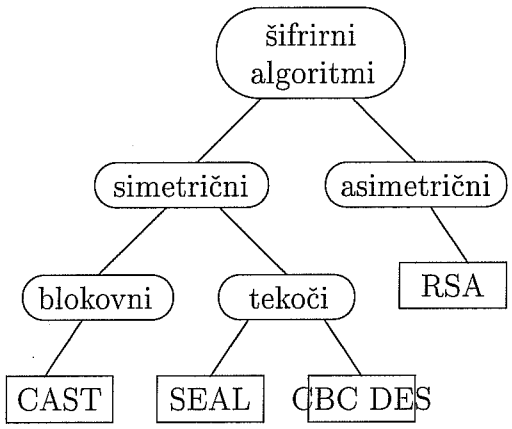
1 PSEVDONAKLJUČNA ZAPOREDJA BITOV

1.1 ŠIFRIRNI ALGORITMI

Varnost mnogih kriptografskih sistemov temelji na količinah nepredvidljivih veličin. V osnovi imamo dve vrsti kriptografskih algoritmov, simetrične in asimetrične. Prvi pa se še naprej delijo na blokovne in tekoče (stream) algoritme. Najbolj znan predstavnik blokovnih algoritmov je DES, ki uporablja 56-bitni ključ, kar pomeni, da je 2^{56} možnih ključev in da moramo v najbolj grobem napadu v povprečju preizkusiti 2^{55} ključev, da najdemo pravega. Obstaja napad z diferenčno kriptoanalizo, ki preveri le 2^{47} parov vhodnih in izhodnih blokov (64 bitnih), da ugotovi ključ. Vendar pa je ta napad spominsko tako zahteven, da v praksi ni mnogo boljši od najbolj grobe metode - preizkušanje vseh ključev. Predstavniki asimetričnih algoritmov pa temeljijo na nekem še nerešenem matematičnem problemu. Tako algoritem RSA uporablja problem faktorizacije celih števil. Za ta problem še ni najden hiter (polinomski) algoritem, ni pa niti dokazano, da obstaja. Obstajajo že algoritmi, ki na dovolj zmogljivih računalnikih faktorizirajo 500-bitna števila.



Slika 1: Klasifikacija šifrirnih algoritmov

[h]

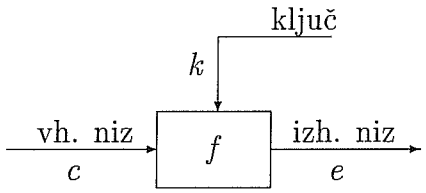
Velikost ključev mora biti danes razreda 2^{90} (90 bitni ključ). Algoritmi, ki naj bi se uporabljali v naslednjih 25 letih, pa naj bi imeli spremenljiv prostor ključev, ki sega vse do 2^{256}, saj *Moorov zakon* pravi, da se hitrost, ki jo zmore dana tehnologija, podvoji vsakih 18 mesecev.

Definicija 1.1.1. Šifrirni algoritem *f je bijektivna preslikava, ki preslika vhodni podatkovni niz c v izhodni podatkovni niz e. Preslikava uporablja tudi en parameter k, ki mu pravimo šifrirni ključ.*

$$e = f_k(c)$$

[h]

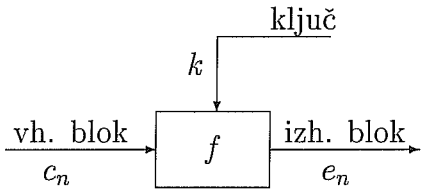
Definicija 1.1.2. Blokovni šifrirni algoritem *f je šifrirni algoritem, ki pri vsaki preslikavi preslika enako velik vhodni niz (blok) v izhodni niz (blok).*



Slika 2: Šifrirni algoritem

Če želimo šifrirati nek niz s, ga najprej razdelimo na enako velike kose $c = c_1 c_2 \dots c_N$ in potem šifriramo

$$e_n = f_k(c_n)$$



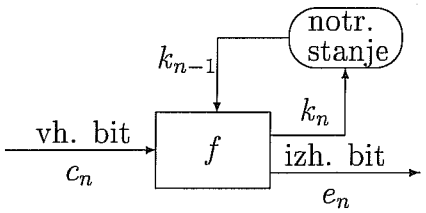
Slika 3: Blokovni šifrirni algoritem

[h]

Definicija 1.1.3. Tekoči (stream) šifrirni algoritem f *je šifrirni algoritem, ki ima notranje stanje* s*, ki se uporabi kot ključ pri preslikavi, poleg tega pa tudi vsaka preslikava tvori novo notranje stanje.*

Če želimo šifrirati nek niz s s tekočim šifrirnim algoritmom, ga najprej razdelimo na enako velike kose (običajno kar posamezne bite) $c = c_1 c_2 \dots c_N$ in potem šifriramo

$$k_0 = k$$
$$(e_n, k_n) = f_{k_{n-1}}(c_n)$$



Slika 4: Tekoči šifrirni algoritem

[h]

Osnovno, kar potrbujemo za varnost sistema, je dovolj velik prostor ključev in naključno izbran ključ, da opazovalcu sistema preprečimo, da bi s poiskušanjem odkril ključ sporočila. Vendar pa to še ni dovolj, saj moramo opazovalcu tudi preprečiti, da bi z nekimi, že poznanimi tehnikami dekriptiral sporočilo.

Algoritem CBC DES je tekoči šifrirni algoritem. Šifriranje temelji na velikem prostoru ključev in in mnogih nelinearnih preslikavah, ki so ključne za varnost sistema.

1.2 TEKOČI ŠIFRIRNI ALGORITMI

Definicija 1.2.1. Generator naključnih zaporedij bitov (GNZB) *je naprava ali algoritem, ki sestavlja zaporedja naključnih, kar pomeni* $P(\texttt{bit} = 1) = \frac{1}{2}$, *in med seboj neodvisnih bitov.*

Taki generatorji so lahko narejeni s strojno ali pa s programsko opremo. Bolj varni so prvi (šumni generator, ki npr. izkorišča termično šumenje tranzistorja), saj je tako opazovalec onemogočen, da bi opazoval oz. vplival na naključnost.

Definicija 1.2.2. Generator psevdonaključnih zaporedij bitov (GPNZB) *je deterministični*[1] *algoritem, ki kot vhodne podatke dobi resnično naključno zaporedje bitov,* seme, *dolžine k ter vrne zaporedje binarnih števil dolžine $l \gg k$, ki izgleda, kot da je naključno.*

Izhodni podatki GPNZB niso naključni, saj izmed vseh možnih zaporedij dolžine l (2^l jih je) lahko dobimo le en del teh, toliko kot je različnih semen (2^k). Seveda pa bi radi naredili tak GPNZB, da kdorkoli bi opazoval izhodno zaporedje, ne bi mogel učinkovito ločiti to zaporedje od takega, ki bi ga dal pravi GNZB.

Da dosežemo zaupanje, da je GPNZB res dober, mora iti skozi čimveč statističnih testov, ki preverjajo, če ima opazovani GPNZB take lastnosti, kot jih imajo pravi GNZB. Takih testov je lahko toliko, kot je lastnosti GNZB. Nekaj je preprostih standardnih testov (npr: ničel naj bi bilo približno toliko kot enk), nekaj pa bolj zapletenih (za naš algoritem bomo naredili Diehard teste, ki jih je razvil George Marsaglia in še nekaj testov iz programa Crypt-X, ki jih Dieherd testi ne vključujejo, večina (ostalih neskončno) pa jih še nikoli ni bila omenjena. Razlog, zakaj smo dali tako velik povdarek na Diehard teste, je njihova preglednost in zahtevnost, saj vključujejo mnogo testov, ki so med seboj neodvisni, in tudi testiranje poteka na veliki testni datoteki, kar omogoča bolj globalen vpogled v izhodno sekvenco algoritma. Da testi ne pokažejo nobenih napak na GPNZB je potreben ne pa tudi zadosten pogoj, da je GPNZB dober in s tem tudi varen.

Definicija 1.2.3. *Pravimo, da GPNZB* ustreza vsem polinomskim statističnim testom, *če noben algoritem polinomske časovne zahtevnosti ne more določiti razlike med zaporedjem generiranim s tem GPNZB in zaporedjem generiranim z GNZB z verjetnostjo večjo od* $\frac{1}{2}$.

