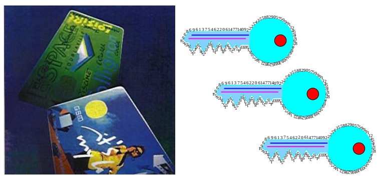
KRIPTOGRAFIJA IN TEORIJA KODIRANJA

Aleksandar Jurišić

Center za kriptografijo in računalniško varnost
Politehnika Nova Gorica

http://valjhun.fmf.uni-lj.si/~ajurisic



Uvod

Odkar so ljudje pričeli komunicirati,
pa naj si bo to preko govora, pisave, radija,
telefona, televizije ali računalnikov,
so želeli tudi *skrivati* vsebino svojih sporočil.

Ta nuja, oziroma že kar obsedenost
po *tajnosti*, je imela dramatičen vpliv
na vojne, monarhije in seveda tudi
na individualna življenja.

Vladarji in generali so odvisni od uspešne
in učinkovite komunikacije že tisočletja,
hkrati pa se zavedajo posledic, v primeru,
če njihova sporočila pridejo v napačne roke,
izdajo dragocene skrivnosti rivalom ali
odkrijejo vitalne informacije nasprotnikom.

Danes vse to velja tudi za moderna vodstva
uspešnih podjetij in tako postaja

"informacijska/računalniška varnost"

eno izmed najbolj pomembnih gesel
informacijske dobe.

Vlade, industrija ter posamezniki,
vsi hranijo informacije v *digitalni obliki*.

Ta medij nam omogoča številne prednosti
pred fizičnimi oblikami:

- je zelo kompakten,

- prenos je takorekoč trenuten,

hkrati pa je omogočen tudi
- organiziran dostop do raznovrstnih podatkovnih baz.

Z razvojem

- telekomunikacij,
- računalniških omrežij in
- obdelovanja informacij

pa je precej lažje prestreči in spremeniti
digitalno (elektronsko) informacijo kot pa
njenega *papirnega predhodnika*.

Zato so se povečale zahteve po **varnosti**.

Informacijska in računalniška varnost

opisuje vse preventivne postopke in sredstva
s katerimi preprečimo nepooblaščeno uporabo
digitalnih podatkov ali sistemov,
ne glede na to ali gre pri ustreznih podatkih kot sta

digitalni denar (nosilec vrednosti) in
digitalni podpis (za prepoznavanje)

za

- razkritje,
- spreminjanje,
- zamenjavo,
- uničenje,
- preverjanje verodostojnosti.

Predlagani so bili številni ukrepi, a niti eden med njimi
ne zagotavlja *popolne varnosti*.

Med preventivnimi ukrepi, ki so na voljo danes, nudi

kriptografija

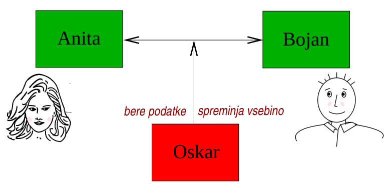
(če je seveda pravilno implementirana ter uporabljana)

največjo stopnjo varnosti

glede na svojo prilagodljivost digitalnim medijem.

Kaj je kriptografija?

Kriptografija je veda o komunikaciji v prisotnosti aktivnega napadalca.



Primer:
pošiljanje papirnih dokumentov po pošti

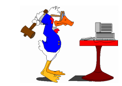
Kakšna zagotovila varnosti so na voljo? In kako?

- **Fizična varnost:** zapečatene kuverte.
- **Zakonska infrastruktura:** ročni podpis je zakonsko sprejeto sredstvo, zakoni proti odpiranju/oviranju pošte, itd.
- **Poštna infrastruktura:** varni in sprejeti mehanizmi za dostavljanje pošte širom po svetu.

Primer: digitalni podatki

- **ZA**: hranjenje je enostavno in poceni, hiter in enostaven transport.
- **PROTI**: enostavno kopiranje; transportni mediji niso varni (npr. pogovor po mobilnem telefonu, internetna seja, ftp seja, komunikacija s pomočjo elektronske pošte).
- **Vprašanje**: Kako lahko omogočimo/ponudimo enake možnosti za papirni kakor tudi digitalni svet?

Odšifriranje (razbijanje) klasičnih šifer



Kriptografske sisteme kontroliramo s pomočjo ključev, ki določijo transformacijo podatkov.
Seveda imajo tudi ključi digitalno obliko (binarno zaporedje: 01001101010101...).

Držali se bomo **Kerckhoffovega principa**, ki pravi, da "nasprotnik"

pozna kriptosistem oziroma algoritme, ki jih uporabljamo, ne pa tudi ključe, ki nam zagotavljajo varnost.

Vohunova dilema

Bilo je temno kot v rogu, ko se je vohun vračal v grad po opravljeni diverziji v sovražnem taboru.

Ko se je približal vratom, je zaslišal šepetajoč glas:

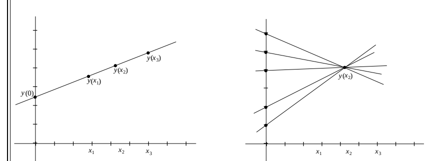
Kako vohun prepriča "stražarja", da pozna geslo, ne da bi ga izdal morebitnemu vsiljivcu/prisluškovalcu?

Deljenje skrivnosti

Problem: *V banki morajo trije direktorji odpreti trezor vsak dan, vendar pa ne želijo zaupati kombinacijo nobenemu posamezniku. Zato bi radi imeli sistem, po katerem lahko odpreta trezor poljubna dva med njimi.*

Ta problem lahko rešimo z $(2, 3)$-stopenjsko shemo.

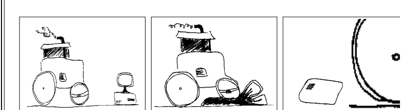
Stopenjske sheme za deljenje skrivnosti sta leta 1979 neodvisno odkrila **Blakey in Shamir**.



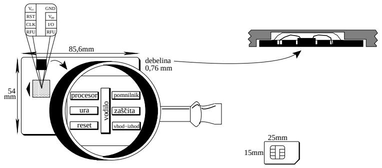
Vsak dobi le y-koordinato svoje točke.
Program v trezorju ima še ustrezne od 0 različne x- koordinate, zato lahko izračuna ključ $y(0)$.
Vsaki točki natanko določata premico in s tem ključ.

Če imamo eno samo točko, ne moremo ugotoviti, kateri ključ je pravi, saj so vsi videti enako dobri.

Pametne kartice



Po računski moči so pametne kartice primerljive z originalnim IBM-XT računalnikom,
kartice s **kripto koprocesorjem** pa v nekaterih opravilih prekašajo celo 50 Mhz 486 računalnik.



Velikost pametne kartice ustreza ISO 7810 standardu, sestavljajo pa jo mikroprocesor, pomnilnik (ROM, RAM, EEPROM), vhodno/izhodna enota (I/O).

Zakaj pametna kartica

Gotovo je najbolj pomembna razlika med pametno kartico in magnetno kartico

varnost.

Pametna kartica ima svoj **procesor**, ki kontrolira vse interakcije med od zunaj nedostopnim spominom in različnimi zunanjimi enotami.

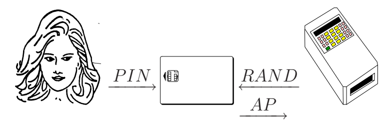
Dodaten, pomemben, del pametne kartice je **non-volatile spomin (ROM)**, t.j. spomin, ki se ga ne da spremeniti in ostane prisoten tudi po prekinitvi napajanja.

Zagotovitev varnosti

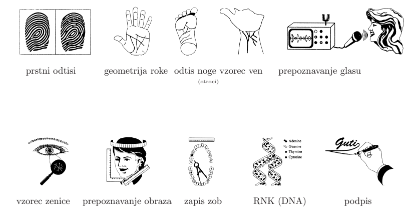
Identifikacija se opravi v dveh delih:

(a) kartica mora biti zares prepričana, da jo uporablja njen lastnik (lokalno overjanje),

(b) kartica komunicira (varno) z računalnikom (dinamično overjanje).



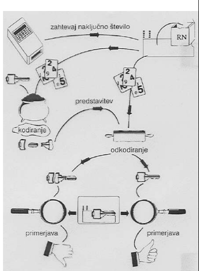
Biometrični testi



Pametna kartica zgenerira naključno število, ter ga pošlje čitalniku.

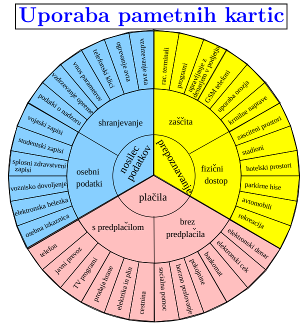
Ta ga zašifrira z zasebnim ključem in rezultat pošlje pametni kartici.

Če pametna kartica uspešno odšifrira naključno število z javnim ključem, potem je prepričana o pristnosti čitalnika.

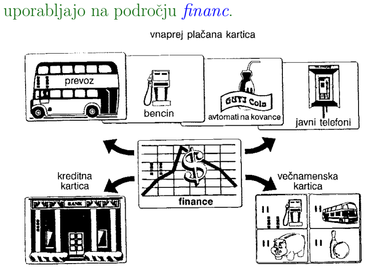


Enak proces poteka v nasprotni smeri.

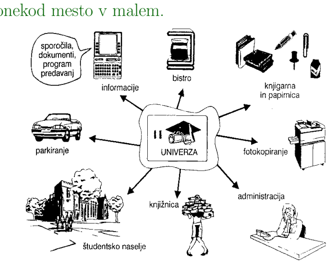
Uporaba pametnih kartic



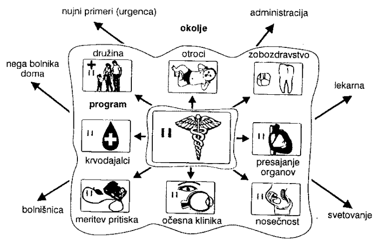
Plačilne, kreditne in večnamenske kartice, ki se uporabljajo na področju *financ*.



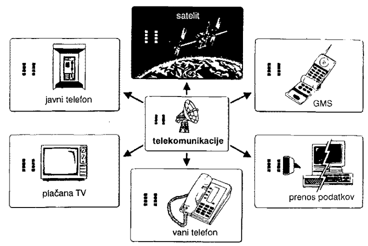
Uporaba pametnih kartic na *univerzi/fakulteti*, ki je ponekod mesto v malem.



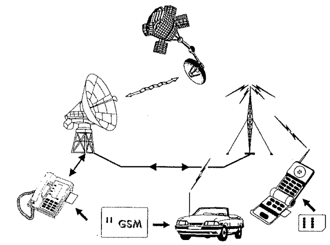
Področja v *zdravstvu*, kjer se uporabljajo pametne kartice.



Uporaba pametne kartice v *telekomunikacijah* in uporabniški elektrotehniki.



GSM (globalni sistem za prenosno komuniciranje)



Javna kriptografija

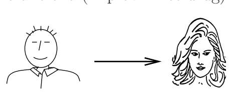
Glede na pomembnost podatkov, ki jih varujemo, se moramo odločiti za ustrezno obliko zaščite:

- Geslo (PIN) in zgoščevalne funkcije predstavljajo osnovno zaščito,
- AES (Advanced Encryption Standard) simetrični kriptosistemi nudijo srednji nivo,
- javna kriptografija (Public Key Scheme) pa visok nivo zaščite.

Odlična uvodna knjiga o moderni kriptografiji je: Albrecht Beutelspacher, **Cryptology**, MAA, 1994.

Koncept javne kriptografije

Bojan pošlje Aniti pismo, pri tem pa si želi, da bi pismo lahko prebrala le ona (in prav nihče drug) **[zaščita]**.



Anita pa si poleg tega želi biti prepričana, da je pismo, ki ga je poslal Bojan prišlo prav od njega **[podpis]**.

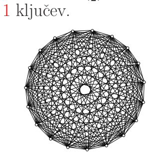
Predpostavimo, da se Anita in Bojan prej dogovorita za **skupen ključ**, ki ga ne pozna nihče drug (simetričen kriptosistem).

Če Bojan z njim zašifrira pismo, je lahko prepričan, da ga lahko odklene le Anita.

Hkrati pa je tudi Anita zadovoljna, saj je prepričana, da ji je pismo lahko poslal le Bojan.

Tak pristop je problematičen vsaj iz dveh razlogov:

1. Anita in Bojan se morata *prej* dogovoriti za skupen ključ,
2. upravljanje s ključi v omrežju z n uporabniki je kradratne zahtevnosti $\binom{n}{2}$, vsak uporabnik pa mora hraniti $n-1$ ključev.



Leta 1976 sta Whit **Diffie** in Martin **Hellman** predstavila koncept kriptografije z javnimi ključi.

Tu ima za razliko od sim. sistema vsak uporabnik **dva** ključa, podatke *zaklepa*, drugi pa jih *odklepa*.

Pomembna lastnost tega sistema:
ključ, ki zaklepa, ne more odklepati
in obratno,
ključ, ki odklepa, ne more zaklepati.

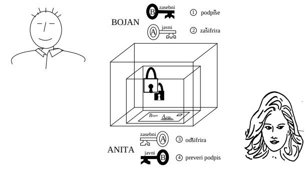
To omogoči lastniku, da en ključ objavi, drugega pa hrani v tajnosti (npr. na pametni kartici). Zato imenujemo ta ključa zaporedoma **javni** in **zasebni**.

Ta pristop omogoča veliko presenetljivih načinov uporabe, npr. omogoča ljudem varno komuniciranje, ne da bi se predhodno srečali zaradi izmenjave/dogovora o tajnem ključu.

Vsak uporabnik najprej objavi svoj javni ključ, zasebnega pa zadrži zase. Vsak lahko nato z javnim ključem zašifrira pismo, bral (odšifriral) pa ga bo lahko le lastnik ustreznega zasebnega ključa.

Bojan pošlje Aniti podpisano zasebno pismo:

(1) podpiše ga s svojim zasebnim ključem Z_B in ga
(2) zašifrira z Anitinim javnim ključem J_A.



(3) Anita ga s svojim zasebnim ključem Z_A odšifrira,
(4) z Bojanovim javnim ključem J_B preveri podpis.

V razvoju javne kriptografije je bilo predlaganih in razbitih veliko kriptosistemov.

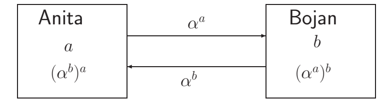
Le nekaj se jih je obdržalo in jih lahko danes smatramo za varne in učinkovite.

Glede na matematični problem na katerem temeljijo, so razdeljene v tri skupine:

- **Sistemi faktorizacije celih števil**
  npr. RSA (Rivest-Shamir-Adleman).
- **Sistemi diskretnega logaritma**
  npr. DSA.
- **Kripto sistemi z eliptičnimi krivuljami**
  (Elliptic Curve Cryptosystems)

Izmenjava ključev

(Diffie-Hellman)



Anita in Bojan si delita skupni element grupe: α^{ab}.

Končne grupe so zanimive zato, ker računanje potenc lahko opravimo učinkovito, ne poznamo pa vedno učinkovitih algoritmov za logaritem (za razliko od $\mathbb{R}$).

Kaj je kriptografija

- cilji kriptografije
- širši pogled na kriptografijo
- gradniki kriptografije

Osnovna motivacija za naš študij je uporaba kriptografije v realnem svetu.

Cilje kriptografije bomo dosegali z matematičnimi sredstvi.

Cilji kriptografije

1. **Zasebnost/zaupnost/tajnost:**
   varovanje informacij pred tistimi, ki jim vpogled ni dovoljen, dosežemo s šifriranjem.
2. **Celovitost podatkov:**
   zagotovilo, da informacija ni bila spremenjena z nedovoljenimi sredstvi (neavtoriziranimi sredstvi).

3. **Overjanje sporočila (ali izvora podatkov):**
   potrditev izvora informacij.
4. **Identifikacija:**
   potrditev identitete predmeta ali osebe.
5. **Preprečevanje tajenja:**
   preprečevanje, da bi nekdo zanikal dano obljubo ali storjeno dejanje.

6. **Drugi kriptografski protokoli:**
   1. grb/cifra po telefonu
   2. mentalni poker
   3. shema elektronskih volitev
      (anonimno glasovanje brez goljufanja)
   4. (anonimni) elektronski denar

Cilji kriptografije:

1. *zasebnost/zaupnost/tajnost*
2. *celovitost podatkov*
3. *overjanje sporočila (ali izvora podatkov)*
4. *identifikacija*
5. *preprečevanje nepriznavanja*
6. *drugi kriptografski protokoli*

NAUK: Kriptografija je več kot samo širfiranje (enkripcija).

Širši pogled na kriptografijo – varnost informacij

Kriptografija je sredstvo, s katerim dosežemo varnost informacij, ki med drugim zajema:

(a) Varnost računalniškega sistema

tj. tehnična sredstva, ki omogočajo varnost računalniškega sistema, ki lahko pomeni samo en računalnik z več uporabniki, lokalno mrežo (LAN), Internet, mrežni strežnik, bankomat, itd.

Med drugim obsega:
- varnostne modele in pravila, ki določajo zahteve po varnosti, katerim mora sistem ustrezati
- varen operacijski sistem
- zaščito pred virusi
- zaščito pred kopiranjem
- kontrolne mehanizme (beleženje vseh aktivnosti, ki se dogajajo v sistemu lahko omogoči *odkrivanje* tistih kršitev varnostnih pravil, ki jih ni mogoče preprečiti)
- analiza tveganja in upravljanje v primeru nevarnosti

(b) Varnost na mreži

Zaščita prenašanja podatkov preko komercialnih mrež, tudi računalniških in telekomunikacijskih.

Med drugim obsega:

- protokole na internetu in njihovo varnost
- požarne zidove
- trgovanje na internetu
- varno elektronsko pošto

Širši pogled na kriptografijo – varnost informacij

1. *varnost računalniškega sistema*
2. *varnost na mreži*

NAUK: Kriptografija je samo majhen del varnosti informacij.

Gradniki kriptografije

1. **matematika** *(predvsem teorija števil)*
2. **računalništvo** *(analiza algoritmov)*
3. **elektrotehnika** *(hardware)*
4. **poznavanje aplikacij** *(finance,...)*
5. **politika** *(restrikcije, key escrow, NSA,...)*
6. **pravo** *(patenti, podpisi, jamstvo,...)*
7. **družba** *(npr. enkripcija omogoča zasebnost, a otežuje pregon kriminalcev)*

NAUK: Uporabna kriptografija je več kot samo zanimiva matematika.

1. poglavje

Klasična kriptografija

- zgodovina (hieroglifi, antika, II. svetovna vojna)
- zamenjalna šifra

Klasične šifre in razbijanje

- prikrita, zamenjalna (pomična, afina), bločna (Vigenerjeva, Hillova)
- Kerckhoffov princip in stopnje napadov
- napad na Vigenerja (Kasiski test, indeks naključja)
- napad na Hillovo šifro
- tokovne šifre

Zgodovina

Kriptografija ima dolgo in zanimivo zgodovino:

– Hieroglifi, Špartanci, Cezar, ...

D. Kahn, **The Codebreakers** (The Story of Secret Writing), hrvaški prevod: (K. and M. Miles), **Šifranti protiv špijuna**, Centar za informacije i Publicitet, Zagreb 1979. (429+288+451+325=1493 strani).

Hieroglifi

Razvili so jih antični Egiptčani.
Komunicirali so v jeziku
sestavljenemu iz sličič namesto besed.

Najbolj izobraženi ljudje so jih razumeli,
toda v religioznemu kontekstu
– **npr. napisi na grobovih** –
so njihovi duhovniki uporabljali tajne
kriptografske verzije znakov, da bi bila vsebina
več vredna (saj je šlo za božje besede) in bolj mistična.

Mnoge religije so uporabljale tajne znake,
ki so jih razumeli le določeni izbranci.

Razbijalci šifer

Obstajajo od kar poznamo šifriranje.

L. 1799 so v Egipčanski Rosetti
našli skoraj 2.000 let star kamen.
Na njemu so bili trije teksti:

- hieroglifi,
- pisava egiptčanov (demotic) in
- starogrščina.

Del kamna iz Rosette, na katerem je bila neznana pisava, dokler je arheologi niso odšifrirali.

Ko je bil končan prevod iz Grščine,
je bilo možno razvozlati tudi hieroglife,
iz katerih smo izvedeli o zgodovini antičnega Egipta.

Še ena antična: o obriti glavi

Medtem, ko je bil genialni Histius na perzijskem sodišču, je hotel obvestiti Aristagorasa iz Grčije, da dvigne upor. Seveda je bilo pomembno, da nihče ne prestreže sporočilo.

Da bi zagotovil tajnost, je Histius obril sužnja, ki mu je nabolj zaupal, mu vtetoviral na glavo sporočilo [sužnju so rekli, da mu začenjajo zdraviti slepoto] in počakal, da mu zrastejo lasje.
Sužnju je bilo ukazano, da reče Aristagorasu:

"*Obrijte mojo glavo in poglejte nanjo.*"

Aristagoras je nato zares dvignil upor.

To je primer **prikrite šifre**,
sporočilo je prisotno, a na nek način prikrito.

Poznamo mnogo takšnih primerov.

Varnost takega sporočila je odvisna od trika prikrivanja.

Tak trik je lahko odkriti, poleg tega pa ne omogoča hitrega šifriranja in odšifriranja.

To ne pride vpoštev za **resno uporabo**.

Anglija: Sir John dobi sporočilo: Worthie Sir John:- Hope, that is ye beste comfort of ye afflicted, cannot much, I fear me, help you now. That I would saye to you, is this only: if ever I may be able to requite that I do owe you, stand not upon asking me. 'Tis not much that I can do: but what I can do, bee ye verie sure I wille. I knowe that, if dethe comes, if ordinary men fear it, it frights not you, accounting it for a high honor, to have such a reward of your loyalty. Pray yet that you may be spared this soe bitter, cup. I fear not that you will grudge any sufferings; only if bie submission you can turn them away, 'tis the part of a wise man. Tell me, an if you can, to do for you anythinge that you wolde have done. The general goes back on Wednesday. Restinge your servant to command. - R.T.

Če vam uspe "med vrsticami" prebrati:

**PANEL AT EAST END
OF CHAPEL SLIDES**

verjetno ne boste občutili enakega olajšanja kot Sir John Trevanion, njemu pa je vsekakor uspelo pobegniti, sicer bi ga v gradu Colcester gotovo usmrtili prav tako, kot so Sir Charlesa Lucasa ter Sir Georga Lislea.

Druga svetovna vojna

– Enigma (Nemčija),

– Tunny (Nemčija),

– Purple (Japonska),

– Hagelin (ZDA).

Zamenjalna šifra

Tomaž Pisanski, Skrivnostno sporočilo
Presek V/1, 1977/78, str. 40-42.

```
YHW?HD+CVODHVTHVO-!JVG:CDCYJ(JV/-V?HV(
-T?HVW-4YC4(?-DJV/-(?S-VO3CWC%J(-V4-DC
V!CW-?CVNJDJVD-?+-VO3CWC%J(-VQW-DQ-VJ+
V?HVDWHN-V3C:CODCV!H+?-DJVD-?+CV3JO-YC
```

(črko Č smo zamenjali s C, črko Ć pa z D)

Imamo 26! = 403291461126605635584000000
možnosti z direktnim preizkušanjem,
zato v članku dobimo naslednje nasvete:

(0) Relativna frekvenca črk in presledkov v slovenščini:
presledek 173,

E	A	I	O	N	R	S	L	J	T	V	D
89	84	74	73	57	44	43	39	37	37	33	30

K	M	P	U	Z	B	G	Č	H	Š	C	Ž	F
29	27	26	18	17	15	12	12	9	9	6	6	1

(1) Na začetku besed so najpogostejše črke
N, S, K, T, J, L.

(2) Najpogostejše končnice pa so
E, A, I, O, U, R, N.

(3) Ugotovi, kateri znaki zagotovo predstavljajo samoglasnike in kateri soglasnike.

(4) V vsaki besedi je vsaj en samoglasnik
ali samoglasniški R.

(5) V vsaki besedi z dvema črkama je ena
črka samoglasnik, druga pa soglasnik.

(6) detektivska sreča

(0)	V	-	C	D	J	?	H	W	O	(	+	3
	23	19	16	12	11	10	9	7	6	6	5	4

Y	4	!	/	Q	:	%	T	N	S	G
4	3	3	2	2	2	2	2	2	1	1

Zaključek `V --> ' '` (drugi znaki z visoko frekvenco ne morejo biti).

Dve besedi se ponovita: `O3CWC%J(-`,
opazimo pa tudi eno sklanjatev:
`D-?+-` ter `D-?+C`.

Torej nadaljujemo z naslednjim tekstom:

```
YHW?HD+C ODH TH O-!J G:CDCYJ(J /- ?H
(-T?H W-4YD4(?-DJ /-(?S- O3CWC%J(- 4-DC
!CW-?C NJDJ D-?+- O3CWC%J(- QW-DQ- J+
?H DWHN- 3C:CODC !H+?-DJ D-?+C 3JO-YC
```


(3) Kanditati za samoglasnike e,a,i,o so znaki z visokimi frekvancami. Vzamemo:

$$\{\text{e,a,i,o}\} = \{\text{-,C,J,H}\}$$

(saj D izključi -,H,J,C in ? izključi -,H,C,
znaki -,C,J,H pa se ne izključujejo)

Razporeditev teh znakov kot samoglasnikov izgleda prav verjetna. To potrdi tudi gostota končnic, gostota parov je namreč:

AV	CV	HV	JV	VO	?H	-D	DC	JM	W-	DJ	UC	CW	-?	VD
7	5	5	5	4	4	4	3	3	3	3	3	3	3	3

(5) Preučimo besede z dvema črkama:

```
   Samoglasnik na koncu
1) da ga na pa ta za (ha ja la)
2) če je le me ne se še te ve že (he)
3) bi ji ki mi ni si ti vi
4) bo do (ho) jo ko no po so to
5) ju mu tu (bu)
6) rž rt
```



```
   Samoglasnik na začetku
1) ar as             (ah aj au)
2) en ep              (ej eh)
3) in iz ig
4) on ob od os on     (oh oj)
5) uk up uš ud um ur (uh ut)
```

in opazujemo besedi: /- ?H
ter besedi: J+ ?H.

J+ ima najmanj možnosti, + pa verjetno ni črka n, zato nam ostane samo še:

```
J+ ?H        DWHN-
/- ?H
iz te       (ne gre zaradi: D-?+C)
ob ta(e,o)  (ne gre zaradi: D-?+C)
od te       (ne gre zaradi: D-?+C)
```

tako da bo potrebno nekaj spremeniti in preizkusiti še naslednje:
on bo; on jo; in so; in se; in je; in ta; en je; od tu ...

(6) Če nam po dolgem premisleku ne uspe najti rdeče niti, bo morda potrebno iskati napako s prijatelji (tudi računalniški program z metodo lokalne optimizacije ni zmogel problema zaradi premajhne dolžine tajnopisa, vsekakor pa bi bilo problem mogoče rešiti s pomočjo elektronskega slovarja).

Tudi psihološki pristop pomaga, je svetoval Martin Juvan in naloga je bila rešena (poskusite sami!).

Podobna naloga je v angleščini dosti lažja, saj je v tem jeziku veliko členov THE, A in AN, vendar pa zato običajno najprej izpustimo presledke iz teksta, ki ga želimo spraviti v tajnopis.

V angleščini imajo seveda črke drugačno gostoto kot v slovenščini.

Razdelimo jih v naslednjih pet skupin:

1. E, z verjetnostjo okoli 0.120,
2. T, A, O, I, N, S, H, R, vse z verjetnostjo med 0.06 in 0.09,
3. D, L, obe z verjetnostjo okoli 0.04,
4. C, U, M, W, F, G, Y, P, B, vse z verjetnostjo med 0.015 in 0.028,
5. V, K, J, X, Q, Z, vse z verjetnostjo manjšo od 0.01.

Najbolj pogosti pari so (v padajočem zaporedju): TH, HE, IN, ER, AN, RE, ED, ON, ES, ST, EN, AT, TO, NT, HA, ND, OU, EA, NG, AS, OR, TI, IS, ET, IT, AR, TE, SE, HI in OF,

Najbolj pogoste trojice pa so (v padajočem zaporedju): THE, ING, AND, HER, ERE, ENT, THA, NTH, WAS, ETH, FOR in DTH.

Klasične šifre

Transpozicijska šifra

V transpozicijski šifri ostanejo črke originalnega sporočila nespremenjene, njihova mesta pa so pomešana na kakšen sistematičen način (primer: permutacija stolpcev).

Te šifre zlahka prepoznamo, če izračunamo gostoto samoglasnikov (v angleščini je ta 40%, in skoraj nikoli ne pade zunaj intervala 35%–45%).

Težko jih rešimo, vendar pa se potrpljenje na koncu običajno izplača.

Simetrična šifra *je peterica* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ *za katero velja:*

1. $\mathcal{P}$ *je končna množica možnih čistopisov*
2. $\mathcal{C}$ *je končna množica možnih tajnopisov*
3. $\mathcal{K}$ *je končna množica možnih ključev.*
4. *Za vsak ključ* $K \in \mathcal{K}$, *imamo šifrirni postopek* $e_K \in \mathcal{E}$ *in ustrezen odšifrirni postopek* $d_K \in \mathcal{D}$.
$$e_K : \mathcal{P} \longrightarrow \mathcal{C} \quad \text{in} \quad d_K : \mathcal{C} \longrightarrow \mathcal{P}$$
sta taki funkciji, da je $d_K(e_K(x)) = x$ *za vsak* $x \in \mathcal{P}$.

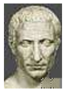
Pomična šifra (angl. shift cipher) je poseben primer zamenjalne šifre.

```
wewillmeetatmidnight

 22  4 22  8 11 11 12  4  4 19  0 19 12  8  3 13  8  6  7 19
  7 15  7 19 22 22 23 15 15  4 11  4 23 19 14 24 19 17 18  4

HPHTWWXPPELEXTOYTRSE
```

Cezarjeva šifra zašifrira njegovo ime v Ehbčt.



Cezar ukazal napad

Ehbčt žnčbčo rčščg

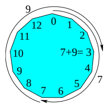
V kriptografiji si na splošno radi omislimo končne množice, kot pri številčnici na uri (npr. praštevilske obsege $\mathbb{Z}_p$).

Kongruence: naj bosta a in b celi števili in m naravno število.

$$a \equiv b \pmod{m} \iff m | b - a.$$

Primer: za $p = 13$ velja
$7 +_{13} 9 = 7 + 9 \mod 13 = 3$ in
$5 *_{13} 4 = 5 * 4 \mod 13 = 7$

(saj ima pri deljenju s 13 vsota 16 ostanek 3, produkt 20 pa ostanek 7), možno pa je tudi deljenje.



Deljenje v primeru $p = 13$:

$*_{13}$	1	2	3	4	5	6	7	8	9	10	11	12
1	**1**	2	3	4	5	6	7	8	9	10	11	12
2	2	4	6	8	10	12	**1**	3	5	7	9	11
3	3	6	9	12	2	5	8	11	**1**	4	7	10
4	4	8	12	3	7	11	2	6	10	**1**	5	9
5	5	10	2	7	12	4	9	**1**	6	11	3	8
6	6	12	5	11	4	10	3	9	2	8	**1**	7
7	7	**1**	8	2	9	3	10	4	11	5	12	6
8	8	3	11	6	**1**	9	4	12	7	2	10	5
9	9	5	**1**	10	6	2	11	7	3	12	8	4
10	10	7	4	**1**	11	8	5	2	12	9	6	3
11	11	9	7	5	3	**1**	12	10	8	6	4	2
12	12	11	10	9	8	7	6	5	4	3	2	**1**

Afina šifra:

$$e(x) = ax + b \pmod{26} \qquad \text{za } a, b \in \mathbb{Z}_{26}$$

Za $a = 1$ dobimo pomično šifro.
Funkcija je injektivna, če in samo če je $D(a, 26) = 1$.

Imamo $|\mathcal{K}| = 12 \times 26 = 312$ možnih ključev.

Za pomično šifro in afino šifro pravimo, da sta **monoabecedni**, ker preslikamo vsako črko v natanko določeno črko.

Vigenèrejeva šifra (1586):

Naj bo $m \in \mathbb{N}$ in

$$\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}_{26})^m.$$

Za ključ $K = (k_1, k_2, \ldots, k_m)$ definiramo

$$e(x_1, \ldots, x_m) = (x_1 + k_1, \ldots, x_m + k_m) \text{ in}$$
$$d(y_1, \ldots, y_m) = (y_1 - k_1, \ldots, y_m - k_m),$$

kjer sta operaciji "+" in "−" opravljeni po modulu 26.


```
    A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A   A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B   B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
C   C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
D   D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
E   E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
F   F G H I J K L M N O P Q R S T U V W X Y Z A B C D E
G   G H I J K L M N O P Q R S T U V W X Y Z A B C D E F
H   H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
I   I J K L M N O P Q R S T U V W X Y Z A B C D E F G H
J   J K L M N O P Q R S T U V W X Y Z A B C D E F G H I
K   K L M N O P Q R S T U V W X Y Z A B C D E F G H I J
L   L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
M   M N O P Q R S T U V W X Y Z A B C D E F G H I J K L
N   N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
O   O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
P   P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
Q   Q R S T U V W X Y Z A B C D E F G H I J K L M N O P
R   R S T U V W X Y Z A B C D E F G H I J K L M N O P Q
S   S T U V W X Y Z A B C D E F G H I J K L M N O P Q R
T   T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
U   U V W X Y Z A B C D E F G H I J K L M N O P Q R S T
V   V W X Y Z A B C D E F G H I J K L M N O P Q R S T U
W   W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
X   X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
Y   Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
Z   Z A B C D E F G H I J K L M N O P Q R S T U V W X Y
```


Sporočilo

`TO BE OR NOT TO BE THAT IS THE QUESTION`

zašifriramo s ključem `RELATIONS`:

```
ključ:      RELAT IONSR ELATI ONSRE LATIO NSREL
čistopis:   TOBEO RNOTT OBETH ATIST HEQUE STION
tajnopis:   KSMEH ZBBLK SMEMP OGAJX SEJCS FLZSY
```

Npr. prvo črko tajnopisa dobimo tako, da pogledamo v tabelo na mesto (`R`, `T`).

Kako pa najdemo iz `T` in `K` nazaj `R`?

To ni monoabecedna šifra.

Pravimo ji **poliabecedna šifra**.

Vigenèrejeva šifra in 26^m možnih ključev.

Za $m = 5$ je število 1.1×10^7 že preveliko, da bi "peš" iskali pravi ključ.

Hillova šifra (1929)

Naj bo m neko naravno število in naj bo

$$\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m.$$

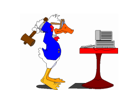
Za K vzemimo obrnljivo $m \times m$ matriko in definirajmo

$$e_K(x) = xK \qquad \text{in} \qquad d_K(y) = yK^{-1},$$

pri čemer so vse operacije opravljene v $\mathbb{Z}_{26}$.

Ponovimo:

Odšifriranje (razbijanje) klasičnih šifer



Kriptografske sisteme kontroliramo s pomočjo ključev, ki določijo transformacijo podatkov.
Seveda imajo tudi ključi digitalno obliko (binarno zaporedje: 01001101010101...).

Držali se bomo **Kerckhoffovega principa**, ki pravi, da "nasprotnik"

pozna kriptosistem oziroma algoritme, ki jih uporabljamo, ne pa tudi ključe, ki nam zagotavljajo varnost.

Ločimo naslednje nivoje napadov na kriptosisteme:

1. **samo tajnopis**: *nasprotnik ima del tajnopisa,*
2. **poznani čistopis**: *nasprotnik ima del čistopisa ter ustrezen tajnopis,*
3. **izbrani čistopis**: *nasprotnik ima začasno na voljo šifrirno mašinerijo ter za izbrani* $x \in \mathcal{P}$ *konstruira* $e(x)$,
4. **izbrani tajnopis**: *nasprotnik ima začasno na voljo odšifrirno mašinerijo ter za izbrani* $y \in \mathcal{C}$ *konstruira* $d(y)$.

Odšifriranje Vigenèrejeve šifre

Test Friedericha Kasiskega (1863):

(in Charles Babbage-a 1854)

poiščemo dele tajnopisa $\boldsymbol{y} = y_1y_2\dots y_n$, ki so identični in zabeležimo razdalje $d_1, d_2, \dots$ med njihovimi začetki. Predpostavimo, da iskani m deli največji skupni delitelj teh števil.

Naj bo $d = n/m$. Elemente tajnopisa $\boldsymbol{y}$ zapišemo po stolpcih v $(m \times d)$-razsežno matriko. Vrstice označimo z $\boldsymbol{y_i}$, tj.

$$\boldsymbol{y_i} = y_i\, y_{m+i}\, y_{2m+i} \dots$$

Indeks naključja (William Friedman, 1920):

Za zaporedje $\boldsymbol{x} = x_1x_2\dots x_d$ je **indeks naključja** (angl. index of coincidence, oznaka $I_c(\boldsymbol{x})$) **verjetnost**, da sta naključno izbrana elementa zaporedja $\boldsymbol{x}$ enaka.

Če so $f_0, f_1, \dots, f_{25}$ frekvence črk $A, B, \dots, Z$ v zaporedju $\boldsymbol{x}$, je

$$I_c(\boldsymbol{x}) = \frac{\sum_{i=0}^{25}\binom{f_i}{2}}{\binom{d}{2}} = \sum_{i=0}^{25}\frac{f_i(f_i-1)}{d(d-1)}.$$

Če so p_i pričakovane verjetnosti angleških črk, potem je

$$I_c(\boldsymbol{x}) \approx \sum_{i=0}^{25} p_i^2 = 0.065.$$

Za povsem naključno zaporedje velja

$$I_c(\boldsymbol{x}) \approx 26\left(\frac{1}{26}\right)^2 = \frac{1}{26} = 0.038.$$

Ker sta števili .065 in .038 dovolj narazen, lahko s to metodo najdemo dolžino ključa

(ali pa potrdimo dolžino, ki smo jo uganili s testom Kasiskega).

Za podzaporedje $\boldsymbol{y_i}$ in $0 \le g \le 25$ naj bo

$$M_g(\boldsymbol{y_i}) = \sum_{i=0}^{25} p_i \frac{f_{i+g}}{d}.$$

Če je $g = k_i$, potem pričakujemo

$$M_g(\boldsymbol{y_i}) \approx \sum_{i=0}^{25} p_i^2 = 0.065$$

Za $g \neq k_i$ je običajno M_g bistveno manjši od 0.065.

Torej za vsak $1 \le i \le m$ in $0 \le g \le 25$ tabeliramo vrednosti M_g, nato pa v tabeli za vsak $1 \le i \le m$ poiščemo tiste vrednosti, ki so blizu 0.065.

Ustrezni g-ji nam dajo iskane zamike $k_1, k_2, \dots, k_m$.

Odšifriranje Hillove šifre

Predpostavimo, da je nasprotnik določil m, ki ga uporabljamo, ter se dokopal do m različnih parov m-teric (2. stopnja – poznan čistopis):

$$x_j = (x_{1,j}, x_{2,j}, ..., x_{m,j}),\ \ y_j = (y_{1,j}, y_{2,j}, ..., y_{m,j}),$$

tako da je $y_j = e_K(x_j)$ za $1 \le j \le m$.

Za matriki $X = (x_{i,j})$ in $Y = (y_{i,j})$ dobimo matrično enačbo $Y = XK$.

Če je matrika X obrnljiva, je $K = YX^{-1}$.

Za Hillovo šifro lahko uporabimo tudi 1. stopnjo napada (samo tajnopis), glej nalogo 1.25.

Koliko ključev imamo na voljo v primeru Hillove šifre? Glej nalogo 1.12.

Za afino-Hillovo šifro glej nalogo 1.24.

Tokovne šifre

Naj bo $x_1x_2\dots$ čistopis.

Doslej smo obravnavali kriptosisteme z enim samim ključem in tajnopis je imel naslednjo obliko.

$$\mathbf{y} = y_1y_2\cdots = e_K(x_1)e_K(x_2)\dots$$

Taki šifri pravimo **bločna šifra** (angl. block cipher).

Posplošitev: iz enega ključa $K \in \mathcal{K}$ napravimo zaporedje (tok) ključev. Naj bo f_i funkcija, ki generira i-ti ključ:

$$z_i = f_i(K, x_1, \dots, x_{i-1}).$$

Z njim izračunamo:

$$y_i = e_{z_i}(x_i) \qquad \text{in} \qquad x_i = d_{z_i}(y_i).$$

Bločna šifra je poseben primer tokovne šifre (kjer je $z_i = K$ za vse $i \geq 1$).

Sinhrona tokovna šifra *je sedmerica* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{L}, \mathcal{F}, \mathcal{E}, \mathcal{D})$ *za katero velja:*

1. $\mathcal{P}$ *je končna množica možnih* **čistopisov**,
2. $\mathcal{C}$ *je končna množica možnih* **tajnopisov**,
3. $\mathcal{K}$ *je končna množica možnih* **ključev**,
4. $\mathcal{L}$ *je končna množica tokovne* **abecede**,
5. $\mathcal{F} = (f_1, f_2, \dots)$ *je generator toka ključev:*
$$f_i : \mathcal{K} \times \mathcal{P}^{i-1} \longrightarrow \mathcal{L} \quad \text{za } i \geq 1$$
6. *Za vsak ključ* $z \in \mathcal{L}$ *imamo šifrirni* ($e_z \in \mathcal{E}$) *in odšifrirni* ($d_z \in \mathcal{D}$) *postopek, tako da je* $d_z(e_z(x)) = x$ *za vsak* $x \in \mathcal{P}$.

Za šifriranje čistopisa $x_1x_2\dots$ zaporedno računamo

$$z_1, y_1, z_2, y_2, \dots,$$

za odšifriranje tajnopisa $y_1y_2\dots$ pa zaporedno računamo

$$z_1, x_1, z_2, x_2, \dots.$$

Tokovna šifra je **periodična** s periodo d kadar, je $z_{i+d} = z_i$ za vsak $i \geq 1$

(poseben primer: Vigenèrejeva šifra).

Začnimo s ključi $(k_1, \dots, k_m)$ in naj bo $z_i = k_i$ za $i = 1, \dots, m$.
Definiramo linearno rekurzijo stopnje m:

$$z_{i+m} \;=\; z_i \;+\; \sum_{j=1}^{m-1} c_j\, z_{i+j} \mod 2,$$

kjer so $c_1, \dots, c_{m-1} \in \mathbb{Z}_2$ vnaprej določene konstante.

Za ustrezno izbiro konstant $c_1, \dots, c_{m-1} \in \mathbb{Z}_2$ in neničelen vektor $(k_1, \dots, k_m)$ lahko dobimo tokovno šifro s periodo $2^m - 1$.

Hitro lahko generiramo tok ključev z uporabo **LFSR** (**Linear Feedback Shift Register**).

V pomičnem registru začnemo z vektorjem

$$(k_1, \dots, k_m).$$

Nato na vsakem koraku naredimo naslednje:

1. k_1 dodamo toku ključev (za XOR),
2. $k_2, \dots, k_m$ pomaknemo za eno v levo,
3. 'nov' ključ k_m izračunamo z
$$\sum_{j=0}^{m-1} c_j\, k_{j+1} \quad \text{(to je “linear feedback”)}.$$

Primer:

$c_0 = 1$, $c_1 = 1$, $c_2 = 0$, $c_3 = 0$,

torej je $k_{i+4} = k_i + k_{i+1}$.

Izberimo $k_0 = 1$, $k_1 = 0$, $k_2 = 1$, $k_3 = 0$.

Potem je $k_4 = 1$, $k_5 = 1$, $k_6 = 0$, $\dots$.

Naj bo $\boldsymbol{k} = (k_0, k_1, k_2, k_3)^t$ in

$$A := \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}.$$

Torej je $A(\boldsymbol{k}) = (k_1, k_2, k_3, k_4)^t$,

$A^2(\boldsymbol{k}) = A(k_1, k_2, k_3, k_4)^t = (k_2, k_3, k_4, k_5)^t$
...
$A^i(\boldsymbol{k}) = (k_i, k_{i+1}, k_{i+2}, k_{i+3})^t$.

Najdaljša možna perioda je 15.

Enkrat dobimo:

$$A^i(\boldsymbol{k}) = A^j(\boldsymbol{k})$$

in ker je A obrnljiva

$$A^{i-j}(\boldsymbol{k}) = \boldsymbol{k}$$

Karakteristični polinom matrike A je

$$f(x) = 1 + x + x^4.$$

Ker je $f(x)$ nerazcepen, je $f(x)$ tudi minimalni polinom matrike A.

Red matrike A je najmanjše naravno število s, tako da je $A^s = I$. Naj bo e najmanjše naravno število, tako da $f(x) \,|\, (x^e - 1)$. Potem je $e = s$.

$$1 + x^{15} = (x+1)(x^2+x+1)(x^4+x+1)(x^4+x^3+1)(x^4+x^3+x^2+x+1).$$

Splošno: če hočemo, da nam rekurzija stopnje m da periodo $2^m - 1$, potem si izberemo nerazcepen f.

Analiza je neodvisna od začetnega neničelnega vektorja.

Kriptoanaliza LFSR tokovne šifre:
uporabimo lahko poznan čistopis, glej nalogo 1.27.

2. poglavje

Shannonova teorija

- Popolna varnost
- Entropija
- Lastnosti entropije
- Ponarejeni ključi
  in enotska razdalja
- Produktne šifre

Popolna varnost

Omenimo nekaj osnovnih principov za študij varnosti nekega kriptosistema:

- **računska varnost**,
- **brezpogojna varnost**,
- **dokazljiva varnost**.

Kriptosistem je **računsko varen**, če tudi najboljši algoritem za njegovo razbitje potrebuje vsaj N operacij, kjer je N neko konkretno in zelo veliko število.

Napadalec (Oskar) ima na razpolago 18 Crayev, 4000 Pentium PC-jev in 200 DEC Alpha mašin (Oskar je "računsko omejen").

Kriptosistem je **dokazljivo varen** (angl. provable secure), če lahko pokažemo, da se njegova varnost zreducira na varnost kriptosistema, ki je zasnovan na dobro preštudiranem problemu.

Ne gre torej za absolutno varnost temveč *relativno varnost.*

Gre za podobno strategijo kot pri dokazovanju, da je določen problem *NP-poln* (v tem primeru dokažemo, da je dani problem vsaj tako težak kot nekdrugi znani NP-poln problem, ne pokažemo pa, da je absolutno računsko zahteven).

Kriptosistem je **brezpogojno varen**, kadar ga napadalec ne more razbiti, tudi če ima na voljo neomejeno računsko moč.

Seveda je potrebno povedati tudi, kakšne vrste napad imamo v mislih. Spomnimo se, da zamične, substitucijske in Vigenère šifre niso varne pred napadom s poznanim tajnopisom (če imamo na voljo dovolj tajnopisa).

Razvili bomo teorijo kriptosistemov, ki so brezpogojno varni pri napadu s poznanim tajnopisom. Izkaže se, da so vse tri šifre brezpogojno varne, kadar zašifriramo le en sam element čistopisa.

Glede na to, da imamo pri brezpogojni varnosti na voljo neomejeno računsko moč, je ne moremo študirati s pomočjo teorije kompleksnosti, temveč s teorijo verjetnosti.

Naj bosta X in Y slučajni spremenljivki,
naj bo $p(x) := P(X = x)$, $p(y) := P(Y = y)$ in
$p(x \cap y) := P((X = x) \cap (Y = y))$ produkt dogodkov.

Slučajni spremenljivki X in Y sta **neodvisni**, če in samo, če je $p(x \cap y) = p(x)p(y)$ za vsak $x \in X$ in $y \in Y$.

Omenimo še zvezo med pogojno verjetnostjo in pa verjetnostjo produkta dveh dogodkov oziroma **Bayesov izrek o pogojni verjetnosti**:

$$p(x \cap y) = p(x/y)p(y) = p(y/x)p(x),$$

iz katerega sledi, da sta slučajni spremenljivki X in Y neodvisni, če in samo, če je $p(x/y) = p(x)$ za vsak x in y.

Privzemimo, da vsak ključ uporabimo za največ eno šifriranje, da si Anita in Bojan izbereta ključ K z neko fiksno verjetnostno porazdelitvijo $p_{\mathcal{K}}(K)$ (pogosto enakomerno porazdelitvijo, ni pa ta nujna) in naj bo $p_{\mathcal{P}}(x)$ verjetnost čistopisa x.

Končno, predpostavimo, da sta izbira čistopisa in ključa neodvisna dogodka.

Porazdelitvi $\mathcal{P}$ in $\mathcal{K}$ inducirata verjetnostno porazdelitev na $\mathcal{C}$. Za množico vseh tajnopisov za ključ K

$$C(K) = \{e_K(x) \,|\, x \in \mathcal{P}\}$$

velja

$$p_{\mathcal{C}}(y) = \sum_{\{K \,|\, y \in C(K)\}} p_{\mathcal{K}}(K)\, p_{\mathcal{P}}(d_k(y))$$

in

$$P(Y = y / X = x) = \sum_{\{K \,|\, x = d_K(y)\}} p_{\mathcal{K}}(K).$$

Sedaj lahko izračunamo pogojno verjetnost $p_{\mathcal{P}}(x/y)$, tj. verjetnost, da je x čistopis, če je y tajnopis

$$P(X=x/Y=y)=\frac{p_{\mathcal{P}}(x)\times\displaystyle\sum_{\{K\,|\,x=d_K(y)\}}p_{\mathcal{K}}(K)}{\displaystyle\sum_{\{K\,|\,y\in C(K)\}}p_{\mathcal{K}}(K)\,p_{\mathcal{P}}(d_k(y))}$$

in opozorimo, da jo lahko izračuna vsakdo, ki pozna verjetnostni porazdelitvi $\mathcal{P}$ in $\mathcal{K}$.

Primer: $\mathcal{P}=\{a,b\}$ in $\mathcal{K}=\{K_1,K_2,K_3\}$:

$$p_{\mathcal{P}}(a)=1/4 \text{ in } p_{\mathcal{P}}(b)=3/4.$$

$$p_{\mathcal{K}}(K_1)=1/2 \text{ in } p_{\mathcal{K}}(K_2)=p_{\mathcal{K}}(K_3)=1/4.$$

Enkripcija pa je definirana z $e_{K_1}(a)=1$, $e_{K_1}(b)=2$; $e_{K_2}(a)=2$, $e_{K_2}(b)=3$; $e_{K_3}(a)=3$, $e_{K_3}(b)=4$.

Potem velja

$$p_{\mathcal{C}}(1)=\frac{1}{8},\quad p_{\mathcal{C}}(2)=\frac{7}{16},\quad p_{\mathcal{C}}(3)=\frac{1}{4},\quad p_{\mathcal{C}}(3)=\frac{3}{16}.$$

$$p_{\mathcal{P}}(a/1)=1,\ p_{\mathcal{P}}(a/2)=\frac{1}{7},\ p_{\mathcal{P}}(a/3)=\frac{1}{4},\ p_{\mathcal{P}}(a/4)=0.$$

Šifra $(\mathcal{P},\mathcal{K},\mathcal{C})$ je **popolnoma varna**, če je

$$P(X=x/Y=y)=p_{\mathcal{P}}(x)\quad \text{za vse}\ \ x\in\mathcal{P}\ \text{ in }\ y\in\mathcal{C},$$

tj. "končna" verjetnost, da smo začeli s tajnopisom x pri danem čistopisu y, je identična z "začetno" verjetnostjo čistopisa x.

V prejšnjem primeru je ta pogoj zadoščen samo v primeru $y=3$, ne pa tudi v preostalih treh.

Izrek 1. *Če ima vseh 26 ključev pri zamični šifri enako verjetnost $1/26$, potem je za vsako verjetnostno porazdelitev čistopisa zamična šifra popolnoma varna.*

Dokaz: $\mathcal{P}=\mathcal{C}=\mathcal{K}=\mathbb{Z}_{26}$, $e_K(x)=x+K \bmod 26$:

$$p_{\mathcal{C}}(y)=\frac{1}{26}\sum_{K\in\mathbb{Z}_{26}}p_{\mathcal{P}}(y-K)=\frac{1}{26},$$

$$P(Y=y/X=x)=p_{\mathcal{K}}(y-x \bmod 26))=\frac{1}{26}.\quad \blacksquare$$

Torej lahko zaključimo, da zamične šifre ne moremo razbiti, če za vsak znak čistopisa uporabimo nov, naključno izbran ključ.

Sedaj pa preučimo popolno varnost na splošno. Pogoj

$$P(X=x/Y=y)=p_{\mathcal{P}}(x)\quad \text{za vse}\ \ x\in\mathcal{P}\ \text{ in }\ y\in\mathcal{C}$$

je ekvivalenten pogoju

$$P(Y=y/X=x)=p_{\mathcal{C}}(y)\quad \text{za vse}\ \ x\in\mathcal{P}\ \text{ in }\ y\in\mathcal{C}.$$

Privzemimo (BŠS), da je $p_{\mathcal{C}}(y)>0$ za vse $y\in\mathcal{C}$. Ker je $P(Y=y/X=x)=p_{\mathcal{C}}(y)>0$ za fiksen $x\in\mathcal{P}$ in za vsak $y\in\mathcal{C}$, za vsak tajnopis $y\in\mathcal{C}$ obstaja vsaj en ključ K, da je $e_K(x)=y$ in zato velja $|\mathcal{K}|\geq|\mathcal{C}|$.

Za vsako simetrično šifro velja $|\mathcal{C}|\geq|\mathcal{P}|$, saj smo privzeli, da je šifriranje injektivno.

V primeru enakosti (v obeh neenakostih) je Shannon karakteriziral popolno varnost na naslednji način:

Izrek 2. *Naj bo $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$ simetrična šifra za katero velja $|\mathcal{K}|=|\mathcal{C}|=|\mathcal{P}|$. Potem je le-ta popolnoma varna, če in samo, če je vsak ključ uporabljen z enako verjetnostjo $1/|\mathcal{K}|$ ter za vsak čistopis x in za vsak tajnopis y obstaja tak ključ K, da je $e_K(x)=y$.*

Dokaz: ($\Longrightarrow$) Ker je $|\mathcal{K}|=|\mathcal{C}|$, sledi, da za vsak čistopis $x\in\mathcal{P}$ in za vsak tajnopis $y\in\mathcal{C}$ obstaja tak ključ K, da je $e_K(x)=y$.

Naj bo $n=|\mathcal{K}|$, $\mathcal{P}=\{x_i\,|\,1\leq i\leq n\}$ in naj za fiksen tajnopis y označimo ključe iz $\mathcal{K}$ tako, da je $e_{K_i}(x_i)=y$ za $i\in[1..n]$. Po Bayesovem izreku velja

$$P(X=x_i/Y=y)=\frac{P(Y=y/x=x_i)\,p_{\mathcal{P}}(x_i)}{p_{\mathcal{C}}(y)}=\frac{p_{\mathcal{K}}(K_i)\,p_{\mathcal{P}}(x_i)}{p_{\mathcal{C}}(y)}.$$

Če je šifra popolnoma varna, velja $P(X=x_i/Y=y)=p_{\mathcal{P}}(x_i)$, torej tudi $p_{\mathcal{K}}(K_i)=p_{\mathcal{C}}(y)$, kar pomeni, da je vsak ključ uporabljen z enako verjetnostjo $p_{\mathcal{C}}(y)$ in zato $p_{\mathcal{K}}(K)=1/|\mathcal{K}|$.

Dokaz obrata poteka na podoben način kot v prejšnjem izreku. $\blacksquare$

Najbolj znana realizacija popolne varnosti je **Vernamov enkratni ščit**, ki ga je leta 1917 patentiral Gilbert Vernam za avtomatizirano šifriranje in odšifriranje telegrafskih sporočil.

Naj bo $\mathcal{P}=\mathcal{C}=\mathcal{K}=(\mathbb{Z}_2)^n$, $n\in\mathbb{N}$,

$$e_K(x)=x\ \mathrm{XOR}\ K,$$

odšifriranje pa je identično šifriranju.

Shannon je prvi po 30-ih letih dokazal, da ta sistem res ne moremo razbiti.

Slabi strani te šifre sta $|\mathcal{K}|\geq|\mathcal{P}|$ in dejstvo, da moramo po vsaki uporabi zamenjati ključ.

Entropija

Doslej nas je zanimala popolna varnost in smo se omejili na primer, kjer uporabimo nov ključ za vsako šifriranje.

Sedaj pa nas zanimata šifriranje vse več in več čistopisa z istim ključem ter verjetnost uspešnega napada z danim tajnopisom in neomejenim časom.

Leta 1948 je Shannon vpeljal v teorijo informacij *entropijo*, tj. matematično mero za informacije oziroma negotovosti in jo izrazil kot funkcijo verjetnostne porazdelitve.

Naj bo X slučajna spremenljivka s končno zalogo vrednosti in porazdelitvijo $p(X)$.

Kakšno informacijo smo pridobili, ko se je zgodil dogodek glede na porazdelitev $p(X)$

oziroma ekvivalentno,

če se dogodek še ni zgodil, kolikšna je negotovost izida?

To količino bomo imenovali **entropija** spremenljivke X in jo označili s $H(X)$.

Primer: metanje kovanca, $p(\text{cifra}) = p(\text{grb}) = 1/2$.

Smiselno je reči, da je entropija enega meta en bit.

Podobno je entropija n-tih metov n, saj lahko rezultat zapišemo z n biti.

Še en primer: slučajna spremenljivka X

$$\begin{pmatrix} x_1 & x_2 & x_3 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}.$$

Najbolj učinkovito zakodiranje izidov je x_1 z 0, x_2 z 10 in x_3 z 11, povprečje pa je

$$\frac{1}{2}\times 1 + \frac{1}{4}\times 2 + \frac{1}{4}\times 2 \;=\; 3/2.$$

Vsak dogodek, ki se zgodi z verjetnostjo 2^{-n}, lahko zakodiramo z n biti.

Posplošitev: dogodek, ki se zgodi z verjetnostjo p, lahko zakodiramo s približno $-\log_2 p$ biti.

Naj bo X slučajna spremenljivka s končno zalogo vrednosti in porazdelitvijo

$$p(X) = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ p_1 & p_2 & \dots & p_n \end{pmatrix}.$$

Potem **entropijo porazdelitve** $p(X)$ definiramo s

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i = -\sum_{i=1}^{n} p(X = x_i) \log_2 p(X = x_i).$$

Za $p_i = 0$ količina $\log_2 p_i$ ni definirana, zato seštevamo samo po neničelnih p_i (tudi $\lim_{x\to 0} x \log_2 x = 0$).

Lahko bi izbrali drugo logaritemsko bazo, a bi se entropija spremenila le za konstantni faktor.

Če je $p_i = 1/n$ za $1 \le i \le n$, potem je $H(X) = \log_2 n$.

Velja $H(X) \ge 0$, enačaj pa velja, če in samo, če je $p_i = 1$ za nek i in $p_j = 0$ za $j \ne i$.

Sedaj pa bomo študirali entropijo različnih komponent simetrične šifre: $H(K)$, $H(P)$, $H(C)$.

Za primer $\mathcal{P} = \{a, b\}$ in $\mathcal{K} = \{K_1, K_2, K_3\}$:

$$p_{\mathcal{P}}(a) = 1/4 \;\text{ in }\; p_{\mathcal{P}}(b) = 3/4.$$

$$p_{\mathcal{K}}(K_1) = 1/2 \;\text{ in }\; p_{\mathcal{K}}(K_2) = p_{\mathcal{K}}(K_3) = 1/4$$

izračunamo

$$H(P) = -\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4} = 2 - \frac{3}{4}\log_2 3 \approx .81\ .$$

in podobno $H(K) = 1.5$ ter $H(C) \approx 1.85$.

Lastnosti entropije

Realna funkcija f je **(striktno) konkavna** na intervalu I, če za vse (različne) $x, y \in I$ velja

$$f\left(\frac{x+y}{2}\right) \;(>)\; \ge \;\frac{f(x)+f(y)}{2}.$$

Jensenova neenakost: če je f zvezna in striktno konkavna funkcija na intervalu I in $\sum_{i=1}^{n} a_i = 1$ za $a_i > 0$, $1 \le i \le n$, potem je

$$f\left(\sum_{i=1}^{n} a_i x_i\right) \ge \sum_{i=1}^{n} a_i f(x_i),$$

enakost pa velja, če in samo, če je $x_1 = x_2 = \cdots = x_n$.

Izrek 3. *$H(X) \le \log_2 n$, enakost pa velja, če in samo, če je $p_1 = p_2 = \cdots = p_n = 1/n$.*

Izrek 4. *$H(X, Y) \le H(X) + H(Y)$, enakost pa velja, če in samo, če sta X in Y neodvisni spremenljivki.*

Dokaz izreka 4: Naj bo

$$p(X) = \begin{pmatrix} x_1 & x_2 & \dots & x_m \\ p_1 & p_2 & \dots & p_m \end{pmatrix},\; p(Y) = \begin{pmatrix} y_1 & y_2 & \dots & y_n \\ q_1 & q_2 & \dots & q_n \end{pmatrix}$$

in $r_{ij} = p((X = x_i) \cap (Y = y_j))$ za $i \in [1..m]$, $j \in [1..n]$.

Potem za $i \in [1..m]$ in $j \in [1..n]$ velja

$$p_i = \sum_{j=1}^{n} r_{ij} \quad \text{in} \quad q_j = \sum_{i=1}^{m} r_{ij}$$

ter

$$H(X) + H(Y) = -\sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij} \log_2 p_i q_j$$

in

$$H(X,Y) - H(X) - H(Y) = \sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij} \log_2 \frac{p_i q_j}{r_{ij}}$$

$$(\text{Jensen}) \leq \log_2 \sum_{i=1}^{m}\sum_{j=1}^{n} p_i q_j = \log_2 1 = 0.$$

Enakost velja, če in samo, če je $p_i q_j / r_{ij} = c$ za $i \in [1..m]$ in $j \in [1..n]$.

Upoštevajmo še

$$\sum_{j=1}^{n}\sum_{i=1}^{m} r_{ij} = \sum_{j=1}^{n}\sum_{i=1}^{m} p_i q_j = 1$$

in dobimo $c = 1$ oziroma za vse i in j

$$p((X = x_i) \cap (Y = y_j)) = p(X = x_i)\, p(Y = y_j),$$

kar pomeni, da sta spremenljivki X in Y neodvisni. ■

Za slučajni spremenljivki X in Y definiramo **pogojni entropiji**

$$H(X/y) = -\sum_{x} p(x/y) \log_2 p(x/y)$$

in

$$H(X/Y) = -\sum_{y}\sum_{x} p(y) p(x/y) \log_2 p(x/y).$$

Le-ti merita povprečno informacijo spremenljivke X, ki jo odkrijeta y oziroma Y.

Izrek 5. $H(X,Y) = H(Y) + H(X/Y)$.

Dokaz: Po definiciji je $P(X = x_i / Y = y_j) = r_{ij}/q_j$ in $H(Y) + H(X/Y) =$

$$= -\sum_{j=1}^{n}\sum_{i=1}^{m} r_{ij} \log_2 q_j - \sum_{j=1}^{n}\sum_{i=1}^{m} q_j\, r_{ij}/q_j \, \log_2 r_{ij}/q_j \quad ■$$

Iz izrekov 4 in 5 sledi:

Posledica 6. $H(X/Y) \leq H(X)$, *enakost pa velja, če in samo, če sta* X *in* Y *neodvisni spremenljivki.*

Ponarejeni ključi in enotska razdalja

Pogojna verjetnost $H(K/C)$ meri, koliko informacije o ključu je odkrito s tajnopisom.

Izrek 7. *Naj bo* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ *simetrična šifra. Potem velja* $H(K/C) = H(K) + H(P) - H(C)$.

Dokaz: Velja $H(K,P,C) = H(C/(K,P)) + H(K,P)$. Ker ključ in čistopis natanko določata tajnopis, je $H(C/K,P) = 0$.

Ker sta P in K neodvisni spremenljivki, dobimo $H(K,P,C) = H(P) + H(K)$ in podobno tudi $H(K,P,C) = H(K,C)$ ter uporabimo še izrek 5. ■

Napadalec privzame, da je čistopis "naravni" jezik (npr. angleščina) in na ta način odpiše mnoge ključe. Vseeno pa lahko ostane še mnogo ključev (med katerimi je le en pravi), ki jih bomo, razen pravega ključa, imenovali **ponarejeni** (angl. spurious).

Naš cilj bo oceniti število ponarejenih ključev.

Naj bo H_L mera povprečne informacije na črko (angl. per letter) v "smiselnem" čistopisu (sledi bolj natančna definicija).

Če so vse črke enako verjetne, je

$$H_L = \log_2 26 \approx 4.70.$$

Kot aproksimacijo *prvega reda* bi lahko vzeli $H(P)$. V primeru angleškega jezika dobimo $H(P) \approx 4.19$.

Tudi zaporedne črke v jeziku niso neodvisne, njihove korelacije pa zmanjšajo entropijo. Za aproksimacijo *drugega reda* bi lahko izračunali entropijo porazdelitve parov črk in potem delili z dve, kajti H_L meri entropijo jezika L na črko.

V splošnem, naj bo P^n slučajna spremenljivka, katere verjetnostna porazdelitev je enaka verjetnostni porazdelitvi n-teric v čistopisu.

Potem je **entropija za naravni jezik** L definirana s

$$H_L = \lim_{n\to\infty} \frac{H(P^n)}{n},$$

odvečnost jezika L pa z

$$R_L = 1 - \frac{H_L}{\log_2 |\mathcal{P}|}.$$

H_L meri entropijo jezika L na črko.

Entropija naključnega jezika je $\log_2 |\mathcal{P}|$.

$R_L \in [0,1)$ meri kvocient "odvečnih znakov" in je 0 v primeru naključnega jezika.

Za angleški jezik je $H(P^2)/2 \approx 3.90$.
Empirični rezultati kažejo, da je $1.0 \leq H_L \leq 1.5$.
Če ocenimo H_L z 1.25, potem je $R_L \approx .75$, kar pomeni, da je angleščina 75% odvečna
(tj. tekst bi lahko zakodirali le z 1/4 prvotnega teksta).

