimo, da prijatelj stavi na grb. Če mu prepustim, da vrže kovanec, ga bo stava morda premamila, da bo goljufal in sporočil, da je pri metu kovanca padel grb – ne glede na to, kaj je v resnici vrgel. Če bom kovanec vrgel jaz, bo pivo morda tudi meni odplaknilo moralne

zadržke in bom prijatelju sporočil, da je stavil narobe – spet ne glede na to, ali je tudi v resnici padla cifra. Naravno je torej, da drug drugemu ne zaupava preveč. Če pa met kovanca prepustiva neodvisni tretji osebi oziroma strežniku, sva lahko precej prepričana, da bo pošteno vrgel kovanec in nama obema sporočil, ali je moj prijatelj stavil pravilno ali narobe. Še toliko lažje bo v njegovo poštenost verjeti, če mu v zameno za njegovo posredovanje obljubiva steklenico piva iz zaboja.

Prav na principu te nagradne steklenice piva v resnici temelji velika večina današnjega spletnega igralništva, le da igralci stavijo denar namesto piva in igrajo poker ali ruleto, namesto da bi metali kovanec. Igralne hiše igralcem po svetu ponudijo strežnike, ki posredujejo pri igrah za denar, in za te strežnike jamčijo nepristranskost in poštenost. V zameno hiše od igralcev poberejo simbolično prijavnino. Ta denar pripomore, da igralne hiše udobno preživijo in jim ni treba sprejemati podkupnin.

Vendar pa tudi igranje z uporabo strežnika še vedno temelji na zaupanju: igralni hiši, katere strežnik prevzame vodenje igre, moramo slepo zaupati, da svoje delo opravlja pošteno. Čeprav v praksi sistem precej dobro deluje, se ne moremo znebiti zoprnega občutka, da bodo ljudje za dovolj velik denar to zaupanje ob prvi priložnosti izigrali, in to se občasno tudi res zgodi. Goljufiv igralec mora z dovolj visoko podkupnino le dobiti dostop do domnevno neodvisnega strežnika – in že ima pregled nad celotno igro, kar mu omogoča, da stavi v natanko pravih trenutkih ter si na račun poštenih igralcev priigra lepe vsote. Strežnik-razsodnik s pregledom nad igro bo zato vedno tarča napadov; dokler v igri uporabljamo tak strežnik, ne moremo doseči popolne varnosti. Po drugi strani je videti, da prav njegova uporaba močno poveča naše zaupanje v poštenost igre. Se uporabi strežnika lahko izognemo, pa vendarle obdržimo garancijo poštenosti?

Odgovor je pritrdilen, vprašanju "kako to storiti?" pa pravimo *problem miselnega pokra* in je tema te diplomske naloge. Za raznovrstne igre, predvsem pa za poker, problem miselnega pokra zahteva matematične in šifrirne postopke, ki zagotavljajo pošteno igro preko spleta *brez uporabe neodvisnega strežnika*, kajti ta je nujno ranljiv. Na prvi pogled je cilj nedosegljiv: spomnimo se le našega primera s kovancem. Če ni tretje osebe, igralca pa drug drugemu ne zaupava in sva za povrh povezana le preko interneta, le kdo naj vrže kovanec, da bo poštenost zagotovljena? Rešitev presenetljivo ni preveč zapletena in je opisana v razdelku 3.4. Met kovanca resda ni ravno razburljiva igra, a rešitev za pošteno igranje brez neodvisnega strežnika obstaja tudi za poker, najpopularnejšo igro spletnih igralnic. Le ustrezno kompleksnejša je: uporablja šifrirne postopke in večkratno izmenjevanje sporočil med igralci po strogo predpisanem zaporedju. Pravimo, da je ta rešitev skupek *kriptografskih protokolov* oziroma *kriptografska shema*. V nadaljevanju se posvetimo različnim kriptografskim shemam, ki rešujejo problem miselnega pokra.

Obravnava teh shem nas bo pripeljala do načina za varno igranje pokra preko spleta, hkrati pa predstavlja tudi dobro vajo iz kriptografske študije protokolov.

Za konec na kratko predstavimo strukturo preostanka diplomske naloge. Po poljudnem uvodu si ogledamo problem miselnega pokra z bolj tehničnega stališča. V drugem poglavju najprej formalno opišemo problem ter zahteve, ki jih mora izpolnjevati njegova rešitev.

V tretjem poglavju sledi pregled obstoječih kriptografskih orodij, ki so tipično uporabljena pri konstrukciji shem za igranje miselnega pokra. Poglavje začnemo z osnovnimi, pretežno matematičnimi problemi, ter nadaljujemo z višjenivojskimi protokoli, ki temeljijo na osnovnih problemih in so neposredneje uporabni kot gradniki shem.

V četrtem poglavju predstavimo tri konkretne sheme, ki so bile predlagane kot rešitev problema miselnega pokra. Prva iz performančnih razlogov uporablja centralni strežnik, vendar hkrati poskusi omejiti njegova pooblastila in možnosti zlorabe. Drugo shemo predstavimo le na kratko. Gre za prvo predlagano shemo za miselni poker sploh, zato je polna pomanjkljivosti, ki pa niso vse očitne in služijo kot nazoren zgled in opozorilo. Tretja shema je ena najboljših danes znanih; zadosti vsem formalnim varnostnim pogojem in je tudi razumno hitra, čeprav je prostora za izboljšave še precej.

V zadnjem, petem poglavju si ogledamo, kakšno vlogo ima spletni poker v današnji družbi ter kakšno vlogo in prihodnost ima miselni poker brez nadzornega strežnika v današnjem svetu spletnega igralništva.

Poglavje 2

Problem miselnega pokra

V uvodu smo neformalno nakazali, kaj je miselni poker. V tem poglavju se teme lotimo sistematičneje.

Pojem 2.0.1. *Miselni poker* (angl. mental poker) je poker, ki ga igramo preko računalniškega omrežja ali druge komunikacijske povezave, vendar brez uporabe fizičnih kart.

Pri tem omenimo, da splošno uveljavljena definicija problema miselnega pokra ne obstaja. Zgornja sicer zajame večino člankov, ki trdijo, da se nanašajo na miselni poker, vendar nekateri avtorji pojem dojemajo širše. Beseda *poker*, trdijo, je zgodovinski relikt; po njihovem miselni poker kot področje zaobjema (skoraj) vse igre s kartami.

To razdvojeno definicijo nehote dobro ponazori Castellà Roca, eden najaktivnejših avtorjev na tem področju. V svoji doktorski nalogi [4] definira miselni poker kot omrežno igranje "iger, kjer je vrednost ali množica vrednosti pridobljena naskrivaj", a se v vseh svojih publikacijah o miselnem pokru kljub temu posveča izključno igri pokra.

Razlikovanje vendarle ni tako pomembno, kot se morda zdi, saj lahko rešitve, razvite za poker, največkrat z minimalnimi spremembami ali celo neposredno uporabimo tudi za "miselne" različice mnogih drugih iger s kartami.

Pojem 2.0.2. *Problem miselnega pokra* lahko formuliramo kot vprašanje: "Kako igrati miselni poker in pri tem zagotoviti poštenost igre, po možnosti brez uporabe neodvisnega posrednika?". Kot odgovor oziroma rešitev pričakujemo algoritmičen opis ustrezne kriptografske sheme.

Preden se lotimo reševanja problema miselnega pokra, ga moramo formalneje opredeliti. S tem namenom v nadaljevanju na kratko opišemo igro pokra in jo raz-

drobimo na posamezne operacije, ki jih mora rešitev podpirati, nato pa naštejemo še splošne varnostne zahteve takšne rešitve.

2.1 Pravila pokra

V namene miselnega pokra moramo poker kot igro opisati le toliko natančno, da postanejo jasno razvidne operacije, ki gradijo igro in jih mora zato podpirati vsaka shema za miselni poker. Marsikakšen del pravil bomo izpustili, poudarili pa tiste, ki so pomembne z varnostnega stališča.

Znanih je mnogo različic pokra; tu bomo opisali široko razširjeni *Five-card draw*. Druge, na primer *Texas hold 'em*, so s stališča zahtevnosti implementacije miselnega različice enakovredne ali celo preprostejše.

Čeprav so pravila stavljenja v igri pomembna, jih bomo tu v celoti izpustili. Vse stave so javne in zato neproblematične za implementacijo prek omrežja.[1]

Predpriprava. Igro igramo z 52 različnimi kartami, ki so s hrbtne strani neoznačene in medsebojno enake. Karte dobro premešamo, da nihče ne ve ničesar o vrstnem redu kart v premešanem kupčku.

Prvi krog. Vsak od igralcev dobi 5 kart, ki jih pogleda, ne da bi jih pokazal soigralcem. Nato oceni svoje karte in stavi nanje ali pa odstopi od igre.

Drugi krog. Vsak od igralcev zavrže največ 3 svoje karte, lahko tudi nobene. To naredi tako, da soigralci ne vidijo, katere karte je zavrgel, vidijo pa njihovo število. Zavržene karte se do konca igre ne uporabijo več. Kolikor kart je igralec zavrgel, toliko novih dobi s kupčka premešanih kart. Nato igralci zopet stavijo na svoje karte ali odstopijo od igre.

Zaključek. Igralci, ki v prvih dveh krogih niso odstopili od igre, razkrijejo svoje karte. Pravila določajo (za nas nepomembno) zaporedje, po katerem se postopoma odkrivajo posamezne karte. Možno je, da nekateri igralci razkrijejo le del svojih kart ali celo sploh nobene.

Po igri. Karte, ki v opisanem postopku niso bile razkrite, morajo ostati zakrite tudi po koncu igre, sicer izdajo preveč informacij o slogu igre posameznih igralcev. Ne da bi kdorkoli razen lastnika videl njihovo vrednost, jih premešamo skupaj s preostalimi kartami in tako pripravimo novo igro.

[1] Drugo vprašanje je, kako igralce, ki izgubijo, prisiliti, da bodo zastavljeni denar res izplačali – oziroma kdo naj hrani zastavljeni denar v primeru, da ga igralci vplačujejo sproti. Zadrego večina člankov o miselnem pokru prikladno zamolči. Najpreprostejša rešitev je uporaba tretje osebe, nekoga, ki za majhno provizijo pošteno hrani in razdeljuje denar.

2.2 Igra kot zaporedje operacij

Za lažjo obravnavo igro pokra predstavimo kot zaporedje preprostejših operacij. Kot smo omenili že uvodoma, bomo rešitev pozneje podali v obliki kriptografske sheme, to je nabora protokolov, ki bodo implementirali pošteno izvajanje teh operacij. Te so:

Inicializacija. Prvi korak v vsaki igri; z njim vzpostavimo pogoje za igranje. Pri miselnem pokru se tu na primer inicializirajo kriptografski protokoli, igralci se dogovorijo za številsko predstavitev kart in podobno. Pri pokru s fizičnimi kartami se inicializacija zgodi nezavedno; del inicializacije je na primer dogovor igralcev o tem, kateri paket kart bodo uporabljali.

Mešanje kart. Vhodni podatek za to operacijo so vse karte; njihov vrstni red je lahko javno znan. Izhodni podatek so iste karte, razporejene v nekem novem (ne nujno drugačnem) vrstnem redu, pri čemer o tem vrstnem redu nihče od igralcev ne sme imeti nobene informacije.

Vlečenje karte. Za izvedbo te operacije mora igralec dobiti dovoljenje od vseh soigralcev. Kot rezultat dobi ta igralec v uporabo prvo še nepovlečeno karto. Pri tem zve vrednost povlečene karte, soigralci pa o njej ne smejo zvedeti ničesar.

Razkrivanje karte. Vhodni podatek je zakrita karta K. Operacija razkrije vrednost K vsem igralcev. Operacijo sme izvesti le lastnik K.

Odmetavanje karte. Vhodni podatek je zakrita karta K. Ko se operacija izvede, se K v igri ne da več uporabiti. Operacijo sme izvesti le lastnik K. Tudi po operaciji mora K ostati zakrita.

Glede na implementacijo in uporabljena kriptografska orodja v rešitvi lahko definiramo tudi drugačno razbitje igre na operacije. V nadaljevanju se bomo držali zgoraj opisanega razbitja.

2.3 Zahtevane formalne lastnosti

Vsaka implementacija miselnega pokra mora očitno spoštovati pravila igre, ki smo jih navedli. Vendar pa to ni dovolj za pošteno igro preko računalniškega omrežja. Potrebujemo še nekatere dodatne lastnosti, ki so pri pokru s fizičnimi kartami večinoma zagotovljene implicitno, pri miselnem pokru pa moramo nanje posebej paziti.

Te lastnosti je urejeno prvič opisal Crépeau [7] leta 1985. Preden si jih ogledamo, pa se moramo seznaniti še s pojmom centra zaupanja.

Pojem 2.3.1. *Center zaupanja* je pooblaščena entiteta (strežnik), ki ji zaupamo skrb za varnost enega ali več delov kriptografske sheme. Tipično ima več informacij o igri in lahko izvaja več operacij kot drugi uporabniki sheme. Center zaupanja mora biti nepristranski; v njegovo poštenost morajo zaupati vsi uporabniki sheme.

Center zaupanja bomo označevali s kratico *TTP* (iz angl. Trusted Third Party). V drugih vejah kriptografije je za soroden pojem pogosto v uporabi izraz Trusted Authority oziroma kratica TA.

Crépeaujeve lastnosti, ki jih zahtevamo za shemo miselnega pokra, so:

a) **Odsotnost centra zaupanja.** Shema ne sme uporabljati centra zaupanja. Tudi nobenemu igralcu ne sme biti dodeljena privilegirana vloga v smislu količine dosegljivih informacij. To ne pomeni nujno, da so vloge vseh igralcev povsem enakovredne – v nekaterih shemah na primer prvi igralec med operacijo mešanja kart opravi več računanja, vendar je njegov položaj v smislu količine dosegljivih informacij ob koncu operacije enakovreden položaju soigralcev.

b) **Unikatnost kart.** Vsaka karta se mora pojaviti natanko enkrat, bodisi v kupčku neuporabljenih kart, bodisi med zavrženimi kartami, bodisi v rokah enega od igralcev. Če se neka karta pojavi dvakrat, sme to biti le posledica izsledljivega goljufanja.

c) **Enakomerno naključna porazdelitev kart.** Za vsakega igralca P in fiksno število kart k mora biti za vsako možno množico k kart enako verjetno, da jih P dobi v roko. V našem primeru, kjer eksplicitno definiramo mešanje kot ločeno operacijo, je ekvivalentno zahtevati, naj bo v operaciji mešanja vsaka možna permutacija enako verjetna.

d) **Odkrivanje goljufij.** Verjetnost, da poljubna goljufija ostane neopažena, mora biti matematično majhna. To pomeni, da mora padati vsaj eksponentno v odvisnosti od nekega varnostnega parametra, ki ga igralci sporazumno določijo pred začetkom igre. Hkrati mora časovna zahtevnost sheme glede na ta varnostni parameter naraščati počasi, največ polinomsko. Velikost varnostnega parametra določimo na podlagi medsebojnega zaupanja med igralci ter njihovih računskih kapacitet.

e) **Tajnost kart.** Noben igralec ne sme imeti dostopa do delne ali popolne informacije o vrednosti katerekoli zakrite karte, razen če to odobrijo vsi soigralci.

Prav tako noben igralec ne sme imeti dostopa do vrednosti katerekoli karte, ki jo v roki drži kak drug igralec, razen če to odobri lastnik karte.

f) **Tajnost strategije.** Igralci morajo biti zmožni dokazati svojo poštenost, ne da bi ob koncu igre razkrili svoje parametre kriptosistema (na primer zasebni ključ). S tem bi namreč nujno razkrili tudi, katere karte so na koncu imeli v roki, s tem pa izničili "blef", pomembno komponento pokra.

g) **Minimalen vpliv zarot.** Če se nekaj igralcev zaroti proti ostalim in si med seboj namerno izmenjajo informacije o kartah in kriptosistemu, smejo o kartah ostalih igralcev zvedeti le toliko, kot če bi si pokazali le karte: namreč natančnejšo verjetnostno porazdelitev kart soigralcev. Ne smejo pa biti sposobni zlomiti ali načeti varnosti kriptosistema.

Izkaže se, da je formalizacija dobra, zato se je do danes uveljavila kot merilo za doslednost shem za miselni poker. Izhajajoč iz praktične rabe pa lahko dodamo še eno danes močno zaželeno lastnost, ki leta 1985, ko je bil brez široko razpredenega interneta miselni poker zanimiv več ali manj v teoriji, ni bila bistvena:

h) **Učinkovitost.** Vse operacije naj bodo računsko hitro izvedljive. V praksi to pomeni, da naj povprečen osebni računalnik vse operacije izvede v času, ki za uporabnika ni moteč – na primer v največ nekaj sekundah.

Združitev vseh teh zahtev se izkaže za trd oreh, celo če za trenutek odmislimo željo po učinkovitosti. Shemo, ki je zadoščala točkam od (a) do (g), je sicer že leta 1986 odkril in opisal kar Crépeau sam, vendar je bilo pozneje objavljenih še mnogo člankov s predlogi shem, ki zavestno niso ustrezale vsaj eni od točk (a), (f) in (g).

Večino tovrstnih kompromisov lahko uvrstimo v eno od dveh skupin. V prvo spadajo praktično usmerjeni predlogi, ki se osredotočijo najprej na točko (h), učinkovitost. S tem namenom prekršijo zahtevo (a) in vpeljejo TTP, saj njegova uporaba močno poenostavi shemo ter s tem zmanjša računsko zahtevnost. Takšen predlog je bil objavljen še leta 2003 [19]. Danes, nekaj let pozneje, že imamo rešitve, ki ne uporabljajo TTP in so hkrati dovolj učinkovite, da so pogojno praktično uporabne. Ker pa nobena od njih še ni bila implementirana, sheme s TTP predstavljajo trenutno edino v praksi uporabljano rešitev in se jim bomo posebej posvetili v razdelku 4.1.

V drugo skupino shem, ki nimajo vseh naštetih lastnosti, spadajo sheme brez TTP. Te največkrat ne zagotavljajo tajnosti strategije igralcev. V shemah brez neodvisnega razsodnika je igralcem namreč težko dokazovati svojo poštenost. Najlaže poštenost zagotovimo tako, da na koncu igre vsi igralci razkrijejo svoje privatne ključe ali druge parametre kriptosistema. S tem lahko vsi igralci pri sebi ponovno

preračunajo celoten potek igre in se prepričajo o njeni korektnosti. Vendar pa s tem lahko tudi izračunajo, katere karte so soigralci zavrgli in katere so zakrite obdržali v rokah, zato takšna shema odpove na točki (f), nerazkritje strategije. Tiste sheme, ki jim uspe zagotoviti tudi lastnost (f), pa so (z izjemo dveh ali treh najnovejših shem) bistveno prepočasne za praktično uporabo.

Poglavje 3

Kriptografska orodja

V tem poglavju se bomo najprej posvetili nekaterim osnovnim matematično-računskim problemom, na zahtevnosti katerih temelji velik del sodobne kriptografije. Nato si bomo ogledali tudi izbrane višjenivojske konstrukte in protokole, ki gradijo na teh osnovnih problemi in se posebej pogosto pojavljajo v kontekstu miselnega pokra.

Pri vseh področjih bomo na kratko opisali tudi njihovo uporabnost pri snovanju shem za miselni poker.

3.1 Problem faktorizacije

Problem faktorizacije [17, razdelek 5.6] je naslednji domnevno težko rešljiv problem: za dano sestavljeno naravno število $n = pq$ poišči faktorja p in q.

Naivni pristop k problemu je, da preprosto preizkusimo vse kandidate za p in q. Ni težko videti, da mora vsaj eno od teh števil biti manjše ali enako $\sqrt{n}$, zato bo iskanje deliteljev vzelo $O(\sqrt{n})$ korakov. To je sprejemljivo za razmeroma majhne n, denimo $n \leq 10^{20}$. V namene kriptografije pa uporabljamo mnogo večja števila, denimo $n \doteq 2^{1024} \doteq 10^{308}$. Naivni pristop bi torej zahteval mnogo preveč časa. Ne le naivni pristop, tudi najnaprednejše znane metode niso zmožne faktorizirati tako velikih n. Na domnevi, da hiter algoritem za faktorizacijo sploh ne obstaja, sloni marsikateri protokol, med drugim dobro znani in množično uporabljani RSA. V nadaljevanju si bomo ogledali nekaj najhitrejših znanih pristopov in se prepričali, da je n velikosti reda 2^{1024} ali celo 2^{512} zanje nedosegljiv v dostojnem času.

Opomnimo, da je pri tem inverzna operacija – izračun števila n iz podanih p in q – učinkovito izvedljiva tudi pri velikih p in q. Časovna zahtevnost takšnega množenja je $O(\log^2 n)$.

V vseh algoritmih bomo privzeli, da je n lih, sicer je faktorizacija trivialna.

3.1.1 Pollardova metoda $p - 1$

Naivni pristop z neposrednim deljenjem ni nujno počasen: tudi pri velikem n bi lahko hitro uspel, če bi n imel kak majhen prafaktor. Seveda se to ne zgodi, saj so protokoli zasnovani tako, da takšnega n prav zaradi tovrstnega napada ne izberejo.

Pollardova metoda $p-1$ se prav tako osredotoči na faktorizacijo števil s posebno lastnostjo, a pristopi bolj zvito: hitro zna faktorizirati take n, ki imajo vsaj en prafaktor p, za katerega $p-1$ delijo samo majhne praštevilske potence. Predpostavimo, da tak p obstaja in da so vse potence praštevil, ki delijo $p-1$, manjše ali enake neki konstanti B. Le-ta nastopa kot parameter v algoritmu 3.1.1: više kot postavimo B, bolj verjetno bo predpostavka držala in bo algoritem uspel, vendar bo potreboval tudi več časa.

Algoritem 3.1.1 POLLARDP1(n, B)

Vhod: Sestavljeno število n, ki ga želimo faktorizirati; naravno število B.
Rezultat: Netrivialen delitelj števila n ali neuspeh.

1) $a := 2^{B!} \bmod n$
2) $d := \gcd(a - 1, n)$
3) **if** $1 < d < n$ **then**
4) **return** d
5) **else**
6) **return** "Pri podanem B ni ustreznega faktorja p"
7) **end if**

Trditev 3.1.1. *Naj ima število n vsaj en prafaktor p, za katerega za vsako praštevilo q velja $\forall k \in \mathbb{N} : q^k \mid p - 1 \Rightarrow q^k \leq B$. Tedaj je število d :=*POLLARDP1(n, B) *netrivialen delitelj števila n, razen če se algoritem ne zaključi uspešno.*

Dokaz. Zaradi naše predpostavke o velikostih praštevilskih potenc, ki delijo $p - 1$, mora biti

$$p - 1 \mid B!.$$

Ker v prvem koraku algoritma postavimo $a \equiv 2^{B!} \pmod{n}$ in velja $p \mid n$, mora biti tudi

$$a \equiv 2^{B!} \pmod{p}.$$

Po Fermatovem izreku velja $2^{p-1} \equiv 1 \pmod{p}$; ker pa je $B!$ večkratnik števila $p-1$, mora biti tudi $2^{B!} \equiv a \equiv 1 \pmod{p}$. To pomeni, da $p \mid a - 1$, vemo pa že, da $p \mid n$. Torej za $d := \gcd(a-1, n)$ velja tudi $p \mid d$. Število d je že po definiciji delitelj števila n, nedeljivost $p \mid d$ pa nam zagotavlja še, da je $d \geq p$ netrivialen faktor.