Definicija 1.2.4. *Pravimo, da GPNZB* ustreza napovednemu testu, *če noben algoritem polinomske časovne zahtevnosti ne more na podlagi danih prvih l bitov izhodnega zaporedja s določiti naslednji, to je $l+1$., bit zaporedja s z verjetnostjo večjo od* $\frac{1}{2}$.

[1] Deterministično pomeni, da so izhodno zaporedje natanko določeno s semenom.

Trditev 1.2.5. univerzalnost napovednega testa. *GPNZB ustreza napovednemu testu natanko tedaj, ko ustreza vsem polinomskim statističnim testom.*

Definicija 1.2.6. *Če GPNZB ustreza napovednemu testu (lahko na podlagi neke zelo verjetne toda nedokazane matematične hipoteze, kot na primer kompleksnost faktorizacije celih števil*[2]*), mu rečemo* kriptografsko varen generator psevdonaključnih zaporedij bitov *- KVGPNZB.*

Tekoči algoritmi uporabljajo ključ za tvorbo začetnega notranjega stanja, vsaka preslikava pa tvori novo notranje stanje. Pri teh algoritmih je najpomembneje, da je niz zaporednih notranjih stanj in s tem tudi izhodni niz čimbolj naključen. Zato je vsak tekoči šifrirni algoritmi obenem tudi GNZB. Če naključnost izhodnega niza ni dobra, lahko s statističnimi metodami ugotovimo lastnosti vhodnega niza, kar je prvi korak k *razbitju šifrirnega algoritma* (dekripcijska funkcija, ki je časovno manj zahtevna kot preizkus vseh ključev).

Za noben šifrirni algoritem še ni dokazano ali je kriptografsko varen. Varnost algoritma je odvisna predvsem od zaupanja vanj. Zaupanje pa se gradi na raznoraznih testih. In dlje, kot je algoritem v uporabi, za bolj varnega se smatra.

1.3 OSNOVE VERJETNOSTNEGA RAČUNA

Za podrobno razumevanje ozadja statističnih testov je potrebno znanje verjetnostnega računa, katerega osnove dobimo v ?? in ??. Za razumevanje rezultata pa ponavadi še osnove statistike, ki jih bralec najde v ??. V našteti literaturi je snov podrobno predstavljena skupaj z dokazi, mi pa bomo našteli le nekaj definicij ter pomembnejše enačbe ter izreke.

Definicija 1.3.1. Celoštevilska slučajna spremenljivka X *je funkcija iz verjetnostnega prostora dogodkov* $\mathcal{D}$ *v množico celih števil* $\mathbb{Z}$*:*

$$X : \mathcal{D} \to \mathbb{Z}\,.$$

Njeno porazdelitev opišemo z verjetnostmi, s katerimi zavzame določeno vrednost. Določimo ji lahko tudi matematično upanje *(povprečje)*

$$E(X) = \sum_{n=-\infty}^{\infty} n \cdot P(X = n)$$

in varianco *(odstopanje od povprečja)*

$$var(X) = E[(X - E(X))^2]\,.$$

Definicija 1.3.2. Realna slučajna spremenljivka X *je funkcija iz verjetnostnega prostora dogodkov* $\mathcal{D}$ *v množico realnih števil* $\mathbb{R}$*:*

$$X : \mathcal{D} \to \mathbb{R}$$

[2]To je le eden izmed NP-polnih problemov, za katere še ni dokazano, da niso rešljivi v polinomskem času.

Njeno porazdelitev lahko opišemo z gostoto f_X (če obstaja), to je z verjetnostmi, s katerimi se slika na dan interval, sicer pa s porazdelitveno funkcijo F_X. Gostota *slučajne spremenljivke X je integrabilna funkcija f_X, za katero velja:*

1. $f_X(x) \geq 0$ *za vsak* $x \in \mathbb{R}$,
2. $\int_{-\infty}^{\infty} f_X(x)dx = 1$ *in*
3. *za poljubna* $a, b \in \mathbb{R}$ *je* $P(a < X \leq b) = \int_a^b f_X(x)dx$.

Spremenljivki X lahko določimo tudi matematično upanje

$$E(X) = \int_{-\infty}^{\infty} x f_X(x)dx$$

in varianco *(odstopanje od povprečja)*

$$var(X) = E[(X - E(X))^2].$$

Varianco v obeh primerih označimo tudi s σ^2 in jo lahko izračunamo kot

$$\sigma^2 = var(X) = E(X^2) - E(X)^2.$$

V statističnih testih bomo uporabljali nekaj najbolj znanih slučajnih spremenljivk, katere bomo definirali tu. Za vsako bomo nasteli njene parametre in bistvene lastnosti.

Metanje kovanca lahko opišemo z *Bernoulijevo slučajno spremenljivko* s parametrom p (ki je pri poštenem kovancu enak $\frac{1}{2}$). $X = 1$ pomeni, da je padel grb.

$$X \sim \begin{pmatrix} 1 & 0 \\ p & 1-p \end{pmatrix}$$
$$P(X = 1) = p$$
$$P(X = 0) = 1 - p$$

Če vržemo n kovancev, je število grbov *binomska slučajna spremenljivka* (celoštevilska).

$$X \sim Bin(n, p)$$
$$q = 1 - p$$
$$P(X = k) = \binom{n}{k} p^k q^{n-k}$$
$$E(X) = np$$
$$var(X) = npq$$

Trditev 1.3.3. Bernsteinovi neenačbi. *Če je S_n slučajna spremenljivka porazdeljena $Bin(n, p)$, potem za vsak $\varepsilon > 0$ velja*

$$P(\frac{S_n}{n} \geq p + \varepsilon) \leq e^{-\frac{n\varepsilon^2}{4}},$$
$$P(\frac{S_n}{n} \leq p - \varepsilon) \leq e^{-\frac{n\varepsilon^2}{4}}.$$

Lahko pa mečemo en kovanec toliko časa, da pade grb. Število potrebnih metov, vključno z zadnjim, je *geometrijska slučajna spremenljivka* (celoštevilska).

$$
\begin{aligned}
&X \sim Geom(p)\\
&q = 1 - p\\
&P(X = k) = q^{k-1}p\\
&E(X) = 1/p\\
&var(X) = q/p^2
\end{aligned}
$$

Čakamo lahko tudi do m-tega grba. Tako dobimo *negativno binomsko slučajno spremenljivko* (celoštevilska).

$$
\begin{aligned}
&X \sim NegBin(m, p)\\
&q = 1 - p\\
&P(X = k) = \binom{k-1}{m-1} p^m q^{k-m}\\
&E(X) = m/p\\
&var(X) = mq/p^2
\end{aligned}
$$

Še ena zelo znana je *Poissonova celoštevilska slučajna spremenljivka*, za katero velja:

$$
\begin{aligned}
&X \sim Po(\lambda)\\
&P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}\\
&E(X) = \lambda\\
&var(X) = \lambda
\end{aligned}
$$

Pri realnih slučajnih spremenljivkah pa so najbolj pogoste naslednje porazdelitve:

1. *enakomerna porazdelitev*

$$
\begin{aligned}
&X \sim I[0, 1]\\
&f_X(x) = \begin{cases} 1 & ; \quad 0 \leq x \leq 1 \\ 0 & ; \quad \text{sicer} \end{cases}\\
&E(X) = 1/2\\
&var(X) = 1/12
\end{aligned}
$$

2. *normalna porazdelitev*

$$
\begin{aligned}
&X \sim N(\mu, \sigma^2)\\
&f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}\\
&E(X) = \mu\\
&var(X) = \sigma^2
\end{aligned}
$$

Spremenljivko $X \sim N(0, 1)$ imenujemo *standardna normalna slučajna spremenljivka.*

3. *hi kvadrat porazdelitev*

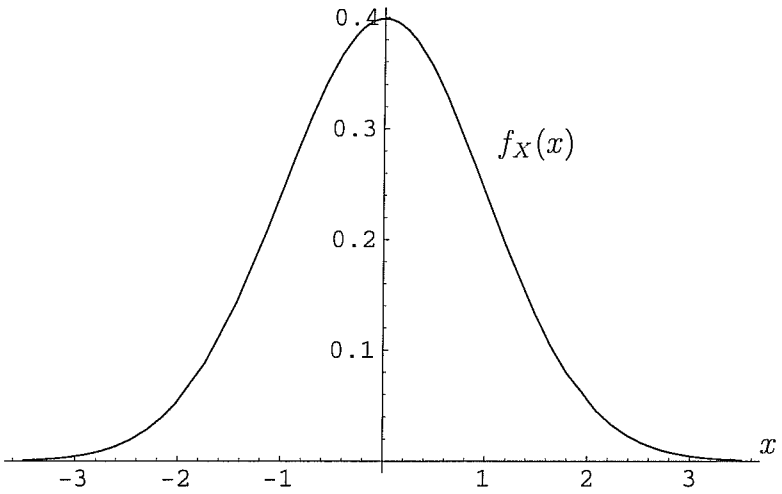
$$X \sim \chi^2(\nu)$$
$$f_X(x) = \begin{cases} \frac{1}{\Gamma(\nu/2)2^{\nu/2}} x^{\nu/2-1} e^{-x/2} & ; \quad x \geq 0 \\ 0 & ; \quad \text{sicer} \end{cases}$$
$$E(X) = \nu$$
$$var(X) = 2\nu$$

