UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Raziskovalna matematika

Janez Povh

Generatorji psevdonaključnih števil

Ljubljana, 2001

Kazalo

1 Uvod

Že dvakrat sem pisal dokument o generatorjih psevdonaključnih števil. Prvič za predstavitev tega področja na tečaju kriptografije, drugič pa pred pol leta, ko sem želel zaključiti svoje obveznosti pri tem tečaju. Pri vsakem pisanju imam pred seboj daljši seznam prebrane literature, kar mi po eni strani daje občutek, da imam o čem pisati, po drugi strani pa sem v vedno večji dilemi, o čem točno pisati. Tako sem se odločil, da bom naredil vsebinsko čimbolj zaokrožen pregled rezultatov, ki se nanašajo na generatorje psevdonaključnih števil.

V začetku sem prebral precej literature, v kateri so bili ti generatorji obravnavani samo kot neke črne škatle, pri katerih je bilo predmet opazovanja le izhodno zaporedje števil, ki ga je ta generator ustvaril. V ta namen so bili razviti mnogi statistični testi, ki so izračunljivi v polinomskem času od dolžine opazovanega zaporedja, zaznavali pa naj bi, ali ima opazovano zaporedje lastnost, katera je zelo verjetno lastna zaporedjem naključnih števil. Mnogo takšnih testov je izpeljanih v [15], medtem ko je v [17] pet osnovnih testov za preizkušanje kvalitete zaporedij bitov v smislu naključnosti. Za avtoriteto na tem področju velja George Marsaglia s svojim Diehard testom. Ta test je nabor 16 statističnih testov, s katerimi je potrebno testirati zaporedje bitov, vključno s kritičnimi vrednostmi za testne statistike. Na internetu so dokumenti z razlago teh testov ([10]), prav tako pa tudi knjižnice s kodami programov za izvedbo Diehard testa ([11], [12]). Zaradi siceršnje obsežnosti tega dokumenta tega testa ne bom predstavil.

Obširnejše in z vidika kriptografije aktualnejše pa je področje, ki obravnava zaporedja števil v tesni povezavi z generatorji, ki ta zaporedja generirajo. Kriterij za presojanje naključnosti teh zaporedij je predvidljivost naslednjega člena. Zanima nas namreč, koliko se da povedati o nekem številu v zaporedju, če poznamo vse njegove predhodnike in uporabljamo vse javnosti znane informacije o generatorju, s katerim je to zaporedje ustvarjeno, ter omejeno računalniško moč (t.j. uporabljamo samo polinomske algoritme v nekem parametru od generatorja). S tem področjem so se ukvarjali skoraj vsi, ki so reševali kriptografske probleme, predstavlja pa lepo spojitev algebre in teoretičnega računalništva. Posebna smer tega področja so kriptografsko varni generatorji, ki so v kriptografiji pogosto edini uporabni generatorji. Prvo področje je sestavni del drugega. Če namreč neki statistični test pri vnaprej predpisani dovolj majhni stopnji tveganja pove, da zaporedje števil, generirano z nekim generatorjem, ni naključno, potem lahko takšen test uporabimo za algoritem, ki bo iz nekaj začetnih števil napovedal naslednje število v zaporedju z verjetnostjo, večjo kot pri navadnem ugibanju. Za primer zaporedij psevdonaključnih bitov bom to pokazal v podpoglavju 4.1.

Že ta uvod kaže, da bom govoril o naključnih in o psevdonaključnih številih. Prva so zgolj težko dosegljiv ideal, zato se bom s poglavjem 3 od njih poslovil in nato ostanek posvetil psevdonaključnim številom.

Predstavil bom samo rezultate, ki se nanašajo na zaporedja bitov, saj dobra (v smislu naključnosti) in dovolj dolga zaporedja bitov porodijo dobra zaporedja naravnih števil v poljubnem območju $[1, N]$, medtem ko obratno to ni res ([3],[15]).

2 Zakaj so zaporedja naključnih števil sploh pomembna?

Vedno, ko želimo ustvariti negotovost, slučajnost, lahko za to uporabimo tudi naključna števila. V nestrokovni javnosti manj znana področja uporabe takšnih zaporedij so:

(a) **Numerična matematika:** mnoge numerične metode (npr. numerično integriranje) temeljijo na uporabi zaporedij naključnih števil.

(b) **Kriptografija:** pri kriptografskih sistemih, ki temeljijo na zasebnih (DES) ali zasebnih in javnih ključih (RSA, ElGamalov kriptosistem), je zelo pomembno, da so zasebni ključi res izbrani naključno. Če je npr. 56-bitni DES-ov ključ izbran naključno, mora napadalec s požrešno metodo poskusiti povprečno 2^{55} ključev, če pa je takšen ključ izbran tako, da se z neko zahtevno (a javnosti poznano) funkcijo napihne naključno 16-bitno število v 56-bitno, potem mora napadalec pri požrešnem napadu povprečno poskusiti le 2^{15} različnih ključev. Zmožnost generiranja naključnih števil pa omogoča tudi izvedbo sistema popolne varnosti (One time pad), kjer zakodiramo čistopis s ključem enake dolžine kot je dolžina čistopisa. Če sta se pošiljatelj in prejemnik tajnopisa sposobna na skrivaj in na kratko dogovoriti, kako bosta ustvarila enako zaporedje naključnih števil, potem je takšen sistem šifriranja izvedljiv.

(c) **Verjetnostni algoritmi:** mnogi problemi so učinkovito rešljivi s pomočjo teh algoritmov. Bistvo teh algoritmov pa je v dejstvu, da imajo vgrajen "metalec kovancev", torej podalgoritem za ustvarjanje naključnih števil. S pomočjo teh števil se algoritem odloča med večimi možnostmi oz. išče rešitev nekega problema.

Naključna števila pa so v ozadju tudi na mnogih področjih, ki jih vsakodnevno srečujemo:

(a) **Statistika:** inferenčna statistika temelji na analizi slučajno izbranega vzorca, s tako pridobljenimi podatki pa ocenjuje parametre populacije. Pri tem je zelo pomembno, da je vzorec res naključen (t.j., da imajo vsi mogoči vzorci iste velikosti enako verjetnost, da so izbrani), pri manjših popoulacijah pa je še pomembno, da ima na vsakem koraku sestavljanja vzorca vsaka enota populacije enako verjetnost, da je izbrana v vzorec. Vse to lahko dosežemo tudi tako, da enote populacije oštevilčimo, nato pa uporabimo zaporedje naključnih števil, ki so vsa iz intervala $[1, N]$, kjer je N moč populacije. Če potrebujemo vzorec velikosti n, vzamemo samo prvih n števil iz tega zaporedja. V tem primeru je za uporabljeno zaporedje števil pomembno samo, da ima čim več elementov naključnosti, sploh pa nas ne zanima proces, ki takšno zaporedje ustvari. Pogosto se uporabljajo kar tabele naključnih števil.

(b) **Razvedrilo:** kockanje, igranje rulete, loto, 3x3 in gotovo še kakšna igra na srečo temeljijo na naključnih številih. V tem primeru je zelo pomemben tudi proces, ki generira takšna števila. Lahko si samo predstavljamo, kaj bi za nekoga pomenilo, če bi znal vnaprej napovedati izid lota.

Ker je obravnavanje zaporedij naključnih števil z vidika njihove (ne)predvidljivosti tesneje povezano s kriptografijo in teoretičnim računalništvom, bom naslednjih nekaj poglavij posvetil definicijam in izrekom, ki spadajo v ta sklop.

3 Generatorji naključnih bitov

Kot je razvidno iz prejšnjega poglavja, marsikje ni dovolj, da razpolagamo s poljubno dolgim, a končnim zaporedjem števil, za katere nam nekdo (ki je sicer zaupanja vreden) jamči, da je zelo dobro v smislu naključnosti. V kriptografiji, pri igrah na srečo itd. potrebujemo napravo, ki nam bo vsakič, ko bomo želeli, vrnila število, ki bo skupaj z vsemi predhodnimi tvorilo zaporedje naključnih števil in ki ga naj ne bi bil nihče sposoben napovedati z verjetnostjo, večjo kot pri ugibanju. Kvaliteta naključnosti takšnih zaporedij je tesno povezana s kvaliteto naprave, ki takšno zaporedje proizvaja.

Najprej vpeljimo definicijo generatorja naključnih bitov.

Definicija 3.1 (Generator naključnih bitov - GNB) *GNB je naprava ali algoritem, ki vrne na svojem izhodu zaporedje bitov* $s_0, s_1, \ldots s_n$, *ki zadoščajo lastnostima:*

(a) so neodvisni, kar pomeni, da je

$$P(s_{i_1} = z_{i_1}, s_{i_2} = z_{i_2}, \ldots, s_{i_k} = z_{i_k}) = P(s_{i_1} = z_{i_1})\, P(s_{i_2} = z_{i_2}) \ldots P(s_{i_k} = z_{i_k}),$$

za vsako podmnožico $\{i_1, i_2, \ldots, i_k\} \subset \{1, 2, \ldots, n\}$ *in* $\{z_{i_1}, z_{i_2}, \ldots, z_{i_k}\} \subset \{0,1\}^k$,

(b) so nepristranski:

$$P(s_i = 0) = P(s_i = 1) = \frac{1}{2}.$$

Iz zgornje definicije sledi, da je GNB absolutno nepredvidljiv. Nihče ni sposoben z verjetnostjo, večjo od $1/2$, napovedati s_i, četudi pozna vse izhodne bite $s_1, s_2, \ldots, s_{i-1}$.

Podobno kot GNB lahko definiramo tudi generator naključnih števil, ki vrne na izhodu zaporedje neodvisnih nepristranskih naravnih števil, ki so vsa na intervalu $[1, N]$ za neki $N > 1$.