Edina izjema je možnost pri $a = 1$, ko za d dobimo trivialno vrednost n. V tem primeru v algoritmu v 6. vrstici sporočimo neuspeh. □

Časovna zahtevnost. Prvi korak algoritma zahteva modularno potenciranje na potenco reda velikosti $O(B^B)$, torej $O(B \log B)$ množenj, če potenciramo učinkovito. Ker množenja potekajo po modulu n, vsako zahteva $O(\log^2 n)$ osnovnih operacij. Drugi korak, izračun največjega skupnega delitelja, z Evklidovim algoritmom zahteva $O(\log^2 n)$ operacij.

Skupaj je časovna zahtevnost torej $O((B \log B + 1) \log^2 n)$. Kot smo pričakovali glede na idejo algoritma, je močno odvisna od parametra B.

Zaščita pred napadom s Pollardovo metodo $p - 1$. Pri snovanju vsake dobre kriptografske sheme, ki temelji na faktorizaciji nekega n, moramo paziti, da vrednosti za n, ki jih bo generiral program, ne bodo ranljiva na napad z zgornjo metodo. Tega na srečo ni pretežko doseči – ker zahtevnost metode hitro narašča z B, moramo le poskrbeti, da bodo vsi prafaktorji števila $p - 1$ za vsak prafaktor p števila n večji od vsakega uporabnega B.

To najpogosteje naredimo takole: najdemo takšni veliki praštevili p' in q', da sta tudi $p := 2p' + 1$ in $q := 2q' + 1$ praštevili. Število $n := pq$ bo tedaj imelo opisano lastnost. Izkaže se, da so kandidati za p' in q' dovolj gosto posejani med naravnimi števili, da jih lahko iščemo z enakimi metodami kot običajna praštevila. Ocenjeno je, da je med 1 in k ustreznih praštevil $O(\frac{n}{\log^2 n})$, ne poznamo pa dokaza za to (empirično) oceno [20].

3.1.2 Pollardova metoda ρ

Ta metoda razmeroma učinkovito (vsaj v primerjavi z naivnim pristopom) faktorizira poljubno sestavljeno število, ne le takšnih s posebnimi lastnostmi.

Naj bo p najmanjši prafaktor števila n. Ker je n sestavljen, gotovo obstajata različni števili $x, x' \in \mathbb{Z}_n$, za kateri velja $x \equiv x' \pmod{p}$. Potem za

$$d := \gcd(x - x', n)$$

velja $p \leq d < n$ in $d \mid n$; povedano drugače, število d je netrivialen delitelj števila n.

Težava je, da ne poznamo vrednosti p. Lahko pa si izberemo kar naključno podmnožico $X \subset \mathbb{Z}_n$ in d izračunamo za vse $x, x' \in X$; za izračun namreč ne rabimo p. Verjetnost, da ne bosta niti dva elementa iz X enaka po modulu p, je po verjetnostnem računu enaka

$$\frac{p!}{(p - |X|)!} / p^{|X|},$$

kajti načinov, na katere lahko izmed p-tih elementov izberemo $|X|$ različnih, je $p(p-1)(p-2)\cdots(p-|X|+1) = \frac{p!}{(p-|X|)!}$; če pa se ne omejimo na izbiranje medsebojno različnih elementov, je možnosti $p^{|X|}$. Z uporabo Stirlingovega približka

lahko ugotovimo, da ta verjetnost pade pod 50% približno pri velikosti podmnožice $|X| \doteq 1.17\sqrt{p}$.

Za solidne možnosti uspeha moramo torej vzeti naključno množico X velikosti $O(\sqrt{p})$. Da v X najdemo dva elementa, ki sta po modulu p enaka, moramo navidez za vsak par izračunati d po formuli iz prejšnjega odstavka in preveriti, ali deli n. Vrednosti p namreč ne poznamo. Vseh parov pa je $O(|X|^2) = O(p) = O(\sqrt{n})$, torej nismo v primerjavi z naivno metodo pridobili zaenkrat ničesar.

Trik Pollardove metode ρ je ravno v tem, da se spretno izogne računanju d za vse pare elementov iz X. Naj bo f celoštevilski polinom ene spremenljivke; pogosto vzamemo $f(x) := x^2 + 1$. Če naključno izberemo še $x_1 \in \mathbb{Z}_n$, lahko generiramo zaporedje $x_1, x_2, x_3, \ldots$, kjer je $x_i := f(x_{i-1}) \bmod n$. Če je f dobro izbran, se vrednost zaporedja giblje statistično gledano psevdonaključno. Zaporedje torej razumno dobro prevzame vlogo množice X iz prejšnjega odstavka: če bomo člene razvijali do približno $(1.17\sqrt{p})$-tega, se bo v njem z verjetnostjo približno $1/2$ pojavil par števil, kongruentnih po modulu p. Pri tem predpostavka o psevdonaključnosti členov zaporedja, generiranega z f, ni matematično osnovana; gre le za empirično ugotovitev, ki pa se v praksi dobro obnese.

Predpostavimo, da v našem zaporedju obstajata indeksa i, j, za katera je $x_i \equiv x_j \pmod{p}$. Ker p deli n, imamo

$$x_{i+1} \equiv f(x_i) \bmod n \equiv f(x_i) \pmod{p}$$

in podobno tudi

$$x_{j+1} \equiv f(x_j) \pmod{p}$$

Zdaj upoštevamo še, da je f polinom in ohranja kongruentnost ter pridemo do naslednjega sklepa: če $x_i \equiv x_j \pmod{p}$, potem tudi $x_{i+1} \equiv x_{j+1} \pmod{p}$. Sklep lahko zdaj induktivno uporabimo poljubno mnogokrat in tako dobimo

$$x_i \equiv x_j \pmod{p} \qquad \Rightarrow \qquad \forall t \in \mathbb{N} > 0 : x_{i+t} \equiv x_{j+t} \pmod{p}$$

Predstavljajmo si zdaj preprost graf, v katerem točke predstavljajo števila iz $\mathbb{Z}_p$, usmerjene povezave pa dodamo na podlagi našega zaporedja – povežemo $x_k \to x_{k+1}$. Z x_i oziroma x_j označimo dva različna elementa zaporedja, ki sta kongruentna po modulu p in zato v grafu predstavljata eno in isto točko. Takšnih parov je, kot smo pravkar pokazali, neskončno. Vzemimo tistega, pri katerem sta i in j minimalna. Potem lahko naš graf, ki ima obliko grške črke ρ, opišemo takole: začne se z "repom"

$$x_1 \to x_2 \to x_3 \to \ldots \to x_i,$$

ki se v točki x_i nadaljuje v cikel

$$x_i \to x_{i+1} \to x_{i+2} \to \ldots \to x_j \equiv x_i$$

Naš cilj je, da vsaj za eno točko na ciklu najdemo dva različna indeksa. To učinkovito storimo, kot opisuje algoritem 3.1.2. Najprej nastavimo $r := x_1$, $s := x_2$; s tem r in s postaneta nekakšna kazalca na prvi točki v grafu. Nato oba kazalca korakoma sprehodimo po grafu vzdolž povezav; prvega premikamo z navadno hitrostjo (eno povezavo v vsakem koraku), drugega pa z dvojno. Ni težko videti, da se bosta v končnem številu korakov kazalca res srečala v neki točki na ciklu.

Algoritem 3.1.2 PollardRho($n, x_1, [f = x \mapsto x^2 + 1]$)

Vhod: Sestavljeno število n, ki ga želimo faktorizirati; število $x_1 \in \mathbb{Z}_n$; celoštevilski polinom f.

Rezultat: Netrivialen delitelj števila n ali neuspeh.

1) $x := x_1$
2) $x' := f(x_1) \bmod n$
3) $k := 1$
4) **repeat**
5) $\quad p := \gcd(x - x', n)$
6) $\quad k++$
7) $\quad x := f(x) \bmod n$
8) $\quad x' := f(f(x')) \bmod n$
9) **until** $p \neq 1$ or $k > 1,17\sqrt[4]{n}$
10) **if** $p = n$ or $p = 1$ **then**
11) $\quad$ **return** "Ne najdem rešitve"
12) **else**
13) $\quad$ **return** p
14) **end if**

V k-ti iteraciji glavne zanke v zgornjem algoritmu imamo $x = x_k$ in $x' = x_{2k}$.

Časovna zahtevnost. V glavni zanki algoritma gradimo množico X oziroma graf G ter se s kazalcema x in x' pomikamo po njem. Pričakujemo, da bosta po največ $O(|X|) = O(\sqrt{p}) = O(\sqrt[4]{n})$ korakih x in x' kazala na isto točko v grafu oziroma bosta kongruentna po modulu p. Če se to zgodi, smo po $O(\sqrt[4]{n})$ korakih končali.

A zanka ne uspe vedno. Množico X namreč gradimo le do velikosti $1,17\sqrt[4]{n}$, kar nam po prejšnjih izračunih zagotavlja le približno 50-odstotno verjetnost, da bomo v glavni zanki generirali ustrezna x in x'. Drugi način, na katerega nam lahko spodleti, je, da najdena x in x' nista kongruentna le po modulu p, temveč tudi po modulu n. Funkcija gcd v tem primeru vrne n, ki je trivialen delitelj. Ta možnost na srečo ni prav verjetna: če je X res naključna podmnožica $\mathbb{Z}_n$, je verjetnost za kaj takega le $|X|/n$, torej zelo majhna. Opozorimo, da analiza ni strogo matematično korektna:

množico X namreč generira funkcija f in v resnici ni prava naključna podmnožica $\mathbb{Z}_n$. Vendar pa se v praksi f obnaša dovolj eratično, da jo za potrebe te analize lahko smatramo za psevdonaključni generator. Tudi empirični rezultati kažejo časovno zahtevnost, povsem podobno tisti, ki jo dobimo v tej časovni analizi.

Ugotovili smo torej, da gre za verjetnostni Las Vegas algoritem, ki uspe s približno 50-odstotno verjetnostjo; če ne uspe, ga enostavno poženemo še enkrat. Ker vsaka iteracija zahteva $O(\sqrt[4]{n})$ korakov, nas takšno zaporedno poganjanje algoritma 3.1.2 pripelje do verjetnostnega algoritma s pričakovano časovno zahtevnostjo $O(\sqrt[4]{n})$.

3.1.3 Dixonova metoda, kvadratno sito in sito v številskih poljih

Poleg opisanih algoritmov je dobro znana še Dixonova metoda naključnih kvadratov, ki uporabi pristop, nekoliko podrobneje opisan v razdelku 3.3.1 o kvadratnih ostankih. Osnovna ideja je, da poiščemo različni števili x in y ($x \not\equiv \pm y \pmod{n}$), za katera je $x^2 \equiv y^2 \pmod{n}$. To preoblikujemo v $(x - y)(x + y) \equiv 0 \pmod{n}$ in sklepamo, da je zato $\gcd(x - y, n)$ delitelj števila n.

Najtežji korak takšnega pristopa je učinkovito generiranje ustreznih x in y. Dixonova metoda na primer hevristično preskusi takšne x in y, katerih kvadrati bodo predvidoma majhna števila. Majhna števila so namreč bolj obvladljiva, kar metoda izkoristi v nadaljnjih korakih [17, razdelek 5.6.3].

Metoda kvadratnega sita ter metoda sita v splošnih številskih poljih temeljita na isti osnovni ideji faktorizacije n s pomočjo razlike kvadratov. Le števila x in y iščeta drugače, s "sejanjem" števil. Sejanje algoritmično poteka s pomočjo velikega polja `boolean` zastavic, ki nam predstavljajo števila-kandidate in ki jih postopoma "sejemo" – nastavljamo na `false`. Na principu sejanja deluje na primer tudi znani Eratostenov algoritem za iskanje praštevil. Pravil sejanja za iskanje kvadratov na tem mestu ne bomo opisovali, saj so bistveno zapletenejša kot na primer pri Eratostenovem situ.

Povejmo le, da je sito v splošnih številskih poljih danes najhitrejši znani algoritem za faktorizacijo števil splošne oblike [17, razdelek 5.6.4]; njegova zahtevnost je

$$O(e^{(1.92+o(1))\ln^{1/3} n \ln^{2/3} \ln n}).$$

Za kratkoročno varnost podatkov v protokolih, ki temeljijo na problemu faktorizacije, moramo zaradi tega algoritma izbirati števila dolžine vsaj 512 bitov, še raje 768 ali 1024, sicer jih je mogoče razmeroma hitro razbiti. Po podatkih spletne strani ameriškega laboratorija RSA, ki je do leta 2007 razpisoval nagrade za faktorizacijo različno velikih števil, je bilo največje število, za katerega so podelili nagrado,

dolgo 663 bitov. Faktorizacija je na 80 osebnih računalnikih trajala tri mesece. Za dolgoročno varnost zato potrebujemo števila dolžine 2048 bitov ali več.

3.1.4 Pomen za miselni poker

Problem faktorizacije je nadvse pomemben ne le na področju miselnega pokra, temveč kriptografije nasploh. Pojavlja se v številnih splošnonamenskih kriptografskih orodjih, ki pridejo prav tudi pri snovanju rešitev za miselni poker. Izpostavimo lahko predvsem shemo RSA, tipično izbiro za vzpostavitev varne komunikacije. Predvsem prav pride pri shemah s TTP, kjer mora vsak od igralcev od TTP na varen način dobiti nekatere informacije, na primer podeljene karte.

Sheme s TTP, ki so danes množično v uporabi, za varen prenos večinoma uporabljajo algoritem RSA, njegova varnost pa temelji med drugim prav na problemu faktorizacije.

Problem faktorizacije je tudi tesno povezan s problemom kvadratnih ostankov, ki v različnih oblikah igra vidno vlogo pri nekaterih shemah brez TTP.

3.2 Problem diskretnega logaritma

Podobno kot problem faktorizacije je tudi *problem diskretnega logaritma (DLP, Discrete Logarithm Problem)* matematično-računski problem, ki služi kot osnova kriptografskim protokolom. Za razliko od prvega, ki ga vedno obravnavamo v množici naravnih števil, pa tu operiramo znotraj multiplikativne grupe G. Problem od nas zahteva, da za dani $\alpha, \beta \in G$ najdemo $a \in G$, za katerega je $\alpha^a = \beta$. Povedano drugače, iščemo logaritem $a := \log_\alpha \beta$; ker pa logaritem operira v multiplikativni grupi, mu pravimo *diskretni*.

Tudi ta problem je podobno kot prejšnji, ki smo ga srečali, domnevno težko rešljiv, čeprav za to ne poznamo dokaza. Dolgotrajno in neuspešno iskanje učinkovitih algoritmov pa nas utrjuje v domnevi, da je diskretni logaritem v splošnem težko izračunati v naslednjih multiplikativnih grupah:

- $\mathbb{Z}_p^*$, kjer je p praštevilo;
- multiplikativna grupa polja $\mathbb{F}_{p^n}$, kjer je p praštevilo;
- grupa točk eliptične krivulje [17, razdelek 6.5], definirane nad končnim poljem.

Označimo velikost grupe z $n := |G|$. Največ koliko časa potrebujemo, da izračunamo diskretni logaritem $\log_\alpha \beta$? Naivni pristop je, da zapovrstjo računamo $\alpha^1, \alpha^2, \alpha^3, \ldots$, dokler ne pridemo do $\alpha^a = \beta$. Ker grupo vedno izberemo tako, da

imajo elementi visok red, bo ta metoda zahtevala $O(n)$ korakov. V nadaljevanju si bomo ogledali nekaj učinkovitejših metod.

Opomnimo, da je DLP podobno kot problem faktorizacije v kriptografiji zanimiv prav zato, ker je težko rešljiv, hkrati pa je njegov inverz enostaven. Vrednost $\beta = \alpha^a$ je mogoče v grupi $\mathbb{Z}_n$ pri danih α in a namreč z metodo "kvadriraj in zmnoži"izračunati v le $O(\log^3 n)$ korakih.

Diffie-Hellmannova problema. V povezavi z diskretnim logaritmom pogosto srečamo še dva problema, prav tako domnevno neobvladljiva. Tudi tu vsi računi potekajo v izbrani multiplikativni grupi. Vpeljala sta ju znana kriptografa Diffie in Hellmann [10, poglavje 3]

Računski Diffie-Hellmannov problem (CDH, Computational D.-H.) zahteva, da pri danih α, α^a in α^b izračunamo α^{ab}.

Odločitveni Diffie-Hellmannov problem (DDH, Decisional D.-H.) zahteva, da pri danih α, α^a, α^b in α^c preverimo, ali velja enakost $\alpha^{ab} = \alpha^c$. (Ekvivalentno, za dane α, β, γ moramo ugotoviti, ali je $\log_\alpha \beta = \log_\alpha \gamma$.)

Ni težko videti, da problema nista težja od problema diskretnega logaritma. Dokažimo to na kratko z opisom polinomskih prevedb DDH $\leq_P$ CDH $\leq_P$ DLP.

Prevedba DDH $\leq_P$ CDH je posebej očitna. Če znamo rešiti CDH, ga lahko uporabimo na prvih treh vhodnih podatkih za DDH, nato pa rezultat le še primerjamo s četrtim vhodnim podatkom.

Tudi CDH $\leq_P$ DLP je preprosto videti. Če znamo rešiti DLP, ga lahko pri reševanju CDH uporabimo na drugem in tretjem vhodnem podatku. Tako dobimo a in b, nato pa le še potenciramo α na ab.

Čeprav sta problema videti nekoliko lažja, ju v splošnem ne znamo rešiti nič bolj učinkovito kot DLP [1]. Sumimo pa, da je to mogoče; v grupi $\mathbb{Z}_p^*$, na primer, najverjetneje obstaja subeksponenten[1] algoritem za DDH [1, razdelek 3.2], ki pa ga ne poznamo.

3.2.1 Shanksova metoda

Metoda je znana tudi pod imenom *veliki korak – mali korak* (giant step – baby step), ki dobro ponazarja idejo metode.

[1] glede na $\log_2 p$

Algoritem 3.2.3 SHANKS(n, α, β)

Vhod: Naravno število n; števili $\alpha \in \mathbb{Z}_n$ in $\beta \in \langle\alpha\rangle$.
Rezultat: Diskretni logaritem $\log_\alpha \beta$ v $\mathbb{Z}_n^*$.

```
1) m := ⌊√n⌋
2) L := prazna razpršena tabela
3) for j := 0, ..., ⌊n/m⌋ do
4)     Vstavi jm v tabelo L pod ključem α^{jm}
5) end for
6) i := 0
7) repeat
8)     i++
9)     β := βα
10) until β je med ključi tabele L
11) return (L[β] − i) mod n
```

"Velike korake" iz imena metode predstavlja prva zanka algoritma 3.2.3. V njej vnaprej izračunamo in shranimo vsako m-to potenco števila α. Tipično izberemo $m = \sqrt{n}$, kot smo naredili tudi v zgornjem zapisu algoritma. Za tako izračunane potence, označili smo jih α^{jm}, seveda poznamo vrednost diskretnega logaritma. Uredimo si jih v razpršeno tabelo L, kar nam pozneje omogoči učinkovito logaritmiranje teh števil.

Sledijo mali koraki: število $\beta = \alpha^a$ (pri čemer je a še neznan) postopoma množimo z α in mu tako višamo potenco, dokler ne postane enako enemu od števil α^{jm}, za katera smo logaritem $L[\alpha^{jm}] = jm$ izračunali v prvem delu algoritma. Ker vemo, koliko takšnih malih korakov smo naredili, od tu ni več težko dobiti vrednosti a.

Časovna zahtevnost. Pri časovni analizi bomo na tem mestu zanemarili logaritemske faktorje. Ker množenje in seštevanje velikih števil zahteva le čas, logaritemski glede na njihovo velikost, to pomeni, da jih bomo obravnavali kot osnovne operacije, izvedljive v $O(1)$ časa.

Prva zanka algoritma vzame $O(n/m)$ časa: najprej izračunamo α^m, sledi n/m zaporednih množenj s tem številom.

Druga zanka zahteva k operacij za nek k. Z velikimi koraki smo izračunali $\alpha^m, \alpha^{2m}, \alpha^{3m}, \ldots$. Če uredimo elemente iz $\langle a\rangle$ glede na vrednost funkcije $\log_\alpha$, leži vhodni podatek $\beta = \alpha^a$ na zaprtem intervalu med nekima $\alpha^{(i-1)m}$ in α^{im}, ki kot ključa že obstajata v L. Ker se z vsako iteracijo zanke pomaknemo na naslednji element glede na pravkar opisano ureditev, bo trajalo največ m iteracij, da prispemo do α^{im}. (V redkih primerih, do katerih pride zaradi zaokroževanja pri definiciji m, lahko

potrebujemo v resnici tudi več iteracij, vendar nikoli več kot $2n/m = 2m$.) Druga zanka torej zahteva $O(m)$ časa. Če nismo prepričani, da je sploh $\beta \in \langle\alpha\rangle$, lahko v algoritmu zanko prekinemo, ko i preseže $2m$, in sporočimo $\beta \notin \langle\alpha\rangle$.

Končna časovna zahtevnost je $O(\frac{n}{m}+m) = O(\sqrt{n})$, če vzamemo za m tipično vrednost $m = \sqrt{n}$. Če vemo, da bomo pogosto računali logaritme z neko fiksno osnovo α, lahko prvo zanko izvedemo celo samo enkrat, ločeno od preostanka funkcije, in tako privarčujemo kar nekaj časa.

3.2.2 Pollardova metoda ρ za diskretni logaritem

Metoda je zelo podobna metodi ρ za faktorizacijo, ki smo jo opisali v razdelku 3.1. Konstruirali bomo zaporedje $x_1, x_2, \ldots$ z uporabo "skoraj psevdonaključne" funkcije. Člene zaporedja bomo gradili na tak način, da bomo iz števil i in j, $i \neq j$, za kateri je $x_i = x_j$, znali izračunati $a = \log_\alpha \beta$. Da privarčujemo čas in spomin, bomo tudi tokrat iskali trčenja oblike $x_i = x_{2i}$. Predpostavili bomo, da logaritem obstaja, da je torej $\beta \in \langle\alpha\rangle$.