Parameter ν imenujemo *število prostostnih stopenj.*

4. *eksponentna porazdelitev*

$$X \sim exp(\lambda)$$
$$f_X(x) = \begin{cases} \lambda e^{-\lambda x} & ; \quad x \geq 0 \\ 0 & ; \quad \text{sicer} \end{cases}$$
$$E(X) = \frac{1}{\lambda}$$
$$var(X) = \frac{1}{\lambda^2}$$

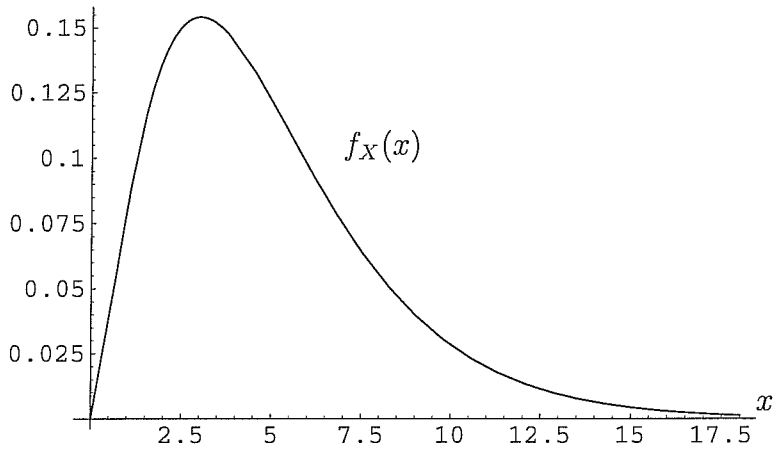
Graf gostote normalne porazdelitve je prikazan na sliki 5, graf gostote χ^2 porazdelitve pa na sliki 6. Na sliki 7 pa je prikazano, da je oblika gostote za $\nu \leq 2$ drugačna kot za $\nu > 2$. Vidi se tudi (kar je seveda res), da se ta oblika za velike ν približuje tisti za normalno porazdelitev $N(\nu, (\sqrt{2\nu})^2)$.



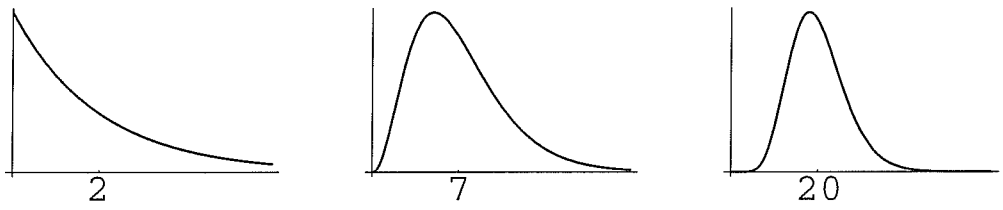
Slika 5: Graf gostote standardne normalne slučajne spremenljivke

[h]
[h]
[h]

Realno slučajno spremenljivko lahko preoblikujemo, tako da ima enakomerno porazdelitev.



Slika 6: Graf gostote $\chi^2(5)$ porazdeljene slučajne spremenljivke



Slika 7: Graf gostot $\chi^2(2)$, $\chi^2(7)$ in $\chi^2(20)$ porazdeljenih slučajnih spremenljivk

Trditev 1.3.4. *Naj bo X slučajna spremenljivka z gostoto f_X. Če določimo*

$$Y = \int_{-\infty}^{X} f_X(x)dx \tag{1}$$

ima Y enakomerno porazdelitev.

Na tak način so narejeni Diehard testi.

Trditev 1.3.5. *Če je slučajna spremenljivka X porazdeljena normalno, $X \sim N(\mu, \sigma^2)$, potem je slučajna spremenljivka $Z = \frac{X-\mu}{\sigma}$ porazdeljena $N(0,1)$. In še če je slučajna spremenljivka X porazdeljena standardno normalno, $X \sim N(0,1)$, potem je slučajna spremenljivka $Z = X^2$ porazdeljena $\chi^2(1)$.*

Trditev 1.3.6. Centralni limitni izrek. *Naj bodo slučajne spremenljivke $X_1, X_2, ..., X_n$ neodvisne in enako porazdeljene z $E(X_1) = \mu$ in $var(X_1) = \sigma^2 < \infty$. Za $S_n = X_1 + X_2 + \ldots + X_n$ velja*

$$\frac{S_n - n\mu}{\sqrt{n\sigma^2}} \overset{d}{\sim} N(0,1).$$

Trditev pove, da se za dovolj velike n porazdelitev S_n približuje porazdelitvi $N(n\mu, n\sigma^2)$. Kako dobra je ta ocena nam pove Berry-Esseenov izrek, ki pa ga tu ne bomo navedli (glej

[?]). V statistiki rečemo vektorju $(X_1, X_2, ..., X_n)$, kjer so komponente neodvisne in enako porazdeljene slučajne spremenljivke (kot nek X), *slučajni vzorec* za X.

Ko bomo delali statistične teste, bomo določili neko preslikavo, ki bo preslikala del testne datoteke v neko število. To bo naša slučajna spremenljivka. Izračunali jo bomo na več, med seboj neodvisnih, delih datoteke in tako bomo dobili slučajni vzorec, v katerem komponente seveda odstopajo od povprečja. Kako so ta odstopanja porazdeljena opisuje naslednja trditev, s pomočjo katere bomo v razdelku o preverjanju domnev povedali, kdaj test vrne pozitiven oz. negativen odgovor.

Trditev 1.3.7. *Naj bo $E_1, E_2, \ldots, E_r$ particija*[3] *množice števil v katero slika slučajna spremenljivka X. Za $i = 1, \ldots, r$ s p_i označimo verjetnost, da je vrednost spremenljivke X v množici E_i. Za spremenljivko X naredimo slučajni vzorec velikosti n. Z N_i označimo število elementov vzorca, ki so v E_i. Slučajna spremenljivka*

$$Y = \sum_{i=1}^{r} \frac{(N_i - np_i)^2}{np_i}$$

se imenuje Pearsonov hi kvadrat *in njena porazdelitev se* asimptotično približuje $\chi^2(r-1)$ *porazdelitvi, kar pomeni*

$$\lim_{n\to\infty} F_Y(y) = \int_0^y \frac{1}{\Gamma(\frac{r-1}{2})2^{(r-1)/2}} x^{(r-1)/2-1} e^{-x/2} dt$$

Slučajne spremenljivke N_i imajo matematično upanje np_i, tako da je v spremenljivki X shranjena informacija o odstopanju slučajnega vzorca od "povprečja".

1.4 PREVERJANJE DOMNEV S STATISTIČNIMI TESTI

Statistična domneva H_0 je izjava o porazdelitvi neke slučajne spremenljivke. *Preverjanje domneve (hypothesis testing)* je procedura, ki na podlagi opazovanja slučajne spremenljivke naredi sklep o sprejetju oz. zavrnitvi domneve H_0. Test zavrne domnevo, če imajo rezultati opazovanja neke ekstremne vrednosti. Zaključek testa zato ni deterministično, ampak verjetnostni.