Opomba: (odnos med GNB in generatorji naključnih števil) GNB lahko uporabimo tudi kot generator naključnih števil iz intervala $[1, N]$ za vsak $N > 1$. Dovolj je, če zaporedoma jemljemo po $n = \lceil \log_2(N) \rceil$ bitov in jih pretvorimo v število med 1 in $2^n - 1$. Če je to število večje od N, ga izpustimo in vzamemo naslednji blok n bitov. Zaradi neodvisnosti in nepristranskosti izhodnih bitov GNB so tako dobljena števila tudi neodvisna od predhodnih in enakomerno porazdeljena na intervalu $[1, N]$.

Prav tako lahko iz generatorja naključnih števil dobimo GNB tako, da iz vsakega števila, ki ga takšen generator vrne, vzamemo npr. samo spodnji bit (torej ostanek vsakega števila po modulu 2). Tako dobljeno dvojiško zaporedje zadošča obema zahtevama iz definicije GNB (glej [17]). ■

Torej je dovolj obravnavati samo GNB. Vendar je v praksi zelo težko narediti GNB. Potrebujemo naravni vir naključnosti in napravo, ki bo ta vir izkoriščala in pretvarjala v zaporedje bitov. V [17, stran 172] so našteti nekateri viri naključnosti:

(a) gibanje elektronov okrog jedra v atomu,

(b) termično sevanje polprevodniške diode,

(c) časi med dvema zaporednima emisijama delcev pri radioaktivnem sevanju,

(d) izhod iz mikrofona ali video kamere, itd.

V [1] je opisana naprava za generiranje zaporedja domnevno naključnih bitov. Temelji na uporabi naključnega gibanja elektronov v uporih, ki ga ojači z zaporedjem operacijskih ojačevalcev, kar se na koncu pokaže v nepredvidljivem utripanju LED diode.
Nasploh je težko narediti napravo, ki bi učinkovito izkoriščala naravni vir naključnosti in ustvarjala zaporedja naključnih bitov. Četudi takšno napravo imamo, je zanjo težko dokazati, da je res GNB. Obstaja pa še ena težava: na mnogo naprav se da vplivati s spreminjanjem zunanjih fizikalnih pogojev:

(a) Če naprava izkorišča naključnost izhoda iz mikrofona, lahko ta mikrofon postavimo v prostor, kjer ustvarjamo zvok iz ozkega frekvenčnega območja določene jakosti, in tako dosežemo, da naprava začne ustvarjati nenaključna zaporedja bitov.

(b) Pri napravi, ki izkorišča naključnost izhoda iz kamere, lahko postavimo kamero pred platno točno določene barve in s tem bistveno vplivamo na naključnost njenega izhoda, itd.

Če govorimo o GNB, radi tudi pomislimo na zaporedje metanj kovanca, ki vrne kot rezultat zaporedje bitov, intuitivno razumljeno kot naključno. Vendar je v tem primeru naključnost samo pogojna. Nekdo, ki zna ugotoviti začetne pogoje vsakega meta in tudi pogoje, ki spremljajo let kovanca po zraku, lahko z uporabo dovolj hitrega računalnika napove rezultat meta - če že ne z gotovostjo, pa vsaj z verjetnostjo, večjo kot 1/2.
Ker so GNB v obliki fizične naprave nerodni za uporabo (še posebej, če je njihov uporabnik neki računalniški program) in tudi fizično ranljivi, so mnogo bolj popularni in uporabni generatorji v obliki računalniškega programa, ki izkoriščajo izvore naključnosti v računalniku. Takšni izvori so lahko gibanje miške, spremenljivke iz operacijskega sistema, časi med dvema zaporednima pritiskoma na tipko tipkovnice,...

Pri teh GNB je prav tako več težav. Ti izvori naključnosti so slabi, na mnoge pa je mogoče tudi vplivati. Prav tako lahko oseba, ki ve, kdo in kdaj je delal z računalnikom, mnoge od teh izvorov naključnosti predvidi, s tem pa tudi izhod takšnega GNB (predvideti se da npr. spremenljivke operacijskega sistema, saj so povezane z imeni aplikacij, ki tečejo na računalniku, s trenutnim časom itd).

Vse to privede do tega, da ne govorimo več o GNB, temveč o *generatorjih psevdonaključnih bitov*, kar je vsebina naslednjih poglavij.

4 Generatorji psevdonaključnih bitov

O teh generatorjih govorimo zato, ker se zavedamo, da so GNB bolj ali manj neuresničljiv ideal ter da imamo v praksi praviloma generatorje v obliki računalniškega programa, ki imajo mnogo slabosti.

Definicija 4.1 *Generator psevdonaključnih bitov (GPNB) je deterministični algoritem, ki iz podanega povsem naključnega binarnega zaporedja dolžine k v polinomskem času od k vrne binarno zaporedje dolžine $\ell > k$, ki "izgleda"naključno. Vhodno zaporedje imenujemo seme, izhodno pa psevdonakjučno zaporedje bitov.*

Opomba: Izhodno zaporedje GPNB gotovo ni naključno, saj se na izhodu pojavi samo $2^k/2^\ell$ vseh možnih zaporedij dolžine ℓ, kar je pri velikem ℓ v primerjavi s k daleč od enakomerne porazdelitve. ■

S to definicijo zajamemo večino generatorjev, ki se uporabljajo na področjih, naštetih v poglavju 2. Pomembni zahtevi za GPNB sta polinomska časovna zahtevnost in to, da je seme res povsem naključno zaporedje bitov. Algoritem mora biti determinističen, kar pomeni, da pri istem vhodnem zaporedju vedno vrne enako izhodno zaporedje. Verjetnostni algoritmi niso uporabni, ker zahtevajo interni G(P)NB in tako se problem obravnave kvalitete osnovnega GPNB prevede na obravnavo kvalitete vgrajenega G(P)NB, ki pa ima lahko spet vgrajen G(P)NB, kar nas lahko privede do neskončne zanke.

Primer 1: Super generator.
Knuth je v [15, stran 4-5] opisal "super" generator psevdonaključnih števil, ki je imel naključno mnogo iteracij, vsaka iteracija pa je imela naključno mnogo korakov in kopico zahtevnih matematičnih operacij, koda za tak algoritem pa naj bi bila tako zahtevna, da bi se bilo zelo težko prebiti skozi njo. Toda kljub vsemu temu se je izkazalo, da je že pri prvem zagonu (pri prvem semenu) generator takoj skonvergiral k nekemu fiksnemu številu, pri naslednjem semenu pa se je začel po 7401 izhodnih številih ponavljati s periodo 3178. Nauk, ki ga Knuth ponuja ob tem primeru, je, da za generiranje naključnih števil ni dobro uporabljati naključnih generatorjev. ■

Primer 2: Linearni kongruenčni generator - LCG.
Linearni kongruenčni generator spada med najpopularnejše generatorje psevdona-

ključnih števil. Iz semena x_0 ustvari zaporedje psevdonaključnih števil v skladu z enačbo:

$$x_{n+1} = ax_n + b \pmod{m}; \quad n = 0, 1, \ldots$$

Knuth v [15] pokaže, da ima tak generator periodo maksimalne dolžine (t.j. dolžine m) v primerih, ko je število b tuje številu m, $c = a - 1$ pa je deljivo z vsakim praštevilom, ki deli m oziroma je deljivo s 4, če je m deljivo s 4.

Takšni generatorji se pogosto uporabljajo v simulacijske namene in prestanejo vse izmed petih osnovnih statističnih testov, ki jih bomo navedli. Vendar so predvidljivi. Kdorkoli pozna a, b, m, lahko iz kateregakoli izhodnega števila x_i izračuna vse naslednje. Plumstead pa je v [20] pokazal še več: nekdo, ki pozna le nekaj zaporednih števil, ki jih je generator že izračunal, lahko napove naslednjega, četudi ne pozna števil a, b, m. Takšni generatorji torej niso uporabni za generiranje zasebnih ključev v kriptografiji. Iz takšnega generatorja lahko dobimo GPNB tako, da vsamemo od vsakega izhodnega števila x_i samo spodnji bit, t.j. $x_i \pmod 2$ ali pa nekaj spodnjih bitov. Pri tem lahko pri ponesrečeno izbranih parametrih dobimo zelo nenaključno zaporedje, ki po padlo vsaj na kakšnem statističnem testu: če vzamemo za m sodo število, za b takšno liho število in za a takšno število, da so izpolnjeni pogoji iz prejšnjega odstavka, potem je pri poljubnem sodem številu x_0 izhodno zaporedje $z_0, z_1, \ldots z_\ell$, kjer je $z_k = x_k \pmod 2$, kar enako 0101010101 .... Čeprav ima prvotno zaporedje števil x_k periodo m, ima to zaporedje bitov periodo samo 2 in bi gotovo padlo na serijskem testu iz podpoglavja 6.1. ■

Primer 3: $1/P$ **generator.**
Pri danem praštevilu P (seme) in številski osnovi $b > 1$, ki je primitivno število glede na P, tak generator vrača kar decimalke števila $1/P$ v številski predstavitvi z osnovo b. Takšen generator ima periodo $P - 1$, kar je lahko zelo veliko, če je P veliko praštevilo (npr. reda velikosti 10^k, kjer je $k \geq 50$). Prav tako ima dobre statistične lastnosti, vendar pa je predvidljiv. V [1] je Susan Bassein pokazala, da je mogoče iz zaporedja tako dobljenih števil dolžine $\lceil \log_b(2P^2) \rceil$ v polinomskem času od $\log_b P$ določiti P z uporabo verižnih ulomkov. ■

Zadnja dva primera kažeta, da lahko generator dobro prestane neki nabor statističnih testov, kar pomeni, da vrača zaporedja števil, ki vsebujejo nekatere lastnosti, za katere pričakujemo, da jih imajo skoraj vsa zaporedja naključnih števil, vendar je lahko kljub temu močno predvidljiv.