Psevdonaključna funkcija f je tokrat nekoliko bolj zapletena. Označimo z m red elementa $\alpha \in G$. Nadalje naj bo $S_1 \cup S_2 \cup S_3$ razbitje G na tri približno enako velike množice. Za $G = Z_k^*$ lahko na primer definiramo

$$S_i = \{x \in Z_k^* : x \equiv i \pmod 3\} \quad \text{za } i = 1, 2, 3$$

Funkcijo f definiramo tako, da preslikuje trojke $(x_i, A_i, B_i) \in \langle\alpha\rangle \times \mathbb{Z}_n \times \mathbb{Z}_n$:

$$f(x, A, B) = \begin{cases} (\beta x, A, B+1) & \text{če } x \in S_1; \\ (x^2, 2A, 2B) & \text{če } x \in S_2; \\ (\alpha x, A+1, B) & \text{če } x \in S_3. \end{cases}$$

Števili A_i in B_i imata zgolj knjigovodski značaj: za vsako trojko velja $x_i = \alpha^A \beta^B$, in f to lastnost ohranja. Paziti moramo le, da bo tudi prva trojka zadoščala enakosti – začnemo lahko na primer z $(1, 0, 0)$. Ko naletimo na trčenje $x_i = x_{2i}$, lahko iz A_i, B_i, A_{2i} in B_{2i} rekonstruiramo $a = \log_\alpha \beta$. Takrat imamo namreč

$$\alpha^{A_{2i}} \beta^{B_{2i}} = \alpha^{A_i} \beta^{B_i}.$$

Ker je $\beta = \alpha^a$, pri čemer je a zaenkrat neznan, lahko enakost preuredimo v

$$\alpha^{A_{2i} + aB_{2i}} = \alpha^{A_i + aB_i}.$$

Ker je a reda m, z logaritmiranjem dobimo

$$A_{2i} + aB_{2i} \equiv A_i + aB_i \pmod m$$

oziroma

$$a(B_{2i} - B_i) \equiv A_i - A_{2i} \pmod{m}. \qquad (*)$$

Če je $\gcd(B_{2i} - B_i, m) = 1$, smemo enačbo deliti in dobimo enolično rešitev za $a \in \mathbb{Z}_n$:

$$a = (A_i - A_{2i})(B_{2i} - B_i)^{-1} \bmod m.$$

V zapisu v obliki algoritma privzamemo, da razbitje $G = S_1 \cup S_2 \cup S_3$ in funkcija f že obstajata in sta definirana kot zgoraj.

Algoritem 3.2.4 PollardRhoDLP$(n, A_1, B_1, \alpha, \beta)$

Vhod: Naravno število n; naravni števili A_1 in B_1; števili $\alpha \in \mathbb{Z}_n$ in $\beta \in \langle\alpha\rangle$.
Rezultat: Diskretni logaritem $\log_\alpha \beta$ v $\mathbb{Z}_n^*$.

1) $(x, A, B) := (\alpha^{A_1}\beta^{B_1}, A_1, B_1)$
2) $(x', A', B') := f(x, A, B)$
3) $k := 1$
4) **while** $x \neq x' \wedge k < 1.17\sqrt{m}$ **do**
5) $\quad (x, A, B) := f(x)$
6) $\quad (x', A', B') := f(f(x', A', B'))$
7) $\quad k++$
8) **end while**
9) **if** $gcd(b' - b, n) \neq 1 \vee k \geq 1.17\sqrt{m}$ **then**
10) $\quad$ **return** "Podana A_1, B_1 ne generirata rešitve"
11) **else**
12) $\quad$ **return** $(A - A')(B' - B)^{-1} \bmod m$
13) **end if**

Časovna zahtevnost. Analiza je nadvse podobna tisti za Pollardovo metodo ρ za faktorizacijo. Tudi v algoritmu 3.2.4 se namreč z dvema kazalcema sprehajamo po namišljenem grafu na $O(\sqrt{m})$ točk iz grupe G, in z verjetnostjo približno ene polovice kazalca trčita. Ko trčita, se sicer lahko zgodi, da $\gcd(b' - b, n) \neq 1$ in enačbe (*) ne moremo enolično rešiti. Vendar pa je verjetnost za to empirično majhna; in celo takrat, ko naletimo na to možnost, se nam morda še vedno splača s poskušanjem preveriti vse možne rešitve za enačbo (*), ki jih je natanko $\gcd(b' - b, n)$ in med katerimi se skriva tudi prava.

Časovna zahtevnost je torej $O(\sqrt{m})$, kjer je m red α v G. Pri grupah, s katerimi imamo tipično opravka, je ta ponavadi kar enak n, zato je zahtevnost enaka kot pri metodi veliki–mali korak: $O(\sqrt{n})$. Da smo prišli do ocene zahtevnosti, smo enako kot v razdelku 3.1.2 morali predpostaviti psevdonaključnost zaporedja, gene-

riranega s funkcijo f. Dokaza za takšno obnašanje f nimamo, vendar ga meritve iz prakse potrjujejo.

3.2.3 Index calculus

Index calculus, zadnja metoda za izračun diskretnega logaritma, ki si jo bomo ogledali, deluje samo v grupi $\mathbb{Z}_p^*$. V primerjavi z zgoraj opisanima metodama je bistveno hitrejša in je zato povzročila manjšo revolucijo v kriptografiji. Ker mnogi protokoli uporabljajo prav problem diskretnega logaritma nad to grupo, je bilo treba velikosti ključev zaradi nove, učinkovitejše metode pošteno povečati, da so (p)ostali odporni na napade. Hkrati se je močno povečalo zanimanje za grupe točk na eliptičnih krivuljah. Te so sicer manj intuitivne kot obsegi ostankov, a zaenkrat niso občutljive na napad z uporabo index calculusa. Protokoli, ki gradijo na njih, si zato lahko privoščijo uporabljati precej krajše ključe, s tem pa se povečata tako praktičnost kot hitrost računanja.

Da bi našli algoritem za DLP, ki je hiter kot index calculus in *generičen* v smislu, da ga lahko uporabimo v katerikoli grupi, ni upanja. Po Stinsonovih navedbah [17, razdelku 6.3] je Shoup pokazal, da noben generičen algoritem ne more biti bistveno učinkovitejši kot npr. Shanksov algoritem, torej z zahtevnostjo $O(\sqrt{n})$. Seveda to ne izključuje obstoja bistveno učinkovitejšega algoritma, ki bi deloval denimo samo na eliptičnih krivuljah.

Algoritem poteka v dveh korakih in ga bomo zaradi njegove kompleksnosti le skicirali. Določitev podrobnosti algoritma zahteva obsežne in poglobljene raziskave.

V prvem koraku določimo *faktorsko bazo* razmeroma majhnih (glede na p) praštevil. Označevali jo bomo $\mathcal{B} = \{p_1, p_2, \ldots, p_B\}$. Zdaj določimo C, nekoliko večji od B, na primer $C = B + 10$. Konstruirali bomo C kongruenc oblike

$$\alpha^{x_j} \equiv p_1^{a_{1,j}} p_2^{a_{2,j}} \cdots p_B^{a_{B,j}} \pmod{p}$$

za $j = 1, \ldots, C$. Elementaren in razmeroma učinkovit način konstruiranja takšnih kongruenc je, da x_j določimo enostavno naključno, izračunamo α^{x_j} in nato s poskušanjem ugotovimo, ali se ga da faktorizirati z elementi iz $\mathcal{B}$. Obstajajo tudi zapletenejši in učinkovitejši načini, ki pa jih tu ne bomo opisovali.

Nato vseh C kongruenc logaritmiramo, da dobimo za vsak $j = 1, \ldots, C$ enačbo

$$x_j \equiv a_{1,j} \log_\alpha p_1 + \ldots + a_{B,j} \log_\alpha p_B \pmod{p-1}.$$

Dobili smo C linearnih enačb z B neznankami $\log_\alpha p_j$ ($j = 1, ..., B$). Precej verjetno je vsaj B izmed teh enačb linearno neodvisnih in lahko z Gaussovo eliminacijo učinkovito izračunamo logaritme elementov faktorske baze $\mathcal{B}$.

Ko nam uspe tako logaritmirati faktorsko bazo, začnemo z drugo fazo. V tej dejansko izračunamo $\log_\alpha \beta$ s preprostim Las Vegas algoritmom. Izberemo naključno število s ($1 \leq s \leq p-2$) in izračunamo

$$\gamma = \beta\alpha^s \bmod p.$$

Nato poskusimo γ faktorizirati na elemente iz $\mathcal{B}$. Najpreprosteje (ne pa tudi najbolj učinkovito) to naredimo s poskušanjem. Če nam uspe, dobimo kongruenco

$$\beta\alpha^s \equiv p_1^{c_1} p_2^{c_2} \cdots p_B^{c_B} \pmod{p},$$

ki jo logaritmiramo, da pridemo do

$$\log_\alpha \beta + s \equiv c_1 \log_\alpha p_1 + \ldots + c_B \log_\alpha p_B \pmod{p-1}.$$

Od tu je trivialno izraziti $\log_\alpha \beta$, saj so vsi preostali členi že znani.

Časovna zahtevnost. Časovna zahtevnost je odvisna od "podrobnosti", ki smo jih v našem opisu izpustili, predvsem od načina konstruiranja kongruenc v prvi fazi, pa tudi od hevristike za izbiro s v drugi. Empirične analize več različic metode index calculus kažejo, da je časovna zahtevnost prve faze

$$O(e^{(1+o(1))\sqrt{\ln p \ln\ln p}}),$$

druge faze pa

$$O(e^{(1/2+o(1))\sqrt{\ln p \ln\ln p}}).$$

Pri tem smo v zgornjem zapisu z $o(1)$ označili količino, ki ima konstantno zgornjo mejo in limitira proti 0, ko gre p proti neskončnosti. Zahtevnost obeh faz smo zapisali ločeno, ker pogosto srečamo primere, ko je treba izračunati več logaritmov z isto osnovo – v tem primeru je pač potrebno prvo fazo opraviti le enkrat.

3.2.4 Pomen za miselni poker

Če smo za problem faktorizacije zapisali, da je uporaben in uporabljan na skoraj vseh področjih kriptografije, je DLP morda edini problem, za katerega ta trditev velja še izraziteje.

Med drugim je v miselnem pokru po zgledu mnogih drugih kriptografskih shem popularno zakrivanje kart s pomočjo potenciranja: karto z znanim čistopisom x igralec zakrije tako, da jo s svojim zasebnim ključem e pretvori v x^e. Če to naredi za vse karte, nato pa dobljene vrednosti še permutira med seboj, preden jih objavi,

soigralci ne znajo več povezati tajnopisov s čistopisi. To je osnovna ideja najpogostejše izvedbe mešanja kart v miselnem pokru. (Če jo hočemo izvesti korektno, stvari niso tako preproste, saj mora igralec dokazati, da so tajnopisi res potence čistopisov.)

Ker je DLP izrazito bazičen problem, v implementacijah miselnega pokra nastopa tudi kot gradnik različnih protokolov. Na njem je na primer osnovan marsikakšen dokaz brez razkritja znanja (glej §3.5), konstrukt, ki zelo olajša snovanje miselnega pokra brez TTP.

Neizmerno uporabna sta tudi oba Diffie-Hellmannova problema. Boneh [1] ju predstavi kar kot "zlato jamo kriptografije". Med drugim je tudi shema za miselni poker, ki jo bomo spoznali v razdelku 4.3, močno odvisna od DDH.

3.3 Problem kvadratnih ostankov in korenjenje v $\mathbb{Z}_p^*$

Problem kvadratnih ostankov je še zadnji od osnovnih matematično-računskih problemov s področja kriptografije, ki si ga bomo pobližje ogledali. Od nas zahteva, da za dano število $a \in \mathbb{Z}_n^*$ povemo, ali je kvadratni ostanek po modulu n.

Definicija 3.3.1. Število $a \in \mathbb{Z}_n^*$ je *kvadratni ostanek (po modulu n)*, če obstaja $y \in \mathbb{Z}_n^*$, za katerega je $a \equiv y^2 \pmod{n}$.

Za lažje zapisovanje vpeljemo Legendrov simbol.

Definicija 3.3.2. Za celo število a ter praštevilo p definiramo *Legendrov simbol* kot

$$\left(\frac{a}{p}\right) := \begin{cases} 0 & \text{če } p \mid a \\ 1 & \text{če } p \nmid a \text{ in } \exists y \in \mathbb{Z}_p : a \equiv y^2 \pmod{p} \\ -1 & \text{sicer} \end{cases}$$

Povedano drugače: če je a večkratnik števila p, je $\left(\frac{a}{p}\right) = 0$, sicer pa velja $\left(\frac{a}{p}\right) = 1$ natanko tedaj, ko je a kvadratni ostanek po modulu p.

Če je n praštevilo, znamo problem kvadratnih ostankov hitro rešiti. Avtor naslednjega zelo učinkovitega kriterija je Gauss.

Izrek 3.3.3. *Za naravno število a ter liho praštevilo p velja*

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p}.$$

Dokaz. Za $a \equiv 0 \pmod{p}$ enakost očitno velja, saj sta obe strani kongruentni 0. Če je $a \equiv y^2 \pmod{p}$ za nek y, potem imamo

$$a^{(p-1)/2} \equiv (y^2)^{(p-1)/2} \equiv y^{p-1} \equiv 1 \pmod{p},$$

kjer sledi zadnja enakost po Fermatovem izreku. Velja tudi obratno. Denimo, da je $a^{(p-1)/2} \equiv 1 \pmod{p}$. Pokazati želimo, da je v tem primeru a nujno kvadratni ostanek po modulu p. Naj bo b generator v $\mathbb{Z}_p^*$. Potem obstaja tako naravno število i, da je $a \equiv b^i \pmod{p}$. Pišemo lahko

$$a^{(p-1)/2} \equiv (b^i)^{(p-1)/2} \equiv b^{i(p-1)/2} \pmod{p}.$$

Ker je generator b reda $p-1$, mora $p-1$ deliti eksponent $i(p-1)/2$. Torej je kvocient $\frac{i}{2}$ celo število, zato števili $\pm b^{i/2} \bmod p$ obstajata in sta korena števila a. □

Problem kvadratnih ostankov lahko za praštevilski n z uporabo izreka 3.3.3 rešimo v polinomskem času. Če je n sestavljen in poznamo njegove faktorje, ni naloga skoraj nič težja, saj je število a po kitajskem izreku o ostankih kvadraten ostanek po modulu n natanko tedaj, ko je kvadraten ostanek po modulu vseh prafaktorjev n. (Nekaj več o tem v naslednjem razdelku, 3.3.1.)

Mnogo težje je, če je n sestavljeno število, za katerega ne poznamo faktorizacije. Zanimivo je, da v tem primeru niso znani splošni algoritmi, ki bi na zgornji, odločitveni problem ("Ali obstaja koren iz a po modulu n?") odgovorili učinkoviteje kot na računsko različico ("Koliko je koren iz a po modulu n?").

V nekaterih primerih vendarle znamo na odločitveno vprašanje odgovoriti hitro, četudi je n sestavljen in ne poznamo njegove faktorizacije. Pri tem nam pomaga posplošitev Legendrovega simbola.

Jacobijev simbol za $a \in \mathbb{Z}$ ter $n \in \mathbb{Z}\backslash\{-1, 0, 1\}$ definiramo kot

$$\left(\frac{a}{n}\right) := \left(\frac{a}{p_1}\right)\left(\frac{a}{p_2}\right)\cdots\left(\frac{a}{p_k}\right),$$

kjer je $p_1 p_2 \cdots p_k$ praštevilska faktorizacija (s ponovitvami) števila n.

Jacobijev simbol ohranja nekatere lastnosti Legendrovega. Če je a kvadratni ostanek po modulu n, je $\left(\frac{a}{n}\right) = 1$, saj je a kvadrat tudi po modulu vsakega od prafaktorjev n. Ne velja pa obratno: če je $\left(\frac{a}{n}\right) = 1$, število a ni nujno kvadratni ostanek. Zadošča, da je nekvadraten ostanek po modulu sodo mnogo prafaktorjev – lahko tudi 0, kar se zgodi, ko je a vendarle kvadraten po modulu n.

Bolje lahko sklepamo na nekvadratnost. Če je $\left(\frac{a}{n}\right) = -1$, mora biti vsaj eden od Legendrovih simbolov iz definicije $\left(\frac{a}{n}\right)$ enak -1. To pomeni, da je a nekvadraten po modulu enega od prafaktorjev n in zato tudi po modulu n.

Zakaj toliko govora o razmerju med Jacobijevim simbolom in (ne)kvadratnostjo? Navidez nam Jacobijev simbol ne pomaga pri učinkovitem reševanju problema kvadratnih ostankov, saj moramo še vedno poznati faktorizacijo n. Vendar pa se izkaže, da lahko vrednost Jacobijevega simbola hitro izračunamo z uporabo izreka 3.3.3 in naslednjih lastnosti, ki jih navajamo brez dokaza:

1. Če je n lih in pozitiven, potem velja
$$\left(\frac{m}{n}\right) = \left(\frac{m \bmod n}{n}\right).$$

2. Če je n lih in pozitiven, potem velja
$$\left(\frac{2}{n}\right) = \begin{cases} 1 & \text{če } n \equiv \pm 1 \pmod 8; \\ -1 & \text{če } n \equiv \pm 3 \pmod 8. \end{cases}$$

3. Če je n lih in pozitiven, potem velja
$$\left(\frac{m_1 m_2}{n}\right) = \left(\frac{m_1}{n}\right)\left(\frac{m_2}{n}\right).$$
Še posebej uporabna je ta lastnost za primer $m_1 = 2$.

4. Če sta m in n liha in pozitivna, potem velja
$$\left(\frac{m}{n}\right) = \begin{cases} -\left(\frac{n}{m}\right) & \text{če } m \equiv n \equiv 3 \pmod 4; \\ \left(\frac{n}{m}\right) & \text{sicer.} \end{cases}$$

Z iterativno uporabo teh lastnosti v vrstnem redu 4, 1, 3, 2 hitro pridemo do končne vrednosti simbola. Iterativna uporaba prve lastnosti močno spominja na Evklidov algoritem, zato se da za izračun Jacobijevega simbola pokazati časovno zahtevnost $O(\log^2 \max(m, n))$. Uporaba preostalih lastnosti med dvema zaporednima uporabama prve namreč le za konstanten faktor raztegne čas izvajanja.

Za tiste nekvadratne ostanke a po modulu n, za katere je $\left(\frac{a}{n}\right) = -1$, lahko torej tudi za sestavljen n hitro ugotovimo, da so nekvadratni, ne da bi poznali faktorizacijo n. Ta ugotovitev je za kriptografijo pomembna. Če hočemo osnovati protokol na problemu kvadratnih ostankov, zdaj vemo, da moramo za modul uporabiti sestavljen n s tajno faktorizacijo, za število a, po katerega kvadratnosti sprašuje problem, pa mora veljati $\left(\frac{a}{n}\right) = 1$. V nasprotnem primeru postane problem lahek.

3.3.1 Povezava med korenjenjem in faktorizacijo

Algoritmi, ki eksplicitno korenijo x v multiplikativni grupi ostankov po modulu n, so zelo dobro raziskani. Razlog je v tesni povezanosti tega problema s problemom faktorizacije.

Trditev 3.3.4. *Problem korenjenja v $\mathbb{Z}_n^*$, kjer je n produkt dveh različnih praštevil, je časovno polinomsko enakovreden problemu faktorizacije števila n.*

Dokaz. Denimo, da znamo učinkovito faktorizirati število $n = pq$ in pri danem a iščemo $x \in \mathbb{Z}_n^*$, za katerega je $a \equiv x^2 \pmod{n}$, ali pa dokaz, da tak x ne obstaja.

Denimo, da tak x obstaja. Tedaj lahko za prafaktor p učinkovito najdemo tak y, da velja $a \equiv y^2 \pmod{p}$. Število $y := a^{p/2}$ namreč zadosti pogoju, saj je p praštevilo in je po Fermatovem izreku

$$y^2 \equiv (a^{p/2})^2 \equiv a^p \equiv a \pmod{p}.$$

Podobno dobimo tudi $z : a \equiv z^2 \pmod{q}$. Po kitajskem izreku o ostankih lahko iz y in z hitro konstruiramo število x, za katerega je $x \equiv y \pmod{p}$ in $x \equiv z \pmod{q}$. Če enakosti kvadriramo, dobimo $x^2 \equiv y^2 \equiv a \pmod{p}$ in $x^2 \equiv z^2 \equiv a \pmod{q}$. Ker sta p in q tuji števili, iz $x^2 \equiv a \pmod{p, q}$ sledi $x^2 \equiv a \pmod{pq}$. Našli smo torej kvadratni koren iz a po modulu $n = pq$.

Videli smo, da iz kvadratnosti po modulih p in q sledi kvadratnost po modulu $n = pq$. Seveda velja tudi v obratno smer: če je neko število a kvadratni ostanek po modulu n, je nujno tudi kvadratni ostanek po modulih p in q, saj iz $a \equiv x^2 \pmod{n}$ za isti x sledi $a \equiv x^2 \pmod{p}$ in $a \equiv x^2 \pmod{q}$. Kvadratnost po modulu n ter hkratna kvadratnost po modulih p in q sta torej ekvivalentni lastnosti.

Poglejmo še možnost, ko a ni kvadratni ostanek po modulu n. Po prejšnjem odstavku a ni kvadratni ostanek tudi po vsaj enem od modulov p in q. To pa bomo po izreku 3.3.3 znali hitro ugotoviti. Poznavanje faktorizacije števila $n = pq$ nam torej res omogoči, da učinkovito korenimo poljubno števila po modulu n.

Dokažimo trditev še v drugo smer. Denimo, da imamo učinkovit algoritem K za korenjenje v multiplikativni grupi $(\mathbb{Z}_n^*, \times)$. Ker je n oblike pq, ima po kitajskem izreku o ostankih vsak kvadratni ostanek a 4 različne korene iz $\mathbb{Z}_n$: označimo jih x, $-x$, y, $-y$. Za faktorizacijo n korene izkoristimo, kot opisuje naslednji odstavek.

Izberemo naključno število x, izračunamo x^2 in poženemo $K(x^2)$. Ker smo vrednost x izbrali neodvisno od rezultata $y := K(x^2)$, bo algoritem z verjetnostjo $\frac{1}{2}$ vrnil $y \not\equiv \pm x \pmod{n}$. Algoritem poženemo večkrat zapored, dokler res ne dobimo takšnega y. Pričakovano potrebujemo dve iteraciji, kar ne vpliva na polinomskost časovne zahtevnosti algoritma.

Zdaj imamo x in $y \in \mathbb{Z}_n^*$, za katera velja $x^2 \equiv y^2 \pmod{n}$, $x \not\equiv \pm y \pmod{n}$. Torej je

$$x^2 - y^2 \equiv 0 \pmod{n}$$

$$(x - y)(x + y) = kn \text{ za nek } k$$

Zaradi $x \not\equiv \pm y \pmod{n}$ ni noben od faktorjev na levi večkratnik n, torej je $\gcd(x - y, n)$ netrivialen delitelj n. □

Za (po naravi odločitven) problem kvadratnih ostankov ter problem korenjenja velja torej podobno kot za Diffie-Hellmannova problema (glej §3.2): odločitveni problem je videti lažji, vendar najučinkovitejše metode za njegovo reševanje vodijo preko algoritmov težjega, računskega problema. Zato tudi odločitveni problem smatramo za težkega in ga s pridom uporabljamo v kriptografiji.

Kot smo videli, je problem korenjenja (in s tem problem kvadratnih ostankov) zelo tesno povezan s problemom faktorizacije. Algoritmi, časovne zahtevnosti in priporočene velikosti ključev, ki smo jih navedli v razdelku 3.1, zato veljajo tudi tu.