Podobno kot P^n definiramo še C^n in za $y \in C^n$ še

$$K(y) = \{K \in \mathcal{K} \,|\, \exists x \in P^n,\, p_{\mathcal{P}^n}(x) > 0,\, e_K(x) = y\},$$

tj. $K(y)$ je množica ključev, za katere je y smiselno šifriranje čistopisa dolžine n,
tj. množica verjetnih ključev, za katere je y tajnopis.

Matematično upanje ponarejenih ključev je torej

$$\overline{s}_n = \sum_{y \in \mathcal{C}^n} p(y)(|K(y)| - 1) = \sum_{y \in \mathcal{C}^n} p(y)|K(y)| \; - \; 1.$$

Izrek 8. *Če je $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ šifra za katero je $|\mathcal{C}| = |\mathcal{P}|$ in so vsi ključi med seboj enakovredni, potem za tajnopis z n znaki (n je dovolj velik) in za matematično upanje ponarjenih ključev $\overline{s}_n$ velja*

$$\overline{s}_n \geq \frac{|\mathcal{K}|}{|\mathcal{P}|^{nR_L}} - 1.$$

Dokaz: Iz izreka 7 sledi

$$H(K/C^n) = H(K) + H(P^n) - H(C^n).$$

Poleg ocene $H(C^n) \leq n \log_2 |\mathcal{C}|$ velja za dovolj velike n tudi ocena $H(P^n) \approx nH_L = n(1 - R_L) \log_2 |\mathcal{P}|$.
Za $|\mathcal{C}| = |\mathcal{P}|$ dobimo

$$H(K/C^n) \geq H(K) - nR_L \log_2 |\mathcal{P}|.$$

Ocenjeno entropijo povežemo še s ponarejenimi ključi

$$H(K/C^n) = \sum_{y \in \mathcal{C}^n} p(y) H(K/y) \leq \sum_{y \in \mathcal{C}^n} p(y) \log_2 |K(y)|$$

$$\leq \log_2 \sum_{y \in \mathcal{C}^n} p(y)|K(y)| = \log_2(\overline{s}_n + 1). \quad \blacksquare$$

Desna stran neenakosti v zadnjem izreku gre z večanjem števila n eksponentno proti 0 (to ni limita, števila $|\mathcal{K}|$, $|\mathcal{P}|$ in R_L so fiksna, število $|\mathcal{K}|$ pa je običajno veliko v primerjavi s $|\mathcal{P}|^{R_L} > 1$).

Enotska razdalja simetrične šifre je tako število n, označeno z n_0, za katerega postane matematično upanje ponarejenih ključev nič, tj. povprečna dolžina tajnopisa, ki jo napadalec potrebuje za računanje ključa pri neomejenem času.

Velja $$n_0 \approx \frac{\log_2 |\mathcal{K}|}{R_L \log_2 |\mathcal{P}|}.$$

V primeru zamenjalnega tajnopisa sta $|\mathcal{P}| = 26$ in $|\mathcal{K}| = 26!$. Če vzamemo $R_L = .75$, potem je enotska razdalja

$$n_0 \approx \frac{88.4}{.75 \times 4.7} \approx 25 \,.$$

Produktne šifre

Še ena Shannonova ideja v članku iz leta 1949 igra danes pomembno vlogo, predvsem pri simetričnih šifrah.

Zanimali nas bodo šifre, za katere $\mathcal{C} = \mathcal{P}$,
tj. **endomorfne šifre**.

Naj bosta $S_i = (\mathcal{P}, \mathcal{P}, \mathcal{K}_i, \mathcal{E}_i, \mathcal{D}_i)$, $i = 1, 2$, endomorfni simetrični šifri. Potem je **produkt** sistemov S_1 in S_2, označen s $S_1 \times S_2$, definiran s

$$(\mathcal{P}, \mathcal{P}, \mathcal{K}_1 \times \mathcal{K}_2, \mathcal{E}, \mathcal{D})$$

ter

$$e_{(K_1,K_2)}(x) = e_{K_2}(e_{K_1}(x))$$

in

$$d_{(K_1,K_2)}(y) = d_{K_1}(e_{K_2}(y)).$$

Njegova verjetnostna porazdelitev pa naj bo

$$p_{\mathcal{K}}(K_1, K_2) = p_{\mathcal{K}_1}(K_1) \times p_{\mathcal{K}_2}(K_2),$$

tj. ključa K_1 in K_2 izberemo neodvisno.

Če sta M in S zaporedoma multiplikativni tajnopis in zamični tajnopis, potem je $M \times S$ afin tajnopis. Malce težje je pokazati, da je tudi tajnopis $S \times M$ afin tajnopis. Ta dva tajnopisa torej **komutirata**.

Vsi tajnopisi ne komutirajo, zato pa je produkt asociativna operacija:

$$(S_1 \times S_2) \times S_3 = S_1 \times (S_2 \times S_3).$$

Če je $(S \times S =)\ S^2 = S$, pravimo, da je sistem **idempotenten**.

Zamični, zamenjalni, afin, Hillov, Vigenèrov in permutacijski tajnopisi so vsi idempotentni.
Če simetrična šifra ni idempotentna, potem se zna zgoditi, da z njeno iteracijo za večkrat povečamo varnost. Na tem so zasnovani **DES** in mnoge druge simetrične šifre.

Če sta simetrični šifri S_1 in S_2 idempotentni in obenem še komutirata, potem se ni težko prepričati, da je tudi produkt $S_1 \times S_2$ idempotentna simetrična šifra.

3. poglavje

Simetrični kriptosistemi

- Bločne šifre, nekaj zgodovine, DES, AES
- Iterativne šifre, zmenjalno-permutacijske mreže
- Produktna šifra in Fiestelova šifra
- Opis šifer DES in AES
- Načini delovanja (ECB, CBC, CFB, OFB) in MAC
- Napadi in velika števila
- 3-DES, DESX in druge simetrične bločne šifre

Bločne šifre

Bločna šifra je simetrična šifra, ki razdeli čistopis na bloke fiksne dolžine (npr. 128 bitov), in šifrira vsak blok posamično (kontrast: *tekoča šifra* zašifrira čistopis po znakih – ponavadi celo po bitih).

Najmodernejše bločne šifre so **produktne šifre**, ki smo jih spoznali v prejšnjem poglavju: komponiranje več enostavnih operacij, katere (vsaka posebej) niso dovolj varne, z namenom, da povečamo varnost: *transpozicije, ekskluzivni ali (XOR), tabele, linearne transformacije, aritmetične operacije, modularno množenje, enostavne substitucije.*

Primeri bločnih produktnih šifer: DES, AES, IDEA.

Nekatere želene lastnosti bločnih šifer

Varnost:

- **razpršitev**: vsak bit tajnopisa naj bo odvisen od vseh bitov čistopisa.
- **zmeda**: zveza med ključem ter biti tajnopisa naj bo zapletena,
- **velikost ključev**: mora biti majhna, toda dovolj velika da prepreči požrešno iskanje ključa.

Učinkovitost

- hitro šifriranje in odšifriranje,
- enostavnost (za lažjo implementacijo in analizo),
- primernost za hardware ali software.

Kratka zgodovina bločnih šifer DES in AES

Konec 1960-ih: IBM – Feistelova šifra in LUCIFER.

1972: NBS (sedaj NIST) izbira simetrično šifro za zaščito računalniških podatkov.

1974: IBM razvije DES, 1975: NSA ga "popravi".

1977: DES sprejet kot US Federal Information Processing Standard (FIPS 46).

1981: DES sprejet kot US bančni standard (ANSI X3.92).

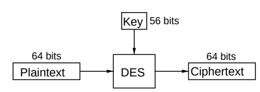
1997: AES (Advanced Encryption Standard) izbor

1999: izbranih 5 finalistov za AES

National Security Agency (NSA)

- www.nsa.gov
- ustanovljena leta 1952,
- neznana sredstva in število zaposlenih (čez 100.000?)
- Signals Intelligence (SIGINT): pridobiva tuje informacije.
- Information Systems Security (INFOSEC): ščiti vse občutljive (classified) informacije, ki jih hrani ali pošilja vlada ZDA,
- zelo vplivna pri določanju izvoznih regulacij ZDA za kriptografske produkte (še posebej šifriranje).

Data Encryption Standard (DES)



Ideja za DES je bila zasnovana pri IBM-u v 60-ih letih (uporabili so koncept Claude Shannona imenovan *Lucifer*).

NSA je zreducirala dolžino ključev s 128 bitov na 56.

V sredini 70-ih let je postal prvi komercijalni algoritem, ki je bil objavljen z vsemi podrobnostmi (FIPS 46-2).

Advanced Encryption Standard

AES je ime za nov FIPS-ov simetrični (bločni) kriptosistem, ki bo nadomestil DES.

Leta 2000 je zanj *National Institute of Standards and Technology (NIST)* izbral belgijsko bločno šifro **Rijndael**.

Dolžina *ključev* oziroma blokov je 128, 192 ali 256

Uporabljala pa ga tudi ameriška vlada, glej `http://csrc.nist.gov/encryption/aes/round2/r2report.pdf`.

Običajno uporabljamo **iterativne šifre**.

Tipični opis:

- krožna funkcija,
- razpored ključev,
- šifriranje skozi N_r podobnih krogov.

Naj bo K naključni binarni ključ določene dolžine.

K uporabimo za konstrukcijo podključev za vsak krog s pomočjo *javno* znanega algoritma.

Imenujemo jih **krožni ključi:** $K^1, \dots, K^{N_r}$.

Seznamu krožnih ključev $(K^1, \dots, K^{N_r})$ pa pravimo **razpored ključev**.

Krožna funkcija g ima dva argumenta:

(i) krožni ključ (K^r) in (ii) *tekoče stanje* (w^{r-1}).

Naslednje stanje je definirano z $w^r = g(w^{r-1}, K^r)$.

Začetno stanje, w_0, naj bo čistopis x.

Potem za tajnopis, y, vzamemo stanje po N_r krogih:

$$y = g(g(\dots g(g(x, K^1), K^2) \dots, K^{N_r-1})K^{N_r}).$$

Da je odšifriranje možno, mora biti funkcija g injektivna za vsak fiksen ključ K_i, tj. $\exists\, g^{-1}$, da je:

$$g^{-1}(g(w, K), K) = w, \qquad \text{za vse } w \text{ in } K.$$

Odšifriranje opravljeno po naslednjem postopku:

$$x = g^{-1}(g^{-1}(\dots g^{-1}(g^{-1}(y, K^{N_r}), K^{N_r-1}) \dots, K^2)K^1).$$

Zamenjalno-permutacijske mreže

(angl. *substitution-permutation network* – (**SPN**)).

Čistopis $\mathcal{P}$ in tajnopis $\mathcal{C}$ so binarni vektorji dolžine ℓm, $\ell, m \in \mathbb{N}$ (tj. ℓm je dolžina bloka).

SPN je zgrajen iz dveh komponent (zamenjave in permutacije):

$$\begin{aligned} \pi_S &: \{0,1\}^\ell \longrightarrow \{0,1\}^\ell, \\ \pi_P &: \{0, \dots, \ell m\} \longrightarrow \{0, \dots, \ell m\}. \end{aligned}$$

Permutacijo π_S imenujemo **S-škatla** in z njo zamenjamo ℓ bitov z drugimi ℓ biti.

Permutacija π_P pa permutira ℓm bitov.

Naj bo $x = (x_1, \dots, x_{\ell m})$ binarno zaporedje, ki ga lahko smatramo za spoj m ℓ-bitnih podzaporedij označenih z $x_{(1)}, \dots, x_{(m)}$.

SPN ima N_r krogov, v vsakem (razen zadnjem, ki je bistveno drugačen) opravimo m zamenjav z π_S in nato uporabimo še π_P. Pred vsako zamenjavo vključimo krožni ključ z XOR operacijo.

SPN šifra
$\ell, m, N_r \in \mathbb{N}$, π_S in π_P permutaciji, $\mathcal{P} = \mathcal{C} = \{0,1\}^{\ell m}$ in $\mathcal{K} \subseteq (\{0,1\}^{\ell m})^{N_r+1}$, ki se sestoji iz vseh možnih razporedov ključev izpeljanih iz ključa K z uporabo algoritma za generiranje razporeda kjučev.
Šifriramo z algoritmom SPN.

Alg. : *SPN*$(x, \pi_S, \pi_P, (K^1, \dots, K^{N_r+1}))$

$w^0 := x$

for $r := 1$ **to** $N_r - 1$ **do** (*krožno mešanje ključev*)

  $u^r := w^{r-1} \oplus K^r$

  for $i := 1$ **to** m **do** $v^r_{(i)} := \pi_S(u^r_{(i)})$

  $w^r := (v^r_{\pi_P(1)}, \dots, v^r_{\pi_P(\ell m)})$

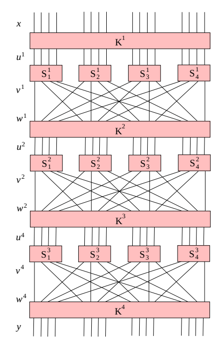
(*zadnji krog*)

$u^{N_r} := w^{N_r-1} \oplus K^{N_r}$

for $i := 1$ **to** m **do** $v^{N_r}_{(i)} := \pi_S(u^{N_r+1}_{(i)})$

$y := v^{N_r} \oplus K^{N_r+1}$

output (y)



Primer: naj bo $\ell = m = N_r = 4$, permutaciji π_S in π_P pa podani s tabelami:

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_S(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

ter

z	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi_P(z)$	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

Naj bo ključ $K = (k_1, \dots, k_{32}) \in \{0,1\}^{32}$ definiran z

$$K = \texttt{0011 1010 1001 0100 1101 0110 0011 1111},$$

sedaj pa izberimo še razpored ključev tako, da je za $1 \le r \le 5$, krožni ključ K^r izbran kot 16 zaporednih bitov ključa K z začetkom pri k_{4r-3}:

$$\begin{aligned} K^1 &= 0011\ 1010\ 1001\ 0100 \\ K^2 &= 1010\ 1001\ 0100\ 1101 \\ K^3 &= 1001\ 0100\ 1101\ 0110 \\ K^4 &= 0100\ 1101\ 0110\ 0011 \\ K^5 &= 1101\ 0110\ 0011\ 1111 \end{aligned}$$

Potem šifriranje čistopisa

$$x = 0010\ 0110\ 1011\ 0111$$

poteka v naslednjem vrstnem redu.

$$
\begin{array}{rlrl}
w^0 & = 0010\ 0110\ 1011\ 0111, & K^1 & = 0011\ 1010\ 1001\ 0100 \\
u^1 & = 0001\ 1100\ 0010\ 0011, & v^1 & = 0100\ 0101\ 1101\ 0001 \\
w^1 & = 0010\ 1110\ 0000\ 0111, & K^2 & = 1010\ 1001\ 0100\ 1101 \\
u^2 & = 1000\ 0111\ 0100\ 1010, & v^2 & = 0011\ 1000\ 0010\ 0110 \\
w^2 & = 0100\ 0001\ 1011\ 1000, & K^3 & = 1001\ 0100\ 1101\ 0110 \\
u^3 & = 1101\ 0101\ 0110\ 1110, & v^3 & = 1001\ 1111\ 1011\ 0000 \\
w^3 & = 1110\ 0100\ 0110\ 1110, & K^4 & = 0100\ 1101\ 0110\ 0011 \\
u^4 & = 1010\ 1001\ 0000\ 1101, & v^4 & = 0110\ 1010\ 1110\ 1001 \\
K^5 & = 1101\ 0110\ 0011\ 1111, & y & = 1011\ 1100\ 1101\ 0110
\end{array}
$$

Možno so številne varijacije SPN šifer.

Na primer, namesto ene S-škatle lahko uporabimo različne škatle. To lahko vidimo pri DES-u, ki uporabi 8 različnih škatel.

Zopet druga možnost je uporabiti obrnljive linearne transformacije, kot zamenjavo za permutacije ali pa samo dodatek. Tak primer je AES.

$\boxed{\text{Feistelova šifra}}$

Feistelova šifra: r krogov (rund)

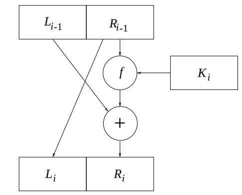
$$(L_{i-1}, R_{i-1}) \xrightarrow{K_i} (L_i, R_i).$$

kjer je $L_i = R_{i-1}$ in $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, in smo podključe K_i dobili iz osnovnega ključa K.

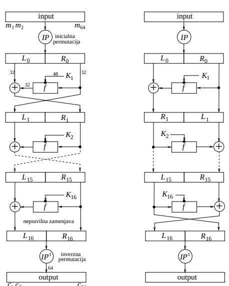
Končamo z (R_r, L_r) (in ne z (L_r, R_r)), zato je šifriranje enako odšifriranju, le da ključe uporabimo v obratnem vrstnem redu.

Funkcija f je lahko produktna šifra in ni nujno obrnljiva.

En krog



Opis šifre DES



DES-ove konstante

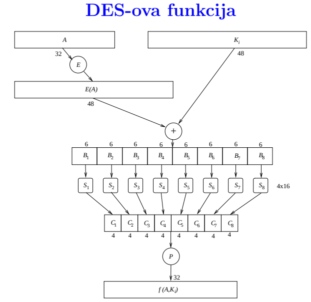
začetna in končna permutacija: IP, IP^{-1}

razširitev: E (nekatere bite ponovimo), permutacija P

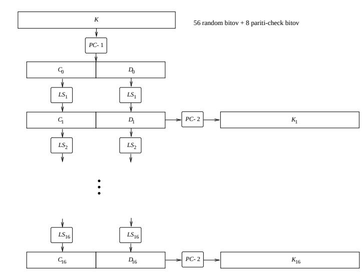
S-škatle: $S_1, S_2, \ldots, S_8$
(tabele: 4×16, z elementi $0 - 15$)

permutacije za gen. podključev: $PC - 1$, $PC - 2$

DES-ova funkcija



Računanje DES-ovih ključev



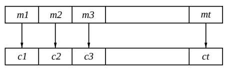
20 let je DES predstavljal delovnega konja kriptografije (bločnih šifer).

- do leta 1991 je NBS sprejel 45 hardwarskih implementacij za DES
- geslo (PIN) za bankomat (ATM)
- ZDA (Dept. of Energy, Justice Dept., Federal Reserve System)

Načini delovanja simetričnih šifer

– electronic codebook mode **(ECB)**
– cipher block chaining mode **(CBC)**
– cipher feedback mode **(CFB)**
– output feedback mode **(OFB)**

Pri **ECB** šifriramo zaporedoma blok po blok: $c = c_1, c_2, \ldots, c_t$, kjer je $c_i = E_k(m_i)$.

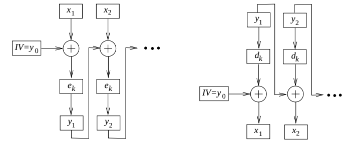


Odšifriranje: $m_i = D_k(c_i)$, $i = 1, 2, \ldots, t$.

Slabost: identični bloki čistopisa se (pri istem ključu) zašifrirajo v identične bloke tajnopisa.

Cipher Block Chaining mode – CBC

čistopis/tajnopis: 64 bitni bloki $x_1, x_2, \ldots / y_1, y_2, \ldots$
Šifriranje: $y_0 := \text{IV}$, $y_i := e_K(y_{i-1} \oplus x_i)$ za $i \geq 1$.

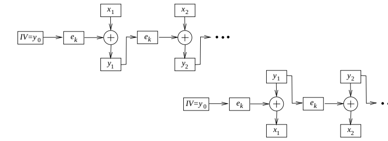


Odšifriranje: $y_0 := \text{IV}$, $x_i := y_{i-1} \oplus d_K(y_i)$ za $i \geq 1$.

Identičena čistopisa z različnimi IV dasta različen tajnopis. Eno-bitna napaka pri tajnopisu pokvari le odšifriranje dveh blokov.

Cipher Feedback mode – CFB

čistopis/tajnopis: 64 bitni bloki $x_1, x_2, \ldots / y_1, y_2, \ldots$
$y_0 := \text{IV}$, šifriranje: $z_i := e_K(y_{i-1})$, $y_i := y_{i-1} \oplus x_i$, $i \geq 1$.



Odšifriranje ($y_0 := \text{IV}$, $x_0 = d_K(\text{IV})$):

$$z_i := d_K(x_{i-1}) \text{ in } x_i := y_i \oplus z_i \quad \text{za } i \geq 1.$$

CFB se uporablja za preverjanje celovitosti sporočila (angl. message authentication code - MAC).

Output Feedback mode – OFB

čistopis/tajnopis: 64 bitni bloki $x_1, x_2, \ldots / y_1, y_2, \ldots$

Inicializacija: $z_0 := \text{IV}$, šifriranje:

$$z_i := e_K(z_{i-1}) \text{ in } y_i := x_i \oplus z_i \quad \text{za } i \geq 1.$$

Odšifriranje:($z_0 := \text{IV}$)

$$z_i := e_K(z_{i-1}) \text{ in } x_i := y_i \oplus z_i \quad \text{za } i \geq 1.$$

OFB se uporablja za satelitske prenose.

Napadi na šifro DES

Požrešni napad: preverimo vseh 2^{56} ključev.

Leta 1993 Michael J. Wiener, Bell-Northern Research, Kanada, predstavi učinkovito iskanje DES ključa:

- **diferenčna kriptoanaliza** z 2^{47} izbranimi čistopisi (Biham in Shamir 1989)
  – je učinkovita tudi na nekaterih drugih bločnih šifrah,
- **linearna kriptoanaliza** z 2^{47} poznanimi čistopisi (Matsui 1993):

Slednja napada sta statistična, saj potrebujeta velike količine čistopisa in ustreznega tajnopisa, da določita ključ. Pred leti sta bila napada zanimiva le teoretično.

Wienerjev cilj je bil precizna ocena časa in denarja potrebnega za graditev čipov za iskanje DES ključa.

Požrešna metoda na prostor ključev: 2^{56} korakov je zlahka paralelizirana.

Dan je par čistopis-tajnopis (P, C) ter začetni ključ K. Registri za vsako iteracijo so ločeni, tako da je vse skupaj podobno tekočemu traku:

- hitrost 50 MHz
- cena \$10.50 na čip
- 50 milijonov ključev na sekundo
- skupaj: \$100 tisoč, 5760 čipov, rabi 35 ur

Pri linearni kriptoanalizi hranjenje parov zavzame 131,000 Gbytov. Implementirano leta 1993: 10 dni na 12 mašinah.

Po odkritju diferenčne kriptoanalize je Don Coppersmith priznal, da je IBM v resnici poznal ta napad (ne pa tudi linearno kriptoanalizo) že ko so razvijali DES:

"Po posvetovanju z NSA, smo se zavedali, da utegne objava kriterijev načrtovanja odkriti tehniko kriptoanalize. To je močno sredstvo, ki se ga da uporabiti proti mnogim tajnopisom. To bi zmanjšalo prednost ZDA pred drugimi na področju kriptografije."

Novejši rezultati napadov

DES izivi pri RSA Security (3 poznani PT/CT pari):

The unkn	own mess	age is:	????????

junij 1997: razbito z internetnim iskanjem (3m).

julij 1998: razbito v treh dneh z DeepCrack mašino (1800 čipov; $250,000).

jan. 1999: razbita v 22 h, 15 min (DeepCrack + porazdeljena.mreža).

V teku (porazdeljena.mreža): RC5 – 64-bitni izziv:

- pričeli konec 1997; trenutna hitrost: 2^{36} ključev/sec (2^{25} secs/leto; pričakovani čas: ≤ 8 let).

Implementacijski napadi na DES

Napadi s pomočjo diferenčne analize porabe moči (angl. differential power analysis (**DPA**) attacks):

- Kocher, Jaffe, Jun 1999,
- procesorjeva poraba moči je odvisna od instrukcij,
- merimo porabo moči inštrukcij, ki se izvedejo v 16-ih krogih DES-a
- $\approx$1000 tajnopisa zadoščajo za odkritje tajnega ključa.

Napadi s pomočjo diferenčne analize napak (angl. differential fault analysis (**DFA**) attacks):

- Biham, Shamir 1997.
- napad: zberi naključne napake v 16-ih krogih DES-a.
- $\approx$200 napačnih odšifriranj zadošča za razkritje tajnega ključa.

Vse o napadih je veljalo za ECB način.

Isti čipe se da uporabiti tudi za druge načine, cena in čas pa se nekoliko povečata. Recimo po Wienerju za CBC način rabimo $1 milijon in 4 ure.

Varnost DES-a lahko enostavno povečamo, če uporabimo **3-DES** (zakaj ne 2-DES?).

$$\boxed{\begin{array}{l}\mathrm{DES_E(P, K_1)} \longrightarrow \mathrm{DES_D(DES_E(P, K_1), K_2)} \\ \longrightarrow \mathrm{DES_E(DES_D(DES_E(P, K_1), K_2), K_3)}\end{array}}$$

Za $K_1 = K_2 = K_3$ dobimo običajni DES.

Običajno pa zamenjamo K_3 s K_1 in dobimo približno za faktor 10^{13} močnejši sistem.

Kako veliko je VELIKO?

sekund v enem letu	$\approx 3 \times 10^7$
(živimo "le" 2-3 milijarde sekund)	
starost našega sončnega sistema	$\approx 6 \times 10^9$
(v letih)	
urinih ciklov na leto (200 MHz)	$\approx 6.4 \times 10^{15}$
01-zaporedij dolžine 64	$\approx 2^{64} \approx 1.8 \times 10^{19}$
01-zaporedij dolžine 128	$\approx 2^{128} \approx 3.4 \times 10^{38}$
01-zaporedij dolžine 256	$\approx 2^{256} \approx 1.2 \times 10^{77}$
75 številčnih praštevil	$\approx 5.2 \times 10^{72}$
elektronov v vsem vesolju	$\approx 8.37 \times 10^{77}$

mega (M)	giga (G)	tera (T)	peta (P)	exa (E)
10^6	10^9	10^{12}	10^{15}	10^{18}

Računska moč

za naše potrebe bomo privzeli, da se smatra:

- 2^{40} operacij za *lahko*,
- 2^{56} operacij za *dosegljivo*,
- 2^{64} operacij za *komaj da dosegljivo*,
- 2^{80} operacij za *nedosegljivo*,
- 2^{128} operacij za *popolnoma nedosegljivo*.

3-DES je trikrat počasnejši od DES-a.

To je pogosto nesprejemljivo, zato je leta 1984 Ron Rivest predlagal **DESX**:

$$\boxed{\mathrm{DESX_{k.k1.k2}(x) = k2 \oplus DES_k(k1 \oplus x).}}$$

DESX ključ $K = k.k1.k2$ ima

$$56 + 64 + 64 = 184 \text{ bitov.}$$

DESX trik onemogoči preizkušanje vseh mogočih ključev (glej P. Rogaway, 1996). Sedaj rabimo več kot 2^{60} izbranega čistopisa.

Hitrost

Preneel, Rijmen, Bosselaers 1997.

Softwarski časi za implementacijo na 90MHz Pentiumu.

šifra	velikost ključa (biti)	hitrost
DES	56	10 Gbits/sec (ASIC chip)
DES	56	16.9 Mbits/sec
3DES	128	6.2 Mbits/sec
RC5-32/12	128	38.1 Mbits/sec
Arcfour	variable	110 Mbits/sec

Opis šifre AES

Dolžina blokov je 128 bitov, ključi imajo tri možne dolžine: 128 ($N_r = 10$), 192 ($N_r = 12$) in 256 ($N_r = 14$),

1. *Za dan čistopis x, inicializira* `State` *z x in opravi* AddRoundKey, *ki z operacijo XOR prišteje* `RoundKey` *k* `State`.
2. *Za vsak od $N_r - 1$ krogov, opravi na* `State` *zaporedoma zamenjavo* SubBytes, *operaciji* ShiftRows *in* MixColumns *ter izvede* AddRoundKey.
3. *Naredi* SubBytes, ShiftRows *in* AddRoundKey.
4. *Za tajnopis y definira* `State`.

Vse operacije v AES so opravljene s pomočjo besed in vse spremenljivke so sestavljene iz določenega števila besed.

Čistopis x je sestavljen iz 16-ih besed: $x_0, \ldots, x_{15}$.

`State` je sestavljen iz (4×4)-dim. matrike besed:

$$\begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix}.$$

`State` dobi vrednosti iz x na naslednji način:

$$\begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix} := \begin{pmatrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{pmatrix}.$$

Na vsako besedo bomo gledali kot na dve šestnajstiški števili.

Operacija SubBytes deluje kot zamenjava, permutacija π_S $\{0,1\}^8$, na vsaki besedi od `State` posebej, z uporabo S-škatel.

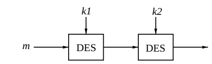
Druge simetrične šifre:

MARS, RC6, Serpent, Twofish
FEAL, IDEA, SAFER,
RC2, RC4, RC5,
LOKI, CAST, 3WAY,
SHARK, SKIPJACK,
GOST, TEA, ...

Dvojno šifriranje

2-DES: ključ $k = (k_1, k_2)$, $\quad k_1, k_2 \in_R \{0,1\}^{56}$.

Šifriranje: $c = \mathrm{DES}_{k_2}(\mathrm{DES}_{k_1}(m))$.



Odšifriranje: $m = \mathrm{DES}_{k_1}^{-1}(\mathrm{DES}_{k_2}^{-1}(c))$.

Dolžina ključa 2-DES-a je 112, torej za požrešno metodo potrebujemo 2^{112} korakov (nemogoče).

Opomba: dolžina blokov se ni spremenila.

Meet-in-the-middle napad na 2-DES

- Iz $c = E_{k_2}(E_{k_1}(m))$ sledi $E_{k_2}^{-1}(c) = E_{k_1}(m)$.
- Input: znani čp/tp pari $(m_1, c_1), (m_2, c_2), (m_3, c_3)$.
- Output: tajni ključ (k_1, k_2).

Za vsak $h_2 \in \{0,1\}^{56}$, izračunaj $E_{h_2}^{-1}(c_1)$ in shrani $[E_{h_2}^{-1}(c_1), h_2]$ v tabelo indeksirano s prvo koordinato.

Za vsak $h_1 \in \{0,1\}^{56}$ naredi naslednje:

1. Izračunaj $E_{h_1}(m_1)$.
2. Išči $E_{h_1}(m_1)$ v tabeli.
3. Za vsako *trčenje* $[E_{h_2}^{-1}(c_1), h_2]$ v tabeli preveri, ali je $E_{h_2}(E_{h_1}(m_2)) = c_2$ in $E_{h_2}(E_{h_1}(m_3)) = c_3$. Če se to zgodi, potem izpiši (h_1, h_2) in se vstavi.

Analiza:

- Število DES operacij je $\approx 2^{56} + 2^{56} = 2^{57}$.
- Pomnilnik: $2^{56}(64 + 56)$ bitov $\approx 983{,}040$ TB.

Zaključek:

- 2-DES ima enako učinkovit ključ kot DES.
- 2-DES ni varnejši od DES-a.

Time-memory tradeoff:

- Čas: 2^{56+s} korakov; pomnilnik: 2^{56-s} enot, $1 \leq s \leq 55$. [DN]

Diferenčna kriptoanaliza

- požrešna metoda in metoda z urejeno tabelo
- diferenčna metoda (za 1, 3, 6 in 16 ciklov)

Bločni tajnopisi s simetričnim ključem se ne uporabljajo samo za šifriranje, temveč tudi za konstrukcijo generatorjev psevdonaključnih praštevil, tokovnih tajnopisov, MAC in hash-funkcij.

1. **Požrešni napad**: preverimo vseh 2^{56} ključev (ne potrebujemo spomina).
2. Sestavimo **urejeno tabelo** $(e_K(x), K)$ za vseh 2^{56} ključev K in poiščemo v njej tak K, da je $y = e_K(x)$. Iskanje y-a je hitro, saj je tabela urejena.

   Ta metoda je praktična samo, če lahko večkrat uporabimo to tabelo.

Danes poznamo dva močna napada na DES: **diferenčno** kriptoanalizo in **linerno** kriptoanalizo.

Oba sta statistična, saj potrebujeta velike količine čistopisa in ustreznega tajnopisa, da določita ključ in zato nista praktična.

Zelo uspešna pa sta pri manjšem številu ciklov, npr. DES z 8imi cikli lahko razbijemo z diferenčno kriptoanalizo v nekaj minutah že na osebnem računalniku.

Diferenčno kriptoanalizo sta v letih 1990 in 1991 vpeljala Eli Biham in Adi Shamir (**izbran čistopis**).

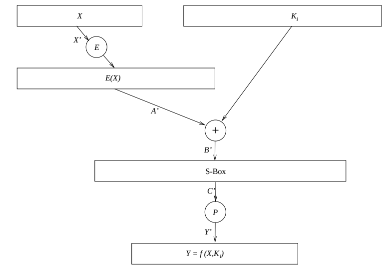
Oglejmo si pare tajnopisa za katere ima čistopis določene razlike. Diferenčna kriptoanaliza spremlja spreminjanje teh razlik, ko gre čistopis skozi nekaj ciklov DES-a in je šifriran z istim ključem.
Če poenostavimo, ta tehnika izbere pare čistopisa s fiksno razliko (čistopis je lahko izbran naključno).

Z uporabo razlik tajnopisa določimo verjetnosti različnih ključev. Analiza mnogih parov tajnopisa nam na koncu da najbolj verjeten ključ.

Naj bosta X in X^* par čistopisov z razliko X'. Tajnopisa Y in Y^* poznamo, zato poznamo tudi njuno razliko Y'. Naj bo $A^{(*)} := E(X^{(*)})$ in $P(C^{(*)}) = Y^{(*)}$.

Ker poznamo tudi razširitev E ter permutacijo P, poznamo A' in C' (glej sliko). $B^{(*)} = A^{(*)} \oplus K_i$ ne poznamo, vendar je njuna razlika B' enaka razliki A'.

Trik je v tem, da za dano razliko A' niso enako verjetne vse razlike C'. Kombinacija razlik A' in C' sugerira vrednosti bitov izrazov $A \oplus K_i$ in $A^* \oplus K_i$. Od tod pa s pomočjo A in A^* dobimo informacije o ključu K_i.



V primeru, ko imamo več kot en cikel, si pomagamo z določenimi razlikami, ki jih imenujemo **karakteristike**. Le-te imajo veliko verjetnost, da nam dajo določene razlike tajnopisa ter se razširijo, tako da definirajo pot skozi več ciklov.

Poglejmo si zadnji cikel DES-a
(začetno in končno permutacijo lahko ignoriramo).
Če poznamo K_{16} poznamo 48 bitov originalnega ključa. Preostalih 8 bitov dobimo s požrešno metodo. Diferenčna kriptoanaliza nam da K_{16}.

Podrobnosti:

Škatla S_i oziroma funkcija $S_i : \{0,1\}^6 \longrightarrow \{0,1\}^4$ ima za elemente cela števila z intervala $[0, 15]$:

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
S_1:	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Naj bo $B_j = b_1 b_2 b_3 b_4 b_5 b_6$.
$S_i(B_j)$ določimo na naslednji način.

Bita $b_1 b_6$ določita vrstico v, biti $b_2 b_3 b_4 b_5$ pa stolpec s v tabeli S_i, katere (v, s)-ti element je $S_i(B_j) \in \{0,1\}^4$

Za razliko $B'_j \in (\mathbb{Z}_2)^6$ definiramo množico z 2^6 elementi: $\Delta(B'_j) := \{(B_j, B_j \oplus B'_j) \mid B_j \in (\mathbb{Z}_2)^6\}$

Primer: oglejmo si škatlo S_1 in naj bo $B'_j = 110100$ razlika (XOR) vhodov.

$\Delta(110100) =$ {(000000,110100),(000001,110101),...,(111111,001011)}

Za vsak urejen par izračunamo razliko izhoda iz S_1:

npr. $S_1(000000) = 1110$ in $S_1(110100) = 1001$
$\Longrightarrow$ razlika izhodov $C'_j = 0111$.

Tabela izhodnih razlik C'_j in možnih vhodov B_j za vhodno razliko $B'_j = 110100$:

```
0000  -
0001  8  000011, 001111, 011110, 011111, 101010, 101011, 110111, 111011
0010 16  000100, 000101, 001110, 010001, 010010, 010100, 100101, 011011,
         100000, 100101, 010110, 101110, 101111, 110000, 110001, 111010
0011  6  000001, 000010, 010101, 100001, 110101, 110110
0100  2  010011, 100111
0101  -
0110  -
0111 12  000000, 001000, 001101, 010111, 011000, 011000, 011101, 100011,
         101001, 101100, 110100, 111001, 111100
1000  6  001001, 001100, 011001, 101101, 111000, 111101
1001  -
1010  -
1011  -
1100  -
1101  8  000110, 010000, 010110, 011100, 100010, 100100, 101000, 110010
1110  -
1111  6  000111, 001010, 001011, 110011, 111110, 111111
```

Tabela izhodnih razlik in porazdelitev vhodov za vhodno razliko 110100 (števila morajo biti soda, zakaj?):

0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	8	16	6	2	0	0	12	6	0	0	0	0	8	0	6

Pojavi se samo 8 od 16ih možnih izhodnih vrednosti.

Če pregledamo vse možnosti (za vsako škatlo S_i in vsako razliko), se izkaže, da je povprečno zastopanih samo 75-80% možnih razlik izhodov.

Ta neenakomerna porazdelitev je osnova za diferenčni napad.

Za vsako škatlo S_j (8 jih je) in za vsako vhodno razliko (2^6 jih je) sestavimo tako tabelo (skupaj 512 tabel).

Velja povdariti, da vhodna razlika ni odvisna od ključa K_i (saj smo že omenili, da je $A' = B'$), zato pa izhodna razlika C' *je* odvisna od ključa K_i.

Naj bo $A = A_1 \dots A_8$, $C = C_1 \dots C_8$ in $j \in \{1, \dots, 8\}$.

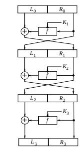
Potem poiščemo razliko $(C')_j$ v tabeli za S_j in $(A')_j$, ki nam določi vse možne vhode B_j iz katerih izračunamo vse $B_j \oplus A_j$, ki morajo vsebovati $(K_i)_j$.

Tako smo dobili nekaj kandidatov za $(K_i)_j$.

Primer: $A_1 = 000001$, $A_1^* = 110101$ in $C_1' = 1101$. Potem dobimo 13-to vrstico iz Tabele 1, ki vsebuje 8 elementov (torej smo zožili število možnosti iz $2^6 = 64$ na 8).

Z naslednjim parom čistopisa dobimo nove kandidate, $(K_i)_j$ pa leži v preseku novih in starih kandidatov...

Napad na DES s tremi cikli



Naj bo L_0R_0 in $L_0^*R_0^*$ par čistopisa in L_3R_3 in $L_3^*R_3^*$ par tajnopisa za katere velja:

$$L_3 = L_2 \oplus f(R_2, K_3) = L_0 \oplus f(R_0, K_1) \oplus f(R_2, K_3)$$

Še L_3^* izrazimo na podoben način in dobimo

$$L_3' = L_0' \oplus f(R_0, K_1) \oplus f(R_0^*, K_1) \oplus f(R_2, K_3) \oplus f(R_2^*, K_3)$$

Predpostavimo še, da je $R_0 = R_0^*$ oziroma $R_0' = 00\dots0$. Od tod dobimo

$$L_3' = L_0' \oplus f(R_2, K_3) \oplus f(R_2^*, K_3),$$

L_3' je razlika tajnopisov, L_0' pa razlika čistopisov, torej poznamo

$$f(R_2, K_3) \oplus f(R_2^*, K_3) \ \ (= L_0' \oplus L_3').$$

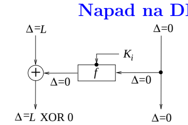
Naj bo $f(R_2, K_3) = P(C)$ in $f(R_2^*, K_3) = P(C^*)$, kjer sta C in C^* definirana enako kot prej (izhoda iz S škatel po tretjem ciklu). Potem je

$$C' = C \oplus C^* = P^{-1}(R_3' \oplus L_0').$$

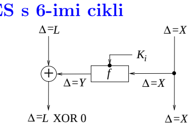
Poznamo tudi $R_2 = R_3$ in $R_2^* = R_3^*$, saj sta R_3 in R_3^* dela tajnopisa.

Torej smo prevedli kriptoanalizo DES-a s tremi cikli na diferenčno kriptoanalizo DES-a z enim ciklom.

Napad na DES s 6-imi cikli



(a)



(b)

(a) Leva stran je karkoli, desna razlika pa je 0. To je trivialna karakteristika in velja z verjetnostjo 1.

(b) Leva stran je karkoli, desna vhodna razlika pa je $0x60000000$ (vhoda se razlikujeta na 1. in 3. bitu). Verjetnost, da bosta izhodni razliki $0x60000000$ in $0x00808200$ je enaka $14/64$.

Karakteristika za n-ciklov, $n \in \mathbb{N}$, je seznam

$$L_0', R_0', L_1', R_1', p_1, \dots, L_n', R_n', p_n,$$

z naslednjimi lastnostmi:

- $L_i' = R_{i-1}'$ za $1 \le i \le n$.
- za $1 \le i \le n$ izberimo (L_{i-1}, R_{i-1}) in (L_{i-1}^*, R_{i-1}^*), tako da je $L_{i-1} \oplus L_{i-1}^* = L_{i-1}'$ in $R_{i-1} \oplus R_{i-1}^* = R_{i-1}'$. Izračunajmo (L_i, R_i) in (L_i^*, R_i^*) z enim ciklom DES-a. Potem je verjetnost, da je $L_i \oplus L_i^* = L_i'$ in $R_i \oplus R_i^* = R_i'$ natanko p_i.

Verjetnost karakteristike je $p = p_1 \times \cdots \times p_n$.

Začnimo s karakteristiko s tremi cikli:

$L_0' = 0x40080000$, $R_0' = 0x04000000$
$L_1' = 0x40000000$, $R_1' = 0x00000000$ $p = 1/4$
$L_2' = 0x00000000$, $R_2' = 0x04000000$ $p = 1$
$L_3' = 0x40080000$, $R_2' = 0x04000000$ $p = 1/4$
Potem velja

$$L_6' = L_3' \oplus f(R_3, K_4) \oplus f(R_3^*, K_4) \oplus f(R_5, K_6) \oplus f(R_5^*, K_6)$$

Iz karakteristike ocenimo $L_3' = 0x04000000$ in $R_3' = 0x40080000$ z verjetnostjo 1/16.
Od tod dobimo razliko vhodov v S škatle 4. cikla: $0010000000000000001010000 \ldots 0$.

Razlike vhodov v škatle S_2, S_5, S_6, S_7 in S_8 so 000000. To nam omogoči, da z verjetnostjo 1/16 določimo v 6-tem ciklu 30 bitov originalnega ključa.

V tabelah ne smemo nikoli naleteti na prazno vrstico (**filtracija**). Tako izključimo približno 2/3 napačnih parov, med preostalimi pa je približno 1/6 pravilnih.

...

Drugi primeri diferenčne kriptoanalize

Iste tehnike napadov na DES lahko uporabimo tudi kadar imamo več kot 6 ciklov.

DES z n cikli potrebuje 2^m izbranega čistopisa:

n	m
8	14
10	24
12	31
14	39
16	47

Na diferenčno kriptoanalizo so občutljivi tudi drugi algoritmi s substitucijami in permutacijami, kot na primer FEAL, REDOC-II in LOKI.

Napad na DES s 16-imi cikli

Bihan in Shamir sta uporabila karakteristiko s 13-imi cikli in nekaj trikov v zadnjem ciklu.

Še več, z zvijačami sta dobila 56-bitni ključ, ki sta ga lahko testirala takoj (in se s tem izognila potrebi po števcih). S tem sta dobila linearno verjetnost za uspeh, tj. če je na voljo 1000 krat manj parov, imamo 1000 manj možnosti da najdemo pravi ključ.

Omenili smo že, da najboljši napad za DES s 16-imi cikli potrebuje 2^{47} izbranih čistopisov. Lahko pa ga spremenimo v napad z 2^{55} poznanega čistopisa, njegova analiza pa potrebuje 2^{37} DES operacij.

Diferenčni napad je odvisen predvsem od strukture S škatel. Izkaže se, da so DES-ove škatle zoptimizirane proti takemu napadu.

Varnost DES-a lahko izboljšamo s tem, da povečamo število ciklov. Vendar pa diferenčna kriptoanaliza DES-a s 17-imi ali 18-imi cikli potrebuje toliko časa kot požrešna metoda (več ciklov nima smisla).

4. poglavje

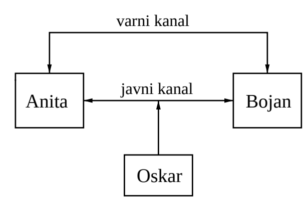
RSA sistem in faktorizacija

- Uvod
  - pomankljivosti simetrične kriptografije
  - kriptografija z javnimi ključi
- Teorija števil
- Opis in implementacija RSA
- Gostota praštevil
- Generiranje praštevil
- Gaussov izrek (o kvadratni recipročnosti)

Uvod

Pomankljivosti simetrične kriptografije

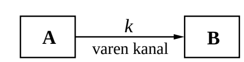
Sodelujoči si delijo *tajno* informacijo.



Dogovor o ključu

Kako Anita in Bojan vzpostavita tajni ključ k?

1. metoda: delitev point-to-point

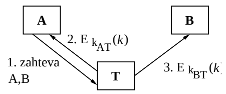


Varni kanal je lahko:
– kurir
– izmenjava na štiri oči (v temnem hodniku/ulici)

To ni praktično za večje aplikacije.

2. metoda: z neodvisnim centrom zaupanja T

– *Vsak* uporabnik A deli tajni ključ k_{AT} s centrom zaupanja T za simetrično šifrirno shemo E.
– Za vzpostavitev tega ključa mora A obiskati center zaupanja T *samo enkrat*.
– T nastopa kot **center za distribucijo ključev**: (angl. key distribution centre - KDC):



1. A pošlje T zahtevek za ključ, ki si ga želi deliti z B.
2. T izbere ključ k, ga zašifrira za A s ključem k_{AT}.
3. T zašifrira ključ k za osebo B s ključem k_{BT}.

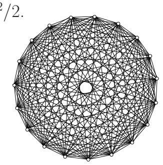
Problemi pri uporabi KDC

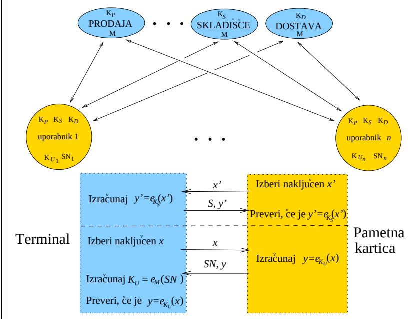
- centru zaupanja T moramo brezpogojno zaupati:
  – to ga naredi za očitno tarčo.
- Zahteva za stalno zvezo (on-line) s centrom T:
  – potencialno ozko grlo,
  – kritično za zanesljivost.

Upravljanje ključev

- v mreži z n uporabniki, mora vsak uporabnik deliti različen ključ z vsakim uporabnikom,
- zato mora hraniti vsak uporabnik $n-1$ različnih tajnih ključev,
- vseh tajnih ključev je $\binom{n}{2} \approx n^2/2$.

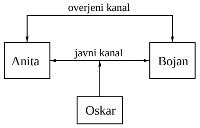
(Tudi preprečevanje tajenja je nepraktično.)





Kriptografija z javnimi ključi

Udeleženci si predhodno delijo *overjeno/avtentično* informacijo.



L. **1976** sta jo predlagala Whitfield **Diffie** in Martin **Hellman** (L. 1970 pa tudi James Ellis, ki je bil član Communication Electronics Security Group pri British Government Communications Headquarters).

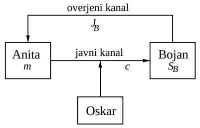
Generiranje para ključev

Vsaka oseba A naredi naslednje:

- generira par ključev (J_A, S_A),
- S_A je A-jev zasebni/tajni ključ,
- J_A je A-jev javni ključ.

Varnostna zahteva: za napadalca mora biti nemogoče priti do kluča S_A iz ključa J_A.

Šifriranje z javnimi ključi



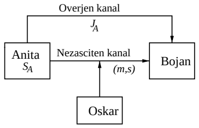
Da bi Bojanu poslala zaupno sporočil m, Anita:

- dobi overjenjo kopijo Bojanovega javnega ključa J_B,
- izračuna $c = E(J_B, m)$, kjer je E šifrirna funkcija,
- pošlje Bojanu tajnopis c.

Za odšifriranje tajnopisa c Bojan naredi naslednje

- Izračuna $m = D(S_B, c)$, kjer je D odšifrirna funkcija.

Digitalni podpisi



Za podpis sporočila m Anita naredi naslednje:

- izračuna $s = \mathsf{Sign}(S_A, m)$,
- pošlje m in s Bojanu.

Bojan preveri Anitin podpis s sporočila m:

- pridobi si overjeno kopijo javnega ključa J_A,
- sprejme podpis, če je $\mathsf{Verify}(J_A, m, s) = \mathsf{Accept}$.

Prednosti kriptosistemov z javnimi ključi

- Ni zahteve po varnem kanalu.
- Vsak uporabnik ima 1 par ključev.
- Poenostavljeno upravljanje s ključi.
- Omogoča preprečevanje tajenja.

Pomanjkljivosti kriptosistemov z javnimi ključi

- Sheme z javnimi ključi so počasnejše.
- Javni ključi so večji od simetričnih.

V praksi uporabljamo skupaj sheme s simetričnimi in javnimi ključi in jim rečemo **hibridne sheme**

Primer: Da bi Bojanu poslala podpisano tajno sporočilo m, Anita naredi naslednje:

- izračuna $s = \mathsf{Sign}(S_A, m)$,
- izbere tajni ključ k simetrične šifrirne sheme (AES),
- pridobi overjeno kopijo Bojanovega javnega ključa J_B,
- pošlje $c_1 = E(J_B, k)$, $c_2 = \mathsf{AES}(k, (m, s))$.

Za odkritje sporočila m in preverjanje avtentičnosti, Bojan:

- odšifrira c_1: $k = D(S_B, c_1)$,
- odšifrira c_2 z uporabo ključa k, da dobi (m, s),
- pridobi overjeno kopijo javnega ključa J_A,
- preveri podpis s sporočila m.

Že l. 1977 so Ronald L. **Rivest**, Adi **Shamir** in Leonard M. **Adleman** naredili prvo realizacijo takšnega kriptosistema (**RSA**)
(tajno pa že l. 1973 **C. Cocks** pri GCHQ).

Temu so sledili številni drugi nesimetrični kriptosistemi, med katerimi pa so danes najbolj pomembni naslednji:

- RSA (faktorizacija),
- Merkle-Hellman Knapsack (metoda nahrbtnika)
- Chor-Rivest
- McEliece (linearne kode),
- ElGamal (diskretni logaritem),
- eliptične krivulje.

Javni kriptosistemi **niso** nikoli brezpogojno varni, zato študiramo računsko/časovno zahtevne sisteme.

Teorija števil

Evklidov algoritem in reševanje Diofantske enačbe

$$ax + by = d, \qquad \text{kjer } \ D(a,b) \mid d.$$

Evklidov algoritem je zasnovan na preprostem dejstvu, da iz $k \mid a$ in $k \mid b$ sledi $k \mid a - b$.

Če je $D(a,b) = 1$ in poznamo eno rešitev (x_0, y_0), tj.

$$ax_0 + by_0 = d,$$

potem ima poljubna rešitev (x, y) naslednjo obliko:

$$x = x_0 - kb, \quad y = y_0 + ka, \qquad \text{za } k \in \mathbb{Z}.$$

Zgodovina Evklidovega algoritma

Evklidov algoritem poišče največji skupni delitelj dveh naravnih števil in je zasnovan na dejstvu, da če število d deli števili a in b, potem deli tudi njuno razliko $a - b$.

V literaturi naletimo nanj prvič 300 p.n.š. v 7. knjigi Evklidovih **Elementov**.

Nakateri strokovnjaki so mnenja, da je njegov avtor Eudoxus (c. 375 p.n.š.). Gre za *najstarejši* netrivialen algoritem, ki je preživel do današnjih dni (glej Knuth).

Eno rešitev lahko poiščemo z **razširjenim Evklidovim algoritmom**.

Privzemimo, da je $a > b$ in zapišimo zgornjo enačbo malo bolj splošno (z zaporedji):

$$ap_i + bq_i = r_i.$$

Poiščimo dve trivialni rešitvi:

$$p_1 = 1, \quad q_1 = 0, \quad r_1 = a$$

in

$$p_2 = 0, \quad q_2 = 1, \quad r_2 = b.$$

Zaradi rekurzije

$$r_{i+1} = r_i - s_i r_{i-1}$$

(kjer je s_i izbran tako, da je $r_{i+1} < r_i$)
si lahko izberemo še

$$p_{i+1} = p_i - s_i p_{i-1} \quad \text{in} \quad q_{i+1} = q_i - s_i q_{i-1}.$$

Ko računamo a^{-1} (po modulu praštevila p), računamo samo r_i ter p_i (ne pa tudi q_i).

Zgled za razširjeni algoritem:

$4864 = 1{\cdot}3458{+}1406$	$p_2 := p_1 - 1 \cdot p_0 = 1$
$3458 = 2{\cdot}1406{+}646$	$p_3 := p_2 - 2 \cdot p_1 = -2$
$1406 = 2{\cdot}646{+}114$	$p_4 := p_3 - 2 \cdot p_2 = 5$
$646 = 5{\cdot}114{+}76$	$p_5 := p_4 - 5 \cdot p_3 = -27$
$114 = 1{\cdot}76{+}38$	$p_6 := p_5 - 1 \cdot p_4 = 32$
$76 = 2{\cdot}38{+}0$	$p_7 := p_6 - 2 \cdot p_5 = -91$

$4864 = 1{\cdot}3458{+}1406$ $\quad p_2 = 1 \;\; q_2 = -1$
$3458 = 2{\cdot}1406{+}646$ $\quad p_3 = -2 \;\; q_3 = 3$
$1406 = 2{\cdot}646{+}114$ $\quad p_4 = 5 \;\; q_4 = -7$
$646 = 5{\cdot}114{+}76$ $\quad p_5 = -27 \;\; q_5 = 38$
$114 = 1{\cdot}76{+}38$ $\quad p_6 = 32 \;\; q_6 = -45$
$76 = 2{\cdot}38{+}0$ $\quad p_7 = -91 \;\; q_7 = 128$

$$4864 \cdot (-91) + 3458 \cdot (128) = 38$$

Čeprav uporabljamo ta algoritem že stoletja, pa je presenetljivo, da ni vedno najboljša metoda za iskanje največjega skupnega delitelja.

R. Silver in **J. Terzian** sta leta **1962** (v lit. J. Stein, *J. Comp. Phys.* **1** (1967), 397-405) predlagala **binarni algoritem**:

B1. *Poišči tak največji $k \in \mathbb{Z}$, da bosta števili a in b deljivi z 2^k; $a \leftarrow a/2^k$ in $b \leftarrow b/2^k$, $K \leftarrow 2^k$.*

B2. *Dokler $2|a$ ponavljaj $a \leftarrow a/2$ in dokler $2|b$ ponavljaj $b \leftarrow a/2$.*

B3. *Če je $a = b$, je $D(a,b) = a * K$, sicer pa v primeru $a > b$, priredi $a \leftarrow a-b$, sicer $b \leftarrow b-a$ in se vrni na korak B2.*

Lehmerjev algoritem deli z majhnimi namesto velikimi števili (izboljšave J. Sorenson, Jaebelan,...).

Dobro vprašanje je kako prenesti te ideje v GF(2^n).

R. Schroeppel je že naredil prvi korak s svojim algoritmom **almost inverse**.

Kitajski izrek o ostankih. *Če so števila $m_1, m_2, ..., m_r$ paroma tuja, tj. $D(m_i, m_j) = 1$ za $i \neq j$, in $a_1, a_2, ..., a_r \in \mathbb{Z}$, potem ima sistem kongruenc*

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_r \pmod{m_r} \end{aligned}$$

enolično rešitev po modulu $M = m_1 \cdot m_2 \cdots m_r$,

$$x = \sum_{i=1}^{r} a_i \cdot M_i \cdot y_i \pmod{M},$$

kjer je $M_i = M/m_i$, $y_i = M_i^{-1} \bmod m_i$, $i = 1, ..., r$.

(angl. Chinese Reminder Theorem oziroma CRT)

Red elementa g v končni multiplikativni grupi je najmanjše celo število m tako, da $g^m = 1$.

Lagrangev izrek: *Naj bo G multiplikativna grupa reda n in $g \in G$, potem red g deli n.*

Naj bo p praštevilo. Generatorju multiplikativne grupe $\mathbb{Z}_p^*$ pravimo **primitiven element**.

DN: Koliko primitivnih elementov ima $\mathbb{Z}_p$?
Naj bo α primitiven element, potem za $\forall \beta \in \mathbb{Z}_p^*$ obstaja tak $i \in \{0, 1, \ldots, p-2\}$, da je $\beta = \alpha^i$.
Pokaži, da je red elementa β enak $\dfrac{p-1}{D(p-1, i)}$.

Eulerjevo funkcijo φ definiramo s

$$\varphi(n) = |\{x \in \mathbb{N} \,|\, x < n \text{ in } D(x,n) = 1\}|.$$

Potem za praštevilo p, naravno število n in poljubni tuji si števili a in b velja

$$\varphi(p^n) = p^n - p^{n-1} \;\text{ in }\; \varphi(ab) = \varphi(a)\varphi(b).$$

Če poznamo faktorizacijo števila n, poznamo tudi $\varphi(n)$.

Fermatov izrek
Za praštevilo p in $b \in \mathbb{Z}_p$ velja $b^p \equiv b \pmod{p}$.

Eulerjev izrek
Če je $a \in \mathbb{Z}_n^*$ oziroma $D(n,a) = 1$, potem velja

$$a^{\varphi(n)} \equiv 1 \pmod{n}.$$

Opis in implementacija RSA

Generiranje ključev: najprej izberemo

- praštevili $\boldsymbol{p}$, $\boldsymbol{q}$ ter izračunamo *modul* $\boldsymbol{n} := pq$, in
- *šifrirni eksponent* $\boldsymbol{e}$, tako da je $D(e, \varphi(n)) = 1$,

nato pa izračunamo *odšifrirni eksponent* $\boldsymbol{d}$ iz kongruence

$$ed \equiv 1 \pmod{\varphi(n)}$$

z razširjenim Evklidovim algoritmom
(ali pa potenciranjem).

Javni ključ je $(\boldsymbol{e}, \boldsymbol{n})$, **zasebni ključ** pa $(\boldsymbol{d}, \boldsymbol{p}, \boldsymbol{q})$.

Šifriranje: $E(e,n)(x) = x^e \pmod{n}$.

Odšifriranje: $D(d,p,q)(y) = y^d \pmod{n}$.

Šifriranje in odšifriranje sta inverzni operaciji.
Za $x \in \mathbb{Z}_n^*$ to sledi iz Eulerjeve kongruence:

$$(x^e)^d \equiv x^{r\varphi(n)+1} \equiv (x^{\varphi(n)})^r x \equiv x \pmod{n},$$

za $x \in \mathbb{Z}_n \backslash \mathbb{Z}_n^*$ pa se prepričajte sami za DN.

Generiranje podpisa:
za podpis sporočila $m \in \{0,1\}^*$, Anita:

1. izračuna $M = H(m)$,
   kjer je H zgoščevalna funkcija (npr. SHA-1),
2. izračuna $s = M^d \bmod n$,
3. Anitin podpis za m je s.

Preverjanje podpisa:
Bojan preveri Anitin podpis s za m, tako da:

1. vzame overjeno kopijo Anitinega
   javnega ključa (n,e),
2. izračuna $M = H(m)$,
3. izračuna $M' = s^e \bmod n$,
4. sprejme (m,s) če in samo če je $M = M'$.

Potenciranje z redukcijo pri RSA je enosmerna funkcija z bližnjico.

Bližnjica: poznavanje števila d oziroma $\varphi(n)$ oziroma števil p in q.

RSA v praksi

- Modul $n = pq$ mora biti dovolj velik, da je njegova faktorizacija računsko prezahtevna.
- Implementacije RSA z dolžino ključev 512 bitov ne jamčijo več dolgoročne varnosti.

Časovna zahtevnost računskih operacij

Naj ima število n v binarni representaciji k bitov, tj.

$$k = \lfloor \log_2 n \rfloor + 1.$$

Potem je časovna zahtevnost

seštevanja $O(k)$,
Evklidovega algoritma $O(k^2)$,
modularne redukcije $O(k^2)$,
potenciranja pa $O(k^3)$.

Potenciranje opravimo učinkovito z metodo "**kvadriraj in množi**".

Izbira šifrirnega eksponenta

$$e = 5, 17, 2^{16}+1$$

Pospešitev odšifriranja z uporabo kitajskega izreka o ostankih (CTR) za faktor 4:

namesto da računamo $y^d \bmod n$ direktno, najprej izračunamo

$C_p := y^{d \bmod (p-1)} \bmod p$ in $C_q := y^{d \bmod (q-1)} \bmod q$,

nato pa po CRT še

$$C := t_p C_p + t_q C_q \bmod n,$$

kjer $p \mid t_p - 1, t_q$ in $q \mid t_p, t_q - 1$.

Algoritem RSA je cca. 1500-krat počasnejši od DES-a. Uporablja se za prenos ključev simetričnega algoritma.

(Za 512-bitno število n lahko dosežemo z RSA-jem hitrost 600 Kb na sekundo, medtem ko DES zmore 1 Gb na sekundo.)

Nekaj lažjih nalog

1. Koliko množenj potrebujemo,
   da izračunamo m^d?
2. Prepričaj se, ali je dovolj, da pri RSA uporabimo
   le Fermatovo kongruenco.

3. Pokaži, da $p \mid \binom{p}{i}$, za $1 < i < p$.
4. Naj bo p praštevilo, potem za poljubni števili a in b velja
   $$(a+b)^p \equiv a^p + b^p \pmod{p}.$$
5. Naj bo p praštevilo, potem za poljubno število m velja $m^p \equiv m \pmod{p}$.

Gostota praštevil

Izrek o gostoti praštevil
[de la Vallée Poussin, Hadamard, 1896]
Funkcija $\pi(x)$ je asimptotično enaka $\frac{x}{\log x}$, ko gre $x \to \infty$.

(angl. **Prime Number Theorem** oziroma PNT)

Domnevo za PNT je prvi postavil leta 1791 (še kot najstnik) **Frederic Gauss** **(1777-1855)**, za testiranje pa je kasneje uporabljal tudi tablice **Jurija Vege** iz leta 1796:

$$\pi(x) \approx \int_2^x \frac{1}{\log n} dn \ .$$

Legendre pa jo je objavil v svoji knjigi iz leta 1808:

$$\pi(x) \approx \frac{x}{\log x - 1.08366} \ .$$

Namesto da bi šteli praštevila, ki so manjša ali enaka številu n, raje poglejmo, kakšna je njihova *gostota*:

$$\pi(n)/n.$$

Primerjamo

$$\pi(10^{10})/10^{10} = .04550525$$

z

$$1/\ln(10^{10}) = .04342945.$$

To je bil **problem 19. stoletja.**

Peter Gustav Lejeune-Dirichlet **(1805-1859)** (začetki analitične teorije števil): za vsaki tuji si celi števili a in b aritmetično zaporedje

$$a, \quad a+b, \quad a+2b, \quad a+3b, \quad \dots, a+nb, \quad \dots$$

vsebuje neskončno praštevil.

Pafnuti Lvovich Chebyshev **(1821-1884)** je leta 1850 pokazal, da, če limita obstaja, potem leži na intervalu

$$[0.92129, 1.10555].$$

Leta 1859 je **Georg Friedrich Bernhard Riemann** **(1826-1866)** naredil briljanten napredek na področju analitične teorije števil s študijem Riemannove **zeta funkcije**.

Leta 1896 sta končno dokazala domnevo

Charles-Jean-Gustave-Nicholas de la Vallée-Poussin **(1866-1962)**

in

Jacques Hadamard **(1865-1963)**.

V Prilogi A si lahko ogledate dokaz izreka, ki sledi D. Zagieru, ki je uporabil analitični izrek namesto Tauberjevih izrekov. (Newman's Short Proof of the Prime Number Theorem, *American Mathematical Monthly*, October 1997, strani 705-709).

Še nekaj zanimivih referenc:

J. Korevaar, On Newman's quick way to the prime number theorem, *Mathematical Intelligencer* 4, **3** 1982, 108-115.

P. Bateman and H. Diamond, A hundred years of prime numbers, *American Mathematical Monthly* **103** 1996, 729-741.

Elementarni dokaz izreka o gostoti praštevil

Prvi dokaz so poenostavili **Landau** in drugi v začetku 20. stoletja. Vsi so uporabljali zapletene metode realne in kompleksne analize.

Leta 1949 sta **Atle Selberg** in **Paul Erdös** odkrila neodvisno elementaren dokaz (brez kompleksne analize).

Leta 1956 je **Basil Gordon** dokazal izrek o gostoti praštevil s pomočjo Stirlingove formule za $n!$.

The Times London, sept. 25, 1996:

Selberg and Erdös agreed to publish their work in back-to-back papers in the same journal, explaining the work each had done and sharing the credit. But at the last minute Selberg ... raced ahead with his proof and published first. The following year Selberg won the Fields Medal for this work. Erdös was not much concerned with the competitive aspect of mathematics and was philosophical about the episode.