V naslednjih dveh podpoglavjih, ki sta povzeti po [22, poglavje 12.2], bom pokazal, da je dejstvo, da neki generator prestane končni nabor statističnih testov, samo potreben pogoj za to, da je nepredvidljiv. Zadosten pogoj je to, da prestane vse statistične teste.

4.1 Neločljivost verjetnostnih porazdelitev

Prvi pomemben kriterij za testiranje kvalitete GPNB je odgovor na vprašanje, ali obstaja polinomski algoritem, ki z verjetnostjo, večjo od $1/2+\varepsilon$, odgovori, ali je neko zaporedje dolžine ℓ ustvarjeno z GPNB ali pa je res naključno.

Definicija 4.2 *Naj bosta p_0 in p_1 dve verjetnostni porazdelitvi na $\mathbb{Z}_2^\ell$ in*

$$A : \mathbb{Z}_2^\ell \to \{0,1\}$$

verjetnostni algoritem, katerega časovna zahtevnost je polinomska funkcija števila ℓ. Za $\varepsilon > 0$ in $j = 0,1$ definirajmo:

$$E_A(p_j) = \sum_{(z_1,\ldots,z_\ell)\in\mathbb{Z}_2^\ell} p_j((z_1,\ldots,z_\ell)) \times P\big(A = 1|(z_1,\ldots,z_\ell)\big).$$

Algoritem A je ε- ločitelj porazdelitev p_0 in p_1, če velja

$$|E_A(p_0) - E_A(p_1)| \geq \varepsilon.$$

Opombi:
1. Če gledamo A kot slučajno spremenljivko, ki slika iz nekega prostora (npr. kartezičnega produkta prostora vseh n-teric iz $\mathbb{Z}_2^\ell$ in $\mathbb{Z}_2$ -prostora rezultatov internega GNB), je $E_A(p_j)$ ravno matematično upanje A, če ga poženemo na zaporedju, ki izhaja iz porazdelitve p_j, $j \in \{0,1\}$.

2. Pri GPNB imamo dve verjetnostni porazdelitvi na $\mathbb{Z}_2^\ell$: p_0 je enakomerna porazdelitev (vsako zaporedje $(z_1,\ldots,z_\ell)$ ima verjetnost $1/2\ell$, p_1 pa je porazdelitev, porojena z GPNB, kar pomeni, da je verjetnost vsakega zaporedja $(z_1,\ldots,z_\ell)$ enaka deležu semen, pri katerih opazovani GPNB vrne to zaporedje. ■

Opazujemo algoritem A, ki za dano binarno zaporedje iz $\mathbb{Z}_2^\ell$ vrne 1, če meni, da je ustvarjeno z GPNB (torej če izhaja iz porazdelitve p_1) oz. vrne 0, če meni, da je to zaporedje ustvarjeno s pravim generatorjem naključnih bitov (da izhaja iz porazdelitve p_0). Če se matematično upanje odgovora A na zaporedjih iz porazdelitve p_0 (torej $E_A(p_0)$) razlikuje od matematičnega upanja odgovora A na zaporedjih iz porazdelitve p_1 vsaj za ε, je A ε-ski ločitelj teh dveh porazdelitev.

Definicija 4.3 *GPNB prestane vse polinomske statistične teste, če za vsak $\varepsilon > 0$ velja, da ne obstaja ε-ski ločitelj med porazdelitvijo, porojeno s tem GPNB, in enakomerno porazdelitvijo.*

4.2 Predvidljivost naslednjega bita

Obravnavanje predvidljivosti naslednjega bita je drugi možni pristop k definiciji kvalitete GPNB (izkaže se, da je ekvivalenten obravnavanju neločljivosti porazdelitev).

Za dani GPNB nas zanima, če obstaja verjetnostni algoritem, ki je sposoben iz zaporedja dolžine $i-1$, ki ga je generator že ustvaril, z uporabo vseh javnosti dostopnih podatkov o GPNB, napovedati i-ti bit z verjetnostjo, večjo[1] od $1/2+\varepsilon$.

Definicija 4.4 *Za dani GPNB, za $1 \leq i \leq \ell$ in za $\varepsilon > 0$ je algoritem B_i ε-ski napovedovalec i-tega bita, če je časovna zahtevnost tega algoritma polinomska funkcija od i in velja:*

$$P(\mathbf{z}_i = B_i) > \frac{1}{2} + \varepsilon.$$

V tej definiciji gledamo B_i kot slučajno spremenljivko, ki slika iz kartezičnega produkta prostora vseh možnih semen in prostora vseh možnih izidov morebitnega notranjega GNB od B_i v $\mathbb{Z}_2$, torej iz $\mathbb{Z}_2^k \times \mathbb{Z}_2$ v $\mathbb{Z}_2$. Na enak način lahko gledamo tudi $\mathbf{z}_i$, kjer vzamemo $\mathbf{z}_i(\omega, 0) = \mathbf{z}_i(\omega, 1) = GPNB_i(\omega)$ za vsak $\omega \in \mathbb{Z}_2^k$, kjer je $GPNB_i(\omega)$ enak i-temu bitu, ki ga vrne GPNB pri semenu ω.

Izrek 4.5 *Naj bo $1 \leq i \leq \ell$, $\varepsilon > 0$ in p_1 porazdelitev na $\mathbb{Z}_2^{i-1}$, porojena z GPNB. Verjetnostni algoritem B_i je ε-ski napovedovalec i-tega bita natanko takrat, ko velja:*

$$\sum_{(z_1,\ldots,z_{i-1})\in\mathbb{Z}_2^{i-1}} p_1(z_1,\ldots,z_{i-1}) \times P(B_i = \mathbf{z}_i | (z_1,\ldots,z_{i-1})) \geq \frac{1}{2} + \varepsilon.$$

Dokaz. Če označimo z $GPNB^i$ slučajno spremenljivko, definirano na istem prostoru kot B_i, ki slika v $\mathbb{Z}_2^{i-1}$, tako da poljubnemu (ω, b) priredi kar prvih $i-1$ bitov, ki jih vrne opazovani GPNB pri semenu ω, neodvisno od b, potem velja:

$$\begin{aligned}
P(\mathbf{z}_i = B_i) &= P(\{(\omega,b); \mathbf{z}_i(\omega,b) = B_i(\omega,b)\}) \\
&= P\Big[\{(\omega,b); \mathbf{z}_i(\omega,b) = B_i(\omega,b)\} \cap \Big(\\
&\qquad \bigcup_{(z_1,\ldots,z_{i-1})\in\mathbb{Z}_2^{i-1}} \{(\omega,b); GPNB^i(\omega,b) = (z_1,\ldots,z_{i-1})\}\Big)\Big] \\
&= \sum_{(z_1,\ldots,z_{i-1})\in\mathbb{Z}_2^{i-1}} P(\mathbf{z}_i = B_i | (z_1,\ldots,z_{i-1})) \times p_1((z_1,\ldots,z_{i-1}))
\end{aligned}$$

■

Definicija 4.6 *GPNB prestane test naslednjega bita, če za poljuben $\varepsilon > 0$ in poljuben $1 \leq i \leq \ell$ ne obstaja ε-ski napovedovalec i-tega bita za ta generator.*

[1] Z verjetnostjo 1/2 je sposoben napovedati naslednji bit vsakdo - gre za naključno ugibanje.

4.3 Univerzalnost testa naslednjega bita

Odgovor na vprašanje, ali je bolje preučevati varnost GPNB z vidika nepredvidljivosti naslednjega bita ali z vidika neločljivosti njegove porazdelitve od enakomerne porazdelitve, je v naslednjih dveh izrekih.

Izrek 4.7 *: Naj bo $1 \leq i \leq \ell$, $\varepsilon > 0$ in B_i ε-ski napovedovalec i-tega bita za neki GPNB . Označimo s p_1 porazdelitev na $\mathbb{Z}_2^\ell$, porojeno s tem generatorjem, s p_0 pa enakomerno porazdelitev na $\mathbb{Z}_2^\ell$. Potem obstaja ε-ski ločitelj porazdelitev p_0 in p_1.*

Dokaz: S pomočjo algoritma B_i definirajmo algoritem $A : \mathbb{Z}_2^\ell \to \{0,1\}$, opisan v naslednjem okvirčku:

Vhod: zaporedje $(z_1, \ldots, z_\ell) \in \mathbb{Z}_2^\ell$.