3.3.2 Pomen za miselni poker

Problem kvadratnih ostankov je pomemben že zaradi svoje prepletenosti z vseprisotnim problemom faktorizacije. Predvsem pa se v miselnem pokru uporablja kot matematična osnova za Goldwasser-Micalijev protokol za zaprisego bitov (glej §3.4.2), mehanizem, brez katerega si težko zamislimo shemo za miselni poker brez TTP.

Pri nekaterih shemah za miselni poker moramo biti posebej pozorni na dejstvo, da znamo v praštevilski grupi hitro določiti, ali je določeno število kvadratni ostanek. Denimo, da imamo shemo, ki karte predstavi s števili iz $\mathbb{Z}_p$. Imejmo dve karti, a in b, pri čemer je le a kvadratni ostanek. Če zdaj izkoristimo problem diskretnega logaritma in karti zašifriramo s ključem e, kot je nakazano v razdelku 3.2.4 in kot sheme za miselni poker dejansko pogosto počnejo, dobimo a^e in b^e. Domnevno nihče, ki ne pozna vrednosti e, ne more povedati, v katerega od obeh tajnopisov je bil zašifriran a. A če smo bili pri šifriranju nepozorni in uporabili lih e, je a^e kvadratni ostanek (enako kot a) in b^e nekvadratni ostanek (enako kot b).

3.4 Zapriseženi biti

Oglejmo si za uvod težavo večnih protagonistov šifrirnih protokolov, Ane in Boruta. Ana in Borut bi se rada po telefonu dogovorila, kam naj odpotujeta na počitnice. Izbiro sta že skrčila na štiri mesta: mistični *Aa*, metropolitanski *Ab*, romantični *Ba* in glamurozni *Bb*. Dogovorita se, da bo Ana izbrala prvo črko imena, Borut pa

drugo. Vendar pa želita izbiro napraviti popolnoma neodvisno drug od drugega, zato nihče ne sme svoje odločitve povedati prvi.

Ena možna rešitev je takšnale: Ana svojo odločitev zapiše na list papirja, ga zaklene v sef, sef po pošti pošlje Borutu, ključ sefa pa obdrži. Njene odločitve zdaj ne more spremeniti nihče več; pravimo, da je bila odločitev *zaprisežena*. Borut po telefonu sporoči svojo odločitev in Ana mu pošlje ključ. Četudi bi zdaj, ko je slišala Borutovo odločitev, morda želela spremeniti svojo, je Borut lahko prepričan, da bo prebral Anino originalno odločitev.

Protokol za zaprisego bitov (angl. *bit commitment protocol*) je šifrirni protokol, ki nam omogoča, da vse skupaj opravimo brez sefov ali drugih fizičnih pripomočkov, saj je pošiljanje fizičnega ključa lahko zelo drago ali nevarno. Zapišimo to še formalno:

Definicija 3.4.1. Naj bo $b \in \{0, 1\}$ bit informacije, znan samo Ani, X in Y pa končni množici. *Protokol za zaprisego bitov* je protokol, v katerem osrednjo vlogo igra funkcija $f : \{0, 1\} \times X \to Y$, ki ima naslednji lastnosti:

- f **prikriva**: Borut iz vrednosti $f(b, x)$ ne more izvedeti ničesar o vrednosti bita b.
- f **zavezuje**: Ana ne sme biti sposobna izračunati x', za katerega bi bilo $f(b, x) = f(1 - b, x')$. To zagotavlja, da bo z razkritjem x Boruta prepričala, da je zašifrirala prav bit b.

Protokol je okoli funkcije zgrajen takole: Ana izbere sporočilo b ter ključ x, uporabi funkcijo f in Borutu sporoči $f(b, x)$, število x pa zadrži. Borut zaradi lastnosti f ne more ugotoviti ničesar o bitu b. Ko Ana želi Boruta prepričati, da je zaprisegla prav b, razkrije x in Borut preveri, da je $f(b, x)$ res zašifrirano sporočilo, ki ga je dobil na začetku.

Po naši analogiji s sefom je element x v zgornji definiciji ključ sefa, element y pa v sefu zaklenjen list papirja z odločitvijo. Čeprav se v definiciji omejimo na eno-bitne odločitve, očitno lahko protokol uporabimo tudi na večjih podatkih: zaprisežemo jih enostavno bit po bit.

Ogledali si bomo dve implementaciji protokola za zaprisego bitov.

3.4.1 Implementacija z zgoščevalno funkcijo

Začnimo z definicijo dobre zgoščevalne funkcije, ki je v kriptografiji nekoliko drugačna kot npr. pri rabi za razpršene tabele.

Definicija 3.4.2. Naj bosta X in Y končni množici. *Zgoščevalna funkcija* (angl. hash function) je funkcija $h : U \to V$, ki ima naslednje lastnosti:

- iz vrednosti $h(u)$ je nemogoče v doglednem času pridobiti kakršnokoli informacijo o prasliki u;
- za poljuben $v \in V$ je nemogoče v doglednem času poiskati u', za katerega je $h(u') = v$. Enako nemogoče je zato za poljuben $u \in U$ izračunati u'', da bo $h(u'') = h(u)$;
- v doglednem času je nemogoče najti različni vrednosti $u, u' \in U$, za kateri je $h(u') = h(u)$.

Zgoščevalne funkcije v resnici delimo glede na vrsto lastnosti, ne le na "dobre" in "slabe", a v to področje se ne bomo brez potrebe poglabljali. Povejmo le, da so zgoraj postavljene zahteve razmeroma ostre, vendar poznamo funkcije, ki jim zadoščajo. Danes so največ v rabi funkcije SHA-1, SHA-256 in SHA-512, še vedno pa je popularna tudi starejša MD5, čeprav so jo kriptografi analitično že dodobra načeli in je varna le še za doseganje kratkoročne varnosti.

Kot vidimo, sta prvi dve lastnosti iz definicije 3.4.1 podobni tistim iz definicije 3.4.2, zato lahko iz zgoščevalne funkcije preprosto izpeljemo shemo za zapriseže bite.

Trditev 3.4.3. *Naj bo* $h : \mathbb{Z}_{2n} \to \mathbb{Z}_n$ *zgoščevalna funkcija, kot smo jo definirali zgoraj. Potem je* $f : \{0,1\} \times \mathbb{Z}_n \to \mathbb{Z}_n, \quad f : (b, x) \mapsto h(2x + b)$, *osnova za protokol za zaprisego bitov.*

Dokaz. Funkcija f ima lastnost prikrivanja neposredno po definiciji zgoščevalne funkcije. Prav tako f zavezuje. Če bi Ana znala učinkovito izračunati x', za katerega je $f(b, x) = f(1 - b, x')$, bi to pomenilo, da zna za $u = 2x + b$ izračunati $u' = 2x' + (1 - b)$, za katerega je $f(1 - b, x') = h(u') = h(u) = f(b, x)$. To pa je v protislovju z drugo lastnostjo zgoščevalne funkcije h. □

Varnost takšnega protokola za zapriseže bite očitno temelji na varnosti uporabljene zgoščevalne funkcije. Sodobne zgoščevalne funkcije, na primer SHA-512, v praksi sicer dobro izpolnjujejo zahteve iz definicije 3.4.2, v teoriji pa tega zaradi njihove kompleksnosti ne znamo dokazati. Za zaprisego bitov se zato navadno uporablja protokol, ki ga bomo predstavili kot naslednjega in katerega varnost je lažje analizirati.

3.4.2 Goldwasser-Micalijev protokol

Goldwasser-Micalijev protokol je bil prvotno zamišljen kot šifrirna shema: zapriseženi biti so bili tajnopis, x iz naše definicije sheme za zaprisego bitov pa je bil ključ. Vendar pa bomo videli, da se tako vsak bit zašifrira v sporočilo, dolgo nekaj sto bitov, kar je v praksi redko sprejemljivo. Razlog takemu razpihovanju je, da gre

za *verjetnostno šifriranje*: isti čistopis se naključno šifrira v enega od mnogih tajnopisov. Za zaprisego bitov je protokol uporaben prav zaradi te lastnosti. "Šifriramo" oziroma zaprisegamo namreč le dve različni vrednosti, 0 ali 1, tajnopisov pa mora biti mnogo različnih, da iz njih ne bomo mogli sklepati na čistopis.

Protokol temelji na problemu kvadratnih ostankov (glej §3.3). Funkcijo f definira kot

$$f(b,(p,q)) := (y^b x^2 \bmod n,\ \ n,\ \ y),$$

kjer je b kot prej bit, ki ga zaprisegamo, x je naključno število, p in q sta praštevili, n zadošča enakosti $n = pq$, število y je pa naključno izbran nekvadratni ostanek po modulu n, za katerega ima Jacobijev simbol vrednost $\left(\frac{y}{n}\right) = 1$. Praštevili p in q morata biti dovolj veliki, da problem faktorizacije števila n ni rešljiv v razumnem času. Ekvivalentno to pomeni, da za poljubno dano število ne znamo razumno hitro določiti, ali je kvadratni ostanek po modulu n.

Glede na omejitve, ki smo jih postavili za število y, mora biti $\left(\frac{m}{p}\right) = \left(\frac{m}{q}\right) = -1$. Tak y lahko preprosto dobimo s poskušanjem, saj je računanje Jacobijevih simbolov učinkovito, števil z zahtevano lastnostjo pa dovolj – v praštevilskih grupah, kot sta $\mathbb{Z}_p$ in $\mathbb{Z}_q$, je nekvadratnih ostankov natanko polovica.

Kot smo opisali na začetku razdelka o shemah za zaprisego bitov, Ana zapriseže bit tako, da objavi vrednost $f(b,(p,q))$ in zase zadrži par (p,q). Ko želi razkriti bit b, objavi še faktorizacijo števila $n = pq$.

Trditev 3.4.4. *Zgoraj opisana Goldwasser-Micalijeva funkcija f je osnova za protokol za zaprisego bitov. Pri tem predpostavimo, da je problem kvadratnih ostankov (§3.3) časovno neobvladljiv.*

Dokaz. Po domeni in kodomeni funkcija f ustreza definiciji protokola za zaprisego bitov. Preveriti moramo še, da ima obe zahtevani lastnosti.

Označimo z $f_1(b,(p,q))$ prvi element trojke $f(b,(p,q))$. Vrednost $f_1(b,(p,q))$ je po definiciji funkcije f kvadratni ostanek natanko tedaj, ko je $b = 0$.

Pokažimo najprej, da funkcija res zakriva. Neodvisno od vrednosti b je

$$\left(\frac{f_1(b,(p,q))}{n}\right) = \left(\frac{y^b}{n}\right)\left(\frac{x^2}{n}\right) = 1 \cdot 1 = 1.$$

Jacobijev simbol nam torej ne pomaga razkriti vrednosti b, ostale tehnike pa so po predpostavki prepočasne. Da bi učinkovito ugotovili, ali je $f_1(b,(p,q))$ kvadratni ostanek (in s tem izvedeli vrednost b), moramo poznati p in q.

Funkcija f tudi zavezuje, celo dosti močneje kot tista, ki smo jo srečali pri implementaciji z zgoščevalno funkcijo. Tam je f zavezovala samo *računsko* – najti ključ, ki bi zašifrirano vrednost odklenil v "napačen" b, je bilo časovno prepotratno, ne pa teoretično nemogoče.

Tokratna funkcija f zavezuje *popolno*. Ne glede na razpoložljivo računsko moč ne moremo potvoriti originalno zapriseženega bita. Število $n = pq$ je namreč del zašifrirane vrednosti, in vsak se lahko prepriča, da je ključ (p, q) res faktorizacija števila n. Ko je par (p, q) znan, z uporabo izreka 3.3.3 ni več težko preveriti, ali je $f_1(b, (p, q))$ kvadratni ostanek in od tod sklepati na vrednost b. □

3.4.3 Pomen za miselni poker

Protokoli za zaprisego bitov igrajo pomembno vlogo pri shemah za miselni poker. Primer iz uvoda v razdelek o zapriseženih bitih, kjer se Ana in Borut odločata o cilju svojega potovanja, se skoraj nespremenjen pojavi v miselnem pokru. Igralci se morajo namreč dogovoriti, s kakšno permutacijo naj se premešajo karte, in če naj bo dogovor pošten, mora vsak igralec svoj prispevek podati neodvisno od drugih – podobno, kot sta Ana in Borut morala neodvisno prispevati vsak po eno črko imena mesta. (Kaj natanko je "prispevek", seveda ni natančno določeno in je odvisno od konkretne sheme. Pogosto vsak od igralcev zapriseže permutacijo 52 elementov, za končno mešanje pa se uporabi kompozitum teh permutacij.)

Zapriseženi biti so tudi element skoraj vseh dokazov brez razkritja znanja (glej naslednji razdelek), še enega orodja, brez katerega si miselni poker brez TTP težko zamislimo.

3.5 Dokazi brez razkritja znanja (ZKP)

Dokaz brez razkritja znanja (ZKP – Zero Knowledge Proof) je neformalno protokol, ki omogoča osebi P, da drugo osebo V prepriča o nekem dejstvu, ne da bi pri tem izdala kakršnokoli informacijo o tem, kako dejstvo dokazati. Osebo P (*prover* – dokazovalec) bomo poimenovali Pika, osebo V (*verifier* – preverjevalec) pa Viktor.

Z dokazom brez razkritja znanja lahko na primer Pika prepriča Viktorja, da pozna faktorizacijo števila $n = pq$, ne da bi mu razkrila en sam bit informacije o faktorjih p ali q. Ali pa ga lahko prepriča, da ve, ali je a kvadratni ostanek po sestavljenem modulu n, ne da bi razkrila faktorizacijo n ali pa predstavila a kot potenco znanega (ne)kvadrata. Dejstvo, o katerem Pika želi prepričati, ni nujno očitno številske narave. Dobro znan je na primer ZKP, s katerim dokažemo, da sta dana grafa izomorfna, ne da bi pri tem razkrili izomorfizem (ki ga je v splošnem domnevno težko izračunati).

Preden definiramo dokaz brez razkritja znanja, moramo najprej formalno opredeliti pojem dokaza.

Definicija 3.5.1. Naj bo Π odločitveni problem. Imejmo protokol med Viktorjem in Piko, v katerem Viktor najprej poda verjetnosten izziv (Π, x), Pika pa nanj odgovori. Viktor odgovor interpretira kot dokaz in ga *sprejme* ali *zavrne* glede na to, ali je $x \in \Pi$. Tak protokol je *interaktivni dokaz* za problem Π, če veljata naslednji lastnosti, čim Viktor sledi protokolu:

1. **polnost** (completeness) – če je $x \in \Pi$, Viktor vedno sprejme dokaz;
2. **uglašenost** (soundness) – če je $x \notin \Pi$, Viktor sprejme dokaz z verjetnostjo, manjšo od 1/2.

Pri tem dovolimo Piki neomejeno računsko moč, Viktorja pa časovno verjetnostno polinomsko omejimo.

Vpeljimo varnostni parameter $s \in \mathbb{N}$. Protokol lahko ponovimo s-krat z istim vhodom x. Ker je Viktorjev algoritem verjetnosten, bosta izziv in odgovor vsakič različna in po s ponovitvah bo verjetnost, da Viktor nepravilno sprejme dokaz za $x \notin \Pi$, manjša od 2^{-s}.

Vse informacije, ki jih dobi Viktor tekom izvajanja protokola, imenujemo Viktorjev *zapis*. Algoritmu S, ki poskuša potvoriti tak zapis, ne da bi komuniciral s Piko, pravimo *simulator*. Če naj interaktivni dokaz ne razkrije nič znanja, ne sme biti v Viktorjevem zapisu nobenih dodatnih informacij glede na simulatorjev zapis.

Definicija 3.5.2. Imejmo interaktivni dokaz za odločitveni problem Π med dokazovalcem P in verjetnostnim preverjevalcem V. Interaktivni dokaz je *dokaz brez razkritja znanja*, če za vsakega preverjevalca V (tudi goljufivega, ki izzivov ne izbira enakomerno naključno) in za vsak pozitiven primer $x \in \Pi$ obstaja simulator S, katerega zapisa v polinomskem času ne moremo ločiti zapisa, ki ga generira preverjevalec ob interaktivnem dokazu za x [14, razdelek 12.3].

Tu smo se omejili na *računsko nerazkritje znanja*, saj zahtevamo, da je obe distribuciji zapisov nemogoče ločiti v doglednem (polinomskem) času. Obstaja mnogo različic definicije za dokaz brez razkritja znanja; nekatere zahtevajo tudi *popolno nerazkritje znanja*, kjer morata biti distribuciji zapisov povsem ekvivalentni.

3.5.1 ZKP za Hamiltonov cikel

Kot ilustracijo si oglejmo dokaz brez razkritja znanja, s katerim lahko Pika dokaže, da v danem grafu G obstaja Hamiltonov cikel, ne da bi ga razkrila.

Algoritem 3.5.5 HAMILTONOVCIKELZKP(G, s)

Vhod: Graf G; varnostni parameter s (npr. $s = 20$).

1) Pika generira graf H, izomorfen grafu G, in ga objavi. Izomorfizem zadrži zase.
2) Viktor naključno izbere enega od dveh izzivov. Prvi je, naj Pika razkrije izomorfizem med G in H. Drugi je, naj mu pokaže Hamiltonov cikel v H.
3) Pika ustrezno odgovori na Viktorjev izziv.
4) Pika in Viktor korake od 1 do 3 ponovita s-krat.

Trditev 3.5.3. *Zgoraj opisani protokol je dokaz brez razkritja znanja. Pri tem predpostavimo, da je problem iskanja izomorfizma med dvema podanima grafoma časovno neobvladljiv.*

Dokaz. Najprej pokažimo, da gre za interaktiven dokaz. Če Pika res pozna Hamiltonov cikel v G, ga bo znala preslikati v Hamiltonov cikel v H s preslikavo, s katero je preslikala G v H. Tako bo vedno znala odgovoriti na oba možna Viktorjeva izziva.

Če Pika Hamiltonovega cikla v G ne pozna, se lahko pripravi le na en Viktorjev izziv. Lahko pripravi graf H, izomorfen grafu G, in upa, da bo Viktor poslal prvi izziv, saj nanj zna odgovoriti. Na drugi izziv v tem primeru seveda ne zna odgovoriti, ker Hamiltonovega cikla v grafu G (in s tem v H) po predpostavki ne pozna. Lahko pa neodvisno od G pripravi nov graf H, v katerem namerno zgradi cikel – v tem primeru bo znala odgovoriti na drugi izziv, ne pa tudi na prvega, saj izomorfizem med G in H najverjetneje sploh ne obstaja (če pa obstaja, ga je po predpostavki nemogoče hitro najti).

Pokažimo še nerazkritje znanja. Viktor na svoj prvi izziv prejme v odgovor le izomorfizem med grafom G in namensko konstruiranim grafom H – očitno nič takšnega, kar bi mu pomagalo izvedeti kaj o Hamiltonovem ciklu v G. Na svoj drugi izziv pa prejme informacijo o Hamiltonovem ciklu v H, vendar ne pozna preslikave iz H v G in je po predpostavki v dostojnem času tudi ne more izračunati niti preveriti, ali taka preslikava sploh obstaja[2]. Viktor torej ne prejme nobene informacije o Hamiltonovem ciklu v G, zato bi lahko njegove zapise tvoril tudi simulator. □

3.5.2 Chaum-Pedersenov ZKP

Če je bil ZKP za Hamiltonov cikel zgolj ilustrativne narave, bomo Chaum-Pedersenov dokaz brez razkritja znanja [6] v četrtem poglavju, ko bomo opisovali konkretno izvedbo sheme za miselni poker, še močno potrebovali.

[2]Predpostavljamo seveda, da je G velik. Za majhne grafe lahko Viktor Hamiltonov cikel poišče kar sam in se mu ni treba ubadati s Piko.

Razlog je, da se tesno navezuje na popularni odločitveni Diffie-Hellmannov problem, ki smo ga opisali v razdelku 3.2. Tam smo povedali, da je odgovor na vprašanje $\log_{g_1} y_1 \stackrel{?}{=} \log_{g_2} y_2$ v grupi $\mathbb{Z}_p^*$ domnevno težek. Če pa nekdo ve, da je odgovor pritrdilen in celo pozna vrednost obeh logaritmov, lahko to brez razkritja znanja dokaže s Chaum-Pedersenovim dokazom.

Protokol 3.5.6 ChaumPedersen$(p, g_1, y_1, g_2, y_2, x)$

Vhod: Praštevilo p; števila $g_1, y_1, g_2, y_2 \in \mathbb{Z}_p^*$; vrednost diskretnega logaritma $x = \log_{g_1} y_1 = \log_{g_2} y_2$. Pri tem je število x znano samo dokazovalcu.

1) Dokazovalec (Pika) izbere naključen $s \in \mathbb{N}$ in objavi $s_1 := g_1^s$ ter $s_2 := g_2^s$.
2) Preverjevalec (Viktor) izbere izziv $c \in \mathbb{N}$ in ga pošlje Piki.
3) Pika vrne $t := s + cx$.
4) Viktor sprejme dokaz, če velja $g_1^t \equiv s_1 y_1^c \pmod{p}$ in $g_2^t \equiv s_2 y_2^c \pmod{p}$.

Trditev 3.5.4. *Chaum-Pedersenov protokol je dokaz brez razkritja znanja. Pri tem predpostavimo, da je problem diskretnega logaritma računsko neobvladljiv.*

Dokaz. Najprej dokažimo, da gre res za interaktiven dokaz. Če Pika pozna x, bo Viktor v 4. koraku sprejel dokaz, saj bo res veljalo

$$g_1^t \equiv g_1^{s+cx} \equiv g_1^s (g_1^x)^c \equiv s_1 y_1^c \pmod{p}$$

in podobno za g_2^t.

Če Pika vrednosti x ne pozna, bi morala, da bo dokaz sprejet, tvoriti tak t, da bo zanj med drugim veljalo $g_1^t \equiv s_1 y_1^c \pmod{p}$. Tvoriti mora torej $t := \log_{g_1}(s_1 y_1^c)$. Pika sicer sama izbere vrednost s_1, vendar še preden izve c, zato na vrednost $s_1 y_1^c$ ne more vplivati. Izračunati mora torej splošen diskretni logaritem, to pa je neobvladljiv problem.

Poglejmo še, ali Viktor res ne prejme nobene informacije o x. Od Pike dobi števila s_1, s_2 in t. Od teh je od vrednosti x odvisno le število t. Da bi Viktor izračunal x, mora torej ugotoviti vrednost s. Dokler je ne ugotovi, deluje s v definiciji števila t kot enkratni ščit za x in Viktor iz vrednosti t ne more razbrati niti bita informacije o x. Da pa bi Viktor prišel do števila s, mora izračunati diskretni logaritem $\log_{g_1} s_1$ ali $\log_{g_2} s_2$, kar je po predpostavki neobvladljiv problem. □

3.5.3 Pomen za miselni poker

V shemah za miselni poker brez TTP so igralci brez centralne, zaupanja vredne avtoritete prepuščeni sami sebi, ko morajo svoje soigralce prepričati, da igrajo

pošteno. “Igrati pošteno” tipično pomeni, da imajo operacije šifriranja, ki jih igralec izvede, določene dogovorjene lastnosti. Če želi igralec te lastnosti dokazati, ne da bi razkril ključ šifriranja, se kot idealen kandidat za dosego tega cilja ponujajo dokazi brez razkritja znanja.