`http://www-groups.dcs.st-and.ac.uk/`
`~history/Mathematicians/Erdos.html`

Posledica: *Če je p_n n-to praštevilo, velja*

$$\lim_{n\to\infty} \frac{n \log n}{p_n} \;=\; 1.$$

Dokaz: Logaritmirajmo limito iz izreka o gostoti praštevil

$$\lim_{x\to\infty} (\log \pi(x) + \log\log x - \log x) \;=\; 0$$

oziroma

$$\lim_{x\to\infty} \left\{ \log x \left(\frac{\log \pi(x)}{\log x} + \frac{\log\log x}{\log x} - 1 \right) \right\} \;=\; 0\,.$$

Ker gre $\log x \to \infty$, velja

$$\lim_{x\to\infty} \left\{ \frac{\log \pi(x)}{\log x} + \frac{\log\log x}{\log x} - 1 \right\} \;=\; 0$$

oziroma ker gre $\log\log x/\log x \to 0$, tudi

$$\lim_{x\to\infty} \frac{\log \pi(x)}{\log x} \;=\; 1.$$

Pomnožimo še z limito iz izreka o gostoti praštevil in dobimo

$$\lim_{x\to\infty} \frac{\pi(x)\log \pi(x)}{x} \;=\; 1,$$

kar pa je že želena limita, če vzamemo $x = p_n$ oziroma $\pi(x) = n$. ■

RSA sistem in faktorizacija

- Probabilistično testiranje praštevilčnosti (ponovitev Monte Carlo algoritem, Solovay-Strassen algoritem in Miller-Rabinov test)
- Napadi na RSA (odšifrirni eksponent, Las Vegas algoritem)
- Rabinov kriptosistem
- Algoritmi za faktorizacijo (naivna metoda, metoda $p-1$, Dixonov algoritem in kvadratno rešeto)

Generiranje praštevil

Za inicializacijo RSA kriptosistema potrebujemo velika (npr. 80-mestna) naključna praštevila.

V praksi generiramo veliko naključno število in testiramo, ali je praštevilo z Monte Carlo algoritmom (npr. Solovay-Stassen ali Miller-Rabin).

Ti algoritmi so hitri, vendar pa so probabilistični in ne deterministični. Po izreku o gostoti praštevil je verjetnost, da je naključno 512-bitno liho število praštevilo, približno $2/\log p \approx 2/177$.

S praštevili, ki so "osnovni gradniki" matematike, so se ukvarjali učenjaki vse od antičnih časov dalje.

Odločitveni problem `praštevilo`
Za dano število n ugotovi ali je praštevilo.

Leta **240 pr. n. št.** se je grški matematik in filozof **Eratostenes**, bibliotekar aleksandrijske knjižnice, domislil prve neoporečne metode (*čas. zahtev.* $O(n)$). V primeru zelo dolgih števil bi za rešitev tega problema potrebovali več časa kot je staro vesolje.

Od tedaj so matematiki poskušali najti algoritem, ki bi dal odgovor v smiselnem času.

Karl Frederich Gauss (1777-1855) je v svoji knjigi Disquisitiones Arithmeticae (1801) zapisal:

> *"Menim, da čast znanosti narekuje, da z vsemi sredstvi iščemo rešitev tako elegantnega in tako razvpitega problema."*

Od prihoda računalnikov dalje poudarek ni več na iskanju matematične formule, ki bi dajala praštevila, ampak na iskanju učinkovitega algoritma za razpoznavanje praštevil.

Večji korak naprej je v 17. stoletju napravil **Fermat**, z že omenjenim **Fermatovim malim izrekom**:

$$a^{p-1} \equiv 1 \pmod p$$

za vsak $a \in \mathbb{N}$ in vsako praštevilo p, ki ne deli a.

Po zaslugi kriptografije so postale raziskave problema `praštevilo` v zadnjih desetletjih še intenzivnejše:

1976	**Miller**: deterministični algoritem polinomske časovne zahtevnosti (temelji na Riemannovi hipotezi)
1977	**Solovay in Strassen**: verjetnostni algoritem časovne zahtevnosti $O(\log^3 n)$.
1980	**Rabin**: modifikacija Millerjevega testa v verjetnostni alg. (pravilnost dokazana)
1983	**Adleman, Pomerance in Rumely**: det. alg. čas. zathev. $O(\log n^{O(\log\log\log n)})$
1986	**Golwasser in Kilian**: polinomski verj. alg. za skoraj vse podatke z uporabo eliptičnih krivulj
2002	**Agrawal, Kayal in Saxena (AKS)**: det. alg. s časovno zahtevnostjo $O(\log^{12} n)$ v praksi $O(\log^6 n)$, tudi $O(\log^3 n)$ a brez dokaza.

Naj bo p liho praštevilo, $0 \le x \le p-1$.
Potem je x **kvadratni ostanek** po modulu p, tj. $x \in$ **QR(p)**, če ima kongruenca

$$y^2 \equiv x \pmod{p}$$

rešitev $y \in \mathbb{Z}_p$.

Eulerjev kriterij
Naj bo p liho praštevilo. Potem je

$$x \in \mathrm{QR}(p) \iff x^{(p-1)/2} \equiv 1 \pmod{p}.$$

Torej obstaja polinomski algoritem za odločitveni problem **kvadratnega ostanka**.

Dokaz:

Naj bo p liho praštevilo in a nenegativno celo število. Potem je **Legendrov simbol** definiran z

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{če } p \mid a, \\ 1, & \text{če je } a \in \mathrm{QR}(p), \\ -1, & \text{sicer.} \end{cases}$$

Po Eulerjevem kriteriju velja

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

Legendrov simbol posplošimo v Jacobijev simbol. Število n naj bo celo liho število z naslednjo praštevilsko faktorizacijo $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$.

Za nenegativno celo število a definiramo **Jacobijev simbol** z

$$\left(\frac{a}{n}\right) := \Pi_{i=1}^{k} \left(\frac{a}{p_i}\right)^{e_i}.$$

Eulerjevo psevdopraštevilo: $91 = 7 \cdot 13$, vseeno pa obstaja tak $a = 10$, da je

$$\left(\frac{10}{91}\right) = -1 = 10^{45} \mod 91.$$

DN: Pokaži, da je za poljubno sestavljeno število n, m Eulerjevo psevdopraštevilo glede na bazo a za največ polovico naravnih števil, ki so manjša od n (glej nalogo 4.14).

DA-naklonjen Monte Carlo algoritem je probabilistični algoritem za odločitveni problem (tj. DA/NE-problem), pri katerem je "DA" odgovor (vedno) pravilen, "NE" odgovor pa je lahko nepravilen.

Verjetnost napake za **DA-naklonjen Monte Carlo** algoritem je ε, če za vsak odgovor "DA" algoritem odgovori z "NE" z verjetnostjo kvečjemu ε.

Solovay-Strassen algoritem

1. *Izberi naključno število $a \in \mathbb{Z}_n$,* $\quad x := \left(\frac{a}{n}\right)$.
2. **if** $x = 0$ **then return**(*"n je sestavljeno število"*).
3. $y := a^{(n-1)/2} \bmod n$,
   if $x \equiv y \pmod{n}$
   then return(*"n je praštevilo"*)
   else return(*"n je sestavljeno število"*).

Verjetnost napake pri Solovay-Strassen algoritmu je kvečjemu $1/2$ (glej nalogo 4.14 v Stinsonu).

Monte Carlo verjetnostni algoritem za odločitveni problem, ali je število sestavljeno: test ponovimo m-krat z naključnimi vrednostmi a. Verjetnost, da bo odgovor napačen m-krat zapored napačen je ε^m, vendar pa iz tega še ne moremo zaključiti, da je verjetnost, da je n praštevilo, $1 - \varepsilon^m$.

Dogodek A:

"naključen lih n določene velikosti je sestavljen"

in dogodek B:

"algoritem odgovori 'n je praštevilo' m-krat zapored."

Potem očitno velja $P(B/A) \le \varepsilon^m$, vendar pa nas v resnici zanima $P(A/B)$, kar pa ni nujno isto.

Naj bo $N \le n \le 2N$ in uporabimo izrek o gostoti praštevil

$$\frac{2N}{\log 2N} - \frac{N}{\log N} \approx \frac{N}{\log N} \approx \frac{n}{\log n}.$$

Sledi $P(A) \approx 1 - 2/\log n$. Bayesovo pravilo pravi:

$$P(A/B) = \frac{P(B/A)P(A)}{P(B)}.$$

Imenovalec je enak $P(B/A)P(A) + P(B/\overline{A})P(\overline{A})$.

Upoštevajmo še $P(B/\overline{A}) = 1$ in dobimo

$$P(A/B) = \frac{P(B/A)(\log n - 2)}{P(B/A)(\log n - 2) + 2} \leq$$

$$\leq \frac{2^{-m}(\log n - 2)}{2^{-m}(\log n - 2) + 2} = \frac{(\log n - 2)}{\log n - 2 + 2^{m+1}},$$

kar pomeni, da gre iskana verjetnost eksponentno proti 0.

Monte Carlo verjetnostni algoritem za odločitveni problem ali je število sestavljeno:

Test ponovimo k-krat z različnimi vrednostmi a. Verjetnost, da bo ogovor k-krat zapored napačen, je za nas ocenjena z ε^k.

DN: Iz naslednjega izreka izpeljite, da za izračun Jacobijevega simbola ne potrebujemo praštevilske faktorizacije števila n.

Gaussov izrek

Izrek o kvadratni recipročnosti (1796)
Če sta p in q različni lihi praštevili, potem velja

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2}\cdot\frac{q-1}{2}}$$

ter za praštevilo 2

$$\left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}.$$

Zakaj je ta izrek tako pomemben?

Pomaga nam, da odgovorimo, kdaj imajo kvadratne kongruence rešitev, saj velja multiplikativno pravilo

$$\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right).$$

Predstavlja pa tudi nepričakovano zvezo med pari praštevil (pravilo, ki ureja praštevila).

Eisensteinova lema. $p > 2$ *praštevilo,* $p \not| q \in \mathbb{N}$. *Naj bo* $A := \{2, 4, 6, \ldots, p-1\}$ *in* $r_a := qa \bmod p$ *za* $a \in A$. *Potem je*

$$\left(\frac{q}{p}\right) = (-1)^{\sum\limits_{a \in A} r_a}.$$

Dokaz: Za $a, a' \in A$, $a \neq a'$, ne more veljati

$r_a(-1)^{r_a} = r_{a'}(-1)^{r_{a'}}$ oziroma $qa \equiv \pm qa' \pmod{p}$,

saj bi od tod sledilo $a = \pm a'$, kar pa ni mogoče.

Opozorimo še, da so vsa števila $r_a(-1)^{r_a} \bmod p$ soda, torej pretečejo ravno vse elemente množice A.

Od tod dobimo

$$\prod a \equiv (-1)^{\sum r} \prod r \pmod{p},$$

očitno pa neposredno iz definicije sledi tudi

$$q^{\frac{p-1}{2}} \prod a \equiv \prod r \pmod{p}.$$

Torej velja $q^{\frac{p-1}{2}} \equiv (-1)^{\sum r} \pmod{p}$ in po Eulerjevem kriteriju še

$$\left(\frac{q}{p}\right) \equiv q^{\frac{p-1}{2}} \pmod{p}. \quad \blacksquare$$

Oglejmo si Eisensteinov *dokaz Gaussovega izreka o kvadratni recipročnosti.* Očitno velja

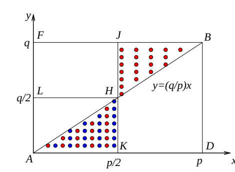
$$\sum qa = p \sum \left\lfloor \frac{qa}{p} \right\rfloor + \sum r\ .$$

Ker so elementi a vsi sodi in je p lih, velja

$$\sum r \equiv \sum \left\lfloor \frac{qa}{p} \right\rfloor \pmod{2}$$

in zato iz Eisensteinove leme sledi

$$\left(\frac{q}{p}\right) = (-1)^{\sum \left\lfloor \frac{qa}{p} \right\rfloor}\ .$$



Vsota $\sum \left\lfloor \frac{qa}{p} \right\rfloor$ je enaka številu celoštevilčnih točk sodo x-koordinato, ki ležijo v notranjosti trikotnika ABD. Sedaj pa si oglejmo točke z x-koordinato večjo od $p/2$. Ker pa je $q-1$ sod, je parnost števila $\left\lfloor \frac{qa}{p} \right\rfloor$ točk z isto x-koordinato pod diagonalo AB enako številu točk z isto sodo x-koordinato nad diagonalo AB.

To pa je po drugi strani enako številu točk pod diagonalo AB z liho x-koordinato $p-a$ (bijektivna korespondenca med točkami s sodo x-koordinato v BHJ in liho x-koordinato v AHK). Od tod sledi, da ima vsota $\sum\left\lfloor\frac{qa}{p}\right\rfloor$ enako parnost kot številu μ celoštevilčnih točk v notranjosti trikotnika AHK, tj.

$$\left(\frac{q}{p}\right) = (-1)^{\mu}.$$

Če zamenjamo p in q, dobimo še število ν celoštevilčnih točk v notranjosti trikotnika AHL, kar nam da

$$\left(\frac{p}{q}\right) = (-1)^{\nu}$$

in skupaj s prejšnjo relacijo Gaussov izrek. ■

Še en Monte Carlo algoritem za testiranje sestavljenosti števil.

Miller-Rabinov test: *testiramo liho število n.*

1. $n-1=2^k m$, *kjer je m liho število,*
2. *izberemo naključno naravno število $a<n$,*
3. *izračunamo $b\equiv a^m \pmod{n}$,*
4. **if** $b\equiv 1 \pmod{n}$ **then** *n je praštevilo;* **exit;**
5. **for** $i=0$ **to** $k-1$ **do**
   if $b\equiv -1 \pmod{n}$
   then *n je praštevilo;*
   exit;
   else $b\equiv b^2 \pmod{n}$,
7. *število n je sestavljeno.*

Izrek: *Miller-Rabinov algoritem za problem sestavljenih števil je DA-naklonjen Monte Carlo algoritem.*

Dokaz: Predpostavimo, da algoritem odgovori "n je sestavljeno število" za neko praštevilo p.

Potem je $a^m \not\equiv 1 \pmod{n}$.

Sledi $a^{2^i m} \not\equiv -1 \pmod{n}$ za $i\in\{0,1,\dots,k-1\}$.

Ker je $n=2^k m+1$ praštevilo, iz Fermatovega izreka sledi

$$a^{2^k m}\equiv 1 \pmod{n}$$

in je $a^{2^{k-1}m}$ koren od 1 po modulu n.

Iz $x^2\equiv 1 \pmod{n}$ oziroma $n|x^2-1=(x-1)(x+1)$ sledi

$$x\equiv 1 \pmod{n} \quad \text{ali} \quad x\equiv -1 \pmod{n}$$

oziroma v našem primeru $a^{2^{k-1}m}\equiv 1 \pmod{n}$. Na isti način pridemo do

$$a^m\equiv 1 \pmod{n},$$

kar je protislovje, saj bi algoritem v tem primeru odgovoril "n je praštevilo". ■

Za konec omenimo brez dokaza še, da je verjetnost napake Miller-Rabinovega algoritma kvečjemu $1/4$.

Napadi na RSA

Odličen pregledni članek "Twenty Years of Attacks on the RSA kriptosysystem", je objavil Dan Boneh v *Notices of AMS*, Feb. 1999, pp. 203-212.

Mi bomo omenili le nekaj osnovnih napadov.

Če poznamo $\varphi(n)$ in n, dobimo p, q iz naslednjega sistema dveh enačb

$$n=pq \qquad \text{in} \qquad \varphi(n)=(p-1)(q-1).$$

Odšifrirni eksponent kriptosistema RSA

Trditev: *Vsak algoritem A, ki najde odšifrirni eksponent d, lahko uporabimo kot podprogram v probabilističnem algoritmu, ki najde faktorje števila n.*

Od tod sledi, da iskanje odšifrirnega eksponenta ni nič lažje kot problem faktorizacije.

Opozorilo: če "izgubimo" d, moramo poleg šifrirnega eksponenta zamenjati tudi modul n.

Naj bo $\varepsilon \in [0,1)$. **Las Vegas algoritem** je probabilističen algoritem, ki za dani primer problema, lahko *ne da odgovora* z verjetnostjo ε (se pravi, da konča s sporočilom "ni odgovora"). Če pa algoritem odgovori, potem je *odgovor gotovo pravilen.*

DN: Pokaži, da je povprečno pričakovano število ponovitev algoritma vse dokler ne dobimo odgovora, enako $1/(1-\varepsilon)$ (glej nalogo 4.15).

Če Las Vegas algoritem faktorizira število n z verjetnostjo vsaj ε in ga ponovimo m-krat, potem bo število n faktorizirano z verjetnostjo vsaj $1-\varepsilon^m$.

Trditev sledi iz algoritma, ki uporablja naslednje: za $n=pq$, kjer sta p,q lihi praštevili,

$$x^2\equiv 1 \pmod{n}, \quad \text{tj.} \quad pq|(x-1)(x+1),$$

dobimo štiri rešitve; dve (trivialni) rešitvi iz enačb

$$x\equiv 1 \pmod{n} \quad \text{in} \quad x\equiv -1 \pmod{n}$$

in s pomočjo kitajskega izreka o ostankih iz

$$x\equiv 1 \pmod{p}, \qquad x\equiv -1 \pmod{q}$$

in

$$x\equiv -1 \pmod{p}, \qquad x\equiv 1 \pmod{q}$$

še dve (netrivialni) rešitvi.

Algoritem za faktorizacijo z danim šifr. eksp. d

1. Izberi naključno naravno število $w < n$,
2. izračunaj $x = D(w, n)$,
3. **if** $1 < x < n$ **then exit**(uspeh $x = p$ ali $x = q$)
4. izračunaj $d = A(e, n)$ in zapiši $de - 1 = 2^s r$, r lih,
5. izračunaj $v = w^r \mod n$,
6. **if** $v \equiv 1 \pmod{n}$ **then exit**(neuspeh)
7. **while** $v \not\equiv 1 \pmod{n}$ **do** $v_0 = v$, $v = v^2 \mod n$
8. **if** $v_0 \equiv -1 \pmod{n}$ **then exit**(neuspeh)
   else izračunaj $x = D(v_0 + 1, n)$
   (uspeh: $x = p$ ali $x = q$) .

Naključne napake
(Boneh, DeMillo in Lipton, 1997)

Če uporabimo CRT in pride pri samo enem izmed C_p in C_q do napake, npr. C_p je pravilen, $\hat{C}_q$ pa ni, potem je $\hat{C} = t_p C_p + t_q \hat{C}_q$ očitno nepravilen podpis, saj je $\hat{C}^e \neq M \mod N$. Vendar pa je

$$\hat{C}^e = M \mod p, \quad \text{medtem, ko je} \quad \hat{C}^e \neq M \mod q$$

in nam $D(n, \hat{C}^e - M)$ odkrije netrivialni faktor števila n.

Rabinov kriptosistem

Temelji na tem, da je težko najti faktorizacijo produkta dveh velikih praštevil p in q.

$n = pq, \quad p \neq q, \quad p, q \equiv 3 \pmod{4}, \quad \mathcal{P} = \mathcal{C} = \mathbb{Z}_n$

$$\mathcal{K} = \{(n, p, q, B);\ 0 \leq B \leq n-1\}$$

Za izbrani ključ $K = (n, p, q, B)$ *naj bo:*

$$e_K(x) = x(x + B) \pmod{n},$$

$$d_K(y) = \sqrt{y + B^2/4} - B/2.$$

Javni ključ *je* (n, B), **zasebni ključ** *pa* (p, q).

Trditev: *Naj bo* $\omega^2 \equiv 1 \pmod{n}$ *netrivialen koren (kongruenca ima 4 rešitve:* 1, -1 *in še dve netrivialni), in* $x \in \mathbb{Z}_n$, *potem velja:*

$$e_K(\omega(x + B/2) - B/2) = e_K(x).$$

Imamo 4 čistopise, ki ustrezajo tajnopisu $e_K(x)$:

$$x, \quad -x - B, \quad \omega(x + \frac{B}{2}) \ \text{ in } \ -\omega(x + \frac{B}{2}).$$

V splošnem se ne da ugotoviti, kateri je pravi.

Odšifriranje

Imamo tajnopis y in iščemo x, ki zadošča naslednji enačbi:

$$x^2 + Bx \equiv y \pmod{n}.$$

Poenostavimo: $x = x_1 - B/2$,

$$x_1^2 \equiv y + B^2/4 \pmod{n}, \qquad C = y + B^2/4.$$

Iščemo kvadratne korene enačbe $\boxed{x_1^2 \equiv C \pmod{n}}$.

To je ekvivalentno sistemu:

$x_1^2 \equiv C \pmod{p}$
$x_1^2 \equiv C \pmod{q}$

Eulerjev izrek:
$C^{(p-1)/2} \equiv 1 \pmod{p}$

predpostavka: $p \equiv 3 \pmod{4}$

$\downarrow \qquad \Rightarrow \quad (\pm C^{(p+1)/4})^2 \equiv C \pmod{p}$

$x_1 \equiv x_{1,2} \pmod{p}$
$x_1 \equiv x_{3,4} \pmod{q}$

$\Rightarrow$ korena prve enačbe sta:
$x_{1,2} = \pm C^{(p+1)/4}$

korena druge enačbe pa:
$x_{3,4} = \pm C^{(q+1)/4}$

$\Downarrow$ KIO
x_1, x_2, x_3, x_4

Primer: $n = 77 = 7 \cdot 11, \quad B = 9$

$e_K(x) = x^2 + 9x \pmod{77}$

$$d_K(y) = \sqrt{y + B^2/4} - B/2 = \sqrt{1 + y} - 43 \pmod{77}$$

Tajnopis: $y = 22$. Poiskati moramo rešitve:

$x^2 \equiv 23 \pmod{7} \qquad (x \equiv \pm 4 \pmod{7})$
$x^2 \equiv 23 \pmod{11} \qquad (x \equiv \pm 1 \pmod{11})$

Dobimo štiri sisteme dveh enačb z dvema neznankama, npr.:

$$x \equiv 4 \pmod{7}, \quad x \equiv 1 \pmod{11}$$

Po kitajskem izreku o ostankih velja:

$$x = 4 \cdot 11 \cdot (11^{-1} \bmod 7) + 1 \cdot 7 \cdot (7^{-1} \bmod 11).$$

Vse rešitve so:

$x_1 = 67 \pmod{77}, \quad x_2 = 10 \pmod{77},$
$x_3 = 32 \pmod{77}, \quad x_4 = -32 \pmod{77}.$

Odšifrirani tekst je:

$$\begin{aligned} d_K(y) = \ & 67 - 43 \pmod{77} = 24 \\ & 10 - 43 \pmod{77} = 44 \\ & 32 - 43 \pmod{77} = 66 \\ & 45 - 43 \pmod{77} = 2, \end{aligned}$$

vse štiri rešitve pa se zašifrirajo v 22.

Varnost Rabinovega kriptosistema

Hipotetični algoritem A za dekripcijo Rabinovega kriptosistema lahko uporabimo kot podprogram v algoritmu tipa Las Vegas za faktorizacijo števila n z verjetnostjo vsaj $1/2$.

1. Izberemo r, $1 \leq r \leq n-1$,
2. $y := r^2 - B^2/4 \pmod{n}$ $(y = e_K(r - B/2))$,
3. $x := A(y)$,
4. $x_1 := x + B/2$ $(x_1^2 \equiv r^2 \pmod{n})$,
5. če velja $x_1 \equiv \pm r \pmod{n}$, potem ni odgovora, sicer ($x_1 \equiv \pm\omega \cdot r \pmod{n}$, kjer je $\omega \equiv 1 \pmod{n}$ netrivialni koren) $D(x_1 + r_1, n) = p$ (ali q).

V zadnjem primeru $n \,|\, (x_1 - r)(x_1 + r)$, vendar $n \nmid (x_1 - r)$ in $n \nmid (x_1 + r)$ $\Rightarrow D(x_1 + r, n) \neq 1$.

Verjetnost, da uspemo v enem koraku:

Def: $r_1 \sim r_2 \Leftrightarrow r_1^2 \equiv r_2^2 \pmod{n}$ $(r_1, r_2 \neq 0)$.
To je ekvivalenčna relacija, ekvivalenčni razredi v $Z_n \backslash \{0\}$ imajo moč 4: $[r] = \{\pm r, \pm\omega r\}$.
Vsak element iz [r] nam da isto vrednost y.
Podprogram A nam vrne x, $[x] = \{\pm x, \pm\omega x\}$,
$r = \pm x$: 4 ni odgovora $r = \pm\omega x$: dobimo odgovor.

Ker izberemo r slučajno, je vsaka od teh možnosti enako verjetna $\Rightarrow$ verjetnost, da uspemo, je $1/2$.

Algoritmi za faktorizacijo števil

Poskušanje
Število n delimo z vsemi lihimi števili do $\sqrt{n}$:

$i := 3$,
until $i \leq \sqrt{n}$ **repeat**
if $i \,|\, n$, potem smo našli faktor,
else $i := i + 2$.

Algoritem je uporaben za manjše n (npr. $n \leq 10^{12}$).
Časovna zahtevnost za k bitov je $2^{k/2-1}$ deljenj.

Metoda $p-1$ (Pollard, 1974)

Podatki: n (lih, želimo faktorizirati) in B (meja)

Algoritem temelji na naslednjem preprostem dejstvu:

če je p praštevilo, ki deli n, in za vsako praštevilsko potenco q, ki deli $p-1$, velja $q \leq B$, potem $(p-1)|B!$

Primer: $B = 9$, $p = 37$, $p - 1 = 36 = 2^2 \cdot 3^2$
$2^2 \leq B, 3^2 \leq B \Rightarrow 2^2 \cdot 3^2 | 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9$

Algoritem

Podatki: n, B
1. $a := 2$
2. $j = 2, ..., B$ $(a \equiv 2^{B!} \pmod{n})$
   $a := a^j \pmod{n}$ $(\Rightarrow a \equiv 2^{B!} \pmod{p})$
3. $d = D(a-1, n)$ (Fermat: $2^{p-1} \equiv 1 \pmod{p}$)
4. Če velja $1 < d < n$, je d faktor števila n (saj $p|d$)
   sicer ni uspeha (to se zgodi, kadar je $d = 1$).

Če $B \geq \sqrt{n}$, vedno uspemo, vendar algoritem ni učinkovit.

Časovna zahtevnost

- $B - 1$ potenciranj po modulu n, za vsako rabimo $2\log_2 B$ množenj po modulu n,
- največji skupni delitelj z Evklid. alg.: $\mathcal{O}((\log n)^3)$.

Skupaj $\mathcal{O}(B \log B (\log n)^2 + (\log n)^3)$, kar pomeni, da je za $B \approx (\log n)^i$ algoritem polinomski.

Primer: $n = 143$, $B = 4$, $a \equiv 2^{2 \cdot 3 \cdot 4} \equiv 131 \pmod{143}$.
Torej je $a - 1 = 130$ in od tod $D(130, 143) = 13$.

Za varen RSA izberemo $p = 2p_1 + 1$ in $q = 2q_1 + 1$, kjer sta p_1 in q_1 praštevili.

Dixonov algoritem in kvadratno rešeto

$$(x \not\equiv \pm y \pmod{n}, x^2 \equiv y^2 \pmod{n} \implies D(x-y, n) \neq 1)$$

Sestavimo bazo faktorjev $\mathcal{B} = \{p_1, \ldots, p_B\}$, kjer so p_i "majhna" praštevila. Naj bo C malo večji kot B (npr. $C = B + 10$). Najdemo C kongruenc:

$$x_j^2 \equiv p_1^{\alpha_{1,j}} \times p_2^{\alpha_{2,j}} \times \cdots \times p_B^{\alpha_{B,j}} \pmod{n}, \quad 1 \leq j \leq C$$

Označimo $a_j := (\alpha_{1,j} \bmod 2, \ldots, \alpha_{B,j} \bmod 2)$.
Če najdemo podmnožico $\{a_1, \ldots, a_C\}$, v kateri se vektorji seštejejo v $(0, 0, \cdots, 0) \bmod 2$, potem bo produkt x_j uporabil vsak faktor iz $\mathcal{B}$ sodo mnogokrat.

Primer: $n = 15770708441$, $\mathcal{B} = \{2, 3, 5, 7, 11, 13\}$

$$8340934156^2 \equiv 3 \times 7 \pmod{n} \qquad a_1 = (0,1,0,1,0,0)$$
$$12044942944^2 \equiv 2 \times 7 \times 13 \pmod{n}, \quad a_2 = (1,0,0,1,0,1)$$
$$2773700011^2 \equiv 2 \times 3 \times 13 \pmod{n}, \quad a_3 = (1,1,0,0,0,1)$$

Iz $a_1 + a_2 + a_3 = (0,0,0,0,0,0) \bmod 2$ sledi

$$(8340934156 \times 12044942944 \times 2773700011)^2 \equiv$$
$$\equiv (2 \times 3 \times 7 \times 13)^2 \pmod{n}$$

oziroma $9503435785^2 \equiv 546^2 \pmod{n}$
in $D(9503435785 - 546, 15770708441) = 115759$.

- Linearno odvisnost med vektorji $\{a_1, a_2, \cdots, a_C\}$ poiščemo npr. z Gaussovo eliminacijo.
- $C > B + 1$: vendar imamo raje več različnih odvisnosti, da bo vsaj ena dala faktorizacijo.
- Števila x_j, za katere se da $x_j^2 \pmod n$ faktorizirati v $\mathcal{B}$, iščemo v množici $\{x_j = j + \lfloor\sqrt{n}\rfloor \mid j = 1, 2, ...\}$ z metodo **kvadratnega rešeta** (Pomerance).
- Če je $\mathcal{B}$ velik, je večja možnost, da se da neko število faktorizirati v $\mathcal{B}$, a potrebujemo več kongruenc, da najdemo linearno odvisnost. ($|\mathcal{B}| \approx \sqrt{e^{\sqrt{\ln n \ln \ln n}}}$).

Algoritmi za faktorizacijo v praksi

Kvadratno rešeto	$O(e^{(1+o(1))\sqrt{\ln n \ln \ln n}})$
Eliptične krivulje	$O(e^{(1+o(1))\sqrt{\ln p \ln \ln p}})$
Številsko rešeto	$O(e^{(1.92+o(1))(\ln n)^{1/3}(\ln \ln n)^{2/3}})$

$o(1) \to 0$, ko $n \to \infty$
p - najmanjši praštevilski faktor n

V najslabšem primeru, ko je $p \approx \sqrt{n}$, imata kvadratno rešeto in eliptične krivulje približno enako časovno zahtevnost, sicer pa je boljše kvadratno rešeto.

Faktorizacije velikih števil s kvadratnim rešetom:
($n = p \cdot q,\ p \approx q$)

leto	stevilo	bitov	metoda	opombe
1903	$2^{67} - 1$	67		F. Cole (3 leta ob ned.)
1988		250	QS	100 rac., e-posta
1994	RSA-129	425	QS	1600 rac. 8 mesecev
1999	RSA-155	512	NFS	300 del.p.+Cray; 5 mes.
2002	RSA-158	524	NFS	30 del.p.+Cray; 3 mes
2003	RSA-174	576	NFS	
2005	RSA-200	663	NFS	(55 let na eni del.p.)

Fermatova števila:

$2^{2^{11}} - 1$ eliptične krivulje: 1988 (Brent)

$2^{2^9} - 1$ številsko rešeto:
1990 (Lenstra, Lenstra, Manasse, Pollard)

Prof. Vidav je leta 1997 zastavil naslednje vprašanje (morda tudi zato, da preveri trenutne moči namiznih računalnikov): poišči prafaktorje števila

$$10^{64} + 1$$

in namignil, da so vsi prafaktorji, če jih je kaj, oblike $128k + 1$.

Večina osebnih računalnikov z Mathematica/Maple hitro najde en faktor:

$$1265011073$$

55-mestni ostanek pa povzroči težave.
V Waterlooju sem končno našel hiter računalnik (cacr: Alpha ???) ter hitro programsko opremo (glej `http://www.informatik.th-darmstadt.de/TI/LiDIA/`), ki je v manj kot 10-ih minutah našla še preostala prafaktorja

$$15343168188889137818369$$

$$515217525265213267447869906815873.$$

5. poglavje

Drugi javni kriptosistemi

- ElGamalovi kriptosistemi in Massey-Omura shema
- Problem diskretnega logaritma in napadi nanj
- Metoda velikega in malega koraka
- Pohlig-Hellmanov algoritem
- Index calculus
- Varnost bitov pri diskretnem logaritmu
- Končni obsegi in eliptične krivulje
- Eliptični kriptosistemi
- Merkle-Hellmanov sistem z nahrbtnikom
- Sistem McEliece

Javna kriptografija

L. 1976 sta Whit **Diffie** in Martin **Hellman** predstavila koncept kriptografije z javnimi ključi.

Le-ta za razliko od simetričnega sistema uporablja dva različna ključa, **zasebnega** in **javnega**.
V prejšnjem poglavju smo spoznali RSA (1978).

Taher ElGamal (1985): enkripcije z javnimi ključi in sheme digitalnih podpisov.

Varianta: algoritem za digitalni podpis
(**Digital Signature Algorithm – DSA**),
ki ga je prispevala vlada ZDA.

V razvoju javne kriptografije je bilo razbitih veliko predlaganih sistemov.

Le tri vrste so se ohranile in jih danes lahko smatramo za varne in učinkovite.

Glede na matematični problem, na katerem temeljijo, so razdeljene v tri skupine:

- **Sistemi faktorizacije celih števil** (Integer Factorization Systems) z RSA (Rivest-Adleman-Shamir) kot najbolj znanim predstavnikom,
- **Sistemi diskretnega logaritma** (Discrete Logarithem Systems), kot na primer DSA,
- **Kriptosistemi z eliptičnimi krivuljami** (Elliptic Curve Cryptosystems).

Problem diskretnega logaritma v grupi G

za dana $\alpha, \beta \in G$, kjer je red elementa α enak n, najdi $x \in \{0, \dots, n-1\}$, tako da je $\alpha^x = \beta$.

Število x se imenuje **diskretni logaritem** osnove α elementa β.

Medtem ko je diskretni logaritem (verjetno) težko izračunati (v splošnem), lahko potenco izračunamo hitro (primer enosmerne funkcije).

Problem diskretnega logaritma v grupi $\mathbb{Z}_p$

Trenutno ne poznamo nobenega polinomskega algoritma za DLP.

Praštevilo p mora imeti vsaj 150 mest (500 bitov), $p-1$ pa mora imeti vsaj en "velik" prafaktor.

ElGamalovi protokoli

Delimo jih v tri razrede:

1. protokoli za izmenjavo ključev,
2. sistemi z javnimi ključi,
3. digitalni podpisi.

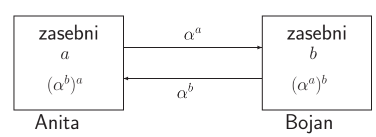
Te protokole lahko uporabimo s poljubno končno grupo G.

Osnovna razloga za uporabo različnih grup:

- operacije v nekaterih grupah so izvedene enostavneje v programih (software) in programski opremi (hardware) kot v drugih grupah,
- problem diskretnega logaritma je lahko v določeni grupi zahtevnejši kot v drugi.

Naj bo $\alpha \in G$ in naravno število n red tega elementa (t.j., $\alpha^n = 1$ in $\alpha^k \neq 1$ za vsak $k < n$).

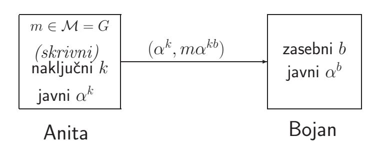
1. Izmenjava ključev
(Diffie-Hellman)



Anita in Bojan si delita skupni element grupe: $(\alpha^a)^b = (\alpha^b)^a = \alpha^{ab}$.

2. ElGamalov kriptosistem javnih ključev

(dva ključa, asimetrični sistem)



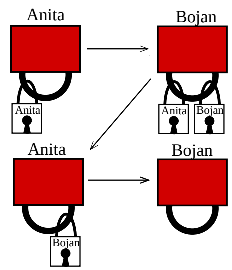
Če je $(y_1, y_2) = e_K(m, k) = (\alpha^k, m\alpha^{kb})$, potem je odšifriranje definirano z $d_K(y_1, y_2) = y_2(y_1^b)^{-1}$.

Sporočilo m lahko prebere le Bojan (s svojim b), ni pa nikjer rečeno, da mu ga je res poslala Anita (saj ni uporabila svojega zasebnega ključa).

V javni kriptografiji smatramo, da nam javni del (npr. α^k, α^b) v ničemer ne pomaga pri iskanju skrivnega/zasebnega dela (npr. k, b).

(Digitalni podpis bo obravnavan v 6. poglavju.)

Massey-Omura shema



Zgled:
za G si izberemo grupo $GF(23)^*$.

Elementi obsega $GF(23)$ so: $0, 1, \dots, 22$.

Definirajmo:
$a + b = r_1$, kjer je r_1 vsota $a + b$ mod 23.
$ab = r_2$, kjer je r_2 produkt ab mod 23.

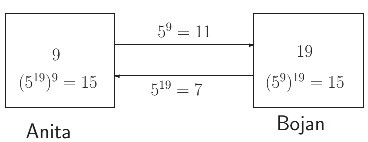
Primer: $12 + 20 = 32 = 9$, $8 \cdot 9 = 72 = 3$.

Multiplikativna grupa $GF(23)^*$

Elementi $GF(23)^*$ so elementi $GF(23)\backslash\{0\}$ in jih lahko generiramo z enim elementom:

$5^0 = 1$	$5^8 = 16$	$5^{16} = 3$
$5^1 = 5$	$5^9 = 11$	$5^{17} = 15$
$5^2 = 2$	$5^{10} = 9$	$5^{18} = 6$
$5^3 = 10$	$5^{11} = 22$	$5^{19} = 7$
$5^4 = 4$	$5^{12} = 18$	$5^{20} = 12$
$5^5 = 20$	$5^{13} = 21$	$5^{21} = 14$
$5^6 = 8$	$5^{14} = 13$	$5^{22} = 1$
$5^7 = 17$	$5^{15} = 19$	

Diffie–Hellmanov protokol v $GF(23)^*$



Anita in Bojan si sedaj delita skupen element $5^{9 \cdot 19} = 15$.

Log tabela

log	elt	log	elt	log	elt
0	1	8	16	16	3
1	5	9	11	17	15
2	2	10	9	18	6
3	10	11	22	19	7
4	4	12	18	20	12
5	20	13	21	21	14
6	8	14	13		
7	17	15	19		

Grupo G in generator α si izberemo tako, da je red elementa α velik (s tem pa je velika tudi log tabela).

Antilog tabela

elt	log	elt	log	elt	log
1	0	9	10	17	7
2	2	10	3	18	12
3	16	11	9	19	15
4	4	12	20	20	5
5	1	13	14	21	13
6	18	14	21	22	11
7	19	15	17		
8	6	16	8		

Algoritmi za računanje diskretnega logaritma

- Shankov algoritem (veliki korak – mali korak),
- Pollardov ρ-algoritem,
- Pohlig-Hellmanov algoritem,
- metoda "index calculus".

Danes si bomo ogledali samo prvega in zadnja dva.

Metoda veliki korak – mali korak:

$GF(23)^*$ z gen. 5: sestavi tabelo elementov $5^0, 5^5, 5^{10}, 5^{15}, 5^{20}$ in njihovih logaritmov.

element	1	20	9	19	12
logaritem	0	5	10	15	20

Izračunaj log(18): računaj $5 \times 18, 5^2 \times 18, \cdots$, vse dokler ne dobiš elementa iz tabele.
$5 \times 18 = 21, \quad 5^2 \times 18 = 13, \quad 5^3 \times 18 = 19.$
Iz tabele dobimo $\log(5^3 \times 18) = \log 19 = 15$.
Sledi $3 + \log 18 = 15$ oziroma $\log 18 = 12$.

$GF(89)^*$ z generatorjem 3: sestavi tabelo elementov $3^0, 3^{10}, 3^{20}, \cdots, 3^{80}$ in njihovih algoritmov.

elt	1	42	73	40	78	72	87	5	32
log	0	10	20	30	40	50	60	70	80

Izračunaj log(36): računaj $3 \times 36, 3^2 \times 36, \cdots$, vse dokler ne dobiš elementa iz tabele.
$3 \times 36 = 19$, $3^3 \times 36 = 82$, $3^5 \times 36 = 26$, $3^2 \times 36 = 57$, $3^4 \times 36 = 68$, $3^6 \times 36 = 78$.
Iz tabele preberemo $\log(3^6 \times 36) = \log 78$. Sledi $6 + \log 36 = 40$ oziroma $\log 36 = 34$.

Čim daljša je tabela, ki jo sestavimo, tem dlje časa jo je treba računati (enkratni strošek), po drugi strani pa hitreje naletimo na element v krajši tabeli.

Običajno sestavimo tabelo velikosti $m = \lfloor \sqrt{|G|} \rfloor$ in za iskanje potrebujemo $O(m)$ časa.

Pollardov ρ algoritem (s Floydovim algoritmom za iskanje ciklov)

Ima isto časovno zahtevnost kot metoda veliki korak – mali korak, porabi pa le malo spomina.

Pohlig-Hellmanov algoritem

$$p - 1 = \Pi_{i=1}^{k} p_i^{c_i}$$

za različna praštevila p_i. Vrednost $a = \log_\alpha \beta$ je natanko določena po modulu $p - 1$.

Najprej izračunamo $a \bmod p_i^{c_i}$ za vsak $i = 1, \ldots, k$ in nato izračunamo $a \bmod (p-1)$ po kitajskem izreku o ostankih.

Predpostavimo, da je q praštevilo in c največje naravno število, za katero velja

$$p - 1 \equiv 0 \pmod{q^c}.$$

Kako izračunamo

$$x = a \bmod q^c, \quad \text{kjer je } 0 \le x \le q^c - 1?$$

Zapišimo x v številskem zapisu z osnovo q:

$$x = \sum_{i=0}^{c-1} a_i q^i, \quad \text{kjer je } 0 \le a_i \le q - 1.$$

Od tod dobimo

$$a = a_0 + a_1 q + \cdots + a_{c-1} q^{c-1} + s q^c,$$

kjer je s neko naravno število in $a = a_0 + Kq$.
a_0 izračunamo iz naslednje identitete

$$\beta^{(p-1)/q} \equiv \alpha^{a_0 (p-1)/q} \bmod p.$$

Dokažimo slednjo kongruenco:

$$\begin{aligned} \beta^{(p-1)/q} &\equiv (\alpha^a)^{(p-1)/q} \bmod p \\ &\equiv (\alpha^{a_0 + Kq})^{(p-1)/q} \mod p \\ &\equiv \alpha^{a_0 (p-1)/q} \, \alpha^{(p-1)K} \mod p \\ &\equiv \alpha^{a_0 (p-1)/q} \mod p. \end{aligned}$$

Najprej torej izračunamo

$$\beta^{(p-1)/q} \mod p.$$

Če je $\beta^{(p-1)/q} \equiv 1 \mod p$, je $a_0 = 0$, sicer pa zaporedoma računamo

$$\gamma = \alpha^{(p-1)/q} \mod p, \quad \gamma^2 \mod p, \quad \ldots,$$

vse dokler ne dobimo

$$\gamma^i \mod p = \beta^{(p-1)/q} \mod p$$

in je $a_0 = i$.

Sedaj moramo določiti $a_1, \ldots, a_{c-1}$
(če je $c > 1$). Naj bo

$$\beta_j = \beta\alpha^{a_0+a_1q+\cdots+a_{j-1}q^{j-1}} \mod p,$$

za $0 \leq j \leq c-1$. Tokrat velja splošnejša identiteta

$$(\beta_j)^{(p-1)/q^{j+1}} \equiv \alpha^{a_j(p-1)/q} \mod p,$$

ki jo dokažemo na enak način kot prejšnjo.

Za dani β_j ni težko izračunati a_j, omenimo pa še rekurzijo

$$\beta_{j+1} = \beta_j\alpha^{-a_jq^j} \mod p.$$

Za dano faktorizacijo števila n je časovna zahtevnost Pohlig-Hellmanovega algoritma $O(\sum_{i=0}^{k} c_i(\log n + \sqrt{p_i}))$ grupnih multiplikacij.

Primer: naj bo $p = 251$, potem je

$$n = p - 1 = 250 = 2 \cdot 5^3.$$

Naj bo $\alpha = 71$ in $\beta = 210$,
torej želimo izračunati $a = \log_{71} 210$.

Modul 2: $\gamma_0 = 1$,

$$\gamma_1 \equiv \alpha^{250/2} \equiv 250 \mod p$$

in

$$\beta^{250/2} \equiv 250 \mod p,$$

torej $a_0 = 1$ in $\log_{71} 210 \equiv 1 \pmod 2$.

Modul 5: $\gamma_0 = 1$,

$$\gamma_1 \equiv \alpha^{250/5} \equiv 20 \mod p$$

in

$$\beta^{250/5} \equiv 149 \mod p,$$

torej $a_0 = 2$. ...
$a_1 = 4 = \log_{20} 113$ in $a_2 = 2 = \log_{20} 149$,

$\log_{71} 210 \equiv 2 + 4 \cdot 5 + 2 \cdot 5^2 \equiv 72 \pmod{125}$.

Končno nam CRT da $\log_{71} 210 = 197$.

Metoda index calculus

$GF(23)^*$ z generatorem 5.
Izberi bazo 'majhnih' faktorjev: $B = \{-1, 2, 3\}$
in sestavi tabelo njihovih logaritmov:

elt	-1	2	3
log	11	2	16

Iščemo logaritem elementa β (Las Vegas).
Poišči 'gladko' potenco elementa β,
tj. β^x, ki se da razstaviti na faktorje iz B.

Izračunaj log(13): $13^2 = 169 = 2^3 \iff$
$\log 13^2 = \log 2^3 \iff 2\log 13 \equiv 3\log 2 \iff$
$2\log 13 \equiv 6 \pmod{22}$

Sledi $\log 13 \equiv 3$ ali $14 \pmod{22}$.
Preverimo $\log 13 = 14$.

Izračunaj log(14):
$14^3 = 2^3 7^3 = 2^3 \cdot 21 = 2^3 \cdot (-2) = -2^4$.
$3\log 14 = \log(-2^4) = \log(-1) + \log 2^4 = 11 + 4 \cdot 2 = 19$,
$\log 14 = \frac{19}{3} = 19 \cdot (-7) = (-3)(-7) = 21$.

Izračunaj log(15):
$15^3 = 3^3 \cdot 5^3 = 3^3 \cdot 2 \cdot 5 = (-1) \cdot 2 \cdot 3$,
$3\log 15 = \log(-1) + \log 2 + \log 3 = 11 + 2 + 16 = 29 = 7$,
$\log 15 = \frac{7}{3} = 7(-7) = -49 = -5 = 17$.

Izračunaj log(7):
$7^3 = 49 \cdot 7 = 3 \cdot 7 = 21 = (-1) \cdot 2$,
$3\log 7 = \log(-1) + \log 2 = 11 + 2 = 13$,
$\log 7 = \frac{13}{3} = 13 \cdot (-7) = 63 = -3 = 19$.

Še en primer: $GF(89)^*$ z gen. 3.

tabela logaritmov:

elt	-1	2	3	5
log	44	16	1	70

Izračunaj log(7):
$7^3 = 76 = 2^2 \cdot 19$, $7^5 = 3 \cdot 5^2$,
$5 \log 7 = \log 3 + 2 \log 5 = 1 + 2 \cdot 70 = 141 = 53$,
$\log 7 = \frac{53}{5} = 53 \cdot (-35) = 81$.

Izračunaj log(53):
$53^3 = 3 \cdot 23$, $53^5 = 2^2 \cdot 17$, $53^7 = 2 \cdot 3^2$,
$7 \log 53 = \log 2 + 2 \log 3 = 16 + 2 = 8$,
$\log 53 = \frac{18}{7} = 18 \cdot (-25) = 78$.

Metoda index calculus (splošno)

1. Izberi bazo faktorjev $\mathcal{B} = \{p_1, \dots, p_t\}$, tako da se da dovolj veliko število elementov grupe G dovolj hitro razstaviti v $\mathcal{B}$.

2. Poišči $t + 10$ lineranih zvez z logaritimi elementov iz $\mathcal{B}$:

izberi število $k < n$, izračunaj α^k in ga poskusi zapisati kot

$$\alpha^k = \Pi_{i=1}^t \, p_i^{c_i} \quad \Longleftrightarrow \quad k \equiv \sum_{i=1}^t c_i \log p_i \bmod p - 1.$$

3. Sestavi tabelo logaritmov elementov iz $\mathcal{B}$.

4. Izberi naključno število $k \in \{1, \dots, n\}$, izračunaj $\beta\alpha^k$ in ga poskusi zapisati kot

$$\beta\alpha^k = \Pi_{i=1}^t \, p_i^{d_i}.$$

Končno dobimo

$$\log_\alpha \beta = \Big(\sum_{i=1}^t d_i \log_\alpha p_i - k\Big) \mod n.$$

Obstajajo različni slučajni algoritmi za metodo Index calculus. Ob sprejemljivih predpostavkah je njihova časovna zahtevnost za pripravljalno fazo

$$\mathcal{O}\Big(e^{1+o(1))\sqrt{\log p \log\log p}}\Big),$$

za izračun vsakega posameznega logaritma pa

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

Varnost bitov pri diskretnem log.

Podatki: (p, α, β, i),
kjer je p praštevilo, α primitiven element grupe $\mathbb{Z}_p^*$ in i poljubno naravno število, ki je manjše ali enako $\log_2(p-1)$.

Cilj: izračunaj i-ti bit (oznaka: $L_i(\beta)$) logaritma $\log_\alpha \beta$ za fiksna α in p (začnemo šteti z desne).

$L_1(\beta)$ lahko najdemo s pomočjo Eulerjevega kriterija za kvadratne ostanke po modulu p:

Ker je $w^2 \equiv x^2 \pmod p \iff p|(w-x)(w+x)$ oziroma $w \equiv \pm x \pmod p$, velja

$$\{x^2 \bmod p \mid x \in \mathbb{Z}_p^*\} = \Big\{\alpha^{2i} \bmod p \;\Big|\; 0 \le i \le \frac{p-3}{2}\Big\}.$$

Od tod pa sledi

$$\beta \text{ kvadratni ostanek} \iff 2|\log_\alpha \beta \;\text{ tj. }\; L_1(\beta) = 0,$$

element β pa je kvadratni ostanek če in samo če je

$$\beta^{(p-1)/2} \equiv 1 \pmod p.$$

Sedaj pa si oglejmo še primer, ko je $i > 1$.

Naj bo $p - 1 = 2^s t$, kjer je t liho število.
Potem za $i \le s$ ni težko izračunati $L_i(\beta)$,
verjetno pa je težko izračunati $L_{s+1}(\beta)$,
kajti v nasprotnem primeru bi bilo možno uporabiti hipotetični podprogram za rešitev DLP v $\mathbb{Z}_p$.

Zgornjo trditev bomo dokazali za $s = 1$ oziroma $p \equiv 3 \pmod 4$. Tedaj sta kvadratna korena iz β po modulu p števili $\pm\beta^{(p+1)/4} \bmod p$.

Za $\beta \neq 0$ velja $L_1(\beta) \neq L_1(p - \beta)$, saj iz

$$\alpha^a \equiv \beta \pmod p \implies \alpha^{a+(p-1)/2} \equiv -\beta \pmod p,$$

ker je $(p-1)/2$ liho število.

Če je $\beta = \alpha^a$ za neko sodo potenco a, potem je

$$\alpha^{a/2} \equiv \beta^{(p+1)/4} \text{ ali } -\beta^{(p+1)/4} \pmod p.$$

Katera izmed teh dveh možnosti je pravilna lahko ugotovimo iz $L_2(\beta)$, saj velja

$$L_2(\beta) = L_1(\alpha^{a/2}).$$

Algoritem za računanje diskretnega logaritma v $\mathbb{Z}_p$ za $p \equiv 3 \pmod 4$:

1. $x_0 = L_1(\beta)$, $\beta = \beta/\alpha^{x_0} \mod p$, $i := 1$
2. **while** $\beta \neq 1$ **do**
3. $\quad x_i = L_2(\beta)$
4. $\quad \gamma = \beta^{(p+1)/4} \bmod p$
5. $\quad$ **if** $L_1(\gamma) = x_i$ **then** $\beta = \gamma$
6. $\quad$ **else** $\beta = p - \gamma$
7. $\quad \beta = \beta/\alpha^{x_i} \bmod p$, $i := i + 1$

Dokaz pravilnosti zgornjega algoritma:

Naj bo
$$x = \log_\alpha \beta = \sum_{i \geq 0} x_i 2^i$$
in definirajmo za $i \geq 0$:
$$Y_i = \left\lfloor \frac{x}{2^{i+1}} \right\rfloor$$
in naj bo β_0 vrednost β v koraku 1.
Za $i \geq 1$, pa naj bo β_i vrednost β v zadnjem koraku pri i-ti iteraciji **while** zanke.

Z indukcijo pokažemo za vsak $i \geq 0$:
$$\beta_i \equiv \alpha^{2Y_i} \pmod p.$$
Iz $2Y_i = Y_{i-1} - x_i$ sledi $x_{i+1} = L_2(\beta_i)$ za $i \geq 0$
ter končno še $x_0 = L_1(\beta)$. ■

Končni obsegi

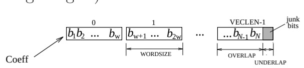
Primer končnega obsega: $\mathrm{GF}(2^4)$

Izberimo $f(x) = 1 + x + x^4 \in \mathrm{GF}(2)[x]$.
Naj bo $a_0 + a_1x + a_2x^2 + a_3x^3 = (a_0, a_1, a_2, a_3)$.
Elementi obsega $\mathrm{GF}(2^4)$ so:

(1000)	(1100)	(1010)	(1111)
(0100)	(0110)	(0101)	(1011)
(0010)	(0011)	(1110)	(1001)
(0001)	(1101)	(0111)	(0000)

Element končnega obsega v predstavimo kot vektor.
V hardwaru ponavadi delamo v $\mathrm{GF}(2)$, torej je v 01-vektor, ki ga hranimo v registru dolžine n,
in je vsota vektorjev enaka XOR po koordinatah.

V softwaru pa hranimo binarni vektor v v besedah (npr. long integers)



V splošnem obstaja veliko število različnih baz za $\mathrm{GF}(q^m)$ nad $\mathrm{GF}(q)$.

Definirajmo operaciji '+' in '×' v $\mathrm{GF}(p^n)$:
$$(a_0, \ldots, a_{n-1}) + (b_0, \ldots, b_{n-1}) = (c_0, \ldots, c_{n-1}),$$
kjer je $c_i = a_i + b_i \pmod p$.
$$(a_0, \ldots, a_{n-1}) \times (b_0, \ldots, b_{n-1}) = (r_0, \ldots, r_{n-1}),$$
kjer je $(r_0, \ldots, r_{n-1})$ ostanek produkta $(a_0, \ldots, a_{n-1}) \times (b_0, \ldots, b_{n-1})$ pri deljenju z nerazcepnim polinomom $f(x)$ stopnje n.

Primer: $(1011) + (1001) = (0010)$

$(1011) \times (1001)$
$= (1 + x^2 + x^3)(1 + x^3) = 1 + x + x^5 + x^6$
$= (x^4 + x + 1)(x^2 + x) + (1 + x + x^2 + x^3)$
$= (1111)$

Končni obseg $\mathrm{GF}(2^4)^*$: izberemo $f(x) = 1 + x + x^4$.
$\mathrm{GF}(2^4)^*$ je generiran z elementom $\alpha = x$.

$\alpha_0 = (1000)$	$\alpha_8 = (1010)$
$\alpha_1 = (0100)$	$\alpha_9 = (0101)$
$\alpha_2 = (0010)$	$\alpha_{10} = (1110)$
$\alpha_3 = (0001)$	$\alpha_{11} = (0111)$
$\alpha_4 = (1100)$	$\alpha_{12} = (1111)$
$\alpha_5 = (0110)$	$\alpha_{13} = (1011)$
$\alpha_6 = (0011)$	$\alpha_{14} = (1001)$
$\alpha_7 = (1101)$	$\alpha_{15} = \alpha^0 = 1$

Log tabela

log	elt	log	elt
0	(1000)	8	(1010)
1	(0100)	9	(0101)
2	(0010)	10	(1110)
3	(0001)	11	(0111)
4	(1100)	12	(1111)
5	(0110)	13	(1011)
6	(0011)	14	(1001)
7	(1101)		

Zech log tabela

$1+\alpha^i=\alpha^{z(i)}$			
i	$z(i)$	i	$z(i)$
∞	0	7	9
0	∞	8	2
1	4	9	7
2	8	10	5
3	14	11	12
4	1	12	11
5	10	13	6
6	13	14	3

Računanje v **polinomski bazi** je odvisno od izbire polinoma $f(x)$.

Da bi pospešili redukcijo (po množenju ali kvadriranju), si ponavadi izberemo za $f(x)$ nerazcepni **trinom** (to je $x^n + x^m + 1$).

Na žalost nerazcepni trinomi ne obstajajo za poljubno velikost končnega obsega. V tem primeru uporabljamo *pentonome* ali *helptonome*.

Znano je, da ima vsak končni obseg GF(p^n) bazo nad podobsegom GF(p) naslednje oblike:

$$B = \{\beta, \beta^p, \cdots, \beta^{p^{n-1}}\}.$$

V praksi so takšne baze, ki jih imenujemo **normalne**, zelo praktične za hardwersko implementacijo množenja v obsegu GF(p^n), še posebej, kadar je $p = 2$.

Implementacija

Potenciranje opravimo z algoritmom kvadriraj in množi:

$$\alpha^{21} = (\alpha)(\alpha^4)(\alpha^{16})$$

Najprej izračunamo faktorje α, α^2, α^4, α^8, α^{16}, in jih nato zmnožimo.

Namesto 20 množenj smo jih potrebovali le 6.

Ali je lahko kvadriranje hitrejše od (splošnega) množenja?

NE!

$$ab = \frac{(a+b)^2 - a^2 - b^2}{2}.$$

Če je kvadriranje 'lahko', potem tudi splošno množenje ni dosti težje od seštevanja.

DA!

V normalni bazi končnega obsega GF(2^n) je kvadriranje ciklični zamik, množenje pa ostane težko v splošnem.

V praksi so normalne baze zelo praktične za hardwarsko implementacijo množenja v obsegu GF(p^n), še posebej, kadar je $p = 2$. in je kvadriranje *cikličen zamik*.

S tem namenom so Mullin, Onyszchuk, Vanstone in Wilson [MOVW88] definirali **optimalne normalne baze** (ONB) kot tiste baze, katerih število koeficientov v reprezentaciji elementov β^{p^i+1}, $i = 0, \cdots, n-1$ glede na bazo B je natanko $2n-1$. Z drugimi besedami $n \times n$-razsežna matrika $T = (t_{mk})$, definirana z $\beta\beta^{p^m} = \sum_{k=0}^{n-1} \beta^{p^k} t_{mk}$, vsebuje natanko $2n-1$ neničelnih elementov.

Ni težko preveriti, da je število $2n-1$ absolutna spodnja meja (DN).

Izrek (Mullin et al. [MOVW]):
Obseg $\mathrm{GF}(p^n)$ *vsebuje optimalno normalno bazo v naslednjih primerih*

(i) $n+1$ *je praštevilo in* p *primitiven element obsega* $\mathrm{GF}(n+1)$,

(ii) $p=2$, $2n+1$ *je praštevilo in bodisi 2 je primitiven element obsega* $\mathrm{GF}(2n{+}1)$ *bodisi* n *je lih in 2 generira kvadratne ostanke obsega* $\mathrm{GF}(2n{+}1)$.

Mullin et al. [MOVW] so postavili hipotezo, da za $p=2$ obstajajo optimalne normalne baze natanko tedaj kadar velja en izmed pogojev (i) in (ii).

Hipotezo sta leta 1992 dokazala Gao in Lenstra.

Grupa na eliptični krivulji

Za kriptografijo sta jo leta 1985 prva predlagala Neal Koblitz in Victor Miller.

Eliptična krivulja E nad obsegom $\mathbb{Z}_p$ je definirana z Weierstrassovo enačbo:

$$\boldsymbol{y^2 = x^3 + ax + b} \qquad (1)$$

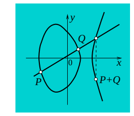
kjer sta $a,b \in \mathbb{Z}_p$ in $4a^3+27b^2 \not\equiv 0 \pmod p$ ($GF(2^m)$: $y^2+xy=x^3+ax^2+b$).

$E(\mathbb{Z}_p) := \{(x,y) \mid x,y \in \mathbb{Z}_p,\ \text{ki ustrezajo (1)}\} \cup \mathcal{O}$.

Pravilo za seštevanje

1. $P=(x_1,y_1)$, $Q=(x_2,y_2) \in E(\mathbb{Z}_p)$, kjer $P \neq -Q := (x_2,-y_2)$.
   Potem je $P+Q=(x_3,y_3)$, kjer je
   $x_3 = \lambda^2 - x_1 - x_2$, $\quad y_3 = \lambda(x_1-x_3) - y_1$, in

$$\lambda = \begin{cases} \dfrac{y_2-y_1}{x_2-x_1} & \text{; za } P \neq Q \\[2mm] \dfrac{3x_1^2+a}{2y_1} & \text{; za } P = Q. \end{cases}$$



2. $P+\mathcal{O} = \mathcal{O}+P = P \quad$ in $\quad P+(-P) = \mathcal{O}$ za vsak $P \in E(\mathbb{Z}_p)$.

Množica $E(\mathbb{Z}_p)$ je sestavljena iz točk (x,y), $x,y \in \mathbb{Z}_p$, ki ustrezajo zgornji enačbi, vključno s točko neskončno $\mathcal{O}$.

Izrek (Hasse).

$$p+1-2\sqrt{p} \leq |E| \leq p+1+2\sqrt{p}$$

Schoofov algoritem izračuna $|E|$ v $O((\log p)^8)$ bitnih operacijah.

Grupa E je izomorfna $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$, kjer je $n_2 | n_1$ in $n_2 | (p-1)$, tako da lahko najdemo ciklično podgrupo $\mathbb{Z}_{n_1}$, ki jo uporabimo za ElGamalov kriptosistem.

Podeksponentno metodo **index calculus** zaenkrat ne znamo uporabiti pri DLP na eliptični grupi (razen če ni eliptična krivulja supersingularna).

Zato si lahko izberemo eliptično krivuljo s ciklično podgrupo velikosti (samo) okoli 2^{160}.

Primer: EC nad $\mathrm{GF}(2^4)$

- Naj bo $\mathrm{GF}(2^4)$ generiran s korenom $\alpha = x$ nerazcepnega polinoma $f(x) = 1+x+x^4$.
- $E_1(\mathrm{GF}(2^4))$ $= \{(x,y) : y^2+xy = x^3+\alpha^4x^2+1\} \cup \{\mathcal{O}\}$.
- $E_1(\mathrm{GF}(2^4))$ tvori grupo za seštevanje z $\mathcal{O}$ kot identiteto.

Rešitve enačbe: $y^2+xy = x^3+\alpha^4x^2+1$ nad $\mathrm{GF}(2^4)$

$(0,1)$	
$(1,\alpha^6)$	$(1,\alpha^{13})$
(α^3,α^8)	(α^3,α^{13})
(α^5,α^3)	(α^5,α^{11})
(α^6,α^8)	(α^6,α^{14})
(α^9,α^{10})	(α^9,α^{13})
(α^{10},α^1)	(α^{10},α^8)
$(\alpha^{12},0)$	$(\alpha^{12},\alpha^{12})$

Primer seštevanja v $E_1(\mathrm{GF}(2^4))$:

Naj bo $P_1 = (\alpha^6, \alpha^8)$, $P_2 = (\alpha^3, \alpha^{13})$.

- $P_1 + P_2 = (x_3, y_3)$:

$$\begin{aligned} x_3 &= \left(\frac{\alpha^8+\alpha^{13}}{\alpha^6+\alpha^3}\right)^2 + \frac{\alpha^8+\alpha^{13}}{\alpha^6+\alpha^3} + \alpha^6 + \alpha^3 + \alpha^4 \\ &= \left(\frac{\alpha^3}{\alpha^2}\right)^2 + \frac{\alpha^3}{\alpha^2} + \alpha^2 + \alpha^4 = 1 \end{aligned}$$

$$\begin{aligned} y_3 &= \left(\frac{\alpha^8+\alpha^{13}}{\alpha^6+\alpha^3}\right)(\alpha^6+1) + 1 + \alpha^8 \\ &= \left(\frac{\alpha^3}{\alpha^2}\right)\alpha^{13} + \alpha^2 = \alpha^{13} \end{aligned}$$

- $2P_1 = (x_3, y_3)$:

$$\begin{aligned} x_3 &= (\alpha^6)^2 + \frac{1}{(\alpha^6)^2} \\ &= \alpha^{12} + \alpha^3 = \alpha^{10} \end{aligned}$$

$$\begin{aligned} y_3 &= (\alpha^6)^2 + \left(\alpha^6 + \frac{\alpha^8}{\alpha^6}\right)\alpha^{10} + \alpha^{10} \\ &= \alpha^3 + (\alpha^6+\alpha^2)\alpha^{10} = \alpha^8 \end{aligned}$$

Še en primer EC nad $\mathrm{GF}(2^4)$

- Naj bo $\mathrm{GF}(2^4)$ generiran s korenom $\alpha = x$ nerazcepnega polinoma $f(x) = 1 + x + x^4$.
- $E_2(\mathrm{GF}(2^4))$ $= \{(x,y) : y^2 + \alpha^6 y = x^3 + \alpha^3 x + 1\} \cup \{\mathcal{O}\}$.
- $E_2(\mathrm{GF}(2^4))$ tvori grupo za seštevanje z $\mathcal{O}$ kot identiteto.

Iščemo rešitve enačbe

$$y^2 + \alpha^6 y = x^3 + \alpha^3 x + 1$$

nad $\mathrm{GF}(2^4)$. Ta enačba ima samo 8 rešitev:

(α^2, α^8)	(α^2, α^{14})
$(\alpha^{10}, 1)$	$(\alpha^{10}, \alpha^{13})$
$(\alpha^{11}, 0)$	(α^{11}, α^6)
(α^{13}, α^5)	(α^{13}, α^9)

Primer: EC nad GF(23)

- Naj bo $p = 23$.
- $y^2 = x^3 + x + 1$, (i.e., $a = 1$, $b = 1$). Velja: $27a^3 + 16b^2 = 3\cdot 1^3 + 16\cdot 1^3 = 19 \neq 0$ v GF(23).
- $E_3(\mathrm{GF}(23)) = \{(x,y) : y^2 = x^3 + x + 1\} \cup \{\mathcal{O}\}$.
- $E_3(\mathrm{GF}(23))$ tvori grupo za seštevanje z $\mathcal{O}$ kot identiteto.