1. *Izračunaj* $z := B_i(z_1, \ldots, z_{i-1})$.
2. **Če** $z = z_i$, **potem**

$$A(z_0, \ldots, z_\ell) = 1,$$

sicer

$$A(z_0, \ldots, z_\ell) = 0.$$

Za ta algoritem velja:

$$A(z_0, \ldots, z_\ell) = 1 \iff B_i(z_1, \ldots, z_{i-1}) = z_i.$$

Prav tako velja, da je rezultat A neodvisen od vrednosti $z_{i+1}, \ldots, z_\ell$. Če gledamo A spet kot slučajno spremenljivko, definirano na istem prostoru kot v opombi 1 podpoglavja 4.1, z $\mathbf{z}$ označimo elemente prostora $\mathbb{Z}_2^\ell$, z $\mathbf{z}^i$ elemente prostora $\mathbb{Z}_2^i$ in z $\mathbf{z}_i$ slučajno spremenljivko kot pri izreku 4.5, potem velja:

$$
\begin{aligned}
E_A(p_1) &= \sum_{(z_1,\ldots,z_\ell)\in\mathbb{Z}_2^\ell} p_1((z_1,\ldots,z_\ell)) \times P\big(A=1|\mathbf{z}=(z_1,\ldots,z_\ell)\big) \\
&= \sum_{(z_1,\ldots,z_\ell)\in\mathbb{Z}_2^\ell} P\big(\{A=1\}\cap\{\mathbf{z}=(z_1,\ldots,z_\ell)\}\big) \\
&= \sum_{(z_1,\ldots,z_i)\in\mathbb{Z}_2^i} p_1((z_1,\ldots,z_i)) \times P\big(A=1|\mathbf{z}^i=(z_1,\ldots,z_i)\big) \\
&= \sum_{(z_1,\ldots,z_i)\in\mathbb{Z}_2^i} p_1((z_1,\ldots,z_i)) \times P\big(B_i=z_i|\mathbf{z}^i=(z_1,\ldots,z_i)\big) \\
&= \sum_{(z_1,\ldots,z_i)\in\mathbb{Z}_2^i} P\big(\{B_i=z_i\}\cap\{\mathbf{z}^{i-1}=(z_1,\ldots,z_{i-1})\}\cap\{\mathbf{z}_i=z_i\}\big) \\
&= \sum_{(z_1,\ldots,z_{i-1})\in\mathbb{Z}_2^{i-1}} p_1(z_1,\ldots,z_{i-1}) \times P\big(B_i=\mathbf{z}_i|\mathbf{z}^{i-1}=(z_1,\ldots,z_{i-1})\big) \\
&\geq \frac{1}{2}+\varepsilon.
\end{aligned}
$$

■

Po drugi strani pa bo vsak ε-ski napovedovalec i-tega bita pri pravem naključnem zaporedju pravilno napovedal naslednji bit z verjetnostjo $1/2$, kar da $E_A(p_0)=1/2$. Torej je

$$|E_A(p_1)-E_A(p_0)|\geq\varepsilon.$$

■

S tem izrekom smo pokazali, da je lastnost GPNB, da prestane test naslednjega bita, potrebni pogoj za to, da GPNB prestane vse polinomske statistične teste. Je pa tudi zadostni, kar pove naslednji izrek.

Izrek 4.8 *Naj bo algoritem A ε-ski ločitelj porazdelitev p_1 in p_0, kjer je p_1 porazdelitev, porojena z nekim GPNB, p_0 pa je enakomerna porazdelitev na $\mathbb{Z}_2^\ell$. Potem za neki $1\leq i\leq\ell$ obstaja ε/ℓ-ski napovedovalec i-tega bita za ta GPNB.*

Dokaz: Za vsak $0\leq i\leq\ell$ definirajmo q_i kot verjetnostno porazdelitev na $\mathbb{Z}_2^\ell$, pri kateri je prvih i bitov dobljenih z opazovanim GPNB, ostalih $\ell-i$ bitov pa je povsem naključnih. Torej je $p_0=q_0$ in $p_1=q_\ell$. Za A tako velja: $|E_A(q_0)-E_A(q_\ell)|\geq\varepsilon$. Po trikotniški neenakosti velja:

$$|E_A(q_0)-E_A(q_\ell)|\leq\sum_{i=1}^{\ell}|E_A(q_{i-1})-E_A(q_i)|.$$

Torej lahko najdemo takšen $1\leq i\leq\ell$, da je

$$|E_A(q_{i-1})-E_A(q_i)|\geq\frac{\varepsilon}{\ell}.$$

Brez škode za splošnost lahko privzamemo:

$$E_A(q_{i-1}) - E_A(q_i) \geq \frac{\varepsilon}{\ell}.$$

Za ta i pa lahko najdemo algoritem, za katerega bomo dokazali, da je ε/ℓ-ski napovedovalec i-tega bita. Ta algoritem je v naslednjem okvirčku:

Vhod: zaporedje $(z_1, \ldots, z_{i-1}) \in \mathbb{Z}_2^{i-1}$.

1. *Izberemo slučajno zaporedje* $(z_i, \ldots, z_\ell) \in \mathbb{Z}_2^{\ell-i+1}$.
2. *izračunamo* $z := A(z_1, \ldots, z_\ell)$.
3. *definiramo* $B_i(z_1, \ldots, z_{i-1}) := z + z_i \pmod 2$.

Ta algoritem najprej iz podanega zaporedja $(z_1, \ldots, z_{i_1})$ ustvari zaporedje v skladu s porazdelitvijo q_{i-1} (seveda, če je vhodno zaporedje ustvarjeno z GPNB). Če A v koraku 2 odgovori z 0, potem to pomeni, da po mnenju A zaporedje $(z_1, \ldots, z_\ell)$ pripada porazdelitvi q_i, sicer pa, da to zaporedje pripada porazdelitvi q_{i-1}. Ti dve porazdelitvi pa se razlikujeta le po tem, da je pri q_{i-1} i-ti bit dobljen naključno, pri q_i pa z GPNB. Z odgovorom 0 algoritem A tako pove, da je po njegovem mnenju z_i število, ki naj bi ga vrnil v i-tem koraku GPNB. Odgovor 1 pa pomeni, da A meni, da GPNB na i-tem koraku ne bo vrnil z_i, temveč $1 - z_i$.

Oglejmo si sedaj verjetnost, da z zgornjim algoritmom pravilno napovemo i-ti bit. Če A pri danem vhodnem zaporedju odgovori z 0, potem je verjetnost pravilne napovedi i-tega bita enaka

$$p_1(z_i|(z_1, \ldots, z_{i-1})), \tag{4.1}$$

če pa odgovori z 1, potem 4.1 pomeni verjetnost napačne napovedi, torej je v tem primeru verjetnost pravilne napovedi kar

$$1 - p_1(z_i|(z_1, \ldots, z_{i-1})). \tag{4.2}$$

V nadaljevanju bo $\mathbf{z}$ pomenil $(z_1, \ldots, z_\ell)$, $\mathbf{z}_i$ pa i-ti bit na izhodu GPNB, gledan kot slučajna spremenljivka. Tako lahko izračunamo:

$$
\begin{aligned}
P(\mathbf{z}_i = B_i) &= \sum_{(z_1,\ldots,z_{i-1})\in\mathbb{Z}_2^{i-1}} P(\mathbf{z}_i = B_i|(z_1,\ldots,z_{i-1}))p_1(z_1,\ldots,z_{i-1}) \\
&= \sum_{\mathbf{z}\in\mathbb{Z}_2^\ell} P(\mathbf{z}_i = B_i|(z_1,\ldots,z_{i-1}))q_{i-1}(\mathbf{z}) \\
&= \sum_{\mathbf{z}\in\mathbb{Z}_2^\ell} \Big[P(A=0|\mathbf{z})\; p_1(z_i|(z_1,\ldots,z_{i-1})) + \\
&\quad + P(A=1|\mathbf{z})(1 - p_1(z_i|(z_1,\ldots,z_{i-1}))\Big] q_{i-1}(\mathbf{z}) \\
&= \sum_{\mathbf{z}\in\mathbb{Z}_2^\ell} \frac{q_i(\mathbf{z})}{2}\; P(A=0|\mathbf{z}) + \sum_{\mathbf{z}\in\mathbb{Z}_2^\ell} q_{i-1}(\mathbf{z})\; P(A=1|\mathbf{z}) \\
&\quad - \sum_{\mathbf{z}\in\mathbb{Z}_2^\ell} \frac{q_i(\mathbf{z})}{2}\; P(A=1|\mathbf{z}) \\
&= \frac{1 - E_A(q_i)}{2} + E_A(q_{i-1)}) - \frac{E_A(q_i)}{2} \\
&= \frac{1}{2} + E_A(q_{i-1}) - E_A(q_i) \\
&\geq \frac{1}{2} + \frac{\varepsilon}{\ell}, \qquad (4.3)
\end{aligned}
$$

kar pa je ravno iskani rezultat. Dokazati še moramo, da je

$$
q_{i-1}(\mathbf{z})\; p_1(z_i|(z_1,\ldots,z_{i-1})) = \frac{q_i(\mathbf{z})}{2}. \qquad (4.4)
$$

To se vidi z upoštevanjem definicij pravila za računanje pogojnih verjetnosti:

$$
\begin{aligned}
& q_{i-1}(\mathbf{z})\; p_1(z_i|(z_1,\ldots,z_{i-1})) \\
&= \; q_{i-1}(z_1,\ldots,z_{i-1})\; \frac{1}{2^{\ell-i+1}}\; p_1(z_i|(z_1,\ldots,z_{i-1})) \\
&= \; \frac{1}{2^{\ell-i+1}}\; q_i(z_1,\ldots,z_i) \\
&= \; \frac{q_i(\mathbf{z})}{2} \qquad (4.5)
\end{aligned}
$$

■

Iz teh dveh izrekov torej sledi, da je dovolj preučevati posamezni GPNB samo z enega vidika: bodisi z vidika predvidljivosti naslednjega bita bodisi z vidika ločljivosti zaporedij, ki jih generira, od zaporedij resnično naključnih bitov. Pri mnogih GPNB je težko dokazati, da prestanejo test naslednjega bita, lahko pa se pokaže, da zaporedja, ki jih generirajo, prestanejo neki nabor statističnih testov. Če je ta nabor dovolj širok, se privzame, da prestanejo takšni GPNB tudi test naslednjega bita.

5 Kriptografsko varni generatorji psevdonaključnih bitov (KVGPNB)

V kriptografiji je varnost celotnega šifrirnega sistema, identifikacijskega protokola itd. pogojena z varnostjo najšibkejšega člena. Torej mora biti tudi GPNB, ki ga morebiti uporablja tak sistem, vsaj toliko varen, kot glavni del sistema. Ker varnost velike večine šifrirnih sistemov (RSA, ElGamalov,...) temelji na nekaterih (nedokazanih) privzetkih iz teoretičnega računalništva, potem je zelo naravno tudi kvaliteto GPNB graditi na teh istih predpostavkah.