Kjer gre za dokazovanje pravilnosti izvedbe preproste in pogoste operacije, lahko sheme za miselni poker izkoristijo dokaze brez razkritja znanja, ki so jih drugi kriptografi odkrili že prej. Tak primer je potenciranje s skrivnim eksponentom – tu nam pogosto lahko pomaga Chaum-Pedersenov dokaz. Včasih pa želimo za rezultat operacije dokazati kakšno lastnost, ki je specifična za konkretno shemo za miselni poker. V tem primeru moramo tudi ZKP zasnovati namensko.

Poglavje 4

Primeri shem za miselni poker

To poglavje predstavlja najpomembnejši del diplomske naloge, saj bomo v njem predstavili rešitev problema, ki smo si ga zadali že v uvodu.

Pri formalnem opisu varnostnih zahtev v razdelku 2.3 smo že omenili, da mnoge izmed doslej predlaganih shem ne izpolnjujejo vseh naštetih pogojev. Med tovrstnimi delnimi rešitvami je še posebej veliko takih, ki se ne izognejo uporabi TTP. Za takšen pristop imajo utemeljen razlog, ki si ga bomo skupaj s primerom rešitve s TTP ogledali v prvem delu tega poglavja.

Sledi zgodovinsko prva predlagana rešitev brez TTP, ki nakaže nekatere težave takšnega pristopa, vendar ne uspe rešiti vseh.

V zadnjem delu bomo predstavili rešitev, ki ustreza vsem zahtevam od (a) do (g) iz razdelka 2.3, pogojno pa tudi dodatni zahtevi (h). To ni najhitrejša danes znana rešitev, vendar vseeno sodi med hitrejše. Za razliko od najučinkovitejše rešitve [2] je obvladljivo kompleksna in hkrati služi kot lep prikaz praktične uporabe dokazov brez razkritja znanja (glej §3.5).

Predpostavke in nekatere oznake. V vseh treh predstavljenih shemah bomo predpostavili, da začetna komunikacija med igralci in vzpostavitev igralne skupine nista del sheme. Igralci morajo torej že biti na nek način povezani; poznati morajo npr. IP naslove svojih soigralcev, če gre za igro preko klasičnega računalniškega omrežja. Prav tako morajo že biti dogovorjeni za medsebojni vrstni red, ki bi ga sicer določal sedežni red za mizo. Glede na ta vrstni red bomo igralce označevali s $P_1, P_2, \ldots, P_n$, pri čemer je n število igralcev.

Prav tako bomo (izključno zaradi preglednejšega zapisa in brez izgube za splošnost) privzeli, da se v igri uporablja 52 kart, zato številu kart ne bomo priredili črkovne oznake.

4.1 Sheme s TTP in Fortune-Merritova shema

Fortune-Merrittova shema [9], ki jo bomo predstavili v tem razdelku, načrtno ne izpolnjuje zahteve (a) iz razdelka 2.3; gre torej za rešitev s TTP. To nikakor ni edina objavljena shema, katere avtorji so se odločili za takšen pristop. Zakaj je temu tako?

Prvi, zgodovinski razlog je, da je shemo brez TTP precej težko razviti. Vse do leta 1986 [8], torej sedem let po objavi problema, takšna shema sploh ni bila znana in sheme s TTP so bili naravni raziskovalni koraki proti temu cilju.

A tudi ko so sheme brez TTP že obstajale, so imele strašanske težave z učinkovitostjo. Zanimive so bile teoretično, praktično pa nikakor ne. Leta 1994 so trije "igralci", vsak s svojo delovno postajo Sparc, potrebovali 8 ur [12], da so premešali karte s Crépeaujevo shemo [8]. Nekateri avtorji so si praktičnost zadali kot glavni cilj, bližnjica do njega pa vodi prek uvedbe TTP, in to je drugi razlog za njegovo uporabo.

Najenostavnejša možna shema s centrom zaupanja je tista, v kateri TTP-ju res povsem zaupamo. V igri meša in razdeli karte ter nadzoruje, da igralci res igrajo samo s kartami, ki jim jih je podelil. Takšno shemo poimenujmo *trivialna shema s TTP*.

A trivialna shema z varnostnega stališča ni prav prepričljiva rešitev. Že v uvodu smo zapisali, da je TTP vedno potencialna točka napadov. Oseba z dostopom do centra zaupanja lahko goljufa na dva načina:

- Kot igralec se prijavi v povsem običajno igro, hkrati pa potek igre spremlja še preko TTP. Na ta način dobi vpogled v karte vseh soigralcev, kar mu omogoča, da stavi v natanko pravih trenutkih.

- Goljufivi administrator spremeni program v TTP tako, da program administratorju, kadar je ta prijavljen kot običajen igralec v običajno igro, dodeli dobre karte z nekoliko večjo verjetnostjo.

Oba napada lahko zaznamo s statistično analizo. V prvem primeru izkazuje goljufivi igralec nenaravno dobro "intuicijo," kdaj staviti. V drugem primeru verjetnostna porazdelitev kart ni več enakomerna. A vendar potrebujemo zapise velikega števila iger, preden lahko utemeljeno posumimo na nepravilnost. Napadoma bi se zato radi preventivno izognili.

Prvega napada ne moremo preprečiti, saj ima TTP po svoji naravi popolno informacijo o igri. Možno bi bilo kvečjemu programsko preprečiti dostop do informacij, ki jih hrani TTP, in ga realizirati kot *tamper-proof device* (napravo, fizično zavarovano pred posegi v njeno strojno in programsko opremo), vendar je tak pristop zunaj okvirov klasične kriptografije.

Drugi napad pa za razliko od prvega lahko preprečimo. Sheme s TTP zato poskušajo doseči vsaj to. To jim največkrat uspe tako, da zmanjšajo avtonomijo TTP do te mere, da o načinu mešanja kart ne odloča več sam. Shema, ki jo bomo predstavili v tem razdelku, pristopa nekoliko drugače.

Fortune-Merrittova shema je nastala že zgodaj, leta 1984, vendar poznejši predlogi s TTP niso bistveno izboljšali njenih lastnosti. Zanimiva je, ker celotno varnost osnuje zgolj na enkratnih ščitih v obliki permutacij ter na zgodnji obliki preproste sheme za zaprisego bitov (glej §3.4). V nadaljevanju bomo opisali osnovne operacije te sheme in analizirali njeno varnost in časovno zahtevnost.

4.1.1 Osnovne operacije

Držali se bomo razbitja igre na operacije, kot smo ga opisali v razdelku 2.2. V obliki algoritma bomo podali operaciji mešanja kart in vlečenja posamezne karte, preostale pa bomo zaradi njihove preprostosti opisali le z besedami.

Inicializacija

Vsak od igralcev mora s TTP vzpostaviti varno povezavo. Drugih zahtev po inicializaciji shema nima.

Mešanje kart

S $\mathcal{S}_{52}$ označujemo množico vseh permutacij 52 elementov, s H pa zgoščevalno funkcijo (§3.4.1), denimo SHA-512.

Protokol 4.1.7 MEŠANJE()

1) TTP naključno izbere permutacijo $\pi_{TTP} \in \mathcal{S}_{52}$ (in jo zadrži zase).
2) Vsak igralec P_i, $i = 1, \ldots, n$:
   i) izbere n permutacij $(\pi_{i,1}, \ldots, \pi_{i,n}) \in \mathcal{S}_{52}^n$;
   ii) pošlje $(\pi_{i,1}, \ldots, \pi_{i,n})$ do TTP po varni povezavi;
   iii) izračuna vrednost $H((\pi_{i,1}, \ldots, \pi_{i,n}))$ in jo razpošlje vsem soigralcem.
3) TTP:
   i) za $i = 1, \ldots, n$ izračuna $\pi_i := \pi_{i+1,i}^{-1} \circ \pi_{i+2,i}^{-1} \circ \cdots \circ \pi_{n,i}^{-1} \circ \pi_{1,i}^{-1} \circ \cdots \circ \pi_{i,i}^{-1} \circ \pi_{TTP}^{-1}$;
   ii) vsem igralcem razpošlje $(\pi_1, \ldots, \pi_n)$.

Permutacija, s katero so karte zares premešane, je preprosto π_{TTP}. Povedano drugače, karte igralcev predstavimo z 52-terico $(\pi_{TTP}^{-1}(1), \ldots, \pi_{TTP}^{-1}(52))$. TTP torej karte premeša povsem samostojno – vendar mu pri morebitnem poskusu goljufanja ne pomaga, če kart ne premeša enakomerno naključno. V naslednjem koraku bomo namreč videli, da igralci pri vlečenju karte ne potegnejo nujno prve še nepovlečene karte. Poleg tega TTP svoj π_{TTP} na nek način zapriseže, ko njegov inverz vplete v permutacije $(\pi_1, \ldots, \pi_n)$. To mu preprečuje, da bi π_{TTP} med igro sproti prirejal, kot bi njegovemu goljufivemu administratorju najbolj ustrezalo. Ob koncu igre namreč vsi igralci razkrijejo svoje permutacije. S tem omogočijo soigralcem, da pri sebi odsimulirajo celotno igro in se prepričajo, da je bila poštena.

Permutacije z dvojnim indeksom $(\pi_{i,j})$ pa služijo za medsebojno prikrivanje kart med igralci; kako natančno, bo razvidno iz algoritmičnega opisa.

Vlečenje karte

V naslednjem protokolu opišemo, kako igralec P_i povleče karto.

Protokol 4.1.8 VLEČENJE(i)

Vhod: Indeks i igralca, ki vleče karto.

1) P_i izbere poljubno še nepovlečeno karto $y = \pi_{TTP}(x)$, pri čemer čistopisa x ne pozna, ker je π_{TTP} tajen. Soigralcem in TTP-ju razpošlje y, igralcu P_{i+1} pa $x_i := \pi_i(y)$.
2) Vsak igralec P_j, $j = i+1, \ldots, n, 1, \ldots, i-1$:
   i) prejme x_{j_-} od prejšnjega igralca; pri tem smo z j_- označili vrednost, ki v zaporedju $i+1, \ldots, n, 1, \ldots, i$ nastopi (ciklično) tik pred j.
   ii) izračuna $x_j := \pi_{j,i}(x_{j_-})$ in ga pošlje naslednjemu igralcu.
3) P_i od P_{i-1} prejme x_{i-1}.
4) P_i izračuna $x = \pi_{i,i}(x_{i-1})$.
5) Soigralci si zapomnijo, da je karta s tajnopisom y že povlečena.

Razkrivanje karte

Razkrivanje karte med igro je nadvse preprosto: igralec le zatrdi, da določeno karto ima. Ali je bilo to res, se bodo soigralci lahko prepričali ob koncu igre, ko bodo razkrite vse permutacije.

Odmetavanje karte

Odmetavanje karte med igro je še bolj neproblematično: če igralec želi odvreči karto x, vsem soigralcem sporoči $y = \pi_{TTP}(x)$, ki ga je javno izbral, ko je to karto povlekel.

4.1.2 Varnostna analiza

Razmislimo po vrsti o varnostnih zahtevah iz razdelka 2.3 in se prepričajmo, da Fortune-Merrittova shema zadošča natanko zahtevam (b), (c), (d), (e) in (g). Hkrati se spomnimo, da je bil cilj avtorjev narediti shemo, ki bi delovala varno tudi, če v TTP ne zaupamo povsem. Pokazali bomo, da v tem primeru zahtevi (c) in (d) nista povsem izpolnjeni.

Dokaz. Oglejmo si po vrsti vse zahteve in za pravkar naštete pokažimo, da veljajo.

a) **Odsotnost centra zaupanja.** Tej zahtevi shema očitno ne zadošča.

b) **Unikatnost kart.** Ker karte šifriramo le s permutacijami, ki so na koncu vse po vrsti razkrite, se lahko vsak prepriča, da ni bila v nobenem koraku nobena karta podvojena.

c) **Enakomerno naključna porazdelitev kart.** Pod predpostavko, da je TTP popolnoma zaupanja vreden, so karte res premešane enakomerno naključno, saj π_{TTP} določi TTP samostojno. Če pa predpostavimo, da je TTP lahko bil neopazno kompromitiran tako, da π_{TTP} izbira nekoliko neenakomerno, je zahtevi zadoščeno le delno. Pošteni igralci res lahko med kartami, ki so na voljo, svojo izberejo enakomerno naključno. Goljufivi administrator TTP-ja pa lahko na primer predhodno doseže, da TTP dobrim kartam z nekoliko večjo verjetnostjo priredi nizke zaporedne številke, nato pa tudi sam z nekoliko večjo verjetnostjo izbira nizke y v koraku 1 protokola 4.1.8. Shemi torej ne uspe povsem preprečiti drugega od napadov, ki smo jih omenili v uvodu v razdelek, čeprav ga omili: goljufivi administrator soigralcem na primer ne more vsiliti slabih kart.

   Pomanjkljivost bi bilo mogoče razmeroma enostavno odpraviti: permutacijo π_{TTP} bi morali določiti vsi igralci v sodelovanju s TTP. To najlaže dosežemo s protokoli za zaprisego bitov (glej §3.4). Primer protokola za skupinsko določitev vrednosti je protokol SKUPINSKAVREDNOST v razdelku 4.3.

d) **Odkrivanje goljufij.** Blágo in s statistično analizo zaznavno goljufanje je možno s strani administratorja TTP-ja, kot smo opisali v prejšnji točki.

Goljufije pa, ki jih igralci poskušajo izvesti sami (torej nepravilna uporaba permutacij), ne morejo ostati prikrite. Ker po koncu igre postanejo vse uporabljene permutacije javne, se lahko vsak prepriča, da so jih igralci korektno uporabljali. Igralci se lahko tudi zanesejo, da so permutacije, ki jih soigralci razkrijejo ob koncu igre, res tiste, ki so jih izbrali v koraku 2.1 protokola 4.1.7. V koraku 2.3 vsak igralec namreč javno objavi zgostitev permutacij, ki si jih je izbral. Na ta način jih zapriseže, saj je zaradi lastnosti zgoščevalne funkcije neizvedljivo, da bi po koncu igre v doglednem času našel in soigralcem sporočil drugačno permutacijo z enako zgostitvijo.

e) **Tajnost kart.** Pri dešifriranju karte, ki jo vleče P_i, je ta v vseh korakih zakrita s $\pi_{i,i}$ ali π_{TTP}. Ker sta obe permutaciji tajni, soigralci res ne morejo izvedeti vrednosti karte. Mešanje izvede le TTP, zato tam igralci v nobenem primeru nimajo možnosti pridobiti kako informacijo o kartah.

f) **Tajnost strategije.** To zahtevo shema izrecno prekrši, ko predvidi, da ob koncu igre vsi igralci razkrijejo permutacije, s katerimi so šifrirali karte.

g) **Minimalen vpliv zarot.** Kot smo opisali v točki (e), je vsaka karta, katere vrednost sme videti le P_i, ves čas zakrita s π_{TTP} ali $\pi_{i,i}$. Nobene od teh permutacij ne more ugotoviti še tako velika skupina zarotnikov.

□

Inherentne varnostne omejitve shem s TTP

V uvodu v opis sheme s TTP smo zapisali, da so takšne sheme dovzetne za dve vrsti goljufanja s strani nekoga, ki je nad centrom zaupanja prevzel nadzor. Prva možnost je, da TTP goljufivcu razkriva domnevno tajne informacije; druga je, da mu nameša dobre karte. Odpravljiva je edino druga šibkost. Zgoraj opisana shema jo res odpravi – kot smo videli v analizi, sicer ne povsem, a bi ne bilo pretežko zakrpati preostale varnostne luknje.

V tem pogledu je torej Fortune-Merrittova shema boljša od trivialne sheme s TTP in enako velja tudi za večino ostalih objavljenih shem s TTP. A varnostna analiza razkrije, da zgornja shema od trivialne ni v vseh pogledih le boljša: trivialna shema namreč ohranja tajnost strategije po koncu igre, Fortune-Merrittova shema pa ne. Presenetljivo, za druge v literaturi objavljene sheme s TTP velja enako – bodisi so trivialne bodisi zahtevajo razkritje strategije.

Videti je torej, da nepopolno zaupanje v TTP s seboj prinese velike težave pri ohranjanju tajnosti strategije. Zakaj? Premislimo, kaj se zgodi, če se na vsak način

želimo izogniti razkritju strategije. Če igralci v TTP-jevo mešanje ne zaupajo povsem, morajo biti soudeleženi pri operaciji mešanja ali vlečenja karte. Mešanja se lahko udeležijo v obliki parametra, ki ga posredujejo in na podlagi katerega TTP opravi mešanje, lahko pa kar sami opravijo kakšno od osnovnih operacij, ki sestavljajo mešanje. Kakršenkoli že prispevek je, mora biti narejen bodisi javno, z razkritjem vseh uporabljenih parametrov, bodisi z uporabo dokazov brez razkritja znanja. V nasprotnem primeru namreč igralcu ne moremo zaupati, da je njegov prispevek pošten. Če so vsi parametri, posredovani s strani igralcev, javni, mora TTP nujno uporabiti še kak lasten parameter ali operacijo, sicer bo mešanje kart povsem predvidljivo. Vendar pa smo začeli s predpostavko, da tudi TTP ni nujno povsem pošten, zato bi moral tudi ta z dokazom brez razkritja znanja pokazati, da je pošteno izbral svoj parameter oziroma izvedel svoj del mešanja.

Dokazom brez razkritja znanja se torej skoraj ne moremo izogniti, ti pa tipično niso preprosti, zato so neprivlačni za uporabo v shemah s TTP, kjer sta enostavnost in učinkovitost ključna cilja. Hkrati so dokazi brez razkritja znanja tako močno orodje, da od njihove uporabe ni več daleč niti do zasnove sheme *brez* TTP.

Poglejmo še drugače: Od TTP v operaciji mešanja zahtevamo, da razkrite karte preoblikuje v zakrite in permutirane karte. Če vanj ne zaupamo povsem, mora dokazati, da je to storil pošteno, pri tem pa nikomur od igralcev ne sme razkriti parametrov, s katerimi je premešal karte. To pa že močno spominja na zahteve, ki jih imamo do vsakega posameznega igralca v shemi brez TTP, ki izpolnjuje vse varnostne zahteve. Tudi to nakazuje, da je izgradnja sheme s TTP, pri kateri ne zaupamo v poštenost mešanja TTP, hkrati pa zahtevamo nerazkritje strategij, približno enako zapletena kot izgradnja sheme brez TTP.

Iz tega razloga danes avtorji sheme s TTP večinoma smatrajo za slepo ulico (npr. [12]) v razvoju miselnega pokra. Poigravanje z delnim zaupanjem v center zaupanja, ki tako le slabo upravičuje svoje ime, ni videti smiselno. Po drugi strani so različne izvedbe trivialnih shem s TTP zaradi svoje preprostosti in hitrosti še vedno močno aktualne v praksi, a nezanimive z raziskovalnega stališča.

4.1.3 Časovna zahtevnost

Protokol je tako preprost, da je časovna zahtevnost komaj omembe vredna. Še najdlje predvidoma traja vzpostavitev varne povezave med TTP in posameznimi igralci med inicializacijo. Glede na protokol, ki ga tu uporabimo, zahteva takšna vzpostavitev nekaj potenciranj velikih števil, hevrističnega preverjanja praštevilskosti velikih števil ali podobno. Ne glede na uporabljeni protokol iz vsakdanje rabe varnih povezav vemo, da jih lahko vzpostavimo v nemoteče kratkem času.

Po inicializaciji je najpotratnejša operacija klic zgoščevalne funkcije H v koraku

2.3 protokola 4.1.7, vendar se ta izvede le enkrat med igro in nato $(n-1)$-krat po igri, ko igralci simulirajo račune svojih soigralcev.

Operacije, ki se izvedejo nekoliko večkrat, $O(n^2)$-krat, so le preslikovanje elementa s permutacijo, komponiranje permutacij ter invertiranje permutacij – vse po vrsti izjemno hitre operacije.

Shema je zato učinkovita in s stališča hitrosti nedvomno uporabna v praksi.

4.2 Shamir-Rivest-Adlemanova shema

Problem miselnega pokra si je zamislila in opisala slavna trojica Shamir-Rivest-Adleman leta 1981 v članku *Mental poker* [15]. Po začetnicah njihovih priimkov jo imenujemo "shema SRA". Njihov cilj ni bil zgraditi praktično uporabno shemo, temveč razmisliti, ali je taka shema sploh teoretično izvedljiva. Pokazali so, da brezpogojno varne sheme ni mogoče razviti, če pa se omejimo na računsko varnost, so rešitve možne.

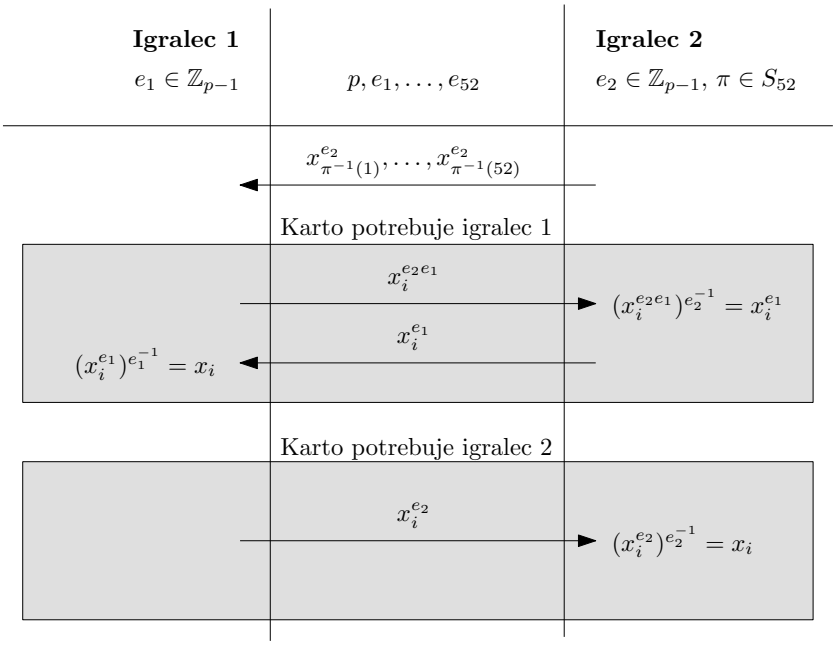
V istem članku so tudi že opisali nadvse preprosto shemo za igranje miselnega pokra za dva igralca, ki je računsko varna. Temelji na problemu diskretnega logaritma. Ponazarja jo slika 4.1, nekoliko natančneje pa njene operacije opišemo v nadaljevanju.