Definicija 1.4.1. *Verjetnost zavrnitve statistične domneve H_0, ki je pravilna, imenujemo* stopnja značilnosti preverjanja domneve H_0 *in jo označimo z α.*

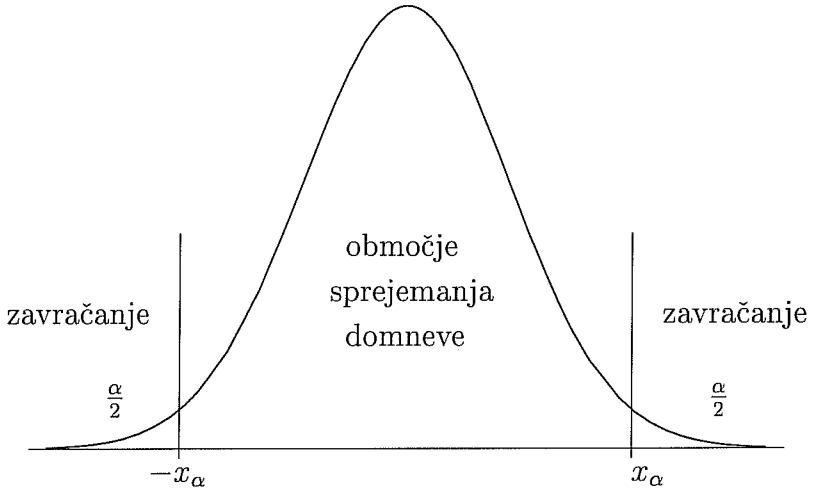
Za testirano slučajno spremenljivko X ponavadi določimo nek interval $[a, b]$, za katerega velja,

1. da je čim ožji in
2. da, če bi imel X res domnevano porazdelitev, bi bila $P(X \notin [a, b]) = \alpha$.

Poimenujemo ga *interval zaupanja.* Na sliki 8 in v tabeli 1 je prikazano, kakšen interval vzamemo pri normalni porazdelitvi, v tabeli 2 pa za χ^2 porazdelitev.

[h]

[3] množica disjunktnih nepraznih množic, ki pokrije celotno množico



Slika 8: Interval zaupanja za normalno porazdeljeno slučajno spremenljivko

α	0,2	0,1	0,05	0,02	0,01	0,005	0,002	0,001
x_α	1,2816	1,6449	1,9600	2,3263	2,5758	2,8070	3,0902	3,2905

Tabela 1: Intervali zaupanja za standardno normalno slučajno spremenljivko

V testih, ki jih bomo uporabljajl mi, bo H_0 domneva o porazdelitvi slučajne spremenljivke iz testa pri predpostavki, da je bilo zaporedje, ki ga testiramo, generirano z generatorjem naključnih zaporedij bitov. Če bo stopnja značilnosti α preverjanja domneve H_0 previsoka, potem nam bo test zavračal tudi zaporedja, ki so bila generirana z generatorjem naključnih zaporedij bitov. Tej napaki rečemo *napaka I. vrste* (primer: slika 9).

Za normalno porazdeljeno slučajno spremenljivko vzamemo interval zaupanja tak, da je sredina intervala povprečje tako porazdeljene spremenljivke. Za $N(0,1)$ porazdelitev je to $[-x_\alpha, x_\alpha]$. Tabelo 1 dobimo z numeričnim reševanjem integralske enačbe (iščemo x_α)

$$\int_{-x_\alpha}^{x_\alpha} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} = 1 - \alpha \, .$$

Za χ^2 porazdeljeno slučajno spremenljivko pa vzamemo interval zaupanja $[0, x_\alpha]$. Tako dobimo vrednosti v tabeli 2 z reševanjem integralske enačbe (iščemo x_α)

$$\int_0^{x_\alpha} \frac{1}{\Gamma(\nu/2)2^{\nu/2}} x^{\nu/2-1} e^{-x/2} = 1 - \alpha \, .$$

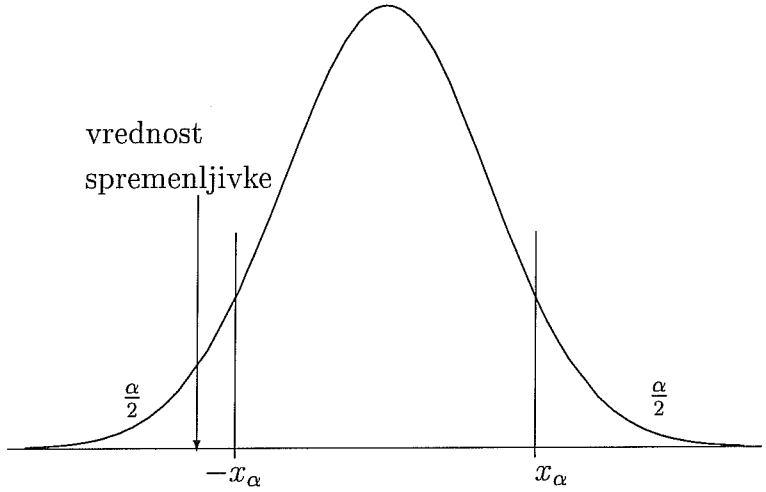
Za velike ν ta interval ni najkrajši, kar smo videli na sliki 7. Kako bi izračunali najkrajši interval je opisano v knjigi [6].

[h]

Lahko pa je stopnja značilnosti tega preverjanja premajhna, kar povzroči, da test sprejme domnevo, četudi zaporedje ni bilo generirano z generatorjem naključnih zaporedij

$\nu \backslash \alpha$	0,1	0,05	0,025	0,01	0,005	0,0025	0,001
1	2,7055	3,8415	5,0239	6,6349	7,8794	9,1406	10,8276
2	4,6052	5,9915	7,3778	9,2103	10,5966	11,9829	13,8155
3	6,2514	7,8147	9,3484	11,3449	12,8382	14,3203	16,2662
4	7,7794	9,4877	11,1433	13,2767	14,8603	16,4239	18,4668
5	9,2364	11,0705	12,8325	15,0863	16,7496	18,3856	20,5150
6	10,6446	12,5916	14,4494	16,8119	18,5476	20,2494	22,4577
7	12,0170	14,0671	16,0128	18,4753	20,2777	22,0404	24,3219
8	13,3616	15,5073	17,5345	20,0902	21,9550	23,7745	26,1245
9	14,6837	16,9190	19,0228	21,6660	23,5894	25,4625	27,8772
10	15,9872	18,3070	20,4832	23,2093	25,1882	27,1122	29,5883
11	17,2750	19,6751	21,9200	24,7250	26,7568	28,7293	31,2641
12	18,5493	21,0261	23,3367	26,2170	28,2995	30,3185	32,9095
13	19,8119	22,3620	24,7356	27,6882	29,8195	31,8831	34,5282
14	21,0641	23,6848	26,1189	29,1412	31,3193	33,4260	36,1233
15	22,3071	24,9958	27,4884	30,5779	32,8013	34,9496	37,6973
16	23,5418	26,2962	28,8454	31,9999	34,2672	36,4557	39,2524
17	24,7690	27,5871	30,1910	33,4087	35,7185	37,9461	40,7902
18	25,9894	28,8693	31,5264	34,8053	37,1565	39,4221	42,3124
19	27,2036	30,1435	32,8523	36,1909	38,5823	40,8850	43,8202
20	28,4120	31,4104	34,1696	37,5662	39,9968	42,3357	45,3147
21	29,6151	32,6706	35,4789	38,9322	41,4011	43,7751	46,7970
22	30,8133	33,9244	36,7807	40,2894	42,7957	45,2041	48,2679
23	32,0069	35,1725	38,0756	41,6384	44,1813	46,6235	49,7282
24	33,1962	36,4150	39,3641	42,9798	45,5585	48,0337	51,1786
25	34,3816	37,6525	40,6465	44,3141	46,9279	49,4354	52,6197
26	35,5632	38,8851	41,9232	45,6417	48,2899	50,8291	54,0520
27	36,7412	40,1133	43,1945	46,9629	49,6449	52,2153	55,4760
28	37,9159	41,3371	44,4608	48,2782	50,9934	53,5943	56,8923
29	39,0875	42,5570	45,7223	49,5879	52,3356	54,9666	58,3012
30	40,2560	43,7730	46,9792	50,8922	53,6720	56,3325	59,7031
31	41,4217	44,9853	48,2319	52,1914	55,0027	57,6923	61,0983

Tabela 2: Intervali zaupanja za $\chi^2(\nu)$ porazdeljeno slučajno spremenljivko
V tabeli je za dan ν in α izračunan x_α.



Slika 9: Premajhen interval zaupanja za normalno porazdeljeno slučajno spremenljivko

bitov. Taki napaki pa rečemo *napaka II. vrste.* Zato je pomembno, da določimo primeren α. V praksi lahko vzamemo α med 0,001 in 0,05.

Pri naših testiranjih bomo potrebovali naslednja statistična testa.