Rešitve enačbe $y^2 = x^3 + x + 1$ nad $\mathbb{Z}_{23}$:

(0, 1)	(6, 4)	(-11,-4)
(0,-1)	(6,-4)	(-10, 7)
(1, 7)	(7, 11)	(-10,-7)
(1,-7)	(7,-11)	(-6, 3)
(3, 10)	(9, 7)	(-6,-3)
(3,-10)	(9,-7)	(-5, 3)
(4, 0)	(11, 3)	(-5,-3)
(5, 4)	(11,-3)	(-4, 5)
(5,-4)	(-11,4)	(-4,-5)

Primera seštevanja na $E_3(\mathrm{GF}(23))$

1. $P_1 = (3, 10)$, $P_2 = (9, 7)$,
   $P_1 + P_2 = (x_3, y_3)$.

   $\lambda = \frac{7-10}{9-3} = \frac{-3}{6} = \frac{-1}{2} = 11 \in \mathbb{Z}_{23}$.

   $x_3 = 11^2 - 3 - 9 = 6 - 3 - 9 = -6$,
   $y_3 = 11(3 - (-6)) - 10 = 11(9) - 10$
   $= 89 = 20 = -3$.
   Sledi $P_1 + P_2 = (-6, -3)$.

2. $P_1 = (3, 10)$, $2P_1 = (x_3, y_3)$,

   $\lambda = \frac{3(3^2)+1}{20} = \frac{5}{20} = \frac{1}{4} = 6$.

   $x_3 = 6^2 - 6 = 30 = 7$,
   $y_3 = 6(3 - 7) - 10 = -24 - 10 = -11$.

   Sledi $2P_1 = (7, -11)$.

$P = (0, 1)$ je generator:

P=(0, 1)	
$2P$=(6,-4)	$15P$=(9, 7)
$3P$=(3,-10)	$16P$=(-6,3)
$4P$=(-10,-7)	$17P$=(1,7)
$5P$=(-5,3)	$18P$=(12,-4)
$6P$=(7,11)	$19P$=(-4,5)
$7P$=(11,3)	$20P$=(5,4)
$8P$=(5,-4)	$21P$=(11,-3)
$9P$=(-4,-5)	$22P$=(7 -11)
$10P$ =(12,4)	$23P$=(-5,-3)
$11P$=(1,-7)	$24P$=(-10,7)
$12P$=(-6,-3)	$25P$=(3,10)
$13P$=(9,-7)	$26P$=(6,4)
$14P$=(4,0)	$27P$=(0,-1)

Log – antilog tabela

log	elt	log	elt
0	$\mathcal{O}$	14	(4,0)
1	(0,1)	15	(9,7)
2	(6,-4)	16	(-6,3)
3	(3,-10)	17	(1,7)
4	(-10,-7)	18	(-11,-4)
5	(-5,3)	19	(-4,5)
6	(7,11)	20	(5,4)
7	(11,3)	21	(11,-3)
8	(5,-4)	22	(7,-11)
9	(-4,-5)	23	(-5,-3)
10	(-11,4)	24	(-10,7)
11	(1,-7)	25	(3,10)
12	(-6,-3)	26	(6,4)
13	(9,-7)	27	(0,-1)

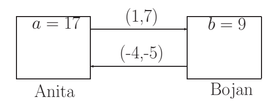
Antilog – log tabela

elt	log	elt	log
$\mathcal{O}$	0	(9,7)	15
(0,1)	1	(9,-7)	13
(0,-1)	27	(11,3)	7
(1,7)	17	(11,-3)	21
(1,-7)	11	(-11,4)	10
(3,10)	25	(-11,-4)	18
(3,-10)	3	(-10,7)	24
(4,0)	14	(-10,-7)	4
(5,4)	20	(-6,3)	16
(5,-4)	8	(-6,-3)	12
(6,4)	26	(-5,3)	5
(6,-4)	2	(-5,-3)	23
(7,11)	6	(-4,5)	19
(7,-11)	22	(-4,-5)	9

Diffie–Hellmanov protokol nad $E(\mathrm{GF}(23))$

Javni parametri:

$$y^2 = x^3 + x + 1$$
$$P = (0, 1)$$



- Anita izračuna $17P = (1, 7)$,
- Bojan izračuna $9P = (-4, -5)$,
- Anita izračuna $17(-4, -5) = (6, 4)$,
- Bojan izračuna $9(1, 7) = (6, 4)$.

Anita in Bojan imata skupno točko $(6, 4)$.

Računanje logaritmov

Izračunaj $\log_P(9, 7)$**.**

Izračunaj naslednjo tabelo:

elt	(0,1)	(7,11)	(-6,-3)	(12,-4)	(-10,7)
log	1	6	12	18	24

Če je $k = \log_P(9, 7)$, potem velja $kP = (9, 7)$.

- Računamo $(9, 7) + P$, $(9, 7) + 2P$, $(9, 7) + 3P, \dots$, vse, dokler ne dobimo element iz tabele.
- Tako dobimo: $(9, 7) + 3P = (12, -4)$.
- Iz tabele preberemo $(12, -4) = 18P$.
- Sledi $(9, 7) + 3P = 18P$ oziroma $(9, 7) = 15P$, torej $k = 15$.

- Če je $|E(\mathrm{GF}(q))| = n$, lahko posplošimo metodo za $E(\mathrm{GF}(23))$ na naslednji način:
  - naredi tabelo (i, iP) velikosti $\sqrt{n}$,
  - za iskanje logaritma elementa v tej tabeli potrebujemo največ $\sqrt{n}$ seštevanj točk.
- Če je $q \approx 10^{40}$, potem je $|E(\mathrm{GF}(q))| \approx 10^{40}$ in ima tabela 10^{20} vrstic.

  To je očitno popolnoma nedosegljivo.

Merkle-Hellmanov sistem z nahrbtnikom

Merkle in Hellman sta leta 1978 predlagala ta sistem, že leta 1980 pa ga je razbil Shamir s pomočjo Lenstrinega algoritma za celoštevilčno programiranje (angl. integer programming).

Njegovo iterativno varianto pa je razbil malo kasneje Brickell.

Drugačen sistem z nahrbtnikom je predlagal Chor, razbil pa ga je Rivest.

Problem "podmnožica za vsoto"

Podatki: $I = (s_1, \dots, s_n, T)$, *T je* **ciljna vsota**, *naravna števila s_i pa so* **velikosti**.

Vprašanje: *Ali obstaja tak binarni vektor $\underline{x} = (x_1, \dots, x_n)$, za katerega velja $\sum_{i=1}^{n} x_i s_i = T$?*

Ta odločitveni problem je NP-poln:
– polinomski algoritem ni znan,
– isto velja tudi za ustrezen iskalni problem.

Ali za kakšno podmnožico problemov morda obstaja polinomskim algoritem?

Zaporedje $(s_1, \dots, s_n)$ je **super naraščajoče**, če velja

$$s_j > \sum_{i=1}^{j-1} s_i \qquad \text{za } 2 \le j \le n.$$

Če je seznam velikosti super naraščajoč, potem lahko iskalno varianto zgornjega problema rešimo v času $O(n)$, rešitev $\underline{x}$ (če obstaja) pa je enolična.

Opišimo tak algoritem:

1. **for** $i = n$ **downto** 1 **do**
2. **if** $T \ge s_i$ **then**
3. $T = T - s_i$, $x_i = 1$
4. **else** $x_i = 0$
5. **if** $T = 0$ **then** $\underline{x} = (x_1, \dots, x_n)$ je rešitev
6. **else** ni rešitve.

Naj bo $\underline{s} = (s_1, \dots, s_n)$ super naraščajoč in

$$e_{\underline{s}} : \{0,1\}^n \longrightarrow \left\{0, \dots, \sum_{i=1}^{n} s_i\right\}$$

funkcija, definirana s pravilom

$$e_{\underline{s}}(x_1, \dots, x_n) = \sum_{i=1}^{n} x_i s_i.$$

Ali lahko to funkcijo uporabimo za enkripcijo?

Ker je $\underline{s}$ super naraščajoče zaporedje, je $e_{\underline{s}}$ injekcija, zgoraj opisani algoritem pa lahko uporabimo za dekripcijo.

Sistem **ni varen**, saj dekripcijo lahko opravi prav vsak.

Morda pa lahko transformiramo super naraščajoče zaporedje tako, da izgubi to lastnost in edino Bojan lahko opravi inverzno operacijo, da dobi super naraščajoče zaporedje.

Če napadalec Oskar ne pozna te transformacije, ima pred seboj primer (na videz) splošnega problema, ki ga mora rešiti, če hoče opraviti dekripcijo.

En tip takih transformacij se imenuje **modularna transformacija**. Izberemo si tak praštevilski modul p, da je

$$p > \sum_{i=1}^{n} s_i$$

ter število a, $1 \le a \le p-1$. Naj bo

$$t_i = \; a s_i \bmod p, \qquad \text{za } 1 \le i \le n.$$

Seznam $\underline{t} = (t_1, \dots, t_n)$ je javni ključ, ki ga uporabimo za enkripcijo, vrednosti a in p, ki definirata modularno transformacijo, pa sta tajni.

Zakaj smo si izbrali za p praštevilo?

Zakaj je bil ta sistem sploh zanimiv?

Primer: Naj bo

$$s = (2, 5, 9, 21, 45, 103, 215, 450, 946)$$

tajni super naraščajoči seznam velikosti.

Za $p = 2003$ in $a = 1289$ dobimo javni seznam velikosti

$$\underline{t} = (575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570).$$

Anita zašifrira sporočilo $\underline{x} = (1, 0, 1, 1, 0, 0, 1, 1, 1)$:

$$y = 575 + 1586 + 1030 + 721 + 1183 + 1570 = 6665$$

ter ga pošlje Bojanu, ki najprej izračuna
$z = a^{-1} y \bmod p = 1643$ in nato
reši problem podmnožice zaporedja $\underline{s}$ za vsoto z.

6. poglavje

Sheme za digitalne podpise

- uvod (podpis z RSA sistemom)
- ElGamalov sistem za digitalno podpisovanje
- Digital Signature Standard
- napadi
- enkratni podpis
- podpisi brez možnosti zanikanja
- Fail-stop podpisi

Digitalni podpis je nadomestek za lastnoročni podpis pri elektronski izmenjavi in digitalnemu hranjeju podatkov.

Konceptualno se način zapisovanja informacij ni dramatično spremenil.

Medtem ko smo prej shranjevali in prenašali informacije na papirju, jih sedaj hranimo na magnetnih in drugih medijih ter jih prenašamo preko telekomunikacijskih sistemov (tudi brezžičnih).

Bistveno pa se je spremenila možnost kopiranja in spreminjanja informacij.

Zlahka naredimo na tisoče kopij neke informacije, ki je shranjena digitalno, pri tem pa se nobena ne razlikuje od originala.

Z informacijo na papirju je to precej težje.

Družba, v kateri so informacije spravljene in prenašane v digitalni obliki, mora poskrbeti za to, da ne bo varnost informacij odvisna od fizičnega medija, ki jih je zapisal ali prenesel.

Varnost informacij mora temeljiti izključno na digitalni informaciji.

Eno izmed osrednjih orodij pri zaščiti informacij je **podpis**. Le-ta preprečuje poneverjanje, je dokaz o izvoru, identifikaciji, pričanju.

Podpis naj bi bil unikat vsakega posameznika, z njim se predstavimo, potrdimo, pooblastimo.

Z razvojem digitalne informacije moramo ponovno obdelati tudi koncept podpisa.

Ni več unikat, ki enolično določa podpisnika, kajti elektronsko kopiranje podpisa je tako lahko, da je skoraj trivialno na nepodpisan dokument pripeti poljuben podpis.

Potrebujemo protokole, ki imajo podobne lastnosti kot trenutni "papirni protokoli".

Družba ima enkratno priložnost, da vpelje nove in učinkovitejše načine, ki nam bodo zagotovili varnost informacij.

Veliko se lahko naučimo iz dosedanjih sistemov, obenem pa moramo odpraviti tudi številne pomanjkljivosti.

Primerjava digitalnega in navadnega (lastnoročnega) podpisa:

- navadni podpis je fizično del podpisanega dokumenta;
- navadni podpis preverjamo s primerjanjem, digitalnega z algoritmom, katerega rezultat je odvisen od ključa in dokumenta;
- kopija digitalnega podpisa je identična originalu;
- digitalni podpis je odvisen od dokumenta, ki ga podpisujemo.

Sistem za digitalno podpisovanje je peterka $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$, za katero velja

1. $\mathcal{P}$ je končna množica sporočil,
2. $\mathcal{A}$ je končna množica podpisov,
3. $\mathcal{K}$ je končna množica ključev,
4. $\forall$ ključ $K \in \mathcal{K}$ obstaja algoritem za podpisovanje

$$\mathrm{sig}_K \in \mathcal{S}, \quad \mathrm{sig}_K : \mathcal{P} \rightarrow \mathcal{A}$$

in algoritem za preverjanje podpisa

$$\mathrm{ver}_K \in \mathcal{V}, \quad \mathrm{ver}_K : \mathcal{P} \times \mathcal{A} \rightarrow \{\mathrm{true}, \mathrm{false}\}.$$

Funkciji sig_K in ver_K imata to lastnost, da za vsako sporočilo $x \in \mathcal{P}$ in vsak podpis $y \in \mathcal{A}$ velja

$$\mathrm{ver}_K(x, y) = \begin{cases} \mathrm{true}, & \text{če } y = \mathrm{sig}_K(x) \\ \mathrm{false}, & \text{če } y \neq \mathrm{sig}_K(x) \end{cases}$$

Zahteve:

- algoritma sig_K in ver_K imata polinomsko časovno zahtevnost
- sig_K je znan le podpisniku
- ver_K je splošno znan
- računsko mora biti nemogoče ponarediti podpis

Primer: Algoritem RSA lahko uporabimo tudi za podpisovanje. Naj bo $n = pq$, kjer sta p in q praštevili.

Če je (n, d) skriti ključ, (n, e) pa javni, pri čemer je $de \equiv 1 \pmod{\varphi(n)}$, potem definiramo:

$$\mathrm{sig}_K(x) = d_K(x) = x^d \bmod n$$

$$\mathrm{ver}_K(x, y) = \mathrm{true} \iff x = e_K(y) = y^e \bmod n$$

za $x, y \in \mathbb{Z}_n$.

Z zgornjim algoritmom je mogoče ponarediti podpis naključnih sporočil.

Ponarejevalec najprej izbere podpis y in nato izračuna

$$x \equiv y^e \pmod{n}.$$

Možnosti takega ponarejanja se izognemo z

- enosmernimi zgoščevalnimi funkcijami ali
- zahtevo, da ima sporočilo x določen pomen.

Pošiljanje podpisanih tajnih sporočil

Vrstni red šifriranja in digitalnega podpisovanja je pomemben.

1. Najprej podpisovanje:

$$x, \mathrm{sig}_{\mathrm{Anita}}(x) \to e_{\mathrm{Bojan}}((x, \mathrm{sig}_{\mathrm{Anita}}(x))).$$

2. Najprej šifriranje $z = e_{\mathrm{Bojan}}(x)$,
   potem podpis $y = \mathrm{sig}_{\mathrm{Anita}}(z)$:

   Bojan prejme (z, y), odšifrira tajnopis
   $x = d_{\mathrm{Bojan}}(z)$ ter preveri podpis $\mathrm{ver}_{\mathrm{Anita}}(z, y)$.

V drugem primeru lahko napadalec Cene zamenja Anitin podpis s svojim:

$$y' = \mathrm{sig}_{\mathrm{Cene}}(z) \to (z, y') \to x = d_{\mathrm{Bojan}}(z),$$
$$\mathrm{ver}_{\mathrm{Cene}}(z, y')$$

in Bojan bo mislil, da je sporočilo prišlo od Ceneta.

Zato se priporoča najprej podpisovanje in nato šifriranje.

V primeru algoritma RSA je potrebno pri zaporednem podpisovanju in šifriranju paziti na velikosti modulov (*reblocking problem*).

Če je $n_{\mathrm{Anita}} > n_{\mathrm{Bojan}}$, se lahko zgodi, da Bojan ne bo mogel razvozlati sporočila. Naj bo

$$(n_{\mathrm{Anita}}, e_{\mathrm{Anita}}, d_{\mathrm{Anita}}) = (62894113, 5, 37726937),$$
$$(n_{\mathrm{Bojan}}, e_{\mathrm{Bojan}}, d_{\mathrm{Bojan}}) = (55465219, 5, 44360237).$$

Anita podpiše sporočilo $x = 1368797$ in podpis zašifrira:

1. $s = x^{d_{\mathrm{Anita}}} \bmod n_{\mathrm{Anita}} = 59847900$,
2. $y = s^{e_{\mathrm{Bojan}}} \bmod n_{\mathrm{Bojan}} = 38842235$.

Bojan izračuna

1. $\widehat{s} = y^{d_{\mathrm{Bojan}}} \bmod n_{\mathrm{Bojan}} = 4382681$,
2. $\widehat{x} = \widehat{s}^{e_{\mathrm{Anita}}} \bmod n_{\mathrm{Anita}} = 54383568$.

Ker je $s > n_{\mathrm{Bojan}}$, je $\widehat{x} \neq x = 1368797$.

Verjetnost tega dogodka je

$$\frac{n_{\mathrm{Anita}} - n_{\mathrm{Bojan}}}{n_{\mathrm{Anita}}}.$$

Delitev shem za digitalno podpisovanje

1. Podpis je dodatek (ElGamal, DSA) sporočilu - sporočilo je možno rekonstruirati iz podpisa (RSA),
2. deterministični - nedeterministični,
3. enkratni - večkratni.

Različni sistemi za digitalno podpisovanje

- RSA
- ElGamal, DSS (*Digital Signature Standard*)
- Enkratni podpisi (*one-time signatures*)
- Slepi podpisi (*blind signatures*)
- Podpisi brez možnosti zanikanja (*undeniable signatures*)
- Skupinski podpisi (*group signatures*)
- Fail-Stop podpisi

ElGamalov sistem za digitalno podpisovanje

Za razliko od algoritma RSA je ElGamalov sistem namenjen predvsem digitalnemu podpisovanju, čeprav se ga da v posebnih primerih uporabiti tudi za šifriranje.

Podpis je nedeterminističen (odvisen od naključnega števila), torej sploh ni natanko določen.

Algoritem

Naj bo p takšno praštevilo, da je v $\mathbb{Z}_p$ težko izračunati diskretni algoritem in $\alpha \in \mathbb{Z}_p^*$ primitivni element.

Naj bo še $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{A} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$ in

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

Število a je skrito (zasebno),

števila p, α in β pa so javno znana.

Podpisovanje: podpisnik s ključem $K = (p, \alpha, a, \beta)$ izbere naključno skrito število $k \in \mathbb{Z}_{p-1}^*$ in določi

$$\mathrm{sig}_K(x, k) = (\gamma, \delta),$$

kjer je

$$\gamma \equiv \alpha^k \bmod p$$

in

$$\delta \equiv (x - a\gamma)k^{-1} \bmod (p-1).$$

Preverjanje podpisa: (samo z javnimi p, α in β)

$$\mathrm{ver}_K(x, \gamma, \delta) = \text{true} \iff \beta^\gamma \gamma^\delta \equiv \alpha^x \pmod{p}.$$

Primer: Naj bo $p = 467, \alpha = 2$ in $a = 127$.
Potem je $\beta \equiv \alpha^a \bmod p = 132$. Recimo, da želimo podpisati $x = 100$, izbrali pa smo si tudi $k = 213$.
Podpis je enak (γ, δ), kjer je

$$\gamma \equiv 2^{213} \bmod 467 = 29$$

in

$$\delta \equiv (100 - 127 \cdot 29)\ 431 \bmod 466 = 51.$$

Pri preverjanju izračunamo

$$132^{29}\, 29^{51} \equiv 189 \bmod 467 \quad \text{in}$$

$$2^{100} \equiv 189 \bmod 467.$$

Zadnji vrednosti se ujemata, zato je podpis pravi.

Varnost ElGamalovega sistema za podpisovanje

Kako bi lahko ponaredili podpis, ne da bi vedeli za vrednost skritega števila a?

1. Za dano sporočilo x je potrebno najti tak par (γ, δ), da bo veljalo $\beta^\gamma \gamma^\delta \equiv \alpha^x \pmod{p}$, torej
   - če izberemo γ: rabimo $\delta = \log_\gamma \alpha^x \beta^{-\gamma} \pmod{p}$,
   - če izberemo δ: glede na γ je potrebno rešiti enačbo $\beta^\gamma \gamma^\delta \equiv \alpha^x \pmod{p}$,
   - hkrati računamo γ in δ (zaenkrat ni še nihče odkril hitrega postopka za reševanje zgornje enačbe).

2. Za podpis (γ, δ) je potrebno najti ustrezno sporočilo x:

$$x = \log_\alpha \beta^\gamma \gamma^\delta \pmod{p}.$$

3. Hkratno računanje x, γ in δ: naj bosta i in j takšni števili, da velja $0 \le i, j \le p-2$ in $D(j, p-1) = 1$. Potem števila

$$\begin{aligned} \gamma &\equiv \alpha^i \beta^j \pmod{p}, \\ \delta &\equiv -\gamma j^{-1} \pmod{(p-1)}, \\ x &\equiv -\gamma i j^{-1} \pmod{(p-1)} \end{aligned}$$

zadoščajo enačbi $\beta^\gamma \gamma^\delta \equiv \alpha^x \pmod{p}$.

Primer: Če je $p = 467, \alpha = 2$ in $\beta = 132$, lahko z izbiro $i = 99$ in $j = 179$, dobimo veljaven podpis $(117, 41)$ za sporočilo 331.

4. Ali lahko pri veljavnem podpisu (γ, δ) za x najdemo še kakšen podpis za neko drugo sporočilo x'?
   Odgovor je "DA".

   Naj bodo h, i in j takšna števila, da zanje velja $0 \le h, i, j \le p-2$ in $D(h\gamma - j\delta, p-1) = 1$.
   Potem je par (λ, μ) veljaven podpis za x', kjer je

$$\begin{aligned} \lambda &= \gamma^h \alpha^i \beta^j \bmod p, \\ \mu &= \delta\lambda(h\gamma - j\delta)^{-1} \bmod (p-1), \\ x' &= \lambda(hx + i\delta)(h\gamma - j\delta)^{-1} \bmod (p-1). \end{aligned}$$

Nevarnosti pri napačni uporabi ElGamalovega sistema

1. Če naključno število k ne ostane skrito, lahko izračunamo

$$a = (x - k\delta)\gamma^{-1} \bmod (p-1).$$

2. Število k lahko uporabimo le enkrat, sicer ga je mogoče zlahka izračunati.

Digital Signature Standard

DSS je modifikacija ElGamalovega sistema za podpisovanje. Kot ameriški standard je bil predlagan leta 1991, sprejet pa leta 1994.

Algoritem: Naj bo p praštevilo velikosti L bitov, kjer je $512 \leq L \leq 1024$ in $64 \,|\, L$, q 160−bitno praštevilo, da $q \,|\, p-1$, ter $\alpha \in \mathbb{Z}_p^*$ q−ti koren enote po modulu p. Definirajmo $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{A} = \mathbb{Z}_q \times \mathbb{Z}_q$ in

$$\mathcal{K} = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

Vrednosti p, q, α in β so javne, število a pa skrito.

Podpisovanje: podpisnik izbere naključno skrito število k, $1 \leq k \leq q-1$ in določi

$$\mathrm{sig}_K(x, k) = (\gamma, \delta),$$

kjer je

$$\gamma \equiv (\alpha^k \bmod p) \bmod q$$

in

$$\delta \equiv (x + a\gamma)\, k^{-1} \pmod{q}.$$

Za število δ mora veljati $\delta \not\equiv 0 \pmod{q}$.

Preverjanje podpisa: najprej izračunamo

$$e_1 \equiv x\delta^{-1} \quad \text{in} \quad e_2 \equiv \gamma\delta^{-1}.$$

Potem je

$$ver_K(x, \gamma, \delta) = \text{true}$$
$$\Updownarrow$$
$$(\alpha^{e_1}\, \beta^{e_2} \bmod p) \bmod q = \gamma.$$

Podobno kot pri ElGamalovi shemi je podpisovanje hitrejše od preverjanja (za razliko od RSA).

Prikrit kanal v algoritmu DSA

V algoritmu DSA obstaja prikrit kanal, ki omogoča:

(a) vključitev šifriranega sporočila v podpis, ki ga lahko prebere le tisti, ki pozna dodaten ključ;

(b) razkritje skritega ključa, brez vednosti njegovega lastnika.

Eno možnost za (a) si oglejmo na naslednji foliji, točko (b) pa prihranimo za domačo nalogo.

Primer: Izberimo n tajnih praštevil $p_1, \dots, p_n$ in poskusimo v podpis skriti binarno zaporedje $b_1, \dots, b_n$. Naključno število k izbiramo toliko časa, da za vsak $1 \leq i \leq n$ velja

$$b_i = 1 \Longrightarrow \gamma \text{ je kvadratni ostanek po modulu } p_i,$$
$$b_i = 0 \Longrightarrow \gamma \text{ ni kvadratni ostanek po modulu } p_i,$$

kjer je $\mathrm{sig}_K(x, k) = (\gamma, \delta)$.

Napadi

Uganjevanje fraz, ki jih uporabljamo za gesla

primer	število znakov	zahtevnost	dolžina gesla	čas za razbijanje
mucka	5	25 (majhne črke)	12 bitov	40 minut
br1a9Az	7	62 (črke in številke)	24 bitov	22 let
TH,X1lb<V+	10	95 (znaki na tipkov.)	40 bitov	nedosegljivo

Če uporabimo angleško ali slovensko besedo, dobimo zaporedje s približno 1.3 biti entropije na en znak (t.j. prostor za besedo proti popolnoma naključnim znakom).

Napadi z grobo silo (angl. Brute Force Attack)

posameznik ima 1 PC in programsko opremo ($2^{17} - 2^{24}$ ključev/sek.),

majhna skupina, 16 PC ($2^{21} - 2^{28}$ ključev/sek.),

akademska omrežja, 256 PC ($2^{25} - 2^{32}$ ključev/sek.),

veliko podjetje z \$1.000.000 za strojno opremo (2^{43} ključev/sek.),

vojaška obveščevalna organizacija z \$1.000.000.000 za strojno opremo in napredno tehnologijo (2^{55} ključev/sek.).

Napadi z grobo silo

dolžina ključa (v bitih)	posamični napadalec	majhne skupine	razisko-valna omrežja	velika podjetja	vojaške obveščevalne službe
40	tedni	dnevi	ure	milisekunde	mikrosekunde
56	stoletja	desetletja	leta	ure	sekunde
64	tisočletja	stoletja	destletja	dnevi	minute
80	∞	∞	∞	stoletja	stoletja
128	∞	∞	∞	∞	tisočletja

Povprečen čas za napad z grobo silo

dolžina ključev (v bitih)	**število možnih ključev**	**potreben čas za eno šifriranje/μsek.**	**potreben čas za 10^6 šifriranj/μsek.**
32	$2^{32} = 4.3 \times 10^9$	$2^{31}\mu\text{sec} \approx 36$ min	≈ 2 milisek.
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55}\mu\text{sec} \approx 1142$ let	≈ 10 ur
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\mu\text{sec} \approx 5 \times 10^{24}$	$\approx 5 \times 10^{18}$ let

Napadi na PKS

Napadi na DSA

- Metoda Index Calculus ($p \approx 2^{1024}$)

- Pollardova ρ-metoda ($\sqrt{\pi q/2}$, $q \approx 2^{160}$)

Napadi na ECDSA

- Pollardova ρ-metoda ($\sqrt{\pi n/2}$, $n \approx 2^{160}$)

Programski napadi

MIPS računalnik lahko opravi 4×10^4 seštevanj točk na eliptični krivulji na sekundo.

(Ta ocena je precej konzervativna. Posebaj prirejeno integrirano vezje s frekvenco ure 40 MHz, ki opravlja operacije na eliptični krivulji nad obsegom $GF(2^{155})$ in lahko izvede 40.000 seštevanj na sekundo.)

Na osnovi tega zaključimo, da je število seštevanj na eliptični krivulji na $GF(2^{155})$ izvedeno na MIPS računalniku v času enega leta naslednje

$$(4 \times 10^4) \cdot (60 \times 60 \times 24 \times 365) \approx 2^{40}.$$

Spodnja tabela nam kaže kolikšno računsko moč potrebujemo za računanje problema diskretnega logaritma z uporabo Pollard ρ-methodo za različne vrednosti števila n. MIPS leto je ekvivalentno računski moči 1 MIPS računalnika, ki je na voljo eno leto.

velikost obsega (v bitih)	velikost števila n	$\sqrt{\pi n/2}$	MIPS let
155	150	2^{75}	3.8×10^{10}
210	205	2^{103}	7.1×10^{18}
239	234	2^{117}	1.6×10^{23}

Npr. če imamo na voljo 10.000 računalnikov z močjo 1.000 MIPS in je $n \approx 2^{150}$, potem je lahko problem diskretnega logaritma na eliptični krivulji rešen v 3.800 letih.

Prejšnjo tabelo je zanimivo primerjati s Odlyzko-vo tabelo, ki kaže kolikšno računsko moč potrebujemo za faktorizacijo celih števil s sedanjo verzijo splošnega NFS algoritma.

velikost števila n (v bitih)	MIPS let
512	3×10^4
768	2×10^8
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

Hardwarski napadi

Za bolj perspektiven napad (s strani dobro financiranega napadalca) na ECC, bi bilo potrebno narediti specializirano programsko opremo za paralelno iskanje na osnovi Pollard ρ-metode.

Van Oorschot and Wiener ocenjujeta: za $n \approx 10^{36} \approx 2^{120}$ bi računalnik z $m = 325.000$ procesorji (cena okoli 10 milijonov USD) lahko izračunal diskretni logaritem v približno 35 dneh.

Poudariti moramo, da računanje diskretnega logaritma na $E(\mathbb{Z}_p)$ v zgoraj omenjenih napadih odkrije en sam *zasebni ključ.*

M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener, January 1996, (`http://theory.lcs.mit.edu/rivest/publications.html`) govorijo o minimalnih dolžinah ključev potrebnih za varen simetrični sistem (npr. DES ali IDEA):

> *Da bi zagotovili ustrezno zaščito proti najbolj resnim grožnjam (npr. velike komercialne ustanove in vladne agencije) mora ključ biti dolg vsaj 75 bitov. Za zaščito za naslednjih 20-let morajo ključi biti dolgi vsaj 90 bitov (pri tem upoštevamo pričakovano rast računske moči).*

Če posplošimo te zaključke na eliptične kripto-sisteme, mora biti praštevilo n, ki zagotovlja kratkoročno varnost, dolgo vsaj 150 bitov, za srednjeročno varnost pa vsaj 180 bitov.

Dolžina ključev

simetrične šifre (AES)	**asimetrične** (RSA, DSA, DH)	**eliptične krivulje**
40 bitov	274 bitov	80 bitov
56 bitov	384 bitov	106 bitov
64 bitov	512 bitov	132 bitov
80 bitov	1024 bitov	160 bitov
96 bitov	1536 bitov	185 bitov
112 bitov	2048 bitov	237 bitov
120 bitov	2560 bitov	256 bitov
128 bitov	3072 bitov	270 bitov

Digitalni podpisi v $\mathbb{Z}_p$ in na EC

grupa	$\mathbb{Z}_p^*$	$E(\mathbb{Z}_p)$
elementi	množica celih števil $\{1, 2, \dots, p-1\}$	točke (x, y), ki zadoščajo enačbi eliptične krivulje E in še točka v neskončnosti
operacija	množenje po modulu p	seštevanje točk na eliptični krivulji
oznake	elementi: g, h množenje: $g \times h$ multiplikativni inverz: h^{-1} deljenje: g/h potenciranje: g^a	elementi: P, Q seštevanje: $P + Q$ nasprotna točka: $-Q$ odštevanje: $P - Q$ skalarno množenje točke: aP
problem diskretnega logaritma	Za dana $g, h \in \mathbb{Z}_p^*$ poišči tako celo število a da je $h = g^a \bmod p$.	Za dani točki $P, Q \in E(\mathbb{Z}_p)$ poišči tako celo število a da je $Q = aP$.

Grupe

Digital Signature Algorithm (DSA) eliptični analog ECDSA

DSA	ECDSA
1. Izberi praštevili p in q velikosti $2^{1023} < p < 2^{1024}$, $2^{159} < q < 2^{160}$, tako da $q \mid p-1$.	1. Izberi tako eliptično krivuljo E: $y^2 = x^3 + ax + b$ nad $\mathbb{Z}_q$, da je število $\lvert E(\mathbb{Z}_p) \rvert$ deljivo s praštevilom $n \approx 160$-bitov.
2. $t \in \mathbb{Z}_p^*$, izračunaj $g = t^{(p-1)/q} \bmod p$, potem je $g \neq 1$ in ima red q v $\mathbb{Z}_p^*$.	2. Izberi točko P na $E(\mathbb{Z}_q)$ katere red je praštevilo n.
3. Uporabi multiplikativno grupo $\{g^0, g^1, \dots, g^{q-1}\}$	3. Uporabi aditivno grupo $\{\mathcal{O}, P, 2P, \dots, (n-1)P\}$

Generiranje ključa pri DSA in ECDSA

DSA	ECDSA
1. Izberi naključno celo število $x \in [2, q-2]$, tj. **zasebni ključ**	1. Izberi naključno celo število $d \in [2, n-2]$, tj. **zasebni ključ**
2. Izračunaj $y = g^x \bmod p$, **javni ključ** je (p, q, g, y).	2. Izračunaj $Q = dP$, **javni ključ** je (E, n, q, Q).

DSA	ECDSA
q	n
g	P
x	d
y	Q

Podpisovanje sporočila m

DSA	ECDSA
1. Izberi naključno celo število $k \in [2, q-2]$.	1. Izberi naključno celo število $k \in [2, n-2]$.
2. Izračunaj $g^k \bmod p$, $r = (g^k \bmod p) \bmod q$, $0 \neq s = k^{-1}(h(m) + xr) \bmod q$.	2. Izračunaj $kP = (x_1, y_1)$, $r = x_1 \bmod n$, $0 \neq s = k^{-1}(h(m) + dr) \bmod n$.
Podpis je par (r, s).	

Preverjanje podpisa (r, s) sporočila m osebe A

DSA	ECDSA
1. Preskrbi si avtentično kopijo javnega ključa osebe A: (p, q, g, y)	(E, n, q, Q)
2. Izračunaj $s^{-1} \bmod p$ in $h(m)$, $u_1 = h(m)s^{-1} \bmod q$, $u_2 = rs^{-1} \bmod q$, $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$.	2. Izračunaj $s^{-1} \bmod n$ in $h(m)$ $u_1 = h(m)s^{-1} \bmod n$, $u_2 = rs^{-1} \bmod n$, $u_1 P + u_2 Q = (x_0, y_0)$ in $v = x_0 \bmod n$.
Sprejmi podpis samo in samo če je $v = r$.	

SigGen z EC

Razvita je bila v **Certicom Corp., Kanada**, v sodelovanju s Schlumberger Smart Cards and Systems.

Uporablja Motorolin čip 68SC28:

- ROM 12.790 zlogov,
- EEPROM 8.112 zlogov,
- RAM 240 zlogov.

Vsebuje tehnologijo MULTIFLEX^{TM} ter tehnologijo eliptičnih krivulj $(CE)^2$, ki jo razvija podjetje Certicom Corp.

SigGen kartica je zelo prikladna za končnega uporabnika ter za proces prepoznavanja:

- je poceni,
- podpis je opravljen v pol sekunde,
- rabi samo 90 zlogov RAM-a,
- program ne zasede niti 4 KB.

Je edina pametna kartica, ki opravi digitalni podpis kar z obstoječim procesorjem.

Eliptični kripto-sistemi nudijo največjo moč glede na število bitov ključa med današnjimi javnimi kripto-sistemi.

Manjši ključi omogočajo

- manjše sistemske parametre,
- manjša potrdila z javnimi ključi,
- hitrejšo implementacijo,
- manjše zahteve po energiji,
- manjše procesorje,

itd.

Enkratni podpis

Z istim ključem lahko podpišemo le en dokument. Ponavadi algoritem temelji na enosmernih funkcijah.

Lamportova shema: $\mathcal{P} = \{0,1\}^{k \in \mathbb{N}}$, $|Y| < \infty$, enosmerna funkcija $f : Y \to Z$.

Naključno izberemo matriko $(y_{ij}) \in Y^{k \times 2}$ in določimo matriko enake velikosti z elementi $z_{ij} = f(y_{ij})$.

Ključ K sestavljata obe matriki, prva je skrita, druga pa javna.

Podpisovanje:

$$\mathrm{sig}_K(x_1, \dots, x_k) = (y_{1,x_1}, \dots, y_{k,x_k}).$$

Preverjanje podpisa:

$$\mathrm{ver}_K(x_1, \dots, x_k, a_1, \dots, a_k) = \text{true}$$
$$\Updownarrow$$
$$f(a_i) = z_{i,x_i}, \quad 1 \le i \le k.$$

Napadalec ne more ponarediti podpisa, saj ne more obrniti enosmerne funkcije f, da bi izračunal y-e.

Če pa bi podpisali dve različni sporočili z isto shemo, potem bi napadalec lahko poneveril podpis novih sporočil.

Primer: Naj bo $f(x) = 3^x \pmod{7879}$, ključ pa sestavljen iz matrik

$$\begin{pmatrix} 5831 & 735 \\ 803 & 2467 \\ 4285 & 6449 \end{pmatrix} \text{ in } \begin{pmatrix} 2009 & 3810 \\ 4672 & 4721 \\ 268 & 5731 \end{pmatrix}.$$

Potem je podpis za $x = (1, 1, 0)$ enak $(y_{1,1}, y_{2,1}, y_{3,0}) = (735, 2467, 4285)$.

Pomanjkljivost te sheme je velikost podpisa (za vsak bit čistopisa število med 1 in Y).

Spernerjeva lema

Naj bo $\mathcal{F}$ taka družina podmnožic n-elementne množice, da noben njen element ni vsebovan v kakem drugem elementu iz $\mathcal{F}$. Potem ima družina $\mathcal{F}$ največ

$$\binom{n}{\lfloor n/2 \rfloor} \quad \text{elementov.}$$

Bos-Chaumova shema za enkratni podpis

$\mathcal{P} = \{0,1\}^{k \in \mathbb{N}}$, $n \in \mathbb{N}$ tak, da je $2^k \le \binom{2n}{n}$.
B je množica z $2n$ elementi in

$$\phi : \{0,1\}^k \to \mathcal{B}$$

injekcija, kjer je $\mathcal{B}$ množica n-teric iz B.

Naj bo $f : Y \to Z$ enosmerna funkcija.

Naključno izberemo vektor $\mathbf{y} = (y_i) \in Y^{2n}$.

Naj bo ključ K tajni vektor $\mathbf{y}$ in javni vektor $(f(y_i))$.

$$\mathrm{sig}_K(x_1, \dots, x_k) = \{y_j \mid j \in \phi(x_1, \dots, x_k)\}.$$

in

$$\mathrm{ver}_K(x_1, \dots, x_k, a_1, \dots, a_n) = \text{true}$$
$$\Updownarrow$$
$$\{f(a_i) \mid 1 \le i \le n\} = \{z_j \mid j \in \phi(x_1, \dots, x_k)\}.$$

Uporabili smo $2^k \le \binom{2n}{n}$. Ocenimo binomski koeficient in dobimo

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2}$$

oziroma z uporabo Stiringove formule $2^{2n}/\sqrt{(\pi n)}$.
Od tod dobimo

$$k \le 2n - \frac{\log_2(n\pi)}{2}.$$

Asimptotično je torej n blizu $k/2$, zato smo dobili 50% redukcijo dolžine podpisa.

Slepi podpis

Želimo, da nam kdo podpiše dokument, hkrati pa nočemo, da bi podpisnik videl njegovo vsebino (npr. notarji, banke pri elektronskem denarju).

Algoritem (Chaum): Anita želi od Bojana podpis dokumenta x, $1 \le x \le n-1$, pri čemer je (n, e) Bojanov javni ključ za algoritem RSA, d pa zasebni ključ.

1. *Anita izbere takšno skrito naključno število k, da velja $0 \leq k \leq n-1$ in $D(n,k)=1$.*
   Nato zastre dokument, tj. izračuna
   $$m = xk^e \bmod n,$$
   in ga pošlje Bojanu.
2. *Bojan podpiše zastrti dokument*
   $$s = m^d \bmod n.$$
3. *Anita odstre podpisani dokument*
   $$y = k^{-1}s \bmod n.$$

Podpisi brez možnosti zanikanja

Podpisa ni mogoče preveriti brez sodelovanja podpisnika, podpisnik pa tudi ne more zanikati, da bi že podpisani dokument res podpisal

(razen če odkloni sodelovanje pri podpisu, kar pa lahko pojmujemo kot priznanje, da je podpis v resnici ponarejen).

Primer algoritma (Chaum-van Antwerpen):

Naj bosta q in $p = 2q+1$ praštevili, $\alpha \in \mathbb{Z}_p^*$ element reda q, $1 \leq a \leq q-1$ in $\beta = \alpha^a \bmod p$.

Grupa G je multiplikativna podgrupa reda q grupe $\mathbb{Z}_p^*$ (G sestavljajo kvadratični ostanki po modulo p).

Naj bo $\mathcal{P} = \mathcal{A} = G$ in
$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta = \alpha^a \bmod p\}.$$

Števila p, α in β so javna, vrednost a pa je skrita.

Podpisovanje (Bojan podpiše dokument $x \in G$):
$$y = \mathrm{sig}_K(x) = x^a \bmod p.$$

Preverjanje podpisa:

1. Anita izbere naključni števili $e_1, e_2 \in \mathbb{Z}_q^*$. Nato izračuna $c = y^{e_1}\beta^{e_2} \bmod p$ in ga pošlje Bojanu.
2. Bojan izračuna $d = c^{a^{-1} \bmod q} \bmod p$ in ga vrne Aniti.
3. Anita sprejme podpis kot veljaven, če je
   $$d = x^{e_1}\alpha^{e_2} \bmod p.$$

Izrek. *Če je $y \not\equiv x^a (\bmod p)$, potem bo Anita sprejela y za veljaven podpis čistopisa x z verjetnostjo $1/q$.*

Poleg algoritmov za podpisovanje in preverjanje obstaja še algoritem (*disavowal protocol*), s katerim lahko podpisnik dokaže, da je ponarejen podpis res ponarejeni, hkrati pa ne more zanikati, da pravega podpisa ni napravil sam.

Primeri podpisov brez možnosti zanikanja

- *Entrusted undeniable signature*: *disavowal* protokol lahko izvede le za to določena ustanova, npr. sodišče.
- *Designated confirmer signature*: ob podpisu sami določimo, kdo bo namesto nas sodeloval pri preverjanjih podpisov. Podpišemo lahko še vedno le mi.
- *Convertible undeniable signature*: shema vsebuje skrito število. Do razkritja tega števila mora pri preverjanju podpisa sodelovati podpisnik.

  Po razkritju lahko kdorkoli preveri podpis sam

  (kot pri običajnem digitalnem podpisu).

Skupinski podpisi

Lastnosti:

- Dokumente lahko podpisujejo le člani določene skupine.
- Kdorkoli lahko preveri, da je dokument podpisal nekdo iz omenjene skupine, vendar ne more ugotoviti, kdo je to bil.
- V primeru spora je možno podpis "odpreti" in identificirati podpisnika.

Fail-stop podpisi

Če bi ponarejevalec z metodo grobe sile našel skriti ključ, bi lahko v večini sistemov za digitalne podpise podpis ponaredil. Fail-stop sistemi takšno možnost onemogočijo tako, da vsakemu javnemu ključu priredijo več skritih ključev.

Algoritem (van Heyst - Pedersen)

Generiranje ključa se razdeli med Anito in TTP (*Trusted Third Party*).

TTP izbere praštevili q in $p = 2q + 1$ (diskretni algoritem je težko izračunljiv), element $\alpha \in \mathbb{Z}_p^*$ reda q ter skrito naključno število a_0, $1 \leq a_0 \leq q - 1$ in izračuna $\beta \equiv \alpha^{a_0} \pmod{p}$. Nato Anita pošlje četverko (p, q, α, β) in izbere skrita naključna števila $a_1, a_2, b_1, b_2 \in \mathbb{Z}_q$, ki predstavljajo njen skriti ključ, ter določi svoj javni ključ $(\gamma_1, \gamma_2, p, q, \alpha, \beta)$, kjer je

$$\gamma_1 = \alpha^{a_1} \beta^{a_2} \pmod{p} \text{ in } \gamma_2 = \alpha^{b_1} \beta^{b_2} \pmod{p}.$$

Podpisovanje: $y = \mathrm{sig}_K(x) = (y_1, y_2)$, kjer je

$$y_1 \equiv a_1 + x\, b_1 \pmod{q}$$

in

$$y_2 \equiv a_2 + x\, b_2 \pmod{q}.$$

Preverjanje podpisa:

$$\mathrm{ver}_K(x, y_1, y_2) = \text{true} \iff \gamma_1 \gamma_2^x \equiv \alpha^{y_1} \beta^{y_2} \pmod{p}.$$

Opombe:

1. Natanko q^2 četverk (a_1', a_2', b_1', b_2'), kjer so elementi iz $\mathbb{Z}_q$, da enaki vrednosti (γ_1, γ_2) v javnem ključu.
2. Teh q^2 četverk da pri istem dokumentu x q različnih podpisov.
3. Naj bo Q_1 množica q četverk, ki da pri x enak podpis. Potem da ta množica pri drugem dokumentu q različnih podpisov.

Varnost sistema

Recimo, da želi nekdo ponarediti Anitin podpis za sporočilo x'.

1. Če ponarejevalec pozna le skriti ključ, ki pripada javnemu, je verjenost $1/q$, da je njegov podpis enak Anitinemu.
2. Ponarejevalec ima dostop do drugega sporočila x in Anitinega podpisa (y_1, y_2). Po tretji opombi je verjetnost spet $1/q$.

7. poglavje

Zgoščevalne funkcije (Hash Functions)

- zgoščevalne funkcije brez trčenj
- verjetnost trčenja
- napad s pomočjo paradoksa rojstnih dnevov
- zgoščevalna funkcija z diskretnim logaritmom

Shema DSS (brez uporabe zgoščevalnih funkcij) podvoji dolžino podpisanega sporočila.

Resnejši problem nastane, ker je mogoče preurejati dele podpisanega sporočila ali pa nekatere celo izpustiti/dodati.

Celovitost podatkov ne more biti zagotovljena izključno s podpisovanjem majhnih delov dokumenta, zato vpeljemo **zgoščevalne funkcije** (angl. Hash Functions), ki poljubno dolgemu sporočilu priredijo kratko zaporedje bitov, ki jih potem podpišemo.

Zgoščevalne funkcije brez trčenj

(angl. Collision-free Hash Functions)

Naj bo (x, y) podpisano sporočilo, kjer je

$$y = \mathrm{sig}_K(h(x)).$$

Preprost napad: izračunamo $z = h(x)$ in nato poiščemo tak od x različen x', da je $h(x') = h(x)$.

Def: Naj bo x sporočilo. Za zgoščevalno funkcijo h pravimo, da je **šibko brez trčenj** (angl. weakly collision-free), če v doglednem času ni možno najti (izračunati) tak od x različen x', da je $h(x) = h(x')$.

Še en napad: poiščemo taka x in x', da je $x \neq x'$ in $h(x') = h(x)$ ter prisilimo Bojana, da podpiše x. Potem je (x', y) poneverjen podpis.

Def: Za zgoščevalno funkcijo h pravimo, da je **krepko brez trčenj** (angl. strongly collision-free), če v doglednem času ni možno najti (izračunati) taka x in x', da je $x \neq x'$ in $h(x) = h(x')$.

Pa še en napad: recimo, da nam je uspelo ponarediti podpis naključnega števila z, nato pa poiščemo tak x, da je $z = h(x)$.

Ta napad preprečimo z enosmernimi funkcijami.

Dokazali bomo, da so funkcije brez trčenj enosmerne. To sledi iz trditve, da je možno algoritem za računanje obrata zgoščevalne funkcije uporabiti kot podprogram Las Vegas probabilističnega algoritma, ki išče trčenja.

Izrek: *Naj bo $h : X \longrightarrow Z$ zgoščevalna funkcija, $|X| < \infty$ in $|X| \geq 2|Z|$ ter naj bo $\mathbf{A}$ algoritem za računanje obrata zgoščevalne funkcije. Potem obstaja Las Vegas probabilistični algoritem, ki najde trčenja z verjetnostjo vsaj $1/2$.*

Dokaz: Naj bo $\mathbf{B}$ naslednji algoritem.

1. Izberi naključen element $x \in X$,
2. izračunaj $z := h(x)$,
3. izračunaj $x_1 := \mathbf{A}(z)$,
4. **if** $x_1 \neq x$ **then** x_1 in x trčita glede na h (uspeh)
   else QUIT(neuspeh).

Izračunajmo verjetnost za uspeh. Najprej definiramo ekvivalenčno relacijo

$$x \sim x' \iff h(x) = h(x').$$

Naj bo C množica ekvivalenčnih razredov, potem je $|C| \leq |Z|$. Velja tudi: $|Z| \leq |X|/2$.

$$P(\text{uspeh}) = \frac{1}{|X|}\sum_{x\in X}\frac{|[x]|-1}{|[x]|} = \frac{1}{|X|}\sum_{c\in C}\sum_{x\in c}\frac{|c|-1}{|c|}$$

$$= \frac{1}{|X|}\sum_{c\in C}(|c|-1) \geq \frac{|X|-|Z|}{|X|} \geq \frac{1}{2}. \quad \blacksquare$$

Verjetnost trčenja

Naj bo $h : X \longrightarrow Z$ zgoščevalna funkcija,

$$s_z = |h^{-1}(z)|$$

in

$$N = |\{\{x_1, x_2\} \mid h(x_1) = h(x_2)\}|,$$

tj. N je število neurejenih parov, ki trčijo pri funkciji h.

Pokazali bomo, da obstaja od nič različna spodnja meja za verjetnost P, da je $h(x_1) = h(x_2)$, kjer sta x_1 in x_2 naključna (ne nujno različna) elementa iz X.

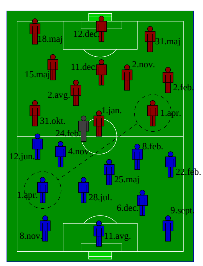
1. $\sum_{z\in Z} s_z = |X|$, tj. povprečje s_z-ov je $\overline{s} = |X|/|Z|$.
2. $N = \sum_{z\in Z}\binom{s_z}{2} = \frac{1}{2}\sum_{z\in Z} s_z^2 - \frac{|X|}{2}$.
3. $\sum_{z\in Z}(s_z - \overline{s})^2 = 2N + |X| - |X|^2/|Z|$.
4. $N \geq \frac{|X|}{2}\left(\frac{|X|}{|Z|} - 1\right)$, pri čemer velja enakost natanko tedaj, ko je $s_z = |X|/|Z|$ za vsak $z \in Z$.
5. $P \geq 1/|Z|$, pri čemer velja enakost natanko tedaj, ko je $s_z = |X|/|Z|$ za vsak $z \in Z$.

Kakšno naključje!!! Mar res?

Na nogometni tekmi sta na igrišču dve enajsterici in sodnik, skupaj **23 oseb**.

Kakšna je verjetnost, da imata **dve osebi** isti rojstni dan?

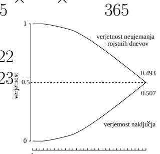
Ali je ta verjetnost lahko večja od 0.5?



Ko vstopi v sobo k-ta oseba, je verjetnost, da je vseh k rojstnih dnevov različnih enaka:

$$\frac{365}{365}\times\frac{364}{365}\times\frac{363}{365}\times\cdots\times\frac{365-k+1}{365}$$

$$= \begin{cases} 0.493, & \text{če je } k=22 \\ 0.507, & \text{če je } k=23 \end{cases}$$



V poljubni skupini 23-ih ljudi je verjetnost, da imata vsaj dva skupni rojstni dan $> 1/2$.

Čeprav je 23 majhno število, je med 23 osebami 253 različnih parov. To število je veliko bolj povezano z iskano verjetnostjo.

Testirajte to na zabavah z več kot 23 osebami.

Organizirajte stave in dolgoročno boste gotovo na boljšem, na velikih zabavah pa boste zlahka zmagovali.

Napad s pomočjo paradoksa rojstnih dnevov

(angl. Birthday Attack)

To seveda ni paradoks, a vseeno ponavadi zavede naš občutek.

Ocenimo še splošno verjetnost.

Mečemo k žogic v n posod in gledamo, ali sta v kakšni posodi vsaj dve žogici.

Poiščimo spodnjo mejo za verjetnost zgoraj opisanega dogodka.

Privzeli bomo, da je $|h^{-1}(x)| \approx m/n$, kjer je $n = |Z|$ in $m = |X|$ (v primeru, da velikosti praslik niso enake se verjetnost le še poveča).

$$\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\cdots\left(1-\frac{k-1}{n}\right) = \prod_{i=1}^{k-1}\left(1-\frac{i}{n}\right)$$

Iz Taylorjeve vrste

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \cdots$$

ocenimo $1 - x \approx e^{-x}$ in dobimo

$$\prod_{i=1}^{k-1}\left(1-\frac{i}{n}\right) \approx \prod_{i=1}^{k-1} e^{\frac{-i}{n}} = e^{\frac{-k(k-1)}{2n}}.$$

Torej je verjetnost trčenja

$$1 - e^{\frac{-k(k-1)}{2n}}.$$

Potem velja

$$e^{\frac{-k(k-1)}{2n}} \approx 1 - \varepsilon$$

oziroma

$$\frac{-k(k-1)}{2n} \approx \log(1-\varepsilon)$$

oziroma

$$k^2 - k \approx 2n \log \frac{1}{1-\varepsilon}$$

in če ignoriramo $-k$, dobimo končno

$$\boxed{k \approx \sqrt{2n \log \frac{1}{1-\varepsilon}}.}$$

Za $\varepsilon = 0.5$ je

$$k \approx 1.17\sqrt{n},$$

kar pomeni, da, če zgostimo nekaj več kot $\sqrt{n}$ elementov, je bolj verjetno, da pride do trčenja kot da ne pride do trčenja.

V splošnem je k proporcionalen z $\sqrt{n}$.

Napad s pomočjo paradoksa rojstnih dnevov s tem določi spodnjo mejo za velikost zaloge vrednosti zgoščevalne funkcije.

40-bitna zgostitev ne bi bila varna, saj bi prišli do trčenja z nekaj več kot 2^{20} (se pravi milijon) naključnimi zgostitvami z verjetnostjo vsaj $1/2$.

V praksi je priporočena najmanj 128-bitna zgostitev in shema DSS z 160-imi biti to vsekakor upošteva.

Zgoščevalna funkcija z diskretnim logaritmom

Varnost Chaum, Van Heijst in Pfitzmannove zgoščevalne funkcije je zasnovana na varnosti diskretnega logaritma.

Ni dovolj hitra, da bi jo uporabljali v praksi, je pa zato vsaj primerna za študij varnosti.

Naj bosta p in $q = (p-1)/2$ veliki praštevili, α in β pa dva primitivna elementa v $\mathbb{Z}_p$, za katera je vrednost $\log_\alpha \beta$ zasebna.

Zgoščevalno funkcijo

$$h : \{0,\dots,q-1\} \times \{0,\dots,q-1\} \longrightarrow \mathbb{Z}_p \backslash \{0\}$$

definirajmo z

$$h(x_1, x_2) = \alpha^{x_1}\beta^{x_2} \bmod p.$$

Pokazali bomo, da je ta funkcija **krepko brez trčenj** (strongly collision-free).

Predpostavimo obratno: za $(x_1, x_2) \neq (x_3, x_4)$ velja

$$h(x_1, x_2) = h(x_3, x_4)$$

oziroma

$$\alpha^{x_1}\beta^{x_2} \equiv \alpha^{x_3}\beta^{x_4} \pmod p$$

ali

$$\alpha^{x_1 - x_3} \equiv \beta^{x_4 - x_2} \pmod p.$$

Če je $d = D(x_4 - x_2, p-1)$, potem imamo zaradi $p - 1 = 2q$ natanko štiri možnost za d:

$$\{1, 2, q, p-1\}.$$

Če je $d = 1$, definiramo

$$y = (x_4 - x_2)^{-1} \mod (p-1)$$

in dobimo

$$\beta \equiv \beta^{(x_4 - x_2)y} \equiv \alpha^{(x_1 - x_3)y} \pmod p,$$

iz česar znamo izračunati diskretni logaritem

$$\log_\alpha \beta \equiv (x_1 - x_3)(x_4 - x_2)^{-1} \mod (p-1).$$

Če je $d = 2$, je $d = D(x_4 - x_2, q) = 1$, tako da lahko definiramo

$$y = (x_4 - x_2)^{-1} \mod q$$

in dobimo za $(x_4 - x_2)y = kq + 1$, kjer je $k \in \mathbb{Z}$,

$$\beta^{(x_4 - x_2)y} \equiv \beta^{kq+1} \equiv (-1)^k \beta \equiv \pm\beta \pmod p$$

zaradi $\beta^q \equiv -1 \pmod p$.

Torej imamo

$$\pm\beta \equiv \beta^{(x_4 - x_2)y} \equiv \alpha^{(x_1 - x_3)y} \pmod p,$$

od koder znamo izračunati diskretni logaritem

$$\log_\alpha \beta \equiv (x_1 - x_3)y \mod (p-1)$$

ali pa

$$\log_\alpha \beta \equiv (x_1 - x_3)y + q \mod (p-1).$$

Primer $d = q$ ni možen, saj iz

$$0 \leq x_2 \leq q-1 \quad \text{in} \quad 0 \leq x_4 \leq q-1$$

sledi

$$-(q-1) \leq x_4 - x_2 \leq q - 1.$$

Končno si poglejmo še primer $d = p - 1$, kar se lahko zgodi le za $x_2 = x_4$. Potem velja

$$\alpha^{x_1} \equiv \alpha^{x_3} \pmod p$$

oziroma $x_1 = x_3$ in $(x_1, x_2) = (x_3, x_4)$. Protislovje! ■

Razširitev zgoščevalne funkcije

Doslej smo študirali zgoščevalne funkcije s končno domeno.

Sedaj pa pokažimo, kako lahko razširimo zgoščevalne funkcije, ki so krepko brez trčenj in imajo končno domeno, do zgoščevalnih funkcij, ki so krepko brez trčenj in imajo neskončno domeno.

Tako bomo lahko podpisovali sporočila poljubne dolžine.

Naj bo h^* zgoščevalna funkcija za katero je $|X| = \infty$.

Naj bo zgoščevalna funkcija $h : (\mathbb{Z}_2)^m \longrightarrow (\mathbb{Z}_2)^t$, $m \geq t + 1$ krepko brez trčenj.

Potem bomo za $X = \cup_{i=m}^{\infty} (\mathbb{Z}_2)^i$ definirali zgoščevalno funkcijo

$$h^* : X \longrightarrow (\mathbb{Z}_2)^t,$$

ki bo tudi krepko brez trčenj.

Elementi množice X so zaporedja bitov, $|x|$, $x \in X$, pa naj predstavlja dolžino elementa x, tj. število bitov x-a. Z $x \,||\, y$ označimo spetje zaporedij x in y.

1.primer: $m \geq t + 2$. Naj bo $|x| = n > m$ in x spetje $x_1 \,||\, x_2 \,||\, \ldots \,||\, x_k$, kjer je

$$|x_1| = |x_2| = \cdots = |x_{k-1}| = m - t - 1$$

in $|x_k| = m - t - 1 - d$, pri čemer je $0 \leq d \leq m - t - 2$. Torej je

$$k = \left\lceil \frac{n}{m - t - 1} \right\rceil.$$

Funkcijo $h^*(x)$ definiramo z naslednjim algoritmom:

1. **for** $i = 1$ **to** $k-1$ **do** $y_i = x_i$
2. $y_k = x_k \,||\, 0^d$
3. naj bo y_{k+1} število d v dvojiškem sistemu
4. $g_1 = h(0^{t+1} \,||\, y_1)$
5. **for** $i = 1$ **to** k **do** $g_{i+1} = h(g_i \,||\, 1 \,||\, y_{i+1})$
6. $h^*(x) = g_{k+1}$

Spetje $y_1 \,||\, y_2 \,||\, \dots \,||\, y_{k+1}$ smo dobili tako, da smo x_k-ju na desni pripeli d ničel, zaporedju y_{k+1} pa smo pripeli ničle na levi, tako da je $|y_{k+1}| = m - t - 1$.

Izrek: *Naj bo $h : (\mathbb{Z}_2)^m \longrightarrow (\mathbb{Z}_2)^t$, $m \geq t+2$ zgoščevalna funkcija krepko brez trčenj. Potem je zgoraj def. zgoščevalna funkcija $h^* : X \longrightarrow (\mathbb{Z}_2)^t$ tudi krepko brez trčenj.*

Dokaz: Predpostavimo, da smo našli $x \neq x'$ tako, da je $h^*(x) = h^*(x')$ in pokažimo, da lahko poiščemo v polinomskem času trčenje za h. Vzamemo $|x| \geq |x'|$.

Naj bo $y(x) = y_1|| \dots ||y_{k+1}$ in $y(x') = y'_1|| \dots ||y'_{j+1}$, kjer sta x in x' dopolnjena z d ničlami po 2. koraku in so $g_1, \dots, g_{k+1}$ in $g'_1, \dots, g'_{j+1}$ zaporedoma vrednosti, ki jih izračunamo v korakih 4 in 5.

Če je $|x| \not\equiv |x'| \pmod{m-t-1}$, potem je $d \neq d'$ in od tod $y_{k+1} \neq y'_{j+1}$, torej dobimo trčenje iz

$$h(g_k||1||y_{k+1}) = g_{k+1} = h^*(x) = h^*(x') = g'_{j+1} = h(g'_j||1||y'_{j+1}).$$

Zato smemo sedaj privzeti, da $m-t-1$ deli $|x|-|x'|$. Iz $y_{k+1} = y'_{j+1}$, tako kot v prejšnjem primeru, sledi

$$h(g_k||1||y_{k+1}) = h(g'_j||1||y'_{j+1}).$$

Če je $g_k \neq g'_j$, smo našli trčenje, v nasprotnem primeru pa je

$$h(g_{k-1}||1||y_k) = g_k = g'_j = h(g'_{j-1}||1||y'_j).$$

Če na ta način s postopnim vračanjem ne pridemo do trčenja, dobimo na koncu

$$h(0^{t+1}||y_1) = g_1 = g'_{j-k} = \begin{cases} h(0^{t+1}||y'_1), & \text{če je } |x| = |x'| \\ h(g'_{j-k}||1||y'_1), & \text{sicer} \end{cases}$$

Za $|x| = |x'|$ oziroma $k = j$ nam da $y_1 \neq y'_1$ trčenje, sicer pa je $y_i = y'_i$ za $1 \leq i \leq k+1$. Od tod $y(x) = y(x')$, toda potem je $x = x'$, saj je preslikava $x \mapsto y(x)$ injekcija. Dobili smo protislovje s predpostavko, da je $x \neq x'$.

Končno v primeru, ko je $m-t-1$ deli $|x|-|x'| \neq 0$ dobimo trčenje, ker je $(t+1)$-vi bit v spetju $0^{t+1}||y_1$ enak 0, $(t+1)$-vi bit v spetju $g'_{j-k} \,||\, 1 \,||\, y'_1$ pa 1. ■

2.primer: $m = t+1$. Naj bo $|x| = n > m$ in definirajmo funkcijo f z $f(0) = 0$ in $f(1) = 01$.

Zgoščevalno funkcijo $h^*(x)$ definiramo z algoritmom:

1. $y = y_1 y_2 \dots y_k := 11||f(x_1)||f(x_2)|| \cdots ||f(x_n)$
2. $g_1 = h(0^t \,||\, y_1)$
3. **for** $i = 1$ **to** $k-1$ **do** $g_{i+1} = h(g_i \,||\, y_{i+1})$
4. $h^*(x) = g_k$

Funkcija $x \mapsto y = y(x)$ iz prvega koraka je injekcija.

Izrek: *Naj bo $h : (\mathbb{Z}_2)^{t+1} \longrightarrow (\mathbb{Z}_2)^t$ zgoščevalna funkcija krepko brez trčenj. Potem je zgoraj def. zgoščevalna funkcija $h^* : X \longrightarrow (\mathbb{Z}_2)^t$ tudi krepko brez trčenj.*

Dokaz: Predpostavimo, da smo našli $x \neq x'$ tako, da je $h^*(x) = h^*(x')$ in naj bo $y(x) = y_1|| \dots ||y_{k+1}$ in $y(x') = y'_1|| \dots ||y'_{j+1}$, kjer sta x in x' dopolnjena z d ničlami po 2. koraku.

Če je $k = j$, dobimo (kot pri prejšnjem dokazu) bodisi trčenje za zgoščevalno funkcijo h bodisi $y = y'$. Slednje nam da $x = x'$, kar pa je protislovje!

Sedaj pa privzemimo, da je $k \neq j$ oziroma kar $j > k$. Če ne pride do trčenja, dobimo naslednje zaporedje enakosti:

$$y_k = y'_j, \quad y_{k-1} = y'_{j-1}, \dots, y_1 = y'_{j-k+1},$$

kar pa ni možno, saj za $x \neq x'$ ter poljubno zaporedje z velja $y(x) \neq z||y(x')$, kajti zaporedni enici se pojavita izključno na začetku zaporedja $y(x)$.

Od tod zaključimo, da je h^* krepko brez trčenj. ■

Za računanje funkcije h^* smo uporabili funkcijo h kvečjemu

$$\left(1 + \left\lceil \frac{n}{m-t-1} \right\rceil\right) - \text{krat} \quad \text{za} \quad m \geq t+2$$

in

$$(2n+2) - \text{krat} \quad \text{za} \quad m = t+1.$$

Zgoščevalne funkcije iz kriptosistemov

Naj bo $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ računsko varen kriptosistem in naj bo

$$\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_n,$$

kjer je $n \geq 128$, da bi preprečili napad z rojstnim dnevom.