Inicializacija. Igralca se dogovorita za veliko praštevilo p in 52 števil $\{x_1, \ldots, x_{52}\}$, $x_i \in \mathbb{Z}_p^*$, ki bodo predstavljala karte. Vse nadaljnjo računanje bo potekalo v $\mathbb{Z}_p^*$. Vsak od igralcev si izbere še svoje zasebno število, označimo ga z e_1 oz. e_2, $e_1, e_2 \in \mathbb{Z}_{p-1}$.

Mešanje kart. Drugi igralec karte premeša s poljubno permutacijo π, ki jo zadrži zase. Nato vse karte zašifrira tako, da jih potencira s svojim tajnim številom e_2. Rezultat je 52-terica $(x_{\pi^{-1}(1)}^{e_2}, \ldots, x_{\pi^{-1}(52)}^{e_2})$, ki jo pošlje prvemu igralcu.

Vlečenje/izbiranje karte. Prvi igralec dobi od drugega po operaciji mešanja permutirane in zašifrirane karte. Ker o njih ne ve ničesar, jih bo gotovo izbiral pošteno, naključno:

i) **Če karto potrebuje drugi igralec**, prvi igralec naključno izbere nek indeks $i \in \{\pi^{-1}(1), \ldots, \pi^{-1}(52)\} = \{1, \ldots, 52\}$. Iz 52-terice premešanih kart nato "povleče" i-to karto $x_i^{e_2}$ in jo preda drugemu. Ta jo potencira na e_2^{-1} in dobi $x_i^{e_2 e_2^{-1}} = x_i$.



Slika 4.1: Protokol SRA. Vrednosti nad zgornjo črto so vnaprej dogovorjene ali pa jih igralca določita samostojno.

ii) **Če karto potrebuje prvi igralec**, jo izvleče sam, zopet tako, da naključno izbere $i \in \{1, \ldots, 52\}$ in uporabi i-to karto $x_i^{e_2}$. Karto nato zakrije s svojim ključem: izračuna $(x_i^{e_2})^{e_1}$. To vrednost pošlje soigralcu.

Drugi igralec prejme $x_i^{e_2 e_1}$. Ker pozna svoj ključ e_2, lahko konstruira njegov inverz $e_2^{-1} \pmod{p-1}$ in s potenciranjem dobi $(x_i^{e_2 e_1})^{e_2^{-1}} = x_i^{e_1}$. Tega pošlje nazaj prvemu igralcu, ta pa od tu podobno kot prej drugi igralec rekonstruira čistopis x_i.

Razkrivanje karte. Igralec, ki želi uporabiti karto x_i, preprosto objavi njen čistopis. Po koncu igre oba igralca objavita svoja zasebna ključa e_1 in e_2. Tako se lahko oba za nazaj prepričata, da je soigralec uporabljal samo karte, ki jih je pred tem dejansko povlekel.

Odmetavanje karte. Ker oba igralca vesta, katera zašifrirana karta je bila podeljena kateremu igralcu, je ta korak preprost. Igralec le objavi tajnopis karte, ki jo želi zavreči.

4.2.1 Časovna in varnostna analiza

Časovno shema očitno ni problematična. Najzahtevnejša je operacija mešanja, ki zahteva 52 operacij potenciranja, kar na večini današnjih računalnikih celo pri 1024-bitnem p ne traja več kot 3 sekunde. (Oceno trajanja osnovnih računskih operacij podajamo v razdelku 4.3.3.)

Precej manj zadovoljiva je s stališča varnosti. Upoštevati moramo, da so avtorji s shemo želeli le utemeljiti koncept miselnega pokra brez TTP; njihov cilj ni bila idealna shema. Poleg tega v času, ko je bil objavljen članek, Crépeaujeve zahteve iz razdelka 2.3 seveda še niso obstajale, zato ne bi bilo povsem pošteno ocenjevati varnosti sheme SRA v luči teh zahtev. Vendarle omenimo nekatere večje pomanjkljivosti sheme.

Prva, očitna, je omejitev na dva igralca. Posplošitev na n igralcev je enostavna in intuitivna, vendar bi morali pri takšni razširitvi sheme posebno pozornost nameniti možnostim zarote, ki se pri samo dveh igralcih pač ne morejo pojaviti.

Še ena očitna pomanjkljivost je, da morata igralca ob koncu igre razkriti svoje ključe. S tem se razkrije strategija igralcev, za katero je pri pokru še posebej zaželeno, da ostane tajna.

Nadalje, shema med inicializacijo zahteva, da se igralca dogovorita za 52 števil, ki bodo služila kot čistopisi kart. Ni pa natančno opredeljeno, kako naj skleneta dogovor, čeprav je to še kako pomembno. Če lahko v tem procesu namreč eden od igralcev vsili svoje vrednosti $\{x_1, \ldots, x_{52}\}$, lahko doseže, da so ta števila njemu znane potence nekega vnaprej izbranega generatorja α. To pomeni, da bo v nadaljevanju v morebitnih napadih na shemo moral računati le diskretne logaritme z njemu daleč vnaprej znano osnovo α. Če pa osnovo pozna daleč vnaprej, lahko tudi velik del logaritmiranja opravi že vnaprej (glej npr. metodo index calculus v razdelku 3.2.3), zato lahko shemo med samo igro zlomi mnogo hitreje, kot bi jo sicer. Kot možno rešitev tovrstnega problema smo že omenili protokol SkupinskaVrednost iz razdelka 4.3.

Shema dopušča še en način goljufanja. Potenciranje števil x_i namreč ohranja kvadratne ostanke (§3.3). Če na to nismo pozorni in ne poskrbimo že na začetku, da so na primer vsi x_i kvadratni (v tem primeru bodo vsa v igri izmenjana števila kvadratna), lahko tako vsak igralec o vsaki karti dobi vsaj en bit informacije.

4.3 Castellàjeva ZKP-shema

V nadaljevanju si bomo ogledali konkretno shemo za igranje miselnega pokra, ki je razmeroma učinkovita, a kljub temu še obvladljivo kompleksna. Njena varnost sloni na problemu diskretnega logaritma in odločitvenem Diffie-Hellmanovem problemu

(§3.2) ter na dokazih brez razkritja znanja (§3.5). Prvič je bila opisana leta 2005 v doktorski nalogi Španca Jordija Castellàja Roce [4].

4.3.1 Implementacija operacij

Vsaki od operacij, ki smo jih našteli v razdelku 2.2, ustreza v tej shemi en ali več omrežnih protokolov. Ogledali si jih bomo v vrstnem redu, v kakršnem se prvič pojavijo v tipični igri pokra. Da ne izgubimo rdeče niti, jih bomo le opisali, pravilnost njihovega delovanja pa bomo naknadno dokazali v naslednjem razdelku, §4.3.2.

Zaradi nekoliko večje kompleksnosti sheme ter številnih interakcij med igralci bomo podobno kot mestoma že v prejšnjem poglavju algoritme in protokole podali z višjenivojskimi postopkovnimi opisi in ne nujno s psevdokodo, ki bi bila le z manjšimi spremembami prepisljiva v konkreten program. Spremenljivke, ki nastopajo v opisih postopkov, vključno z vhodnimi in izhodnimi, so strogo lokalne. Izjema so tiste, ki so kot parametri sheme dogovorjene vnaprej; tiste, ki jih med postopkom eksplicitno objavimo na oglasni deski (glej spodaj); ter tiste, za katere posebej navedemo, da kot rezultat izvajanja postopka postanejo znane vsaj enemu igralcu. Te spremenljivke so globalno dostopne v vseh postopkih.

Inicializacija

Predpostavimo, da imamo vzpostavljeno "oglasno desko" (*bulletin board*), preko katere igralci izmenjujejo sporočila. Vsa komunikacija je javna in poteka preko te oglasne deske. Če v te namene nimamo vzpostavljene centralizirane infrastrukture ali druge namenske rešitve (npr. *multicast*), lahko pošiljanje sporočila na oglasno desko smatramo za pošiljanje $n - 1$ enakih sporočil vsem soigralcem.

Predpostavimo tudi, da so igralci že dogovorjeni za vrednosti naslednjih parametrov sheme:

- Veliko praštevilo p oblike $p = 2q + 1$, kjer je tudi q praštevilo. Priporočena velikost p je 512 do 1024 bitov. Kjer ni posebej označeno drugače, vse računanje v protokolih poteka po modulu p. Izjema je multiplikativni inverz po modulu $p - 1$, ki ga bomo označevali z inv_{p-1}.
- Nekvadraten ostanek $\alpha \in \mathbb{Z}_p^*$, ki generira ciklično grupo $G \subset \mathbb{Z}_p^*$ moči q.
- Varnostni parameter $s \in \mathbb{N}$ s priporočeno vrednostjo približno 10.

Ko je zgornjima predpostavkama zadoščeno, se igra začne s protokolom INICIALIZACIJA.

Protokol 4.3.9 Inicializacija()

Rezultat: Vsakemu igralcu P_i je znan njegov zasebni ključ a_i, javni ključi $\beta_1, ..., \beta_n$ ter javno število β.

1) Vsak igralec P_i, $i \in \{1, \ldots, n\}$, izbere zasebni ključ a_i in objavi javni ključ $\beta_i = \alpha^{a_i}$.
2) Igralci z 52-kratno uporabo protokola SkupinskaVrednost$(q/2)$ skupaj izberejo naključno množico $X = \{x_1, \ldots, x_{52}\}$, $|X| = 52$, ki bo predstavljala čistopise igralnih kart. Vsak $x_i \in X$ mora biti lih in zanj mora veljati $2 < x_i < q$.
3) Igralci izračunajo $\beta = \alpha^{a_1 \cdots a_n}$ na naslednji način:
   i) Igralec P_1 objavi $\tilde{\beta}_1 = \alpha^{a_1}$.
   ii) Za vsak P_i, $i = 2, \ldots, n$:
      a) P_i izračuna $\tilde{\beta}_i := (\tilde{\beta}_{i-1})^{a_i} = \alpha^{a_1 \cdots a_i}$;
      b) P_i objavi $\tilde{\beta}_i$.
   iii) Igralci si zapomnijo število $\beta := \tilde{\beta}_n$

V zgornjem protokolu smo uporabili protokol SkupinskaVrednost(d). Ta nam omogoča, da n igralcev skupaj enakomerno naključno izbere vrednost iz množice $\mathbb{Z}_d$.

Protokol 4.3.10 SkupinskaVrednost(d)

Vhod: Naravno število d.

Rezultat: Javno znano je enakomerno naključno izbrano naravno število iz $\mathbb{Z}_d$.

1) Vsak igralec P_i, $i \in \{1, \ldots, n\}$:
   i) enakomerno naključno izbere število $r_i \in \mathbb{Z}_d$;
   ii) zapriseže r_i s poljubno vnaprej dogovorjeno shemo za zapriseženo bite ter objavi pripadajoči tajnopis;
   iii) počaka, da vsi soigralci objavijo svoje tajnopise ter nato objavi svojo zapriseženo vrednost.
2) Skupinsko izbrana vrednost je $r := (r_1 + \ldots + r_n) \bmod d$. Izračuna jo lahko vsak igralec zase, saj so vrednosti $r_1, \ldots, r_n$ na tem mestu javne.

Mešanje kart

Da dosežemo čim večjo odpornost sheme proti koalicijam zarotnikov, tudi takšnim velikosti $n - 1$, karte premeša vsak od igralcev. Pri tem jih hkrati tudi zašifrira, da drugi igralci ne morejo ugotoviti uporabljene permutacije.

Označimo nepremešane karte z množico $C_0 = \{c_{0,1}, \ldots, c_{0,52}\}$. Za vsak $i = 1, \ldots, 52$ je $c_{0,i}$ par $(d_{0,i}, \alpha_{0,i})$, pri čemer je $d_{0,i} = \alpha^{x_i}$ in $\alpha_{0,i} = \beta$. Prvi element

para bo torej skrival vrednost karte x_i. Drugega bomo v nadaljevanju uporabili za preverjanje poštenosti igralcev in kot pomoč pri odšifriranju prvega.

Protokol 4.3.11 MEŠANJE(C_0)

Vhod: Množica nepremešanih kart
$C_0 = \{(d_{0,1}, \alpha_{0,1}), \ldots, (d_{0,52}, \alpha_{0,52})\} = \{(\alpha^{x_1}, \beta), \ldots, (\alpha^{x_{52}}, \beta)\}$.

Rezultat: Javno znana je množica premešanih in šifriranih kart C_n.

1) Vsak igralec P_i, $i = 1, \ldots, n$:
   i) Izračuna (C_i, R_i, π_i) :=MEŠANJEKORAK(C_{i-1});
   objavi C_i, preostanek rezultata pa si zapomni.
   ii) S protokolom MEŠANJEDOKAZ($i, C_{i-1}, C_i, \pi_i, R_i$) brez razkritja znanja dokaže poštenost mešanja.

Kot zadnji korak zgornjega protokola n-ti igralec izračuna C_n, kar bomo uporabili kot dokončni premešani kupček kart. Oglejmo si še posamezni korak mešanja, MEŠANJEKORAK, na katerega se sklicuje zgornji protokol.

Algoritem 4.3.12 MEŠANJEKORAK(C)

Vhod: Množica kart $C = \{(d_1, \alpha_1), \ldots, (d_{52}, \alpha_{52})\}$.

Rezultat: Izhodne vrednosti: množica delno premešanih kart C^*; množica R in permutacija π, ki definirata ta korak mešanja.

1) Izberi naključno množico $R = \{r_1, \ldots, r_{52}\}$, kjer je vsak r_i lih in v mejah $2 < r_i < q$.
2) Izberi naključno permutacijo π nad 52 elementi.
3) Za $j = 1, \ldots, 52$ izračunaj $c'_j = (d_j^{r_j}, \alpha_j^{r_j})$.
4) Uporabi π nad pravkar izračunanimi c'_j; rezultat je $C^* := \{c'_{\pi^{-1}(1)}, \ldots, c'_{\pi^{-1}(52)}\}$.
5) Vrni (C^*, R, π).

Kot zadnji, a najpomembnejši gradnik mešanja kart opišimo protokol, s katerim lahko igralec P_i dokaže, da je preslikavo $C_{i-1} \mapsto C_i$ res naredil z uporabo postopka MEŠANJEKORAK, ne da bi pri tem razkril uporabljena R in π. Dokaz je verjetnosten, stopnjo zanesljivosti (ki seveda pride za ceno računske zahtevnosti) pa določa varnostni parameter s.

Dokaz izvede igralec P_i s pomočjo protokola MEŠANJEDOKAZ, v sklopu tega pa morajo njegovi soigralci pri sebi simulirati njegovo mešanje, da se prepričajo o pravilnosti le-tega. Simulacijo jim omogoča funkcija TESTMEŠANJEKORAK(C, R, π, C^*).

Algoritem 4.3.13 TestMešanjeKorak(C, R, π, C^*)

Vhod: Množica kart $C = \{(d_1, \alpha_1), \ldots, (d_{52}, \alpha_{52})\}$; množica R in permutacija π, ki definirata korak mešanja; množica delno premešanih kart C^*.

Rezultat: Izhodna vrednost: `true` če in samo če bi klic funkcije MešanjeKorak(C), ki bi v prvih dveh korakih izbral podani vrednosti R in π, preslikal C v C^*.

1) Za $j = 1, \ldots, 52$ izračunaj $c'_j = (d_j^{r_j}, \alpha_j^{r_j})$.
2) Uporabi π nad pravkar izračunanimi c'_j; rezultat je $C^{**} := \{c'_{\pi^{-1}(1)}, \ldots, c'_{\pi^{-1}(52)}\}$.
3) Vrni `true`, če velja $C^{**} = C^*$, `false` sicer.

Sledi opis že napovedanega dokaza brez razkritja znanja. Denimo, da pravilnost svojega mešanja želi dokazati igralec P_i.

Protokol 4.3.14 MešanjeDokaz(i, C, C^*, π, R)

Vhod: Indeks igralca i, ki dokazuje pravilnost mešanja; množici kart C in C^*, kjer je C^* domnevno rezultat koraka mešanja kart iz C; množica R in permutacija π, ki sta definirali ta korak mešanja in sta znani samo igralcu P_i.

Rezultat: Če je P_i pri mešanju goljufal in množica C^* ni rezultat funkcijskega klica MešanjeKorak(C), soigralci to z veliko verjetnostjo ugotovijo. Če pa je mešal pošteno, soigralci sprejmejo dokaz o pravilnosti mešanja.

1) Za $k = 1, \ldots, s$ igralec P_i izračuna (C_k^*, R_k, π_k) :=MešanjeKorak(C^*) in objavi množico C_k^*.
2) Igralci $P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n$ uporabijo SkupinskaVrednost(2^s) in s tem določijo s-bitni izziv $(u_1, \ldots, u_s)$
3) Za $k = 1, \ldots, s$:
   i) Če je bit $u_k = 1$:
      a) P_i objavi R_k ter π_k.
      b) Soigralci sprejmejo dokaz, če TestMešanjeKorak(C^*, R_k, π_k, C_k^*) vrne `true`.
   ii) Če je bit $u_k = 0$:
      a) P_i izračuna $\pi'_k := \pi_k \circ \pi$ in $R'_k := \{r'_{k,1}, \ldots, r'_{k,52}\}$, kjer je $r'_{k,j} := r_j \cdot r_{k,\pi(j)}$ za vsak j.
      b) P_i objavi R'_k in π'_k.
      c) Soigralci sprejmejo dokaz, če TestMešanjeKorak(C, R'_k, π'_k, C_k^*) vrne `true`.

Vlečenje karte

Denimo, da želi igralec P_i povleči novo karto. Lahko mu popolnoma zaupamo, da si bo karto izbral naključno – izbrati mora enega od elementov iz C_n, in o nobenem ne ve ničesar. Prav tako je ves čas javno znano, katere zašifrirane karte so že bile izvlečene, zato tudi tu goljufanje ni mogoče. Naš protokol zato le poskrbi, da bo P_i izvlečeno karto lahko dešifriral.

Zaradi enostavnejšega zapisa naslednjega protokola brez škode za splošnost privzamemo, da karto vleče igralec P_n. Nikjer v opisu protokola, kjer P_n nastopa kot odlikovani igralec, to ni povezano s tem, da je po vrstnem redu zadnji od igralcev.

Spomnimo se še, da si vsak igralec v prvem koraku protokola MEŠANJE zapomni množico R, ki jo vrne njegov klic funkcije MEŠANJEKORAK. Med postopkom mešanja je za vsak i i-ta karta iz kupčka šifriranih kart $c_{n,i}$ šifrirana v n korakih – na vsakem koraku z enim od števil iz ene od množic $R_1, \ldots, R_n$. Posamezna števila, s katerimi je bila šifrirana karta $c_{n,i}$, na tem mestu niso pomembna, zanimal nas bo le njihov produkt; označimo ga z $r_{\Pi,i}$.

Protokol 4.3.15 VLEČENJE(i)

Vhod: Indeks i šifrirane karte $c_{n,i} = (d_{n,i}, \alpha_{n,i}) \in C_n$, ki jo želi povleči P_n.
Rezultat: Igralec P_n pozna čistopis, skrit v tajnopisu $c_{n,i}$.

1) Igralec P_n objavi $e_{0,i} := \alpha_{n,i} = \beta^{r_{\Pi,i}} = \alpha^{a_1 \cdots a_n \cdot r_{\Pi,i}}$.
2) Soigralci z $e_{0,i}$ odstranijo svoje privatne ključe $(a_1, \ldots, a_{n-1})$ na sledeči način: Za $k = 1, \ldots, (n-1)$:
   i) Igralec P_k izračuna in objavi $e_{k,i} := (e_{k-1,i})^{\mathrm{inv}_{p-1}(a_k)}$.
   ii) Igralec P_k z vsakim od soigralcev uporabi protokol CHAUMPEDERSEN($p, \alpha, \beta_i, e_{k,i}, e_{k-1,i}, a_k$) in tako brez razkritja a_k dokaže, da je prejšnji korak izvedel korektno.
3) Igralec P_n odstrani še svoj privatni ključ: izračuna $e_{n,i} := (e_{n-1,i})^{\mathrm{inv}_{p-1}(a_n)} = \alpha^{r_{\Pi,i}}$ in si ga zapomni.
4) Z vstavljanjem vseh 52 možnih vrednosti za x igralec P_n najde čistopis karte, ki jo je povlekel: to je tisti x, ki zadosti enakosti $(e_{n,i})^x = d_{n,i}$. Opomnimo, da enako kot vse izračune tudi to preverjanje enakosti opravljamo po modulu p.

Razkrivanje karte igralcem

Karte $c_{n,i}$ ni težko pokazati soigralcem – njen čistopis preprosto objavimo na oglasni deski. Vendar pa jih moramo prepričati, da razkrivamo karto, ki smo jo nekoč res povlekli. To naredimo s sledečim kratkim protokolom. Kot v protokolu VLEČENJE tudi tu predpostavimo, da karto razkriva igralec P_n.

Protokol 4.3.16 RAZKRIVANJE(i)

Vhod: Indeks šifrirane karte $c_{n,i} \in C_n$, ki jo P_n želi razkriti.

1) Igralec P_n objavi x ter $e_{n,i} = (e_{n-1,i})^{\mathrm{inv}_{p-1}(a_n)} = \alpha^{r_{\Pi,i}}$.
2) Igralec P_n z vsakim od soigralcev uporabi protokol CHAUMPEDERSEN($p, e_{n,i}, e_{n-1,i}, \alpha, \beta_n, a_n$) in tako brez razkritja a_n dokaže, da je prejšnji korak izvedel korektno.
3) Ostali igralci vsak zase preverijo, da x res zadosti enakosti $(e_{n,i})^x = d_{n,i}$.

Ker je P_n lahko $e_{n,i}$ dobil le tako, da so mu ga nekoč pri vlečenju karte $c_{n,i}$ pomagali izračunati soigralci, s tem prepriča ostale igralce o svojem lastništvu karte.

Odmetavanje karte

Pri uporabljenem pristopu je ta operacija preprosta in ne potrebuje kriptografije. Ker igralci karte vlečejo z javnim izbiranjem elementov v množici šifriranih kart C_n, se za vsako šifrirano karto ve, kdo je njen lastnik. Ko igralec želi izločiti določeno karto, ne da bi razkril njen čistopis, mora zato le objaviti njen indeks v C_n.

4.3.2 Varnostna analiza

V tem razdelku najprej pokažemo, da opisani postopki zares naredijo, kar smo trdili. Nato preverimo, da Castellàjeva ZKP-shema zadosti vsem zahtevam iz razdelka 2.3. Pri tem privzamemo, da so problemi, opisani v 3. poglavju (problem faktorizacije, DLP, DDH, problem kvadratnih ostankov), zares neobvladljivi.