Definicija 5.1 *GPNB, ki prestane test naslednjega bita (četudi samo na osnovi neke verjetne, a nedokazane matematične predpostavke), se imenuje kriptografsko varen GPNB.*

5.1 Težki problemi

V tem podpoglavju bom opisal tri izmed problemov, za katere velja (nedokazana) predpostavka, da so težki, t.j., da niso rešljivi v polinomskem času.

5.1.1 Problem kvadratičnih ostankov

Definicija 5.2 (Jakobijev simbol) *Naj bo n liho pozitivno število in $a \in \mathbb{N}_0$. Potem je Jakobijev simbol $\left(\frac{a}{n}\right)$ definiran takole:*

$$\left(\frac{a}{n}\right) = \begin{cases} a^{\frac{n-1}{2}} \pmod{n}; & \text{če } n \in \mathbb{P}, \\ \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{c_i}; & \text{če } n = \prod_{i=1}^{k} p_i^{c_i}. \end{cases} \tag{5.6}$$

Preden definiramo problem kvadratičnih ostankov, vpeljimo še nekatere standardne oznake:

$$\mathbb{Z}_n^* = \left(\left\{x \in \{1, 2, \ldots, n-1\}; (x, n) = 1\right\}, *\right),$$

torej multiplikativna grupa vseh števil med 1 in $n-1$, ki so tuja n. Prav tako definiramo multiplikativno grupo kvadratičnih ostankov po modulu n:

$$\mathbb{Z}_n^{*2} = \left(\left\{x \in \mathbb{Z}_n^*; \exists u \in \mathbb{Z}_n^* : \; x = u^2 \pmod{n}\right\}, *\right).$$

S tema oznakama lahko definiramo:

Definicija 5.3 (Problem kvadratičnih ostankov)

Podano: *Naj bo $n = pq$, kjer sta p in q neznani praštevili, in $x \in \mathbb{Z}_n^*$, tak da velja: $\left(\frac{x}{n}\right) = 1$.*

Problem: *Ali je $x \in \mathbb{Z}_n^{*2}$?*

Za ta problem velja predpostavka, da je težak, kar pomeni, da najbrž ne obstaja algoritem, katerega (pričakovana) časovna zahtevnost bi bila polinomska funkcija n, ki bi znal rešiti ta problem z verjetnostjo, večjo od $1/2 + \varepsilon$, za neki $\varepsilon > 0$. Ta predpostavka velja tudi, če za števili p in q, katerih produkt je n, vemo: $p \equiv q \equiv 3 \pmod 4$ (glej [22]). Če pa v tem primeru poznamo p in q, potem je ta problem enostavno rešljiv (glej podpoglavje 5.2).

5.1.2 Problem diskretnega logaritma

Če imamo neko praštevilo p, potem je znano dejstvo iz algebre, da je multiplikativna grupa $\mathbb{Z}_p^*$ ciklična, torej, da ima vsaj en generator (glej [22, stran 123]). Naj bo torej g neki generator grupe $\mathbb{Z}_p^*$. Potem za vsak $x \in \mathbb{Z}_p^*$ velja, da obstaja enoličen $y \in \{1, 2, \ldots, p-1\}$, da velja $x = g^y \pmod p$. Takšen y imenujemo *diskretni logaritem x pri osnovi g.*

Definicija 5.4 (Problem diskretnega logaritma)

Podano: *Naj bo $p \in \mathbb{P}$, g generator grupe $\mathbb{Z}_p^*$ in $x \in \mathbb{Z}_p^*$.*

Problem: *Izračunaj diskretni logaritem x pri osnovi g.*

Za ta problem prav tako velja predpostavka, da je težak, kar pomeni, da ne obstaja polinomski algoritem v p, ki bi ga rešil. Obstajajo sicer algoritmi, ki v posebnih primerih vseeno lahko učinkovito rešijo ta problem, vendar se jih da onesposobiti, če se izbere število p tako, da je med delitelji števila $p-1$ vsaj eno zelo veliko praštevilo. Več o tem je v [22, poglavje 5].

5.1.3 Problem glavnega korena

Na problemu diskretnega logaritma temelji problem glavnega korena, ki je povzet po [3]. Če imamo neki kvadratični ostanek $x \in \mathbb{Z}_p^{*2}$, kjer je $p \in \mathbb{P}$ in g generator $\mathbb{Z}_p^*$, potem velja, da je $x \equiv g^{2s} \pmod p$. Ta x ima točno dva kvadratna korena, ki ju označimo z x_1 in x_2. Vemo, da velja: $\{x_1, x_2\} = \{g^s, g^{s+(p-1)/2}\}$. Tistega izmed njiju, ki je oblike g^s, imenujemo glavni koren od x. Oba kvadratna korena se da učinkovito

izračunati, ne da bi poznali s, zelo težko pa je ugotoviti, kateri izmed njiju je glavni. Ravno to pa je *problem glavnega korena.*

Definicija 5.5 (Problem glavnega korena)

Podano: *Naj bo* $p \in \mathbb{P}$, g *generator grupe* $\mathbb{Z}_p^*$, $x \in \mathbb{Z}_p^*$.

Problem: *Ali je* x *glavni koren od* $x^2 \pmod{p}$*?*

Izrek 5.6 (povzeto po [3]) *Naj bo* $\varepsilon > 0$. *Če obstaja orakelj (označimo ga z* O_{GK}*), ki za vsak* $p \in \mathbb{P}$ *in za vsak* g *generator* $\mathbb{Z}_p^*$ *reši problem glavnega korena za vsak* $x \in \mathbb{Z}_p^*$, *potem obstaja verjetnostni algoritem, katerega pričakovana časovna zahtevnost je polinomska funkcija od* p, *ki reši problem diskretnega logaritma.*

■

Torej je problem glavnega korena vsaj tako težak kot problem diskretnega logaritma. Ker je ta po predpostavki težak, je torej tudi problem glavnega korena težak.

5.2 BBS generator

Ta generator je dobil ime po avtorjih L. Blumu, M. Blumu in M. Shubu, ki so ga predstavili v članku [2]. Njegova varnost temelji na privzeti domnevi, da je problem kvadratičnih ostankov težak problem. Opisan je v naslednjem okvirčku:

Vhod: k bitno naravno število N, ki je produkt dveh tujih $k/2$ bitnih praštevil P in Q, za kateri velja:

$$P \equiv Q \equiv 3 \pmod{4}.$$

1. *Izberemo seme, ki je slučajno število* $x_0 \in \mathbb{Z}_N^{*2}$.
2. **Za** $k = 1, 2, \ldots \ell$ *izračunaj tak:* $x_k \in \mathbb{Z}_N^{*2}$, *da velja:*

$$x_k^2 \equiv x_{k-1} \pmod{N}.$$

3. **Vrni:** $z_0, z_1, \ldots, z_\ell$, *kjer je*

$$z_k = x_k \pmod{2}.$$

O časovni zahtevnosti tega GPNB govori naslednji izrek:

Izrek 5.7 *Če nekdo pozna faktorja P in Q, katerih produkt je N, potem lahko iz danega x_k učinkovito izračuna x_{k+1}.*

Dokaz:

1. *Velja:* $x \in \mathbb{Z}_N^* \iff x \in \mathbb{Z}_P^*$ *in* $x \in \mathbb{Z}_Q^*$.

   ($\Rightarrow$) : Ta smer je očitna - potrebna je samo redukcija po modulu P oz. Q.

   ($\Leftarrow$) : Naj bo $x \equiv y^2 \pmod{P}$ in $x \equiv z^2 \pmod{Q}$. Ker sta P in Q tuji si praštevili, obstajata taki števili p in q, da je $pP \equiv 1 \pmod{Q}$ in $qQ \equiv 1 \pmod{P}$. Potem pa velja, da je

   $$(\pm zpP \pm yqQ)^2 \equiv \begin{cases} y^2 & (\mathrm{mod}\ P), \\ z^2 & (\mathrm{mod}\ Q). \end{cases}$$

   Torej je $(\pm zpP \pm yqQ)^2 \equiv x \pmod{N}$. Prav tako velja, da so ti štirje elementi (namreč $\pm zpP \pm yqQ$) tudi edini taki v $\mathbb{Z}_N^*$, da je njihov kvadrat kongruenten številu x po modulu N.

2. *Če* $x \in \mathbb{Z}_N^{*2}$ *in* $P \equiv Q \equiv 3 \pmod{4}$, *potem za* $y = x^{\frac{P+1}{4}} \pmod{P}$ *in* $z = x^{\frac{Q+1}{4}} \pmod{Q}$ *velja:* $y^2 \equiv x \pmod{P}$ *in* $z^2 \equiv x \pmod{Q}$.

   Ker je $x \in \mathbb{Z}_N^{*2}$, velja tudi $x \in \mathbb{Z}_P^{*2}$ ter $x \in \mathbb{Z}_Q^{*2}$. Iz teorije števil velja, da je neki $k \in \mathbb{Z}_P^{*2} \iff k^{\frac{P-1}{2}} \equiv 1 \pmod{P}$. Torej je $y^2 = x^{\frac{P-1}{2}} x \equiv x \pmod{P}$ ter $z^2 = x^{\frac{Q-1}{2}} x \equiv x \pmod{Q}$.

3. *Če sta* y *in* z *taka kot v točki 2, potem za* $u = qQy + pPz \pmod{N}$ *in* $v = qQy - pPv \pmod{N}$ *(*p *in* q *sta dobljena kot v točki 1) velja:*

   $$u^2 \equiv v^2 \equiv x \pmod{N}.$$

   Natanko eden od $\pm u$, $\pm v$ *je v* $\mathbb{Z}_N^{*2}$. Ker je $P \equiv Q \equiv 3 \pmod{4}$, je točno eden od $\pm u$ v $\mathbb{Z}_P^{*2}$, prav tako tudi v $\mathbb{Z}_Q^{*2}$. Enako velja za $\pm v$. Torej lahko ta štiri števila razporedimo v naslednjo tabelo tako, da bo v vsakem polju točno eden:

	element $\mathbb{Z}_P^{*2}$	element $\mathbb{Z}_Q^{*2}$
element $\mathbb{Z}_P^{*2}$		
element $\mathbb{Z}_Q^{*2}$		

   Tisto število, ki je levo zgoraj, je iskani kvadratni koren od x iz $\mathbb{Z}_N^{*2}$.