Ta pogoj izključi **DES** (pa tudi DES-ov čistopis ni tako dolg kot DES-ov ključ).

Naj bo dano zaporedje

$$x_1 \,||\, x_2 \,||\, \ldots \,||\, x_k, \quad \text{kjer je } x_i \in (\mathbb{Z}_2)^n, \;\; 1 \leq i \leq k.$$

Če število bitov v zaporedju ne bi bilo večkratnik števila n, bi lahko dodali nekaj ničel...

Začnemo z neko začetno vrednostjo $g_0 = \mathrm{IV}$ (initial value) in nato konstruiramo zaporedje

$$g_i = f(x_i, g_{i-1}),$$

kjer je f šifrirna funkcija izbranega kriptosistema. Potem je $h(x) = g_k$.

Definiranih je bilo veliko takih funkcij in mnoge med njimi so razbili (tj. dokazali, da niso varne), ne glede na to, ali je ustrezna šifra varna ali ne.

Naslednje štiri variacije pa zaenkrat izgledajo varne:

$$\begin{aligned}
g_i &= e_{g_{i-1}}(x_i) \oplus x_i,\\
g_i &= e_{g_{i-1}}(x_i) \oplus x_i \oplus g_{i-1},\\
g_i &= e_{g_{i-1}}(x_i \oplus g_{i-1}) \oplus x_i,\\
g_i &= e_{g_{i-1}}(x_i \oplus g_{i-1}) \oplus x_i \oplus g_{i-1}.
\end{aligned}$$

Zgoščevalna funkcija MD4

Preglejmo nekaj hitrih zgoščevalnih funkcij.

MD4 je predlagal Rivest leta 1990, njeno izboljšano verzijo **MD5** pa leta 1991.

Funkcija **Secure Hash Standard** (**SHS**) iz leta 1992/93 je bolj komplicirana, a je zasnovana na istih principih. Njeno "tehnično napako" pa so odpravili šele leta 1994 (**SHA-1**).

Iz danega zaporedja bitov x najprej sestavimo zaporedje

$$M = M[0]\, M[1] \ldots M[N-1],$$

kjer je $M[i]$ 32-bitna beseda in je $N \equiv 0 \bmod 16$.

1. $d = (447 - |x|) \bmod 512$,
2. naj bo j binarna reprezentacija števila $x \bmod 2^{64}$, pri čemer je $|j| = 64$,
3. $M = x \,||\, 1 \,||\, 0^d \,||\, j$.

1. $A = 67452301$ (hex), $B = efcdab89$ (hex), $C = 98badcfe$ (hex), $D = 10325476$ (hex)
2. **for** $i = 1$ **to** $N/16 - 1$ **do**
3.   **for** $j = 0$ **to** 15 **do**
4. $X[j] = M[16i + j]$.
5. $AA = A, \ldots, DD = D$.
6.   1. krog
7.   2. krog
8.   3. krog
9. $A = A + AA, \ldots, D = D + DD$.

Osnovne operacije:

$X \wedge Y$	po bitih
$X \vee Y$	po bitih
$X \oplus Y$	XOR po bitih
$\neg X$	negacija
$X + Y$	seštevanje po modulu 2^{32}
$X \lll s$	ciklični pomik v levo za s mest

V *big-endian* arhitekturi (kot npr. Sun SPARC postaja) predstavimo število na naslednji način

$$a_1 2^{24} + a_2 2^{16} + a_3 2^8 + a_4,$$

v *little-endian* arhitekturi (kot npr. Intel 80xxx), ki jo je privzela funkcija MD4 pa z

$$a_4 2^{24} + a_3 2^{16} + a_2 2^8 + a_1.$$

V 1., 2. in 3. krogu funkcije **MD4** uporabimo zaporedoma funkcije f, g, in h, definirane spodaj.

$$f(X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$$

$$g(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$$

$$h(X,Y,Z) = X \oplus Y \oplus Z$$

1. krog

1. $A = (A + f(B,C,D) + X[0]) \lll 3$
2. $D = (D + f(A,B,C) + X[1]) \lll 7$
3. $C = (C + f(D,A,B) + X[2]) \lll 11$
4. $B = (B + f(C,D,A) + X[3]) \lll 19$
5. $A = (A + f(B,C,D) + X[4]) \lll 3$
6. $D = (D + f(A,B,C) + X[5]) \lll 7$
7. $C = (C + f(D,A,B) + X[6]) \lll 11$
8. $B = (B + f(C,D,A) + X[7]) \lll 19$
9. $A = (A + f(B,C,D) + X[8]) \lll 3$
10. $D = (D + f(A,B,C) + X[9]) \lll 7$
11. $C = (C + f(D,A,B) + X[10]) \lll 11$
12. $B = (B + f(C,D,A) + X[11]) \lll 19$
13. $A = (A + f(B,C,D) + X[12]) \lll 3$
14. $D = (D + f(A,B,C) + X[13]) \lll 7$
15. $C = (C + f(D,A,B) + X[14]) \lll 11$
16. $B = (B + f(C,D,A) + X[15]) \lll 19$

2. krog

1. $A = (A + g(B,C,D) + X[0] + 5A827999) \lll 3$
2. $D = (D + g(A,B,C) + X[4] + 5A827999) \lll 5$
3. $C = (C + g(D,A,B) + X[8] + 5A827999) \lll 9$
4. $B = (B + g(C,D,A) + X[12] + 5A827999) \lll 13$
5. $A = (A + g(B,C,D) + X[1] + 5A827999) \lll 3$
6. $D = (D + g(A,B,C) + X[5] + 5A827999) \lll 5$
7. $C = (C + g(D,A,B) + X[9] + 5A827999) \lll 9$
8. $B = (B + g(C,D,A) + X[13] + 5A827999) \lll 13$
9. $A = (A + g(B,C,D) + X[2] + 5A827999) \lll 3$
10. $D = (D + g(A,B,C) + X[6] + 5A827999) \lll 5$
11. $C = (C + g(D,A,B) + X[10] + 5A827999) \lll 9$
12. $B = (B + g(C,D,A) + X[14] + 5A827999) \lll 13$
13. $A = (A + g(B,C,D) + X[3] + 5A827999) \lll 3$
14. $D = (D + g(A,B,C) + X[7] + 5A827999) \lll 5$
15. $C = (C + g(D,A,B) + X[11] + 5A827999) \lll 9$
16. $B = (B + g(C,D,A) + X[15] + 5A827999) \lll 13$

3. krog

1. $A = (A + h(B,C,D) + X[0] + 6ED9EBA1) \lll 3$
2. $D = (D + h(A,B,C) + X[8] + 6ED9EBA1) \lll 9$
3. $C = (C + h(D,A,B) + X[4] + 6ED9EBA1) \lll 11$
4. $B = (B + h(C,D,A) + X[12] + 6ED9EBA1) \lll 15$
5. $A = (A + h(B,C,D) + X[2] + 6ED9EBA1) \lll 3$
6. $D = (D + h(A,B,C) + X[10] + 6ED9EBA1) \lll 9$
7. $C = (C + h(D,A,B) + X[6] + 6ED9EBA1) \lll 11$
8. $B = (B + h(C,D,A) + X[14] + 6ED9EBA1) \lll 15$
9. $A = (A + h(B,C,D) + X[1] + 6ED9EBA1) \lll 3$
10. $D = (D + h(A,B,C) + X[9] + 6ED9EBA1) \lll 9$
11. $C = (C + h(D,A,B) + X[5] + 6ED9EBA1) \lll 11$
12. $B = (B + h(C,D,A) + X[13] + 6ED9EBA1) \lll 15$
13. $A = (A + h(B,C,D) + X[3] + 6ED9EBA1) \lll 3$
14. $D = (D + h(A,B,C) + X[11] + 6ED9EBA1) \lll 9$
15. $C = (C + h(D,A,B) + X[7] + 6ED9EBA1) \lll 11$
16. $B = (B + h(C,D,A) + X[15] + 6ED9EBA1) \lll 15$

Zgoščevalna funkcija **MD4** še ni bila razbita, vendar pa je ni težko razbiti, če bi opustili prvi ali pa zadnji krog.

Zato zgoščevalna funkcija **MD5** uporablja 5 krogov, a je 30% počasnejša (.9Mbytes/sec na SPARC-u).

Zgoščevalna funkcija **SHA** je še počasnejša (0.2Mbytes/sec na SPARC-u).
Opisali bomo le nekaj njenih modifikacij:

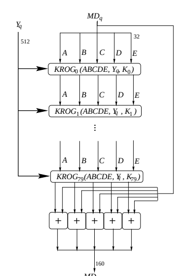
1. **SHS** privzame *big-endian* arhitekturo namesto *little-endian*.
2. **SHS** dobi 160-bitni rezultat (5 registrov).
3. **SHS** obdela 16 besed naenkrat, vendar jih najprej razširi v 80 besed, potem pa uporabi zaporedje 80-ih operacij na vsaki besedi.

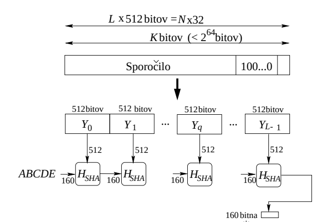
   $X[j] = X[j-3] \oplus X[j-8] \oplus X[j-14] \oplus X[j-16]$

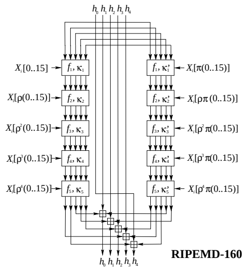
   za $16 \leq j \leq 79$.
4. **SHA-1** pa uporabi

   $X[j] = X[j-3] \oplus X[j-8] \oplus X[j-14] \oplus X[j-16] \lll 1$

   za $16 \leq j \leq 79$.







RIPEMD-160

HMAC

(Keyed-Hashing for Message Authentication)

Prednosti:

1. Kripto. zgoščevalne funkcije so v splošnem hitrejše v softwaru kot pa simetrične šifre (kot na primer DES).
2. Knjižnice zgoščevalnih funkcij so široko dostopne (medtem ko so bločne šifre, tudi kadar so uporabljene samo za MAC, omejene v smislu izvoznih dovoljenj).

Za design objectives v HMAC algoritmu in njegovo varnost glej:

M. Bellare, R. Canetti in H. Krawczyk, CRYPTO'96

(in `http://wwwcse.ucsd.edu/users/mihir`),

ki ga trenutno poskušajo vključiti v IETF
(Internet Engineering Task Force).

Časovne oznake/žigi (Timestamping)

Potrebujemo pričo o obstoju določenih podatkov ob določenem času, na primer na področju

- zaščite intelektualne lastnine
  (angl. intelectual property - IP), ali pa
- zanesljivega servisa za preprečevanje zanikanja
  (za dokaz, da je bil digitalni podpis generira
  v času veljavnosti ustreznega javnega ključa).

Če želi Bojan imeti dokaz o obstoju podatkov x ob nekem določenem času, potem naredi naslednje:

1. najprej izračuna zgostitev $z = h(x)$,
2. nato še zgostitev spoja
$$z' = h(z \,||\, \text{javna\_informacija}),$$
3. rezultat podpiše $y = \text{sig}_K(z')$, in
4. naslednji dan v časopisu objavi podatke
$$(z, \text{javna\_informacija}, y).$$

Časovne oznake s TS

Časovne žige omogoča pooblaščena organizacija za podpise, (angl. **Timestamper - TS**), ki je elektronski notar (angl. trusted timestamping service) oz. center zaupanja (TTP, angl. trusted third party).

Bojan najprej izračuna
$$z = h(x), \quad y = \text{sig}_K(x)$$
in pošlje par (z, y) notarju TS,

ki doda še datum D in podpiše trojico (z, y, D).

Zgornji algoritem je varen le pod pogojem, če je notar nepodkupljiv.

Potencialno se TS sooči z enormno odgovornostjo, če so časi kompromitirani.

Na primer, uporabnik lahko zanika vse podpise, ki jih je opravil kdaj koli.

Morda lahko nastane hujša škoda kot kompromitacija CA-jevega zasebnega ključa, o katerem bomo govorili v naslednjem poglavju.

Sicer pa si pomagamo z naslednjim algoritmom:

1. TS najprej izračuna
$$L_n = (t_{n-1}, \text{ID}_{n-1}, z_{n-1}, y_{n-1}, h(L_{n-1})),$$
2. nato še $C_n = (n, t_n, z_n, y_n, \text{ID}_n, L_n)$,
3. rezultat podpiše $s_n = \text{sig}_{\text{TS}}(h(C_n)))$, ter
4. pošlje $(C_n, s_n, \text{ID}_{n+1})$ osebi ID_n.

8. poglavje

Upravljanje ključev

- **Distribucija ključev**
  (Blomova shema, Diffie-Hellmanova shema)
- **Certifikati**
  (avtentikacijska drevesa, certifikatna agencija, infrastruktura javnih ključev, proces certifikacije, modeli zaupanja)
- **Uskladitev ključev**
  (Kerberos, Diffie-Hellmanova shema, MTI protokoli, Giraultova shema)
- **Internetne aplikacije**
  (Internet, IPsec: Virtual Private Networks, Secure Sockets Layer, varna e-pošta)

Vprašanja

- Od kje dobimo ključe?
- Zakaj zaupamo ključem?
- Kako vemo čigav ključ imamo?
- Kako omejiti uporabo ključev?
- Kaj se zgodi, če je kompromitiran (izgubljen) zasebni ali tajni ključ? Kdo je odgovoren?
- Kako preklicati ključ?
- Kako lahko obnovimo ključ?
- Kako omogočimo servis preprečitve zanikanja?

Ta vprašanja veljajo tako za simetrične (tajne) ključe kakor tudi za javne in zasebne ključe.

Upravljanje ključev je množica tehnik in postopkov, ki podpirajo dogovor in vzdrževanje relacij ključev med pooblaščenimi strankami/sogovorniki.

Infrastruktura javnih ključev (PKI): podporni servisi (tehnološki, pravni, komercialni, itd.), ki so potrebni, da lahko tehnologijo javnih ključev uporabimo za večje projekte.

Sistemi z javnimi ključi imajo prednost pred sistemi s tajnimi ključi, saj za izmenjavo tajnih ključev ne potrebujejo varnega kanala.

Večina sistemov z javnimi ključi (npr. RSA) je tudi do 100-krat počasnejša od simetričnih sistemov (npr. DES). Zato v praksi uporabljamo za šifriranje *daljših* besedil simetrične sisteme.

Obravnavali bomo več različnih protokolov za tajne ključe. Razlikovali bomo med *distribucijo ključev* in *uskladitvijo ključev*.

Sistem distribucije ključev je mehanizem, kjer na začetni stopnji verodostojna agencija generira in distribuira tajne podatke uporabnikom tako, da lahko vsak par uporabnikov kasneje izračuna ključ, ki je nepoznan ostalim.

Uskladitev ključev označuje protokol, kjer dva ali več uporabnikov sestavjo skupen tajni ključ, s komunikacijo po javnem kanalu. Vrednost ključa je določena s funkcijo vhodnih podatkov.

Obstaja potreba po zaščiti pred potencialnimi nasprotniki, tako pasivnimi kot tudi aktivnimi.

Pasivni sovražnik je osredotočen na prisluškovanje sporočilom, ki se pretakajo po kanalu.
Več nevšečnosti nam lahko naredi *aktivni* sovražnik:

- spreminjanje sporočil,
- shranjevanje sporočil za kasnejšo uporabo,
- maskiranje v uporabnika omrežja.

Cilj *aktivnega* sovražnika uporabnikov U in V je lahko:

- prelisičiti U in V tako, da sprejmeta neveljaven ključ kot veljaven,
- prepričati U in V, da sta si izmenjala ključ, čeprav si ga v resnici nista.

Center zaupanja

V omrežju, ki ni varno, se v nekaterih shemah pojavi agencija, ki je odgovorna za

- potrjevanje identitete,
- izbiro in prenos ključev
- itd.

Rekli ji bomo **center zaupanja** ali **verodostojna agencija** (angl. Trusted Authority – TA ali Trusted Third Party – TTP). Uporabljali bomo oznako **TA**.

Distribucija ključev

- "Point-to-point" distribucija po varnem kanalu:
  - zaupni kurir,
  - enkratna registracija uporabnikov,
  - prenos po telefonu.
- Neposreden dostop do overjene javne datoteke:
  - avtentična drevesa,
  - digitalno podpisana datoteka.
- Uporaba "on-line" zaupnih strežnikov,
- "Off-line" certifikatna agencija (CA).

Point-to-point

Predpostavimo, da imamo

– omrežje z n uporabniki,
– agencija TA generira in preda enolično določen ključ vsakemu paru uporabnikov omrežja.

Če imamo varen kanal med TA in vsakim uporabnikom omrežja, potem dobi vsak posameznik $n-1$ ključev, zahtevnost problema pa je vsaj $\mathcal{O}(n^2)$.

Ta rešitev ni praktična celo za relativno majhne n.

Želimo si boljšo rešitev, npr. z zahtevnostjo $\mathcal{O}(1)$.

Blomova shema

Naj bo javno p praštevilo večje od danega $n \in \mathbb{N}$ in naj bo $k \in \mathbb{N}$ za katerega velja $k \leq n-2$.

TA pošlje po varnem kanalu $k+1$ elementov $\mathbb{Z}_p$ vsaki osebi in nato si lahko vsak par $\{U, V\}$ izračuna svoj ključ $K_{U,V} = K_{V,U}$.

Število k je velikost največje koalicije, proti kateri bo shema še vedno varna.

Paul R. Halmos

"...the source of all great mathematics is the special case, the concrete example. It is frequent in mathematics that every instance of a concept of seemingly great generality is in essence the same as a small and concrete special case."

I Want to be a Mathematician, Washington: MAA Spectrum, 1985.

???

" Sometimes a research is a lot of hard work in looking for the easy way."

David Hilbert (-1900)

"The art of doing mathematics consists in finding that special case which contains all the germs of generality."

Najprej opišimo shemo v primeru, ko je $k = 1$.

- *Izberemo javno praštevilo p.*
- *TA izbere tri naključne elemente $a, b, c \in \mathbb{Z}_p$ (ne nujno različne) in oblikuje polinom*

$$f(x, y) = a + b(x + y) + c\,xy \mod p.$$

- *Za vsakega uporabnika U izbere TA javni $r_U \in \mathbb{Z}_p$, tako da so le-ti medseboj različni.*

- *Za vsakega uporabnika U izračuna TA polinom*

$$g_U(x) = f(x, r_U) \bmod p$$

in mu ga pošlje po varnem kanalu.

Opozorimo, da je $g_U(x)$ linearen polinom, tako da ga lahko zapišemo v naslednji obliki

$$g_U(x) = a_U + b_U x,$$

kjer je

$$a_U = a + br_U \bmod p \quad \text{in} \quad b_U = b + cr_U \bmod p.$$

- *Za medsebojno komunikacijo osebi U in V uporabita ključ*

$$K_{U,V} = K_{V,U} = f(r_U, r_V) = a + b(r_U + r_V) + c\,r_U r_V \bmod p.$$

Uporabnika U in V izračunata svoja ključa $K_{U,V}$ in $K_{U,V}$ zaporedoma s

$$f(r_U, r_V) = g_U(r_V) \quad \text{in} \quad f(r_U, r_V) = g_V(r_U).$$

Izrek 1. *Blomova shema za $k = 1$ je brezpogojno varna pred posameznimi uporabniki.*

Dokaz: Recimo, da želi uporabnik W izračunati ključ

$$K_{U,V} = a + b(r_U + r_V) + c\,r_U r_V \bmod p.$$

Vrednosti r_U in r_V so javne, a, b in c pa ne.
Oseba W pozna vrednosti

$$a_W = a + br_W \bmod p \quad \text{in} \quad b_W = b + cr_W \bmod p,$$

ker sta to koeficienta polinoma $g_W(x)$, ki ju je dobila od agencije TA.

Pokažimo, da je informacija, poznana osebi W, konsistentna s poljubno vrednostjo $\ell \in \mathbb{Z}_p$ za ključ $K_{U,V}$, tj. W ne more izločiti nobene vrednosti za $K_{U,V}$.

Poglejmo si naslednjo matrično enačbo v $(\mathbb{Z}_p)$:

$$\begin{pmatrix} 1 & r_U + r_V & r_U r_V \\ 1 & r_W & 0 \\ 0 & 1 & r_W \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \ell \\ a_W \\ b_W \end{pmatrix}.$$

Prva enačba vsebuje hipotezo, da je $K_{U,V} = \ell$, drugi dve enačbi pa sledita iz definicije števil a_W in b_W.

Determinanta zgornje matrike je

$$r_W^2 + r_U r_V - (r_U + r_V) r_W = (r_W - r_U)(r_W - r_V).$$

Iz $r_W \neq r_U$ in $r_W \neq r_V$ sledi, da je determinanta različna od nič in zato ima zgornji sistem enolično rešitev za a, b in c.

Koalicija uporabnikov $\{W, X\}$ pa ima štiri enačbe ter tri neznanke in od tod zlahka izračuna a, b in c ter končno še polinom $f(x,y)$, s katerim dobi vsak ključ. ■

Posplošitev

Za splošno shemo (tj. shemo, ki je varna pred koalicijo velikosti k) je potrebna ena sama sprememba. Pri drugem koraku TA uporablja polinom $f(x,y)$ naslednje oblike

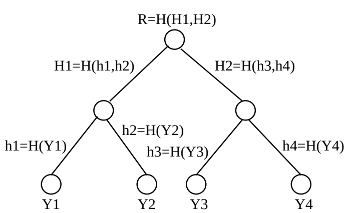
$$f(x,y) = \sum_{i=0}^{k} \sum_{j=0}^{k} a_{ij}\, x^i y^j \mod p,$$

kjer je $a_{ij} \in \mathbb{Z}_p$ za $0 \leq i, j \leq k$ in $a_{ij} = a_{ji}$ za vsak i, j. Ostali del protokola se ne spremeni.

Avtentična drevesa

- Merkle, 1979.
- metoda za hranjenje javno dostopnih in preverljivo overjenih podatkov
- Uporaba:
  - avtentičnost velike datoteke javnih ključev,
  - servis časovnih oznak (Timestamping).

Primer: H je zgoščevalna funkcija brez trčenj.



Vzdržujemo avtentičnost korenske vrednosti R (npr. s podpisom agencije TA).

Za avtenticiranje javne vrednosti Y_2:

- sledi (natanko določeno) pot od Y_2 do korena,
- pridobi vrednosti h_1, H_2, R,
- preveri avtentičnost R,
- preveri $R = H(H(h_1, H(Y_2)), H_2)$.

Če ima drevo n javnih vrednosti, je dolžina avtenticiranja kvečjemu $\lceil \log_2 n \rceil$.

Slaba stran: dodajanje in brisanje javnih vrednosti je lahko precej zamudna.

Diffie-Hellmanova distribucija ključev

Zaradi enostavnosti bomo delali v obsegu $\mathbb{Z}_p$, kjer je p praštevilo in α generator grupe $\mathbb{Z}_p^*$.

Naj bo ID(U) oznaka za določeno informacijo, ki enolično identificira osebo U (npr. ime, e-pošta, telefonska številka itd).

Vsak uporabnik si izbere tajni/zasebni $a_U \in \{0, 1, \ldots, p-2\}$, in naj bo

$$b_U = \alpha^{a_U} \mod p.$$

Agencija TA si izbere shemo za digitalni podpis z javnim algoritmom za preverjanje podpisov $\mathrm{ver}_{\mathrm{TA}}$ in tajnim algoritmom za podpisovanje $\mathrm{sig}_{\mathrm{TA}}$.

Nazadnje privzemimo še, da so vse informacije zgoščene z javno zgoščevalno funkcijo, preden jih podpišemo, vendar pa zaradi estetskih razlogov ne bomo omenjali zgoščevalne funkcije pri opisu protokolov.

Za osebo U bo agencija TA izdala naslednji **certifikat**:

$$C(U) = (\,\mathrm{ID}(U),\, b_U, \mathrm{sig}_{\mathrm{TA}}(\mathrm{ID}(U),\, b_U)\,)$$

(TA ne potrebuje zasebne vrednosti a_U).

- *Izberemo javno praštevilo p in javen primitivni element $\alpha \in \mathbb{Z}_p^*$.*
- *Oseba V izračuna*
  $$K_{U,V} = \alpha^{a_U a_V} \bmod p \;=\; b_U^{a_V} \bmod p,$$
  z uporabo javne vrednosti b_U iz certifikata osebe U in s svojo zasebno vrednostjo a_V.
- *Oseba U izračuna*
  $$K_{U,V} = \alpha^{a_U a_V} \bmod p \;=\; b_V^{a_U} \bmod p,$$
  z uporabo javne vrednosti b_V iz certifikata osebe V in s svojo zasebno vrednostjo a_U.

Podpis agencije TA preprečuje osebi W, da spreminja certifikate, torej je dovolj preprečiti pasivne napade.

Ali lahko oseba W izračuna $K_{U,V}$, če je $W \neq U, V$, tj, če poznamo $\alpha^{a_U} \bmod p$ in $\alpha^{a_V} \bmod p$ ne pa tudi a_U ali a_V, ali je mogoče izračunati $\alpha^{a_U a_V} \bmod p$?

To bomo imenovali **Diffie-Hellmanov** problem.

Očitno je **Diffie-Hellmanova distribucija ključev** varna natanko tedaj, ko je varen **Diffie-Hellmanov** problem.

Izrek 2. *Razbitje ElGamalovega kriptosistema je ekvivalentno reševanju Diffie-Hellmanovega problema.*

Dokaz: Spomnimo se, kako potekata ElGamalovo šifriranje in odšifriranje. Ključ je $K = (p, \alpha, a, \beta)$, kjer $\beta = \alpha^a \bmod p$ (a je tajni in p, α in β so javni). Za tajno naključno število $k \in \mathbb{Z}_{p-1}$ je

$$e_K(x, k) = (y_1, y_2),$$

kjer $y_1 = \alpha^k \bmod p$ in $y_2 = x\beta^k \bmod p$.

Za $y_1, y_2 \in \mathbb{Z}_p^*$ je $d_K(y_1, y_2) = y_2(y_1^a)^{-1} \bmod p$.

Predpostavimo, da imamo algoritem A, ki reši Diffie-Hellmanov problem in podano ElGamalovo šifriranje (y_1, y_2). Z uporabo algoritma A na podatkih p, α, y_1 in β dobimo vrednost

$$A(p, \alpha, y_1, \beta) = A(p, \alpha, \alpha^k, \alpha^a) =$$
$$= \alpha^{ka} \bmod p = \beta^k \bmod p.$$

Potem odšifriranje (y_1, y_2) lahko enostavno izračunamo:

$$x = y_2(\beta^k)^{-1} \bmod p.$$

Predpostavimo, da imamo še algoritem B, ki izvrši ElGamalovo odšifriranje. Torej B vzame podatke p, α, β, y_1 in y_2 in izračuna

$$x = y_2(y_1^{\log_\alpha \beta})^{-1} \bmod p.$$

Naj bodo p, α, β in γ podatki Diffie-Hellmanovega problema. Torej je $\beta = \alpha^b$ in $\gamma = \alpha^c$ za neka $b, c \in \mathbb{N}$, ki nista poznana, pa vendar lahko izračunamo

$$(B(p, \alpha, \beta, \gamma, 1))^{-1} = (1(\gamma^{\log_\alpha \beta})^{-1})^{-1} \bmod p =$$
$$= \gamma^{\log_\alpha \beta} \bmod p = \alpha^{c \cdot b} \bmod p,$$

torej DH-ključ, kar smo tudi želeli. ■

Certifikati

Certifikatna agencija (CA) izda certifikat ***C(U)***, ki poveže uporabnika U z njegovim javnim ključem.

Sestavljen je iz:

- **podatkovnega dela *D(U)***:
  uporabnikova identifikacija, njegov javni ključ in druge informacije kot npr. veljavnost,
- **podpisanega dela $\mathrm{sig}_{\mathrm{CA}}(D(U))$**:
  CA-jev podpis podatkovnega dela.

B pridobi avtentično kopijo A-jevega javnega ključa na naslednji način:

- pridobi avtentično kopijo javnega ključa CA (npr. dobljenega z brskalnikom ali operacijskim sistemom),
- pridobi $C(U)$ (preko nezavarovanega kanala),
- preveri podpis $\mathrm{sig}_{\mathrm{CA}}(D(U))$.

Opombe: 1. CA ni potrebno zaupati uporabniških zasebnih ključev.

2. CA moramo zaupati, da ne bo izdajala ponarejenih certifikatov.

Infrastruktura javnih ključev (PKI)

Nekatere komponente:

- format certifikata,
- proces certificiranja,
- razdeljenanje certifikatov,
- modeli zaupanja,
- preklic certifikatov,
- politika certificiranja: podrobnosti o namenu in obsegu uporabe določenega certifikata.
- Izjava o prakticiranju certificiranja (CPS) (postopki in politike CA).

Format certifikata: X.509 Ver.3

- X.509 originalno predlagan za podporo X.500, ki omogoča servis imenikov na velikih računalniških mrežah.
- Ver. 1 izide leta '88;
  Ver. 2 leta '93;
  Ver. 3 pa leta '97.
- Najnovejši PKI produkti uporabljajo Ver.3.
- Dopušča precejšnjo fleksibilnost.

Podatkovna polja zajemajo:

- verzijo številke certifikata,
- certifikatovo serijsko številko,
- CA-jev podpisni algoritem ID,
- CA-jevo ime v X.500,
- rok veljave,
- uporabnikovo X.500 ime,
- uporabnikova informacija o javnem ključu,
  – algoritmov ID, vrednost javnega ključa,
- Ext. polja: omogočajo vključevanje poljubnega števila dodatnih polj. Primeri:
  – politika certifikata in politika prirejanja, pot certificiranja, omejitve.

Proces certifikacije

1. Generiranje para ključev za CA-jev podpis:
   - varnost zasebnega ključa CA je osrednja,
   - po možnosti opravljena v nepropustni napravi,
   - deljenje delov zasebnega ključa večim modulom, tako da certifikat ne more biti izdan s strani posameznega modula.
2. Generiranje para ključev osebe A:
   - bodisi s stani osebe A ali CA.
3. Zahteva za A-jev certifikat:
   - lahko, da bo CA kasneje potrebovala to zahtevo,
   - avtentičnost zahteve je potrebna.

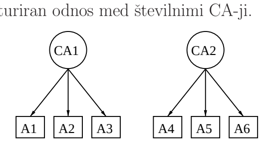
4. Identiteta osebe A je preverjena:
   - to je lahko zamudno in drago v praksi,
   - preložiti to delo na Registration Authority (RA); npr. pošto ali banko,
   - RA generira registracijski certifikat in ga prosledi CA za izdajo certifikata.
5. A-jev par ključev je preverjen:
   - CA preveri, da je javni ključ veljaven, tj. zasebni ključ logično obstaja,
   - A dokaže, da ima zasebni ključ.
6. CA naredi A-jev certifikat.
7. A preveri, da je certifikat izpraven:
   - CA lahko zahteva od A še potrdilo od prejemu.

Primer: Verisignov digitalni ID

- www.verisign.com/client/index.html
- Certifikat za javno podpisovanje in javno šifriranje.
- Certifikati so hranjeni v brskalniku ali e-poštni programski opremi.
- Brezplačni certifikati za 60-dnevno preiskusno dobo.
- Trije razredi certifikatov:
  – odgovornost prevzema Verisign (US \$100, \$5,000, \$100,000),
  – potrditev identitete,
  – zaščita CA-jevega zasebnega ključa,
  – zaščita posameznih uporabnikovih zasebnih ključev.
- `www.versign.com/repository/index.html`

Model zaupanja

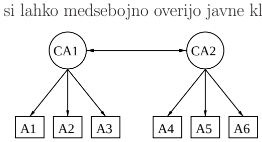
- strukturiran odnos med številnimi CA-ji.



- Stranke dobijo avtentične kopije CA-jevega javnega ključa (zunaj tekočega obsega - out-of-band, npr. med certifikacijo).
- Kako lahko A_1 preveri podpis sporočila osebe A_5? Tj. kako lahko dobi overjeno kopijo javnega ključa od A_1?
- A_1 potrebuje overjeno kopijo javnega ključa od CA_2.

Navzkrižna certifikacija

- CA-ji si lahko medsebojno overijo javne ključe

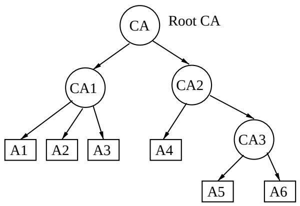


- A_1 pridobi A_5-jev overjeni javni ključ:
  – Pridobitev certifikatov CA_2 in A_5 z javnega (nezaščitenega, ne-overjenega) imenika.
  – Preveri od CA_1 podpisan certifikat CA_2 (s tem dobi overjeno kopijo javnega ključa CA_2).
  – Preveri od CA_2 podpisan certifikat A_5 (s tem dobi overjeno kopijo javnega ključa A_5).

Pomisleki glede navskrižnega certificiranja

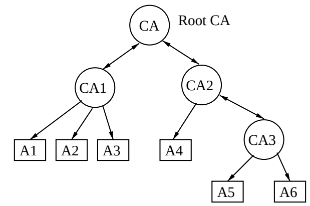
- Ali je CA_1 odgovoren osebi A_1 za varnostne probleme v domeni CA_2?
  – Potencialni problemi so lahko omejeni z izjavo v politiki CA_1 za CA_2 certifikate.
  – CA_1 mora previdno preveriti CA_2jev CPS.
  – Neodvisni pregled politike CA_2 bo pomagal.
- Ali je CA_1 odgovoren osebam iz CA_2 domene za varnostne probleme v svoji domeni?
- Vprašanje: ali bodo problemi navskrižnega certificiranja za obsežnejše aplikacije *kdaj* rešeni?

Strogo hierarhičen model



- Vsi vipis začenjajo z overjeno kopijo korenskega javnega ključa.
- Zadrški:
  - vse zaupanje je odvisno od korenskega CA,
    - ∗ rešitev: razdeli dele zasebnega ključa;
  - Certifikatne verige lahko postanejo predolge,
    - ∗ rešitev: nekatere certifikate spravimo v cache.
  - Certifikatne verige zahtevane celo za osebe znotraj iste CA,
    - ∗ rešitev: nekatere certifikate spravimo v cache.

Povratni hierarhičen model



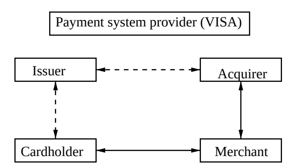
- CA lahko preveri javni ključ starševskega CA.
- Vsaka oseba prične z overjenim javnim ključem svojega CA.
- Najkrajša veriga zaupanja med A in B je pot od A do najmlajšega skupnega prednika od A in B, in nato navzdol do B.

Secure Electronic Transaction (SET)

- Standard, ki sta ga predlagala Visa in MasterCard (Feb 1996).
- Glej www.setco.org
- Cilj: varne transakcije s kreditnimi karticami prko Interneta.

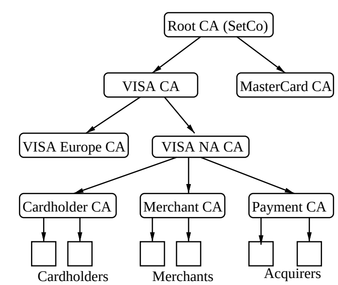
- Sodelujoči pri transakciji s kreditno kartico:
  - *Izdajatelj*: finančno podjetje, ki izdaja kreditne kartice.
  - *Lastnik kartice*: Nepooblaščen imetnik kreditne kartice holder of a credit card who is registered with the corresponding issuer.
  - *Prodajalec*: trgovec, services, or information, who accepts payment electronically.
  - *Dobavitelj*: finančna inštitucija, ki podpira prodajalca s tem, da ponuja servis za procesiranje transakcij z bančnimi karticami.

- Plačilo s kreditno kartico:



- Po Internetu: $C \longleftrightarrow M$ in $M \longleftrightarrow A$.
- Šifriranje se uporabi za zaščito številk kreditnih kartic med prenosom po Internetu; številke niso razkrite prodajalcu.
- Digitalni podpisi se uporabljajo za celovitost podatkov in overjanje udeleženih strank.

SET-ov hierarhični PKI



Preklic certifikata

- Razlogi za preklic certifikata:
  - kompromitiran ključ (redko).
  - Lastnik zapusti organizacijo.
  - Lastnik spremeni vlogo v organizaciji.
- Primer: Scotiabank tele-banking PKI:
  - Čez 90,026 certifikatov izdanih do aprila 21, 1999.
  - Čez 19,000 certifikatov preklicanih.
- Uporabnik naj bi preveril veljavnost certifikata pred njegovo uporabo.
- Preklic je enostaven v primeru on-line CA.

Certifikatne preklicne liste (CRL)

- Lista preklicanih certifikatov, ki je podpisana in periodično izdana od CA.
- Uporabnik preveri CRL predno uporabi certifikat.

Problemi z CRLs

- časovna prerioda CRL
  - Čas med preklicom in obnovitvijo CRL.
- velikost CRL
  - Delta CRL: vključuje le zadnje preklicane certifikate.
  - Groupiraj razloge za preklic.
  - Delitvene točke: revocation data is split into buckets; each certificate contains data that determines the bucket it should be placed in (patent: Entrust Technologies).
  - Uporabi avtentikacijska drevesa (komercializacija: Valicert).

Kerberos

Doslej smo spoznali sisteme, kjer vsak par uporabnikov izračuna fiksen ključ, ki se ne spreminja.
Zaradi tega je preveč izpostavljen nasprotnikom.

Zato bomo vpeljali tako imenovan sejni ključ, ki se oblikuje brž, ko se pojavita dva, ki želita komunicirati.

Tak sistem, ki uporablja simetrične sisteme, je Kerberos. Slabost tega sistema pa je zahteva po sinhronizaciji ur uporabnikov omrežja.
Določena časovna variacija je dovoljena.

Predpostavimo, da vsak uporabnik deli z agencijo TA tajni DES ključ K_U. Tako kot prej imejmo tudi $\mathrm{ID}(U)$.

Ko dobi agencija TA zahtevo po novem sejnem ključu, si TA izbere naključni ključ K, zabeleži časovno oznako T (timestamp), določi življenjsko dobo L (lifetime) za ključ K ter vse skupaj pošlje uporabnikoma U in V.

Prenos sejnega ključa z uporabo Kerberosa

- *Uporabnik U zahteva od agencije TA sejni ključ za komunikacijo z uporabnikom V.*
- *Agencija TA izbere naključni sejni ključ K, časovno oznako T in življenjsko dobo L.*
- *TA izračuna* $m_1 = e_{K_U}(K, \mathrm{ID}(V), T, L)$ *in* $m_2 = e_{K_V}(K, \mathrm{ID}(U), T, L)$ *ter ju pošlje uporabniku U.*
- *U uporabi odšifrirno funkcijo d_{K_U}, da dobi iz m_1 K, T, L in $\mathrm{ID}(V)$. Potem izračuna $m_3 = e_K(\mathrm{ID}(U), T)$ in ga pošlje osebi V skupaj s sporočilom m_2, ki ga je dobil od agencije TA.*

- *V uporabi odšifrirno funkcijo d_{K_V}, da dobi iz m_2 K, T, L in $\mathrm{ID}(U)$. Potem uporabi d_K, da dobi T in $\mathrm{ID}(U)$ iz m_3. Preveri, da sta tako dobljeni vrednosti za T in $\mathrm{ID}(U)$ enaki prejšnjim. Če je tako, potem izračuna še*

$$m_4 = e_K(T+1)$$

  in ga pošlje uporabniku U.
- *U odšifrira m_4 z uporabo e_K in preveri, ali je rezultat enak $T+1$.*

V tem protokolu se prenašajo različne funkcije sporočil.

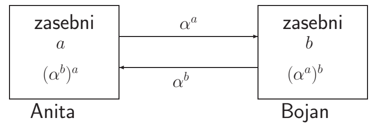
Sporočili m_1 in m_2 poskrbita za tajnost pri prenosu sejnega ključa K.

Sporočili m_3 in m_4 se uporabljata kot potrdilo sejnega ključa K tako, da se U in V prepričata, da imata res isti sejni ključ K.

Diffie-Hellmanova uskladitev ključev

Naj bo p praštevilo in α generator multiplikativne grupe $\mathbb{Z}_p^*$. Naj bosta oba javno poznana (ali pa naj ju oseba U sporoči osebi V).

1. *Oseba U izbere naključen a_U,* $0 \leq a_U \leq p-2$, *izračuna* $\alpha^{a_U} \bmod p$ *in ga pošlje osebi V.*
2. *Oseba V izbere naključen a_V,* $0 \leq a_V \leq p-2$, *izračuna* $\alpha^{a_V} \bmod p$ *in ga pošlje osebi U.*
3. *Osebi U in V izračunata zaporedoma* $K = (\alpha^{a_V})^{a_U} \bmod p$ *in* $K = (\alpha^{a_U})^{a_V} \bmod p$.



Anita in Bojan si delita skupni element grupe:

$$(\alpha^a)^b = (\alpha^b)^a = \alpha^{ab}.$$

Edina razlika med tem protokolom in pa Diffie-Hellmanovim protokolom za distribucijo ključev je, da si izberemo nova eksponenta a_U in a_V uporabnikov U in V zaporedoma vsakič, ko poženemo ta protokol.

Varnost Diffie-Hellmanovega protokola

Protokol ni varen pred aktivnim napadalcem, ki prestreže sporočila in jih nadomesti s svojimi. Ta napad bomo imenovali **napad srednjega moža**.

$$U \quad \underset{\alpha^{a'_V}}{\overset{\alpha^{a_U}}{\rightleftarrows}} \quad W \quad \underset{\alpha^{a_V}}{\overset{\alpha^{a'_U}}{\rightleftarrows}} \quad V$$

Na koncu sta osebi U in V vzpostavili z napadalcem W zaporedoma ključa $\alpha^{a_U a'_V}$ in $\alpha^{a'_U a_V}$.

Tako bo zašifrirano sporočilo osebe U odšifriral napadalec W ne pa oseba V.

Uporabnika U in V bi bila rada prepričana, da ni prišlo namesto medsebojne izmenjave sporočil do izmenjave z napadalcem W.

Potrebujeta protokol za medsebojno avtentikacijo (predstavitev).

Dobro bi bilo, če bi potekala avtentikacija istočasno z uskladitvijo ključev, saj bi s tem onemogočili aktivnega napadalca.

Overjena uskladitev ključev

Diffie, Van Oorschot in Wiener so predlagali protokol **uporabnik-uporabniku** (station-to-station - STS), ki je protokol za *overjeno uskladitev ključa* in je modifikacija Diffie-Hellmanove uskladitve ključev.

Vsak uporabnik ima **certifikat (potrdilo)**

$$C(U) = \Big(\mathrm{ID}(U), \mathrm{ver}_U, \mathrm{sig}_{\mathrm{TA}}\big(\mathrm{ID}(U), \mathrm{ver}_U\big)\Big),$$

kjer je shranjena njegova identifikacija $\mathrm{ID}(U)$.

Poenostavljen protokol uporabnik-uporabniku

1. *Oseba U izbere naključen* $a_U \in \{0, \ldots, p-2\}$, *izračuna* $\alpha^{a_U} \bmod p$ *in pošlje osebi V.*
2. *Oseba V izbere naključen* $a_V \in \{0, \ldots, p-2\}$, *izračuna* $\alpha^{a_V} \bmod p$,
   $K = (\alpha^{a_U})^{a_V} \bmod p$ in $y_V = \mathrm{sig}_V(\alpha^{a_V}, \alpha^{a_U})$,
   ter pošlje potrdilo $(C(V), \alpha^{a_V}, y_V)$ *osebi U.*

3. *Oseba U izračuna* $K = (\alpha^{a_V})^{a_U} \bmod p$ *ter preveri podpis y_V z uporabo* ver_V *in potrdilo $C(V)$ z* $\mathrm{ver}_{\mathrm{TA}}$.

   Nato izračuna $y_U = \mathrm{sig}_U(\alpha^{a_U}, \alpha^{a_V})$ *in pošlje potrdilo* $(C(U), y_U)$ *osebi V.*
4. *Oseba V preveri podpis y_U z uporabo* ver_U *in potrdilo $C(U)$ z uporabo* $\mathrm{ver}_{\mathrm{TA}}$.

Varnost protokola STS

Uporabnika U in V si izmenjata naslednje informacije (izpustimo potrdila):

$$U \quad \begin{array}{c} \xrightarrow{\quad \alpha^{a_U} \quad} \\ \xleftarrow{\quad \alpha^{a_V},\ \mathrm{sig}_V(\alpha^{a_V},\alpha^{a_U}) \quad} \\ \xrightarrow{\quad \mathrm{sig}_U(\alpha^{a_U},\alpha^{a_V}) \quad} \end{array} \quad V$$

Kaj lahko naredi napadalec W (mož na sredini):

$$U \quad \begin{array}{c} \xrightarrow{\quad \alpha^{a_U} \quad} \\ \xleftarrow{\alpha^{a'_V},\ \mathrm{sig}_V(\alpha^{a'_V},\alpha^{a_U})=?} \\ \xrightarrow{\quad \mathrm{sig}_U(\alpha^{a_U},\alpha^{a'_V}) \quad} \end{array} \quad W \quad \begin{array}{c} \xrightarrow{\quad \alpha^{a'_U} \quad} \\ \xleftarrow{\alpha^{a_V},\ \mathrm{sig}_V(\alpha^{a_V},\alpha^{a'_U})} \\ \xrightarrow{\quad \mathrm{sig}_U(\alpha^{a'_U},\alpha^{a_V})=? \quad} \end{array} \quad V$$

Poenostavljeni STS protokol je torej varen pred napadom srednjega moža.

Tako oblikovan protokol ne vsebuje potrditve ključa, kakor je slučaj v Kerberosovi shemi.

Protokol, v katerem je vključena potrditev ključa:

$$y_V = e_K(\mathrm{sig}_V(\alpha^{a_V},\alpha^{a_U})), \quad y_U = e_K(\mathrm{sig}_U(\alpha^{a_U},\alpha^{a_V}))$$

se imenuje STS protokol.

MTI protokoli

Matsumoto, Takashima, Imai so modificirali Diffie-Hellmanovo uskladitev ključev, tako da uporabniki U in V ne potrebujejo podpisov.

Kadar moramo izmenjati dve pošiljki, pravimo, da gre za **protokole z dvema izmenjavama**.

Predstavili bomo en njihov protokol.

Osnovne predpostavke so enake kot pri Diffie-Hellmanovi uskladitvi ključev: praštevilo p in generator α multiplikativne grupe $\mathbb{Z}_p^*$ sta javna.

Vsak uporabnik U ima svoj *zasebni* eksponent a_U $(0 \leq a_U \leq p-2)$ in *javno* vrednost $b_U = \alpha^{a_U} \bmod p$.

Agencija TA ima shemo za digitalni podpis, z *javnim* algoritmom $\mathrm{ver}_{\mathrm{TA}}$ in *tajnim* algoritmom $\mathrm{sig}_{\mathrm{TA}}$.

Vsak uporabnik U ima svoj certifikat:

$$C(U) = (\mathrm{ID}(U), b_U, \mathrm{sig}_{\mathrm{TA}}(\mathrm{ID}(U), b_U)).$$

1. *Oseba U izbere naključen $r_U \in \{0,\dots,p-2\}$, izračuna $s_U = \alpha^{r_U} \bmod p$ in pošlje osebi V $(C(U), s_U)$.*
2. *Oseba V izbere naključen $r_V \in \{0,\dots,p-2\}$, izračuna $s_V = \alpha^{r_V} \bmod p$ in pošlje osebi U $(C(V), s_V)$.*
3. *Osebi U in V izračunata zaporedoma*
   $$K = s_V^{a_U} b_V^{r_U} \bmod p \quad \text{in} \quad K = s_U^{a_V} b_U^{r_V} \bmod p,$$
   kjer sta b_V in b_U zaporedoma iz $C(V)$ in $C(U)$.

Varnost protokola MTI

Ta MTI protokol je enako varen pred pasivnimi sovražniki kot Diffie-Hellmanov protokol.

Varnost pred aktivnimi sovražniki je bolj vprašljiva. Brez uporabe podpisnega algoritma nismo varni pred napadom srednjega moža.

$$U \quad \begin{array}{c} \xrightarrow{\quad C(U), \alpha^{r_U} \bmod p \quad} \\ \xleftarrow[\quad C(V),\ \alpha^{a_V} \bmod p \quad]{} \end{array} \quad V$$

Ključ uporabnikov, ki komunicirata, je težko izračunati, ker je v ozadju težko izračunljiv diskretni logaritem.

Tej lastnosti pravimo **implicitna overitev ključev**.

Uskladitev ključev s ključi, ki se sami overijo

Giraultova shema ne potrebuje certifikatov, saj uporabnike razlikujejo že njihovi javni ključi in identifikacije.

Vsebuje lastnosti RSA sheme in diskretnega logaritma.

Uporabnik naj ima identifikacijo $\mathrm{ID}(U)$.
Javni ključ za osebno overitev dobi od agencije TA.

Naj bo $n = pq$, kjer je $p = 2p_1 + 1$, $q = 2q_1 + 1$, in so p, q, p_1, q_1 velika praštevila. Potem je

$$(\mathbb{Z}_n^*, \cdot) \sim (\mathbb{Z}_p^* \times \mathbb{Z}_q^*, \cdot).$$

Največji red poljubnega elementa v $\mathbb{Z}_n^*$ je najmanjši skupni večkratnik elementov $p-1$ in $q-1$ oziroma $2p_1q_1$.

Naj bo α generator ciklične podgrupe v $\mathbb{Z}_p^*$ reda $2p_1q_1$, problem diskretnega logaritma v tej podgrupi pa naj bo računsko prezahteven za napadalca.

Javni ključ za osebno overitev

Naj bosta števili n, α javni, števila p, q, p_1, q_1 pa naj pozna samo agencija TA.

Število e je javni RSA šifrirni eksponent in ga izbere agencija TA, $d = e^{-1} \bmod \varphi(n)$ pa je tajni odšifrirni eksponent.

1. *Oseba U izbere tajni eksponent a_U, izračuna $b_U = \alpha^{a_U} \bmod n$ in izroči a_U ter b_U agenciji TA.*
2. *Agencija TA izračuna $p_U = (b_U - \mathrm{ID}(U))^d \bmod n$ ter ga izroči osebi U.*

Giraultov protokol za uskladitev ključev

1. *Oseba U izbere naključen zasebni r_U, izračuna*
$$s_U = \alpha^{r_U} \bmod n$$
ter pošlje $\mathrm{ID}(U)$, p_U in s_U osebi V.
2. *Oseba V izbere naključen zasebni r_V, izračuna*
$$s_V = \alpha^{r_V} \bmod n$$
ter pošlje $\mathrm{ID}(V)$, p_V in s_V osebi U.
3. *Osebi U in V izračunata ključ K zaporedoma z*

$$s_V^{a_U}(p_V^e + \mathrm{ID}(V))^{r_U} \bmod n, \quad s_U^{a_V}(p_U^e + \mathrm{ID}(U))^{r_V} \bmod n.$$

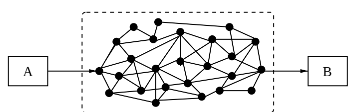
Varnost Giraultovega protokola

Ključ za osebno overitev varuje pred sovražniki.

Protokol implicitno overi ključe, zato napad srednjega moža ni možen.

Agencija TA je prepričana, da uporabnik pozna vrednost števila a predno izračuna ključ za osebno overitev.

Internetne aplikacije



- ftp: File Transfer Protocol
- http: HyperText Transfer Protocol
- smtp: Simple Mail Transfer Protocol

TCP – Transport Control Protocol
IP – Internet Protocol

TCP/IP

Protokolov sklad:

Application: http, ftp, smtp
TCP
IP
⋮
physical

TCP/IP paket:

IP header	TCP header	Application header	Data

IP header:

source address	destination address	------

Nekateri napadi

- **IP address spoofing** (slov. ponarejanje naslovov)
  rešitev: overi glavo IP paketa
- **IP packet sniffing** (slov. vohljanje za IP paketi)
  rešitev: zašifriraj IP payload (vse kar se prenaša)
- **Traffic analysis** (slov. Analiza prometa)
  rešitev: zašifriraj pošiljateljev in prejemnikov naslov

Varnost znotraj TCP/IP

Varnostni protokoli so prisotni na različnih nivojih TCP/IP sklada.

1. IP nivo: IPsec.
2. Transportni nivo: SSL/TLS.
3. Aplikacijski nivo: PGP, S/MIME, SET, itd.

Internet Engineering Task Force (IETF)

- Sprejema standarde za razvoj Internetne arhitekture in omogoča nemoteno delovanje Interneta.
- Odprta za vse zainteresirane posameznike: `www.ietf.org`
- Delo, ki ga opravljajo delovne skupine povezane z varnostjo (Security Area) pokrivajo:

– IP Security Protocol (IPsec)

– Transport Layer Security (TLS)

– S/MIME Mail Security

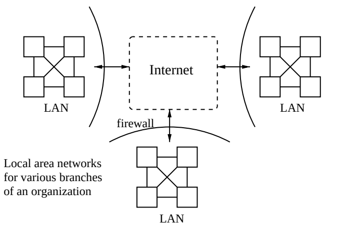
– Odprto specifikacijo za PGP (OpenPGP)

– Secure Shell (secsh)
(Nova verzija ssh protokola, ki omogoča varno prijavo na oddaljene šifre in varen prenos datotek.)

– X.509 Public-Key Infrastructure (PKIX)

IPsec: Virtual Private Networks (VPNs)

Omogočajo šifriranje in overjanje (overjanje izvora podatkov, celovitost podatkov) na IP layer.



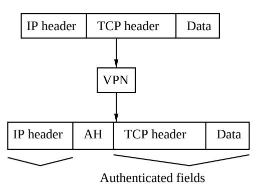
Gradniki IPsec

- Security Association (SA):
  - upravlja algoritme in ključe med sogovorniki,
  - vsaka glava IPsec se nanaša na Security Association preko Security Parameter Index (SPI).
- Upravljanje s ključi:
  - dogovor o ključu z Diffie-Hellmanovo shemo (OAKLEY),
  - kreira ključe za Security Association,
  - upravljanje z javnimi ključi, ki ni pokrito v IPsec.
- Trije načini IPsec servisov:
  - AH: overjanje,
  - ESP: šifriranje + overjanje.

IPsec glava za overjanje (AH)

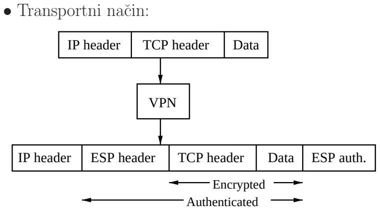
– Podpira MACs: HMAC-MD5-96, HMAC-SHA-1-96.

– Transportni način:



IPec ESP glava

- Encapsulating Security Payload.
- Podprti šifrirni algoritmi: 3-DES, RC5, IDEA, ...
- Transportni način:

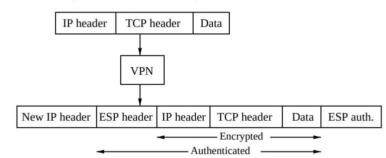


- Opomba: analiza prometa je še vedno možna (ker IP glave niso šifrirane).

ESP v tunelskem načinu

– Požarni zid vključi novo IP glavo
(IP naslov pošiljateljevega požarnega zidu in IP naslov prejemnikovega požarnega zidu).

– Možna je samo zelo omejena analiza prometa.



Secure Sockets Layer (SSL)

- SSL je naredil Netscape.
- TLS (Transport Layer Security) je IETF-ova verzija SSL-a.
- SSL uporabljamo v brskalnikih (npr. Netscape) za zaščito mrežnih transakcij.
- Osnovne komponente SSL/TLS:

  handshake protocol: dopusti strežniku in klientu, da se overita in dogovorita za kriptografske ključe,

  record protocol: uporabljan za šifriranje in overjanje prenašanih podatkov.

Upravljanje z javnimi ključi v SSL/TLS

- Korenski CA ključ je vnaprej inštaliran v brskalnik.
  - Klik na "Security" in nato na "Signers", da najdete seznam ključev korenskih CA v Netscape-u.
- Mrežnim strežnikom certificirajo javne ključe z enim izmed korenskih CA-jev (seveda brezplačno).
  - Verisign-ov certification business za mrežne strežnike `www.verisign.com/server/index.html`

- Klienti (uporabniki) lahko pridobijo svoje certifikate. Večina uporabnikov trenutno nima svojih lastnih certifikatov.
  - Če klienti nimajo svojih certifikatov, potem je overjanje samo enostransko (strežnik se avtenticira klientu).
  - Obiščite varno internetno stran kot npr. webbroker1.tdwaterhouse.ca in kliknite na "padlock" v Netscapu, da si ogledate informacijo o strežnikovem certifikatu.

SSL/TLS handshake protocol

Na voljo so naslednji kriptografski algoritmi:

- MAC: HMAC-SHA-1, HMAC-MD5.
- šifriranje s simetričnimi ključi: IDEA, RC2-40, DES-40, DES, Triple-DES, RC4-40, RC4-128.
- Osnovne sheme za dogovor o ključu so:

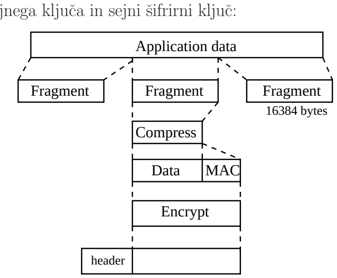
- RSA transport ključev: deljeno skrivnost izbere klient in jo zašifrirana s strežnikovim javnim RSA ključem.
- Fixed Diffie-Hellman: strežnikov Diffie-Hellman-ov javni ključ g^x je v njegovem certifikatu. Klient ima lahko g^y v svojem certifikatu, ali generira enkratno vrednost g^y.
- Ephemeral Diffie-Hellman: Strežnik izbere enkratni Diffie-Hellman-ov javni ključ g^x in ga podpiše s svojim RSA ali DSA ključem za podpise. Klient izbere enkratni g^y in ga podpiše če in samo če ima certifikat.
- MAC in šifrirni ključi so izpeljani iz skupne skrivnosti.

SSL/TLS handshake protokol (2)

1. faza: Določi varnostne zmožnosti.
   - Verzija protokola, način kompresije, kriptografski algoritmi,...
2. faza: Strežnikovo overjanje in izmenjava ključev.
   - Strežnik pošlje svoj certifikate, in (morda še) parametre za izmenjavo ključev.
3. faza: Klientovo overjanje in izmenjava ključeve.
   - Klient pošlje svoj certifikat (če ga ima) in parametre za izmenjavo ključev.
4. faza: Zaključek.

SSL/TLS record protocol

Predpostavimo, da klient in strežnik delita MAC tajnega ključa in sejni šifrirni ključ:



9. poglavje

Identifikacijske sheme

oziroma **sheme za predstavljanje**:

- Uporaba in cilji identifikacijskih shem
- Protokol z izzivom in odgovorom
- Schnorrova identifikacijska shema
- Okomotova identifikacijska shema
- Guillou-Quisquater identifikacijska shema
- Pretvarjanje identifikacijske sheme v shemo za digitalni podpis

Pogosto hočemo dokazati svojo identiteto, npr.:

- **dvig denarja** (na bankomatu rabimo kartico in PIN)
- **nakup/plačilo** (prek telefona, potrebujemo kartico in rok veljave)
- **telefonska kartica** (telefonska številka in PIN)
- **prijava na svojo šifro na računalniku** (uporabniško ime in geslo)

Cilji identifikacijskih shem

- priča Anitine predstavitve Bojanu se ne more kasneje lažno predstaviti za Anito,
- tudi Bojan se ne more po Anitini predstavitvi lažno predstaviti za Anito,
- enostavnost (npr. za pametno/čip kartico)

Anita s svojo predstavitvijo ne izda informacije, ki jo identificira/predstavlja.

Kartica se predstavi sama, nepooblaščeno uporabo (kraja/izguba) pa preprečimo s PIN-om.

Protokol z **izzivom in odgovorom**:

Anita in Bojan delita tajni (skrivni) ključ K, ki ga uporabljata za šifriranje.

1. *Bojan izbere 64-bitni izziv x in ga pošlje Aniti.*
2. *Anita izračuna $y = e_K(x)$ in ga pošlje Bojanu,*
3. *Bojan izračuna $y' = e_K(x)$ in preveri $y = y'$.*

Skoraj vse sheme uporabljajo protokole z izzivom in odgovorom, vendar pa najbolj koristne ne uporabljajo skupnih ključev.

Schnorrova identifikacijska shema

Je ena od najbolj praktičnih shem in potrebuje agencijo TA.

1. *praštevilo p, za katero je DLP nedosegljiv problem (npr. $p \geq 2^{512}$),*
2. *velik delitelj q števila $p-1$ (npr. $q \geq 2^{140}$),*
3. *element $\alpha \in \mathbb{Z}_p^*$ reda q,*
4. *varnostni parameter t, za katerega je $q > 2^t$ (v praksi ponavadi vzamemo $t = 40$),*
5. *TA z algoritmoma za tajno podpisovanje $\mathrm{sig}_{\mathrm{TA}}$ in javno preverjanje $\mathrm{ver}_{\mathrm{TA}}$,*
6. *predpisana varna zgoščevalna funkcija.*

Parametri p, q in α, algoritem za preverjanje $\mathrm{ver}_{\mathrm{TA}}$ in zgoščevalna funkcija so javni.

Agencija TA izda Aniti certifikat:

1. *TA preveri Anitino identiteto po običajni poti (potni list, rojstni list, osebna izkaznica itd.) in izda $\mathrm{ID}(\mathrm{Anita})$, ki vsebuje identifikacijske podatke,*
2. *Anita si izbere zasebno naključno število $a \in [0, \dots, q-1]$, izračuna $v = \alpha^{-a} \bmod p$ in ga izroči agenciji TA.*
3. *Agencija TA izračuna $s = \mathrm{sig}_{\mathrm{TA}}(\mathrm{ID}(\mathrm{Anita}), v)$ ter izroči Aniti potrdilo*
$$C(\mathrm{Anita}) = \big(\mathrm{ID}(\mathrm{Anita}), v, s\big).$$

Bojan preveri Anitino identiteto:

1. *Anita si izbere naključno število $k \in [0, \dots, q-1]$ in izračuna $\gamma = \alpha^k \bmod p$, ki ga pošlje hkrati s svojim potrdilom $C(\mathrm{Anita})$ Bojanu.*
2. *Bojan preveri podpis TA, izbere naključno število $r \in [1, \dots, 2^t]$ in ga pošlje Aniti.*
3. *Anita izračuna $y = k + ar \bmod q$ in ga da Bojanu.*
4. *Bojan preveri, ali je $\gamma \equiv \alpha^y v^r \pmod{p}$.*

Podpis s potrdi Anitin certifikat (tako kot pri uskladitvi ključa).

V drugem delu tajno število a deluje kot nekakšen PIN, saj prepriča Bojana, da je Anita res lastnica certifikata.

Za razliko od PIN-a Anita (oziroma bolj natančno pametna kartica) ne izda števila a, kljub temu, da "dokaže" z odgovorom na izziv z računanjem y-a v 3. koraku, da ga pozna.

Tej tehniki pravimo **dokaz brez razkritja znanja**.

Namen varnostnega parametra t je preprečiti, da bi napadalka, ki bi se hotela predstaviti za Anito, vnaprej uganila Bojanov izziv r (verjetnost $> 2^{40}$).

Če bi napadalka uganila r, bi si lahko za y izbrala poljubno število, izračunala

$$\gamma = \alpha^y v^r \bmod p$$

in ga poslala v 1. koraku Bojanu.
Ko bi prejela Bojanov izziv v drugen koraku, bi mu v 3. koraku dala že izbrani y in identiteta bi bila potrjena v 4. koraku.

Očitno Bojan ne sme uporabiti isti izziv r dvakrat.

Napadalka ne more ponarediti Anitin certifikat:

$$C'(\text{Anita}) = (\text{ID}(\text{Anita}), v', s'), \quad \text{kjer je} \;\; v \neq v',$$

saj bi v tem primeru znala ponarediti podpis s' od $(\text{ID}(\text{Anita}), v')$, ki ga v drugem koraku preveri Bojan.

(Vrednosti v' si ne moremo prosto izbirati, saj bi v tem primeru morali izračunati DLP, da bi dobili ustrezen a'.)

Napadalka ne more uporabiti niti Anitinega pravega certifikata $C(\text{Anita}) = (\text{ID}(\text{Anita}), v, s)$ (lahko bi ga spoznala pri prejšnjem preverjanju identitete), ker ne pozna a, ki ga potrebuje v 3. koraku za računanje y-a.

Izrek 1. *Če napadalka pozna število γ, za katero se zna z verjetnostjo $\varepsilon \geq 1/2^{t-1}$ predstaviti kot Anita, potem zna napadalka izračunati število a v polinomskem času.*

Dokaz: Predpostavimo, da lahko napadalka za ε od 2^t možnih izzivov r izračuna vrednost y, ki jo bo Bojan sprejel. Potem lahko zaradi $2^t\varepsilon \geq 2$ napadalka poišče taka para (y_1, r_1) in (y_2, r_2), da je

$$y_1 \not\equiv y_2 \pmod q \quad \text{in} \quad \gamma \equiv \alpha^{y_1}v^{r_1} \equiv \alpha^{y_2}v^{r_2} \pmod p.$$

Potem je

$$\alpha^{y_1-y_2} \equiv v^{r_2-r_1} \pmod p$$

in zaradi $v = \alpha^{-a}$ velja

$$y_1 - y_2 \equiv a(r_2 - r_1) \pmod q.$$

Končno je $0 < |r_2 - r_1| < 2^t$, število $q > 2^t$ pa je praštevilo, torej $D(r_2 - r_1, q) = 1$ in lahko izračunamo

$$a = (y_1 - y_2)(r_1 - r_2)^{-1} \bmod q.$$

■

Ugotovili smo, da Anita zna potrditi svojo identiteto (*polnost*), vsak drug, ki zna to storiti z neznatno verjetnostjo (z uporabo identifikacijskega protokola) pa bodisi pozna zasebni a bodisi ga zna izračunati v polinomskem času (*uglašenost*).