Trditev 4.3.1. *Postopek 4.3.9,* INICIALIZACIJA(), *ne izda nobene zasebne informacije.*

Dokaz. Edina občutljiva informacija v postopku INICIALIZACIJA() so zasebni ključi a_i, $i \in \{1, \ldots, n\}$. Če želi igralec P_j ugotoviti vrednost ključa a_i, $i \neq j$, ga lahko izračuna le kot $a_i \log_{\tilde{\beta}_{i-1}} \tilde{\beta}_i$. To pa je problem diskretnega logaritma in za dovolj veliko grupo ni obvladljiv. Element α je generator grupe G, zato se celoten DLP odvija v tej grupi. Na začetku smo zahtevali $|G| = |\mathbb{Z}_p^*|/2 = q$, število q pa posredno definirali kot veliko praštevilo s 512 do 1024 biti; grupa je torej dovolj velika. □

Trditev 4.3.2. *Vrednost, ki je rezultat protokola* SKUPINSKAVREDNOST(d)*, je enakomerno naključno izbran element množice $\mathbb{Z}_d$, čim je vsaj eden od igralcev pošten.*

Dokaz. Denimo, da je igralec P_i pošten. Ker koraka 1.3 ne bo izvršil, dokler vsak soigralec P_j, $j = 1, \ldots, n$, ne zapriseže svoje vrednosti r_j, bodo vse vrednosti r_j, $j \neq i$, izbrane neodvisno od r_i.

Ker igralec P_i igra pošteno, je r_i izbran enakomerno naključno iz množice $\mathbb{Z}_d$. Sledi, da je tudi vsota

$$r_1 + \cdots + r_n = r_i + (r_1 + \cdots + r_{i-1} + r_{i+1} + \cdots + r_n)$$

enakomerno naključno izbran element množice $\mathbb{Z}_n$. □

Trditev 4.3.3. *Protokol* MešanjeDokaz(i, C, C^*, π, R) *je dokaz brez razkritja znanja. Igralcu P_i omogoča dokazati, da je C^* rezultat uporabe funkcije* MešanjeKorak(C)*, ne da bi razkril uporabljena π in R. Verjetnost uspešnega goljufanja s strani dokazovalca je 2^{-s}, kjer je s varnostni parameter.*

Dokaz. Oglejmo si k-to od s iteracij, kjer je s varnostni parameter sheme. Pokažimo najprej, da gre za interaktiven dokaz. Če MešanjeKorak preslika $C \overset{R,\pi}{\longmapsto} C^* \overset{R_k,\pi_k}{\longmapsto} C_k^*$, potem zaradi svojih lastnosti (uporabljena sta le permutacija in potenciranje) in načina, kako smo konstruirali R'_k, π'_k, preslika tudi $C \overset{R'_k,\pi'_k}{\longmapsto} C_k^*$ – to pa je prav preslikava, ki jo preverjajo soigralci v koraku 3.ii.c.

Če torej P_i pošteno uporabi MešanjeKorak in preslika $C \mapsto C^*$ ter $C^* \mapsto C_k^*$, bodo soigralci dokaz res sprejeli ne glede na vrednost izziva u_k: če je $u_k = 1$, bo korak 3.i.b očitno uspešen. Če pa je $u_k = 0$, bo po prejšnjem odstavku uspel test iz koraka 3.ii.c. Interaktivni dokaz je zato poln.

Če pa je P_i konstruiral C^* drugače kot z uporabo MešanjeKorak(C), pokažimo, da se lahko pripravi le na enega od obeh možnih izzivov.

Igralec P_i lahko nepravilno izračunan C^* poskusi vsiliti v igro na dva načina. Prva možnost je, da izračuna C_k^* na pravilen način, nato pa vmesni rezultat C^* zavrže in soigralcem podtakne lažnega. V tem primeru bo test iz koraka 3.a.ii spodletel in bo dokaz zavrnjen.

Drugi možni način goljufanja je, da igralec kar takoj nepravilno generira C^*, nato pa iz njega pravilno izpelje C_k^*. Vendar je v tem primeru C_k^* nepravilno generiran glede na C, zato bo goljufija opažena v koraku 3.ii.c.

Po trditvi 4.3.2 je izziv določen enakomerno naključno, čim vsaj en igralec igra pošteno. Goljufivemu igralcu P_i zato preostane le ugibanje vrednosti izziva. Verjetnost, da mu bo uspelo in bo dokaz nezasluženo uspel, je 1/2, v vseh s iteracijah pa le 2^{-s}. Interaktivni dokaz je zato res uglašen.

Prepričajmo se še o nerazkritju znanja. Edina podatka, ki sta vpletena v protokol in v katere P_i-jevi soigralci ne smejo dobiti vpogleda, sta R in π.

Če je izziv $u_k = 1$, množici R in π niti posredno ne nastopata nikjer v izmenjanih podatkih, zato ni nevarnosti razkritja.

Če je izziv $u_k = 0$, pa soigralci dobijo $\pi'_k = \pi_k \circ \pi$, iz katerega bi potencialno lahko razbrali vsaj kakšno informacijo o π. Vendar v tem primeru π_k ostaja tajen

in služi kot enkraten ščit za π, ki tako ostane povsem na varnem. Podobno velja za elemente iz R, ki jih enkratno ščitijo elementi iz nikoli razkritega R_k. □

Trditev 4.3.4. *Za šifrirano karto $c_{n,i} \in C_n$ nihče ne more ugotoviti njenega čistopisa x, ne da bi mu pri tem pomagali vsi soigralci. Pri tem predpostavimo, da je odločitveni Diffie-Hellmanov problem neobvladljiv.*

Dokaz. Ker je možnih čistopisov le 52, se moramo prepričati, da čistopisa ni mogoče ugotoviti niti s poskušanjem. Definirajmo spet enako kot v razdelkih o vlečenju in razkrivanju karte število $r_{\Pi,i}$ kot produkt vseh števil iz množic $R_1, \ldots, R_n$, ki so bila uporabljena za šifriranje karte $c_{n,i}$. Spomnimo se, da ima šifrirana karta obliko

$$c_{n,i} = (d_{n,i}, \alpha_{n,i}) = (\alpha^{x_i \cdot r_{\Pi,i}}, \beta^{r_{\Pi,i}}).$$

Če želimo za nek fiksen x_j preveriti, ali ustreza čistopisu, preverjamo

$$\log_\alpha d_{1,j} \stackrel{?}{=} x_i \cdot \log_\beta \alpha_{1,i}.$$

Če bi za ta problem imeli učinkovit algoritem A, bi lahko učinkovito rešili tudi odločitveni Diffie-Hellmanov problem $\log_{g_1} y_1 \stackrel{?}{=} \log_{g_2} y_2$, in sicer tako, da bi v A vstavili poljuben x ter preostale parametre nastavili na

$$\alpha := g_1, \;\; d_{n,i} := y_1^x \;\; \beta := g_2 \;\; \alpha_{n,i} := y_2.$$

Po predpostavki je zato problem dešifriranja karte neobvladljiv, enako kot odločitveni Diffie-Hellmannov problem. □

Trditev 4.3.5. *Po i-tem klicu funkcije* MešanjeKorak *iz protokola* Mešanje *nihče razen P_i ni sposoben ugotoviti, ali sta $c_{i-1,j} \in C_{i-1}$ in $c_{i,k} \in C_i$, $j, k \in \{1, \ldots, 52\}$, ena in ista karta. (Z drugimi besedami, karte med mešanjem niso sledljive.)*

Dokaz. Če $c_{i-1,j}$ in $c_{i,k}$ skrivata isti čistopis, je $k = \pi_{i-1}(j)$ in je $c_{i,k}$ oblike

$$(d_{i,k}, \alpha_{i,k}) = ((d_{i-1,j})^{r_{i-1,j}}, (\alpha_{i-1,j})^{r_{i-1,j}}).$$

Da bi ugotovili $c_{i-1,j} \stackrel{?}{=} c_{i,k}$, moramo torej preveriti $\log_{d_{i-1,j}} d_{i,k} \stackrel{?}{=} \log_{\alpha_{i-1,j}} \alpha_{i,k}$. To pa je primer odločitvenega Diffie-Hellmanovega problema in zato po predpostavki iz začetka razdelka ni učinkovito rešljiv.

Opozorimo še na označevanje kart. Kot smo že omenili v razdelku 3.3.2 doslej predlagane sheme, ki temeljijo na potenciranju, pogosto razkrivajo en bit informacije o vsaki karti. Pri potenciranju se namreč ohranjajo kvadratni ostanki. V opisani shemi se temu že v začetku izognemo tako, da izberemo generator α, ki je nekvadratni ostanek, nato pa eksplicitno zahtevamo, da so vsi eksponenti, uporabljeni v igri, lihi. Da se njegovi soigralci držijo tega pravila, lahko preveri vsak igralec, saj je vsa komunikacija javna. □

S pomočjo zgornjih trditev se prepričajmo, da opisana shema zadosti vsem zahtevam, naštetim v razdelku 2.3. Dokazovali bomo le računsko varnost, pri tem pa predpostavili, da so parametri sheme – praštevilo p ter varnostni parameter s – dovolj veliki. Priporočene vrednosti smo navedli v začetku razdelka 4.3.1

Odsotnost centra zaupanja. Vse operacije v vseh opisanih protokolih so popolnoma simetrične glede na vse udeležence, zato ni noben od igralcev privilegiran.

Unikatnost kart. Do podvajanja kart v kupčku lahko pride edino med mešanjem ali razkrivanjem, saj se le tam spreminjajo opisi kart. Vendar pa je malo verjetno, da bo goljufanje ostalo neopaženo: pravilnost vsakega posameznega koraka mešanja mešalec dokazuje s protokolom MešanjeDokaz in po trditvi 4.3.3 mu bo zato goljufija uspela z verjetnostjo samo $(\frac{1}{2})^s$, kjer je s varnostni parameter. Pri razkrivanju pa je ohranjanje čistopisov zagotovljeno s Chaum-Pedersenovim ZKP.

Edini preostali način za podvajanje kart je, da bi iz kupčka sicer unikatnih kart večkrat potegnili eno in isto karto. To ni mogoče, saj vsak igralec za karto, ki jo želi povleči, javno objavi njen indeks v kupčku premešanih kart. Ker se kupček ne spreminja, lahko vsak preveri, da je bila vsaka karta povlečena največ enkrat.

Enakomerno naključna razporeditev kart. Permutacija, s katero premešamo karte, je kompozitum permutacij posameznih igralcev. Že en sam pošten igralec, katerega permutacija je resnično naključna in skrita pred soigralci, zadošča, da bodo karte premešane enakomerno naključno.

Odkrivanje goljufij. Po trditvi 4.3.3 bo goljufanje med mešanjem kart odkrito z verjetnostjo $1 - (1/2)^s$. Goljufanje pri inicializaciji je po trditvi 4.3.1 enako težko kot problem diskretnega logaritma. Tudi goljufanje pri vlečenju ali razkrivanju karte je zaradi Chaum-Pedersenovega dokaza vsaj tako težko kot odločitveni Diffie-Hellmannov problem.

Tajnost kart. Če vsaj nek igralec P_i igra pošteno in svojo permutacijo π_i zadrži zase, ne more nihče slediti, v katero karto iz $c_{n,i} \in C_n$ je bila preslikana neka nešifrirana karta iz C_0. Igralec P_j, $j \neq i$, jo bo v primeru, da se poveže z igralci $P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n$ v zarotniško skupino, lahko sledil kvečjemu v množicah $C_0, C_1, \ldots, C_{i-1}$, s preslikavo iz C_{i-1} v C_i pa bo po trditvi 4.3.5 za karto izgubil sled.

Da pa iz $c_{n,i}$ ni mogoče neposredno razbrati čistopisa, ne da bi poznali vse eksponente $r_{1,i}, \ldots r_{n,i}$ ali vse osebne ključe $a_1, \ldots, a_n$, smo pokazali v dokazu trditve 4.3.4.

Tajnost strategije. Igralcem ob koncu igre ni treba razkriti svojih privatnih ključev ali kakršnihkoli drugih informacij, da bi dokazali svojo poštenost; ta se preverja že sproti. Potek igre ter karte, ki jih igralci obdržijo v roki, tako ostanejo tajne.

Minimalen vpliv zarot. Vsi izpeljani dokazi o varnosti kot predpostavko zahtevajo kvečjemu, da zlonamerni igralec za vsaj en u ne pozna eksponentov $r_{u,i}$, osebnega ključa a_u oziroma permutacije π_u. En sam pošten in nekompromitiran igralec P_u torej onesposobi zaroto preostalih $n - 1$ igralcev.

4.3.3 Časovna zahtevnost

Za vsakega od protokolov in funkcij v shemi bomo prešteli operacije potenciranja in pošiljanja sporočil po omrežju. Ker so sporočila kratka, bomo privzeli, da pošiljanje oziroma sprejemanje traja konstantno mnogo časa, N sekund. Trajanje enega modularnega potenciranja bomo označili z E. Trajanje ostalih operacij (vključno z množenjem velikih števil) bomo zanemarili. Čas, potreben za izvedbo posameznih protokolov, je nadalje odvisen od števila igralcev n, varnostnega parametra s in velikosti praštevilske grupe $\mathbb{Z}_p^*$; število bitov v zapisu p bomo označili s $|p|$.

Za objavo na oglasni deski bomo privzeli, da traja N sekund in ne $(n-1)N$. Kadar je potrebno uporabiti protokol za zaprisego bitov, bomo privzeli Goldwasser-Micalijev protokol (§3.4.2). Kadar mora določeno operacijo izvesti vseh n igralcev, pa lahko to storijo vzporedno, štejemo, bomo trajanje operacije v časovno bilanco šteli le enkrat. Če določen protokol nima deterministične časovne zahtevnosti, bomo zapisali pričakovano zahtevnost.

ChaumPedersen	$6E + 3N$
GoldwasserMicali	$2E + N$
Inicializacija	$E + N + 52(2nE + nN) + n(E + N)$ $= (105n + 1)E + (53n + 1)N$
SkupinskaVrednost	$2nE + nN$
Mešanje	$n(104E + (260s + 2n)E + (2s + n)N)$ $= (104n + 260sn + 2n^2)E + (2sn + n^2)N$
MešanjeKorak	$104E$
TestMešanjeKorak	$104E$
MešanjeDokaz	$s(104E + N) + (2nE + nN) +$ $+ s(\frac{1}{2}(104E + N) + \frac{1}{2}(104E + N + 104E))$ $= (260s + 2n)E + (2s + n)N$
Vlečenje	$N + (n - 1)(E + N + (6E + 2N)) + E + \frac{1}{2} \cdot 52E$ $= (7n + 20)E + (3n - 2)N$
Razkrivanje	$N + (n - 1)(6E + 2N) + E$ $= (6n - 5)E + (2n - 2)N$

Vidimo, da je kot pri veliki večini do zdaj predlaganih shem za miselni poker tudi pri naši mešanje kart časovno daleč najbolj problematično.

Shema žal ni bila nikoli implementirana, niti s strani avtorja niti v okviru te diplomske naloge. Da dobimo občutek za čas, ki bi ga operacija mešanja potrebovala v realni implementaciji, pa lahko eksperimentalno ocenimo trajanje vsake od osnovnih operacij ter s pomočjo zgornje tabele sklepamo na trajanje izvajanja večjih segmentov igre, kot je na primer mešanje.

Povprečno trajanje zahtevnejših operacij nad velikimi števili povzemimo po [4], kjer so bili časi izmerjeni na zmogljivem osebnem računalniku izpred nekaj let (ThinkPad T41, procesor Centrino 1.5GHz).

velikost števil ($\|p\|$)	256	512	768	1024
potenciranje (E)	1.27ms	7.88ms	24.51ms	55.64ms
množenje	12µs	27µs	50µs	87µs

Trajanje operacije množenja navajamo le zato, da upravičimo zanemarjanje operacij množenja v časovni bilanci. Privzamemo še, da traja izmenjava kratkega omrežnega sporočila 20ms. (Za lokalno omrežje je ocena previsoka, za današnji internet pa optimistična, a realna. To ni zelo pomembno, ker v naši shemi večino časa vzamejo računske operacije.)

Združimo zdaj rezultate obeh tabel in si oglejmo, koliko časa shema potrebuje, da premeša karte. Predstavljamo nekaj zanimivih naborov vrednosti parametrov:

$|p| = 256$**,** $s = 5$**,** $n = 2$**: 4.1 sekunde.** Najhitrejša še pogojno smiselna izvedba: le dva igralca in zmerno prepričljiv nadzor nad mešanjem ($s = 5$). Tudi p je majhen. Trajanje mešanja je zaznavno, a nikakor moteče.

$|p| = 256$**,** $s = 10$**,** $n = 5$**: 19.7 sekund.** Tudi če glede na prejšnji primer število igralcev razumno povečamo ter dvignemo varnostni parameter s na nesporno uporabno vrednost 10 (možnost neopaženega goljufanja med mešanjem je potem manj kot 1:1000), se bo mešanje predvidoma izvedlo v manj kot 20 sekundah. Vendar pa 256-biten p ne nudi prav posebne varnosti. Resda je dovolj velik, da poštenost same igre ni ogrožena – napad na DLP tudi pri tako skromnem p traja predolgo, da bi ga bilo mogoče neopaženo izvesti med dvema potezama v igri. Po drugi strani lahko napadalec z zmernimi računskimi viri (npr. gručo nekaj deset osebnih računalnikov) problem diskretnega logaritma zlomi v nekaj dneh do nekaj tednih. Le toliko časa po igri torej ostanejo strategije igralcev skrite pred radovednimi.

$|p| = 512$**,** $s = 10$**,** $n = 5$**: 109.4 sekund.** V primerjavi s prejšnjim primerom smo povečali le velikost grupe $\mathbb{Z}_p^*$. Parameter $p = 512$ je še vedno kompromis med varnostjo in učinkovitostjo: dovolj zagrizen napadalec lahko shemo razbije v nekaj tednih do nekaj mesecih, velike korporacije ali vojska pa najbrž v nekaj dneh. Napad ni zelo verjeten, saj bi uspešen napad razkril le karte, ki igralcem po koncu igre ostanejo v roki; ni pa povsem izključen. Po drugi strani si ne moremo privoščiti nadaljnjega povečevanja p, saj sta dve minuti le še zelo pogojno sprejemljiva čakalna doba pred začetkom vsake partije.

$|p| = 1024$**,** $s = 20$**,** $n = 10$**: 50 minut.** To so parametri, za katere si potihem želimo, da bi shema še vedno delovala učinkovito. Pri tako velikem p se tudi najbogatejšim napadalcem ne splača nameniti enega ali več računskih let celotnega računskega centra za napad. Varnostni parameter s skoraj povsem preprečuje neopazno goljufanje med mešanjem – verjetnost za kaj takega je tu manj kot ena proti milijon.

Glede na izračunano časovno zahtevnost bo igra s takšnimi parametri primerna kvečjemu za zagrizene varnostne zanesenjake: ponoči lahko pustijo, da računalnik inicializira kakih deset partij, ki jih nato čez dan odigrajo ...

V primerjavi s shemami, objavljenimi v starejših člankih, je opisana časovna zahtevnost opazen napredek, vendar hkrati z njo očitno še ne moremo biti dokončno zadovoljni.

Poglavje 5

Miselni poker v vsakdanu

V prejšnjem poglavju smo si ogledali Castellàjevo ZKP-shemo, eno najučinkovitejših danes znanih shem za miselni poker brez TTP. Njena učinkovitost je v praksi že skoraj sprejemljiva; morda celo ne bi bilo neupravičeno pričakovati, da bodo uporabniki pripravljeni čakati dvajset ali trideset sekund pred vsako partijo, če bodo s tem privarčevali provizijo, ki jo sicer plačujejo poker hišam.

Kljub temu ni videti, da bi v bližnji prihodnosti shemam brez TTP uspel preboj. Kriptografi se zavedajo, da karakteristike obstoječih rešitev še niso dovolj dobre in se trudijo najti še primernejše sheme.

Pomanjkljivost shem brez TTP, ki je bila v zadnjem času deležna precej pozornosti, je računska zahtevnost. Leta 2005, ko je Castellà Roca objavil razmeroma hitro shemo iz prejšnjega poglavja, je podobna pohitritev uspela tudi Francozu Golleju. Njegova shema [12] mešanje kart (ki je tipično časovno problematično) pohitri tako, da potrebne izračune opravi za vsako karto posebej. To omogoča, da račune opravljamo sproti, ko je treba posamezno karto podeliti igralcu. Vendar rezultati žal niso ravno revolucionarni – izračunov, ki jih je Golle uspel časovno tako razbiti, je enostavno preveč. V svojem članku je ocenil število potenciranj, potrebnih za podeljevanje vsake karte; čeprav zahtevnosti ne moremo neposredno primerjati z "našo" shemo, ker je slednja manj občutljiva na število igralcev, lahko zaključimo, da Gollejeva shema za eno karto potrebuje najmanj četrtino časa, ki ga Castellàjeva potrebuje za premešanje vseh kart.

Razpravljanje o tem, katera od shem je hitrejša, najbrž niti ni smiselno. Castellà Roca je namreč že dve leti prej, 2003, objavil shemo [2], za katero zatrjuje, da je še učinkovitejša. Številsko to težko preverimo: avtor ni objavil analize ali meritev časovne zahtevnosti, ročno štetje operacij pa ni preprosto. Šibka točka sheme je namreč precejšna kompleksnost; protokola inicializacija in vlečenja karte na primer zahtevata vsak po 15 korakov. Ne glede na to je časovna zahtevnost najbrž res

razmeroma nizka: v [4] avtor zatrdi, da je shema patentirana, prodana podjetju *Scytl Online World Security, S.A.* in tudi implementirana. Implementorji so shemo menda opisali kot "časovno praktično uporabno". Sodeč po literaturi je to najhitrejša danes znana shema za miselni poker brez TTP.

Razlog za nerazširjenost shem brez TTP pa ne tiči samo v časovni zahtevnosti. Ena od njihovih pomanjkljivosti je tudi precej manjša fleksibilnost v primerjavi s centraliziranimi shemami. Objavljen je bil na primer članek, ki predstavi shemo brez TTP, odporno na predčasen odhod igralcev iz igre [3], pa članek o shemi brez TTP, ki je odporna na začasno odsotnost igralcev sredi igre [13] – oboje sta lastnosti, ki jih v trivialni shemi s TTP pridobimo brez težav. Svoje naredi najbrž tudi dejstvo, da so sprejemljivo hitre sheme brez TTP razmeroma nova iznajdba.

Vendarle noben od naštetih razlogov ni videti usoden. Najverjetneje se sheme brez TTP ne uveljavijo tudi zaradi inercije. Sistem TTP-jev, torej igralnih strežnikov za poker, je dobro razvit in uveljavljen, igralci so ga navajeni, provizije so zaradi masovne udeležbe nizke. Poleg tega prehod na rešitve brez TTP paradoksalno ne bi pomenil nujno, da igralci ne potrebujejo več strežnikov, razen, če bi igrali le za zabavo. Če je v igri denar, morajo biti vložki igralcev med igro nekje spravljeni, in ker si igralci med seboj ne zaupajo, je najnaravnejša rešitev TTP. Opozarjamo, da TTP v tem smislu nikakor ni več varnostno problematičen, saj ima celo manjša pooblastila kot npr. spletna banka. A zaradi spremenjenih pristojnosti ne bo takšen strežnik nič manj upravičeno pobiral provizije od igralcev.

Možnosti za razmah miselnega pokra nove generacije je pač treba gledati (tudi) skozi oči denarja. Najbliže temu pride predlog [5] iz leta 2005, ki se trudi poiskati kriptografske rešitve ne le za miselni poker v najožjem pomenu besede, temveč za vse, kar igro pokra spremlja v realnem življenju: avtentikacija igralcev, stave in izplačila.