1. **Hi kvadrat test (Chi-square test)**
   Vrednosti slučajnega vzorca razdelimo na primerne razrede ter določimo vrednost Pearsonove hi kvadrat slučajne spremenljivke Y, kot je opisano v trditvi 1.3.7. Nato pri izbrani stopnji značilnosti preverjanja naše domneve določimo interval zaupanja oblike $[0, x_\alpha]$ (lahko kar iz tabele 2). Test lahko uporabimo le za dovolj velike n, saj trditev govori o limitni porazdelitvi. Konvergenca dejanske porazdelitve je odvisna od $p_1, \ldots, p_r$, zato ni točnega pravila, kdaj je n dovolj velik. Navadno vzamemo, da takrat ko je $np_i \geq 5$ za vse $i = 1, \ldots, r$.

2. **KS[4] test**
   ¡———————¿

1.5 PET OSNOVNIH STATISTIČNIH TESTOV ZA G(P)NZB

Razdelek opisuje pet osnovnih testov, navedenih v knjigi [1], katere se uporablja za testiranje nekega zaporedja bitov $s = s_0, s_1, \ldots, s_n$. Vsak od testov pove, ali ima zaporedje testirno lastnost, kot bi jo imela večina zaporedij, generiranih z GNZB.

1. **Monobit test (frequency test)**
   Cilj tega testa je ugotoviti, ali ima dano zaporedje s približno toliko bitov enakih nič (ničel) kot tistih enakih ena (enk). Z n_1 označimo število enk v zaporedju s. Slučajna spremenljivka n_1 je pri zaporedju s, generiranim z GNZB, binomsko porazdeljena, $Bin(n, \frac{1}{2})$, kar pomeni, da je število enk v povprečju enako $n/2$ in po

[4]Kolmogorov, Smirnov

centralnem limitnem izreku je

$$X_1 = \frac{n_1 - \frac{n}{2}}{\frac{\sqrt{n}}{2}}$$

približno normalno $N(0,1)$ porazdeljena slučajna spremenljivka. Če vzamemo $n = 20000$, je za izbran $\alpha = 0.001$ (najmilejši pogoj izmed priporočljivih[5]) dobljena še sprejemljiva vrednost za n_1 na intervalu $[9767, 10233]$. Interval smo dobili z izračunom integrala gostote spremenljivke X_1

$$P(X_1 < -a) + P(X_1 > a) = 1 - P(-a \leq X_1 \leq a) \doteq \int_{-a}^{a} \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}$$

Ta verjetnost je po predpostavki enaka 0.001 in zato numerično dobljen $a = 3,2905$. Ker se a nanaša na spremenljivko X_1 je ustrezno odstopanje b od povprečja $n/2$ za spremenljivko n_1 enako $b = 232,67$. Izračun lahko preverimo tudi z dejansko binomsko porazdelitvijo. Vemo, da je $P(n_1 = k) = \binom{n}{k}/2^n$. Da računalnik izračuna tako velike binomske simbole v realnem času, jih zapišemo analitično, $\binom{n}{k} = \frac{1}{(n-k)B(n-k,k+1)}$, kjer je B poznana beta funkcija. In tako je

$$P(n_1 < 9767) + P(n_1 > 10233) = 1 - P(9767 \leq n_1 \leq 10233) = \sum_{k=9767}^{10233} \frac{\binom{n}{k}}{2^n} = \sum_{k=9767}^{10233} \frac{\frac{1}{(n-k)B(n-k,k+1)}}{2^n} = 0,9990411309$$

Rezultat je bil pričakovan, saj smo vzeli dovolj velik n.

2. **Test parov (two-bit test, serial test)**
Cilj tega testa je ugotoviti, ali ima dano zaporedje približno enako število pojavitev podzaporedij 00, 01, 10 in 11. Naj n_0 in n_1 zaporedoma označujeta število ničel in število enk ter n_{00}, n_{01}, n_{10} in n_{11} zaporedoma število pojavitev para 00, 01, 10 in 11. Očitno je $n_{00}+n_{01}+n_{10}+n_{11} = n-1$, saj se dovolimo prekrivanje podzaporedij. Porazdelitev slučajne spremenljivke

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

se pri zaporedju, generiranim z GNZB, za dovolj velike n ($n \geq 21$) približuje $\chi^2(2)$ porazdelitvi.

3. **Poker test**
Naj bo m naravno število, tako da je $\lfloor \frac{n}{m} \rfloor \geq 5 \cdot 2^m$ in naj bo $k = \lfloor \frac{n}{m} \rfloor$. Zaporedje s razdelimo na k neprekrivajočih se delov dolžine m. Naj bo n_i število pojavitev števila i zapisanega v binarnem zapisu (z vodilnimi ničlami, npr. za $m = 5$ in $i = 14$ imamo zaporedje 01110), $0 \leq i \leq 2^m - 1$. Poker test preveri, ali se vsak i pojavi približno enakokrat, kakor to pričakujemo za pravo naključno zaporedje. Tu definiramo slučajno spremenljivko

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

[5] V [1] se priporoča α med 0.001 in 0.05

ki se pri zaporedju, generiranim z GNZB, za velike n približuje $\chi^2(2^m - 1)$ porazdelitvi. Za $m = 1$ je pri enaki stopnji značilnosti α poker test ekvivalenten monobit testu.

4. **Bločni test (runs test)**
   Zaporedje s razdelimo na *bloke* in *vrzeli*, to je na maksimalno dolga podzaporedja enk oz. ničel (11100011000100010100111010111100 bi tako razdelili na 111-000-11-000-1-000-
   Pričakovano število pojavitev bloka dolžine i v zaporedju s dožine n, generiranim z GNZB, je $e_i = (n - i + 3)/2^{i+2}$. Naj bo k enak največjemu i, za katerega je $e_i \geq 5$ (ker je e_i odvisen od n, je tudi k odvisen od n). Z B_i in V_i označimo število blokov in vrzeli v s dolžine i. Slučajna spremenljivka

$$X_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} \sum_{i=1}^{k} \frac{(V_i - e_i)^2}{e_i}$$

   se pri zaporedju, generiranim z GNZB, za velike n približuje $\chi^2(2k-2)$ porazdelitvi.

5. **Autokorelacijski test (autocorrelation test)**
   Namen testa je preveriti *korelacijo* med zaporedjem s in zaporedjem dobljenim s *premikom zaporedja* s za d mest, $1 \leq d \leq \lfloor n/2 \rfloor$. Z $A(d)$ označimo število bitov zaporedja s, ki niso enaki istoležnim bitom v premiku s. Tako je $A(d) = \sum_{i=0}^{n-d-1} s_i \oplus s_{i+d}$, kjer $\oplus$ označuje XOR (ekskluzivni ali) operator. Tu se slučajna spremenljivka

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

   za velike n ($n - d \geq 10$) približuje $N(0,1)$ porazdelitvi.

Primer 1.5.1. *Vzemimo zaporedje s dolžine $n = 160$,*

$$\begin{aligned} s \;=\; & 11100\ 01100\ 01000\ 10100\ 11101\ 11100\ 10010\ 01001 \\ & 11100\ 01100\ 01000\ 10100\ 11101\ 11100\ 10010\ 01001 \\ & 11100\ 01100\ 01000\ 10100\ 11101\ 11100\ 10010\ 01001 \\ & 11100\ 01100\ 01000\ 10100\ 11101\ 11100\ 10010\ 01001 \end{aligned}$$

in ga stestirajmo pri stopnji značilnosti 0,05. Poker test naredimo za $m = 3$ in torej štejemo pojavitve podzaporedij 000, 001, 010, 011, 100, 101, 110, 111 *na mestih 0, 3, 6, ..., 156. Podzaporedja se v zgoraj zapisanem vrstnem redu pojavijo 5, 10, 6, 4, 12, 3, 6 in 7-krat. Za bločni test si najprej naračunamo števila e_i. $e_1 = 20,25$, $e_2 = 10,06$, $e_3 = 5,00$, $e_4 = 2,48$,... Torej je $k = 3$. Preštejemo bloke in vrzeli: $B_1 = 25$, $B_2 = 4$, $B_3 = 5$, $V_1 = 8$, $V_2 = 20$ in $V_3 = 12$. Autokorelacijski test naredimo pri premiku $d = 3$.*

test	*interval zaupanja*	*vrednost X_i*	*rezultat testa*
monobit test	*[-1'9600,1'9600]*	*0'6325*	$\surd$
test parov	*[0,5'9915]*	*0'6252*	$\surd$
poker test	*[0,14'0671]*	*9'6415*	$\surd$
bločni test	*[0,9'4877]*	*31'7913*	//
autokorelacijski test	*[-1'9600,1'9600]*	*-6'9434*	//

1.6 STANDARD FIPS 140-1

Ameriški standard FIPS[6] 140-1 določa štiri teste za preverjanje naključnosti generatorjev zaporedij bitov. Najprej generiramo zaporedje s dolžine 20000 bitov in preverimo, če so slučajne spremenljivke posameznih testov za s na predpisanih intervalih zaupanja.