Za izračun x_{k+1} iz x_k je torej potrebno izračunati p in q, za kar je potrebno izvesti en Evklidov algoritem, ki zahteva $O(\log N)$ množenj. Za izračun y in z je potrebno $O(\log^2 N)$ množenj, za preizkus, kateri od $\pm v, \pm u$ je v $\mathbb{Z}_P^{*2}$ in hkrati v $\mathbb{Z}_Q^{*2}$, pa je potrebno prav tako izvesti po $O(\log^2 N)$ množenj. ■

Izrek 5.8 *Če obstaja za BBS generator ε-ski napovedovalec naslednjega bita, potem obstaja verjetnostni algoritem, ki reši problem kvadratičnih ostankov z napako, manjšo od δ, za vsak $\delta > 0$.*

Dokaz: glej [22]. ■

Posledica 5.9 *Ker je problem kvadratičnih ostankov ocenjen kot težak problem, iz tega sledi, da za BBS generator ne obstaja ε-ski napovedovalec naslednjega bita.*

Opomba: Po izreku (4.8) ta BBS generator prestane tudi vse statistične teste. ■

5.3 Zadostni pogoji za konstrukcijo KVGPNB

M. Blum in S. Micali sta v [3] dokazala, da lahko zgradimo KVGPNB, če imamo dobro enosmerno funkcijo (t.j. takšno funkcijo, katere obrat je težko izračunati), kombinirano z neko drugo funkcijo, ki je izračunljiva v polinomskem času. Njune rezultate bom prevedel v jezik tega dokumenta, tako da jih bom rahlo poenostavil.

Definicija 5.10 *Naj bo D množica z $2n$ elementi in $P : D \to \{0,1\}$ funkcija, ki ima vrednost 1 na točno n elementih D. Takšno funkcijo imenujemo predikat.*

Ideja, ki tiči v ozadju celega tega podpoglavja, je sledeča: če bi znali naključno izbirati elemente $x_i \in D$, tako da bi na vsakem koraku imeli vsi elementi D verjetnost, da so izbrani, enako $1/2n$, potem bi bilo zaporedje $b_0, b_1, \ldots b_\ell$, dobljeno po predpisu $b_i = P(x_i)$, kar zaporedje naključnih števil. Težava te ideje je v tem, da za naključno izbiro elementa $x_i \in D$ potrebujemo $\lceil \log_2(2n) \rceil$ naklučnih bitov, z njihovo pomočjo pa ustvarimo le enega, namreč b_i.

Torej ta ideja v osnovni obliki ni uporabna. Obstaja pa njeno nadaljevanje: samo prvi element $x_i \in D$ bi izbrali naključno, ostale pa bi izračunali z neko polinomsko funkcijo f. Vendar vsaka funkcija ni dobra. Če vzamemo npr. za D množico števil $\{1, 2, \ldots, 2n\}$, za predikat funkcijo $P(x) = x \pmod 2$ in za f tole funkcijo: $f(x) = x + 2 \pmod{2n}$, potem bo izhodno binarno zaporedje sestavljeno bodisi iz samih ničel bodisi iz samih enk, kar pa nikakor ni naključno zaporedje.

Definicija 5.11 *Naj bo k naravno število. P je množica predikatov, če je $P = \{P_i : D_i \to \{0,1\}; i \in S_k\}$, kjer je S_k neka podmnožica števil, ki se izražajo s točno k-biti, D_i neka podmnožica sode moči, sestavljena iz števil, ki se izražajo z največ k biti in P_i predikat.*

Primer 4: Primer množice predikatov.
Za k naj bo S_k množica vsek k bitnih števil n, ki so produkt dveh tujih si $k/2$ bitnih praštevil p in q. Za $n \in S_k$ definiramo D_n kot množico tistih števil $x \in \mathbb{Z}_n^*$, za katera velja:

$$(\frac{x}{n}) = 1.$$

Če definiramo za vsak $n \in S_k$ predikat $P_n : D_n \to \{0,1\}$ po pravilu:

$$P_n(x) = 1 \iff x \in \mathbb{Z}_n^{*2},$$

potem je $P = \{P_i\}$ množica predikatov. ■

Definicija 5.12 *Množica predikatov P je dopustna, če obstajata takšni pozitivni konstanti c_1 in c_2 ter verjetnostni algoritem A, ki se, če mu damo na vhod število k, ustavi po k^{c_1} korakih, vrne pa "?" z verjetnostjo $1/2^{c_2 k}$, sicer pa element iz $I_k = \{(i,x); i \in S_k, x \in D_i\}$ z enakomerno porazdelitvijo.*

Za množico predikatov je pomembno, da je dopustna, saj v tem primeru lahko učinkovito za vsak k izberemo najprej $i \in S_k$, za ta i pa še en element iz D_i, tako da ja rezultat tega postopka, kadar ni enak "?", kar enakomerno porazdeljena slučajna spremenljivka z vrednostmi v $S_k \times D_i$.

Definicija 5.13 *Množica predikatov P je neocenljiva, če za vsak k in vsak $\varepsilon > 0$ ne obstaja algoritem s polinomsko časovno zahtevnostjo od k, ki bi za poljuben $(i,x) \in S_k \times D_i$ znal napovedati $P_i(x)$ z verjetnostjo, večjo od $1/2 + \varepsilon$.*

Izrek 5.14 (Zadostni pogoji za konstrukcijo KVGPNB) *Naj bo $k \in \mathbb{N}$, $P = \{P_i : D_i \to \{0,1\}; i \in S_k\}$ dopustna in neocenljiva množica predikatov in $I_k = \{(i,x); i \in S_k, x \in D_i\}$. Naj bo*

(a) $f : (i,x) \in I_k \mapsto f(i,x) \in D_i$ v polinomskem času izračunljiva funkcija,

(b) Za vsak $i \in S_k$ naj bo $f_i := f(i,.) : D_i \to D_i$ permutacija,

(c) $h : (i,x) \in I_k \mapsto P_i(f_i(x))$ naj bo v polinomskem času izračunljiv predikat.

Potem za vsak polinom Q velja, da za $\ell = Q(k)$ obstaja KVGPNB, ki iz k-bitnega semena vrne ℓ psevdonaključnih bitov.

Dokaz: Natančen dokaz je v [3], sam ga bom samo skiciral. Najprej izberemo k in polinom Q ter izračunamo $\ell = Q(k)$. Od nekod dobimo k-bitno seme s. Osnovni koraki delovanja tega KVGPNB so:

1. *S pomočjo algoritma A, ki vzame za svoj interni GNB kar seme s, dobimo $(i, x) \in I_k$.*

2. *Generiramo zaporedje*

$$T_{i,x} = x, f_i(x), f_i^2(x), \ldots, f_i^{\ell-1}(x).$$

3. *Izračunamo zaporedje $b_0, b_1, \ldots b_{\ell-1}$ po formuli*

$$b_j = P_i(f_i^j(x).$$

4. *Vrnemo zaporedje $b_{\ell-1}, b_{\ell-2}, \ldots, b_1, b_0$.*

Zaradi predpostavk izreka ta generator potrebuje polinomsko mnogo korakov za izračun izhodnega zaporedja. Zaradi dopustnosti množice predikatov algoritem v prvem koraku vrne potrebni element I_k. Kriptografska varnost tega algoritma pa je utemeljena z neocenljivostjo množice predikatov P. ■

V nadaljevanju tega podpoglavja bom opisal KVGPNB, zgrajen v skladu z izrekom 5.14.

Primer 5: KVGPNB, temelječ na problemu glavnega korena.
Ta KVGPNB temelji na lemi iz [3], ki pove, da obstaja (verjetnostni) algoritem, ki za vsak $k \in \mathbb{N}$ v pričakovanem polinomskem času od k vrne naravno število n skupaj z njegovo praštevilsko faktorizacijo, pri tem pa so števila, ki jih vrača pri istem k, enakomerno porazdeljena na množici $\{1, 2, \ldots, 2^k - 1\}$. Za razumevanje opisa delovanja tega KVGPNB, ki je v naslednjem okvirčku, je potrebno definirati še množico S_{2k} pri poljubnem številu k. To je množica vseh $2k$ bitnih števil, ki imajo vodilni bit 1, prvih k bitov sestavlja neko praštevilo p, preostalih k bitov pa sestavlja generator grupe $\mathbb{Z}_p^*$. Torej lahko vsak element $i \in S_{2k}$ zapišemo v obliki $i = p \sim g$, kjer znak $\sim$ pomeni stik dveh števil

Vhod: naravno število k, $\ell = Q(k)$ za neki polinom Q.

1. **Izberi** $i = p \sim g \in S_{2k}$, *tako da je* i *enakomerno porazdeljen na* S_{2k}.
2. **Določi** $D_i := \mathbb{Z}_p^*$ *in* $P_i(x) = 1$ *natanko takrat, ko je* x *glavni koren od* $x^2 \pmod p$.
3. **Izberi** $x \in D_i$, *tako da bo* x *enakomerno porazdeljen na* D_i.
4. **Za** $j = 0, 1, 2, \ldots, \ell - 1$ *generiraj zaporedje* x_j *po predpisu:*

$$x_j = f_i^j(x),$$

   kjer je $f_i(x) = g^x \pmod p$.

5. **Vrni** *zaporedje* $b_0, b_1, \ldots, b_{\ell-1}$, *kjer je*

$$b_j = P_i(x_{\ell-1-j}).$$

S pomočjo leme, opisane v prejšnjem odstavku, se da pokazati, da je mogoče izvesti takšni izbiri, kot sta opisani v korakih 1 in 3 (t.j., da je množica predikatov P dopustna). Iz podpoglavja 5.1.3 pa sledi, da je množica predikatov P tudi neocenljiva. Ker je g iz koraka 1 generator $\mathbb{Z}_p^*$, je funkcija $f_i : D_i \to D_i$ permutacija. Množica predikatov zadošča tudi pogoju 3 iz izreka 5.14, saj za vsak $x \in D_i$ velja:

$$P_i(f_i(x)) = P_i(g^x \pmod p) = 1 \iff 1 \leq x \leq \frac{p-1}{2}.$$

■

6 Statistični testi

V praski je težko imeti oboje: GPNB, ki bi bil hiter in bi hkrati prestal tudi vse statistične teste. Kriptografsko varni GPNB so praviloma počasni, saj izvajajo precej operacij nad velikimi števili, zato se v praksi, kjer kriptografska varnost ni nujno zahtevana, teži k hitrim GPNB, ki sicer niso kriptografsko varni, vendar vseeno generirajo zaporedja z vsemi pomembnimi lastnostmi, sicer lastnimi naključnim zaporedjem. Kvaliteto takšnih GNPB preverjamo z omejenim naborom statističnih testov.

Različni statistični testi preverjajo različne lastnosti, ki so zelo verjetne pri naključnih zaporedjih in se morajo v primeru, da je testirano zaporedje ustvarjeno s kvalitetnim GPNB, z veliko verjetnostjo pokazati tudi pri njem. Rezultat testov ni

dokončen, ampak kvečjemu verjetnostni. Če namreč izhodno zaporedje nekega GPNB prestane dani test, pomeni samo, da ta GPNB ne moremo zavrniti kot slab ter ga lahko testiramo dalje z naslednjimi testi. Če prestane vse teste (v praksi nek nabor testov), potem tak GPNB lahko vzamemo kot generator zaporedij, ki imajo z veliko verjetnostjo testirane lastnosti.

6.1 Pet osnovnih testov

Teh pet osnovnih testov (povzeti so po [17]) preverja, če so v zaporedju, ustvarjenim z nekim GPNB, prisotne najbolj tipične lastnosti, za katere lahko z veliko verjetnostjo domnevamo, da jih vsebuje vsako zaporedje, ustvarjeno z dovolj kvalitetnim GPNB.

Naj $s = s_0, s_1, \ldots, s_{n-1}$ označuje binarno zaporedje dolžine n, dobljeno z nekim GPNB. V vseh teh testih preverjamo alternativno domnevo H_1, da to zaporedje ni naključno. Zato testiramo ničelno domnevo H_0, ki trdi, da zaporedje je naključno. Če lahko z uporabo ustreznih statistik pri nekem testu zavrnemo H_0 s predpisano stopnjo tveganja, potem lahko sprejmemo domnevo H_1, kar pomeni, da lahko zaporedje s s predpisanim tveganjem zavrnemo kot nenaključno, s tem pa lahko kot slabega zavrnemo tudi GPNB, ki je ustvaril to zaporedje.

6.1.1 Frekvenčni test

S tem testom preverimo, ali ima s približno enako število enk in ničel. Naj označuje n_0 število ničel in n_1 število enk v s. Za test uporabimo statistiko

$$X_1 = \frac{(n_0 - n_1)^2}{n}, \tag{6.7}$$

ki je porazdeljena približno s porazdelitvijo χ^2 z eno stopinjo prostosti, če je le n vsaj 10. To število 10 je zato, da se zadosti znanemu pogoju pri χ^2 testu kot prilagoditvenemu testu: vse hipotetične frekvence morajo biti vsaj 5. Sicer se v praksi priporoča, da je $n \geq 10000$. Ta test je posebni primer poker testa, zato ga nisem izpeljal.

6.1.2 Serijski test

Ta test preverja lastnost naključnih števil, da je število podzaporedij oblike 00, 01, 10, 11 približno enako. Naj n_0 označuje število ničel, n_1 pa število enic v s. Nadalje naj n_{00} označuje število podzaporedij oblike 00 v s, n_{01} število podzaporedij oblike 01 v s, ter podobno tudi n_{10} in n_{11}. Ker se ta podzaporedja dolžine 2 prekrivajo, velja: $n_{00} + n_{01} + n_{10} + n_{11} = n - 1$. Za test uporabimo statistiko:

$$X_2 = \frac{4}{n-1}(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2) - \frac{2}{n}(n_0^2 + n_1^2) + 1, \tag{6.8}$$

ki se porazdeljuje po zakonu χ^2 z 2 stopinjama prostosti, če je $n > 20$. Ta test je izpeljan v [13].

6.1.3 Poker test

Naj bosta m in k takšni pozitivni naravni števili, da velja $k = \lfloor \frac{n}{m} \rfloor \geq 5 \cdot 2^m$. Najprej razdelimo zaporedje s v k neprekrivajočih se podzaporedij dolžine m. Nadalje naj bo n_i število pojavitev podzaporedja dolžine m, ki ustreza binarnemu zapisu števila i, kjer je $(0 \leq i \leq 2^m - 1)$. Ta test preverja, če se vsa možna podzaporedja dolžine m pojavljajo približno z enako frekvenco, kar je pričakovana lastnost naključnih števil. Uporabimo statistiko

$$X_3 = \frac{2^m}{k}\Big(\sum_{i=0}^{2^m-1} n_i^2\Big) - k. \tag{6.9}$$

Ta test je posplošitev frekvenčnega testa, izpeljemo pa ga takole: treba je samo testirati, ali je slučajna spremenljivka Y, ki naj pomeni številsko vrednost binarnega podzaporedja dolžine m, enakomerno porazdeljena na teh podzaporedjih dolžine m v s. Če je s naključno (hipoteza H_0), velja za vsak odsek s dolžine m: $P(Y = i) = 1/2^m$ za vsak $0 \leq i \leq 2^m - 1$. V tem primeru je torej za vsak $0 \leq i \leq 2^m - 1$ pričakovano število podzaporedij s, ki so dolga m in enaka i, kar $k/2^m$. Odstopanja števil n_i od teh pričakovanih prekvenc izmerimo s χ^2 testom ([14, stran 118]):

$$\begin{aligned} X_3 &= \sum_{i=0}^{2^m-1} \frac{(n_i - k/2^m)^2}{k/2^m} = \frac{2^m}{k} \sum_{i=0}^{2^m-1} (n_i - k/2^m)^2 \\ &= \frac{2^m}{k}\Big(\sum_{i=0}^{2^m-1} n_i^2 - 2\sum_{i=0}^{2^m-1} \frac{n_i k}{2^m} + \sum_{i=0}^{2^m-1} \frac{k^2}{2^{2m}}\Big) \\ &= \frac{2^m}{k}\Big(\sum_{i=0}^{2^m-1} n_i^2 - \frac{k^2}{2^m}\Big) = \frac{2^m}{k}\Big(\sum_{i=0}^{2^m-1} n_i^2\Big) - k. \end{aligned} \tag{6.10}$$

X_3 se porazdeljuje po zakonu χ^2 z $2^m - 1$ stopinjami prostosti.

Opomba: Če vstavimo v statistiko X_3 vrednost $m = 1$, dobimo ravno statistiko X_1.

6.1.4 Test števila blokov ničel in blokov enic

Definicija 6.1 *V nekem binarnem zaporedju je blok ničel takšno podzaporedje, ki je sestavljeno iz samih ničel, levo in desno pa je bodisi* 1 *bodisi rob zaporedja. Na enak način je definiran tudi blok enic.*

Test omogoča ugotavljanje, če je število blokov ničel in blokov enic neke dolžine približno takšno, kot se pričakuje za naključno zaporedje. Za naključno zaporedje pa je pričakovano število blokov ničel (enako velja za bloke enic) dolžine i enako $e_i = (n - i + 3)/2^{i+2}$ (glej [17], [18]). Naj bo k največje število i, za katero je $e_i = 5$ (to je klasična zahteva χ^2 testa kot prilagoditvenega testa). Nadalje označimo z A_i število blokov ničel dolžine i v s, z B_i pa število blokov enic dolžine i v s.

Uporabimo testno statistiko:

$$X_4 = \sum_{i=1}^{k} \frac{(A_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i}, \tag{6.11}$$

ki se porazdeljuje po zakonu χ^2 z $2k-2$ stopinjami prostosti. Natančnejša izpeljava testne statistike X_4 je v [18].

6.1.5 Avtokorelacijski test

Test je pripraven za ugotavljanje morebitne korelacije med zaporedjem s in zaporedjem s^d, ki je dobljeno iz s z zamikom v levo za d mest, kjer je $1 \leq d \leq \lfloor n/2 \rfloor$. Število bitov v s, ki niso enaki istoležnim bitom v s^d, je

$$A(d) = \sum_{i=0}^{n-d-1} s_i \oplus s_{i+d},$$

kjer $\oplus$ pomeni XOR bitno operacijo. Če označimo z $z_i = s_i \oplus s_{i+d}$, je ta z_i v primeru, ko je s res naključno zaporedje (hipoteza H_0), slučajna spremenljivka, porazdeljena po Bernoullijevem zakonu s $p = 1/2$. Torej je $E(z_i) = 1/2$ in $\sigma^2(z_i) = 1/4$ za vsak i. Ker so ob veljavnosti hipoteze H_0 vsi z_i med sabo neodvisni, je po centralnem limitnem izreku ([5, stran 175]) za dovolj velik n (kar 1000 ali več vsekakor je) $A(d)$ porazdeljen približno normalno z matematičnim upanjem:

$$\mu = \sum_{i=0}^{n-d-1} E(z_i) = \frac{n-d}{2}$$

in z varianco:

$$\sigma^2 = E(A(d)^2) - E(A(d))^2 = \frac{n-d}{4}.$$

Torej ima pri konkretnem d slučajna spremenljivka

$$X_5 = \frac{2\left(A(d) - \frac{n-d}{2}\right)}{\sqrt{n-d}}. \tag{6.12}$$

približno standardizirano normalno porazdelitev. Ta X_5 uporabimo kot testno statistiko, uporabiti pa je treba dvostranski test, ker so majhne vrednosti X_5 prav toliko nepričakovane kot velike.

Primer 6: Uporaba osnovnih statističnih testov.

Preverimo z zgoraj opisanimi testi naključnost (sicer zaradi cikličnosti očitno nenaključnega) zaporedja s, ki ga dobimo, če zaporedje 11100 01100 01000 10100 11101 11100 10010 01001 ponovimo štirikrat. Dolžina s je tako 160.

(a) **Frekvenčni test:** $n_0 = 84, n_1 = 76$, vrednost statistike X_1 znaša 0.4.

(b) **Serijski test:** $n_{00} = 44, n_{01} = 40, n_{10} = 40, n_{11} = 35$, vrednost statistike $X_2 = 0.6252$.

(c) **Poker test:** vzamemo $m = 3, k = 53$. Bloki $000, 001, 010, 011, 100, 101, 110, 111$ se pojavijo s frekvencami $5, 10, 6, 4, 12, 3, 6, 7$, vrednost statistike X_3 pa znaša 9.6415.

(d) **Test blokov enic in blokov ničel:** $e_1 = 20.25, e_2 = 10.0625, e_3 = 5, k = 3$. Zaporedje vsebuje $25, 4, 5$ blokov enic dolžine $1, 2, 3$ ter $8, 10, 12$ blokov ničel dolžine $1, 2, 3$. Vrednost statistike X_4 je 31.7913.

(e) **Avtokorelacijski test:** Če je $d = 3$, je $A(3) = 35$, vrednost statistike X_5 pa znaša -6.9434.

Pri stopnji značilnosti $\alpha = 0.05$ so kritične vrednosti za X_1, X_2, X_3, X_4, X_5 zaporedoma 3.842, 5.992, 14.067, 9.488 in 1.96. Torej zaporedje s prestane prve tri teste, zadnjih dveh pa ne. Zato ga upravičeno zavrnemo, s tem pa tudi vsak GPNB, ki bi ustvaril takšno zaporedje. ■

6.2 Spektralni test

Ta test je izjemno močan test za preverjanje kvalitete Linearnih kongruenčnih generatorjev (primer 2). Ti generatorji ob primerni izbiri parametrov a, b, in m prestanejo vse teste iz prejšnjega podpoglavja in so bili vrsto let zelo popularni zaradi svoje enostavnosti in aplikativnosti. Spektralni test pa pokaže, da imajo vsaj eno izrazito neslučajno lastnost. Naj bo $\mathbf{X} = \{X_i\}$ zaporedje, dobljeno s tem generatorjem in $\mathbf{U} = \{U_i = X_i/m\}$ zaporedje realnih števil iz intervala $[0, 1)$, dobljeno iz originalnega zaporedja. Če je $\mathbf{X}$ slučajno, je tudi $\mathbf{U}$. Če iz $\mathbf{U}$ naredimo novo zaporedje $\mathbf{Z}^2 = \{Z_i^2 = (U_i, U_{i+1});\ i \geq 1\}$, potem bi pričakovali, da točke Z_i ležijo enakomerno porazdeljene znotraj območja $[0, 1) \times [0, 1)$. A ni tako. Izkaže se, da obstaja več družin vzporednih enakomerno razmaknjenih premic, ki pokrivajo vse te točke. Ker je teh točk končno mnogo, seveda vedno obstajajo take družine vzporednih enakomerno razmaknjenih premic, ki jih pokrijejo. Vendar je v primeru enakomerne porazdeljenosti točk teh premic vsaj toliko kot točk. V našem primeru pa je teh premic mnogo manj kot točk.

Če podobno kot $\mathbf{Z}^2$ definiramo tudi

$$\mathbf{Z}^t = \{Z_i^t = (U_i, U_{i+1}, \ldots U_{i+t-1});\ i \geq 1\},$$

potem se izkaže, da členi $\mathbf{Z}^t$, gledani kot točke t-dimenzionalne kocke $[0, 1)^t$, niso enakomerno porazdeljeni po tej kocki, ampak obstaja družina vzporednih enakomerno razmaknjenih $t-1$-dimenzionalnih hiperravnin, ki pokrijejo vse te točke. Tudi v tem primeru vsebujejo take družine mnogo manj ravnin kot pa je točk.

Knuth v [15, stran 90] vpelje *natančnost* ν_t kot mero za kvaliteto t-dimenzionalne porazdelitve zaporedja $\mathbf{Z}^t$, definirano z: $1/\nu_t$ je maksimalna razdalja med vzporednima hiperravninama po vseh družinah vzporednih enakomerno porazdeljenih hiperravnin, ki pokrivajo $\mathbf{Z}^t$. Zastavlja se vprašanje, kako velik je ν_t za posamezni t in za posamezni generator ter katere vrednosti ν_t kažejo, da je generator (pre)slab. Računanje ν_t se prevede na problem iskanja naslednjega minimuma:

$$\nu_t^2 = \min_{(x_1,\ldots x_t)\in\mathbb{N}^t/0^t} \left((mx_1 - ax_2 - a^2x_3 - \cdots - a^{t-1}x_t)^2 + x_2^2 + \cdots x_t^2\right) \tag{6.13}$$

V [15, stran 102] je tabela, kjer so izračunane vrednosti ν_t za LCG pri nekaterih vrednostih za m in a, omenjena je pa tudi spodnja meja za ν_t, ki jamči še sprejemljivo kvaliteto generatorja:

$$\nu_t \geq 2^{30/t}.$$

Takšen je npr. LCG pri $m = 2^{35}$ in $a = 17059465$.

7 Zaključek

O generatorjih psevdonaključnih števil bi se dalo povedati še precej več, še posebej o KVGPNB. Zanimiv problem, ki je delno že rešen, je vprašanje, kako izboljšati izkoristek KVGPNB s tem, da bi namesto enega bita vsakokrat vzeli več bitov. Eden od odgovorov na to vprašanje je tudi v [23], vendar je to še dokaj odprto področje. Prav tako je mogoče mnogo več povedati o statističnih testih, čeprav sem prepričan, da je mogoče iz poglavja 6 dovolj dobro razbrati ideje, ki so prisotne pri tovrstnem obravnavanju GPNB.

Ukvarjanje z GPNB mi je (bilo) v veliko veselje in zadovoljen sem, da sem izbral ta projekt, ne glede na to, da ga oddajam tako pozno.

Literatura

[1] BASSEIN, Susan, *A Sampler of Randomnes*, American Mathematical Monthly, 103 (1996), 483-490.

[2] BLUM, L., BLUM, M., SHUB M.: *A simple unpredictable pseudorandom number generator*, SIAM J. Comput., Vol. 15, No. 2, 1986, 364-383

[3] BLUM, Manuel, Micali, Silvio: *How to generate cryptographically strong sequences of pseudo-random bits*, SIAM J. Comput., Vol. 13, November 1984, 850-864

[4] CHAITIN, J. Gregory: *Randomnes in Arithmetic*, Scientific American, July 1988, 52-57.

[5] GRIMMETT, G.R., STIRZAKER, D.R.: *Probability and Random Processes*, second edition, Clarenton Press - Oxford, 1992, 5. poglavje.

[6] `http://valjhun.fmf.uni-lj.si/~ajurisic/tecaj1/index.html`

[7] `http://random.mat.sbg.ac.at/~cgarly/server/node3.html`

[8] `http://csep1/phy/ornl/gov/rn/node18/html`

[9] `http://webnz.com/robert/true_rng.html`

[10] `http://www.stat.com/support/cert/diehard/randnumb.out`

[11] `http://stat.fsu.edu/~geo/diehard.html`

[12] `http://www.evensen.org/marsaglia`

[13] GOOD, I. J.: *The serial test for sampling numbers and other tests for randomness* Proc. Cambr. Phil. Soc., volume 49, 1953, 276-284.

[14] Jamnik, Rajko: *Verjetnostni račun in statistika*, DMFA Slovenije, 1995, 25. poglavje.

[15] KNUTH, Donald Ervin: *The Art of Computer Programming*, third edition, Addison Wesley, Inc., 1996, 3. poglavje, ISBN 0-201-89684-2.

[16] LAGARIS, J. C.: *Pseudorandom number generators*, Proceedings of Symposia in Applied Mathematics, V. 42, 1990.

[17] MENEZES, Alfred, van OORSCHOT, Paul, VANSTONE, Scott: *Handbook of Apllied Cryptography*, CRS Press, 1996, 5. poglavje

[18] MOOD, A. M.: *The distribution theory of the runs*, The Annals of Mathematical Statistics, 11, 1940, 267-392.

[19] PLUMB, Colin: *Truly Random Numbers*, Dr. Dobb's Journal, November 1996, 113-114.

[20] PLUMSTEAD, J. B. *Inferring a sequence generated by a linear congruence*, Proc. 23rd IEEE Symposium on Foundations of Computer Science, 1982, pp. 153-159.

[21] SHAMIR, A. *On the generation of cryptographically strong pseudo-random sequences*, 8th International Colloquium on Automata, Languages, and Programming, Lecture Notes in Computer Science, 62, Springer-Verlag, New York, 1981.

[22] STINSON, Douglas: *Cryptography, Theory and Practice*, CRC Press, 1995, 12. poglavje.

[23] VAZIRANI U. V., VAZIRANI, V. V.: *Efficient and secure pseudorandom number generation*, Advances in Cryptology - Proceedings of Crypto 84 (LNCS 196), 1985, pp. 193-202.