Z avtentikacijo igralcev ne mislimo tiste klasične, ki poveže sporočilo z uporabniško identiteto. Seveda je pomembna tudi ta, a [5] se pionirsko posveti tudi avtentikaciji, ki uporabniško identiteto poveže s fizično osebo. V ta namen v shemo vpelje *regulatorja*, centralno, po možnosti državno institucijo, ki igralcem podeljuje spletno identiteto v obliki pametnih kartic. Pametno kartico je možno prevzeti le osebno, s čimer lahko onemogočimo ali vsaj močno otežimo dostop do spletnega igralništva mladoletnim, znanim prestopnikom in odvisnikom od iger na srečo. Takšna možnost omejevanja je zelo pomembna, saj je brez nje dostop do elektronskih igralnic hitrejši in enostavnejši kot do fizičnih, še posebej za mladoletnike. Postati in ostati odvisen od igralništva, celo že v mladosti, je zato žal vse lažje. V Združenih državah Amerike je po raziskavah državne agencije [22] od igralništva odvisen en odstotek vseh odraslih, nadaljnja dva do trije odstotki pa so v rizični skupini.

Pametne kartice naj bi po ideji iz opisane sheme prevzele tudi velik del računskega bremena. Vsaki igri bi sicer pasivno še vedno sledila igralna hiša (ki skrbi za izmenjavo denarja) in državni regulator, a za varnost bi se igralci zanesli na domnevo, da so od regulatorja izdane kartice poštene in odporne na posege in spremembe (*tamper-resistant*).

Tudi če bo [5] ali kak podoben predlog uspel in zaživel, se to ne bo zgodilo čez noč. Do takrat bodo svetu spletnega igralništva enako kot danes vladale trivialne sheme brez TTP. Te so začele svoj pohod leta 1994, 1998 je bila ustanovljena prva igralna hiša, kjer so igralci lahko stavili pravi denar, danes pa se oglasom za spletni poker med vsakdanjo uporabo interneta skoraj ne moremo izogniti.[1]

Zapisali smo že, da je uporaba TTP inherentno nevarna, saj ima TTP nadzor nad celotno igro. Občasno ta nevarnost res pokaže zobe. Na internetu je mogoče najti zapise o igralnih hišah, ki so priznale goljufanje s strani katerega od svojih uslužbencev z administratorskim dostopom do TTP, vendar ni videti, da bi bili taki dogodki zaskrbljujoče pogosti. Po drugi strani lahko o številu zamolčanih incidentov seveda samo špekuliramo. Igralne hiše bi lahko goljufale na številne neočitne načine: v igro lahko nalašč pošljejo navidezne igralce, ki dobivajo polno informacijo o igri in zato nadpovprečno pogosto zmagujejo; lahko premešajo karte tako, da novi igralci dobijo nekoliko boljše in ne zgubijo volje do igre; lahko dvema igralcema v isti igri razdelijo izrazito dobre karte, da povečajo možnost visokih stav in s tem višje provizije; in še kaj podobnega se najde.

Igralne hiše popolno informacijo o igri lahko zlorabijo, a njihova vloga centralnega nadzornika nad potekom igre ima tudi dobre strani. Predvsem jim to omogoča, da s pomočjo statistične analize vedenjskih vzorcev lovijo zarotniške skupine, znotraj katerih se igralci dogovorijo, da bodo drug drugemu kazali svoje karte. Če so na primer od šestih igralcev v igri štirje zarotniki, imajo očitno precej boljše možnosti za zmago.

Podobno kot se igralne hiše s statistiko lotijo nepoštenih igralcev, se igralnih hiš lotijo revizijske hiše. Igralne hiše jim v svojem prizadevanju, da se igralcem predstavijo kot poštene in zaupanja vredne, celo plačajo, da opravijo revizijo vseh odigranih in zabeleženih iger. Primer poročila s takšne revizije je v prilogi A. Če se pojavi utemeljen sum, da igralna hiša goljufa, običajno lahko – odvisno od zakonov v državi, kjer je hiša registrirana – revizijo zahteva tudi centralno državno regulacijsko telo, zadolženo za igralništvo.

Shemam s TTP gre torej v svetu prav dobro. V poznih devetdesetih letih se je spletno igralništvo komaj rojevalo, danes pa se letni promet meri v desetinah milijard dolarjev na leto; pričakuje se, da bo še nekaj let rasel za 20 odstotkov letno

[1] Iz etičnih razlogov mnoge spletne strani prepovedujejo oglase za spletno igralništvo, med drugimi Google in Yahoo. Vendar je strani, ki so takšne oglase še vedno pripravljene objaviti, precej.

[24]. Ocenjujejo, da spletni poker predstavlja 13% tega prometa. Seveda pa se tako eksplozivna rast ne more nadaljevati v nedogled. Roy Cooke, avtor številnih knjig o pokru in eden od začetnikov spletnega pokra, je v svoji kolumni [25] zapisal, da je trg spletnega pokra vedno opazneje nasičen, marketinški stroški za pridobitev novih igralcev strmo rasejo. V Veliki Britaniji poročajo, da se v zadnjem letu število spletnih igralcev ni opazno spremenilo [23]. V ZDA je "House of representatives" leta 2006 že sprejel zakon *Internet Gambling Prohibition and Enforcement Act*, ki povsem prepoveduje spletno igralništvo, pa ga je pozneje zavrnil kongres.

Kljub takim težavicam se za usodo spletnega pokra ni bati. V zadnjih letih je namreč močno pridobil na priljubljenosti. Zasluga gre eni glavnih razlik med spletnim in klasičnim pokrom: klasičnim igralnicam se postavitev velikega števila poker miz ne obrestuje, ker so zaslužki hiše tipično mnogo večji na igralnih avtomatih; na internetu pa so stroški postavitve mize minimalni in si spletne hiše lahko privoščijo na milijone uporabnikov.

Z izjemo dostopnosti pa je s stališča družbe razlik med miselnim in klasičnim pokrom malo: oba sta deležna nasprotujočih si pritiskov in ambicij s strani oblasti in igralcev, oba podobno sooblikujeta družbeno življenje. Miselni poker, četudi ne tisti *cutting-edge* brez TTP, je postal ustaljen del našega sveta.

Poglavje 6

Zaključek

Prepričali smo se, da je problem miselnega pokra težaven, a prav zaradi tega tudi izjemno zanimiv. Ponovimo samo nekaj najzanimivejših vprašanj, ki se zastavijo, ko želimo ustvariti shemo za decentralizirano igranje spletnega pokra: če nobenemu od igralcev ne zaupamo, kdo naj premeša karte? In če že uspemo s shemo doseči, da karte enakopravno premeša in zašifrira vsak od igralcev, kako naj igralec, ki je povlekel karto, soigralcu pove, kateri ključ za dešifriranje rabi, ne da bi mu hkrati povedal, katero karto je potegnil?

Predstavljene sheme v 4. poglavju ter uvod v 5. poglavje dokazujeta, da so odgovori danes znani. Še več, zadnjih nekaj let znamo na vprašanja odgovoriti s shemami, ki niso le varne, temveč tudi razmeroma učinkovite. Resda bi bilo koristno, če bi jih uspeli še nekoliko pohitriti, a glavni del problema, zastavljenega leta 1979, je danes zadovoljivo rešen. V preteklih petindvajsetih letih je pomagal buditi zanimanje za zaprisego bitov in dokaze brez razkritja znanja, danes pomembni orodji kriptografije. Tudi z njihovo pomočjo je danes miselni poker brez TTP pripravljen za uporabo v praksi.

Pogled na obstoječo prakso pa pravi drugače. Centralizirane sheme so dobro uveljavljene in sprejete. Je miselni poker, kot so si ga izvorno zamislili Shamir, Rivest in Adleman, res odgovor na težave današnjih spletnih igralnic? Se res želimo rešiti centralnega nadzornika v igri? Najboljša se zdi kompromisna rešitev, po kateri bi pregled nad igro dobile samo pooblaščene državne agencije z ustreznimi certifikati – a takšne sheme zaenkrat nimamo. Če pa bo naslednjih deset ali dvajset let področje miselnega pokra raziskovalno tako živo, kot je bilo zadnjih pet, se prav lahko zgodi, da se najde shema, nedvoumno boljša od današnjih rešitev s TTP.

Čas bo torej pokazal, ali bo miselni poker ostal le zanimiv teoretičen problem ali pa bo postal eden temeljev večmilijardne industrije.

Dodatek A

Primer poročila o reviziji spletne igralnice

V ilustracijo je priložen del poročila, na podlagi katerega je neodvisna revizorska agencija eni od večjih spletnih igralniških hiš izdala certifikat o poštenosti obratovanja.

Prva stran priloge je del statistične analize generatorja naključnih števil. Poleg izvedbe generičnih testov naključnosti revizorji preštejejo tudi, kolikokrat so bili igralcem podeljeni določeni "handi" (odlikovane kombinacije kart v pokru) in preverijo, ali so vrednosti v 95-odstotnem intervalu zaupanja.

Druga stran priloge predstavlja analizo izplačil igralcem. Odstotek vložkov, ki ga hiša pobere kot provizijo, namreč ni fiksen. Razlog je, da so nekatere provizije proporcionalne vložkom, nekatere pa fiksne. Analiza preveri, če je v daljšem časovnem obdobju odstotek vplačil, pobran kot provizija, v statistično pričakovanem intervalu.

Poker Card Analysis

The Directors

Cassava Enterprises (Gibraltar) Limited; 888 Holdings Plc
Suite 601-701, Europort
Gibraltar

This is to confirm that iTech Labs has examined the game logs for Poker games for the period July 1, 2007 to December 31, 2007 as recorded by the respective game servers and analyzed the Poker cards for statistical randomness. The results of the analysis are given below.

URLs: www.888.com, www.pacificpoker.com

The following tables compare the observed proportion of times the various types of events occurred (**Sample** column). **Probability** shows the theoretical values. **Lower** and **Upper** are the 95% confidence limits.

Actual and theoretical probabilities for various Poker hands

Hands	Theoretical Probability	Lower	Sample	Upper
Royal Flush	0.000032	0.000027	0.000028	0.000039
Straight Flush	0.000279	0.000262	0.000286	0.000296
4 of a kind	0.001681	0.001639	0.001657	0.001723
Full House	0.025961	0.025797	0.025974	0.026126
Flush	0.030255	0.030078	0.030117	0.030432
Straight	0.046194	0.045977	0.046163	0.046411
3 of a kind	0.048299	0.048077	0.048132	0.048521
2 pairs	0.234955	0.234517	0.234882	0.235394
1 pair	0.438225	0.437713	0.438628	0.438738
High card	0.174119	0.173728	0.174134	0.174511

Calculations using live cards:

Notes:

a) In the above table, the numbers in the **Sample** column (observed probabilities) lie within 95% confidence intervals.

b) Theoretical probability of 0. 438225 means there are 43,822 chances in 100,000 for getting a pair.

Actual and theoretical probabilities for Ranks

Rank	Theoretical Probability	Lower	Sample	Upper
A	0.077	0.076819	0.076854	0.077027
2	0.077	0.076819	0.076941	0.077027
3	0.077	0.076819	*0.077033*	0.077027
4	0.077	0.076819	0.076934	0.077027
5	0.077	0.076819	0.076876	0.077027
6	0.077	0.076819	0.076857	0.077027
7	0.077	0.076819	0.076951	0.077027
8	0.077	0.076819	0.076948	0.077027
9	0.077	0.076819	0.076952	0.077027
10	0.077	0.076819	0.076991	0.077027
J	0.077	0.076819	0.076860	0.077027
Q	0.077	0.076819	0.076928	0.077027
K	0.077	0.076819	0.076877	0.077027

Casino Game Logs Audit Report

The Directors

Cassava Enterprises (Gibraltar) Limited; 888 Holdings Plc
Suite 601-701, Europort, Gibraltar

This is to confirm that iTech Labs has examined the casino game logs in the gaming system of Cassava Enterprises (Gibraltar) Limited for the period **April 01, 2008 to April 30, 2008** as recorded by the respective game servers. iTech Labs has calculated the Return To Player (RTP) for all the games using the data from the game logs.

The table below lists calculated RTPs for real games played at the Cassava Enterprises (Gibraltar) Limited game servers. The Cassava game servers service the following internet gaming sites:
www.casino-on-net.com www.pacificpoker.com www.reefclubcasino.com www.888.com
www.towertorneos.com www.rileyspoker.com www.luckyacepoker.com
www.luckyacecasino.com www.goldchip-gaming.com

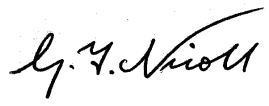
Game Name	Actual RTP
Baccarat	98.36%
Blackjack	97.85%
Craps	97.76%
Roulette	96.80%
Video Poker, Power Video Poker	96.58%
Video Slot	95.23%
Slots	93.35%
All other games combined*	96.69%
Average payout for all games	96.51%

* 'All other games combined' includes Keno, Pai Gow Poker and Caribbean Poker.

The actual RTPs shown are percentages of total actual winnings to bets for each game type. They do not reflect outcomes for individual games or future games. iTech Labs has done limited sanity checks to verify the integrity of the game logs. iTech Labs also maintains a copy of the game logs for verification purposes. There were a large enough number of game records to give the RTP calculations sufficient statistical power. The actual RTPs are greater than normally required by gaming jurisdictions. For all card games, the actual RTPs are very close to the maximum theoretical expectation.

The scope of the review did not include reviews of financial controls or casino operations. We believe that the RTPs were calculated correctly from the game logs based on the actual bets and winnings.

Signed:



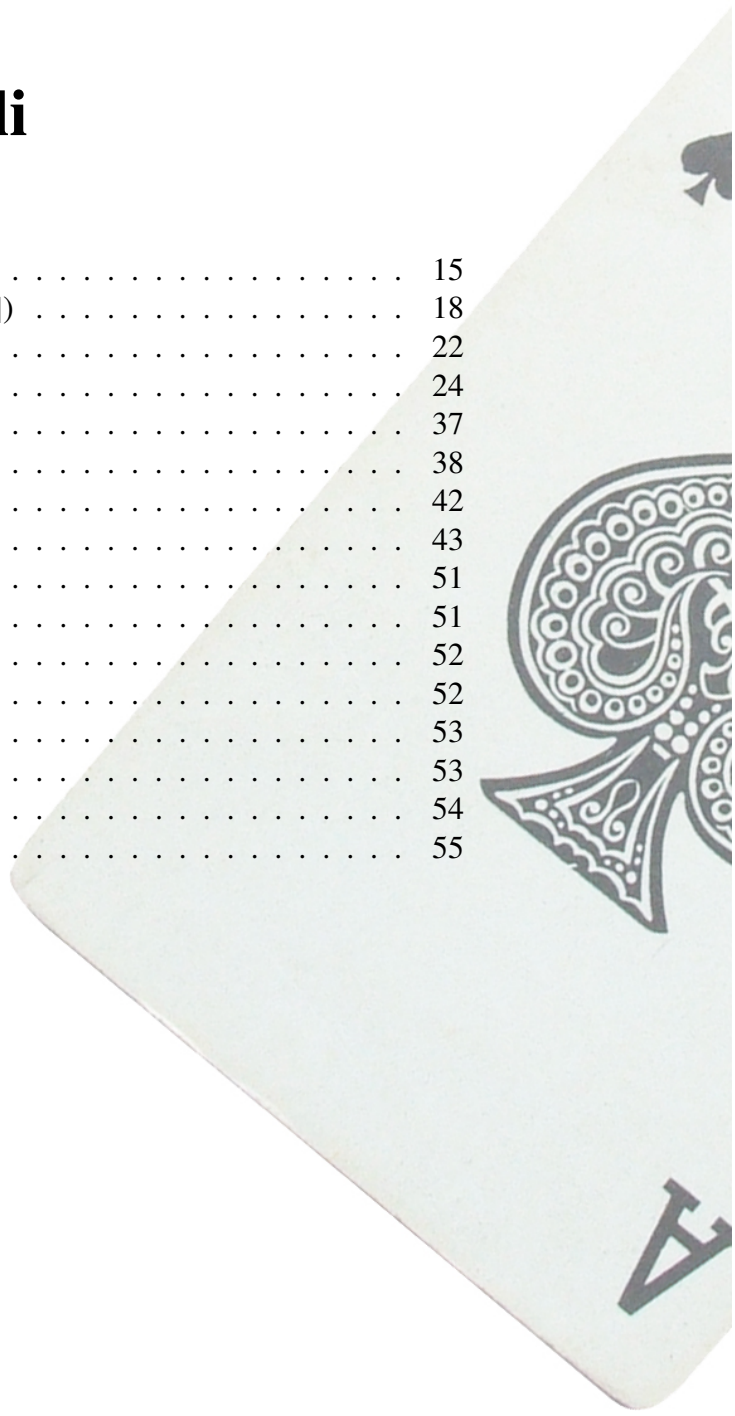
Geoff Nicoll
Principal Consultant
iTech Labs Australia

Date: July 10, 2008

Disclaimer.
While it is not possible to test all possible scenarios in a laboratory environment, iTech Labs has conducted a level of testing appropriate for a component test of this type.

Algoritmi in protokoli

3.1.1 PollardP1(n, B) 15
3.1.2 PollardRho($n, x_1, [f = x \mapsto x^2 + 1]$) 18
3.2.3 Shanks(n, α, β) 22
3.2.4 PollardRhoDLP($n, A_1, B_1, \alpha, \beta$) 24
3.5.5 HamiltonovCikelZKP(G, s) 37
3.5.6 ChaumPedersen(p, g_1, y_1, g_2, y_2, x) 38
4.1.7 Mešanje() 42
4.1.8 Vlečenje(i) 43
4.3.9 Inicializacija() 51
4.3.10 SkupinskaVrednost(d) 51
4.3.11 Mešanje(C_0) 52
4.3.12 MešanjeKorak(C) 52
4.3.13 TestMešanjeKorak(C, R, π, C^*) 53
4.3.14 MešanjeDokaz(i, C, C^*, π, R) 53
4.3.15 Vlečenje(i) 54
4.3.16 Razkrivanje(i) 55



Literatura

[1] Dan Boneh, "The Decision Diffie-Hellman Problem," v zborniku *Algorithmic Number Theory, Third International Symposium, ANTS-III*, Portland, Oregon, USA, junij 1998, strani 48–63.

[2] Jordi Castellà Roca in drugi, "Practical Mental Poker Without a TTP Based on Homomorphic Encryption," v zborniku *Progress in Cryptology (INDOCRYPT 2003), 4th International Conference on Cryptology*, New Delhi, India, december 2003, strani 280–294.

[3] Jordi Castellà Roca, Francesc Sebé, Josep Domingo-Ferrer, "Dropout-Tolerant TTP-Free Mental Poker," v zborniku *Trust, Privacy and Security in Digital Business: Second International Conference (TrustBus 2005)*, Copenhagen, Denmark, avgust 2005, strani 30–40.

[4] Jordi Castellà Roca, *Contributions to Mental Poker*, doktorska disertacija, Universitat Autònoma de Barcelona, 2005.

[5] Jordi Castellà Roca, Josep Domingo-Ferrer, Francesc Sebé, "Smart-Card Based Mental Poker," v zborniku *Smart Card Research and Advanced Applications, 7th IFIP WG 8.8/11.2 International Conference (CARDIS 2006)*, Tarragona, Spain, april 2006.

[6] David Chaum, Torben Pryds Pedersen, "Wallet Databases with Observers (Extended Abstract)," v zborniku *Advances in Cryptology - CRYPTO'92 Proceedings*, Santa Barbara, California, USA, avgust 1992, strani 89–105.

[7] Claude Crépeau, "A Secure Poker Protocol that Minimizes the Effect of Player Coalitions," v zborniku *Advances in Cryptology - CRYPTO'85 Proceedings*, Santa Barbara, California, USA, avgust 1985, strani 73–86.

[8] Claude Crépeau, "A zero-knowledge poker protocol that achieves confidentiality of the players' strategy or how to achieve an electronic poker face," v

zborniku *Advances in Cryptology - CRYPTO'86 Proceedings*, Santa Barbara, California, USA, avgust 1986, strani 239–247.

[9] Steven Fortune in Michael Merritt, "Poker protocols," v zborniku *Advances in Cryptology - CRYPTO'84 Proceedings*, Santa Barbara, California, USA, avgust 1984, strani 454–464.

[10] Withfeld Diffie in Martin Hellmann, "New Directions in Cryptography," *IEEE Trans. on Info. Theory, IT-22*, november 1976, strani 644–654.

[11] Shafi Goldwasser, Silvio Micali in Charles Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, zvezek 18, strani 186–208, 1989.

[12] Phillippe Golle, "Dealing Cards in Poker Games," v zborniku *International Symposium on Information Technology: Coding and Computing (ITCC 2005), Volume 1*, Las Vegas, Nevada, USA, april 2005, strani 506–511.

[13] Kaoru Kurosawa, Yutaka Katayama in Wakaha Ogata, "Reshufflable and Laziness Tolerant Mental Card Game Protocol," v zborniku *IEICE Transactions Fundamentals*, januar 1997, strani 72–78.

[14] Josef Pieprzyk, Thomas Hardjono in Jennifer Seberry, *Fundamentals of Computer Security*, Springer, 2003.

[15] Adi Shamir, Ron Rivest in Leonard Adleman, "Mental Poker," v *MIT Technical Report*, februar 1979.

[16] Heiko Stamer, *Bibliography on Mental Poker*, 2007.

[17] Douglas R. Stinson, *Cryptography: Theory and Practice*, tretja izdaja, CRC Press, 1995.

[18] Andrew Chi-Chih Yao, "Protocols for secure computations", v zborniku *Proceedings of the twenty-third annual IEEE Symposium on Foundations of Computer Science*, Chicago, Illinois, USA, november 1982, strani 160–164.

[19] Weiliang Zhao, Vijay Varadharajan and Yi Mu, "A Secure Mental Poker Protocol Over The Internet", v zborniku *ACSW Frontiers 2003, 2003 ACSW Workshops*, Adelaide, Avstralija, februar 2003, strani 105–109.

[20] (2008) Sophie Germain prime. Dostopno na:
`http://planetmath.org/encyclopedia/GermainPrime2.html`

[21] (2008) Online Poker History. Dostopno na:
http://www.pokertips.org/history/online-poker.php

[22] (2008) National Council on Problem Gambling: Problem Gamblers. Dostopno na:
http://www.ncpgambling.org/i4a/pages/Index.cfm?pageID=3315

[23] (2008) UK Online Gambling Numbers Static. Dostopno na:
http://www.igamingbusiness.com/article-detail.php?articleID=18238

[24] (2008) iGaming Industry Snapshot. Dostopno na:
http://gamingpublic.com/

[25] (2008) Roy Cooke: The Business of Internet Poker. Dostopno na:
http://www.cardplayer.com/magazine/article/14257

Izjava

Izjavljam, da sem diplomsko nalogo izdelal samostojno pod vodstvom mentorja prof. dr. Aleksandra Jurišića. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Ljubljana, 10. september 2008 Mitja Trampuš