1. *Monobit test.* Število enic, n_1, mora biti na intervalu $[9655, 10345]$.
2. *Poker test.* Slučajna spremenljivka X_3, definirana v enačbi (2) se računa za $m = 4$ in njena vrednost mora biti na intervalu $[1'03, 57'4]$.
3. *Bločni test.* Preštejemo vse bloke in vrzeli dolžine največ 6 (bloke in vrzeli daljše od 6 bitov štejemo kar k B_6 oz. V_6). Nato preverimo, če B_i in V_i ležijo na danih intervalih:

dolžina bloka / vrzeli	interval zaupanja
1	2267 – 2733
2	1079 – 1421
3	502 – 748
4	223 – 402
5	90 – 223
6	90 – 223

4. *Test dolgih blokov.* Noben blok niti nobena vrzel ne smeta biti daljša od 33 bitov.

Če generator ne ustreza kateremu od testov, pravimo, da ne ustreza standardu. Zato pa je interval zaupanja zelo velik. Za aplikacije, ki zahtevajo visoko stopnjo varnosti, standard zahteva, da se testi izvedejo ob vsakem zagonu G(P)NZB. FIPS 140-1 dopušča, da se ti štirje testi lahko nadomestijo tudi s kakšnimi drugimi, ki so ekvivalentni tem ali pa strožji.

Za monobit in poker test lahko iz danih intervalov na preprost določimo stopnjo značilnosti testa, za prvega tako kot pri opisu pri osnovnih testih, za drugega pa je

$$\alpha = 1 - \int_{1,03}^{57,4} f_{\chi^2(15)}(x)dx\,.$$

V obeh primerih je $\alpha = 0,000001$, kar pomeni, da je napaka I. vrste s temi testi zelo redka, zato pa pogosteje pride do napake II. vrste. Razloga za izbran tako majhen α sta najbrž ta, ker poženemo teste mnogokrat in na zelo kratkih sekvencah.

1.7 DIEHARD TESTI

DIEHARD program vrši 17 testov, razvitih za potrebe testiranja generatorjev naključnosti, ki se uporabljajo pri igrah na srečo. Za testiranje potrebujemo 10 Mbytov veliko datoteko, generirano z G(P)NZB. Opis testov je vzet iz datoteke rezultatov testiranja ter člankov [11] in [12]. Imen testov ne bomo prevajali.

[6]Federal Information Processing Standards

Večina Diehard testov vrne vrednost spremenljivke p, ki naj bi bila slučajna in enakomerno porazdeljena na intervalu $[0,1)$. To je pravzaprav naša domneva, ki jo preverjamo. Ti p-ji so dobljeni iz porazdelitve slučajne spremenljivke X (ki jo računamo v posameznem testu), kot je opisano v trditvi 1.3.4. Za porazdelitev X naredimo aproksimacijo z neko realno slučajno spremenljivko. Ker je X ponavadi omejena, bo ta aproksimacija najslabša na za velike vrednosti. Tudi ne smemo biti presenečeni, če je kakšna vrednost p blizu 0 ali 1 (npr. 0,0012 ali pa 0.9983), saj je to povsem normalno ob stotinah testov, ki jih naredi program Diehard. Če je niz res slab, bo kar nekaj (šest ali več) p-jev enakih 0 ali 1 (zaokroženo na 6 decimalk).

DIEHARD testi vsebuje naslednje teste:

1. **Birthday spacings test**
   Naključno izberemo m rojstnih dni $I_1, I_2 \dots, I_m$ (poljuben dan izberemo z verjetnostjo $\frac{1}{m}$) v letu, ki ima n dni. Nato I-je uredimo naraščajoče po velikosti in tako dobljena števila po vrsti označimo z $I_{(1)}, I_{(2)} \dots, I_{(m)}$. Nato določimo dolžine presledkov med posameznimi praznovanji

   $$I_{(1)}, I_{(2)} - I_{(1)}, I_{(3)} - I_{(2)}, \dots, I_{(m)} - I_{(m-1)} .$$

   Če je

   $j = m -$ "število vrednosti, ki se v zadnjem zaporedju pojavijo natanko enkrat" ,

   potem se porazdelitev spremenljivke j asimptotično približuje[7] Poissonovi porazdelitvi $Po(\frac{m^3}{4n})$. Izkušnje kažejo, da mora biti n dovolj velik, da je ta aproksimacija dobra, recimo $n \geq 2^{18} \doteq 2,6 \cdot 10^5$. Ta test uporablja $n = 2^{24} = 1,7 \cdot 10^7$ in $m = 2^9 = 512$, tako da je $j \sim Po(2)$. Na tak način naračunamo 500 j-jev in to je vorec za hi kvadrat test, ki je opisan na strani 11. Tu bomo potek testa podrobno opisali. Tvorimo particijo množice nenegativnih celih števil $E_1 = \{0\}, E_2 = \{1\}, \dots, E_6 = \{5\}$ in $E_7 = \{6, 7, \dots\}$. Izračunamo koliko j-jev je v posamezni particijski množici in to so naše vrednosti N_i. Verjetnost, da ima spremenljivka j vrednost v množici E_i, določimo iz $Po(2)$ porazdelitve. Te verjetnosti so $p_1 = \frac{2^0}{0!}e^{-2} \doteq 0'13534, p_2 = \frac{2^1}{1!}e^{-2} \doteq 0'27067, p_3 = \frac{2^2}{2!}e^{-2} \doteq 0'27067, p_4 = \frac{2^3}{3!}e^{-2} \doteq 0'18045, p_5 = \frac{2^4}{4!}e^{-2} \doteq 0'09022, p_6 = \frac{2^5}{5!}e^{-2} \doteq 0'03609, p_7 = 1 - \sum_{i=1}^{6} p_i \doteq 0'01656$, kar pomeni, da so pričakovane vrednosti za N_i enake $500p_i$ oz. $67'668, 135'34, 135'34, 90'224, 45'112, 18'045, 8'2818$ zaporedoma. Iz teh vrednosti določimo vrednost spremenljivke $Y = \sum_{i=1}^{7} \frac{(N_i - 500p_i)^2}{500p_i}$, ki je $\chi(7-1)$ porazdeljena. Iz nje določimo vrednost p s pomočjo enačbe iz trditve 1.3.4, torej $p = \int_{-\infty}^{Y} f_{\chi(6)}(t)dt$. Na tak način bomo izračunali 9 p-jev, ki bodo vsak zase predstavljali nek delen rezultat testa. Nato pa bomo še iz teh p-jev, ki naj bi bili enakomerno porazdeljeni, naredili s KS testom končen rezultat Birthday testa.

   KSTEST

   Za dejansko izvedbo testa na testni datoteki, sestavljeni iz bitov $b_1, b_2, \dots$, pa je seveda potrebno določiti, kateri biti bodo določali posamezne rojstne dneve. Za prvi

[7] To je dokazal Janos Komlos

p (od devetih) nam bodo rojstne dneve predstavljala števila $b_{32k+1}b_{32k+2}\dots b_{32k+24}$, torej $b_1b_2\dots b_{24}, b_{33}b_{34}\dots b_{56},\dots$ Za vsakega od 500 j-jev potrebujemo $m = 512$ števil. Torej mora biti datoteka velika 8.192.000 bitov (1 Mbyte). Za drugi p vzamemo bite $b_{32k+2}b_{32k+3}\dots b_{32k+25}$ in tako naprej do devetega p, kjer potrebujemo bite $b_{32k+9}b_{32k+10}\dots b_{32k+32}$.