To pa še ne pomeni, da je Schnorrov protokol varen, saj ima protokol, po katerem bi se Anita identitificirala enostavno tako, da bi odkrila svoj zasebni eksponent a, obe zgornji lastnosti.

Če napadalka ne izračuna nobene informacije o zasebnem eksponentu a medtem, ko je priča polinomskemu številu ponovitev Anitinega identifikacijskega protokola, potem je ta protokol **varen**.

Odprt problem: *Ali je Schnorrova shema varna?*

Naj ima ID(Anita) 512 bitov. Tudi v ima 512 bitov. Podpis s bo imel 320 bitov, če uporabimo DSS. Potem ima $C(\text{Anita})$ 1344 bitov. V prvem koraku mora Anita potencirati po modulu p, vendar pa lahko te vrednosti izračunamo vnaprej, če je potrebno.

$$\text{Anita} \quad \begin{array}{c} \xrightarrow{\quad C,\gamma \quad} \\ \xleftarrow{\quad r \quad} \\ \xrightarrow{\quad y \quad} \end{array} \quad \text{Bojan}$$

Anita pošlje 1344+512=1856 bitov, nato Bojan pošlje 40 bitov in končno Anita pošlje še 140 bitov.

Okomotova identifikacijska shema

Izberimo parametra p, q tako kot v Schnorrovi shemi.

Naj imata elementa $\alpha_1, \alpha_2 \in \mathbb{Z}_p^*$ red q, vrednost $c = \log_{\alpha_1} \alpha_2$ pa naj ne pozna niti Anita.

Kot pri Schnorrovi shemi si agencija TA izbere shemo za digitalni podpis in zgoščevalno funkcijo.

Agencija TA izda Aniti certifikat:

1. *Agencija TA preveri Anitino identiteto in ji izda* ID(Anita),
2. *Anita si izbere zasebni naključni števili* $a_1, a_2 \in [0, \dots, q-1]$, *izračuna* $v = \alpha_1^{-a_1}\alpha_2^{-a_2} \bmod p$ *in ga izroči agenciji TA.*
3. *TA izračuna* $s = \text{sig}_{\text{TA}}(\text{ID}(\text{Anita}), v)$ *ter izroči Aniti potrdilo*
$$C(\text{Anita}) = \big(\text{ID}(\text{Anita}), v, s\big).$$

Bojan preveri Anitino identiteto:

1. *Anita si izbere naključni števili* $k_1, k_2 \in [0,..., q-1]$ *in izračuna* $\gamma = \alpha_1^{k_1}\,\alpha_2^{k_2} \bmod p$, *ki ga pošlje hkrati s svojim potrdilom* $C(\text{Anita})$ *Bojanu.*
2. *Bojan preveri podpis TA, izbere naključno število* $r \in [1, \ldots, 2^t]$ *in ga da Aniti.*
3. *Anita izračuna* $y_i = k_i + a_i r \bmod q$, *za* $i = 1, 2$ *in ju da Bojanu.*
4. *Bojan preveri, ali je* $\gamma \equiv \alpha_1^{y_1}\,\alpha_2^{y_2}\,v^r \pmod{p}$.

Okomotova shema je *polna*, za razliko od Schnorrove shema pa zanjo znamo pokazati, da je *varna*, kakor hitro je diskretni logaritem $\log_{\alpha_1}\alpha_2$ prezahteven.

Predspostavimo, da se je Anita identificirala tako, da je ponovila dani protokol polinomsko število krat in da je napadalka uspela priti do informacije o tajnih eksponentih a_1 in a_2. Pokazali bomo, da v tem primeru znamo izračunati c v polinomskem času, kar je seveda v protislovju s predpostavko.

Izrek 2. *Če napadalka pozna število* γ, *za katero se zna z verjetnostjo* $\varepsilon \geq 1/2^{t-1}$ *predstaviti kot Anita, potem zna napadalka v polinomskem času izračunati taki števili* b_1 *in* b_2, *da je* $v \equiv \alpha_1^{-b_1}\,\alpha_2^{-b_2} \pmod{p}$.

Dokaz: Predpostavimo, da lahko napadalka za ε od 2^t možnih izzivov r izračuna vrednost y, ki jo bo Bojan sprejel. Potem lahko zaradi $2^t\varepsilon \geq 2$ napadalka poišče taka para (y_1, y_2, r) in (z_1, z_2, s), da je $r \neq s$ in

$$\gamma \equiv \alpha_1^{y_1}\,\alpha_2^{y_2}\,v^r \equiv \alpha_1^{z_1}\,\alpha_2^{z_2}\,v^s \pmod{p}.$$

Definirajmo $b_i \equiv (y_i - z_i)(r - s)^{-1} \bmod q$ za $i = 1, 2$ in preverimo

$$v \equiv \alpha_1^{-b_1}\,\alpha_2^{-b_2}\,v^r \pmod{p}.$$

■

Izrek 3. *Če napadalka pozna število* γ, *za katero se zna z verjetnostjo* $\varepsilon \geq 1/2^{t-1}$ *predstaviti kot Anita, potem znata z verjetnostjo* $1 - 1/q$ *Anita in napadalka v polinomskem času izračunati* $\log_{\alpha_1}\alpha_2$.

Dokaz: Iz prejšnjega izreka sledi, da zna napadalka priti do števil b_1 in b_2, za kateri velja:

$$v \equiv \alpha_1^{-b_1}\,\alpha_2^{-b_2} \pmod{p}.$$

Anita izda vrednosti a_1 in a_2 tako, da imamo

$$v \equiv \alpha_1^{-a_1}\,\alpha_2^{-a_2} \pmod{p}$$

in od tod

$$\alpha_1^{a_1 - b_1} \equiv \alpha_2^{b_2 - a_2} \pmod{p}.$$

Če je $(a_1, a_2) \neq (b_1, b_2)$, potem obstaja $(a_2 - b_2)^{-1} \bmod q$ in je

$$c = \log_{\alpha_1}\alpha_2 = (a_1 - b_1)(a_2 - b_2)^{-1} \bmod q.$$

Naj bo sedaj $(a_1, a_2) = (b_1, b_2)$. Pokazali bomo, da se lahko to zgodi le z zelo majhno verjetnostjo $1/q$, kar pomeni, da Anita in napadalka lahko skoraj vedno izračunata c.

Definirajmo množico vseh urejenih parov, ki bi bili lahko Anitini tajni eksponenti:

$$\mathcal{A} = \{(a_1', a_2') \in \mathbb{Z}_q \times \mathbb{Z}_q \mid \alpha_1^{-a_1'}\alpha_2^{-a_2'} \equiv \alpha_1^{-a_1}\alpha_2^{-a_2} \pmod{p}\}.$$

Potem ima množica $\mathcal{A}$ natanko q elementov, saj je

$$\mathcal{A} = \{(a_1 - c\theta, a_2 + \theta) \mid \theta \in \mathbb{Z}_q\}.$$

Po Okomotovemu protokolu si izbere Anita γ, napadalka si izbere r, Anita pa izračuna

$$\gamma \equiv \alpha_1^{y_1}\,\alpha_2^{y_2}\,v^r \pmod{p}$$

iz

$$y_i = k_i + a_i r \bmod q, \quad \text{za} \quad i = 1, 2,$$

kjer je

$$\gamma \equiv \alpha_1^{k_1}\,\alpha_2^{k_2} \pmod{p}$$

in ne izda k_1, k_2 (ne a_1 in a_2).

Ena četverica (γ, r, y_1, y_2) je navidez odvisna od urejenega para (a_1, a_2). Pokažimo, da bi lahko ta četverica bila generirana od poljubnega drugega para $(a_1', a_2') \in \mathcal{A}$, tj. $a_1' = a_1 - c\theta$ in $a_2' = a_2 + \theta$, $\theta \in [0..q-1]$:

$$y_1 = k_1 + a_1 r = k_1 + (a_1' + c\theta)r = (k_1' + rc\theta) + a_1' r,$$

in

$$y_2 = k_2 + a_2 r = k_2 + (a_2' - \theta)r = (k_2' - r\theta) + a_2' r,$$

Torej če bi začeli z (a_1', a_2') in bi si lahko izbrali $k_1' = k_1 + r\,c\,\theta$ in $k_2' = k_2 - r\,\theta$, bi dobili isti γ. ■

Guillou-Quisquaterjeva identifikacijska shema

Ta shema je zasnovana na sistemu RSA.

Agencija TA si izbere dve praštevili p in q ter izračuna $n = pq$. Slednje število je javno, medtem ko sta vrednosti p in q zasebni in izbrani tako, da je problem faktorizacije prezahteven.

TA si izbere še shemo za digitalni podpis, zgoščevalno funkcijo ter 40-bitno praštevilo b, ki bo služilo kot varnostni parameter in šifrirni eksponent.

Izdaja certifikata poteka na naslednji način:

1. *Agencija TA preveri Anitino identiteto in izda* ID(Anita).
2. *Anita si izbere zasebno naključno število* $u \in [0, \dots, n-1]$, *izračuna* $v = u^{-b} \bmod n$ *in ga izroči agenciji TA.*
3. *Agencija TA izračuna* $s = \mathrm{sig}_{\mathrm{TA}}(\mathrm{ID}(\mathrm{Anita}), v)$ *ter izroči Aniti potrdilo*
$$C(\mathrm{Anita}) = \big(\mathrm{ID}(\mathrm{Anita}), v, s\big).$$

GQ-identifikacija (Bojan preveri Anitino identiteto):

1. *Anita si izbere naključno število* $k \in [0, ..., n-1]$ *in izračuna* $\gamma = k^b \bmod n$, *ki ga da hkrati s svojim certifikatom* $C(\mathrm{Anita})$ *Bojanu.*
2. *Bojan preveri podpis agencije TA, izbere naključno število* $r \in [1, \dots, b-1]$ *in ga da Aniti.*
3. *Anita izračuna* $y = ku^r \bmod n$ *in ga da Bojanu.*
4. *Bojan preveri, ali je* $\gamma \equiv y^b v^r \pmod{n}$.

Prepričali se bomo, da je ta shema polna in uglašena, nihče pa ni uspel dokazati, da je tudi varna (tudi če bi privzel, da je krpitosistem RSA varen).

Polnost je očitna
$$v^r y^b \equiv (u^{-b})^r (ku^r)^b \equiv k^b \equiv \gamma \pmod{n},$$
za uglašenost pa privzamemo, da ni mogoče izračunati števila u iz v (le-tega smo dobili iz u z RSA šifriranjem).

Izrek. *Če napadalka pozna število* γ, *za katerega se zna z verjetnostjo* $\varepsilon \geq 1/b$ *predstaviti kot Anita, potem zna napadalka izračunati število* u *v polinomskem času.*

Dokaz: Za nek γ izračuna napadalka take vrednosti y_1, y_2, r_1, r_2, da je $r_1 > r_2$ in
$$\gamma \equiv v^{r_1} y_1^b \equiv v^{r_2} y_2^b \pmod{n}.$$
Potem velja
$$(y_2/y_1)^b \equiv v^{r_1 - r_2} \pmod{n} \quad \text{in} \quad r_1 - r_2 < b,$$
tako da lahko izračunamo $t = (r_1 - r_2)^{-1} \bmod b$ z razširjenim Evklidovim algoritmom. Od tod dobimo tak s, da je $(r_1 - r_2)t = sb + 1$ in
$$(y_2/y_1)^{bt} \equiv v^{(r_1 - r_2)t} \equiv v^{sb+1} \pmod{n} \quad \text{oziroma}$$
$$v \equiv (y_2/y_1)^{bt} v^{-sb} \pmod{n}.$$

Obe strani zgornje kongruence potenciramo na $b^{-1} \bmod \varphi(n)$ ter ju nato invertiramo po modulu n
$$u \equiv (y_1/y_2)^t v^s \pmod{n} \qquad \blacksquare$$

Popularne identifikacijske sheme so še Brickel in McCurleyjeva shema, Feige-Fiat-Shamirjeva shema in Shamirjeva shema s permutiranim jedrom. Zanjo je Shamir dokazal, da je varna s pomočjo metod za dokazovanje brez razkrivanja znanja.

Pretvarjanje identifikacijske sheme v shemo za digitalni podpis

Pokazali bomo še standarden način za pretvarjanje identifikacijske sheme v shemo za digitalni podpis.

Bojana, ki preverja Anitino identiteto, je potrebno zamenjati z javno zgoščevalno funkcijo (sporočilo torej ni zgoščeno pred podpisom, ampak zgoščevanje postane del podpisovanja).

Postopek si poglejmo kar na primeru Schnorrove sheme:

Naj bo p *tako 512-bitno praštevilo, da je DLP v* $\mathbb{Z}_p^*$ *nedosegljiv problem,* q *160-bitni delitelj števila* $p-1$ *in* $\alpha \in \mathbb{Z}_p^*$ *element reda* q. *Naj bo* h *zgoščevalna funkcija z zalogo vrednosti* $\mathbb{Z}_q$, $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{A} = \mathbb{Z}_p^* \times \mathbb{Z}_q$ *in*
$$\mathcal{K} = \{(p, q, \alpha, a, v) \,|\, v \equiv \alpha^{-a} \pmod{p}\}.$$
Vrednosti p, q, α *so javne, vrednost* a *pa zasebna.*

V praksi si običajno za zgoščevalno funkcijo h izberemo SHS, s 160-bitno zalogo vrednosti in z rezultatom, zreduciranim po modulu q (odšteti je potrebno največ en q).

V prehodu iz identifikacijske sheme na shemo za podpisovanje zamenjamo 40-bitni izziv z 160-bitno zgostitvijo sporočila:

Za $K = (p, q, \alpha, a, v)$ in za tajno naključno število $k \in \mathbb{Z}_q^*$, definirajmo

$$\mathrm{sig}_K(x, k) = (\gamma, y),$$

kjer $\gamma = \alpha^k \bmod p$ *in* $y = k + ah(x, \gamma) \bmod q$.

Za $x, \gamma \in \mathbb{Z}_p^*$ in $y \in \mathbb{Z}_q$ definirajmo

$$\mathrm{ver}(x, \gamma, y) = \mathrm{true} \iff \gamma \equiv \alpha^y v^{h(x,\gamma)} \pmod p.$$

Za domačo nalogo poskusite pretvoriti še kakšno izmed opisanih identifikacijskih shem v shemo za podpis.

10. poglavje

Kode za overjanje

(angl. **authentication codes**)

- Uvod
- Računanje verjetnosti prevare
- Kombinatorične ocene
  - pravokotne škatle (angl. orthogonal arrays, OA)
  - konstrukcije in ocene za OA
- Karakterizaciji kod za overjanje
- Ocene entropije
- Incidenčne strukture

Kode za overjanje nam nudijo metode za zagotavljanje *integritete* sporočil, tj. da kljub aktivnemu napadalcu vemo, da

- sporočilo pošilja pričakovana oseba in da
- sporočilo ni spremenjeno.

Shema za overjanje mora biti *brezpogojno varna*, medtem ko smo preučevali sheme za digitalne podpise in MAC-e glede na *računsko varnost*.

Uporaba

Veliki datoteki priredimo potrdilo (hranjeno poleg te datoteke), ki omogoči Bojanu, da preveri, ali je vsebina še vedno nespremenjena (s ključem, ki je hranjen na varnem).

Avtentičnost lahko preveri le tisti, ki mu je sporočilo namenjeno (digitalni podpis pa lahko preveri vsak).

Koda za overjanje *je četverka $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$, za katero velja:*

1. *$\mathcal{S}$ je končna množica vseh začetnih stanj.*
2. *$\mathcal{A}$ je končna množica vseh potrdil.*
3. *$\mathcal{K}$ je končna množica vseh ključev.*
4. *Za vsak $K \in \mathcal{K}$ je dano pravilo za overjanje $e_K : \mathcal{S} \longrightarrow \mathcal{A}$.*

Množica sporočil pa je $\mathcal{M} = \mathcal{S} \times \mathcal{A}$.

Za pošiljanje podpisanega sporočila preko nezavarovanega kanala opravita Anita in Bojan naslednji protokol:

1. *Anita in Bojan skupaj izbereta naključni ključ $K \in \mathcal{K}$ (to storita tajno, tako kot v primeru simetrične kriptografije).*
2. *Anita za sporočilo $s \in \mathcal{S}$ izračuna $a = e_K(s)$ in pošlje Bojanu par (s, a).*
3. *Bojan dobi (s, a), izračuna $a' = e_K(s)$ in preveri, če je $a = a'$.*

Lažna prestavitev (angl. impersonation)
Napadalec vstavi v kanal sporočilo (s, a) v upanju, da ga bo Bojan sprejel za overjenega.

$$\text{napadalec} \xrightarrow{(s,a)} \text{Bojan}$$

Zamenjava
Napadalec opazi na kanalu sporočilo (s, a) in ga zamenja s sporočilom (s', a') v upanju, da ga bo Bojan sprejel za overjenega.

$$\text{Anita} \xrightarrow{(s,a)} \text{napadalec} \xrightarrow{(s',a')} \text{Bojan}$$

Vsakemu od zgornjih napadov priredimo ustrezno **verjetnost prevare** in ju označimo s Pd_0 in Pd_1.

Računanje verjetnosti prevare

Primer: $\mathcal{S} = \mathcal{A} = \mathbb{Z}_3$, $\mathcal{K} = \mathbb{Z}_3 \times \mathbb{Z}_3$ in

$e_{ij}(s) = is + j \bmod 3$ za vsak $(i, j) \in \mathcal{K}$ in $s \in \mathcal{S}$.

Sestavimo $(|\mathcal{K}| \times |\mathcal{S}|)$-dim. matriko M za overjanje, tako da v K-ti vrstici na s-to mesto postavimo element $e_K(s) \in \mathcal{A}$.

Če je v zgornjem primeru $p_{\mathcal{K}}(K)$=1/9 za vsak $K \in \mathcal{K}$, se ni težko prepričati, da je $Pd_0 = Pd_1 = 1/3$.

$$\begin{array}{c} \\ (0,0)\\(0,1)\\(0,2)\\(1,0)\\(1,1)\\(1,2)\\(2,0)\\(2,1)\\(2,2) \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0\\ 1 & 1 & 1\\ 2 & 2 & 2\\ 0 & 1 & 2\\ 1 & 2 & 0\\ 2 & 0 & 1\\ 0 & 2 & 1\\ 1 & 0 & 2\\ 2 & 1 & 0 \end{pmatrix} \end{array}$$

Sedaj pa izračunajmo verjetnosti prevare v splošnem.

Označimo z $\boldsymbol{I(s,a)}$, $s \in \mathcal{S}$, $a \in \mathcal{A}$, verjetnost, da bo Bojan sprejel sporočilo (s,a) za avtentično. Potem je

$$I(s,a) = P(a = e_K(s)) = \sum_{\{H \in \mathcal{K} \,|\, e_H(s)=a\}} p_{\mathcal{K}}(H).$$

Torej izračunamo $I(s,a)$ tako, da v matriki za overjanje izberemo vrstice, ki imajo v stolpcu s vrednost a in nato seštejemo verjetnosti ustreznih ključev.

Napadalec bo izbral tak (s,a), da bo $I(s,a)$ največji:

$$Pd_0 = \max\{I(s,a) \,|\, s \in \mathcal{S},\ a \in \mathcal{A}\}.$$

Medtem ko Pd_0 ni odvisna od porazdelitve $p_{\mathcal{S}}$, pa je Pd_1 lahko. Predpostavimo, da je napadalka na kanalu dobila (s,a) in ga hoče zamenjati s (s',a'), $s' \neq s$.

Za $s, s' \in \mathcal{S}$ in $a, a' \in \mathcal{A}$ je verjetnost, da Bojan ne bo opazil zamenjave, enaka

$$\begin{aligned} I(s',a';s,a) &= P(a' = e_K(s')/a = e_K(s)) \\ &= \frac{P\big((a' = e_K(s')) \cap (a = e_K(s))\big)}{P\big(a = e_K(s)\big)} \\ &= \frac{\displaystyle\sum_{\{H \in \mathcal{K} \,|\, e_H(s)=a,\ e_H(s')=a'\}} p_{\mathcal{K}}(H)}{I(s,a)}. \end{aligned}$$

Napadalec maksimizira svoje možnosti, zato izračuna

$$p_{s,a} = \max\{I(s',a';s,a) \,|\, s' \in \mathcal{S},\ s \neq s' \text{ in } a' \in \mathcal{A}\}.$$

Torej je Pd_1 matematično upanje izrazov $p_{s,a}$ glede na porazdelitev $p_{\mathcal{M}}(s,a)$ in je enako

$$Pd_1 = \sum_{(s,a)\in\mathcal{M}} p_{\mathcal{M}}(s,a)\, p_{s,a}.$$

Verjetnostno porazdelitev za $p_{\mathcal{M}}$ preoblikujemo

$$\begin{aligned} p_{\mathcal{M}}(s,a) &= p_{\mathcal{S}}(s)\, p_{\mathcal{K}}(a/s) \\ &= p_{\mathcal{S}}(s) \sum_{\{K \in \mathcal{K}, | e_K(s)=a\}} p_{\mathcal{K}}(K) \\ &= p_{\mathcal{S}}(s)\, I(s,a). \end{aligned}$$

Za vse $s \in \mathcal{S}$ in $a \in \mathcal{A}$ označimo s $q_{s,a}$ maksimalno vrednost vsote

$$\sum_{\{H \in \mathcal{K} \,|\, e_H(s)=a, e_H(s')=a'\}} p_{\mathcal{K}}(H)$$

glede na vse pare (s',a'), kjer je $s' \in \mathcal{S}\backslash\{s\}$ ter $a' \in \mathcal{A}$.

Od tod dobimo nekoliko bolj priročno formulo za verjetnost prevare

$$Pd_1 = \sum_{(s,a)\in\mathcal{M}} p_{\mathcal{S}}(s)\, q_{s,a}.$$

Kombinatorične ocene

Pri kodah za overjanje si želimo naslednje lastnosti:

- verjetnosti prevare Pd_0 in Pd_1 morata biti dovolj majhni,
- množica začetnih stanj $\mathcal{S}$ mora biti dovolj velika (saj želimo imeti dovolj veliko množico sporočil),
- množica ključev $\mathcal{K}$ naj bo kar se da majhna (saj pošiljamo ključe po varnem kanalu).

Izrek 1. *Za kodo za overjanje $(\mathcal{S},\mathcal{A},\mathcal{K},\mathcal{E})$ velja $Pd_0 \geq 1/|\mathcal{A}|$. Enakost velja, če in samo, če je*

$$\sum_{\{K \in \mathcal{K} \,|\, e_K(s)=a\}} p_{\mathcal{K}}(K) = \frac{1}{|\mathcal{A}|} \quad \text{za vsak } s \in \mathcal{S},\ a \in \mathcal{A}.$$

Dokaz: Za fiksen $s \in \mathcal{S}$ velja:

$$\begin{aligned} \sum_{a\in\mathcal{A}} I(s,a) &= \sum_{a\in\mathcal{A}} \sum_{\{H\in\mathcal{K} \,|\, e_H(s)=a\}} p_{\mathcal{K}}(H) \\ &= \sum_{H\in\mathcal{K}} p_{\mathcal{K}}(H) = 1. \quad \blacksquare \end{aligned}$$

Izrek 2. *Za kodo za overjanje $(\mathcal{S},\mathcal{A},\mathcal{K},\mathcal{E})$ velja $Pd_1 \geq 1/|\mathcal{A}|$. Enakost velja, če in samo, če je*

$$\frac{\displaystyle\sum_{\{H\in\mathcal{K} \,|\, e_H(s)=a,\ e_H(s')=a'\}} p_{\mathcal{K}}(H)}{\displaystyle\sum_{\{H\in\mathcal{K} \,|\, e_H(s)=a\}} p_{\mathcal{K}}(H)} = \frac{1}{|\mathcal{A}|}$$

za vse $s, s' \in \mathcal{S}$, $s' \neq s$ in $a \in \mathcal{A}$.

Dokaz: Za fiksne $s, s' \in \mathcal{S}$, $s' \neq s$ in $a \in \mathcal{A}$, podobno kot v dokazu Izreka 1, izračunamo

$$\sum_{a'\in\mathcal{A}} I(s,a;s',a') = \sum_{a'\in\mathcal{A}} \frac{\displaystyle\sum_{\{H\in\mathcal{K} | e_H(s)=a, e_H(s')=a'\}} p_{\mathcal{K}}(H)}{\displaystyle\sum_{\{H\in\mathcal{K} \,|\, e_H(s)=a\}} p_{\mathcal{K}}(H)} = 1.$$

Od tod pa sledi

$$p_{s,a} = \max_{s \neq s'} I(s', a'; s, a) \geq 1/|\mathcal{A}|.$$

Verjetnost $\quad Pd_1 = \sum_{(s,a)\in\mathcal{M}} p_{\mathcal{M}}(s,a)\, p_{s,a}$

je torej navzdol omejena z

$$\sum_{(s,a)\in\mathcal{M}} \frac{p_{\mathcal{M}}(s,a)}{|\mathcal{A}|} = \frac{1}{|\mathcal{A}|}. \quad \blacksquare$$

Omenimo še dve očitni posledici izrekov 1 in 2.

Posledica 3. *Za kodo za overjanje $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$ velja $Pd_0 = Pd_1 = 1/|\mathcal{A}|$, če in samo, če je*

$$\sum_{\{H\in\mathcal{K}\,|\,e_H(s)=a,\ e_H(s')=a'\}} p_{\mathcal{K}}(H) = \frac{1}{|\mathcal{A}|^2}$$

za vse $s, s' \in \mathcal{S}$, $s' \neq s$ in $a, a' \in \mathcal{A}$. $\blacksquare$

Posledica 4. *Za kodo za overjanje $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$, v kateri so vsi ključi enako verjetni, velja $Pd_0 = Pd_1 = 1/|\mathcal{A}|$, če in samo, če je*

$$|\{H \in \mathcal{K}\,|\, e_H(s) = a,\ e_H(s') = a'\}| = \frac{|\mathcal{K}|}{|\mathcal{A}|^2}$$

za vse $s, s' \in \mathcal{S}$, $s' \neq s$ in $a, a' \in \mathcal{A}$. $\blacksquare$

Pravokotne škatle

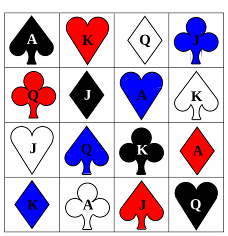
Pravokotna škatla (angl. orthogonal array) $OA(v, s, \lambda)$ je taka $(\lambda v^2 \times s)$-razsežna matrika z v simboli, da se v vsakih dveh stolpcih vsak izmed v^2 možnih parov simbolov pojavi v natanko λ vrsticah.

Te in njim ekvivalentne strukture (npr. transverzalni designi, paroma pravokotni latinski kvadrati, mreže...) so del teorije designa.

Če dva stolpca $OA(v, s, 1)$ uporabimo za koordinate, lahko iz 3. stolpca sestavimo **latinski kvadrat**, tj. $v \times v$-razsežno matriko, v kateri vsi simboli iz $\{1, \ldots, v\}$ nastopajo v vsaki vrstici in vsakem stolpcu.

Primer : $OA(3, 3, 1)$

$$\begin{pmatrix} 0&0&0\\ 1&1&1\\ 2&2&2\\ 0&1&2\\ 1&2&0\\ 2&0&1\\ 0&2&1\\ 1&0&2\\ 2&1&0 \end{pmatrix} \qquad \begin{pmatrix} 0&2&1\\ 2&1&0\\ 1&0&2 \end{pmatrix}$$



Trije paroma ortogonalni latinski kvadrati reda 4, tj. vsak par znak-črka ali črka-barva ali barva-znak se pojavi natanko enkrat.

Izrek 5. *Naj bo $OA(v, s, \lambda)$ pravokotna škatla. Potem obstaja koda za overjanje $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$, kjer je $|\mathcal{S}| = s$, $|\mathcal{A}| = v$, $|\mathcal{K}| = \lambda v^2$ in*

$$Pd_0 = Pd_1 = \frac{1}{v}.$$

Dokaz: Vsako vrstico $OA(v, s, \lambda)$ uporabimo kot pravilo za overjanje z verjetnostjo $1/(\lambda v^2)$:

pravokotna škatla	**koda za overjanje**
`vrstica`	`pravilo za overjanje`
`stolpec`	`začetno stanje`
`simbol`	`potrdilo`

$\blacksquare$

Konstrukcije in ocene za OA

v je število potrdil, s določa število začetnih stanj, λ pa je povezan s številom ključev (λv^2).

Naj bo $Pd_0 \leq \varepsilon$ in $Pd_1 \leq \varepsilon$.
Potem naj za $OA(v, s, \lambda)$ velja

– $v \geq 1/\varepsilon$,
– $s \geq |\mathcal{S}|$ (nekaj stolpcev OA lahko izpustimo),
– λ naj bo čim manjši.

Izrek 6. *Če obstaja $OA(v, s, \lambda)$, potem za $\lambda = 1$ velja $\quad s \leq v + 1$, $\quad$ v splošnem pa*

$$\lambda \geq \frac{s(v-1)+1}{v^2}.$$

Transverzalni design $TD_\lambda(s, v)$ je incidenčna struktura z bloki velikosti s, v katerem so točke razdeljene v s skupin velikosti v tako, da sta poljubni točki v λ blokih, če sta v različnih skupinah, sicer pa ne obstaja noben blok, ki bi ju vseboval.

Dokaz Izreka 6: Število vseh premic, ki sekajo eno premico transverzalnega designa $TD_1(s,v)$ je enako $(v-1)s$ in je kvečjemu enako številu vseh premic brez začetne premice, tj. v^2-1.

V transverzalnem designu $TD_\lambda(s,v)$, $\lambda \neq 1$, pa štejemo na podoben način in nato uporabimo še neenakost med aritmetično in kvadratno sredino (ki jo lahko izpeljemo iz Jensenove neenakosti). ■

Izrek 7. *Za praštevilo p obstaja $OA(p,p,1)$, za $d \in \mathbb{N}\backslash\{1\}$ pa tudi $OA(p,(p^d-1)/(p-1),p^{d-2})$.*

Dokaz: Naj bo $\lambda=1$. Za $\mathcal{K}=\mathbb{Z}_p\times\mathbb{Z}_p$ in $\mathcal{S}=\mathcal{A}=\mathbb{Z}_p$ definiramo $e_{ij}(s)=is+j \bmod p$.

Za $\lambda \neq 1$ pa bomo ekzistenco izpeljali v zadnjem razdelku iz konstrukcije projektivnega prostora $PG(n,d)$. ■

Za DN se prepričaj, da se da vsak $OA(n,n,1)$ razširiti za en stolpec do $OA(n,n+1,1)$.

Karakterizaciji kod za overjanje

Kode za overjanje z najmanjšimi verjetnostmi prevare so prav tiste, ki jih dobimo iz ortogonalnih škatel.

Izrek 8. *Če je $(\mathcal{S},\mathcal{A},\mathcal{K},\mathcal{E})$ taka koda za overjanje, da je $|\mathcal{A}|=v$ in $Pd_0=Pd_1=1/v$, potem je*

$$|\mathcal{K}| \geq v^2.$$

Enačaj velja natanko tedaj, ko obstaja $OA(v,s,1)$

$$\text{in je } |\mathcal{S}|=s \quad \text{ter} \quad p_{\mathcal{P}}(K)=\frac{1}{v^2} \quad \forall K\in\mathcal{K}.$$

Dokaz: Naj bosta $s,s'\in\mathcal{S}$, $s\neq s'$. Za vsak par (a,a') potrdil sledi iz Posledice 3

$$\sum_{\{H\in\mathcal{K}\,|\,e_H(s)=a,\ e_H(s')=a'\}} p_{\mathcal{K}}(H)=\frac{1}{|\mathcal{A}|^2}.$$

Zato je

$$\mathcal{K}_{a,a'}=\{K\in\mathcal{K}\,|\,e_K(s)=a,\ e_K(s')=a'\}\neq\emptyset.$$

Ker je vseh takih množic v^2 in so disjunktne, velja $|\mathcal{K}|\geq v^2$. V primeru enakosti velja $|\mathcal{K}_{a,a'}|=1$ za vsak par potrdil (a,a'), kar pomeni, da v matriki za overjanje v stolpcih s in s' vsak par potrdil pojavi natanko enkrat in imamo $OA(v,s,1)$ za $s=|\mathcal{S}|$.

V primeru enakosti dobimo iz zgornje enakosti tudi

$$p_{\mathcal{K}}(K)=1/v^2 \quad \text{za vsak} \quad K\in\mathcal{K}. \ ■$$

Naslednja karakterizacija je še močnejša in jo predstavljamo brez dokaza.

Izrek 9. *Če je $(\mathcal{S},\mathcal{A},\mathcal{K},\mathcal{E})$ taka koda za overjanje, da je $|\mathcal{S}|=s$, $|\mathcal{A}|=v$ in $Pd_0=Pd_1=1/v$, potem je*

$$|\mathcal{K}|\geq s(v-1)+1.$$

Enačaj velja natanko tedaj, ko obstaja $OA(v,s,\lambda)$ z

$$\lambda=\frac{s(v-1)+1}{v^2} \quad \text{in} \quad p_{\mathcal{P}}(K)=\frac{1}{s(v-1)+1} \quad \forall K\in\mathcal{K}.$$

Ocene entropije

Z Jensenovo neenakostjo lahko izpeljete še naslednji spodnji meji in se prepričate, da ju druga konstrukcija iz Izreka 7 doseže.

Izrek 10. *Če je $(\mathcal{S},\mathcal{A},\mathcal{K},\mathcal{E})$ koda za overjanje, potem velja*

$$\log Pd_0 \geq H(K/M)-H(K)$$

in

$$\log Pd_1 \geq H(K/M^2)-H(K/M).$$

Incidenčne strukture

t-(v,s,λ_t) design je

- zbirka s-elementnih podmnožic (**blokov**)
- množice z v elementi (**točkami**),
- kjer je vsaka t-elementna podmnožica vsebovana v natanko λ_t blokih.

Če je $\lambda_t=1$, imenujemo t-design **Steinerjev sistem** in ga označimo s $S(t,s,v)$.

Naj bo za $i\in\mathbb{N}$, $0\leq i\leq t$, in λ_i število blokov, ki vsebujejo neko i-elementno podmnožico točk S. Potem velja

$$\lambda_i\binom{s-i}{t-i}=\lambda_t\binom{v-i}{t-i}$$

in je število λ_i neodvisno od izbire podmnožice S.

Za $\lambda_0=b$ in $\lambda_1=r$, kadar je $t\geq 2$, velja

$$bs=rv \qquad \text{in} \qquad r(s-1)=\lambda_2(v-1).$$

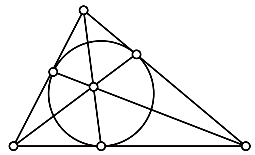
Projektivni prostor $PG(d,q)$ (razsežnosti d nad q) dobimo iz vektorskega prostora $[GF(q)]^{d+1}$, tako da naredimo kvocient po 1-razsežnih podprostorih.

Projektivna ravnina $PG(2,q)$ je incidenčna struktura z 1- in 2-dim. podprostori prostora $[GF(q)]^3$ kot **točkami** in **premicami**, kjer je "$\subset$" incidenčna relacija. To je 2-$(q^2+q+1, q+1, 1)$-design, tj.,

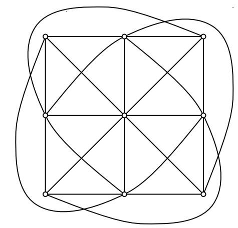
- $v = q^2+q+1$ je število točk (in število premic b),
- vsaka premica ima $k = q+1$ točk (in skozi vsako točko gre $r = q+1$ premic),
- vsak par točk leži na $\lambda = 1$ primicah (in vsaki premici se sekata v natanko eno točki).

Primeri:

1. Projektivno ravnino $PG(2,2)$ imenujemo **Fano ravnina** (7 točk in 7 premic).



2. $PG(2,3)$ lahko skonstruiramo iz 3×3 mreže oziroma afine ravnine $AG(2,3)$.



3. $PG(2,4)$ lahko konstruiramo iz $\mathbb{Z}_{21}$: točke $= \mathbb{Z}_{21}$ in premice $= \{S+x \mid x \in \mathbb{Z}_{21}\}$, kjer je S 5-elementna podmnožica $\{3,6,7,12,14\}$.

$\{0,3,4,9,11\}$ $\{1,4,5,10,12\}$ $\{2,5,6,11,13\}$
$\{3,6,7,12,14\}$ $\{4,7,8,13,15\}$ $\{5,8,9,14,16\}$
$\{6,9,10,15,17\}$ $\{7,10,11,16,18\}$ $\{8,11,12,17,19\}$
$\{9,12,13,18,20\}$ $\{10,13,14,19,0\}$ $\{11,14,15,20,1\}$
$\{12,15,16,0,2\}$ $\{13,16,17,1,3\}$ $\{14,17,18,2,4\}$
$\{15,18,19,3,5\}$ $\{16,19,20,4,6\}$ $\{17,20,0,5,7\}$
$\{18,0,1,6,8\}$ $\{19,1,2,7,9\}$ $\{20,2,3,8,10\}$

Opozorilo: Podobno lahko Fano ravnino konstruiramo iz $\{0,1,3\}$ v $\mathbb{Z}_7$.

Graf $\Gamma = (V,E)$ je sestavljen iz množice **vozlišč** V in družine 2-elementnih podmnožic E, katere elementom pravimo **povezave** (tako definiran graf je brez zank in večkratnih povezav).

Naj bo $V(\Gamma) = \{1,\dots,n\}$. Potem je A $(n \times n)$-razsežna **matrika sosednosti** grafa Γ, če velja

$$A_{i,j} = \begin{cases} 1, & \text{če je } \{i,j\} \in E, \\ 0, & \text{sicer} \end{cases}$$

Število $\theta \in \mathbb{R}$ je lastna vrednost grafa Γ, če za nek vektor $x \in \mathbb{R}^n \backslash \{0\}$ velja

$$Ax = \theta x \quad \text{oziroma} \quad (Ax)_i = \sum_{\{j,i\} \in E} x_j = \theta x_i.$$

Graf je **regularen**, če ima vsako vozlišče enako število sosedov. Podobni pogoji so:

(a) sosednji vozlišči imata natanko λ skupnih sosedov,
(b) nesosednji vozlišči imata natanko μ skupnih sosedov.

Graf je **krepko regularen**, če je regularen ter ima lastnosti (a) in (b).

Za $2 \le s \le v$ je **graf blokov** transverzalnega designa $TD(s,v)$ (dva bloka sta sosednja, če se sekata) krepko regularen graf s parametri $n = v^2$,

$$k = s(v-1), \quad \lambda = (v-2)+(s-1)(s-2), \quad \mu = s(s-1).$$

in lastnimi vrednostmi $s(v-1)^1, v-s^{s(v-1)}, -s^{(v-1)(v-s+1)}$.

Naj bo J $(n \times n)$-razsežna matrika samih enic.

Graf na n vozliščih je krepko-regularen, če in samo, če za njegovo matriko sosednjosti A velja

$$A^2 = kI + \lambda A + \mu(J - I - A),$$

in je $AJ = kJ$ za neka naravna števila k, λ in μ.

(Z matematično indukcijo se lahko hitro prepričamo, da velja $(A^h)_{ij} = \#$sprehodov od i do j dolžine h.)

Od tod sledi, da je ena lastna vrednost k z večkratnostjo 1, preostali vrednosti, ki ju označimo z σ in τ, pa korena naslednje kvadratne enačbe

$$x^2 - (\lambda - \mu)x + (\mu - k) = 0$$

in zato $\lambda - \mu = \sigma + \tau$, $\mu - k = \sigma\tau$.

Število povezav med sosedi in nesosedi nekega vozlišča krepko-regularnega grafa je enako

$$\mu(n-1-k) = k(k-\lambda-1),$$

torej je za povezan graf, tj. $\mu \neq 0$

$$n = \frac{(k-\theta)(k-\tau)}{k+\theta\tau},$$

večkratnosti lastnih vrednosti σ in τ pa sta

$$m_\sigma = \frac{(n-1)\tau + k}{\tau - \sigma} = \frac{(\tau+1)k(k-\tau)}{\mu(\tau-\sigma)}$$

in $m_\tau = n - 1 - m_\sigma$.

Asociativne sheme

Za dani d-terici $\boldsymbol{a}$ in $\boldsymbol{b}$ elementov iz abecede z $n \geq 2$ simboli, imamo glede na ujemanje $d+1$ možnih relacij:

lahko sta enaki, lahko se ujemata na $d-1$ mestih, lahko se ujemata na $d-2$ mestih, ..., ali pa sta različni prav na vseh mestih.

Za dani d-elementni podmnožici A in B množice z n elementi, kjer je $n \geq 2d$, imamo $d+1$ možnih relacij:

lahko sta enaki, lahko se sekata v $d-1$ elementih, lahko se sekata v $d-2$ elementih, ..., ali pa sta disjunktni.

Zgornja primera, skupaj s seznamom relacij, sta primera **asociativnih shem**, ki jih bomo bolj natančno še definirali.

Prva sta konec tridesetih let prejšnjega stoletja vpeljala asociativne sheme **Bose** in **Nair** za potrebe statistike.

Toda **Delsarte** je pokazal, da nam lahko služijo kot povezava med številnimi področji matematike, naprimer teorijo kodiranja in teorijo načrtov. Tu so še
– teorija grup (primitivnost in neprimitivnost),
– linerna algebra (spektralna teorija),
– metrika,
– študij dualnosti in povezava s teorijo karakterjev,
– reprezentacije in ortogonalni polinomi.

Bannai in Ito:

Algebraično kombinatoriko se da opisati kot

“študij kombinatoričnih objektov s pomočjo teorije karakterjev”

ali pa kot

“teorijo grup brez grup”.

Še nekaj zanimivih povezav z asociativnimi shemami:

– teorija vozlov (spin moduli),
– linearno programiranje,
– končne geometrije.

(Simetrična) **asociativna shema** z d razredi in n vozlišči je množica neničelnih, simetričnih, $(n \times n)$-razsežnih 01-matrik $I = A_0, A_1, \ldots, A_d$, za katere velja:

(a) $\sum_{i=0}^{d} A_i = J$, kjer je J matrika samih enic,

(b) za vsaka $i, j \in \{0, 1, \ldots, d\}$ je produkt A_iA_j linearna kombinacija matrik $A_0, \ldots, A_d$.

Asociativno shemo bomo označevali z $\boldsymbol{\mathcal{A}}$ in ji rekli na kratko kar **shema**.

Podprostor $n \times n$ razsežnih matrik nad $\mathbb{R}$, ki je generiran s matrikami $A_0, \ldots, A_d$, je zaradi lastnosti (b) *komutativna algebra*.

Poznamo jo pod imenom **Bose-Mesnerjeva algebra** asociativne sheme $\mathcal{A}$ in jo označimo z $\boldsymbol{\mathcal{M}}$.

Ker je A_i simetrična binarna matrika, je matrika sosednosti nekega (neusmerjenega) grafa Γ_i na n vozliščih.

Če sta vozlišči x in y povezani v grafu Γ_i, bomo to simbolično zapisali z $\boldsymbol{x \, \Gamma_i \, y}$ in rekli, da sta v **i-ti relaciji**.

Iz pogoja (a) sledi, da za poljubni vozlišči x in y obstaja natanko en i, da je $x \, \Gamma_i \, y$, ter da graf Γ_i, $i \neq 0$, nima zank.

Iz pogoja (b) pa sledi, da obstajajo take konstante p_{ij}^h , $i, j, h \in \{0, \ldots, d\}$, da velja

$$A_iA_j = \sum_{h=0}^{d} p_{ij}^h A_h. \tag{2}$$

Pravimo jim **presečna števila** asociativne sheme $\mathcal{A}$. Ker so matrike A_i simetrične, med seboj komutirajo. Zato za vsa presečna števila velja $p_{ij}^h = p_{ji}^h$.

Iz (2) pa razberemo kombinatorični pomen presečnih števil p_{ij}^h, ki zagotovi, da so nenegativna cela števila.

Naj bosta x in y poljubni vozlišči, za kateri je $x \, \Gamma_h \, y$.

$$p_{ij}^h = |\{z\,;\; z \, \Gamma_i \, x \;\text{ in }\; z \, \Gamma_j \, y\}|. \tag{3}$$

Torej je Γ_i regularen graf stopnje $k_i := p_{ii}^0$ in je $p_{ij}^0 = \delta_{ij} k_i$.

Če štejemo trojice vozlišč (x, y, z), kjer je

$$x \, \Gamma_h \, y, \quad z \, \Gamma_i \, x \;\text{ in }\; z \, \Gamma_j \, y,$$

na dva različna načina, dobimo še zvezo $k_h \, p_{ij}^h = k_j \, p_{ih}^j$.

Oglejmo si sedaj nekaj primerov asociativnih shem.

Shema z enim razredom je sestavljena iz identične matrike in matrike sosednosti grafa, v katerem sta sosednji vsaki vozlišči, tj. grafa premera 1 oziroma polnega grafa K_n.

Rekli bomo, da gre za **trivialno shemo**.

Hammingova shema $H(d, n)$

Naj bosta d in n poljubni naravni števili in $\Sigma = \{0, 1, \ldots, n-1\}$.

Vozlišča asociativne sheme $H(d, n)$ so vse d-terice elementov iz Σ. Naj bo $0 \leq i \leq d$.

Vozlišči x in y sta v i-ti relaciji, natanko takrat, ko se razlikujeta v i mestih.

Dobimo asociativno shemo z d razredi in n^d vozlišči.

Shema bilinearnih form $\mathcal{M}_{d\times m}(q)$

(različica iz linearne algebre) Naj bosta d in $m \geq d$ naravni števili ter q potenca nekega praštevila.

Vse $(d\times m)$-razsežne matrike nad GF(q) predstavljajo vozlišča sheme,

vozlišči pa sta v i-ti relaciji, $0 \leq i \leq d$, če je rang njune razlike enak i.

Johnsonova shema $J(n, d)$

Naj bosta n in d poljubni naravni števili, za kateri je $d \leq n$ in X poljubna množica z n elementi.

Vozlišča asociativne sheme $J(n, d)$ so vse d-elementne podmnožice množice X.

Vozlišči x in y sta v i-ti relaciji, $0 \leq i \leq \min\{d, n-d\}$, natanko takrat, ko ima njun presek $d - i$ elementov.

Dobimo asociativno shemo z $\min\{d, n-d\}$ razredi in $\binom{n}{d}$ vozlišči.

q-analogija Johnsonove sheme $J_q(n, d)$ (Grassmanova shema)

Za vozlišča vzamemo vse d-razsežne podprostore n-razsežnega vektorskega prostora V nad GF(q).

Podprostora A in B razsežnosti d sta v i-ti relaciji, $0 \leq i \leq d$, če je $\dim(A \cap B) = d - i$.

Ciklomatične sheme

Naj bo q potenca praštevila in d delitelj števila $q - 1$.

Naj bo C_1 podgrupa multiplikativne grupe obsega GF(q) indeksa d, in naj bodo $C_1, \ldots, C_d$ odseki podgrupe C_1.

Vozlišča sheme so vsi elementi obsega GF(q). Vozlišči x in y sta v i-ti relaciji, ko je $x - y \in C_i$ (in v 0-ti relaciji, ko je $x = y$).

Da bi dobili asociativno shemo, mora biti $-1 \in C_1$, tako da so relacije simetrične, tj. $2 \,|\, d$, če je q lih.

Kako preveriti ali določene matrike sestavljajo asociativno shemo?

Pogoj (b) ni potrebno preverjati neposredno.

Dovolj je, da se prepričamo, da je vrednost izraza na desni strani (3) neodvisna od vozlišč (ne da bi računali p_{ij}^h).

Pomagamo si s *simetrijo*.

Naj bo X množica vozlišč in $\Gamma_1, \ldots, \Gamma_d$ množica grafov za katere velja $V(\Gamma_i) = X$ in katerih matrike sosednosti, skupaj z identično matriko, ustrezajo pogoju (a).

Avtomorfizem te množice grafov je permutacija vozlišč, ki za vsak graf ohranja sosednost.

Matrike sosednosti grafov $\Gamma_1, \ldots, \Gamma_d$, skupaj z identično matriko, tvorijo asociativno shemo, kakor hitro grupa avtomorfizmov deluje za vsak i tranzitivno na parih vozlišč, ki so sosedni v grafu Γ_i

(to je le zadosten pogoj).

Asociativna shema je ***P*-polinomska**, če obstajajo taki polinomi p_i stopnje i, da velja $A_i = p_i(A_1)$ (za neko permutacijo indeksov matrik A_i).

Ekvivalentno je asociativna shema ***P*-polinomska** če obstaja taka permutacija indeksov matrik sosednjosti A_i, da presečna števila zadovoljujejo

trikotniški pogoj: $\forall\ h,\ i,\ j \in \{0, \ldots, d\}$, presečna števila $p_{ij}^h = 0$ (zaporedoma $p_{ij}^h \neq 0$) kakor hitro je eno število izmed $h,\ i,\ j$ $\geq$ (zaporedoma $=$) vsoti preostalih dveh.

Zato P-polinomski asociativni shemi pravimo tudi **metrična**.

Dve bazi, dualnost

Naj bo $E_0, E_1, \ldots, E_d$ množica minimalnih primitivnih idempotentov Bose-Mesnerjeve algebre $\mathcal{M}$. Potem velja

$$E_i \circ E_j = \frac{1}{|X|} \sum_{h=0}^{d} q_{ij}^h E_h, \quad A_i = \sum_{h=0}^{d} p_i(h) E_h$$

$$\text{in} \quad E_i = \frac{1}{|X|} \sum_{h=0}^{d} q_i(h) A_h \quad (0 \le i, j \le d),$$

kjer "$\circ$" označuje produkt po posameznih koordinatah, takoimenovan **Schurov produkt**.

Parametre q_{ij}^h imenujemo **Kreinovi parameteri**, $p_i(0), \ldots, p_i(d)$ so **lastne vrednosti** matrike A_i, in $q_i(0), \ldots, q_i(d)$ so **dualne lastne vrednosti** od E_i.

Za **Kreinove parametre** oziroma **dualne** velja

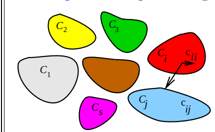
$$\boxed{q_{ij}^h \ge 0.}$$

Asociativna shema $\mathcal{A}$ je ***Q*-polinomska** oziroma **kometrična**
(za na dano permutacijo indeksov idempotentov E_i), če permutacija Kreinovih parameterov q_{ij}^h zadovoljuje trikotniško neenakost.

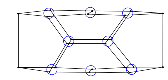
Ekvitabilna particija grafa Γ je taka razdelitev množice $V(\Gamma)$ na **celice** $C_1, C_2, \ldots, C_s$, da velja

(a) vsaka celica C_i inducira *regularen* graf,

(b) povezave med vsakim parom celic C_i, C_j inducirajo *biregularen* bipartiten graf.

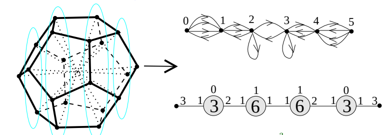


Primer: 1−skeleton dodekaedra

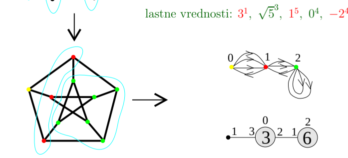


Če je P karakteristična matrika particije π, potem je π *ekvitabilna* če in samo če $AP = PB$ ter če in samo če je span(col(P)) matrike $A(\Gamma)$-*invarianten.*

Particije in lastne vrednosti



lastne vrednosti: 3^1, $\sqrt{5}^3$, 1^5, 0^4, -2^4, $-\sqrt{5}^3$

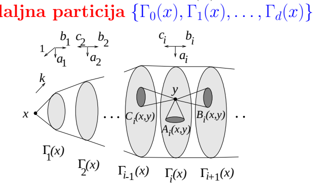


Karakteristična matrika $P = P(\pi)$ particije $\pi = \{C_1, \ldots, C_s\}$ množice z n elementi je $n \times s$ matrika s stolpci, ki jih predstavljajo karakteristični vektorji elementov particije π (tj., ij-ti element matrike P je 1 ali 0 glede na to ali je i v celici C_j ali ne).

> **Izrek.** *Particija π množice vozlišč $V(G)$ s karakteristično matriko P je ekvitabilna natanko tedaj ko obstaja taka $s \times s$ matrika B, da velja $A(G)P = PB$. Če je π ekvitabilna, potem je $B = A(G/\pi)$.*

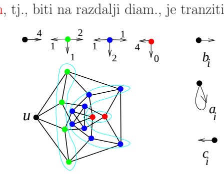
Razdaljna-regularnost:

Γ graf, diameter d, $\forall\, x \in V(\Gamma)$
razdaljna particija $\{\Gamma_0(x), \Gamma_1(x), \ldots, \Gamma_d(x)\}$



je *ekvitabilna*, **zaporedje** $\{b_0, b_1, \ldots, b_{d-1}; c_1, c_2, \ldots, c_d\}$ pa je *neodvisno* od x.

Primer majhnega razdaljno-regularnega grafa:
(antipodalen, tj., biti na razdalji diam., je tranzitivna relacija)



Zgornja presečna števila so enaka za vsako vozlišče u: $\{b_0, b_1, b_2; c_1, c_2, , c_3\} = \{4, 2, 1; 1, 1, 4\}$.

To je točkovni graf posplošenega četverokotnika GQ(2, 2) brez enega paralelnega razreda premic.

11. poglavje

Sheme za deljenje skrivnosti

(angl. **Secret sharing schemes**)

- Uvod
- Stopenjske sheme za deljenje skrivnosti
- Strukture dovoljenj
- Vizualne sheme za deljenje skrivnosti
- Formalne definicije
- Informacijska stopnja
- Ekvivalenca stopenjske sheme in OA

Deljenje skrivnosti

Kombinatorni problem:

n znanstvenikov dela na tajnem projektu, katerega materiali so spravljeni v trezorju z več ključavnicami.

Dostop do materialov je dovoljen, le kadar je prisotna večina znanstvenikov (tj. več kot polovica).

Vsak znanstvenik dobi enako število ključev.

Najmanj koliko ključavnic potrebujemo in koliko ključev mora dobiti vsak znanstvenik?

Rešitev: Naj bo $k = \lfloor (n+2)/2 \rfloor$ in $s = \binom{n}{k}$. Potem imamo s različnih k-elementnih množic znanstvenikov: $G_1, G_2, \dots, G_s$.

Osebe izven skupine G_i nimajo vseh ključev. Naj bo K_i množica, ključev, ki jim manjkajo. $K_i \neq \emptyset$, $i \in [1..s]$.

Skupaj s katerimkoli članom skupine G_i pa imajo vse ključe, torej ima vsaka oseba iz G_i vse ključe iz K_i.

Naj bo $i \neq j$. V množici G_i obstaja oseba, ki ni v G_j. Ta oseba nima nobenega izmed ključev iz K_j, torej je

$$K_i \cap K_j = \emptyset \quad \text{in zato} \quad \#\text{ključev} \geq s.$$

Pokažimo, da je $s = \binom{n}{k}$ ključev, tj, $k_1, \dots, k_s$, dovolj za rešitev tega problema.

Ključe razdelimo tako, da dobijo ključ k_i le osebe iz skupine G_i. Torej dobi vsak znanstvenik $\binom{n-1}{k-1}$ ključev.

Le večinska skupina znanstvenikov ima neprazen presek z vsemi skupinami G_i, tako da lahko le taka skupina odpre trezor. ■

Problem: *V banki morajo trije direktorji odpreti trezor vsak dan, vendar pa ne želijo zaupati kombinacijo nobenemu posamezniku. Zato bi radi imeli sistem, po katerem lahko odpreta trezor poljubna dva med njimi.*

Ta problem lahko rešimo z $(2,3)$-stopenjsko shemo.

Stopenjske sheme za deljenje skrivnosti sta leta 1979 neodvisno odkrila **Blakley in Shamir**.

V splošnem je **(t, n)-stopenjska shema** za deljenje skrivnosti K med n oseb (množica $\mathcal{P}$), $2 \leq t \leq n$, metoda, za katero velja

- poljubnih t oseb lahko izračuna vrednost K,
- nobena skupina s $t-1$ osebami (ali manj) ne more izračunati prav nobene informacije o vrednosti K.

Varnost te sheme mora biti *brezpogojna*, tj. neodvisna od kakšnega računsko zahtevnega problema, kot je na primer faktorizacija v primeru RSA.

Uporaba:

- varno večstrankarsko računanje (*npr. kriptografske volilne sheme*)
- stopenjska kriptografija, večnivojske kontrole (*npr. skupinski podpisi*)
- upravljanje in delitev ključev (*npr. key escrow and keyrecovery schemes*)
- finance in bančništvo (*npr. elektronski denar*)

Revija Time (4. maj, 1992, str. 13)

V Rusiji imajo $(2,3)$-stopenjsko shemo za kontrolo **nuklearnega orožja**:

- predsednik,
- obrambni minister,
- obrambno ministrstvo.

(2,2)-stopenjska shema

1. *Naj bo* $K = k_1 k_2 \dots k_n$, $\;k_i \in \mathbb{Z}_2$ (**skrivnost**).
2. *Delivec izbere naključna števila*
$$a_i \in \mathbb{Z}_2, \quad 1 \leq i \leq n$$
in izračuna $b_i = a_i + k_i \bmod 2, \quad 1 \leq i \leq n$.
3. *Anita in Bojan dobita zaporedoma* **dela** $A = a_1 a_2 \dots a_n$ *in* $B = b_1 b_2 \dots b_n$ *za skrivnost* K.

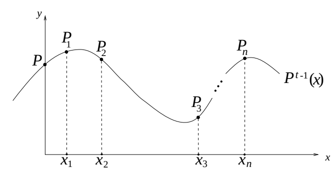
Ne Anita ne Bojan ne moreta vsak zase odkriti nobene informacije o skrivnosti, skupaj pa njuna dela A in B omogočata izračun ključa: $\;K = A + B \bmod 2$.

(t,t)-stopenjska shema

1. *Naj bo $K \in \mathbb{Z}_p$ (**skrivnost**).*
2. *Delivec $D \notin \mathcal{P}$ izbere neodvisno naključna števila $y_1, y_2, \ldots, y_{t-1} \in \mathbb{Z}_m$, $m \geq r+1$, in izračuna*
$$y_t = K - \sum_{i=1}^{t-1} y_i \bmod m.$$
3. *Oseba P_i dobi **del** y_i, $1 \leq i \leq t$.*

Osebe $P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_t$ lahko izračunajo samo $K - y_i$, kar pa jim nič ne pomaga, saj je bilo število y_i naključno izbrano.

Shamir je skonstruiral tudi splošno (t, n)-stopenjsko shemo, za poljubna naravna števila t in n, $2 \leq t \leq n$:



1. *Delivec $D \notin \mathcal{P}$ izbere n različnih elementov $x_1, x_2, \ldots, x_n \in \mathbb{Z}_p^*$, $p \geq n+1$, in da x_i osebi $P_i \in \mathcal{P}$ (vrednosti x_i so javne).*

---

2. *Za delitev ključa K delivec D izbere naključno (neodvisno) $t-1$ elementov $a_1, \ldots, a_{t-1} \in \mathbb{Z}_p$ ter izračuna $y_i = a(x_i)$ in ga da osebi P_i, $1 \leq i \leq n$,*

kjer je $$a(x) = K + \sum_{j=1}^{t-1} a_j x^j \bmod p.$$

Osebe $P_1, P_2, \ldots, P_t$ določijo ključ K iz:
$$y_i = a(x_i) = a_0 + a_1 x_i + \cdots + a_t x_i^t, \quad \text{za } 1 \leq i \leq t$$
oziroma če zapišemo sistem enačb v matrični obliki
$$\begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{t-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{t-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_t & x_t^2 & \ldots & x_t^{t-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{t-1} \end{pmatrix}.$$
Koeficienti tvorijo Vandermondovo matriko z determinanto
$$\det A = \prod_{1 \leq i < j \leq t} (x_i - x_j) \bmod p \neq 0,$$
zato ima sistem enolično rešitev v $\mathbb{Z}_p$.

$t-1$ oseb ima $t-1$ enačb in t neznank.
Za poljuben $a_0 \in \mathbb{Z}_p$ dodamo še enačbo $a_0 = a(0)$ in zopet dobimo sistem z Vandermondovo matriko, katere determinanta je različna od nič.

Torej ne morejo izključiti nobenega ključa K in to je res (t, n)-stopenjska shema za deljenje skrivnosti.

Do enakega zaključka bi lahko prišli tudi z Lagrangovo interpolacijsko formulo za polinome:
$$a(x) = \sum_{i=1}^{t} y_i \prod_{1 \leq j \leq t, j \neq i} \frac{x - x_j}{x_i - x_j}.$$

Pravzaprav potrebujemo samo:
$$K = \sum_{i=1}^{t} y_i \prod_{1 \leq j \leq t, j \neq i} \frac{x_j}{x_j - x_i}.$$
Za $1 \leq i \leq t$ definirajmo
$$b_i = \prod_{1 \leq j \leq t, j \neq i} \frac{x_j}{x_j - x_i}.$$
Potem je ključ linearna kombinacija delov y_i:
$$K = \sum_{i=1}^{t} b_i y_i.$$

Strukture dovoljenj

L.1987 so **Ito, Saito** in **Nishizeki** vpeljali idejo shem za deljenje skrivnosti za poljubno strukturo dovoljenj.

Naj bo $\mathcal{P} = \{P_1, \ldots, P_n\}$ množica oseb, med katere želimo razdeliti skrivnost K. V splošnem si lahko želimo predpisati, katere podmnožice oseb iz $\mathcal{P}$ lahko izračunajo ključ in katere ga ne morejo.

Če so podmnožice iz družine $\Gamma \subseteq 2^{\mathcal{P}}$ natanko tiste množice oseb iz $\mathcal{P}$, ki lahko izračunajo ključ, potem množico Γ imenujemo **struktura dovoljenj**, njene elemente pa **pooblaščene** množice.

Popolna shema za deljenje skrivnosti, ki ustreza strukturi dovoljenj Γ, je metoda za deljenje ključa K na n oseb ($\mathcal{P}$) tako, da velja:

1. vsaka pooblaščena množica $B \subseteq \mathcal{P}$ lahko določi ključ K,
2. vsaka nepooblaščena množica $B \subseteq \mathcal{P}$ ne more odkriti čisto nič o ključu K.

Shamirjeva (t, n)-stopenljska shema je popolna, saj realizira strukturo dovoljenj
$$\{B \subseteq \mathcal{P} \,|\, t \leq |B|\}.$$

Študirali bomo brezpogojno varnost shem za deljenje skrivnosti (nepooblaščene množice imajo na voljo neomejeno računsko moč).

Monotonost: supermnožica pooblaščene množica je tudi pooblaščena.

Zanimale nas bodo samo monotone sheme za deljenje skrivnosti.

$B \in \Gamma$ je **minimalna** pooblaščena množica, če je $A \notin \Gamma$ za vsako podmnožico $A \subset B$.

Γ_0 je množica minimalnih pooblaščenih množic, **baza** za strukturo dovoljenj Γ. Množica

$$\Gamma = \{C \subseteq \mathcal{P} \mid B \subseteq C,\ B \in \Gamma_0\}$$

je potem zaprtje množice Γ_0 in jo bomo označili tudi z $\mathrm{cl}(\Gamma_0)$.

Konstrukcija z monotonim vezjem

Elegantna konstrukcija Benaloha in Leichera nas prepriča, da za vsako (monotono) strukturo dovoljenj obstaja popolna shema za deljenje skrivnosti.

Najprej bomo zgradili vezje, ki "prepozna" strukturo dovoljenj, potem pa iz njegovega opisa še shemo za deljenje skivnosti.

Naj bo $\mathcal{C}$ (booleansko) vezje z vhodi $x_1, \dots, x_n$ (ki ustrezajo osebam $P_1, \dots, P_n$) ter "OR" in "AND" vrati, tj. vrata "NOT" niso dovoljena (vsaka vrata imajo lahko poljubno število vhodov in le en izhod).

Takemu vezju $\mathcal{C}$ bomo rekli **monotono** vezje.

Za $B(x_1, \dots, x_n) := \{P_i \mid x_i = 1\}$ je struktura dovoljenj

$$\Gamma(\mathcal{C}) = \{B(x_1, \dots, x_n) \mid \mathcal{C}(x_1, \dots, x_n) = 1\}$$

monotona (to sledi iz monotonosti vezja $\mathcal{C}$).

Ni se težko prepričati, da obstaja bijektivna korespondenca med monotonimi vezji in boolenskimi formulami z operatojema $\wedge$ ("AND"), $\vee$ ("OR") in brez negacije.

Naj bo Γ_0 baza za strukturo dovoljenj $\Gamma(\mathcal{C})$ in

$$\bigvee_{B \in \Gamma_0} \Big(\bigwedge_{P_i \in B} P_i \Big)$$

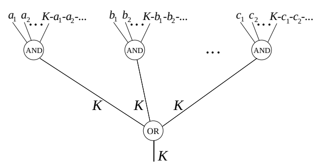
disjunktna normalna forma.

Primer: Za $\Gamma_0 = \{\{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}, \{P_2, P_3\}\}$ dobimo

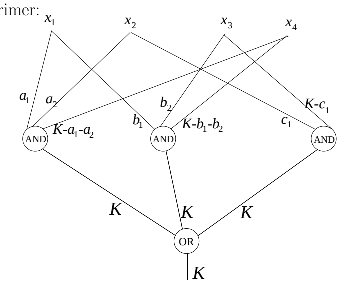
$$(P_1 \wedge P_2 \wedge P_4) \vee (P_1 \wedge P_3 \wedge P_4) \vee (P_2 \wedge P_3).$$

Skupno število vrat v zgornjem vezju je $|\Gamma_0| + 1$.

Sedaj pa naj bo $\mathcal{C}$ poljubno monotono vezje za strukturo dovoljenj Γ (ne nujno zgornje vezje) in $\mathcal{K} = \mathbb{Z}_m$, $m \in \mathbb{N}$. Uporabimo (t,t)-stopenjsko shemo.



Primer:



Drugačen pristop pa nam da konjunktivna normalna forma:

$$(P_1 \vee P_2) \wedge (P_1 \vee P_3) \wedge (P_2 \vee P_3) \wedge (P_2 \vee P_4) \wedge (P_3 \vee P_4)$$

- P_1 dobi a_1 in a_2,
- P_2 dobi a_1, a_3 in a_4,
- P_3 dobi a_2, a_3 in $K - a_1 - a_2 - a_3 - a_4$,
- P_4 dobi a_4 in $K - a_1 - a_2 - a_3 - a_4$.

Izrek 1. *Če je $\mathcal{C}$ monotono vezje, potem nam konstrukcija z monotonim vezjem da popolno shemo za deljenje skrivnosti, ki realizira strukturo dovoljenj $\Gamma(\mathcal{C})$.*

Dokaz: Popolna indukcija po številu vrat vezja $\mathcal{C}$.

Če imamo samo ena vrata, potem je trditev očitna. Sedaj pa naj bo $j > 1$ število vrat.

Zadnja vrata so "OR": $\Gamma(C) = \bigcup_{i=1}^{t} \Gamma(C_i)$.

Zadnja vrata so "AND": $\Gamma(C) = \bigcap_{i=1}^{t} \Gamma(C_i)$. ■

Vizualne sheme za deljenje skrivnosti

sta vpeljala Naor in Shamir leta 1994.

Sliko razdelimo na dele (pravzaprav na prosojnice) z belimi in črnimi pikami/kvadrati), rekonstruiramo pa jo tako, da nekaj prosojnic prekrijemo, tj. naložimo eno na drugo.

Sledimo Stinsonovemu članeku:

Visual Cryptography and Threshold Schemes, Dr. Dobb's Journal, #284, April 1998, pp. 36-43.

Oglejmo si $(2,2)$-stopenjsko shemo (□ $\to 0$, ■ $\to 1$):

- če prekrijemo "**belo**" in "**belo**" dobimo "**belo**" $(0+0=0$, ODLIČNO!),
- če prekrijemo "**belo**" in "**črno**" dobimo "**črno**" $(0+1=1$, ODLIČNO!),
- če prekrijemo "**črno**" in "**črno**" dobimo "**črno**" $(1+1=1$, NE GRE!!!),

Naš vizualni sistem naredi booleanski *ali*, mi pa bi potrebovali booleanski *ekskluzivni ali*.

Naor in Shamir sta se domislila, da nadomestimo vsak kvadrat na sliki z nekaj manjšimi pravokotniki, ki bodo predstavljali dele skrivnosti. Število manjših pravokotnikov označimo z m.

Če je "sivina" črnih kvadratov (v t prekritih delih) temnejša kot sivina belih kvadratov, potem se sliko da prepoznati.

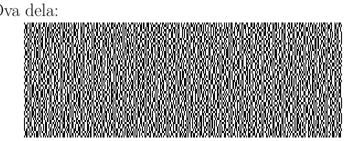
Želimo, da $t-1$ ali manj delov ne more ugotoviti nobene informacije o kvadratu.

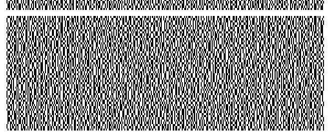
pixel	verjetnost	delitev 1. del	2. del	skupaj
□	p=0.5	◧	◧	◧
	p=0.5	◨	◨	◨
■	p=0.5	◧	◨	■
	p=0.5	◨	◧	■

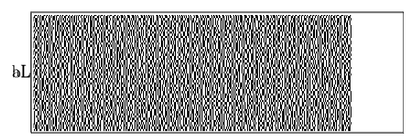
Kvadrat (angl. piksel) P razdelimo na dva pravokotnika (dva dela), glej zgoraj.

Varnost je zagotovljena, **kontrast**, tj. razmerje med črnim in belim, pa je 50%

Dva dela:







Žal poskus, da bi dela sestavil skupaj ni uspel, tako da je potrebno res izpisati prejšnjo prosojnico in naložiti en del čez drugega (vendar pa se lahko zgodi, da vročina pri izpisu deformira prosojnice).

Za opis splošne sheme bomo uporabili $n \times m$-razsežni binarni matriki M_0 in M_1.

Za vsak kvadrat P, naredimo naslednje korake.

1. Generiraj naključno permutacijo π množice $\{1, \ldots, m\}$.
2. Če je P črn kvadrat, potem uporabi permutacijo π nad stolpci matrike M_1, sicer nad stolpci matrike M_0. Dobljeno matriko označimo s T_P.
3. Za $1 \le i \le n$, naj se i-ta vrstica matrike T_P sestoji iz m pravokotnikov kvadrata P v i-tem delu.

Primeri baznih matrik:

1. $(2,2)$-VTS z $\quad m=2$ in $\gamma=1/2$

$$M_0=\begin{pmatrix}1&0\\1&0\end{pmatrix}, \quad \text{in} \quad M_1=\begin{pmatrix}1&0\\0&1\end{pmatrix}.$$

2. $(2,3)$-VTS z $\quad m=3$ in $\gamma=1/3$

$$M_0=\begin{pmatrix}1&0&0\\1&0&0\\1&0&0\end{pmatrix}, \quad \text{in} \quad M_1=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}.$$

3. $(2,4)$-VTS z $\quad m=6$ in $\gamma=1/3$

$$M_0=\begin{pmatrix}1&1&1&0&0&0\\1&1&1&0&0&0\\1&1&1&0&0&0\\1&1&1&0&0&0\end{pmatrix}$$

in

$$M_1=\begin{pmatrix}1&1&1&0&0&0\\1&0&0&1&1&0\\0&1&0&1&0&1\\1&1&1&0&1&1\end{pmatrix}.$$

4. $(3,3)$-VTS z $\quad m=4$ in $\gamma=1/4$

$$M_0=\begin{pmatrix}0&1&1&0\\0&1&0&1\\0&0&1&1\end{pmatrix}$$

in

$$M_1=\begin{pmatrix}0&0&1&1\\0&1&0&1\\1&0&0&1\end{pmatrix}.$$

Sedaj vas gotovo že zanima kakšne lastnosti morata imeti matriki M_0 in M_1. Predno se poglobimo v to, si oglejmo še enkripcijo v primeru $(2,3)$-sheme.

V splošnem imamo $m!$ permutacij elementov množice $\{1,\dots,m\}$. V primeru $m=3$ jih imamo torej 6:

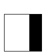
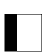
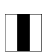
$$\pi_1=(1,2,3),\quad \pi_2=(1,3,2),\quad \pi_3=(2,1,3),$$
$$\pi_4=(2,3,1),\quad \pi_5=(3,1,2),\quad \pi_6=(3,2,1).$$

Naključno permutacijo lahko izberemo na primer z metanjem kocke.

Če hočemo zašifrirati črn kvadrat P in pade 4, potem konstruiramo N_P tako, da vzamemo zaporedoma drugi, tretji in prvi stoplec matrike M_1:

$$N_1=\begin{pmatrix}0&0&1\\1&0&0\\0&1&0\end{pmatrix}.$$

Dobimo:

 za del s_1,  za del s_2,  za del s_3.

Naj bo sta x in y dva binarna vektorja in **wt**(x) število enic v x, booleanski ali nad vektorjema x in y pa označimo z x **or** y.

Binarni $n\times m$ razsežni matriki M_0 in M_1, $m\le n$ sta **bazni matriki** za (t,n)-VTS (angl. visual threshold scheme) z

- m-kratno *ekspanzijo kvadrata* in
- relativnim *kontrastom* γ,

kadar za vsako podmnožico $\{i_1,\dots,i_p\}\subseteq\{1,\dots,n\}$, kjer je $p\le t$, velja

1. za $p=t$ je razlika velikosti nosilcev booleanskega ali vrstic $i_1,\dots,i_p$ matrik M_1 in M_0 vsaj γm,
2. za $p\le t-1$ sta matriki M_0 in M_1 omejeni na vrstice $i_1,\dots,i_p$, enaki do permutacije stolpcev.

Izrek (Naor in Shamir): *Za $n\in\mathbb{N}\backslash\{1\}$ obstaja (n,n)-VTS z $m=2^{n-1}$ in $\gamma=2^{1-m}$.*

Skica dokaz: Matriki M_0 in M_1 naj se zaporedoma sestojita iz vseh binarnih n-teric, ki vsebujejo sodo število enic in liho število enic. ■

Izrek (Blundo, De Santis in Stinson): *V vsaki popolni $(2,n)$-VTS velja*

$$\gamma\le\gamma^*(n):=\frac{\lfloor\frac{n}{2}\rfloor\lfloor\frac{n}{2}\rfloor}{n(n-1)}\,.$$

Ideja: Definirajmo

$$T=\{(i,j,c)\,|\,M_1(i,c)=1,\ M_1(j,c)=1\}.$$

Potem je

$$n(n-1)\gamma m\le|T|\le m\lfloor\frac{n}{2}\rfloor\lfloor\frac{n}{2}\rfloor. \quad ■$$

Konstrukcija $(2, n)$-VTS iz 2-(n, k, λ) designa $\mathcal{D}$

Spomnimo se, da je število blokov designa $\mathcal{D}$ enako

$$nr/k = \lambda(n^2 - n)/(k^2 - k).$$

Naj bo M_1 incidenčna matrika designa $\mathcal{D}$ in M_0 $(n \times b)$-dim. matrika, katere vsako vrstico sestavlja r enic, ki jim sledi $b - r$ ničel.

Naj bo $m = b$.

Poljubna vrstica matrik M_0 in M_1 vsebuje r enic, skalarni produkt poljubnih dveh vrstic matrike M_1 je enak λ. Zato ima nosilec ***booleanskega ali*** dveh vrstic matrike M_1 $2r - \lambda$ elementov, kontrast pa je enak

$$\gamma = \frac{2r - \lambda - r}{b} = \frac{r - \lambda}{b}.$$

Izrek *(Blundo, De Santis in Stinson)***:**
Če obstaja 2-(n, k, λ)-design, potem obstaja $(2, n)$-VTS z ekspanzijo kvadrata $m = b$ in relativnim kontrastom $\gamma = (r - \lambda)/b$.

Posledica:
Če obstaja 2-$(4s - 1, 2s - 1, s - 1)$-design, potem obstaja $(2, 4s - 1)$-VTS z ekspanzijo kvadrata $m = 4s - 1$ in optimalnim relativnim kontrastom $\gamma^*(4s - 1) = s/(4s - 1)$.

Natančen opis postopka za deljenje skrivnosti z vizualno kriptografijo, vse do konkretnega (večjega primera) in Hadamardjevih matrik, si oglejte na

`http://www.cacr.math.uwaterloo.ca/~dstinson/visual.html`

$(n \times n)$-dim. matrika H z elementi ± 1, za katero velja

$$HH^T = nI_n$$

imenujemo **Hadamardjeva matrika** reda n.

Taka matrika obstaja le, če je $n = 1$, $n = 2$ ali pa $4 \mid n$.

Hadamardjeva matrika reda $4s$ je ekvivalentna 2-$(4s - 1, 2s - 1, s - 1)$ designu.

Slavna **Hadamardjeva matrična domneva** iz leta 1893 pravi, da obstaja Hadamardjeva matrika reda $4s$ za vsak $s \in \mathbb{N}$.

Domneva je bila preverjena za vse $s \leq 107$.

$$n = 2: \quad \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad n = 4: \quad \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

$$n = 8: \quad \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 \end{pmatrix}$$

Formalne definicije

Distribucijsko (delilno) pravilo je funkcija

$$f : \mathcal{P} \cup \{D\} \longrightarrow \mathcal{S} \cup \mathcal{K},$$

ki predstavlja eno izmed možnih razdelitev delov iz množice $\mathcal{S}$ ključa $K \in \mathcal{K}$ osebam iz $\mathcal{P}$ (oseba P_i dobi del $f(P_i)$).

Za vsak ključ $K \in \mathcal{K}$ (porazdelitev $p_{\mathcal{K}}$) naj bo $\mathcal{F}_K$ množica distribucijskih pravil, ki ustrezajo ključu K, tj. $\{f \in \mathcal{F} \mid f(D) = K\}$ (porazdelitev $p_{\mathcal{F}_K}$) in

$$\mathcal{F} = \bigcup_{K \in \mathcal{K}} \mathcal{F}_K.$$

Medtem ko so $\mathcal{F}_K$ javne, pa je delilec tisti, ki izbere za ključ $K \in \mathcal{K}$ distribucijsko pravilo $f \in \mathcal{F}_K$ ter razdeli dele.

Za $B \subseteq \mathcal{P}$ naj bo

$$\mathcal{S}(B) = \{f|_B : f \in \mathcal{F}\},$$

kjer je $f|_B : B \longrightarrow \mathcal{S}$ in

$$f|_B(P_i) = f(P_i) \quad \forall P_i \in B,$$

tj. množica vseh možnih distribucij delov oseb iz B.

Verjetnostno porazdelitev na $\mathcal{S}(B)$ označimo s $p_{\mathcal{S}(B)}$.

Naj bo $f_B \in \mathcal{S}(B)$. Potem za vse $f_B \in \mathcal{S}(B)$ in $K \in \mathcal{K}$ izračunamo verjetnostno porazdelitev z

$$p_{\mathcal{S}(B)}(f_B) = \sum_{K\in\mathcal{K}} p_{\mathcal{K}}(K) \sum_{\{f\in\mathcal{F}_K\,:\,f|_B=f_B\}} p_{\mathcal{F}_K}(f)$$

in

$$p_{\mathcal{S}(B)}(f_B/K) = \sum_{\{f\in\mathcal{F}_K\,:\,f|_B=f_B\}} p_{\mathcal{F}_K}(f).$$

Naj bo Γ struktura za deljenje skrivnosti in $\mathcal{F} = \bigcup_{K\in\mathcal{K}} \mathcal{F}_K$ množica distribucijskih pravil.

Potem je $\mathcal{F}$ **popolna shema za deljenje skrivnosti** *za strukturo dovoljenj, če velja*

1. *za vsako pooblaščeno množico oseb $B \subseteq \mathcal{P}$ ter poljubna distribucijska pravila $f \in \mathcal{F}_K$ in $f' \in \mathcal{F}_{K'}$, za katera je $f|_B = f'|_B$ velja $K = K'$*
2. *za vsako nepooblaščeno množico oseb $B \subseteq \mathcal{P}$ in za vsako distribucijo delov $f_B \in \mathcal{S}_B$, $p_{\mathcal{K}}(K/f_B) = p_{\mathcal{K}}(K)$ za vsak $K \in \mathcal{K}$.*

Prva lastnost pravi, da vsaka delitev delov članom poljubne pooblaščene množice B natanko določi vrednost ključa.

Druga lastnost pravi, da je distribucija pogojne verjetnosti na $\mathcal{K}$ pri dani delitvi delov f_B članom nepooblaščene množice B enaka distribuciji verjetnosti na $\mathcal{K}$.

Z drugimi besedami: člani nepooblaščene množice B nimajo nobene informacije o ključu.

Druga lastnost je zelo podobna konceptu popolne verjetnosti (od tu ime).

Verjetnost $p_{\mathcal{K}}(K/f_B)$ lahko izračunamo iz verjetnostne porazdelitve s pomočjo Bayesovega izreka:

$$p_{\mathcal{K}}(K/f_B) = \frac{p_{\mathcal{S}(B)}(f_B/K)\,p_{\mathcal{K}}(K)}{p_{\mathcal{S}(B)}(f_B)}$$

(glej primer 11.5, ki je povezan s primerom konjuktivne normalne forme).

Stopenjske sheme iz OA

Pravokotna škatla

$OA_\lambda(t,k,v)$ je taka $(\lambda v^t \times k)$-razsežna matrika z v simboli, da se v vsakih t stolpcih vsaka k-terica simbolov pojavi natanko λ-krat

(za $t = 2$ dobimo staro definicijo).

Naj bo M pravokotna škatla $OA_1(t, w+1, v)$ (privzeli smo torej $\lambda = 1$) in A množica njenih simbolov.

Skonstruirali bomo (t, w)-stopenjsko shemo, za katero je $\mathcal{S} = \mathcal{K} = A$.

Stolpce matrike M označimo s $\mathcal{P} \cup \{D\}$, njene vrstice pa z v^t elementi množice $\mathcal{F}$. Vsaka vrstica f matrike M ustreza distribucijskemu pravilu, tj.

$$f(X) = M(f, X) \quad \text{za vsak } f \in \mathcal{F} \text{ in } X \in \mathcal{P} \cup \{D\}.$$

Potem je za vsak $K \in \mathcal{K}$:

$$\mathcal{F}_K = \{f \in \mathcal{F} \mid M(f, D) = K\}$$

in zato $|\mathcal{K}| = v^{t-1}$ za vsak $K \in \mathcal{K}$. Torej lahko za $K \in \mathcal{K}$ in $f \in \mathcal{F}$ definiramo

$$p_{\mathcal{F}_K}(f) = \frac{1}{v^{t-1}}.$$

Da bi dokazali, da je $\mathcal{F}$ popolna shema za deljenje skrivnosti, moramo preveriti lastnosti (1) in (2). Prva lastnost sledi iz definicije pravokotne škatle in $\lambda = 1$. Vrednosti katerihkoli t delov določijo vrstico matrike M in s tem natanko določen ključ.

Za drugo lastnost moramo pokazati, da za vsak $K \in \mathcal{K}$ in $f_B \in \mathcal{S}(B)$, kjer je $|B| \le t-1$.

$$\frac{p_{\mathcal{S}(B)}(f_B/K)\,p_{\mathcal{K}}(K)}{p_{\mathcal{S}(B)}(f_B)} = p_{\mathcal{K}}(K)$$

oziroma $p_{\mathcal{S}(B)}(f_B/K) = p_{\mathcal{S}(B)}(f_B)$.

Naj bo $|B| = i \le t-1$. Za vsak $K \in \mathcal{K}$ imamo natanko v^{t-i-1} distribucijskih pravil $f \in \mathcal{F}_K$, za katera je $f|_B = f_B$ (saj je $i+1$ simbolov v določenih $i+1$ stolpcih v natanko v^{t-i-1} vrsticah matrike M). Ker je $|\mathcal{F}_K| = v^{t-1}$, velja

$$p_{\mathcal{S}(B)}(f_B/K) = \frac{1}{v^i} \quad \text{za vsak } f_B \text{ in vsak } K.$$

Sedaj ni več težko izračunati za vsak $K \in \mathcal{K}$

$$p_{\mathcal{S}(B)}(f_B) = \sum_{K\in\mathcal{K}} p_{\mathcal{S}(\mathcal{B})}(f_B/K)\,p_{\mathcal{K}}(K)$$

$$= v^{-i} \sum_{K\in\mathcal{K}} p_{\mathcal{K}}(K) = v^{-i} = p_{\mathcal{S}(B)}(f_B/K). \blacksquare$$

Shamirjeva shema je poseben primer te konstrukcije.

Naj bodo $x_0{=}0, x_1, \ldots, x_w$ različni elementi končnega obsega $GF(q)$.

Če za poljubno t-terico $(a_0, \ldots, a_{t-1}) \in (GF(q))^t$ definiramo

$$M((a_0, \ldots, a_{t-1}), i) = \sum_{j=0}^{t-1} a_j (x_i)^j,$$

dobimo ravno Shamirjevo shemo.

Ekvivalenca stopenjske sheme in OA

Sedaj pa pokažimo še obrat (da lahko iz določene stopenjske sheme skonstruiramo pravokotno škatlo).

Izrek 2. *Naj bo M matrika, katere vrstice in stolpci so označeni zaporedoma z elementi iz $\mathcal{F}$ in elementi iz $\mathcal{P} \cup \{D\}$ ter za katero je $M(f, X) = f(X)$. Potem je M pravokotna škatla $OA_1(t, w+1, v)$ z $v = |\mathcal{S}|$.*

Dokaz tega izreka razbijemo na več korakov.

Iz lastnost (2) sledi naslednji rezultat.

Lema 3. *Naj bo $\mathcal{F}$ množica distribucijskih pravil (t, w)-stopenjske sheme in $B \subseteq \mathcal{P}$, $|B| = t-1$. Za $f \in \mathcal{F}$ in za vsak ključ $K \in \mathcal{K}$ obstaja distribucijsko pravilo $g_B \in \mathcal{F}_K$, za katerega je $g_K|_B = f|_B$.* ■

Lema 4. *Za (t, w)-stopenjsko shemo je $|\mathcal{S}| \geq |\mathcal{K}$.*

Dokaz: Naj bo $P \in \mathcal{P} \backslash \{B\}$, kjer je $B \subseteq \mathcal{P}$ in $|B| = t-1$. Iz lastnosti (1) sledi $g_K(P) = g_{K'}(P)$ za $K \neq K'$, kjer smo g definirali v prejšnji lemi.

Potem je funkcija

$$\theta : \mathcal{K} \longrightarrow \mathcal{S}, \quad \text{s pravilom} \quad \theta(K) = g_K(P)$$

injektivna in trditev sledi. ■

Odslej privzemimo $|\mathcal{S}| = |\mathcal{K}|$, se pravi, da je funkcija θ bijekcija (in lahko privzamemo kar $\mathcal{S} = \mathcal{K}$) iz česar sledi:

Lema 5. *Naj bo $\mathcal{F}$ množica distribucijskih pravil (t, w)-stopenjske sheme z $|\mathcal{S}| = |\mathcal{K}|$.*
Naj bo $B \subseteq \mathcal{P}$ in $|B| = t-1$. Če je $f, g \in \mathcal{F}_K$ za nek ključ K in je $f|_B = g|_B$, potem je $f = g$. ■

Posledica 6. *V poljubnih t stolpcih matrike M se pojavi vsaka t-terica v največ eni vrstici.*

Dokaz: Naj bo $C \subseteq \mathcal{P} \cup \{D\}$, $|C| = t$.

Če je $D \in C$, potem rezultat sledi iz Leme 5.

Sedaj pa naj bo $C \subseteq \mathcal{P}$ in $f|_C = g|_C$.
Ker je $|C| = t$ iz lastnosti (1) sledi $f(D) = g(D)$.
Naj bo $C' = C \cup \{D\} \backslash \{X\}$ za nek $X \in C$.
Iz prvega primera sledi $f = g$. ■

Lema 7. *Če je $1 \leq i \leq t$, potem se v poljubnih i-tih stolpcih matrike M vsaka i-terica elementov pojavi vsaj v eni vrstici.*

Dokaz: Naj bo $C \subseteq \mathcal{P}$, $|C| = i$. Dokazovali bomo z indukcijo na i. Če je $i = 1$, vzemimo $C = \{P\}$. Naj bo $B \subseteq \mathcal{P}$, $|B| = t-1$ in $B \cap C = \emptyset$. Potem uporabimo Lemo 4.

Sedaj pa naj bo $i \leq 2$. Ločimo dva primera glede na to ali je $D \in C$. Če je $C \subseteq \mathcal{P}$. Potem je $P \in C$ in $C' \subseteq \mathcal{P}$, kjer je $|C'| = t - i$ in $C \cap C' = \emptyset$.

Po indukcijski predpostavki je vsaka $(i-1)$-terica v stolpcih iz $C'' = C \backslash \{P\}$. Uporabimo Lemo 4 na $B = C' \cup C''$.

V drugem primeru, ko je $D \in C$ postopamo podobno: naj bo $C' \subset \mathcal{P}$, kjer je $|C'| = t - i$ in $C \cap C' = \emptyset$. Po indukcijski predpostavki se vsaka $(i-1)$-terica pojavi v stolpcih iz $C'' = C \backslash \{D\}$. Končno uporabimo Lemo 2 za $B = C' \cup C''$. ■

Izrek 2 sedaj sledi iz Posledice 6 in Leme 7.

Informacijska mera

Radi bi ocenili učinkovitost dobljenih shem.

Informacijska mera za osebo P_i je

$$\rho_i = \log_2 |\mathcal{K}| / \log_2 |\mathcal{S}(P_i)|,$$

kjer so $\mathcal{S}(P_i)$ možni deli za osebo P_i.

Informacijska mera sheme pa je

$$\rho = \max_{1 \le i \le n} \rho_i.$$

Izrek 8. *Naj bo* **C** *monotono vezje. Potem obstaja popolna shema za deljenje skivnosti, ki realizira strukturo dovoljenj* $\Gamma(\mathbf{C})$ *z informacijsko mero*

$$\rho = \max\{1/r_i \,|\, 1 \le i \le n\},$$

kjer r_i *označuje število vhodnih žic vezja* **C** *z delom* x_i.

Izrek 9. *Za vsako popolno shemo za deljenje skrivnosti, ki realizira strukturo dovoljenj* Γ*, je* $\rho \le 1$.

Če velja enačaj, pravimo, da je shema **idealna**. Brickellova konstrukcija z vektorskim prostorom nam da idealno shemo (to in dokaze zgornjih izrekov izpustimo).

21. poglavje

Teorija kodiranja

- Uvod
- Enostavnejše kode za odpravljanje napak
- Glavni mejniki teorije kodiranja
- Singletonova meja
- Linearne kode
- Odkodiranje linearnih kod

Slovenski uvod:

Sandi Klavžar, O teoriji kodiranja, linearnih kodah in slikah z Marsa, *OMF* **45** (1998), 97-106.

in pa R. Jaminik, Elementi teorije informacije, ...

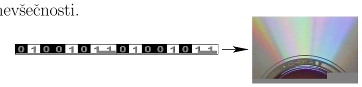
Na začetku so bili računalniški programi dovolj enostavni, tako da so tehnične napake (ponavadi je odpovedala elektronka) hitro postale očitne.

Z razvojem strojne opreme so postajali programi vse obsežnejši in bolj zapleteni, s tem pa je postalo upanje, da bi lahko hitro opazili majhne napake, ki spremenijo delovanje naprave, zanemarljivo in zato tudi resna skrb.

Možnost, da se nam izmuzne kakšna napaka, je vse večja tudi zato, ker so elektronska vezja iz dneva v dan manjša, računalniki pa vse hitrejši.

Tudi če je možnost napake ena sama milijardinka (npr. industrijski standard za trde diske je ena napaka na 10 milijard bitov), se bo 2GHz računalnik, zmotil približno $2\times/s$.

Glede na količino podatkov, ki jih obdelujemo dandanes, je to pravšnji recept za vsakodnevne nevšečnosti.



V času informacijske tehnologije (zgoščenke, GSM telefoni, bančne kartice, internet) se vsi dobro zavedamo pomena hitrega in natančnega prenosa, obdelovanja in hranjenja informacij.

Še tako **popolne naprave** delajo napake, le-te pa lahko hitro spremenijo sicer izredno koristno programsko in strojno opremo v ničvredno ali celo **nevarno orodje**.

Dolgo časa so se ljudje trudili izdelati računalnike in pomnilnike, ki bodo naredili oziroma vsebovali, kar se da malo napak (cene izdelkov pa so se višale).

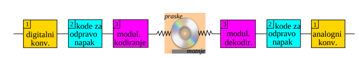
Potem pa so se domislili, da bi raje računalnike same naučili iskati in odpravljati napake. Raziskovalci so našli odgovor v **kodah za odpravljanje napak**.

Koda je skupina simbolov, ki predstavlja informacijo. Kode obstajajo že tisočletja. To so npr.

- hieroglifi,
- grška abeceda,
- rimske številke ali pa
- genetska koda za sestavljanje ribonukleinskih kislin.

Nastale so za različne potrebe:
za zapis govora ali glasbe, Morsejeva abeceda
za prenos informacij, za shranjevanje podatkov itd.

Kode za popravljanje napak
(angl. *error correcting codes*)
nam omogočajo, da popravljamo naključne napake, ki se pojavijo ob motnjah pri prenosu oziroma hranjenju (binarnih) podatkov.



Claude Shannon je postavil teoretične osnove teorije informacij in zanesljivega prenosa digitalnih podatkov kmalu po koncu druge svetovne vojne.

Za povečanje zanesljivosti prenosa in obdelave informacij smo dolgo časa uporabljali **kontrolne bite** (angl. parity-check bits), kot npr. pri številki bančnega čeka, ki pa so služili le za odkrivanje napak.

Richard Hamming je leta 1948 izumil metodo za *popravljanje* ene napake in *odkrivanja* dveh napak.

Ko je vnašal v računalnik programe s pomočjo luknjača kartic in mu je nato računalnik večkrat zavrnil paket kartic zaradi napak, se je zamislil:

“**Če zna računalnik sam odkriti napako, zakaj ne zna najti tudi njenega mesta in jo odpraviti.**”

Enostavnejše kode za odpravljanje napak

Bistvo vseh metod za odpravljanja napak je *dodajanje kontrolnih bitov*. Najenostavnejša koda za odpravljanje napak je zasnovana na **ponavljanju**.

Na primer, če pričakujemo, da pri prenosu ne bo prišlo do več kot ene same napake, potem je dovolj, da ponovimo vsak bit 3× in pri sprejemu uporabimo “**večinsko pravilo**”

Primer: 1101 zakodiramo v 111 111 000 111, če prejmemo 111 011 000 111, popravimo sporočilo v 111 111 000 111 in ga končno še odkodiramo v 1101.

V splošnem lahko odpravimo
n napak z $(2n+1)$-kratnim ponavljanjem
in uporabo večinskega pravila.

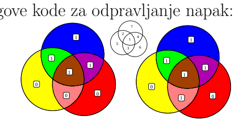
Toda ta metoda je preveč *potratna*.
V času, ko si želimo hitrega prenosa
čim večje količin podatkov,
je to popolnoma *nesprejemljivo*.

Namesto tega si želimo dodati

manjše število kontrolnih bitov,

ki bodo ravno tako ali pa še bolj učinkoviti.

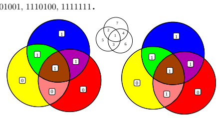
Oglejmo si najpreprostejši primer
Hammingove kode za odpravljanje napak:



Zelo kratko “simfonijo” 1101 spravimo zaporedoma na rjavo (1), zeleno (2), oranžno (3) in vijoličasto (4) polje, preostala polja pa dopolnimo tako, da bo v vsakem krogu vsota števil *soda*.

Dobimo 1101001, kjer zadnja tri mesta predstavljajo zaporedoma rumeno (5), rdeče (6) in modro (7) polje.

Naštejmo vse kodne besede, ki jih dobimo na ta način:
0000000, 0001011, 0010110, 0011101, 0100101, 0101110, 0110011, 0111000, 1000111, 1001100, 1010001, 1011010, 1100010, 1101001, 1110100, 1111111.



Recimo, da je prišlo do ene same napake in da smo prejeli vektor 1111001.

Potem bo prejemnik lahko ugotovil, da je napaka v rumenem in rdečem krogu, ne pa v modrem, kar pomeni, da je potrebno popraviti oranžno (3) polje.

Ni se težko prepričati, da je možno na tak način odpraviti napako na poljubnem bitu (tudi kontrolnem), pri pogoju, da je bila to edina napaka.

S Hammingovo kodo nam je uspelo zmanjšati število kontrolnih bitov z 8 na 3, tj. dobili smo kodo z **informacijsko stopnjo** 4/7 namesto 4/12=1/3.

Zgornjo Hammingovo kodo lahko seveda posplošimo. Običajno to storimo z nekaj linearne algebre (matrike)

Hammingova koda odkrije, da je prišlo do napake pri prenosu tudi kadar je prišlo do dveh napak, saj ne morejo vsi trije krogi vsebovati obeh polj na katerih je prišlo do napake (če pa na dveh mestih zaznamo samo izbris, potem seveda znamo ti mesti tudi popraviti - DN).

Če bi tekst samo podvojili, bi dobili kodo z informacijsko stopnjo 1/2, ki pa lahko odkriva samo samostojne napake, ne more pa jih odpravljati.

V grobem lahko rečemo, da je cilj teorije kodiranja, najti **smiselen kompromis med metodo s kontrolnimi biti in metodo s ponavljanji**. Hammingova koda predstavlja prvi korak v to smer.

Glavni mejniki teorije kodiranja

1947-48: začetki teorije informacij: znamenita izreka o “Source Coding” in pa “Channel Capacity” (C. Shannon)

1949-50: odkritje prvih kod za odpravljanje napak (M. Golay, R. Hamming).

1959-60: odkritje BCH-kod (R. Bose, D. Ray-Chaudhuri, A. Hochquenghem).

1967: Viterby algoritm za odkodiranje konvolucijskih kod.

1993: razvoj turbo kod (C. Berrou, A. Glavieux, P. Titimajshima).

Teorija kodiranja predstavlja varnostno mrežo, svojevrstno matematično zavarovanje pred muhastim materialnim svetom, v katerem živimo.

Tehnologija kod za popravljanje napak je danes tako razširjena kot zgoščenke (CD).

Omogoča nam, da poslušamo priljubljeni
Mozartov ali Madonnin CD
brez kakršnih koli motenj,
četudi nam ga mačka prav pošteno spraska.

Enako tehnologijo uporabljajo za komunikacijo tudi vesoljske ladje in sonde, ki raziskujejo naše osončje.

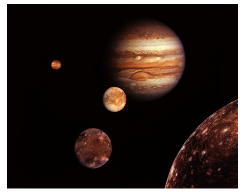
Kode za odpravljanje napak omogočajo, da pridejo na Zemljo, kljub elektromagnetnim motnjam, **kristalno jasni** posnetki oddaljenih planetov, pri tem pa za prenos porabijo

manj energije kot hladilnikova žarnica.

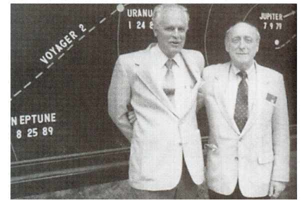
Gre torej za

šepetanje,

ki mora prepotovati več milijard km.



Reed-Solomonove kode doživljajo vrhunec s svojo uporabo na področju hranjenja podatkov (CD, DVD) ter prenašanja podatkov v našem osončju (te dni bo sonda **Cassini** vstopila v Saturnovo orbito in od tam pošiljala slike na Zemljo).



Koda je podmnožica nekega prostora z razdaljo, njeni elementi pa so **kodne besede**. **Razdalja kode** je najmanjša razdalja med različnimi kodnimi besedami.

Običajno razbijemo dano sporočilo na bloke fiksne dolžine (n), ki jih nato povežemo s kodnimi besedami z neko bijektivno korespondenco. V tem primeru rečemo, da gre za **bločne kode dolžine n**.

Najpogosteje si za prostor izberemo množico vseh n-teric s simboli iz neke končne množice F, imenovane tudi **abeceda**:

$$F^n = \{(a_0, a_1, \ldots, a_{n-1}) \mid a_i \in F,\ i = 0, 1, \ldots, n-1\}.$$

Razdalja med dvema n-tericama je število mest, na katerih se razlikujeta.

Pri kodi nas najbolj zanima, koliko napak lahko odpravimo, glede na to koliko kontrolnih bitov smo dodali osnovni informaciji.

(Singletonova meja)
Naj bo C bločna koda dolžine n nad abecedo s q elementi in d njena razdalja. Potem velja

$$|C| \leq q^{n-d+1}.$$

Proof. Naj bo C' koda, ki jo konstruiramo iz kode C tako, da izbrišemo skupino katerihkoli $d-1$ koordinat v vseh kodnih besedah.

Ker je razdalja kode C enaka d, velja $|C| = |C'|$.

Dolžina kode C' pa je $n - d + 1$, zato ima največ q^{n-d+1} kodnih besed, kar smo želeli pokazati. ■

Če so sporočila vse možne k-terice nad abecedo s q elementi ter obstaja bijekcija med sporočili ter kodnimi besedami, je $|C| = q^k$ in pravimo, da gre za **(n, k)-kodo**.

V tem primeru se Singletonova meja prevede v zgornjo mejo za razdaljo kode:

$$d \leq n - k + 1. \tag{4}$$

Naj bo k-terica $\underline{x}$ informacija, ki jo Anita zakodira v n-terico $\underline{y}$ ter pošlje po nekem kanalu.

Bojan prejme n-terico $\underline{r}$, ki ni nujno enaka $\underline{y}$, in jo odkodira po principu "**najbližjega soseda**", tj. najprej poišče kodno besedo $\underline{y}'$, ki je najbližja n-terici $\underline{r}$ in nato izračuna k-terico $\underline{x}'$, ki se zakodira v $\underline{y}'$, v upanju, da je $\underline{y} = \underline{y}'$ in $\underline{x} = \underline{x}'$.

V tem primeru ima koda, ki odpravi t napak, razdaljo $d \geq 2t+1$, saj morajo biti krogle s središčem v kodnih besedah in radijem t disjunktne.

Če torej pride pri prenosu do največ $(d-1)/2$ napak, tj. $d \geq 2t+1$, se nam po principu najbližjega soseda v resnicu posreči popraviti vse napake.

Zato iz neenakosti (4) sledi, da ima taka koda vsaj $2t$ kontrolnih bitov, tj.

$$t \leq \left\lfloor \frac{n-k}{2} \right\rfloor. \tag{5}$$

Trditev: *(n, k)-koda odpravi po principu najbližjega soseda kvečjemu $\lfloor (n-k)/2 \rfloor$ napak.*

Naj bosta n in k pozitivni števili, $k \leq n$.
linearna (n,k)-koda C
je k-razsežni vektorski podprostor v $\mathbb{F}^n$.

Za $k \times n$ razsežno matriko pravimo, da **generira** linearno codo C, če so njene vrstice baza za C.

Za vektorja $\underline{x}, \underline{y} \in \mathbb{F}^n$ je **Hammingova razdalja**, število kordinat, v katerih se $\underline{x}$ in $\underline{y}$ razlikujeta. Označimo jo z $d(\underline{x}, \underline{y})$,

Razdalja linearne (n,k)-kode C je

$$d(C) = \min\{d(\underline{x}, \underline{y}) \,|\, \underline{x}, \underline{y} \in C,\ \underline{x} \neq \underline{y}\}.$$

Oznaka: **(n,k,d)-koda**.

Odkodiranje v praksi

Če bi Bojan primerjal dobljeni vektor $\underline{r}$ z vsako kodno besedo, bi morali opraviti eksponentno število operacij ($|C| = 2^k$) glede na k (to ni polinomski algoritem).

Nadzorna matrika linearne (n,k,d)-kode C je $(n-k) \times n$-dim. binarna matrika H, ki generira ortogonalni komplementa podprostora C.
Le-tega označimo s $C^{\perp}$ in
ga imenujemo **dualna koda** kode C.

Za dani vektor $\underline{r} \in \mathbb{F}^n$ naj bo $(n-k)$-terica $H\underline{r}^T$ njegov **sindrom**.

Izrek: *Naj bo C linearna (n,k)-koda, ki jo generira matrika G, njena nadzorna matrika pa H. Potem za $\underline{x} \in \mathbb{F}^n$ velja*

$$\underline{x} \in C, \text{ tj. } x \text{ je kodna beseda} \iff H\underline{x}^T = 0.$$

Če je $\underline{x} \in C$, $\underline{e} \in \mathbb{F}^n$ in $\underline{r} = \underline{x} + \underline{e}$, potem velja $H\underline{r}^T = H\underline{e}^T$ (tj. sindrom je odvisen samo od napak, ne pa tudi kodne besede).

Teža vektorja $\underline{x} \in (\mathbb{F})^n$, oznaka $w(\underline{x})$, je število njegovih neničelnih koordinat, teža (n,k)-kode C pa je

$$w(C) = \min\{w(\underline{x}) \,|\, \underline{x} \in C \backslash \{0\}\}.$$

Lema: *Če je d razdalja (n,k)-kode C, potem je*

$$d = w(C).$$

Izrek: *Naj bo C linearna (n,k)-koda ter H njena nadzorna matrika.*

Potem ima koda C razdaljo vsaj s natanko tedaj, ko je poljubnih $s-1$ stolpcev matrike H linearno neodvisnih.

Sindromsko odkodiranje

Izračunaj $\underline{s} = H\underline{r}^T$.

Če je $\underline{s}$ ničelni vektor, odkodiraj $\underline{r}$ kot $\underline{r}$.

Sicer pa generiraj vse vektorje napak s težo 1 in njihove sindrome.

Če je za katerega od teh vektorjev $H\underline{e}^T = \underline{s}$, potem odkodiraj $\underline{r}$ kot $\underline{r} - \underline{e}$.

V nasprotnem primeru pa generiraj vse vektorje napak s težo $2, \dots, \lfloor (d-1)/2 \rfloor$ in preverjaj, ali je $H\underline{e}^T = \underline{s}$,...

Po tem postopku odkodiramo dobljeni vektor v največ

$$1 + \binom{n}{1} + \cdots + \binom{n}{\lfloor (d-1)/2 \rfloor}$$

korakih ali pa ugotovimo, da je prišlo do več kot $\lfloor (d-1)/2 \rfloor$ napak.

Medtem ko ta metoda deluje za vsako linearno kodo, pa jo lahko za nakatere kode bistveno pospešimo.

V splošnem pa je odločitvena verzija tega problema NP-poln problem (kadar število napak ni omejeno z $\lfloor (d-1)/2 \rfloor$).

Poseben primer linearnih kod, za katere obstaja hiter algoritem za odkodiranje, so **Goppa kode**.
So lahke za generiranje in imajo veliko število neekvivalentnih kod z istimi parametri.

$$n = 2^m, \quad d = 2t + 1 \ \text{ in } \ k = n - mt\ .$$

Za prakso je McEliece predlagal $m = 10$ in $t = 50$, ki nam da linearno $(1024, 524, 101)$-kodo.

Čistopis je binarna 524-terica, tajnopis pa binarna 1024-terica. Javni ključ je (524×1024)-dim. binarna matrika.

Opis kriptosistema McEliece

Naj bo G matrika, ki generira (n,k,d) Goppa kodo C.

Naj bo S $(k \times k)$-dim. binarna matrika, ki je obrnljiva v $\mathbb{Z}_2$, P $(n \times n)$-dim. permutacijska matrika in naj bo $G' = SGP$, $\mathcal{P} = (\mathbb{Z}_2)^k$, $\mathcal{C} = (\mathbb{Z}_2)^n$,

$$\mathcal{K} = \{(G, S, P, G')\}.$$

Matrika G' je javna, matriki S in P pa tajni (privatni).

Za $K = (G, S, P, G')$ naj bo

$$e_K(\underline{x}, \underline{e}) = \underline{x}G' + \underline{e},$$

kjer je $e \in (\mathbb{Z}_2)^n$ naključni binarni vektor s težo t.

Bojan odšifrira tajnopis $\underline{y} \in (\mathbb{Z}_2)^n$ na naslednji način:

1. izračuna $\underline{y}_1 = \underline{y} P^{-1}$,
2. odkodira $\underline{y}_1$ tako, da najde $\underline{e}_1 = \underline{y}_1 - \underline{x}_1$, kjer je $\underline{x}_1 \in C$,
3. izračuna tak $\underline{x}_0 \in (\mathbb{Z}_2)^k$, da je $\underline{x}_0 G = \underline{x}_1$,
4. izračuna $\underline{x} = \underline{x}_0 S^{-1}$.

Za abecedo si izberimo elemente končnega obsega s q elementi, kjer je q potenca nekega praštevila, oznaka $\mathbb{F} = \mathrm{GF}(q)$. Če je q praštevilo, je to kar praobseg $\mathbb{Z}_q$.

Potem je $\mathbb{F}^n$ z običajnim seštevanjem in množenjem po komponentah vektorski prostor nad $\mathbb{F}$.

Čeprav ne bi bilo nujno, bomo obravnavo poenostavili in v nadaljevanju privzeli, da je dolžina kodnih besed enaka kar $n = q - 1$.

Multiplikativna grupa končnega obsega je $\mathbb{F}$ ciklična.

To pomeni, da obstaja v $\mathbb{F}$ **primitiven** element α, tj. tak element $\alpha \in \mathbb{F}$, da je $\alpha^n = 1$ in $\alpha^i \neq 1$ za vsak $i \in \{1, \ldots, n-1\}$.

Reed-Salomonove kode

Po odkritju Hammingove kode je sledilo obdobje številnih poskusov s kodami za odpravljanje napak. Ko je bila teorija kod stara 10 let sta Irving Reed in Gustave Salomon (takrat zaposlena v Lincolnovem laboratoriju na MIT) zadela v polno.

Namesto ničel in enic sta uporabila skupine bitov, ki jim tudi v računalništvu pravimo kar *besede*.

Ta lastnost je pripomogla k odpravljanju grozdnih napak, tj. napak, pri katerih se pokvari več zaporednih bitov.

Npr. šest zaporednih napak lahko pokvari največ dva bajta. Reed-Salomonova koda (na kratko R-S koda) za odpravljanje dveh napak torej predstavlja že precej dobro zaščito.

Današnje implementacije R-S kod v CD tehnologiji lahko odpravijo grozdne napake dolžine do celo 4000 bitov.

Reed in Solomon sta vpeljala RS(n, k)-kode s pomočjo polinomov. Za **sporočilo**

$$m = (m_0, m_1, \ldots, m_{k-1}) \in \mathbb{F}^k$$

s prirejenim polinomom

$$m(x) = m_0 + m_1 x + \cdots + m_{k-1} x^{k-1}$$

izračunamo vrednosti

$$c_i = m(\alpha^i), \quad i \in \{0, \ldots, n-1\}$$

in iz njih sestavimo **kodno besedo**:

$$c = (c_0, c_1, \ldots, c_{n-1}).$$

Da bo odkodiranje možno, mora seveda veljati $k < n$.

V tem primeru nas dobro znana formula za polinomsko interpolacijo prepriča, da ni preveč pričakovati obstoj odkodirnega algoritema za RS-kode, ki bi opazil morebitne nepravilnosti in jih odpravil.

Bistveno vprašanje pa je, ali je tak algoritem učinkovit.

Prvi postopek za odkodiranje sta predlagala Reed in Solomon. Temelji na reševanju velikega števila sistemov enačb.

Ko sprejmemo kodno besedo

$$c = (c_0, c_1, \ldots, c_{n-1}),$$

lahko sporočilo $m = (m_0, m_1, \ldots, m_{k-1})$ izračunamo iz naslednjega (predoločenega) sistema enačb

$$\begin{aligned}
c_0 &= m_0 + m_1 + m_2 + \cdots + m_{k-1} \\
c_1 &= m_0 + m_1\alpha + m_2\alpha^2 + \cdots + m_{k-1}\alpha^{k-1} \\
c_2 &= m_0 + m_1\alpha^2 + m_2\alpha^4 + \cdots + m_{k-1}\alpha^{2(k-1)} \\
&\vdots \\
c_{n-1} &= m_0 + m_1\alpha^{n-1} + m_2\alpha^{(n-1)\cdot 2} + \cdots + m_{k-1}\alpha^{(n-1)(k-1)}.
\end{aligned} \tag{6}$$

Poglejmo množico poljubnih k enačb, ki ustrezajo k-elementni podmnožici

$$\{a_1, a_2, \ldots, a_k\} \subseteq \{1, \alpha, \ldots, \alpha^{n-1}\}.$$

Njihovi koeficienti tvorijo Vandermondovo matriko z determinanto

$$\begin{vmatrix} 1 & a_1 & a_1^2 & \ldots & a_1^{k-1} \\ 1 & a_2 & a_2^2 & \ldots & a_2^{k-1} \\ & \vdots & & & \vdots \\ 1 & a_k & a_k^2 & \ldots & a_k^{k-1} \end{vmatrix} = \prod_{1 \le i < j \le k} (a_j - a_i).$$

Le-ta je v obsegu $\mathbb{F}$ različna od 0, saj je $a_i \neq a_j$ za vse $i, j \in \{1, \ldots, k\}$, za katere velja $i \neq j$.

Zato ima sistem enolično rešitev v $\mathbb{F}$.

Če se pri prenosu ne bi pojavila napaka, bi lahko z izbiro poljubne k-elementne podmnožice obrnljivih elementov v $\mathbb{F}$ dobili sistem enačb, iz katerega bi lahko določili celotno sporočilo

$$(m_0, \ldots, m_{k-1}).$$

Tako k-elementno podmnožico lahko izberemo na $\binom{n}{k}$ načinov.

Če pa pri prenosu nastanejo napake, nam lahko različni sistemi enačb dajo različne rešitve.

Naslednja lema nam zagotovlja, da se prava rešitev pojavi največkrat, če le število napak ni preveliko.

Lema 2. *Če pride pri prenosu ali branju kodne besede $(c_0, \ldots, c_{n-1})$ RS(n,k)-kode do s napak, se pri reševanju podsistema k-tih enačb iz (6) pojavi napačna rešitev (k-terica) največ*

$$\binom{s+k-1}{k}\text{-krat.}$$

Dokaz: Enačbe sistema (6) ustrezajo k-razsežnim hiperravninam. Zaradi linearne neodvisnosti poljubnih k vektorjev, ki določajo te hiperravnine, se poljubnih k hiperravnin seka v eni točki.

V napačni točki pa se lahko seka največ $s+k-1$ hiperravnin, saj je med njimi lahko največ $k-1$ takih, ki se pri prenosu niso spremenile (k nespremenjenih enačb nam namreč že da pravo rešitev) in največ s takih, ki so se spremenile. ■

Izrek 3. *RS(n,k)-koda je linearna (n,k)-koda.*

Dokaz: Naj bosta c in c' poljubni kodni besedi RS-kode ter $m(x)$ in $m'(x)$ polinoma sporočila, katerima ustrezata ti dve kodni besedi.
Potem za $\lambda, \lambda' \in \mathbb{F}$ in $i \in \{0, 1, \ldots, n-1\}$ velja

$$(\lambda c + \lambda' c')_i = \lambda m(\alpha^i) + \lambda' m'(\alpha^i) = p(\alpha^i),$$

kjer je $p(x) = \lambda m(x) + \lambda' m'(x)$. Od tod sledi, da je $\lambda c + \lambda' c'$ kodna beseda, ki ustreza sporočilu $\lambda m + \lambda' m'$ in je RS-koda linearna. Kodne besede $a_i := (1, \alpha^i, \alpha^{2i}, \ldots, \alpha^{(n-1)i})$ s prirejenimi polinomi x^i, $i \in \{0, 1, \ldots, k-1\}$ so linearno neodvisne, saj jih lahko zložimo v Vandermondovo matriko, katere determinanta je različna od nič, ker so števila $1, \alpha, \alpha^2, \ldots, \alpha^{k-1}$ paroma različna.

Potrebno je le še preveriti, da je poljubna kodna beseda c, ki ustreza nekemu polinomu sporočila $m(x) = \sum_{i=0}^{k-1} m_i x^i$, linearna kombinacija le-teh:

$$c = \left(\sum_{i=0}^{k-1} m_i (\alpha^0)^i, \sum_{i=0}^{k-1} m_i (\alpha^1)^i, \ldots, \sum_{i=0}^{k-1} m_i (\alpha^{n-1})^i \right)$$
$$= \sum_{i=0}^{k-1} m_i \big((\alpha^0)^i, (\alpha^1)^i, \ldots, (\alpha^{n-1})^i \big) = \sum_{i=0}^{k-1} m_i a_i.$$

Torej je RS-koda res k-razsežna. ■

Sedaj pa se prepričajmo, da za RS(n,k)-kode v Singletonovi oceni velja enakost, tj. za dani naravni števili n in k RS(n,k)-kode odpravijo največje možno število napak.

Izrek 4. *RS(n,k)-koda odpravi $\lfloor (n-k)/2 \rfloor$ napak, njena razdalja pa je $n-k+1$.*

Dokaz: Privzemimo, da je pri prenosu RS-kodne besede prišlo do s napak.

Po Lemi 2 dobimo pri reševanju vseh možnih podsistemov k-tih enačb vsako napačno rešitev

$$\text{največ} \binom{s+k-1}{k}\text{-krat, pravo pa} \binom{n-s}{k}\text{-krat.}$$

Slednje število je večje natanko tedaj, ko je

$$n - s > s + k - 1 \quad \text{oziroma} \quad s < (n-k+1)/2.$$

Ker je s celo število, lahko RS-koda na ta način odpravi poljubnih $\lfloor (n-k)/2 \rfloor$ napak.

Torej je njena razdalja vsaj $n-k+1$.]

Iz izreka 3 sledi, da ima RS-koda q^k elementov. Zaradi Singletonove meje (4) oziroma (5) pa je razdalja enaka $n-k+1$. ■

Seveda je ta način za odkodiranje prepočasen, saj zahteva reševanje $\binom{n}{k}$ sistemov enačb velikosti $k \times k$, kar je eksponentna časovna zahtevnost glede na k.

Ciklične kode

Gre za enega najbolj pomembnih razredov linearnih kod. V splošnem je te kode veliko lažje implementirati, zato imajo izjemen praktičen pomen. Iz algebraičnega vidika pa so prav tako izredno zanimive.

Podprostor S n-razsežnega vektorskega prostora je **cikličen podprostor**, če iz

$$(a_1, a_2, \ldots, a_{n-1}, a_n) \in S \ \text{ sledi } \ (a_n, a_1, a_2, \ldots, a_n) \in S.$$

Linearna koda C je **ciklična koda**, če je C cikličen podprostor.

Kodni besedi c, podobno kot prej pri sporočilu, priredimo polinom

$$c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}.$$

Cikličnemu pomiku potem ustreza polinom $c'(x)$, tj.,

$$c_{n-1} + c_0 x + c_1 x^2 + \cdots + c_{n-2} x^{n-1} = x \cdot c(x) - c_{n-1}(x^n - 1).$$

V kolobarju polinomov $R_n = \mathbb{F}^n[x]/(x^n-1)$, kjer gledamo polinome po modulu polinoma $x^n - 1$, dobimo ciklični pomik kar z množenjem s polinomom x.

Zato bomo pogosto enačili kodne besede s polinomi po modulu polinoma $x^n - 1$, tj. delali v kolobarju R_n.

Kolobarji in ideali

Bertrand Russell:
"*Matematiko lahko definiramo kot predmet, pri katerem nikoli ne vemo, o čem govorimo niti nikoli ne vemo, ali je tisto, kar pravimo, resnično.*"

Če v neki množici G z binarno operacijo $\circ$, velja:
(G1) $\forall a, b \in G$ je $a \circ b \in G$,
(G2) $\exists e \in G$, tako da za $\forall g \in G$ velja $e \circ g = g \circ e = g$,
(G3) $\forall g \in G \;\; \exists f \in G$, tako da velja $g \circ f = f \circ g = e$,
(G4) $\forall a, b, c \in G$ velja $(a \circ b) \circ c = a \circ (b \circ c)$,

potem pravimo, da je par $(G, \circ)$ **grupa**.

Če za neko množico $\mathcal{K}$ z binarnima operacijama, ki ju bomo označili s $+$ in $*$, velja

(K1) par $(\mathcal{K}, +)$ je grupa z enoto 0,
(K2) $\forall a, b, c \in \mathcal{K}$ velja $(a * b) * c = a * (b * c)$,
(K3) $\forall a, b \in \mathcal{K}$ velja $a * b = b * a$,
(K4) $\forall a, b, c \in \mathcal{K}$ velja $a * (b + c) = a * b + b * c$.
(K5) $\exists 1 \in \mathcal{K}$, tako da za $\forall a \in \mathcal{K}$ velja $e * g = g$,

potem imenujemo trojico $(\mathcal{K}, +, *)$ **komutativen kolobar z enoto**.

Ker bomo imeli opravka samo s komutativnimi kolobarji z enoto, jim bomo rekli kar kolobarji.

Primeri:

Množica vseh celih števil z običajnim seštevanjem in množenjem $(\mathbb{Z}, +, *)$, ponavadi označena kar z $\mathbb{Z}$.

Množica celih števil po modulu $n \in \mathbb{N}$, ponavadi označena kar z $\mathbb{Z}_n$.

Množica vseh polinomov (spremenljivke x) s koeficienti iz obsega $\mathbb{F}$, in običajnim seštevanjem in množenjem polinomov, običajna oznaka $\mathbb{F}[x]$.

Za neničelen polinom $f(x) \in \mathbb{F}[x]$ lahko definiramo še kolobar polinomov nad $\mathbb{F}$ po modulu $f(x)$,
oznaka $\mathbb{F}[x]/(f(x))$.

Neprazna podmnožica $\mathcal{I}$ kolobarja $(\mathcal{K}, +, *)$ se imenuje ideal kolobarja, če velja
(I1) par $(\mathcal{I}, +)$ je grupa,
(I2) $i * k \in \mathcal{I}$ za $\forall i \in \mathcal{I}$ in za $\forall k \in \mathcal{K}$.

Opišimo preprosto konstrukcijo ideala. Za neničelen element $g \in \mathcal{K}$ vzamemo naslednjo množico

$$\mathcal{I} = \{g * k \mid k \in \mathcal{K}\}.$$

Ni se težko prepričati, da gre za ideal. Pravimo mu ideal generiran z g. Vsakega ideala ne moremo dobiti na ta način, če pa je možno, mu pravimo glavni ideal.

Kolobar v katerem je vsak ideal glavni ideal
(tj. je generiran z enim samim elementom)
imenujemo glavni kolobar.

Izrek: *$\mathbb{F}[x]$ in $\mathbb{F}[x]/(f(x))$ sta glavna kolobarja.*

Izrek: *Neprazna množica S n-razsežnega vektorskega prostora V je cikličen podprostor če in samo če je množica polinomov $\mathcal{I}$, ki ustreza množici S, ideal v kolobarju, ki ustreza prostoru V.*

Izrek: *Naj bo $\mathcal{I} \neq \emptyset$ ideal v $V = \mathbb{F}^n$ in $g(x)$ moničen polinom najmanjše stopnje, ki predstavlja nek razred iz $\mathcal{I}$.*

Potem $[g(x)]$ (ali kar $g(x)$) generira ideal $\mathcal{I}$ in $g(x)$ deli $x^n - 1$.

Izrek: *Obstaja natanko določen moničen polinom najmanjše stopnje, ki generira ideal $\mathcal{I} \neq \emptyset$ n-razsežnega vektorskega prostora V.*

Izrek: *Naj bo $h(x)$ moničen delitelj polinoma $x^n - 1$. Potem je $h(x)$ generator ideala*

$$\mathcal{I} = \{a(x)h(x) \mid a(x) \in \mathcal{K}\}$$

kolobarja $\mathcal{K} = \mathbb{F}[x]/(x^n - 1)$.

Izrek: *Obstaja bijektivna korespondenca med cikličnimi podprostori vektorskega prostora $\mathbb{F}^n$ in moničnimi polinomi $g(x) \in \mathbb{F}[x]$, ki delijo binom $x^n - 1$.*

Izrek 5: *Naj bosta $n, k \in \mathbb{N}$, $n > k$, $g(x)$ moničen polinom stopnje $n - k$, ki deli polinom $x^n - 1$. Potem je*

$$S = \{a(x)g(x) \mid \deg(a) < k\}$$

cikličen podprostor vektorskega prostora R_n in $B = \{g(x), x\, g(x), \ldots, x^{k-1}g(x)\}$ baza podprostora S.

Dokaz: Očitno je S podprostor v R_n. Pokažimo, da je S cikličen, tj. za polinom $p(x) := a(x)g(x) \in S$ je

$$p_1(x) := x\, p(x) \mod (x^n - 1) \quad \text{v podprostoru } S.$$

To je očitno, saj je razlika $p_1(x) - x\, p(x)$ deljiva z $x^n - 1$, ki je deljiv z $g(x)$, polinom $p(x)$ pa je tudi deljiv z $g(x)$. Zato je z $g(x)$ deljiv tudi polinom $p_1(x)$.

Prepričajmo se, da je množica B baza podprostora S. Predpostavimo, da je poljubna linearna kombinacija

$$\sum_{i=0}^{k-1} \lambda_i x^i g(x) = 0.$$

Če obstaja največji indeks j, za katerega je $\lambda_j \neq 0$, potem je koeficient ob x^{n-k+j} enak λ_j, kar pomeni, da mora biti $\lambda_j = 0$. Torej je B linearno neodvisna.

Vektorji iz B napenjajo cel podprostor S, saj za poljuben $p(x) \in S$, velja $p(x) = a(x)g(x)$ za nek $a(x) = a_0 + a_1x + \cdots + a_{k-1}x^{k-1}$, tj.

$$p(x) = a_0g(x) + a_1xg(x) + \cdots + a_{k-1}x^{k-1}g(x)$$

je res linearna kombinacija polinomov iz B. ■

Izrek 6: *Naj bo $\mathbb{F}$ končen obseg s q elementi in $n := q-1$. Naj bo k tako število, da velja $1 \leq k < n$ in $d := n-k+1$ ter α primitiven element v $\mathbb{F}$.*

Koda C_1 naj bo linearna ciklična koda z generatorskim polinomom

$$g(x) = (x-\alpha)(x-\alpha^2)\dots(x-\alpha^{d-1}),$$

koda C_2 pa naj bo RS-koda, pri kateri sporočilu $m \in \mathbb{F}^k$ s prirejenim polinomom

$$m(x) = m_0 + m_1x + \cdots + m_{k-1}x^{k-1}$$

priredimo kodno besedo

$$(m(\alpha), m(\alpha^2), \dots, m(\alpha^n)).$$

Potem kodi C_1 in C_2 sestavljajo iste kodne besede.

Opozorimo, da zgornji izrek ne trdi, da istemu sporočilu v obeh primerih priredimo isto kodno besedo in da izrek velja tudi, če pogoj $n = q-1$ zamenjamo s $q-1 \mid n$.

Dokaz: Ker sta kodi C_1 in C_2 linearni in k-razsežni, je dovolj preveriti, da je beseda $c = (c_0, c_1, \dots, c_{n-1})$, katere prirejeni polinom

$$c(x) = c_0 + c_1x + \cdots + c_{k-1}x^{k-1}$$

je oblike $c(x) = m(x)g(x)$ (tj. beseda iz kode C_1, ki pripada sporočilu m), tudi v kodi C_2, tj. $C_1 \subseteq C_2$.

Torej je treba poiskati tak polinom $f(x)$ stopnje $k-1$, da bo $c_i = f(\alpha^i)$ za $i \in \{0, \dots, n-1\}$.

Naj bo $f(x) = f_0 + f_1x + \cdots + f_{n-1}x^{n-1}$, tako da velja

$$f_j = \frac{c(\alpha^{-j})}{n}, \quad j = 0, \dots, n-1. \tag{7}$$

Polinom $c(x)$ je deljiv s polinomom $g(x)$, zato so $\alpha, \alpha^2, \dots, \alpha^{d-1}$ tudi njegove ničle.

Ker je $d-1 = n-k$, to pomeni, da za $j \in \{n-1, n-2, \dots, k\}$ velja $c(\alpha^{-j}) = c(\alpha^{n-j}) = 0$ in zato tudi $f_j = 0$.

Torej ima polinom $f(x)$ stopnjo največ $k-1$.

Izračunajmo še vrednosti $f(\alpha^i)$, $i \in \{0, \dots, n-1\}$.

Iz (7) sledi

$$f(\alpha^i) = \sum_{j=0}^{n-1} \frac{c(\alpha^{-j})}{n} \cdot (\alpha^i)^j = \frac{1}{n} \sum_{j=0}^{n-1} \Big(\sum_{h=0}^{n-1} c_h \alpha^{-jh} \Big) \alpha^{ij}$$
$$= \frac{1}{n} \sum_{h=0}^{n-1} c_h \cdot \Big(\sum_{j=0}^{n-1} \alpha^{(i-h)j} \Big) = c_i.$$

Pri zadnjem enačaju smo upoštevali, da je izraz v zadnjem oklepaju enak n za $h = i$, sicer pa 0.

To vidimo takole: α je primitiven element, zato je $\alpha^n = 1$ in $\alpha \neq 1$, se pravi, da je α ničla polinoma $(x^n - 1)/(x-1) = 1 + x + x^2 + \cdots + x^{n-1}$;

enako velja tudi za vse potence α, ki so različne od 1. ■

Pravkar opisana transformacija, ki preslika $c(x)$ v $f(x)$, je znana kot **(inverzna) Fourierova transformacija** v končnih obsegih in je diskreten analog Fourierove transformacije v analizi.

Naj bo $\mathbb{F}$ končen obseg s q elementi in $n := q-1$. Naj bo k tako število, da velja $1 \leq k < n$ in $d := n-k+1$.
Naj bo α primitiven element v $\mathbb{F}$ in

$$g(x) = (x-\alpha)(x-\alpha^2)\dots(x-\alpha^{d-1}).$$

Obravnavamo odkodiranje pri RS(n,k)-kodi, generirani s polinomom $g(x)$.

Naj bo $c(x) = a(x)g(x)$ poslana kodna beseda, $r(x)$ pa prejeta beseda. Lahko jo zapišemo v obliki

$$r(x) = c(x) + e(x), \tag{8}$$

kjer je $e(x)$ **polinom napake**.

Če pri prenosu ni prišlo do napake, je $e(x)$ enak nič in je polinom $r(x)$ deljiv z $g(x)$.

Polinom sporočila $a(x)$ dobimo iz $r(x)$ kar z deljenjem s polinomom $g(x)$.

V primeru, da je prišlo do napake, pa bo odkodiranje težje. Najprej bomo odkodiranje prevedli na reševanje sistema linearnih enačb.

Vemo, da obstajata taka polinoma $h(x)$ in $s(x)$, da je

$$r(x) = h(x) \cdot g(x) + s(x),$$
$$\text{in } \deg(s(x)) < \deg(g(x)).$$

$s(x)$ imenujemo **sindrom** prejete besede $r(x)$.
Ker so α, $\alpha^2, \dots, \alpha^{d-1}$ ničle polinoma $g(x)$ in zato tudi polinoma $c(x)$, velja zaradi (8) in zgornje enačbe naslednja zveza:

$$r(\alpha^i) = e(\alpha^i) = s(\alpha^i) \quad \text{za } i = 1, \dots, d-1. \tag{9}$$

Predpostavimo, da pri prenosu ni prišlo do več kot $\ell \leq \lfloor (d-1)/2 \rfloor$ napak, kolikor jih koda največ lahko odpravi.

Naj bodo $a_0, a_1, \dots, a_{\ell-1} \in \{0, \dots, n-1\}$ mesta v kodni besedi, na katerih je prišlo do napake.

Potem lahko polinom $e(x)$ zapišemo v obliki

$$e(x) = \sum_{j=0}^{\ell-1} \lambda_j x^{a_j}.$$

$S_i := s(\alpha^i)$. Eksponenti a_j v potenci α^{a_j} nam povedo položaje napak, zato števila α^{a_j} imenujemo **lokatorji napak**. Vrednosti λ_j pa so **velikosti napak**.

Iz (9) dobimo za $i = \{1, \dots, d-1\}$ sistem enačb

$$S_i = \sum_{j=0}^{\ell-1} \lambda_j (\alpha^i)^{a_j} = \sum_{j=0}^{\ell-1} \lambda_j (\alpha^{a_j})^i, \tag{10}$$

z neznankami λ_j in α^{a_j}, $j = 0, \dots, \ell-1$.

Z uvedbo oznak $X_j = \alpha^{a_j}$, $j = 0, \dots, \ell-1$, sistem zapišemo v naslednji obliki

$$\begin{array}{lcllll} S_1 & = \lambda_0 X_0 & + \lambda_1 X_1 & + \cdots + & \lambda_{\ell-1} X_{\ell-1}, \\ S_2 & = \lambda_0 X_0^2 & + \lambda_1 X_1^2 & + \cdots + & \lambda_{\ell-1} X_{\ell-1}^2, \\ \vdots & & & & \\ S_{d-1} & = \lambda_0 X_0^{d-1} & + \lambda_1 X_1^{d-1} & + \cdots + & \lambda_{\ell-1} X_{\ell-1}^{d-1}. \end{array} \tag{11}$$

Ta sistem $d-1$ enačb z 2ℓ neznankami (λ_j in X_j) se je v preteklosti pojavil pri reševanju različnih problemov.

L. 1975 baron de Prony rešuje interpolacijski problem.

Najprej poiščemo vrednosti X_j, nato pa lahko iz sistema poiščemo še velikosti napak, saj je sistem enačb za λ_i, $i = 0, \dots, \ell-1$, linearen.

Naj bo

$$\sigma(x) = 1 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_\ell x^\ell$$

polinom lokatorjev napake oziroma bolj precizno polinom, ki ima za ničle ravno inverzne vrednosti lokatorjev napak, tj. $\prod_{i=0}^{\ell-1}(1 - X_j x)$. Zato velja:

$$\lambda_j X_j^{\ell+u} \sigma(X_j^{-1}) = 0 \quad \text{za } j = 0, \dots, \ell-1, \tag{12}$$

kjer je u naravno število manjše ali enako ℓ. Seštejmo enačbe (12), upoštevajmo še sistem in dobimo

$$0 = \sum_{j=0}^{\ell-1} \lambda_j X_j^{\ell+u} \Big(1 + \sum_{i=1}^{\ell} \sigma_i X_j^{-i}\Big)$$

$$= S_{u+\ell} + \sum_{i=1}^{\ell} \sigma_i \sum_{j=0}^{\ell-1} \lambda_j X_j^{\ell+u-i} = S_{u+\ell} + \sum_{i=1}^{\ell} \sigma_i S_{\ell+u-i},$$

To je rekurzivna enačba za zaporedje $\{S_i\}$:

$$\sigma_1 S_{u+\ell-1} + \sigma_2 S_{u+\ell-2} + \cdots + \sigma_\ell S_u = -S_{u+\ell}. \tag{13}$$

Ko u teče od $1, \dots, \ell$, dobimo sistem linearnih enačb za σ_i, $i = 1, \dots \ell$, ki ga lahko zapišemo v matrični obliki

$$\begin{bmatrix} S_1 & S_2 & \dots & S_\ell \\ S_2 & S_3 & \dots & S_{\ell+1} \\ \vdots & \vdots & & \vdots \\ S_\ell & S_{\ell+1} & \dots & S_{2\ell-1} \end{bmatrix} \begin{bmatrix} \sigma_\ell \\ \sigma_{\ell-1} \\ \vdots \\ \sigma_1 \end{bmatrix} = - \begin{bmatrix} S_{\ell+1} \\ S_{\ell+2} \\ \vdots \\ S_{2\ell} \end{bmatrix}. \tag{14}$$

Vnaprej ne poznamo ℓ, zato namesto z ℓ računamo z $\lfloor (d-1)/2 \rfloor$.

Rang matrike sistema je v tem primeru enak številu napak. Ko poznamo število napak, lahko iz sistema izračunamo koeficiente polinoma $\sigma(x)$.

Da dobimo lokatorje napak, moramo poiskati ničle $\sigma(x)$ in njihove inverze. Ker smo v končnem obsegu, ničle lahko poiščemo tudi tako, da kar po vrsti preizkušamo elemente obsega (v praksi namreč obseg nima več kot 32 elementov).

Algoritem za odkodiranje Reed-Solomonovih kod, ki smo ga predstavili zgoraj, je bistveno hitrejši od tistega iz drugega razdelka, saj je polinomski.

Rešimo le dva sistema enačb (14) in (11) velikosti $O(d \times d)$, iščemo inverze ℓ elementov, ki so lahko shranjeni tudi v tabeli, ter vrednosti polinoma $\sigma(x)$ v največ n točkah. Skupna zahtevnost algoritma je v najslabšem primeru enaka $O(n^3)$.

Primer: RS$(15, 9)$-koda nad obsegom GF(2^4).
Za primitivni element obsega izberemo ničlo α polinoma $f(x) = x^4 + x + 1$.

Razdalja kode je enaka $d = 15 - 9 + 1 = 7$
(koda popravi do tri napake).

Stopnja generatorskega polinoma $g(x)$ je $n - k = 15 - 9 = 6$. Z uporabo ZechLog tabele izračunamo

$$g(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4)(x-\alpha^5)(x-\alpha^6)$$
$$= \alpha^6 + \alpha^9 x + \alpha^6 x^2 + \alpha^4 x^3 + \alpha^{14} x^4 + \alpha^{10} x^5 + x^6.$$

Kodiranje je množenje s polinomom $g(x)$. Besedo $m = (0, 0, 1, 0, \alpha^{10}, 0, \alpha^2, 0, 0)$ zakodiramo torej kot

$$\begin{aligned} c(x) &= m(x) \cdot g(x) = \alpha^6 x^2 + \alpha^9 x^3 + \alpha^{11} x^4 + \alpha^{11} x^6 \\ &\quad + \alpha^{11} x^8 + \alpha^9 x^9 + \alpha^8 x^{10} + \alpha^{12} x^{11} + \alpha^2 x^{12} \end{aligned}$$

oziroma

$$c = (0, 0, \alpha^6, \alpha^9, \alpha^{11}, 0, \alpha^{11}, 0, \alpha^{11}, \alpha^9, \alpha^8, \alpha^{12}, \alpha^2, 0, 0).$$

Poglejmo sedaj še, kako poteka odkodiranje.
Če je prirejeni polinom $c(x)$ kodne besede c deljiv s polinomom $g(x)$, potem je polinom sporočila $m(x)$ enak $c(x)/g(x)$.

Poskusimo odkodirati še prejeto besedo r s prirejenim polinomom $r(x) = \alpha^6 x^2 + \alpha^9 x^3 + x^4 + x^5 + x^6 + \alpha^{10} x^7 + \alpha^3 x^8 + \alpha^3 x^9 + \alpha^2 x^{12}$. Polinom $r(x)$ ni deljiv z $g(x)$, saj je ostanek enak

$$s(x) = \alpha^5 + \alpha^{10} x + \alpha x^2 + \alpha^{10} x^3 + \alpha^3 x^4 + \alpha^9 x^5.$$

Izračunamo $S_i = s(\alpha^i)$ za $i = 1, \dots, 6$ in dobimo naslednje vrednosti

S_1	S_2	S_3	S_4	S_5	S_6
α^{12}	0	α^3	α^2	α^3	1

Sestavimo matriko iz sistema (14).

$$\begin{bmatrix} \alpha^{12} & 0 & \alpha^3 \\ 0 & \alpha^3 & \alpha^2 \\ \alpha^3 & \alpha^2 & \alpha^3 \end{bmatrix} \tag{15}$$

Matriko (15) enostavno prevedemo na zgornje-trikotno obliko. Od tretje vrstice odštejemo prvo, pomnoženo z α^6, in nato še drugo, pomnoženo z α^{14} (ker ima obseg karakteristiko 2, je odštevanje kar enako seštevanju).

Dobimo matriko ranga 2, kar pomeni, da je pri prenosu kodne besede najverjetneje prišlo do dveh napak. Zato je treba rešiti sistem dveh enačb z dvema neznankama

$$\begin{bmatrix} \alpha^{12} & 0 \\ 0 & \alpha^3 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = -\begin{bmatrix} \alpha^3 \\ \alpha^2 \end{bmatrix}, \tag{16}$$

ki nam da rešitev $\sigma_1 = \alpha^{14}$ in $\sigma_2 = \alpha^6$.

Sedaj poznamo polinom $\sigma(x) = 1 + \alpha^{14}x + \alpha^6x^2$. Z računanjem njegovih vrednosti v vseh elementih obsega GF(2^4) preverimo, da sta njegovi ničli α^4 in α^5.

Njuna inverza α^{11} in α^{10} nam povesta, da sta napaki pri prejeti besedi na 10. in 11. mestu.

Preostane nam le še, da izračunamo velikosti teh napak. V našem primeru bo to najenostavneje kar z reševanjem sistema (11).

Le-ta je predoločen; če nima rešitve, je bila predpostavka, da je prišlo do največ treh napak, napačna.

Velikosti napak izračunamo iz prvih dveh enačb

$$\begin{aligned} \alpha^{12} &= \lambda_0\alpha^{11} + \lambda_1\alpha^{10} \\ 0 &= \lambda_0(\alpha^{11})^2 + \lambda_1(\alpha^{10})^2 \end{aligned} \tag{17}$$

in z deljenjem s polinomom $g(x)$ preverimo, da smo res dobili kodno besedo.

Velikosti napak sta $\lambda_0 = \alpha^{12}$ in $\lambda_1 = \alpha^{14}$.

Polinom poslane kodne besede je potem

$$\begin{aligned} c_1(x) = {} & \alpha^6x^2 + \alpha^9x^3 + x^4 + x^5 + x^6 + \alpha^{10}x^7 \\ & +\alpha^3x^8 + \alpha^3x^9 + \alpha^{14}x^{10} + \alpha^{12}x^{11} + \alpha^2x^{12}. \end{aligned}$$

Ker velja $c_1(x) = g(x)\cdot(x^2+\alpha^7x^4+\alpha^2x^6)$, je polinom sporočila enak $x^2+\alpha^7x^4+\alpha^2x^6$, samo sporočilo pa je enako $(0, 0, 1, 0, \alpha^7, 0, \alpha^2, 0, 0)$.

12. poglavje

Generator psevdonaključnih števil

- Kaj je naključno število
- Algoritmično naključno število
- Uporaba in primeri
- Generator $1/P$
- Algoritem za prevdonaključne bite
- Blum-Blum-Shub generator

M. Kac (*Amer. Scientist* **71** (1981), 405-406)

Kaj je naključno število?

Na to vprašanje ni absolutnega odgovora (teorija informacij, teorija števil, teorija kompleksnosti, fizika).

Za začetek moramo ločiti med naključnim zaporedjem števil in generiranjem naključnega zaporedja.

Najbolj pogost primer naključnega procesa je **metanje kovanca** (idealno). Če ga vržemo n-krat zaporedoma, potem je očitno, da lahko dobimo vsakega izmed 2^n zaporedij grbov ali cifr, tj. da ima vsako od 2^n zaporedij grbov ali cifr enako verjetnost.

Mi se bomo ukvarjali s psevdonaključnim zaporedjem števil, tj. zaporedjem, ki je "videti naključno" oziroma demonstrira *neurejenost/kaos*.

Knuth (*The Art of Computer Programming*, 2nd ed., Addison-Wesley, Reading (1981), 689 pp.) je predstavil številne **statistične teste**, ki merijo neurejenost.

V zaporedju se pojavi z enako frekvenco vsako podzaporedje dolžine $1, 2, \ldots$ Potem pa so tu še serijski testi, poker test, avtokorelacijski testi itd.

Chaitin in Kolmogorov pa pravita, da (dolgo) končno zaporedje bitov, ki se ga dobiti iz programa, ki je precej krajši kot dano zaporedje, ni naključno.

1. Set i equal to 1.
2. Print "1".
3. If $i = n$, then stop.
4. Add 1 to i.
5. Go back to Step 2.

Ne glede na to, ali je število n veliko, ima ta program le fiksno število več bitov, kot jih je v binarni reprezentaciji števil i in n, ki ne presega $2\log_2 n$ (na binarnem računalniku).

Če pa je zaporedje dovolj neurejeno, potem tudi program, ki ga izpiše, ne bo dosti krajši.

Običajno preštevanje pa nam zagotavlja, da bo veliko manj programov, katerih dolžina bo občutno manjša od števila n.

Chaitin pa uporabi **problem zaustavljanja** ter pokaže, da, če je dano zaporedje tako dolgo, da je njegova kompleksnost večja kot kompleksnost sistema aritmetike, potem je v splošnem nemogoče dokazati, da gre za naključno zaporedje.

Algoritmično naključno število

V kriptografiji potrebujemo naključna števila na številnih mestih, npr. za generiranje ključev in za digitalni podpisi.

Generiranje naključnih števil z metanjem kovanca ali drugih fizičnih procesov je zamudno in drago, zato v praksi uporabimo

generatorje psevdonaključnih bitov

(angl. *pseudorandom bit generator* ali **PRBG**).

Ti začnejo s kratkim zaporedjem bitov in ga podaljšajo v bistveno daljše zaporedje bitov, ki je videti naključno.

Za $k, \ell \in \mathbb{N}$, $\ell \geq k+1$

(kjer je ℓ neka določena polinomska funkcija od k)

je **(k,ℓ)-PRBG** funkcija

$$f : (\mathbb{Z}_2)^k \longrightarrow (\mathbb{Z}_2)^\ell,$$

ki jo lahko izračunamo v polinomskem času

(kot funkcijo števila k),

tj., generator bitov, ki razširi naključno izbrano zaporedje dolžine k do polinomsko dolgega psevdonaključnega zaporedja v polinomskem času.

Vhod $s_0 \in \mathbb{Z}_2^k$ bomo imenovali **seme**, izhod $f(s_0) \in (\mathbb{Z}_2)^\ell$ pa **psevdonaključno zaporedje bitov**.

Funkcija f je deterministična (odvisna samo od semena).

Namesto nereda, statistike, je za kriptografijo bolje študirati (ne)napovedovanje naslednjega člena.

Kdo skuša napovedovati?

Ali lahko kdorkoli (z izjemo generatorja) v polinomskem času poišče polinomski algoritem za napoved naslednjega bita?

(Dodatno dovolimo še poznavanje podzaporedja bitov.)

Krepko psevdonaključno zaporedje bitov $\{b_i\}$ ima lastnost, da ne obstaja polinomski algoritem, ki bi iz zaporedja $b_j b_{j+1} \dots b_{j+m-1}$ napovedal bit b_{j-1}.

Od tod pa sledi, da noben polinomski algoritem ne loči krepko psevdonaključnega zaporedja bitov od resnično naključnih bitov.

Uporaba

Eden izmed konceptov popolne varnosti, ki smo ga študirali v poglavju o entropiji, je **enkratni ščit**

(Čistopis in ključ sta dve zaporedji bitov, ki ju zašifriramo tako, da ju seštejemo z XORjem.)

Njegov praktični problem je generiranje, *izmenjava in dolžina ključa,* tj. naključnega zaporedja bitov.

PRBG zmanjša potrebno količino naključnih bitov.

Primeri PRNG

- LFSR (Linear Feedback Shift Register) stopnje k za dano k-bitno seme generira dodatnih $2^k - k - 1$ bitov, preden se začne ponavljati,
- zaporedna uporaba
  - potenciranja (DLP),
  - simetričnega sistema,
  - zgoščevalne funkcije itd.,
- majhne, enostavne in poceni elektronske naprave

Intel načrtuje vgraditev v Pentium III mikroprocesor hardwarski RNG, glej našo domačo stran).

Intel starts preaching about security
EE Times Print, By Craig Matsumoto
(01/21/99, 3:27 p.m. EDT)

Intel: We won't track ID chips PC chip giant says it's walking on glass over the privacy considerations of new processor ID scheme
ZDNN, By Robert Lemos
(January 21, 1999 5:49 PM PT)

... For starters, Intel will burn a unique, secret identification number into every Pentium III that will ship.

...

Because the ID number also could be a privacy threat, Intel plans to allow end users to block transmission of the number, reportedly through a software patch. ...

For companies that sell into corporate networking environments, the ID number is a long-awaited relief. "We had dreamed of having a 'serial number' on the motherboard," ...

... Intel plans to provide a hardware-based random-number generator in every PC. The flaw in computer-generated pseudorandom numbers is that they fall in deterministic sequence; each "random" number is calculated based on its predecessor, making cycles and subtle patterns inevitable. Truly random numbers can only be gathered through physical phenomena, such as radioactive decay or, in Intel's case, thermal noise.

... Chances are, the hardware random-number generator will be used to select a "seed", or starting point, for an application's pseudorandom generator. ...

V primeru LFSR potrebujemo $2k$ zaporednih bitov za izračun semena. Torej PRBG iz LFSR ni varen.

Zalo hiter način za konstrukcijo PRBG s semenom dolžine k_1+k_2 iz dveh LFSR (stopnji k_1, k_2) so predlagali **Coppersmith, Krawczyk in Mansour** (angl. Shrinking Generator):

Če nam da prvi LFSR $a_1, a_2, \ldots$, drugi LFSR pa $b_1, b_2, \ldots$, definiramo zaporedje prevdo-naključnih bitov $z_1, z_2, \ldots$ s pravilom

$$z_i = a_{i_k},$$

kjer je i_k mesto k-te enice v zaporedju $b_1, b_2, \ldots$.

Čeprav je zgornja metoda za genereranje naključnih bitov izredno učinkovita in odporna proti mnogim napadom, pa se ni nikomur posrečilo, da bi dokazal njeno varnost.

Naj bosta p in q dve $(k/2)$-bitni praštevili (privatni) in $n = pq$ (javen). Izberimo si tak b (javen), da je $D(b, \phi(n)) = 1$.

Naj ima seme $s_0 \in \mathbb{Z}_n^*$ k-bitov. Za $i \geq 1$ definiramo

$$s_{i+1} = s_i^b \bmod n \;\text{ in }\; f(s_0) = (z_1, \ldots, z_\ell),$$

kjer je $z_i = s_i \bmod 2$, $1 \leq i \leq \ell$.

Potem je **(k, ℓ)-RSA generator**.

Generator $\frac{1}{P}$

P je dano praštevilo, b pa baza številskega sistema, $1 < b < P$, ki je primitiven koren po modulu P.

To zaporedje je generirano s števkami števila $1/P$ v številskem sistemu z osnovo b.

Čeprav zaporedje "izgleda naključno" zaradi periode $P-1$ (če je P recimo 50-mestno) in ima dobre statistične lastnosti, se izkaže, da se ta generator da napovedati.

Izrek. *Iz $\lceil \log_b 2P^2 \rceil$-bitnega podzaporedja lahko opazovalec v polinomskem času od $\log_b P$ določi naslednji člen zaporedja.*

Primer: Naj bo $b = 10$, dodatno pa predpostavimo, da ne poznamo P.

Generator vprašamo za 3 števke in dobimo: 407.

Število $0.407 = 407/1000$ zapišemo v obliki verižnega ulomka:

$$\frac{407}{1000} = 0 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{5 + \cfrac{1}{3 + \cfrac{1}{5 + \cfrac{1}{2}}}}}} = [0, 2, 2, 5, 3, 5, 2],$$

Zaporedje: 0, .5, .4, .4074, .406, .4070, .407=407/1000.

Prvi člen zgornjega zaporedja, ki ujame .407, je 11/27 in napove, da bo naslednja števka 4.

Generator pa nam da 3.

Ponovimo proces za 4073/10000 in dobimo 145/356 = .40730..., ki ujame .4073 in napove 0.

Generator pa nam da 3.

Ponovimo proces za 40733/100000 in dobimo

200/491 = .40733197556008146639511201629327902...

Ker se naslednjih 30 števk generatorja ujema z našimi, sprejmemo za $P = 491$ (SASA!!! preveri referenco).

Algoritem za psevdonaključne bite

Naj bo $n = pq$, kjer sta p in q praštevili, in naj bo c_0 tako naravno število, da je $(c_0/p) = (c_0/q) = 1$, kjer je $(-/-)$ Legendrov simbol.

1. Izberi $x_0 \in \mathbb{Z}_n^*$ tako, da je $D(x_0 + c_0, n) = 1$
2. Za začetni vrednosti x_0 in c_0 obnovi x_i in c_i $(i \geq 0)$ z $x_{i+1} \equiv x_i - c_i x_i^{-1} \pmod{n}$ in $c_{i+1} \equiv 4c_i \pmod{n}$.
3. Za $i \geq 0$ izračunaj in izpiši zaporedje $\{b_i\}$:

$$b_i = \begin{cases} 1, & \text{če je } x_i > -c_i x_i^{-1}; \\ 0, & \text{sicer.} \end{cases}$$

Blum-Blum-Shub generator

Za različni praštevili p in q naj bo $n = pq$. Potem velja za Jacobijev simbol naslednje:

$$\left(\frac{x}{n}\right) = \begin{cases} 0, & \text{če } D(x,n) > 1 \\ 1, & \text{če } \left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = 1 \text{ ali } \left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1 \\ -1, & \text{če je } \left(\frac{x}{p}\right)\left(\frac{x}{q}\right) = -1. \end{cases}$$

Naj bo $\quad \mathrm{QR}(n) = \{x^2 \bmod n \mid x \in \mathbb{Z}_n^*\}.$

Spomnimo se, da je $x \in \mathrm{QR}(n)$ če in samo, če je

$$\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = 1.$$

Potem definiramo množico **psevdokvadratov po modulu n** z

$$\tilde{\mathrm{QR}}(n) = \{x \in \mathbb{Z}_n^* \backslash \mathrm{QR}(n) \mid \left(\frac{x}{n}\right) = 1\}$$

oziroma

$$\tilde{\mathrm{QR}}(n) = \{x \in \mathbb{Z}_n^* \mid \left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1\}.$$

Naj bo seme s_0 poljuben element iz $\mathrm{QR}(n)$. Za $i \geq 0$ definiramo

$$s_{i+1} = s_i^2 \bmod n \;\text{ in }\; f(s_0) = (z_1, \dots, z_\ell),$$

kjer je $z_i = s_i \bmod 2$, $1 \leq i \leq \ell$.

Potem je (k,ℓ)-PRBG, imenovan **BBS-generator**.

Reference

[GW] I. Goldberg and D. Wagner, Randomness and the Netscape Browser, *Dr. Dobb's Journal*, January 1996, 66-70.

[Ba] S. Bassein, A Sampler of Randomness, *American Mathematical Monthly* **103** (1996), 483-490.

[KSWH] J. Kelsey, B. Schneier, D. Wagner and C. Hall, Cryptoanalytic Attacks on Pseudorandom Number Generators, probably from the Internet, 21 pages.

[P] C. Plumb, Truly Random Numbers, *Dr. Dobb's Journal*, November 1994, 113-114, 137-139.

[NGGLG] J. No, S. W. Solomon, G. Gong, H. Lee and P. Gaal, Binary Pseudorandom Sequences of Period $2^n - 1$ with Ideal Autocorrelation, *IEEE Transactions of Information Th.* **44**, March 1998, 814-817

[NGGLG] A. Chang, P. Gaal, S. W. Golomb, G. Gong and P. V. Kumar, Some Results Relating to a Sequence Conjectured to have Ideal Autocorrelation, submitted to *IEEE Transactions of Information Th.* in July 1998.

[Ch] G. J. Chaitin, Randomness in Arithmetic, *Scientific American*, July 1988, 52-57.

13. poglavje

Dokazi brez razkritja znanja

- sistemi za interaktivno dokazovanje
- popolni dokazi brez razkritja znanja
- zapriseženi biti (angl. bit commitments)
- računski dokazi brez razkritja znanja
- argumenti brez razkritja skrivnosti (angl. zero-knowledge arguments)

Dokaz brez razkritja znanja omogoča eni osebi, da prepriča drugo osebo o nekem dejstvu, ne da bi pri tem izdala katerokoli informacijo o dokazu.

Vohunova dilema

Bilo je temno kot v rogu, ko se je vohun vračal v grad po opravljeni diverziji v sovražnem taboru. Ko se je približal vratom, je zaslišal šepetajoč glas:

Geslo ali streljam!

ALI ŠEPETA PRIJATELJ ALI SOVRAŽNIK?

Kako lahko vohun prepriča "stražarja", da pozna geslo, ne da bi ga pri tem izdal morebitnemu vsiljivcu/prisluškovalcu?

Sistemi za interaktivno dokazovanje

(angl. Interactive Proof System)

Primož ("prover") ima neko skrivnost in bi rad dokazal Veri ("verifier"), da jo res ima.

Privzemimo, da sta Primož in Vera probabilistična algoritma, ki komunicirata preko javnega kanala. Vsak od njiju bo privatno računal in imel privatni generator naključnih števil.

Na začetku imata Primož in Vera skupen podatek x. Cilj interaktivnega dokaza je, da ima ta x neko določeno lastnost. Bolj natančno, x je DA-primer konkretnega odločitvenega problema Π.

Protokol je sestavljen iz več krogov, ki se sestojijo iz Verinega izziva in Primoževega odgovora. Na koncu postopka Vera bodisi **sprejme** ali **zavrne** dokaz, glede na to ali, Primož uspešno prestal izzive ali ne.

Protokol je **interaktivni dokaz** *za odločitveni problem Π, če sta izpolnjeni naslednji lastnosti, kadar Vera sledi protokolu:*

polnost: *če je odgovor odločitvenega problema Π pozitiven, Vera vedno sprejme Primožev dokaz,*

uglašenost: *če je odgovor odločitvenega problema Π negativen, je verjetnost, da Vera sprejme Primožev dokaz, zelo majhna.*

Omejili se bomo na sisteme interaktivnih dokazov, v katerih so Verini računi opravljeni v polinomskem času.

Po drugi stani pa ne postavljamo *nobene* omejitve za računsko moč, ki jo ima na voljo Primož.

Problem (izomorfizem grafov):
dva grafa z n vozlišči $G_i = (V_i, E_i)$, $i = 1, 2$.

Vprašanje:
Ali obstaja izomorfizem grafov $\pi : V_1 \longrightarrow V_2$?

Za ta problem ne poznamo polinomskega algoritma, kljub temu pa ni znano, ali je ta problem NP-poln.

Sistem za interaktivno dokazovanje, ki Primožu omogoči, da dokaže, da določena grafa nista izomorfna:

Podatki: *grafa G_1 in G_2 z vozlišči* $\{1, 2, \dots, n\}$.

Protokol: *n-krat ponovi naslednje korake:*

1. *Vera izbere naključno permutacijo vozlišč π in število $i \in \{1, 2\}$ ter pošlje Primožu graf H, ki ga dobi iz G_i s permutacijo π.*
2. *Primož ugotovi, za kateri j je G_j izomorfen grafu H, in pošlje j Veri, ki preveri, ali je $i = j$.*

Vera sprejme Primožev dokaz, če je vedno $i = j$.

Polnost: če grafa G_1 in G_2 nista izomorfna (in mora biti odgovor *pozitiven*), bo $j = i$ v vsakem krogu in bo Vera gotovo sprejela Primožev dokaz.

Uglašenost: če sta G_1 in G_2 izomorfna grafa (in naj bi bil odgovor *negativen*),
Primož nima možnosti, da bi ugotovil, če je Vera skonstruirala H iz G_1 ali G_2 in lahko v najboljšem primeru poskusi v svojem odgovoru uganiti, ali je $j = 1$ ali pa $j = 2$. Torej je verjetnost, da Vera sprejme vseh n pravilnih odgovorov 2^{-n}.

Verini algoritmi so polinomski, medtem ko je Primož imel na voljo neomejeno računsko moč (kar je dovoljeno/potrebno).

Popolni dokazi brez razkritja znanja

Sedaj pa si poglejmo poseben primer sistemov za interaktivno dokazovanje, ki jih imenujemo dokazi brez razkritja znanja.

Primož prepriča Vero, da ima x neko določeno lastnost, pri tem pa Vera še vedno ne ve, kako bi sama dokazala, da ima x to lastnost.

Formalna definicija je precej zapletena, zato si najprej oglejmo primer:
sistem za popolni dokaz brez razkritja znanja za izomorfizem grafov.

Podatki: *grafa G_1 in G_2 z vozlišči* $\{1, 2, \dots, n\}$.

Protokol: *n-krat ponovi naslednje korake:*

1. *Primož izbere naključno permutacijo vozlišč π ter pošlje Veri graf H, ki ga dobi iz G_1 s permutacijo π.*
2. *Vera izbere naključno število $i \in \{1, 2\}$ in ga pošlje Primožu.*
3. *Primož izračuna permutacijo vozlišč ρ, s katero dobimo graph H iz G_i, ter jo pošlje Veri, ki preveri, ali z njo res dobi H iz G_i.*

Vera sprejme Primožev dokaz, če v vsakem krogu res dobimo H iz G_i s Primoževo permutacijo ρ.

Permutacija ρ, ki jo Primož izračuna v 3. koraku, je za $i = 1$ enaka π, za $i = 2$ pa kompozitumu permutacij σ in π, kjer je σ permutacija, s katero dobimo G_1 iz G_2.

Polnost je očitna, kakor tudi uglašenost, saj je edini način, da Primož prevara Vero, da si vsakokrat pravilno izbira i, ki ga bo dobil od Vere, Veri pa pošlje $H = \pi(G_i)$.

Vse Verine operacije imajo polinomsko zahtevnost, enako pa velja tudi za Primoževe operacije, vendar le s pogojem, če pozna izomorfizem med G_1 in G_2.

Vse, kar je dobila Vera, je nekaj naključnih permutacij grafov G_1 in G_2 (ki bi jih lahko skonstruirala tudi sama).

Informacije, ki jih je dobila Vera, imenujmo **zapis** in se sestojijo iz:

1. grafa G_1 in G_2 z vozlišči $\{1, 2, \ldots, n\}$,
2. vseh sporočil, ki sta jih poslala Primož in Vera.

V primeru problema izomorfnosti grafov ima zapis naslednjo obliko:

$$T_{IG} = ((G_1, G_2); (H_1, i_1, \rho_1); \ldots; (H_n, i_n, \rho_n)).$$

Poudarjamo, da lahko vsakdo ponaredi zapis (ne da bi poznal pravi zapis), če sta vhodna grafa G_1 in G_2 res izomorfna. Tak algoritem se imenuje **simulator**.

Definicija 1: *Predpostavimo, da imamo*

1. *sistem za interaktivni dokaz odločitvenega problema Π s polinomsko časovno zahtevnostjo,*
2. *simulator S s polinomsko časovno zahtevnostjo.*

Naj bo $\mathcal{T}(x)$ množica vseh možnih zapisov, ki ju lahko ustvarita Primož in Vera med interaktivnim dokazom za DA-primer x.

Naj bo $\mathcal{F}(x)$ množica vseh možnih ponarejenih zapisov za simulator S.

Za poljuben zapis $T \in \mathcal{T}(x)$ naj bo $p_{\mathcal{T}}(T)$ verjetnost, da je T zapis interaktivnega dokaza.

Za poljuben zapis $T \in \mathcal{F}(x)$ naj bo $p_{\mathcal{F}}(T)$ verjetnost, da je T ponarejen zapis simulatorja S.

Če je $\mathcal{T}(x) = \mathcal{F}(x)$ in za vsak $T \in \mathcal{T}(x)$ velja $p_{\mathcal{T}}(T) = p_{\mathcal{F}}(T)$, potem je sistem za interaktivni dokaz **popoln dokaz za razkritje znanja** *(za Vero).*

Z drugimi besedami: Vera lahko stori potem, ko je izvedla protokol, samo toliko kot simulator, potem ko je zgeneriral ponarejeni zapis.

Izrek 1: *Sistem z interaktivnim dokazom za problem* **izomorfizema grafov** *je popoln dokaz brez razkritja znanja za Vero.*

Dokaz: Naj bosta G_1 in G_2 grafa z vozlišči $\{1, 2, \ldots, n\}$. Prepis (pravi ali ponarejeni) se sestoji iz trojic (H, i, ρ), kjer je $i \in \{1, 2\}$, ρ permutacija vozlišč in H graf, ki ga dobimo iz G_i s permutacijo ρ.

Taki trojici bomo rekli, da je **veljavna**, množico vseh veljavnih trojic pa bomo označili z $\mathcal{R}$. Potem je $|\mathcal{R}| = 2n!$, verjetnosti posameznih trojic pa so med seboj enake in je

$$p_{\mathcal{T}}(T) = p_{\mathcal{F}}(T) = \frac{1}{(2n!)^n} \quad \text{za vsak zapis } T. \blacksquare$$

V dokazu smo privzeli, da Vera sodeluje pri protokolu. Kaj pa če temu ni tako? Recimo, da Vera v vsakem krogu namenoma izbere $i = 1$.

Potem bo za razliko od Vere simulator sestavil tak zapis le z verjetnostjo 2^{-n}.

Iz tega razloga bomo morali pokazati, da je polinomski simulator, ki bo sestavil ponarejen zapis, ki je videti kot Primožev zapis, nastal v sodelovanju z goljufivo Vero.

Definicija 2: *Predpostavimo, da imamo*

1. *sistem za interaktivni dokaz odločitvenega problema Π s polinomsko časovno zahtevnostjo,*
2. *za probabilistični algoritem V^* (po možnosti goljufiv) s polinomsko časovno zahtevnostjo naj bo $S^* = S^*(V^*)$ simulator s polinomsko časovno zahtevnostjo.*

Naj bo $\mathcal{T}(V^*, x)$ množica vseh možnih zapisov, ki ju lahko ustvarita Primož in V^* med interaktivnim dokazom za DA-primer x.

Naj bo $\mathcal{F}(V^*, x)$ množica vseh možnih ponarejenih zapisov za simulator S^*.

Za poljuben zapis $T \in \mathcal{T}(V^*, x)$ naj bo $p_{\mathcal{T}}(T)$ verjetnost, da je T zapis interaktivnega dokaza z V^*.

Za poljuben zapis $T \in \mathcal{F}(V^*, x)$ naj bo $p_{\mathcal{F}}(T)$ verjetnost, da je T ponarejen zapis simulatorja S.

Če je $\mathcal{T}(V^*, x) = \mathcal{F}(V^*, x)$ in za vsak $T \in \mathcal{T}(x)$ velja $p_{\mathcal{T},V^*}(T) = p_{\mathcal{F},V^*}(T)$, potem je sistem za interaktivni dokaz **popoln dokaz brez razkritja znanja** *(brez oporekanja).*

Če vzamemo za V^* pošteno Vero, dobimo staro definicijo popolnega dokaza brez razkritja znanja za Vero.

Da bi pokazali, da je sistem za dokazovanje popoln dokaz brez razkritja znanja, potrebujemo generično transformacijo, ki skonstruira simulator S iz poljubnega V^*.

Storimo to na primeru problema izomorfizma grafov.

Simulator S^* poskuša uganiti izziv i_j, ki ga pošlje V^* v vsakem krogu j, tj. S^* generira trojico (H_j, i_j, ρ_j) in pokliče V^*, da dobi izziv i'_j:

- če je $i_j = i'_j$, potem priključimo trojico k zapisu,
- če je $i_j \neq i'_j$, potem S^* izbere nov izziv i_j ter ponovno požene V^* s starim stanjem.

Čeprav je možno, da se simulator spoh ne bi ustavil, lahko pokažemo, da je povprečen čas simulatorja polinomski in da sta verjetnostni porazdelitvi $p_{\mathcal{T},V^*}(T)$ in $p_{\mathcal{F},V^*}(T)$ identični.

> **Izrek 2:** *Sistem z interaktivnim dokazom za problem* **izomorfizem grafov** *je popoln dokaz brez razkritja znanja.*

Dokaz: Ne glede na to, kako V^* generira izziv i_j, je verjetnost, da ga bo simulator S^* zadel z i'_j enaka $1/2$. Torej potrebuje S^* v povprečju dve trojici za en dodatek ponarejenemu zapisu in je njegov povprečni čas polinomsko odvisen od n.

Drugi del ($p_{\mathcal{T},V^*}(T) = p_{\mathcal{F},V^*}(T)$ za vsak T) je težji, saj je izbira izziva lahko odvisna od prejšnjih izzivov in Primoževih odzivov nanje. Uporabimo indukcijo na število krogov.

Za $0 \leq j \leq n$ naj bosta $p_{\mathcal{T},V^*,j}(T)$ in $p_{\mathcal{F},V^*,j}(T)$ verjetnostni distribuciji na množici delnih zapisov $\mathcal{T}_j$, ki jih lahko dobimo na koncu j-tega kroga.

Očitno je $\;p_{\mathcal{T},V^*,0}(T) = p_{\mathcal{F},V^*,0}(T)\;$ za vsak $T \in \mathcal{T}_0$.

Sedaj pa predpostavimo, da sta verjetnostni distribuciji $p_{\mathcal{T},V^*,j}$ in $p_{\mathcal{F},V^*,j}$ identični na $\mathcal{T}_{j-1}$ za $j \geq 1$.

Naj bo verjetnost, da je v j-tem krogu iterativnega dokaza izziv $i'_j = 1$ neko število $p_1 \in \mathbb{R}$, ki je odvisno od stanja algoritma V^*.

Potem je verjetnost, da je (H, i, ρ) trojica na zapisu enaka $p_1/n!$, če je $i-1$ in $(1-p_1)/n!$ sicer.

Simulator pa bo zapisal v zapis trojico (H, i, ρ) z verjetnostjo

$$\frac{p_1}{2n!}\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots\right) = \frac{p_1}{n!},$$

če je $i = 1$ in verjetnostjo $(1-p_1)/n!$ sicer.

Po indukciji je dokaz končan. ■

Problem (kvadratni ostanki)

Podatki: *število n z neznano faktorizacijo $n = pq$, kjer sta p in q praštevili in $x \in \mathrm{QR}(n)$.*

Protokol: $\log_2 n$*-krat ponovi naslednje korake:*

1. *Primož izbere naključno število $v \in \mathbb{Z}_n^*$, izračuna $y = v^2 \bmod n$ in ga pošlje Veri.*
2. *Vera izbere $i \in \{1, 2\}$ in ga pošlje Primožu.*
3. *Primož izračuna $z = u^i v \bmod n$, kjer je u kvadratni koren iz x, in ga pošlje Veri.*
4. *Vera preveri, ali je $z^2 \equiv x^i y \pmod{n}$.*

Vera sprejme Primožev dokaz, če je bilo v vsakem krogu res $z^2 \equiv x^i y \pmod{n}$.

Poglejmo si še en problem. Le-ta je povezan s problemom diskretnega logaritma.

Problem (članstvo v podgrupi):

števila $n, \ell \in \mathbb{N}$ in različna elementa $\alpha, \beta \in \mathbb{Z}_n^*$, pri čemer je ℓ red elementa α v grupi $\mathbb{Z}_n^*$.

Vprašanje:
ali je $\beta = \alpha^k$ za neko število k, $0 \leq k \leq \ell - 1$, tj. ali je β element podgrupe, generirane z α?

Za D.N. preverite, da je naslednji interaktivni dokaz zares popoln dokaz brez razkritja znanja.

Podatki: *število $n \in \mathbb{N}$ in različna elementa $\alpha, \beta \in \mathbb{Z}_n^*$, pri čemer je ℓ (javen) red elementa α v grupi $\mathbb{Z}_n^*$.*

Protokol: $\log_2 n$*-krat ponovi naslednje korake:*

1. *Primož izbere naključno število $j \in \mathbb{Z}_\ell$, izračuna $\gamma = \alpha^j \bmod n$, in ga pošlje Veri.*
2. *Vera izbere $i \in \{1, 2\}$ in ga pošlje Primožu.*
3. *Primož izračuna $h = j + ik \bmod \ell$, kjer je $k = \log_\alpha \beta$ in ga pošlje Veri.*
4. *Vera preveri, ali je $\alpha^h \equiv \beta^i \gamma \pmod{n}$.*

Vera sprejme Primožev dokaz, če je bilo v vsakem krogu res $\alpha^h \equiv \beta^i \gamma \pmod{n}$.

Zapriseženi biti

(angl. Bit Commitments)

Izomorfizem grafov je gotovo zanimiv problem, a bi bilo še lepše, če bi poznali kakšen sistem z dokazom brez razkritja znanja za **NP-poln** problem.

Teoretični rezultati kažejo, da ne obstajajo popolni dokazi brez razkritja znanja za NP-polne probleme.

Opisali pa bomo sisteme za dokazovanje, ki imajo za odtenek šibkejšo stopnjo "nerazkritja", tako imenovano *računsko nerazkritje*. Te sisteme bomo opisali v naslednjem razdelku, tu pa opišimo metode, ki jih uporabljamo v sistemih dokazovanja.

Oglejmo si naslednji scenarij:

Primož napiše na list papirja neko sporočilo, ga zaklene v sef s samo njemu poznano kombinacijo in sef izroči Veri.

Čeprav Vera ne pozna sporočila, dokler je sef zaklenjen, je Primož "*zaobvezan*", tj. ne more več spremeniti sporočila.

Če Vera ne pozna kombinacije, ne more priti do sporočila, dokler ji Primož ne izda kombinacije (oziroma odpre sef).

Naj bo sporočilo en sam bit $b \in \{0,1\}$, ki ga Primož zašifrira. To šifriranje bomo imenovali **shema zaobvezanih bitov**. V splošnem je to funkcija $f : \{0,1\} \times X \longrightarrow Y$, kjer sta X in Y končni množici.

Za **shemo zaobvezanih bitov** si želimo naslednjih lastnosti:

1. **prikrivanje** (angl. concealing): Vera ne more določiti vrednost bita b iz funkcijske vrednosti $f(b,x)$,
2. **vezava** (angl. binding): Primož lahko odpre $f(b,x)$ z odkritjem vrednosti x in s tem prepriča Vero, da je bil zašifriran b. Pri tem pa ne more odpreti $f(b,x)$ v oba bita 0 in 1.

Če želimo zapriseči zaporedje bitov, lahko to storimo bit za bitom. V poglavju o generatoru naključnih števil je omenjena taka metoda: **Goldwasser-Micali probabilističen kriptosistem**.

Poleg sistemov za dokazovanje lahko te sheme za zaprisego bitov uporabimo tudi za *metanje kovanca po telefonu*:

Anita in Bojan se želita skupaj odločiti na osnovi meta kovanca, vendar pa se ne nahajata na istem mestu. Torej je nemogoče, da eden izmed njiju vrže kovanec, drugi pa preveri izid.

Metoda z zapriseženimi biti nam pomaga iz zagate:

1. Anita izbere bit b ter pošlje $f(b,x)$ Bojanu,
2. Bojan poskuši uganiti b,
3. Anita odklene b.

Prikrivanje onemogoča Bojanu, da bi iz $f(b,x)$ izračunal b, *vezava* pa preprečuje Aniti, da bi se premislila za Bojanovim ugibanjem.

Za $p \equiv 3 \pmod 4$ smo se v 5.1.2 prepričali, da nam neizračunljivost DLP v $\mathbb{Z}_p^*$ zagotavlja varnost drugega (najmanjšega) bita (SLB) diskretnega logaritma.

Naj bo $X = \{1, \dots, p-1\}$ in $Y = \mathbb{Z}_p^*$ in

$$\mathrm{SLB}(x) = \begin{cases} 0, & \text{če } x \equiv 0,1 \pmod 4 \\ 1, & \text{če } x \equiv 2,3 \pmod 4), \end{cases}$$

shemo za zaprisežene bite pa definirajmo z

$$f(b,x) = \begin{cases} \alpha^x \bmod p, & \text{če je } \mathrm{SLB}(x) = b \\ \alpha^{p-x} \bmod p, & \text{če je } \mathrm{SLB}(x) \neq b. \end{cases}$$

Računski dokazi brez razkritja znanja

(angl. computational Zero-knowledge Proofs)

Definirajmo NP-poln problem:

Problem (pravilno 3-barvanje grafa):
graf $G = (V,E)$ z n vozlišči.

Vprašanje:
ali obstaja pravilno 3-barvanje grafa G,
tj. ali obstaja taka funkcija $\phi : V(G) \longrightarrow \{1,2,3\}$,
da iz $\{u,v\} \in E$ sledi $\phi(u) \neq \phi(v)$?

Naj bo $V = V(G) = \{1, \dots, n\}$ in $m = |E|$.

Shema za zaprisežene bite $f : \{0,1\} \times X \longrightarrow Y$ naj bo javna.

Zakodiranje barv: $1 \rightarrow 01$, $2 \rightarrow 10$, $3 \rightarrow 11$.

Interaktivni dokaz z m^2 krogi:

1. Primož zapriseže (se obveže za) barvanje, ki je permutacija nekega fiksnega barvanja ϕ.
2. Vera zahteva od Primoža, da odkrije barvi dveh naključno izbranih sosedov.
3. Primož to stori, Vera pa preveri, če sta barvi zares različni.

Polnost je očitna. ***Uglašenost*** pa sledi iz naslednjega razmisleka. Izračunajmo verjetnost, da je Vera sprejela nepravilno 3-barvanje grafa G. Obstajati mora vsaj en par sosednjih vozlišč iste barve.

Verjetnost, da Vera izbere to povezavo, je $1/m$, torej po m^2 krogih je verjetnost prevare kvečjemu

$$\left(1 - \frac{1}{m}\right)^{m^2}.$$

Ker je $\lim_{m\to\infty}(1-1/m)^m = e^{-1}$, obstaja število m_0, za katero je $(1-1/m)^{m^2} \leq (2/e)^m$ za vse $m \geq m_0$.

Sedaj pa preverimo še, kako je s popolnim dokazom brez razkritja znanja. Vse, kar Vera vidi po vsakem krogu, je zašifrirano 3-barvanje na dveh sosednjih vozliščih (ki ga je prej zaprisegel Primož).

Ker Primož po vsakem krogu spremeni permutacijo, Vera ne more kombinirati informacij iz različnih krogov in rekonstruirati 3-barvanje.

Ta sistem ni popoln dokaz brez razkritja znanja, je pa zato *računski dokazi brez razkritja znanja*. Le-tega definiramo na enak način kot prvega, le da zahtevamo za ustrezna zapisa, da sta kvečjemu polinomsko neločljiva.

Argumenti brez razkritja skrivnosti

(angl. Zero-knowledge Arguments)

Če uporabimo $f(b,x)$, kjer je **prikrivanje** brezpogojno, Primož pa ima na voljo le polinomski čas, dobimo **argument** brez razkritja skivnosti.

Poglejmo si še enkrat shemo za zaprisežene bite z DLP. Naj bo p praštevilo, za katerega je DLP v $\mathbb{Z}_p^*$ neizračunljiv. Naj bo α primitiven element iz $\mathbb{Z}_p^*$ in $\beta \in \mathbb{Z}_p^*$.

Naj bo $X = \{0,\dots,p-1\}$, $Y = \mathbb{Z}_p^*$ in funkcija f definirana z

$$f(b,x) = \beta^b \alpha$$

Dodatek A

Dokaz izreka o gostoti praštevil

- pomožni izreki z dokazi
- dve posledici analitičnega izreka
- izrek o gostoti praštevil izpeljemo kot direktno posledico druge posledice analitičnega izreka

Dokaz: Sledimo D. Zagieru
(Newman's Short Proof of the PNT,
American Mathematical Monthly, October 1997, strani 705-709).

Riemannova funkcija zeta

$$\zeta(s) := \sum_{n=1}^{\infty} \frac{1}{n^s},$$

kjer je $s = \sigma + i\tau$, $\sigma, \tau \in \mathbb{R}$.

Izrek I: *V območju $\sigma > 1$*
(a) je vrsta $\zeta(s)$ absolutno konvergentna,
(b) $\zeta(s) = \prod_p \frac{1}{1-\frac{1}{p^s}}$ [Euler, 1737],
(c) funkcija $\zeta(s)$ nima ničel.

Dokaz izreka I: (a) Velja

$$|n^s| = |n^\sigma||n^{i\tau}| = |n^\sigma|.$$

Majoranta $\sum_{n=1}^{\infty} \left|\frac{1}{n^s}\right|$ je konvergentna po Raabejevem kriteriju (ali pa integralskem kriteriju) za $\sigma > 1$.

(b) $\sum_n n^{-s} = \sum_n \left(2^{r_2} 3^{r_3} \cdots p_m^{r_m}\right)^{-s}$,

po drugi strani pa imamo za različni praštevili p in q.

$$\left(1+\frac{1}{p^s}+\frac{1}{p^{2s}}+\cdots\right)\left(1+\frac{1}{q^s}+\frac{1}{q^{2s}}+\cdots\right) =$$

$$= 1 + \left(\frac{1}{p^s}+\frac{1}{q^s}\right) + \left(\frac{1}{p^{2s}}+\frac{1}{p^s}\frac{1}{q^s}+\frac{1}{q^{2s}}\right) + \cdots =$$

$$= (1-p^{-s})^{-1}\,(1-q^{-s})^{-1}$$

Ker je $|p^{-s}|, |q^{-s}| < 1$ za $\sigma > 1$, lahko zaradi absolutne konvergence (člene lahko seštevano v poljubnem vrstnem redu) sklepamo, da velja

$$\sum_n n^{-s} = \prod_p \Big(\sum_{r\in\mathbb{N}_0} p^{-rs}\Big) = \prod_p \left(1-p^{-s}\right)^{-1}.$$

(c) Z uporabo trikotniške neenakosti

$$\big||a|-|b|\big| \;\le\; |a \pm b| \;\le\; |a|+|b|$$

za poljubni kompleksni števili a in b,
za $a = 1$ in $b = p^{-s}$, zapišemo

$$\big|1-|p^{-s}|\big| \;\le\; |1-p^{-s}| \;\le\; 1+|p^{-s}|\,.$$

Ker je $|p^s| = |p^\sigma|$ velja

$$\left|\frac{1}{1-p^{-s}}\right| > \frac{1}{1+p^{-\sigma}}\,.$$

Torej je

$$|\zeta(s)| > \prod_p (1+p^{-\sigma})^{-1}.$$

Toda

$$\prod_p (1+p^{-\sigma})^{-1} = e^{-\sum_p \log(1+p^{-\sigma})}$$

$$= e^{-\sum_p \left(\frac{1}{p^\sigma}+\frac{1}{2p^{2\sigma}}-\frac{1}{3p^{3\sigma}}+\dots\right)} >$$

$$> e^{-\sum_p \frac{1}{p^\sigma}} > 0$$

za $\sigma > 1$. ■

Izrek II: *Kompleksno funkcijo*

$$\zeta(s) - \frac{1}{s-1}$$

je mogoče razširiti holomorfno v območje $\mathcal{R}(s) > 0$.

Posledica: *Riemannova funkcija zeta* $\zeta(s)$ *ima v območju* $\mathcal{R}(s)>0$ *en sam pol* $s=1$ *z residuumom 1.*

Dokaz izreka II: Za $\sigma > 1$ lahko zapišemo

$$\zeta(s) - \frac{1}{s-1} = \sum_{n=1}^{\infty} \frac{1}{n^s} - \int_1^{\infty} \frac{dx}{x^s}$$

$$= \sum_{i=1}^{\infty} \int_n^{n+1} \left(\frac{1}{n^s} - \frac{1}{x^s} \right) dx.$$

Ocenimo

$$\left| \int_n^{n+1} \left(\frac{1}{n^s} - \frac{1}{x^s} \right) dx \right| = \left| s \int_n^{n+1} \left(\int_n^x \frac{du}{u^{s+1}} \right) dx \right|$$

oziroma za $n < \xi < n+1$

$$\left| s \int_n^{n+1} \left(\int_n^x \frac{du}{u^{s+1}} \right) dx \right| = |s| \left| \int_n^{\xi} \frac{du}{u^{s+1}} \right| \le$$

$$\le |s| \int_n^{\xi} \frac{du}{|u^{s+1}|} \le \frac{|s|}{|n^{s+1}|} = \frac{|s|}{n^{1+\mathcal{R}(s)}} .$$

Majoranta $\sum_{n=1}^{\infty} \frac{1}{n^{1+\delta}}$ absolutno konvergira za vsak $\delta > 0$. ■

Vpeljimo

$$\vartheta(x) := \sum_{p \le x} \log p \qquad \text{za } x \in \mathbb{R}.$$

Ponovimo: če za realni funkciji $f(x)$ in $g(x) \ge 0$ obstaja konstanta M, tako da je

$$|f(x)| \le Mg(x) \quad \text{za vsak } x \in \mathbb{R},$$

pišemo $f(x) = \mathcal{O}(g(x))$ ali na kratko $f = \mathcal{O}(g)$.

Izrek III: $\vartheta(x) = \mathcal{O}(x)$

Dokaz izreka III: Za naravno število n velja

$$(1+1)^{2n} = \sum_{j=0}^{2n} \binom{2n}{j} \ge \binom{2n}{n} \ge \prod_{n<p\le 2n} p = e^{\vartheta(2n)-\vartheta(n)}$$

oziroma

$$\vartheta(2n) - \vartheta(n) \le 2n \log 2.$$

Od tod sledi za konstanto $C > \log 2$

$$\vartheta(x) - \vartheta(x/2) \le Cx, \quad \forall x \ge x_0 = x_0(C).$$

Seštejmo zgornjo neenakost za $x, \frac{x}{2}, \frac{x}{4}, \ldots, \frac{x}{2^r}$, kjer je $2^r \ge x$, in dobimo

$$\vartheta(x) \le 2Cx + \mathcal{O}(1), \quad \forall x . \quad ■$$

Trditev: *Vrsta* $\sum_p \frac{\log p}{p^s}$, $s = \sigma + i\tau$, $\sigma > 1$ *je absolutno konvergentna.*

Dokaz: Če vstavimo v vrsto dodatne ničle, je njena majoranta $\sum_{n \in \mathbb{N}} \frac{\log n}{n^s}$.

Ker je $\sigma > 1$, $\exists \Delta > 0$, tako da je $\sigma = 1 + \Delta$ in $\exists \delta > 0$, da je $\delta < \Delta$, potem velja

$$\frac{\log n}{n^{\sigma}} = \frac{\log n}{n^{\Delta - \delta}} \cdot \frac{1}{n^{1+\delta}}$$

kjer gre prvi faktor na desni strani $\to 0$ za $x \to \infty$ (po L'Hospitalu). ■

Sedaj lahko vpeljemo kompleksno funkcijo

$$\Phi(s) := \sum_p \frac{\log p}{p^s} \quad \text{za } s \in \mathbb{C},$$

ki je holomorfna v območju $\mathcal{R}(s) > 1$.

Izrek IV: *Kompleksna funkcija*

$$\Phi(s) - \frac{1}{s-1}$$

je holomorfna v območju $\mathcal{R}(s) \ge 1$.

Dokaz trditve IV: Obravnavati je potrebno samo še enačaj, tj. $\sigma = 1$. Za $\sigma > 1$ iz izreka I sledi z logaritmičnim odvajanjem

$$-\frac{\zeta'(s)}{\zeta(s)} = \sum_p \frac{p^{-s} \log p}{1 - p^{-s}} \left(= \sum_p \frac{\log p}{p^s - 1} \right).$$

oziroma

$$\boxed{-\frac{\zeta'(s)}{\zeta(s)} = \Phi(s) + \sum_p \frac{\log p}{p^s(p^s - 1)}}$$

Ker je funkcija $-\dfrac{\zeta'(s)}{\zeta(s)}-\dfrac{1}{s-1}$ holomorfna pri $\sigma>1$ po izreku I, sklepamo za $\sigma>1$

$$\lim_{\sigma\to 1} -\frac{\zeta'(\sigma)}{\zeta(\sigma)} \Big/ \frac{1}{\sigma-1} = 1.$$

Od tod po izreku II iz identitete

$$\Phi(s)-\frac{1}{s-1} = -\frac{\zeta'(s)}{\zeta(s)}-\frac{1}{s-1}-\sum_p \frac{\log p}{p^s(p^s-1)}$$

sledi izrek IV, kakor hitro pokažemo, da $\zeta(1+i\tau)\neq 0$ za vsak $\tau\in\mathbb{R}$.

Naj bosta za funkcijo $\zeta(s)$ števili a in b zaporedoma stopnji domnevnih ničel

$$w=1+i\alpha \ \text{ in } \ z=1+i2\alpha, \quad \text{kjer je } \alpha\neq 0,$$

torej dopuščamo, da $a=0$ ali $b=0$.
Pokazali bomo, da je $a=0$.

Najprej se prepričajmo, da velja

$$\lim_{\varepsilon\searrow 0} \varepsilon\ \Phi(1+\varepsilon+i\alpha) \ = \ -a.$$

Zapišimo $\zeta(s)=(s-w)^a P(s-w)$, kjer je $P(w)\neq 0$, $\sigma>1$ in od tod tudi

$$\frac{\zeta'(s)}{\zeta(s)} \ = \ \frac{a}{s-w}+\frac{P'}{P}\,.$$

V identiteto

$$-\frac{\zeta'(s)}{\zeta(s)} \ = \ \Phi(s)+\sum_p \frac{\log p}{p^s(p^s-1)}\,,$$

ki velja za $\sigma>1$, vstavimo $s=1+\varepsilon+i\alpha$, $\varepsilon>0$ in jo pomnožimo z ε (tj. $s-w$) ter dobimo

$$\lim_{\varepsilon\searrow 0} \varepsilon\ \Phi(1+\varepsilon+i\alpha) \ = \ -a$$

Podobno se prepričamo, da velja tudi

$$\lim_{\varepsilon\searrow 0} \varepsilon\ \Phi(1+\varepsilon-i\alpha) \ = \ -a$$

$$\lim_{\varepsilon\searrow 0} \varepsilon\ \Phi(1+\varepsilon\pm i2\alpha) \ = \ -b$$

in z uporabo naslednje limite

$$\lim_{\sigma\to 1} -\frac{\zeta'(\sigma)}{\zeta(\sigma)} \Big/ \frac{1}{\sigma-1} = 1$$

še

$$\lim_{\varepsilon\searrow 0} \varepsilon\ \Phi(1+\varepsilon) \ = \ 1.$$

Naj bo $\sigma=1+\varepsilon$. Izračunane limite bomo povezali na podlagi naslednjih identitet in neenakosti

$$\sum_{k=-2}^{2}\binom{4}{2+k}\Phi(\sigma+ik\alpha)=\phi(\sigma-i2\alpha)+$$

$$+4\phi(\sigma-i\alpha)+6\phi(\sigma)+4\phi(\sigma+i\alpha)+\phi(\sigma+i2\alpha)=$$

$$=\sum_p \frac{\log p}{p^\sigma}\,(p^{i2\alpha}+p^{-i2\alpha}+4(p^{i\alpha}+p^{-i\alpha})+6)$$

in od tod

$$\sum_p \frac{\log p}{p^\sigma}\,(p^{i\alpha/2}+p^{-i\alpha/2})^4 \ \geq \ 0.$$

Sledi

$$\lim_{\varepsilon\searrow 0} \varepsilon \sum_{k=-2}^{2}\binom{4}{2+k}\Phi(1+\varepsilon+ik\alpha)\geq 0.$$

Torej je $-2b-8a+6\geq 0$ in mora biti $a=0$. ■

Analitični izrek

(a) Naj bo $f(t)$ za $t\geq 0$ omejena in lokalno integrabilna funkcija.

(b) Naj v območju $\mathcal{R}(z)>0$ obstaja funkcija

$$g(z):=\int_0^\infty f(t)e^{-zt}\,dt\,,$$

ki jo lahko holomorfno razširimo v $\mathcal{R}(z)\geq 0$.

Potem obstaja integral $\int_0^\infty f(t)\,dt$ in je enak $g(0)$.

Posledica:

$$\int_1^\infty \frac{\vartheta(x)-x}{x^2}\,dt \ < \ \infty\,.$$

Dokaz: Posplošeni Stieltjesov integral funkcije x^{-s}, $\sigma:=\mathcal{R}(s)>1$, obstaja glede na stopničasto funkcijo $\vartheta(x)$ in velja enakost

$$\Phi(s)=\int_1^\infty \frac{d\vartheta(x)}{x^s},$$

kar sledi iz dejstva, da za poljuben $\varepsilon>0$ obstaja $a\in[1,\infty)$, tako da za vsak $b>a$ velja

$$\int_a^b \left|\frac{1}{x^s}\right| d\vartheta \ = \ \int_a^b \frac{1}{x^\sigma}\,d\vartheta \ = \ \sum_{a\leq p\leq b}\frac{\log p}{p^\sigma} \ < \ \varepsilon.$$

Z integracijo po delih ($u = x^{-s}$, $dv = d\vartheta(x)$) in uvedbo nove spremenljivke ($x = e^t$) hitro pokažemo, da je

$$\Phi(s) \;=\; s\int_1^\infty \frac{\vartheta(x)}{x^{s+1}}\,dx \;=\; s\int_0^\infty e^{-st}\vartheta(e^t)\,dt.$$

Funkcija

$$f(t) := \vartheta(e^t)e^t - 1, \quad t \geq 0$$

je omejena in lokalno integrabilna po izreku III.

Funkcija

$$g(z) =: \int_0^\infty f(t)e^{-zt}\,dt \;=\; \int_0^\infty (e^{-(z+1)t}\vartheta(e^t) - e^{-zt})\,dt \;=$$

$$= \;\frac{\Phi(z+1)}{z+1} - \frac{1}{z} \;=\; \frac{\Phi(s)}{s} - \frac{1}{s-1},$$

kjer je $s = z+1$, obstaja in jo lahko po izreku IV holomorfno razširimo v območje $\mathcal{R}(s) \geq 1$, tj. $\mathcal{R}(z) \geq 0$, se pravi, da ustreza pogojem analitičnega izreka. Zato integral $\int_0^\infty f(t)\,dt$ obstaja in z uvedbo spremenljivke $x = e^t$ (ta funkcija je monotona na $[0,\infty)$) preide v željeni integral. ■

Posledica: $\displaystyle\lim_{x\to\infty} \frac{\vartheta(x)}{x} = 1.$

Dokaz (z redukcijo protislovja): Če pri nekem $\lambda > 1$ in pri poljubno velikih x velja $\vartheta(x) \geq \lambda x$, je

$$\int_x^{\lambda x} (\vartheta(t) - t)t^{-2}\,dt \;\geq\; \int_x^{\lambda x} \frac{\lambda t - t}{t^2}\,dt \;=\; \int_1^\lambda \frac{\lambda - u}{u^2}\,du > 0,$$

kar je v nasprotju s prejšnjo posledico. Na podoben način pridemo do protislovja pri domnevi $\lambda < 1$. ■

Dokaz izreka o gostoti praštevil:

Očitno je

$$\vartheta(x) \;=\; \sum_{p\leq x}\log p \;\leq\; \sum_{p\leq x} 1\cdot\log x \;=$$

$$=\; (\log x)\sum_{p\leq x} 1 \;=\; \pi(x)\log x.$$

Torej je

$$\boxed{\vartheta(x) \;\leq\; \pi(x)\log x\;.}$$

Zaradi $\pi(x) \leq x$ za poljuben $\varepsilon > 0$ velja

$$\vartheta(x) \;\geq\; \sum_{x^{1-\varepsilon}<p\leq x}\log p \;\geq\; (\log x^{1-\varepsilon}) \sum_{x^{1-\varepsilon}<p\leq x} 1 \;\geq$$

$$\geq \log x^{1-\varepsilon}\left(\pi(x) - \pi(x^{1-\varepsilon})\right) \geq (1-\varepsilon)\log x(\pi(x) - x^{1-\varepsilon})$$

oziroma

$$\boxed{\vartheta(x) \;\geq\; (1-\varepsilon)\log x\;(\pi(x) - x^{1-\varepsilon})\;.}$$

Iz obeh ocen dobimo za vsak $x > 0$ in $\varepsilon \in (0,1)$

$$(1-\varepsilon)\left(\frac{\pi(x)}{\frac{x}{\log x}} - \frac{\log x}{x^\varepsilon}\right) \;\leq\; \frac{\vartheta(x)}{x} \;\leq\; \frac{\pi(x)}{\frac{x}{\log x}}$$

oziroma

$$\frac{\vartheta(x)}{x} \;\leq\; \frac{\pi(x)}{\frac{x}{\log x}} \;\leq\; \frac{\vartheta(x)}{x(1-\varepsilon)} \;+\; \frac{\log x}{x^\varepsilon}\;.$$

Sedaj pošljemo $x \to \infty$ in upoštevamo

$$\lim_{x\to\infty}\frac{\vartheta(x)}{x} = 1 \;\text{ in }\; \lim_{x\to\infty}\frac{\log x}{x^\varepsilon} = 0\;.$$

Za poljuben $\varepsilon \in (0,1)$ dobimo

$$1 \;\leq\; \lim_{x\to\infty}\frac{\pi(x)}{\frac{x}{\log x}} \;\leq\; \frac{1}{1-\varepsilon},$$

se pravi, da limita obstaja in je enaka 1:

$$\boxed{\lim_{x\to\infty}\;\pi(x)\;/\;\tfrac{x}{\log x} \;=\; 1\;.} \qquad (*)$$

■

Posledica: *Če je p_n n-to praštevilo, velja*

$$\lim_{n\to\infty}\frac{n\log n}{p_n} \;=\; 1.$$

Dokaz: Logaritmirajmo limito iz izreka o gostoti praštevil

$$\lim_{x\to\infty}(\log\pi(x) + \log\log x - \log x) \;=\; 0$$

oziroma

$$\lim_{x\to\infty}\left\{\log x\left(\frac{\log\pi(x)}{\log x} + \frac{\log\log x}{\log x} - 1\right)\right\} \;=\; 0\;.$$

Ker gre $\log x \to \infty$, velja

$$\lim_{x\to\infty} \left\{ \frac{\log \pi(x)}{\log x} + \frac{\log \log x}{\log x} - 1 \right\} \;=\; 0$$

oziroma ker gre $\log \log x / \log x \to 0$, tudi

$$\lim_{x\to\infty} \frac{\log \pi(x)}{\log x} \;=\; 1.$$

Pomnožimo še z (*) in dobimo

$$\lim_{x\to\infty} \frac{\pi(x) \log \pi(x)}{x} \;=\; 1,$$

kar pa je že želena limita, če vzamemo $x = p_n$ oziroma $\pi(x) = n$. ■