2. **Overlapping 5-permutation test**
Za test potrebujemo datoteko 1.000.000 32-bitnih naključnih celih števil $u_1, u_2, \dots$. Ker bomo test izvedli dvakrat, je to 8 Mbytov. V posameznem testiranju tvorimo zaporedje peteric $(u_1, u_2, u_3, u_4, u_5), (u_2, u_3, u_4, u_5, u_6), (u_3, u_4, u_5, u_6, u_7), \dots$ Vsaka od teh peteric nam predstavlja neko stanje S Markovske verige V na sledeč način:

$$S\big((x_1,x_2,x_3,x_4,x_5)\big) = \begin{cases} 1 & ;\ x_1 \le x_2 \le x_3 \le x_4 \le x_5 \\ 2 & ;\ x_1 \le x_2 \le x_3 \le x_5 < x_4 \\ 3 & ;\ x_1 \le x_2 \le x_4 < x_3 \le x_5 \\ & \vdots \\ 120 & ;\ x_5 < x_4 < x_3 < x_2 < x_1 \end{cases}$$

Vsaka od zgornjih peteric je odvisna od svoje predhodnice (od nje prevzame 4 vrednosti od petih). Sedaj lahko rečemo, da vsako od števil u_i, $i = 5, 6, 7, \dots$, določi neko stanje $S\big((u_i, u_{i-1}, u_{i-2}, u_{i-3}, u_{i-4})\big)$ verige V. Tako iz zaporedja $u_5, u_6, \dots$ dobimo zaporedje stanj, recimo $17, 4, 102, 78, \dots$. Z w_{ijk} označimo število pojavitev podzaporedja i, j, k v zaporedju stanj, s C pa *kovariančno matriko*[8] slučajnega vektorja $[w_{i,j,k}]_{i,j,k=1}^{120} \in \mathbb{Z}^{1728000}$. Z *metodo največjega verjetja (method of maximum likelihood*[9]*)* dobimo, da se porazdelitev slučajne spremenljivke

$$X = \sum_{i,j,k,r,s,t=1}^{120} (w_{ijk} - \mu_{ijk}) c^{-}_{ijk,rst} (w_{rst} - \mu_{rst})$$

asimptotično približuje $\chi^2(99)$ porazdelitvi. Z μ_{ijk} smo označili *matematično upanje vektorja*[10] $[w_{i,j,k}]$, s C^- pa katerikoli *posplošen inverz* matrike C, to je matrika C^-, za katero velja $CC^-C = C$. Če pravi inverz C^{-1} obstaja, zanj gotovo velja $CC^{-1}C = C$. Iskanje tega inverza za tako veliko matriko zahteva zelo podrobno znanje linearne algebre in z njo povezanih numeričnih metod. Nato iz vrednosti X le še določimo vrednost p.

3. **Binary rank test for 31x31 matrices**
Najbolj levih 31 bitov od 31 celih števil (32-bitnih) iz testne datoteke uporabimo za matriko A nad obsegom $0, 1$. Na tak način tvorimo 40.000 matrik in vsaki določimo rang[11], ki je med 0 in 31, vendar pa so rangi manjši od 28 redki in zato za hi kvadrat

[8] Razlaga bi le še podaljšala že tako obsežno matematično ozadje, zato le napotek bralcu k [3], tu pa le opomba, da so v matriki shranjene informacija o odstopanju slučajnega vektorja od njegovega upanja ter informacija o medsebojni odvisnosti posameznih komponent, skratka posplošitev variance slučajne spremenljivke na vektor.

[9] opisana v [5]

[10] glej [3]

[11] maksimalno število linearno neodvisnih vrstic

test na vzorcu rangov vzamemo particijo množice $\mathbb{Z}_{32}$: $E_1 = \{0, 1, 2, \ldots, 28\}, E_2 = \{29\}, E_3 = \{30\}$ in $E_4 = \{31\}$. Nato določimo vrednost p. Za test potrebujemo 4'73 MBytov veliko datoteko.

4. **Binary rank test for 32x32 matrices**
   Vseh 32 bitov od 32 celih števil iz testne datoteke uporabimo za matriko A nad obsegom $0, 1$. Na tak način tvorimo 40.000 matrik in vsaki določimo rang, ki je med 0 in 32, vendar pa so rangi manjši od 29 redki in zato za hi kvadrat test na vzorcu rangov vzamemo particijo množice $\mathbb{Z}_{33}$: $E_1 = \{0, 1, 2, \ldots, 29\}, E_2 = \{30\}, E_3 = \{31\}$ in $E_4 = \{32\}$. Nato določimo vrednost p. Za test potrebujemo 4'88 MBytov veliko datoteko.

5. **Binary rank test for 6x8 matrices**
   Najbolj levih 8 bitov od 6 celih števil (32-bitnih) iz testne datoteke uporabimo za matriko A nad obsegom $0, 1$. Na tak način tvorimo 100.000 matrik in vsaki določimo rang, ki je med 0 in 6, vendar pa so rangi manjši od 4 redki in zato za hi kvadrat test na vzorcu rangov vzamemo particijo množice $\mathbb{Z}_7$: $E_1 = \{0, 1, 2, 3, 4\}, E_2 = \{5\}$ in $E_3 = \{6\}$. Nato določimo vrednost p. Test ponovimo 25-krat, najprej smo to naredili na bitih 1–8 v izbranem 32 bitnem številu, sedaj pa to ponovimo še na bitih 2–9, 3–10, ..., 25–32. Na tako dobljenih 25 p-jih naredimo še

   KSTEST

   Za test potrebujemo 2'29 MBytov veliko datoteko.

6. **Overlapping 20-tuples bitstream test**
   Iz testne datoteke vzamemo $2^{21} + 19 = 2097171$ bitov (256 kBytov) $b_1, b_2, ..., b_{2^{21}+19}$. Zamislimo si abecedo z dvema črkama, 0 in 1. Potem iz testne datoteke sestavimo 20 črk dolge besede $b_1 b_2 \ldots b_{20}$, $b_2 b_3 \ldots b_{21}$, ... Tako dobimo 2^{21} besed. Vseh možnih 20 črk dolgih besed je 2^{20}. Z j označimo število tistih, ki manjkajo. Spremenljivka j naj bi bila porazdeljena $N(141909, 428^2)$, torej $\frac{j-1419009}{428} \sim N(0, 1)$. Vrednost p tega testa dobimo z enačbo iz trditve 1.3.4, torej

$$p = \int_{-\infty}^{\frac{j-1419009}{428}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx$$

7. **Overlapping-pairs-sparse-occupancy (OPSO)**
   V tem testu bomo jemali 10-bitne črke iz abecede z $2^{10} = 1024$ črkami. Iz testne datoteke bomo vzeli $(2^{21} + 1) \cdot 10$ bitov $b_1 b_2 \ldots$ in z njimi tvorili

   NI RAVNO TAKO (ZADNJI ODSTAVEK ????

   črke $C_1 = b_1 b_2 \ldots b_{10}$, $C_2 = b_{11} b_{12} \ldots b_{20}$,... Nato tvorimo besede (ki se prekrivajo, torej med sabo niso neodvisne) $B_1 = C_1 C_2$, $B_2 = C_2 C_3$, $B_3 = C_3 C_4$, ..., $B_{2^{21}} = C_{2^{21}} C_{2^{21}+1}$ Vseh možnih besed sestavljenih iz dveh črk je $(2^{10})^2 = 2^{20}$ in če z j označimo število manjkajočih dvočrkovnih besed v zaporedju $B_1, B_2, \ldots B_{2^{21}}$, potem je

$$\frac{j - 141909}{290} \sim N(0, 1)$$

Tu sta in matematično upanje in varianca dobljena z izračunom in ne tako kot v nekaj naslednjih testih s simulacijo.

missing words -¿ Z=(j-141...)/290 -¿ p=$\int$?

8. **Overlapping-quadruples-sparse-occupancy (OQSO**
Test je podoben testu OPSO, le da delamo z abecedo iz $2^5 = 32$ črk in tvorimo 4-črkovne besede. Z j označimo število manjkajočih 4-črkovnih besed v zaporedju $B_1, B_2, \ldots B_{2^{21}}$ in

$$\frac{j - 141909}{295} \sim N(0,1)$$

Kot je navedeno v izpisu Diehard testa, je matematično upanje (141909) spremenljivke j dobljeno s teorijo, varianca (295^2) pa z izčrpno simulacijo.

9. **DNA test**
Tu vzamemo abecedo iz $2^2 = 4$ črk, ki jih označimo z `C`, `G`, `A` in `T`. Iz testne datoteke z biti $b_1, b_2, b_3, \ldots$ tvorimo črke C_1, C_2, z njimi pa 10-črkovne besede $B_1 = C_1C_2 \ldots c_{10}$, $B_2 = C_2C_3 \ldots c_{11}$, ..., $B_{2^{21}} = C_{2^{21}}C_{2^{21}+1} \ldots C_{2^{21}+9}$ Določimo število j manjkajočih besed porazdelitev j naj bi bila $j \sim N(141909, 339^2)$. Tudi tu je varianca σ^2 dobljena s simulacijo.

10. **Count-the-1's test na zaporedju bytov**
Testno datoteko si mislimo kot zaporedje bytov (8 bitov). Vsak byte lahko vsebuje od 0 pa do 8 enic z verjetnostmi $\frac{1}{256}$, $\frac{8}{256}$, $\frac{28}{256}$, $\frac{56}{256}$, $\frac{70}{256}$, $\frac{56}{256}$, $\frac{28}{256}$, $\frac{8}{256}$ in $\frac{1}{256}$. Sedaj vsak byte spremenimo v črko `A`, `B`, `C`, `D` ali `E` z naslednjim pravilom:

število enic v bytu	ustrezna črka
0,1 ali 2	A
3	B
4	C
5	D
6,7 ali 8	E.

To je tako kot bi tipkali črke `A`, `B`, `C`, `D` in `E` z verjetnostmi $\frac{37}{256}$, $\frac{56}{256}$, $\frac{70}{256}$, $\frac{56}{256}$ in $\frac{37}{256}$. Na razpolago imamo 5^5 možnih 5-črkovnih besed. Podobno kot v prejšnjih testih tu naredimo iz 256004 bytov 256000 petčrkovnih besed. Preštejemo pojavitve posameznih besed. Tvorimo KVADRATIČNO FORMO in ŠIBKI INVERZ KOVARIANČNE MATRIKE posameznih pojavitev in iz teh podatkov naredimo hi kvadrat test: $Q5 - Q4$, razlike OF NAIVE PEARSON SUMS od $\frac{(observed-expected)^2}{expected}$ na štetju 5 in 4-črkovnih besedah ??????

Count-the-1's test na točno določenih bytih
Testno datoteko si mislimo kot zaporedje celih števil (integer), vsakega sestavljenega iz štirih bytov. Izmed teh štirih si izberemo enega, recimo najbolj levega. Ta vsebuje od 0 do 8 enic z verjetnostmi $\frac{1}{256}$, $\frac{8}{256}$, $\frac{28}{256}$, $\frac{56}{256}$, $\frac{70}{256}$, $\frac{56}{256}$, $\frac{28}{256}$, $\frac{8}{256}$, $\frac{1}{256}$

BLA, BLA, BLa ... isto kot prej

11. **Parking lot test**
Na parkirišču - kvadratu s stranico dolgo 100 - parkiramo avtomobile - kroge z radijem 1 - "po posluhu". To pomeni, da si naključno izberemo neko mesto in tam probamo parkirati avto. Če se zaletimo V drug avto, nič zato, naključno izberemo novo parkirišče, vse dokler nam ne uspe. Z n označimo število poskusov parkiranja, s k pa število uspešno parkiranih avtomobilov. V testu vzamemo $n = 12000$. S simulacijo (pa ne na parkirišču) so ugotovili, da se porazdelitev

$$\frac{k - 3523}{21,9^2}$$

približuje standardni normalni porazdelitvi. aha
KATERE BITE VZAMEMO?!
Za naš test določimo 10 k-jev. aha
ZA NAS TEST DOLOCIMO POVPRECEN k IN σ. KAJ JE KSTEST ?! PREVERI SVINCNIK NA LISTIH...

12. **Minimum distance test**
V kvadratu s stranico 10000 naključno izberemo $n = 8000$ točk in nato poiščemo najkrajšo razdaljo d med dvema točkama.

$$d = \min_{\substack{1<i<10000 \\ 1<j<10000}} d(T_i, T_j)$$

Spremenljivka d^2 naj bi bila eksponentno porazdeljena s parametrom 0'995, kar pomeni, da je

$$X = 1 - e^{-\frac{1}{0'995}d^2} \sim I[0,1]$$

Iz datoteke poberemo 100 vzorcev in na njih naredimo KSTESTTTTTTTTTTTTTTTTTT

13. **3D-spheres test**
V kocki s stranico 1000 naključno izberemo $n = 4000$ točk in nato poiščemo najkrajšo razdaljo d med dvema točkama.

$$d = \min_{\substack{1<i<10000 \\ 1<j<10000}} d(T_i, T_j)$$

Spremenljivka d^3 naj bi bila eksponentno porazdeljena s parametrom 30. Le ta je dobljen s simulacijo. To pomeni, da je

$$X = 1 - e^{-\frac{1}{30}d^2} \sim I[0,1]$$

Iz datoteke poberemo 20 vzorcev in na njih naredimo KSTESTTTTTTTTTTTTTTTTTT

14. **Sqeeze test**
Na datoteko gledamo kot na zaporedje celih števil (4 byti). Vsakega od teh števil pretvorimo v realno število U (npr. $45 \rightarrow 00000000\,00000000\,00000000\,00101101_{(2)} \rightarrow 0,00000000\,00000000\,00000000\,00101101_{(2)} \rightarrow 0,00000000000000000000003124545555MATHEM$
– Posamezen test določi število iteracij j potrebnih, da reduciramo začetni $k = 2^31 = 2.147.483.648$ na 1 z redukcijo $k = \lceil k \cdot U \rceil$. Test poišče 100000 j-jev, prešteje posamezne frekvence $j \leq 6$, $j = 7$, $j = 8$, ..., $j = 47$ in $j \geq 48$ in naredi na njih hi kvadrat test.

15. **Overlapping sums test**
    Tudi tu iz 4-bytnih celih števil naredimo realna števila U_1, U_2, ... Le ta naj bi bila enakomerno porazdeljena na $[0, 1)$. Sestavimo (med seboj odvisne) vsote $S_1 = U_1 + \ldots + U_100$, $S_2 = U_2 + \ldots + U_101$, ... Porazdelitev posameznih S-jev je NAVIDEZNO normalna z neko kovariančno matriko. Nato z linearno transformacijo S-jev naredimo zaporedje med seboj neodvisnih standardnih normalnih spremenljivk, ki jih nato z enačbo iz trditve 1.3.4 pretvorimo v enakomerno porazdelitev, primerno za KSTEST. Test ponovimo 10-krat in iz tako dobljenih desetih p-jev dobimo s še enim KSTESTOM končno vrednost p.

16. **Runs test**
    Tu iz 10000 4-bytnih celih števil naredimo realna števila U_1, U_2, ..., U_{10000}. Med njimi postavimo neenačaje $\geq$ in $<$ in z njihovo pomočjo tvorimo zaporedje bitov, kot prikazuje primer:

```
   0'123 < 0'357 < 0'789 > 0'425 > 0'224 < 0'416 < 0'950 ...
1       1       1       0       0       1       1  ...
```

    V tem zaporedju preštejemo dolžine blokov in vrzeli:

```
        3                       2                       ?       ...
```

    In tvorimo novi zaporedji: zaporedje blokov (3?...) in zaporedje vrzeli (2...) Kovariančna matrika za tak test je ZNANA———- LEADING TO CHISQUARES TEST FOR QUADRATIC FORMS IN THE WEAK INVERSE OF THE COVARIANCE MATRICES. Test ponovimo dvakrat.

17. **Craps test**

Literatura

[1] A. J. MENEZES, P. C. van OORSCHOT, S. A. VANSTONE. Handbook of Applied Cryptography. CRC Press LLC, 1997

[2] D. R. STINSON. Cryptography, Theory and Practice, CRC Press, 1995

[3] R. JAMNIK. Verjetnostni račun. Mladinska knjiga, Ljubljana, 1971

[4] G. R. GRIMMETT, D. R. STIRZAKER. Probability and Random Processes. Clarendon Press, Oxford, 1992

[5] M. FISZ. Probability Theory and Mathematical Statistics. PWN, Polish Scientific Publishers, New York, 1967

[6] R. JAMNIK. Matematična statistika. Državna založba Slovenije, Ljubljana, 1980

[7] A. FERLIGOJ. Osnove statistike na prosojnicah. Samozaložba, Ljubljana, 1995

[8] D. E. KNUTH. The art of computer programming. ???, ???, ????

[9] navodila za uporabo CRYPT-X testa

[10] Originalne razlage Diehard testov

[11] G. MARSAGLIA, A Current View of Random Number Generators. www:...

[12] G. MARSAGLIA, Monkey Tests for Random Number Generators. www:...

[13] www: test Za C++ Random()

[14] www: