# IZVORI SLUČAJNIH BROJEVA I NJIHOVA UPOTREBA

**Babić, Ana**



**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**
**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**Sveučilišni preddiplomski studij elektrotehnike**

# IZVORI SLUČAJNIH BROJEVA I NJIHOVA UPOTREBA

**Završni rad**

**Ana Babić**

**Osijek, 2019.**

## Sadržaj

# 1. UVOD

Koncept slučajnosti i dobivanja slučajnih brojeva dio je ljudske kulture već tisućama godina. Još se u atenskoj demokraciji koristio slučajni odabir u političke svrhe, a kasnije je uslijedilo miješanje karata i bacanje kocke ili novčića koji su omogućili ljudima slučajnost pri donošenju odluka ili slučajnost pri raznim igrama. Iako većina ljudi u današnjem svijetu nastoji uvesti red u svoj život, i danas se slučajnost i slučajni brojevi koriste u različite svrhe. Tako i dalje postoje igre s kockicom tipa Čovječe ne ljuti se, kao i takozvane igre na sreću čija je glavna karakteristika upravo slučajnost. Razvojem računala ljudi su došli do shvaćanja da uvođenje slučajnosti u računalne programe ima veliki značaj. Svaka kodirana veza koristi slučajne brojeve, a kako sigurnost na internetu i sigurna komunikacija dobivaju sve veću pažnju, broj kodiranih veza još će više rasti. Zbog toga slučajni brojevi imaju veliku ulogu u kriptografiji, a dobar izvor slučajno generiranih brojeva često predstavlja temelj sigurnosti računalnog sustava. Ovisno o primjeni, ponekad su potrebne velike količine slučajnih brojeva pa je nužna njihova brza i efikasna implementacija. Iz tog razloga izvor slučajno generiranih brojeva treba biti testiran na različite načine kako bi se provjerili eventualni nedostatci jer se oni u stvarnoj upotrebi mogu iskoristiti kao sigurnosne slabosti i tako načiniti štetu. U ovome je završnom radu prikazano kako su se izvori slučajnih brojeva razvijali tijekom povijesti i za što su sve slučajni brojevi koristili ljudima, kako danas – tako i u prošlosti.

Nakon uvoda, u drugom su poglavlju opisani slučajni brojevi općenito i njihova podjela na pseudoslučajne i prave slučajne brojeve. U trećem je poglavlju dan prikaz razvoja metoda izvora generatora slučajnih brojeva kroz povijest. Četvrto poglavlje sastoji se od primjene i važnosti modernih generatora i opisa nekih od njih. U petom poglavlju dano je programsko rješenje za generiranje slučajnih brojeva.

## 1.1. Zadatak završnog rada

U ovom završnom radu potrebno je istražiti upotrebu slučajnih brojeva – kako kroz povijest, tako i danas. Također je potrebno proučiti izvore i objasniti njihovu važnost. Isto tako, dio zadatka je obraditi neke određene primjere i dati vlastiti prijedlog izvora.

# 2. SLUČAJNI BROJEVI I NJIHOVA PODJELA

Slučajni brojevi predstavljaju niz brojeva čiju je pojavu i vrijednosti nemoguće predvidjeti. Budući da se u stvarnosti javljaju samo u prirodnim pojavama i situacijama koje su i same po sebi nepredvidljive, slučajni se brojevi dobivaju pomoću računala, odnosno generatora, primjenom različitih sklopovskih ili programskih rješenja. Dakle, slučajno generirane brojeve dijelimo u dvije kategorije, a to su „istinski“ slučajni brojevi i pseudoslučajni brojevi. Shodno tomu postoje generatori „istinski“ slučajnih brojeva (engl. *true random number generator,* TRNG) i generatori pseudoslučajnih brojeva (engl. *pseudo random number generator,* PRNG)

U praksi, prilikom proučavanja slučajnosti nekog niza brojeva, provode se razna testiranja koja provjeravaju zadovoljava li niz određene karakteristike koje se očekuju od niza slučajnih brojeva i pojavljuju li se neke nepravilnosti koje se ne bi smjele pojaviti u takvom nizu. Niz mora zadovoljavati uvjete slučajnosti, a to su nepredvidljivost i uniformnost. Niz brojeva je nepredvidljiv ako znajući bilo koji početak tog niza, ne možemo predvidjeti sljedeći član tog niza. Odnosno ako znamo $X_0$, $X_1$, $X_2$, $X_3$,.., $X_n$, ne možemo predvidjeti sljedeći član $X_{n+1}$ bez obzira na to koliko velik *n* bio. Niz brojeva je uniforman ako se u njemu sve moguće vrijednosti pojavljuju u podjednakim omjerima. Uzmimo za primjer bacanje standardne igraće kockice: ako ju bacimo tisuću ili više puta i zapišemo redom brojeve koji su se pojavili na gornjoj strani kockice, analizirajući zapisane brojeve ne postoji način na koji bismo mogli predvidjeti koji će se broj pojaviti prilikom sljedećeg bacanja. Razlog tomu je što svaki broj od 1 do 6 ima istu vjerojatnost da se pojavi, tj. ti su brojevi uniformno raspoređeni. [1]

## 2.1. „Istinski“ slučajni brojevi

„Istinski“ slučajni brojevi su oni brojevi koji se generiraju na temelju prirodnih pojava ili fizikalnih procesa koji su nepredvidljive prirode. Njihov je izvor nedeterministički što znači da je slučajan broj u nizu neovisan o svom prethodniku. Za dobivanje istinski slučajnih brojeva najčešće se koriste generatori koji su sklopovske naravi. Vrlo je teško dizajnirati takav generator koji će prikupljati vrijednosti neke pojave koje ne smiju biti predvidljive. Sama izvedba takvih generatora izrazito je osjetljiva i skupa, a sklop je podložan kvarovima. [2] Vrijeme i brzina koji su potrebni da se skupi entropija iz prirodnih izvora ovisi o fizičkim pojavama koje se mjere. Zbog toga se kaže su prirodni izvori pojavljivanja „prave“ entropije blokirajući – ograničeni su dokle god nije skupljeno dovoljno entropije da bi se mogli generirati „istinski“ slučajni brojevi. Generiranje niza brojeva može biti sporo, a time je njihova primjena ograničena.

Postoje dvije vrste izvora koji omogućavaju dobivanje istinski slučajnih brojeva, a to su oni koji se temelje na sklopovlju i programski orijentirani izvori.

Izvori temeljeni na sklopovlju koriste nepredvidljive fizikalne pojave za generiranje slučajnih brojeva i često je potrebna dodatna obrada dobivenih podataka. U ovom je slučaju generator periferni uređaj, dakle izvori se fizički povezuju s računalom. Jedan od primjera sklopovskih izvora je vrijeme između emisija čestica za vrijeme radioaktivnog raspada. Prema kvantnoj teoriji ne postoji način pomoću kojeg bi sa sigurnošću mogli odrediti kada će točno doći do radioaktivnog raspada pa tako taj proces predstavlja čistu slučajnost i omogućuje dobivanje „istinskih“ slučajnih brojeva. [3] Sljedeći je primjer izvora šum u poluvodičkoj diodi. Kako je taj šum po svojoj prirodi slučajna pojava, računalo može električne impulse prouzrokovane tim šumom pretvoriti u slučajne brojeve. Još neki od primjera sklopovskih izvora su zvuk iz mikrofona ili video ulaz iz kamere te frekvencijska nestabilnost oscilatora. [2]

Programski orijentirani izvori mogu biti sistemski sat, specifične varijable operacijskog sustava ili sadržaj ulaznog/izlaznog spremnika. Također, kao izvor entropije istinski slučajnih brojeva može poslužiti i vrijeme između dva pritiska na tipku tipkovnice ili miša. Vrlo je teško predvidjeti koji će nizovi brojeva proizaći iz izvora koji se koristi takvom entropijom. S druge strane, ponašanje sistemskog sata lakše je predvidjeti pa iako ovi izvori daju nepredvidive nizove brojeva, to ne mora nužno biti slučaj. Zbog toga je najbolje koristiti kombinaciju više različitih izvora kako bi se dobio dobar generator slučajnih brojeva. [2]

## 2.2. Pseudoslučajni brojevi

Pseudoslučajni brojevi druga su vrsta slučajnih brojeva, ali ti brojevi nisu u potpunosti slučajni, što nam govori i njihov naziv – *pseudo* (grč.) – lažno. Zbog potrebe za dobivanjem manje ili veće količine slučajnih brojeva razvile su se teorije za stvaranje raznih formula i algoritama koje računalo koristi za ostvarivanje pseudoslučajnih brojeva, što znači da brojevi generirani programom tehnički nisu slučajni jer svaki broj u nizu ovisi o prethodnim brojevima iz niza. Računalo koristi početnu vrijednost, tj. sjeme (eng. *seed value*) i pomoću razvijenih rekurzivnih aritmetičkih i/ili logičkih formula daje niz brojeva. Ulaz u generator, tj. sjeme, mora biti „istinski” slučajan, a dobiva se iz generatora „istinski” slučajnih brojeva (npr. vrijeme sistemskog sata). Početna sjemenska vrijednost tada služi kao koeficijent ili kao konstanta na temelju koje se računaju ostali članovi niza, što znači da izvori pseudoslučajnih brojeva imaju deterministička karakteristiku. Dakle, iako je takav niz naizgled slučajan, njegova stvarna karakteristika je

predvidljivost. To znači da te brojeve možemo predvidjeti, odnosno rekonstruirati algoritam ako znamo početne uvjete niza.

Prilikom generiranja pseudoslučajnih brojeva niz mora zadovoljavati određena svojstva. Vjerojatnost pojavljivanja svakog broja mora biti približno jednaka i mora postojati ravnomjeran raspored brojeva u prostoru. Također, ne smije postojati korelacija kod nizova dobivenih pomoću generatora pseudoslučajnih brojeva, tj. nijedan podniz u nizu ne smije ovisiti o nekom drugom podnizu. Period pseudoslučajnog generatora treba biti velik koliko god je to moguće. Generiranje se vrši deterministčkim putem te unatoč tomu što generator prolazi kroz puno različitih stanja, njihov broj je ipak konačan pa će se stanja u nekom trenutku početi ciklički ponavljati. Broj brojeva u nizu koji se ciklički ponavlja naziva se period. [2]

Pseudoslučajnost se može postići upotrebljavanjem jednostavnih formula koje su izvedive u samo jednoj liniji programskog koda, ovisno o potrebi korištenja. Naprimjer, ako igramo videoigricu, nije nam važno jesu li događaji u njoj proizašli iz „istinski“ slučajnih brojeva ili pseudoslučajnih brojeva. S druge strane, ako ih koristimo pri enkripciji, vrlo je važno da algoritam bude kvalitetan i da se niz brojeva ne može jednostavno predvidjeti. Tada bi algoritam trebao sadržavati više aritmetičkih operacija i parametara te logičke operacije jer se tako način generiranja bolje sakriva i otežava se eventualno pogađanje formule kojom su dobiveni brojevi.

Glavna razlika između generatora pseudoslučajnih brojeva i generatora istinski slučajnih brojeva je ta da se kod izlaza iz pseudoslučajnog generatora brojevi nakon određenog vremenskog perioda počinju periodično ponavljati, dok se generator istinski slučajnih brojeva ponaša na drugačiji način, odnosno svi su brojevi slučajni i potpuno nepredvidljivi. Unatoč tomu u raznim se testiranjima pseudoslučajni brojevi znaju pokazati uspješniji od slučajnih brojeva koji su dobiveni iz fizičkih izvora. Iako se pseudoslučajni nizovi dobivaju pomoću metoda s deterministčkim karakteristikama, pažljivim odabirom početnih uvjeta, kao i samog generatora, te se karakteristike mogu prikriti i tako osigurati veću slučajnost niza. S druge strane, deterministčke karakteristike niza pseudoslučajnih brojeva mogu biti vrlo korisne. Generatori pseudoslučajnih brojeva imaju mogućnost reproduciranja niza, a ta je mogućnost od velikog značaja pri detekciji i ispravljanju pogrešaka uzorkovanih od strane generatora jer loš generator može uzrokovati neispravan rad programa.

Općenito gledajući – generatori pseudoslučajnih brojeva brži su, lakše ostvarivi i prenosivi. U različitim simulacijama često su potrebne velike količine slučajnih brojeva i tada je bitno da generator bude brz koliko god je to moguće, a da pritom ne ugrozi kvalitetu generiranog niza, tj.

određene karakteristike slučajnosti. Što se tiče prenosivosti, važno je da prenosivi generator pokazuje ista svojstva u različitim uvjetima rada kao što su npr. različiti strojevi, operacijski sustavi ili programski jezici ugradnje. Uz iste početne uvjete, generator treba generirati isti niz brojeva.

# 3. RAZVOJ METODA I IZVORA SLUČAJNIH BROJEVA KROZ POVIJEST

Još u dalekoj povijesti ljudi su pokušavali stvoriti načine pomoću kojih bi si osigurali slučajnost. Najranije metode generiranja slučajnih brojeva zasnivale su se na ručnim metodama. Tako su najstarije poznate igrače kockice otkrivene još u 24. stoljeću prije Krista u grobnici na Bliskom istoku. Nakon toga je otkriveno da su oko 1100. g. pr. Kr. u Kini pomoću žarača zagrijavali kornjačin oklop sve dok nije puknuo. Te su nasumične pukotine koristili za interpretaciju i predviđanje sudbine. Stoljećima nakon toga nastali su I Ching heksagrami za proricanje sudbine. Oni su se sastojali od prekinutih i neprekinutih linija načinjenih od stabljike stolisnika, a putanje su tih linija bile određene bacanjem triju novčića. [4]

No razvojem modernog svijeta nastala je potreba za puno većim brojem slučajnih brojeva nego što bi to mogla ponuditi igrača kockica ili stabljika stolisnika. 50-ih godina prošloga stoljeća američka korporacija RAND izdala je knjigu koja se sastojala od tablica slučajnih brojeva pod nazivom „*A Million Random Digits with 100,000 Normal Deviates*“. Slučajni brojevi dobiveni su pomoću korištenja generatora impulsa, a prije nego što su se iskoristili za stvaranje tablica slučajnih brojeva, pažljivo su testirani i filtrirani. Ta je knjiga bila preokret u povijesti jer su po prvi put duži nizovi kvalitetnih slučajnih brojeva postali dostupni svima, od znanstvenika i istraživača do matematičara. Najviše je primjene imala u statistici i znanstvenim istraživanjima.

```
73735 45963   78134 63873
02965 58303   90708 20025
98859 23851   27965 62394
33666 62570   64775 78428
81666 26440   20422 05720

15838 47174   76866 14330
89793 34378   08730 56522
78155 22466   81978 57323
16381 66207   11698 99314
75002 80827   53867 37797

99982 27601   62686 44711
84543 87442   50033 14021
77757 54043   46176 42391
80871 32792   87989 72248
30500 28220   12444 71840
```

**Sl. 3.1.** *Izvadak iz knjige „A Million Random Digits with 100,000 Normal Deviates“* [4]

Za generiranje slučajnih brojeva koristila se i pulsirajuća električna vakuumska cijev koja je proizvodila 50 slučajnih brojeva u sekundi. Također, 50-ih godina prošloga stoljeća nastao je

uređaj za generiranje slučajnih brojeva „*ERNIE*“ (Electronic Random Number Indicator Equipment) koji se koristio za određivanje dobitnika lutrije Velike Britanije. Razvojem računalne simulacije sve se više pažnje posvećivalo metodama koje bi odgovarale radu računala. Postojala je ideja da se uređaj tipa ERNIE poveže direktno na računalo, ali to nije bilo dobro rješenje jer je postojao problem nemogućnosti kasnije reprodukcije sekvence dobivenih slučajnih brojeva. Drugo moguće rješenje bilo je usmjereno na korištenje baze generiranih slučajnih brojeva. Problem je bio što bi tada bila mala brzina rada, a veliki memorijski kapaciteti tada su bili skupi. [4]

Kao posljedica istraživanja počele su se pojavljivati aritmetičke metode generiranja slučajnih brojeva koje su se zasnivale na tome da slučajni broj daju na osnovi rezultata jednog ili više prethodno generiranih slučajnih brojeva, a prema unaprijed definiranoj matematičkoj formuli. Pri tome se skup prethodno generiranih vrijednosti, najčešće jedna vrijednost, koje determiniraju sljedeći broj u nizu nazivaju sjeme. Prvi takav generator predložio je John von Neumann, američki matematičar mađarskog podrijetla. Generatori navedeni u sljedećim potpoglavljima neki su od prvih i najvažnijih primjera generatora slučajnih brojeva u povijesti, a neki se od njih, ili barem njihove inačice, koriste i danas za razne primjene.

## 3.1. John von Neumannov generator slučajnih brojeva

John von Neumann 1946. godine stvorio je prvi generator slučajnih brojeva. Algoritam je bio jednostavan, a zasnivao se na metodi sredine kvadrata. U toj se metodi koristi n-teroznamenkasti prirodni broj kao početna vrijednost, tj. sjeme, taj se broj kvadrira i potom se uzima srednjih n znamenaka dekadskog zapisa tog rezultata i postupak se ponavlja. Uvijek se uzima srednjih n znamenaka, a ako znamenaka ima manje, onda se dodaje potreban broj nula s lijeve strane zapisa. Na slici 3.1. prikazan je primjer dobivanja slučajnih brojeva ovom metodom. Broj znamenaka n postavljen je na 6, a za sjeme odabran je broj 458926. Kvadrat tog broja je 210613073476, što znači da je sljedeći slučajni broj načinjen od šesteroznamenkaste sredine ovog broja 613073. Broj 613073 zatim se kvadrira, uzima se šesteroznamenkasta sredina dobivenog rezultata i postupak se konstantno ponavlja. Ako se dovoljno puta ponovi postupak, dobit će se slijed brojeva koji pokazuju statistička svojstva slučajnosti. [1]

**n=6**

broj: 458926

$458926^2 = 210\,\mathbf{613\,073}\,476$

broj: 613073

$613073^2 = 375\,\mathbf{858\,503}\,329$

broj: 858503

$858503^2 = 737\,\mathbf{027\,401}\,009$

broj: 027401

…

**Sl. 3.2.** *Prikaz primjera John von Neumannove metode sredine kvadrata*

Ipak, ova metoda ima svoje slabosti. Ako se za vrijednost broja n uzme neparan broj, tada postoji mogućnost da će kvadriranjem nastati takav niz brojeva iz kojeg se ne može točno odabrati srednjih n znamenki. Ako se, naprimjer, kvadrira troznamenkasti broj 456 (n=3), za rezultat se dobije šesteroznamenkasti broj 207936. Budući da je n=3, za sljedeći broj trebamo uzeti srednje 3 znamenke. To znači da ostaju još 3 znamenke koje se trebaju raspodijeliti lijevo i desno od te sredine, a nemoguće ih je rasporediti jednako s obje strane sredine. Tada je dozvoljeno dodati potreban broj nula s lijeve strane zapisa kako bi se stvorila definirana sredina. Osim navedenoga metoda sredina kvadrata ima još jedan ključan nedostatak. Neovisno o tome koja je bila početna vrijednost, nakon nekoga će vremena metoda početi generirati isti broj ili isti ciklus brojeva (npr. 8100, 6100, 4100, 8100, 6100, 4100…) i tako stvoriti petlju. Isto tako, vrlo brzo konvergira k nuli i tako svi ostali članovi niza postaju nule. [5]

## 3.2. Linearno kongruentni generator – LCG

Linearno kongurentni generator (engl. *Linear congruential generator*, LCG) jedan je od najutjecajnijih i najpoznatijih generatora u povijesti. Ovaj je generator nastao na temelju ideje koju je predstavio američki matematičar Derrick Henry Lehmer 1948. godine. Ideja je da generator osim sjemenom bude određen još trima unaprijed definiranim parametrima, a to su cjelobrojne konstante $a$, $c$ i $m$ ($0 \leq a, c < m$). Ideja se bazira na primjeni sljedećeg izraza: [5]

$$X_{n+1} = (aX_n + c) \bmod m, \ n = 0,1,2 \ldots \quad (3\text{-}1)$$

gdje je:

$X_n$ - n-ti slučajni cijeli broj

$X_{n+1}$ - n+1 slučajni cijeli broj

$X_0$ – početna vrijednost, sjeme

$a$ – koeficijent (multiplikator)

$c$ – konstanta (inkrement)

$m$ – modul za modulsku aritmetiku

Da bi se dobio prvi broj ovog niza, potrebno je odrediti početnu vrijednost $X_0$, tj. sjeme. Početno sjeme može biti postavljeno kao konstanta ili zadana vrijednost, može se dobiti pomoću nekog istinski slučajnog generatora kao što je sistemski sat, a može biti i određeno od strane korisnika.

Za primjer uzmimo da je $m$=10, $X_0$= $a$ = $c$ = 6. Izraz će sada glasiti $X_{n+1} = (6X_n + 6) \bmod 10$, a prvih nekoliko članova niza su:

$$X_1 = (6 \times 6 + 6)\ mod\ 10 = 42\ mod\ 10 = 2$$
$$X_2 = (6 \times 2 + 6)\ mod\ 10 = 18\ mod\ 10 = 8$$
$$X_3 = (6 \times 8 + 6)\ mod\ 10 = 54\ mod\ 10 = 4$$
$$X_4 = (6 \times 4 + 6) mod\ 10 = 30\ mod\ 10 = 0$$
$$X_5 = (6 \times 0 + 6)\ mod\ 10 = 6\ mod\ 10 = 6$$

i prema njima se vidi da će se niz nakon nekog vremena početi ponavljati, tj. niz je periodičan. [1] Broj brojeva u nizu prije prvog ponavljanja naziva se period tog niza, a u gornjem primjeru imamo slučaj gdje je period četiri. To ponavljanje niza loša je osobina ove metode. Zbog jednostavne aritmetike kojom se generiraju slučajni brojevi, potrebno je pripaziti jer će se za isto početno sjeme $X_0$ uvijek dobiti isti niz brojeva kao i kada se prvi put generira niz. Svaki niz dobiven metodom linearne kongruencije periodičan je, tj. u nizu se pojavljuje konačno mnogo brojeva jer se pojavljuju brojevi između 0 i $m$-1 pa se nakon nekog vremena mora dogoditi ponavljanje jednog od prijašnjih članova niza. No moguće je odabrati takve brojeve $a$, $c$ i $m$ kako bi period niza bio toliko velik da se u praksi ponavljanje rijetko događa. Dakle cilj je osigurati maksimalan period generatora pažljivim biranjem parametara $a$, $c$ i $m$. Danas se za parametar $m$ najčešće uzima broj oblika $2^b$ gdje je $b$ broj bita u riječi koju obrađuje procesor računala ili najveći prosti broj prikaziv u računalu. Ukoliko je $m$ oblika $2^b$, tada se maksimalni period dobije ako je $a$-1 višekratnik svakog

prostog faktora modula *m* i uz to *a*-1 mora biti djeljiv s 4 ako je i *m* djeljiv s 4. Uvijek se može uzeti *c*=1. [6]

Zahtjevi su ovog generatora za memorijom minimalni što ga čini brzim pa se LCG algoritam koristio kroz povijest u mnogim programskim jezicima za generiranje pseudoslučajnih brojeva. Zbog svoje brzine i malih memorijskih zahtjeva LCG se i danas često koristi jer je za većinu primjena sasvim dovoljan generator. No kada se radi o većim računalnim simulacijama gdje je potreban veliki broj slučajnih brojeva, LCG nije primjenjiv. LCG nije dovoljno siguran te zbog toga nije namijenjen i ne bi se trebao koristiti za primjenu u kriptografiji ili aplikacijama koje zahtijevaju visoku razinu kvalitete generiranja slučajnih brojeva.

Najpoznatiji LCG generator naziva se RANDU, stvoren je 1968. godine od strane IBM-a, a njegovi parametri odabrani su na način da bude jednostavan za izvedbu i dovoljno brz. [5]

## 3.3. Lagged Fibonacci generator

Uz LFC jedan od osnovnih i najranijih generatora je LFG. Kao što i sam naziv govori, LFG se temelji na uopćenom izrazu za niz Fibonacijevih brojeva. Niz Fibonaccijevih brojeva definiran je sljedećim izrazom: [7]

$$X_n = X_{n-1} + X_{n-2},\ X_1 = 0, X_2 = 1 \tag{3-2}$$

Fibonacijev niz počinje brojevima 0 i 1, a nakon te dvije početne vrijednosti, svaki sljedeći broj predstavlja zbroj dvaju prethodnika. Dakle pomoću izraza (3-2) dobit ćemo sljedeći niz brojeva: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, …

Opći oblik formule za generiranje slučajnih brojeva pomoću LFG: [2]

$$X_n = X_{n-p} \circ X_{n-q} \tag{3-3}$$

Pri čemu su p i q cijeli broj i za njih vrijedi p > q > 0, a ° je binarna operacija (zbrajanje, množenje, mod *m*, XOR, …). Najčešće se upotrebljava modulo *m* artimetika, a *m* se uzima u

obliku potencije broja dva ($m$=$2^M$). Pri tome izraz (3-2) poprima sljedeći oblik:

$$X_n = \left(X_{n-p} + X_{n-q}\right) mod 2^M \tag{3-4}$$

LFG se tada označava kao uređena trojka LFG (p, q, M). Ako se parametri *p* i *q* odaberu pažljivo, moguće je postići duže periode generatora nego što su kod LCG.

S obzirom na to da je odabir početnih vrijednosti kompleksan proces, najbolje je koristiti literaturu za dobar odabir parametara potrebnih za određenu primjenu. Američki matematičar i informatičar Donald Knuth u svojoj je knjizi „*The Art of Computer Programming"* dao popis mogućih parova vrijednosti parametara *p* i *q*, a neki su od njih:

(30, 127), (83, 258), (107, 378), (273, 607), (1029, 2281), (576, 3217), (4187, 9689), (7083, 19937), (9739, 23209)

Parametri manjih vrijednosti za rezultat daju kraće periode, generirat će se mali broj slučajnih brojeva prije nego što se prvi slučajni broj ponovno pojavi. S obzirom na to da postoje početne sjemenske vrijednosti, izlazne vrijednosti su predvidljive i ponovljive, a niz će postati cikličan ako dovoljno dugo čekamo. No uz pravi odabir početnih vrijednosti period će biti dovoljno dugačak da bi se cikličko ponavljanje moglo zanemariti.

Različita testiranja pokazala su da LFG generira niz brojeva s boljim slučajnim karakteristikama nego LCG, ali isto tako LFG zahtijeva veći memorijski prostor nego LCG. Iako postoje određene nepravilnosti, LFG i danas ima široku svrhu primjene u računalnim simulacijama. [2]

## 3.4. Mersenne Twister

Mersenne Twister generator je pseudoslučajnih brojeva, a predstavili su ga Makoto Matsumoto i Takuji Nishimura 1998. godine. Daleko je najčešće korišteni PRNG opće namjene. Naziv ovog algoritma proizlazi iz činjenice da mu je dužina perioda Mersennov prosti broj. Mersennovi prosti brojevi oblika su $2^n$-1, a najčešće korišten Mersenne Twister algoritam je onaj kojemu je period jednak prostom broju $2^{19937}$-1. Taj algoritam označava se kao MT19937 i koristi 32-bitne računalne riječi. Osim ove verzije postoji i verzija algoritma koja koristi 64-bitne riječi, MT19937-64. [8]

Mersenne Twister algoritam implementiran je u mnoga softverska rješenja i sustave kao što su Microsoft Excel, MATLAB, Python, R, itd.

Prednosti ovog generatora su da prolazi razne statističke testove uključujući *Diehard* testove i većinu *TestU01* testova. Ima dugačak period $2^{19937}$-1 što ne jamči kvalitetu generatora, ali svakako je prednost u odnosu na kratke periode koji mogu biti problematični. Također, implementacije ovog algoritma obično generiraju brojeve brže od ostalih metoda. [8]

S druge strane, potreban mu je relativno veliki međuspremnik i nije kriptografski siguran, odnosno pogodan za korištenje u kriptografiji.

# 4. PRIMJENA MODERNIH GENERATORA

Slučajni su brojevi neizostavni dio današnjeg suvremenog društva u kojemu tehnologija igra sve veću ulogu i koje se temelji na razmjeni informacija, digitalnoj obradi podataka u računalima, mobilnim uređajima i slično. U ovome su poglavlju navedene neke od primjena i važnost uporabe generatora slučajnih brojeva, kao i primjeri modernih generatora.

Generatori slučajnih brojeva imaju široku primjenu. U znanstvenim i inženjerskim područjima računalne simulacije stvarnih procesa zahtijevaju upotrebu slučajnih brojeva kako bi simulirani procesi bili što realniji. Slučajni se brojevi koriste u svrhu odabira testnih uzoraka, u numeričkim analizama, računalnim igrama, programiranju, zabavnoj industriji i lutrijama, pa čak i u pravosudnim sustavima i bankarstvu. Današnji automati za kockanje, kako stvarni, tako i virtualni, *online* igre na sreću i lutrije zasnivaju se na upotrebi slučajnih brojeva. Time se otklanja mogućnost da igrač poveća svoju vjerojatnost za pobjedu otkrivanjem sklonosti prema određenim ishodima u postupku igre i osigurava se jednaka vjerojatnost pobjede svakom igraču. Banke i instituti za financije imaju neke od najzahtjevnijih IT zahtjeva jer moraju osigurati konstantnu dostupnost podataka za bankarske transakcije i istodobno štititi povjerljive podatke o klijentima i njihovom vlasništvu, stoga je i u ovom području vrlo važna upotreba slučajnih brojeva prilikom šifriranja.

Uz sve nabrojane primjene slučajnih brojeva njihova je najvažnija primjena u kriptografiji. Osnovni je cilj kriptografije omogućiti očuvanje tajnosti informacija, čak i u komunikaciji koja se odvija nesigurnim komunikacijskim kanalom. U današnjem svijetu povjerljivi podaci i imovina vlade, poduzeća i pojedinaca sve se više čuvaju u digitalnom obliku i osiguravanje je tih podataka od vitalne važnosti. Upravo su zato slučajni brojevi neizostavni dio kriptografskih protokola u svrhu osiguravanja privatnosti, sigurnosti i integriteta podataka.

Današnji kriptografski sustavi temelje se na matematičkim modelima (algoritmima) koji običan tekst pretvaraju u šifrirani. Svaki algoritam radi na unaprijed definiran način određen matematičkim formulama i transformacijama. Ono što je jedinstveno u svakom kriptosustavu jest ključ za enkripciju (šifriranje). On predstavlja početak i kraj svakog algoritma što znači da sigurnost sustava zapravo ovisi o njemu. Još je davne 1883. godine matematičar Auguste Kerckhoffs tvrdio da se sigurnost kriptosustava ne smije temeljiti na tajnosti algoritma, nego na tajnosti ključa. Dakle, prema njegovu principu, kriptosustav bi trebao biti siguran čak i ako je sve o sustavu, osim ključa, javno znanje. Ovaj je princip i danas prihvaćen kao temelj stvaranja kompaktnih i pouzdanih kriptosustava. Da bi to funkcioniralo, ključ bi trebao biti potpuno slučajan. Sigurnosni se rizik pojavljuje kada brojevi generirani za stvaranje ključa nisu dovoljno

slučajni jer sustav tada postaje ranjiv. Mnogi se ključevi danas generiraju koristeći pseudoslučajne generatore, a kao što je već pojašnjeno, pseudoslučajni brojevi nisu uistinu slučajni. Postojanje ključa u sustavu koji nije generiran istinski slučajnih generatorom stvara veliki rizik, a posljedice napada na sustav mogu biti neizmjerne.

Jedna od novijih današnjih tehnologija, *blockchain*, temelji se na korištenju kriptografije i slučajnih brojeva. *Blockchain* jedna je vrsta baze podataka u kojoj se podaci nalaze u lancu zapisa, odnosno tzv. podatkovnim blokovima i usko je povezana s kriptovalutama. Svaki se novi blok nadovezuje na prošli, a međusobno su povezani pomoću kriptografije. Entiteti *blockchain*-a identificiraju se adresom, obično s par kriptografski povezanih slučajno generiranih ključeva koji se, između ostalog, koriste za digitalno potpisivanje transakcija. Adrese blokova stvaraju se generiranjem slučajnih brojeva i kriptografskim *hash* funkcijama koje proizvoljan niz znakova pretvaraju u fiksni niz znakova, *hash*. *Hash* funkcije sprječavaju utjecaj treće strane unutar komunikacijskog kanala i osiguravaju poruke od krivotvorenja. Kao takve, zahtijevaju generiranje slučajnih brojeva izrazite kvalitete, odnosno visoku i konstantno dostupnu entropiju kako bi se izbjeglo kašnjenje u obradi transakcija. Dakle, slučajni su brojevi ključni za pouzdanost *blockchain*-a i jer svaku slabost korištenih slučajnih brojeva potencijalni napadač može iskoristiti i oštetiti sustav. [9]

Telekomunikacije su također jedan od primjera korištenja kriptografije. Svake sekunde stvaraju se enormne količine podataka prilikom prijenosa, bilo da se radi o govornim pozivima, elektroničkoj pošti ili o drugim raznim alatima i *streaming* uslugama. Zbog toga su telekomunikacijske mreže meta za napadače koji mogu presresti i prisluškivati komunikaciju te tako ugroziti integritet podataka. Svaki takav napad rezultira klijentovim gubitkom povjerenja u pružatelja usluge. Kako bi se borili protiv takvih napada i osigurali prijenos podataka, pružatelji telekomunikacijskih usluga okreću se šifriranju.

Kao što je već rečeno, u kriptografskom sustavu tajnost ključa ima najvažniju ulogu. Ne smije postojati način kojim bi se utjecalo na generiranje ključeva, kao što ne smije postojati ni neki uzorak u generiranju koji bi se mogao iskoristiti kao sigurnosna slabost. Trebala bi postojati trenutna dostupnost generiranja ključeva kako bi se u slučaju ponovnog pokretanja sustava izbjegle odgode i prekidi u obradi podataka. Također, trebale bi postojati obavijesti upozorenja u slučaju da je nešto u generiranju ključeva pošlo po krivu. Rješenje generiranja brojeva za potrebe stvaranja ključeva korištenjem generatora pseudoslučajnih brojeva nije dovoljno dobro jer se nakon nekog

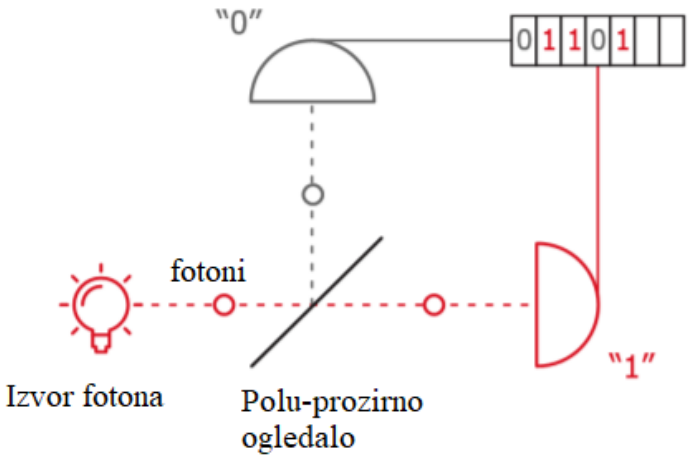
vremena uvijek pojave ciklički uzorci. Najbolje rješenje za generiranje slučajnih brojeva u svrhu kriptografije generatori su istinski slučajnih brojeva koji se temelje na nekom fizičkom fenomenu.

Jedan od takvih generatora, kvantni generator Quantis koji zahvaljujući svojoj vrhunskoj izvedbi ima široku primjenu u kriptografiji, opisan je u potpoglavlju 4.1. U potpoglavlju 4.2. dan je opis modernog generatora „Ranger“ koji također posjeduje potrebne karakteristike za primjenu u kriptografiji. U završnom potpoglavlju 4.3. opisuje se *online* servis *Random.org* koji generira slučajne brojeve pomoću atmosferskog šuma.

## 4.1. Kvantni generator slučajnih brojeva - Quantis

Jedan od najvažnijih primjera današnjih generatora kvantni je generator Quantis koji je proizvod švicarske tvrtke *ID Quantique*. Quantis generator je *state-of-the-art* generator istinskih slučajnih brojeva koji koristi kvantnu optiku kao izvor istinske slučajnosti.

Na slici 4.1. prikazan je shematski princip optičkog sustava koji koristi Quantis. Optika je znanost o svjetlu, a svjetlo se sastoji od elementarnih čestica – fotona. Fotoni u određenim situacijama pokazuju karakteristike nasumičnosti. Jedna je od takvih situacija prijenos fotona u odnosu na poluprozirno ogledalo. Činjenica je da je pojava refleksije ili transmisije na takvoj komponenti slučajna i na nju ne mogu utjecati nikakvi vanjski parametri. Upravo zbog tog Quantis generatori koriste izvor fotona koji odašilje fotone jedan po jedan na poluprozirno ogledalo pri čemu dolazi do slučajnih pojava. Ovisno o tome radi li se o refleksiji ili transmisiji, te se pojave detektiraju kao „0“ ili „1“ stanje bita. Ovaj kvantni proces dokazano je slučajan i kao takav pruža trenutan i neiscrpan izvor entropije za generiranje slučajnih brojeva. [10]




**Sl. 4.1**. *Shematski prikaz principa optičkog sustava kvantnih generatora* [10]

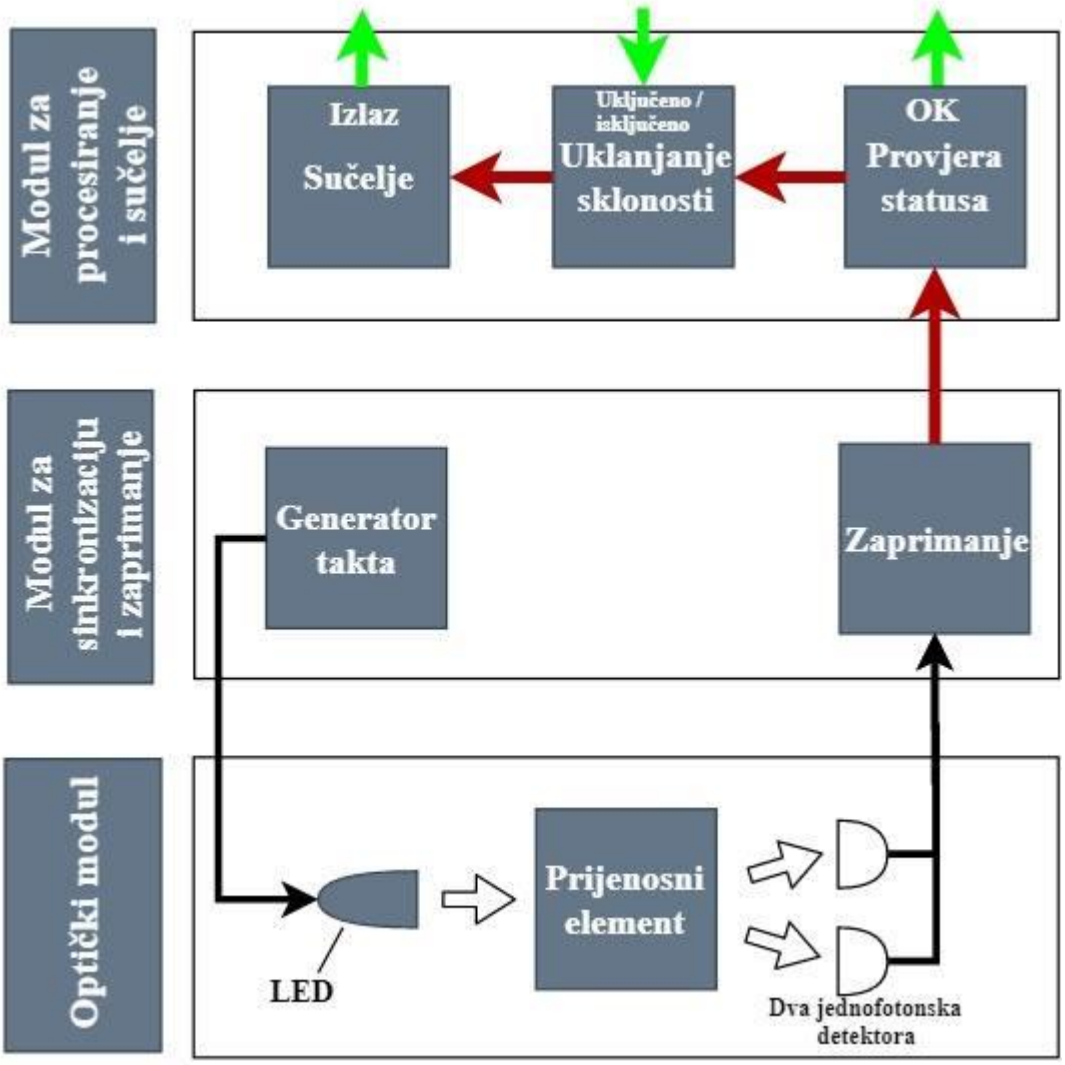
Slika 4.2. prikazuje sam princip rada Quantis generatora. Iz blok-dijagrama vidi se da se cjelokupni sustav rada Quantis generatora temelji na tri modula.

Prvi je od njih optički modul koji predstavlja jezgru generatora i sadrži optičke elemente za provođenje slučajnih procesa i slučajnih rezultata. On uključuje svjetleću diodu ili LED, prijenosni element u kojem se odvija slučajni proces i dva jednofotonska detektora koji bilježe ishod.

Modul za sinkronizaciju i zaprimanje provodi kontrolu optičkog modula. On se sastoji od generatora takta i pokretačke elektronike za izvor te elementa za zaprimanje izlaza dva jednofotonska detektora.

Treći modul je modul za procesiranje i sučelje. U ovome se modulu provode statističke i hardverske provjere te radnje za uklanjanje sklonosti naginjanja jednoj vrijednosti u odnosu na drugu. Naime, fizičke je procese teško precizno i u potpunosti uravnotežiti. Isto tako, teško je postići da vjerojatnost pojave stanja bita „0“ bude jednaka pojavi stanja „1“, odnosno da je vjerojatnost tih pojava 50%. Kod Quantis generatora te se vjerojatnosti kreću u rasponu od 45% do 55%. S obzirom na to da te vrijednosti vjerojatnosti nisu prihvatljive kod nekih aplikacija, modul za procesiranje izvodi naknadne obrade niza slučajnih brojeva za uklanjanje sklonosti. Također, ovaj podsustav oblikuje izlazne elektroničke signale. [10]

Izlazni tok Quantis generatora doseže brzine do 16 Mb/s, a da pritom ne pokazuje nikakvu korelaciju i prolazi sve statističke testove slučajnosti.




**Sl. 4.2.** *Blok-dijagram rada generatora* [10]

Jedna od glavnih karakteristika kvantnih generatora slučajnih brojeva je da se temelje na jednostavnom i osnovnom slučajnom procesu koji se lako modelira i nadzire. Jedinica za procesiranje Quantis generatora provodi trenutne provjere njegove funkcionalnosti i stalno nadzire njegov rad te tako čini izvor slučajnosti zaštićen od poremećaja u okolišu. Konstantno se provjerava rade li izvor svjetlosti i dva detektora ispravno i jesu li statistike izlaznog toka unutar određenih granica. Ukoliko su svi uvjeti ispunjeni, stanje bita bit će „1“, a ako jedan od uvjeta nije ispunjen, tj. postoji neka greška u radu, status bita postavlja se na „0“ i odmah se zaustavlja izlazni tok bitova. Ovo svojstvo pruža korisnicima Quantisa visoku razinu povjerenja u vjerodostojnost slučajnih brojeva koje koriste.

S obzirom na to da se Quantis temelji na kvantnim mehaničkim procesima koji su dobro poznati i definirani, relativno ga je lako certificirati pa tako Quantis generatori imaju najviše certifikata kao TRNG od cjelokupne ponude na tržištu. Priznanje za Quantisov rad potvrđuju sljedeći nabrojani certifikati ili vladine ovjere: [11]

- NIST SP800-22, skup statističkih ispitivanja za slučajnost
- iTech Labs, pojedinačni certifikat
- METAS certifikat
- usklađenost s BSI AIS31 standardom
- CTL certifikat

Quantis je dostupan kao OEM komponenta u obliku kompaktnog metalnog paketa koji se može montirati, PCI Express (PCI-E) kartice ili kao USB uređaj i lako se integrira s postojećim aplikacijama.


**Sl. 4.3.** *Quantis u dostupnim oblicima*

U usporedbi s ostatkom generatora na tržištu Quantis generatori vodeći su što se tiče ekonomičnosti, pouzdanosti i ostvarivanja potrebnih funkcija te certifikacije. Za razliku od pseudoslučajnih generatora koji trebaju vremena kako bi osigurali entropiju, Quantis generatori

trenutno pružaju potpunu slučajnost (entropiju) već od prvog fotona (bita). Također, Quantis generatori kompatibilni su sa svim značajnijim operacijskim sustavima. Uz Quantis dolazi i aplikacija *EasyQuantis* koja omogućuje čitanje slučajnih brojeva u obliku binarnih i cijelih brojeva te u obliku brojeva s pomičnim zarezom. Slučajni se brojevi mogu prikazati u prozoru ili spremiti u obliku datoteke. Aplikacija omogućuje napredne funkcije poput skaliranja ili ekstrakcije slučajnosti. [10]

Quantis bliblioteka (engl. *library*) može se koristiti za pristup Quantis generatorima. Biblioteka aplikacijsko-programskog sučelja (engl. *application programming interface*, *API*) jednaka je za PCI, PCIe i USB biblioteke i dostupna je na svim podržanim operacijskim sustavima. Biblioteka nudi mogućnost stvaranja slučajnih binarnih podataka, cijelih brojeva i brojeva s pomičnim zarezom. Biblioteka sadrži ekstenzije s implementacijom ekstraktora slučajnosti koji se može koristiti za naknadnu obradu izlaza Quantis generatora. [10]

Uzevši u obzir sve nabrojane prednosti i mogućnosti koje pruža Quantis, primjena je ovih generatora vrlo široka. Generatori su dizajnirani za bilo koju aplikaciju koja koristi kriptografiju ili zahtijeva visoku kvalitetu entropije i slučajnosti.

Najčešća područja primjene Quantis generatora:

- bankarstvo – rješenja za šifriranje koja nude Quantis generatori omogućuju bankama implementaciju platforme za šifriranje na razini cijele tvrtke sa sigurnosnim pravilima kojima je lako upravljati
- podatkovni centri – Quantis generatori osiguravaju prijenos podataka velikim brzinama bez smanjenja redudancije potrebne za sigurnosnu kopiju podatkovnog centra
- lutrije i kockanje – švicarska i francuska lutrija, PokerMatch *online* platforma za igranje pokera i mnogi vodeći sustavi za klađenje koriste Quantis generatore
- kriptografija – generatori rješavaju osnovni problem svih kriptosustava, a to je potreba za istinskom slučajnosti generiranih brojeva
- telekomunikacije – generatori omogućavaju sigurnosna rješenja za zaštitu podataka u prijenosu preko telekomunikacijskih mreža od postojećih i novih prijetnji; britanski telekom i telekom Južne Koreje koriste Quantis generatore
- istraživanja

## 4.2. „*Ranger*“ samostalni generator

Jedan od novijih generatora koji vrijedi spomenuti je uređaj „*Ranger*“, odnosno „*R4nG3r*“. To je samostalni (engl. *stand-alone*) generator slučajnih brojeva koji je razvio projektni tim s Fakulteta elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, a temelji se na teoriji kaosa. Prema [12] teorija kaosa opisuje ponašanje nekih nelinearnih dinamičkih sustava koji se pod određenim uvjetima ponašaju na prividno nepredvidljiv način, odnosno izvode fenomen poznat kao kaos. Cilj je ove teorije u naizgled nasumičnim podatcima pronaći temeljni poredak.

Ovaj je uređaj proizašao kao nastavak projekta BIOCRO PoC koji je realiziran 2013. godine, a u kojem su prvi put realizirali ideju upotrebe teorije kaosa u kriptiranju. Uz konstantni rad na daljnjem razvoju ove tehnologije koja bi omogućila razmjenu poruka u nesigurnom okruženju i zaštitu podataka, došlo je do nastanka uređaja „*Ranger*“. O kvaliteti ovog uređaja govori i činjenica da je „*Ranger*“ nakon detaljnih testiranja dobio uvjerenje Zavoda za sigurnost informacijskih sustava Republike Hrvatske o primjenjivosti u kriptografskim sustavima za zaštitu nacionalnih klasificiranih podataka svih stupnjeva tajnosti. [13]

Uređaj sadrži hardverski i softverski kriptografski sustav pri čemu oba generiraju kaotični signal. Hardverski sustav „*Ranger*“ uređaja generira slučajne brojeve pomoću teorije kaosa i Chuinog elektroničkog kruga. Chua krug sastoji se od standardnih komponenti (otpornici, kondenzatori, induktori) i proizvodi kaotične oscilacije, tj. u njemu su naponi i struje kaotični. Prema [13] iz Chuinog kruga uzorkuju se podaci kao nizovi od maksimalno 8 znamenaka. Sekvenca koja se dobije teoretski je deterministička. No s obzirom na to da su kaotične sekvence neperiodične i osjetljive na odstupanja parametara, a u fizičkoj realizaciji Chuinog kruga uvijek ima odstupanja parametara (npr. promjena parametara zbog odstupanja temperature), može se reći da je sekvenca praktički nepredvidljiva. Zbog postojanja inherentnog determinizma sekvence, korištenjem CBC metode (engl. *Cipher Block Chaining)* uvodi se dodatna nasumičnost.

Softverski sustav nudi programsko rješenje za generiranje ključeva koje omogućuje odabir tipa sekvence koja će biti generirana, dakle hoće li biti binarna ili numerička. Postoji mogućnost odabira između dvaju prijenosnih medija za pohranu datoteke koja sadrži sekvencu. Također, moguće je odabrati različit broj datoteka koje će biti generirane. Svaka binarna datoteka prolazi razna testiranja uključujući 7 standardnih NIST testova. [13]

„*Ranger*“ je *state-of-the-art* uređaj čiji se slučajni generirani brojevi koriste kao ključevi ostalih kriptografskih sustava, stoga ga mogu primijeniti sve tvrtke ili institucije koje se bave zaštitom klasificiranih podataka i sigurnom komunikacijom.


**Sl. 4.4.** „*Ranger*“ *uređaj* [13]

## 4.3. Random.org servis

Servis *Random.org* omogućava dostupnost slučajno generiranih brojeva svim korisnicima interneta. Ovaj je servis kreirao Mads Haar, profesor informatike, 1998. godine, a koristi se za izvlačenje dobitnika lutrije ili nagradnih igara, za vođenje *online* igrica, znanstvenu primjenu, umjetnost i glazbu. Na slici 4.5. prikazan je jedan od alata *Random.org* servisa, generator slučajnih brojeva koji omogućuje odabir početne i krajnje vrijednosti unutar kojih će se generirati slučajni broj. [14]


**Sl. 4.5.** *Generator slučajnih brojeva Random.org servisa* [14]

Generiranje slučajnih brojeva temelji se na korištenju atmosferskog šuma zabilježenog od strane nekolicine postavljenih radija. Svaki radio generira približno 12 000 bita po sekundi. Nasumični bitovi koji proizlaze iz radija koriste se kao temelj različitih alata na servisu. Svaki put kada klijent zatraži slučajne brojeve, bacanje novčića ili neku drugu uslugu, iskoristi se određeni broj generiranih bitova. Koliko će točno bitova biti iskorišteno, ovisi o tome koja usluga se koristi.

Naprimjer, virtualno bacanje jednog novčića preko alata *Coin Flipper* iskoristi točno jedan bit. Ako je bit jednak 1, novčić se okrene na prednju stranu koja obično predstavlja *glavu,* a ako je bit jednak 0, novčić se okreće na obrnutu stranu, odnosno *rep*. Prilikom korištenja drugih usluga, ukoliko nam je potreban veliki broj slučajnih brojeva, korisno je znati koliko će to slučajnih bitova zahtijevati. Količina potrebnih bitova utječe na to koliko će se brzo generirati slučajni brojevi, a *Random.org* razvio je sustav koji ograničava korisnike na količinu bitova koju mogu iskoristiti dnevno.

Ugrožavanje rada generatora servisa *Random.org* moguće je ukoliko postoji razašiljanje signala od strane napadača na frekvencijama koje koristi *Random.org*. No to nije toliko jednostavno iz više razloga. Prvo, frekvencije koje se koriste nisu poznate javno, pa bi napadač morao razašiljati na svim frekvencijama svih pojasa koji se koriste za FM i AM razašiljanje. Drugo, napad se ne može pokrenuti s bilo kojeg mjesta na svijetu, nego samo dovoljno blizu generatora. *Random.org* trenutno ima radioprijemnike u više različitih država što bi otežalo koordinaciju ove vrste napada. I na kraju, ako bi napadač uistinu uspio u razašiljanju regularnih signala (npr. perfektni sinusni signal) na točno odgovarajućim frekvencijama s prave lokacije, statistika u stvarnom vremenu *Random.*org servisa vrlo će brzo primijetiti pad u kvaliteti. [14]

Statistika u stvarnom vremenu konstantno provodi provjere rada generatora uključujući provjeru razine međuspremnika, čistoću i pouzdanost izvora te razne testove iz NIST (engl. *National Institute of Standards and Technology*) paketa.

# 5. PROGRAMSKO RJEŠENJE

Izradit će se dva rješenja za generiranje pseudoslučajnih brojeva, gdje će jedno rješenje biti implementacija već dobro utemeljenog LCG generatora, dok će drugo rješenje biti implementacija vlastitog generatora. Za implementaciju rješenja korišten je alat JSBin [15]. U potpoglavlju 5.1. opisat će se implementacija i rezultati testiranja LCG generatora na konačnom skupu parametara. U potpoglavlju 5.2. opisat će se implementacija i rezultati testiranja vlastitog generatora. U potpoglavlju 5.3. dana je analiza testiranja oba rješenja te su dani određeni zaključci.

## 5.1. Implementacija i testiranje LCG generatora

U ovome je potpoglavlju opisana izrada i testiranje LCG generatora na konačnom skupu parametara. U opisnom jeziku HTML izrađena je jednostavna web stranica koja sadrži element *canvas* dimenzija 600 x 600 piksela. U skriptnom jeziku JavaScript napisan je LCG generator te funkcija koja u ovisnosti o generiranom broju u rasponu 0.0 do 1.0 ostavlja trag na elementu *canvas* (programski kod 5.1.).

```
1.  var a = 1664525,
2.      c = 1013904223,
3.      m = Math.pow(2, 32),
4.      seed = 12234;
5.
6.  function lcgRand() {
7.    seed = (a * seed + c) % m;
8.    return seed;
9.  }
10.
11. function lcgRandFloat() {
12.   return lcgRand() / m;
13. }
14.
15.
16. var canvas = document.getElementById(„canvas“),
17.     context = canvas.getContext(„2d“);
18.
19. var y = 0;
20. draw();
21. function draw() {
22.   for(var x = 0; x < 600; x++) {
23.     if(lcgRandFloat() < 0.5) {
24.       context.fillRect(x, y, 1, 1);
25.     }
26.   }
27.   y++;
28.   if(y < 600) {
29.     requestAnimationFrame(draw);
30.   }
31. }
```

**Programski kod 5.1.** *LCG generator u skriptnom jeziku JavaScript*

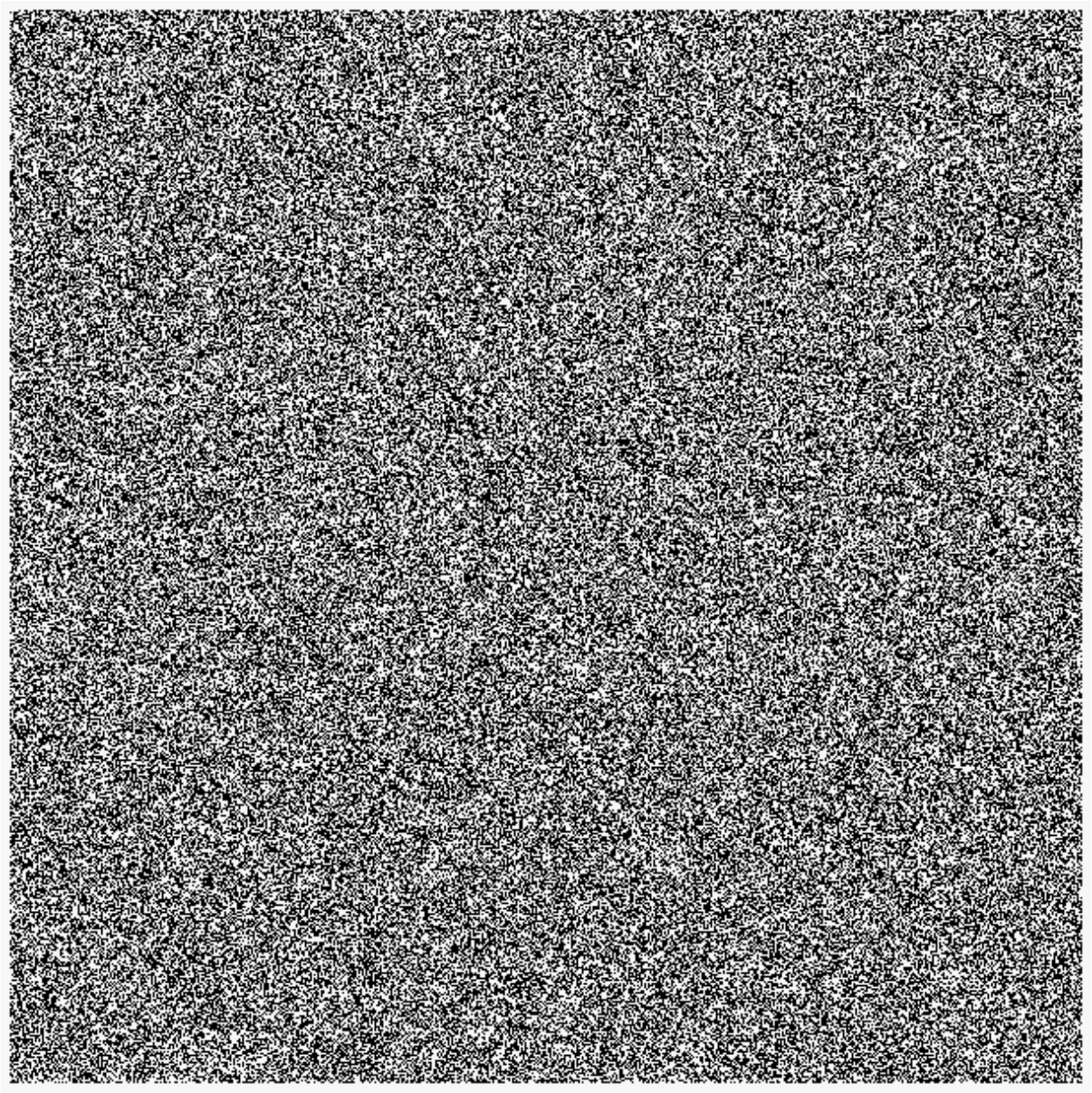
Na početku koda deklarirane i inicijalizirane su varijable (*a, c, m, seed*) koje se koriste u LCG generatoru. Zatim je napisana funkcija *lcgRand()* koja implementira formulu (3-1) LCG generatora opisanog u potpoglavlju 3.2. Funckija *lcgRandFloat()* vraća generirani nasumični broj u rasponu 0.0 do 1.0. Zatim se deklarira varijabla *canvas* koja dohvaća element *canvas* iz HTML koda. Funkcija *draw()* služi za iscrtavanje uzoraka generiranja. Ukoliko je rezultat funkcije *lcgRandFloat()* manji od 0.5, na elementu *canvas* iscrtava se crna točka dimenzija 1 x 1 px, inače se ostavlja prazno te se pomiče za 1 px.

Za testiranje rada LCG generatora napravljeno je 5 skupova parametara (*a, c, m, seed*) koji su prikazani u tablici 5.1.

**Tab. 5.1.** *Tablica skupova parametara ( a, c, m, seed) LCG generatora*

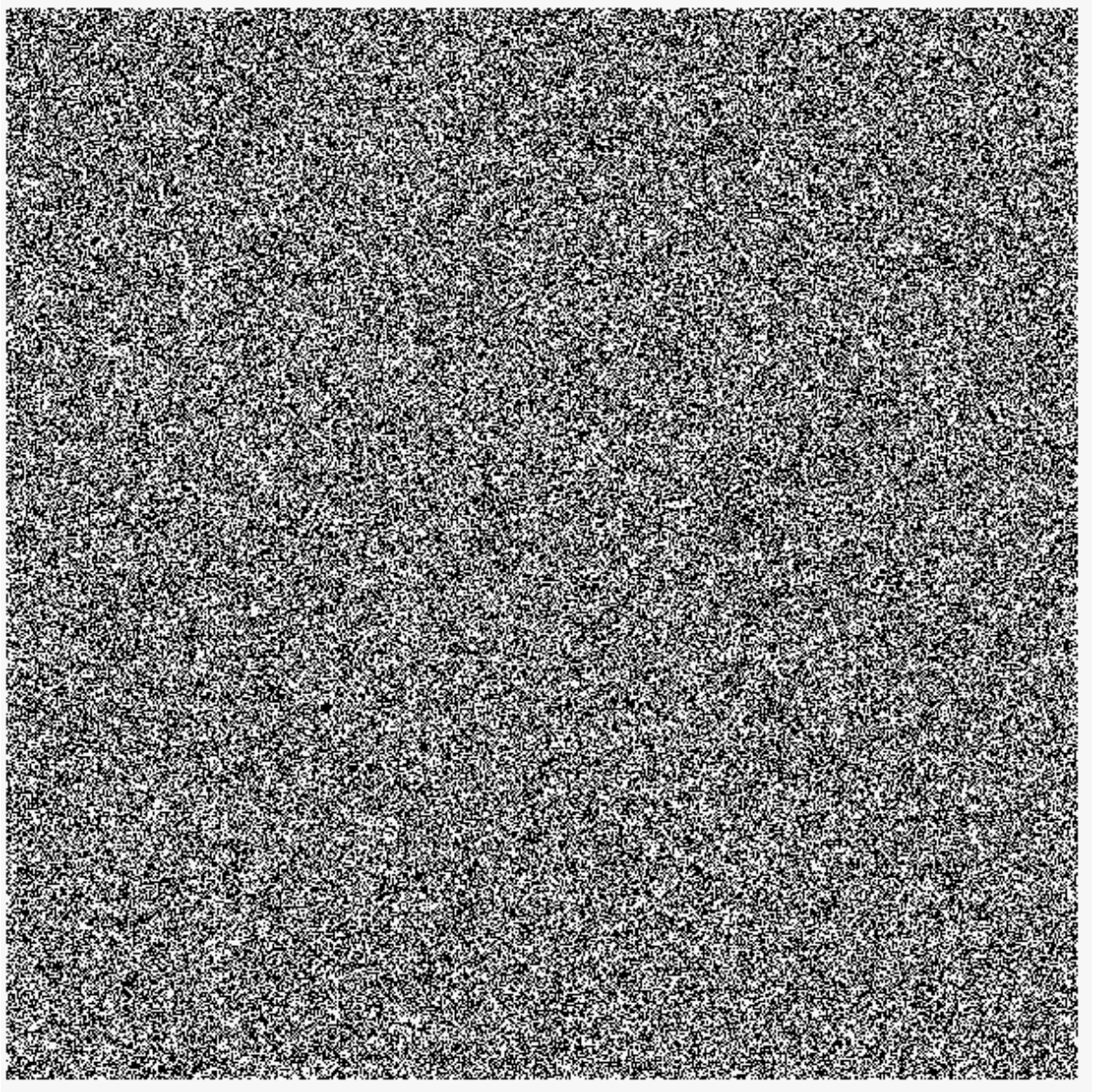
| | **A)** | **B)** | **C)** | **D)** | **E)** |
|---|---|---|---|---|---|
| ***A*** | 1664525 | 1234567 | 10655142 | 1 | 1022911421 |
| ***C*** | 1013904223 | 9876543210 | 1045361522 | 5134514 | 1048361521 |
| ***M*** | $2^{32}$ | $2^{32}$ | $2^{31}$ | $2^{31}$ | $2^{31}$ |
| ***Seed*** | 12234 | 15071996 | 19835713 | 25031996 | 1 |

Za svaki skup podataka A-E prikazan je rezultat u obliku slike rezolucije 600 x 600 px (slika 5.1. – slika 5.5.), odnosno svakim pokretanjem generirano je 360000 nasumičnih brojeva između 0.0 i 1.0.



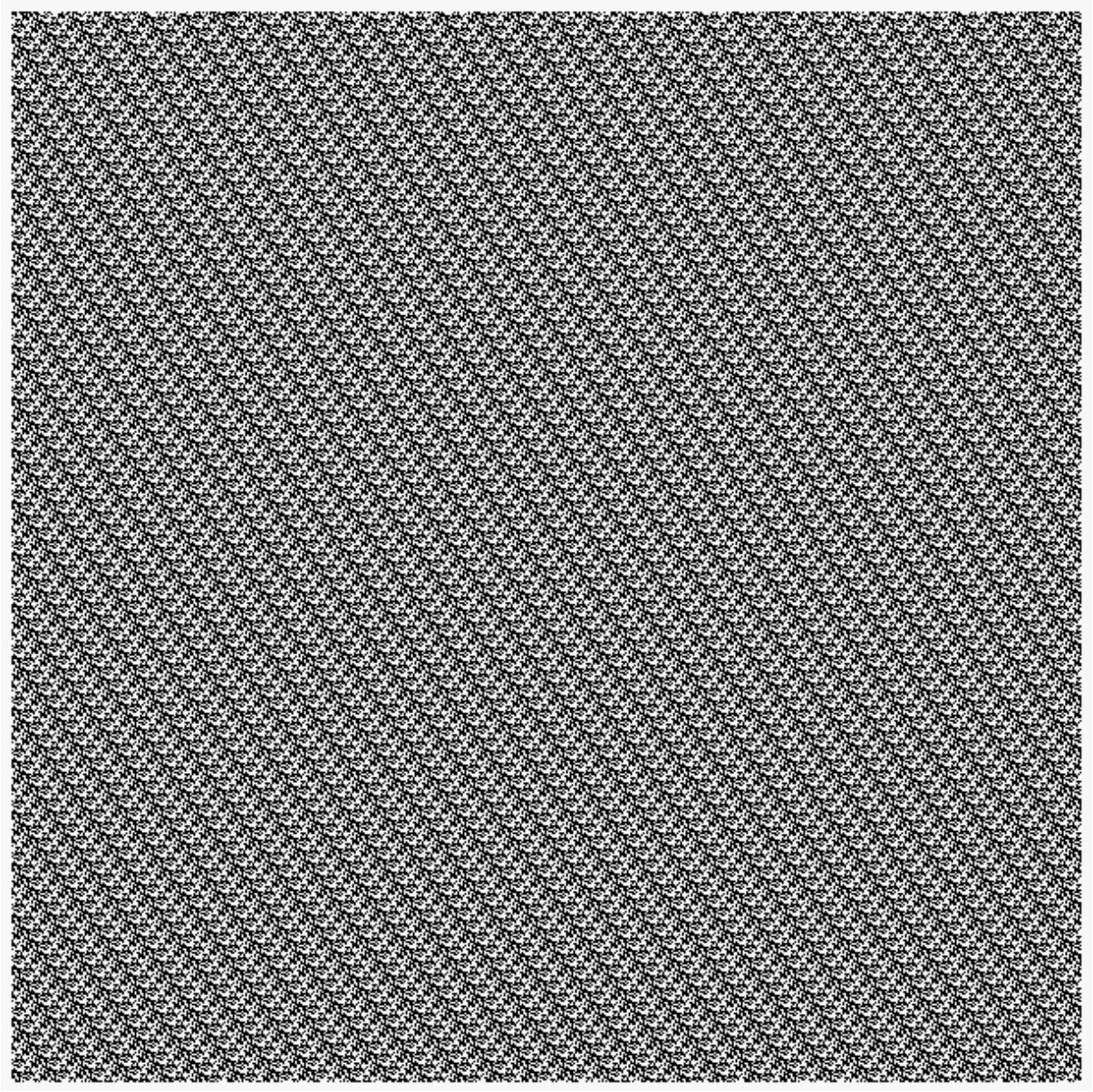
**Sl. 5.1.** *Rezultat izvršavanja LCG generatora za skup parametara A*

Prema slici 5.1. za skup parametara A može se vidjeti da nakon generiranja 360 000 brojeva nije vidljiv uzorak u generiranju, čime se može zaključiti da su brojevi nakon ovolikog broja iteracija nasumični.



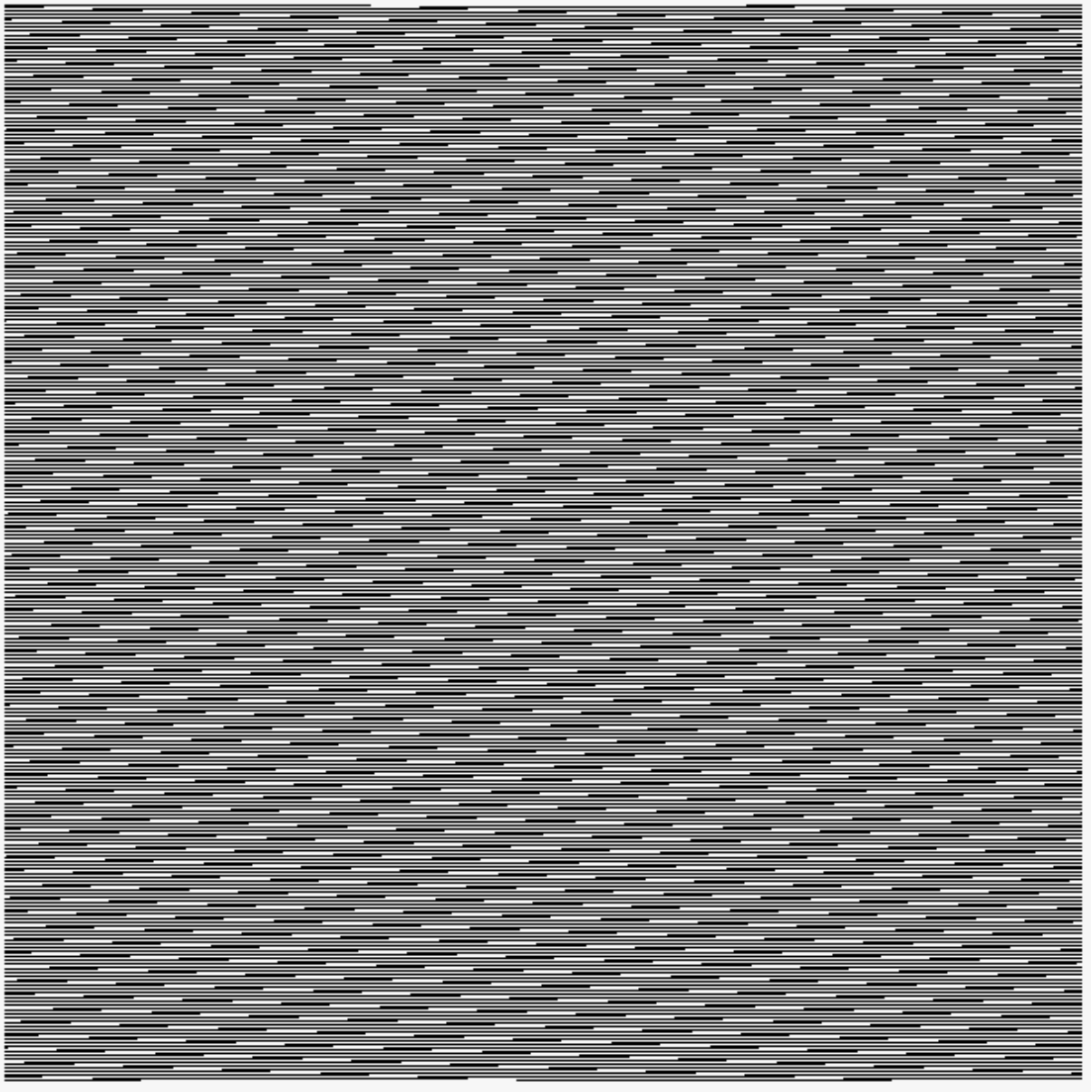
**Sl. 5.2.** *Rezultat izvršavanja LCG generatora za skup parametara B*

Za skup parametara B također se može vidjeti da ne postoji određeni uzorak generiranja (slika 5.2.), što znači da su brojevi nakon ovolikog broja iteracija nasumični i da su početni parametri dobro odabrani.



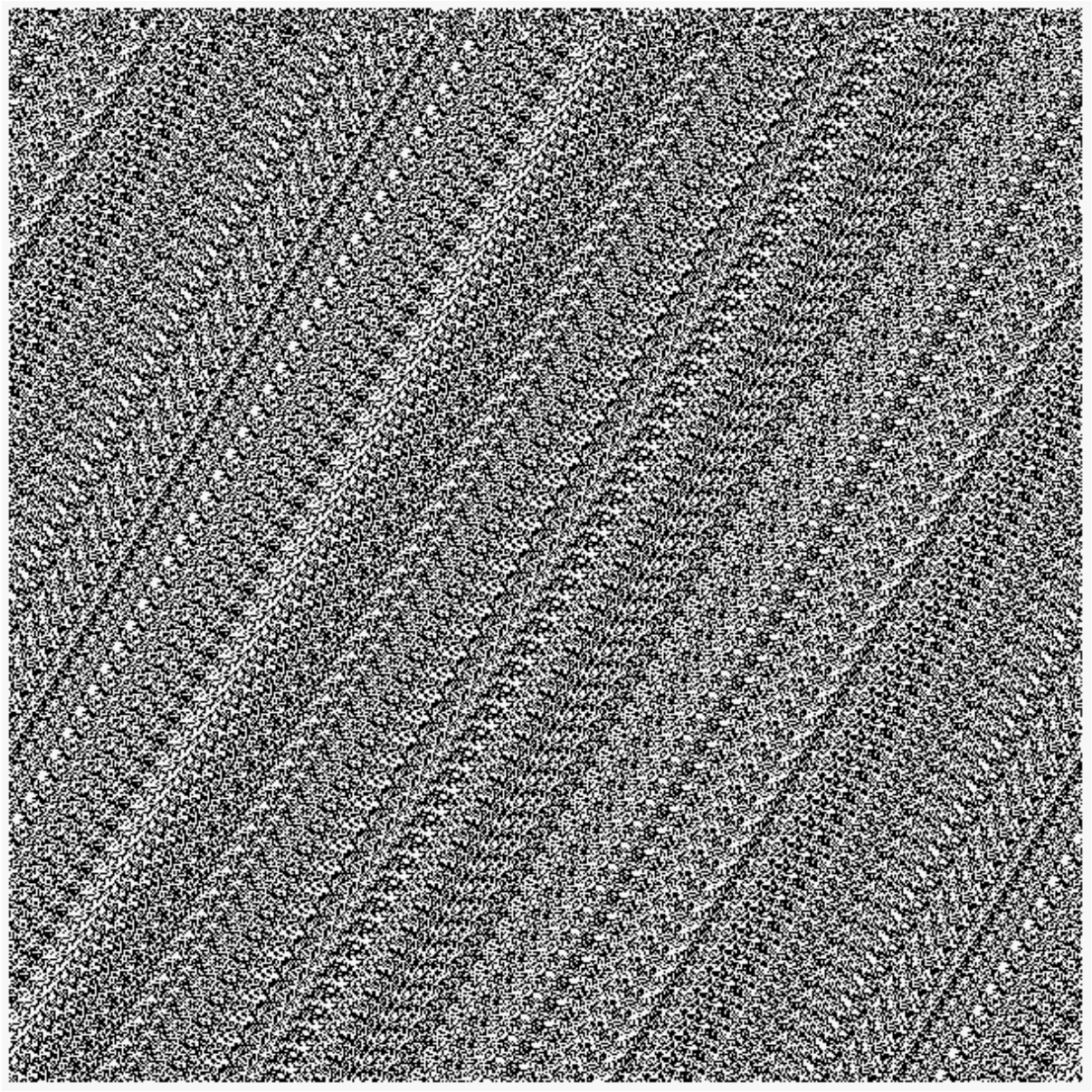
**Sl. 5.3.** *Rezultat izvršavanja LCG generatora za skup parametara C*

U odnosu na skupove parametara A i B, za skup parametara C vidljiv je određeni uzorak u generiranju brojeva, što znači da oni nisu dovoljno nasumični, odnosno da se ciklički ponavljaju.



**Sl. 5.4.** *Rezultat izvršavanja LCG generatora za skup parametara D*

Prema slici 5.4. za skup parametara D također je vidljiv uzorak generiranja brojeva, dakle početni parametri nisu dobro odabrani.



**Sl. 5.5.** *Rezultat izvršavanja LCG generatora za skup parametara E*

Kao i za skupove parametara C i D, tako se i za skup parametara E može vidjeti uzorak u generiranju što generirane brojeve čini nedovoljno nasumičnima.

Analizom rada generatora sa skupovima A-E može se zaključiti da, unatoč tomu što je LCG generator općepriznat i korišten, na njegovu pouzdanost i kvalitetu generiranja slučajnih brojeva uvelike utječe početni skup parametara (*a, c, m, seed*) koji nije jednostavno odabrati. Potrebno je pažljivo i pomno izabrati parametre kako bi dobiveni generirani brojevi bili dovoljno slučajni te kako bi period niza bio dovoljan velik da se ponavljanje brojeva rijetko događa. Svaki vidljiv uzorak u generiranju brojeva predstavlja sigurnosnu slabost generatora i time ga čini neadekvatnim za primjenu.

## 5.2. Implementacija i testiranje vlastitog generatora

U ovom potpoglavlju opisana je izrada i testiranje vlastitog generatora. Cilj je razviti vlastiti generator na temelju analize metoda opisanih u poglavlju 3. te analizirati njegov rad.

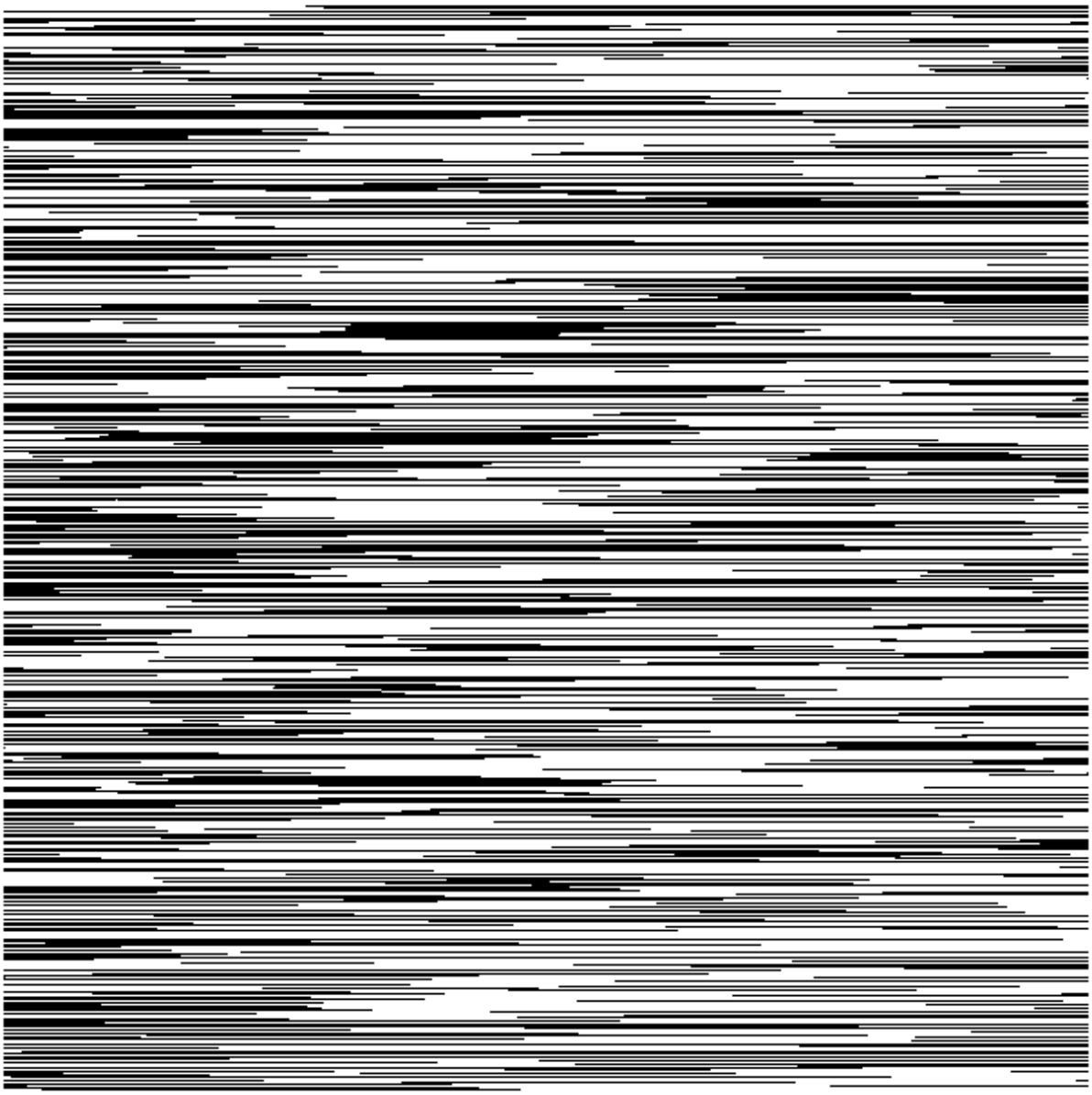
Vlastiti generator također je implementiran u JavaScript skriptnom jeziku (programski kod 5.2.).

```
1.  function myRand() {
2.    d = new Date();
3.    n = d.getTime();
4.    n = ((n/1000 * 214013135613 + 253101114221) >> 10) & 131071;
5.    return ((n%2642462462) + 1);
6.  }
7.
8.  function myRandFloat() {
9.    return myRand() / 131071;
10. }
11.
12.
13. var canvas = document.getElementById("canvas"),
14.     context = canvas.getContext("2d");
15.
16. var y = 0;
17. draw();
18. function draw() {
19.   for(var x = 0; x < 600; x++) {
20.     if(myRandFloat() < 0.5) {
21.       context.fillRect(x, y, 1, 1);
22.     }
23.   }
24.   y++;
25.   if(y < 600) {
26.     requestAnimationFrame(draw);
27.   }
28. }
```

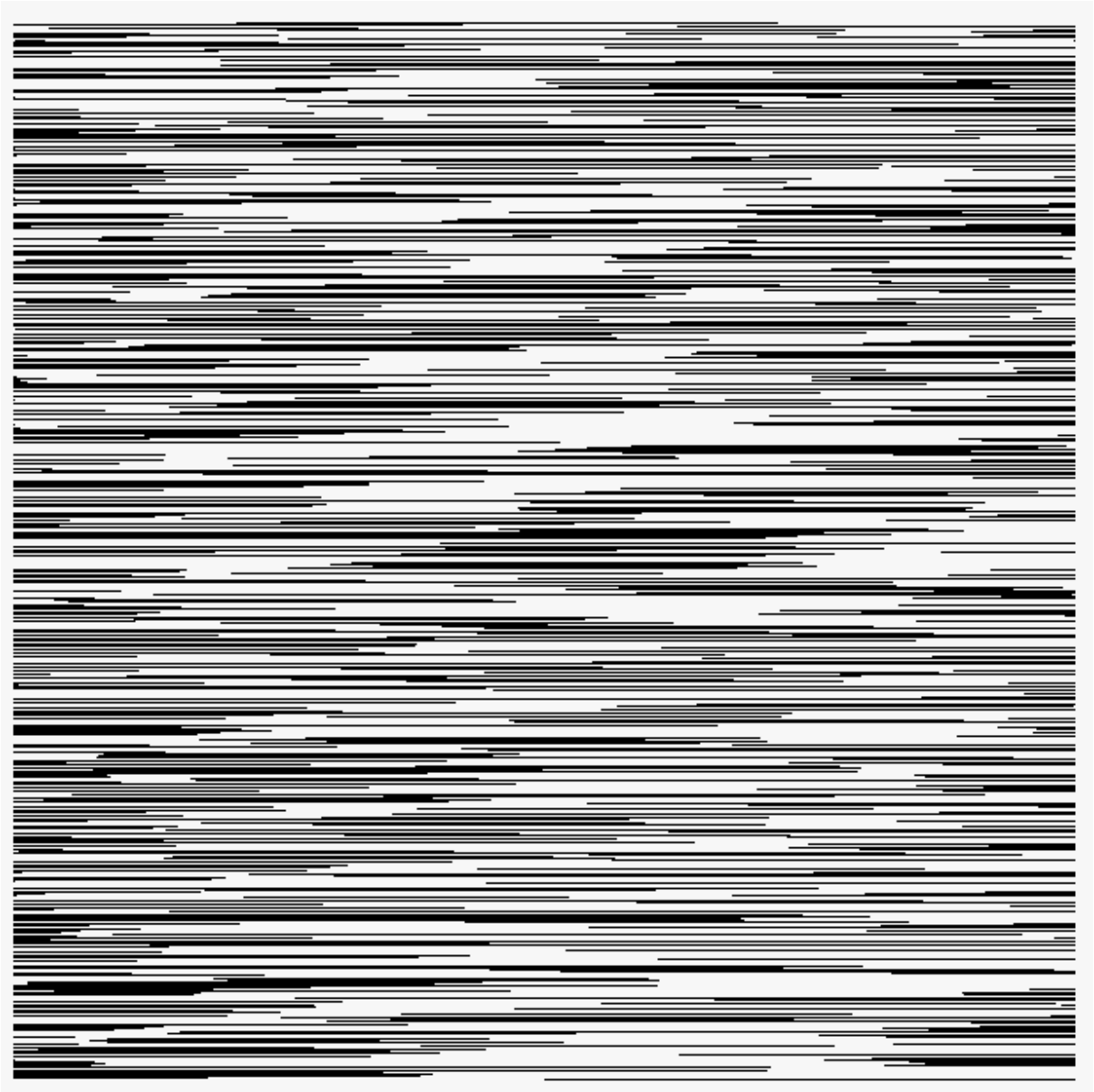
**Programski kod 5.2.** *Programsko rješenje vlastitog generatora*

Generira se 360000 brojeva pomoću funkcije *myRand()*. Kao početni parametar za generiranje svakog broja uzima se broj sekundi koji je protekao od ponoći 1. 1. 1970. Zatim se taj broj sekundi množi s nasumično odabranim velikim cijelim brojem te se dodaje drugi nasumično odabrani cijeli broj. Rezultat se dijeli s $2^{10}$ te se zatim binarno množi s $2^{17}$. Uzima se ostatak pri dijeljenju s nasumično odabranim velikim brojem te se dodaje 1. Rezultat funkcije *myRand*() dijeli se s $2^{17}$ da bi se dobio broj između 0.0 i 1.0.

Na slikama 5.6. i 5.7. prikazani su rezultati izvršavanja vlastitog generatora.



**Sl. 5.6.** *Primjer rezultata izvršavanja vlastitog generatora*



**Sl. 5.7.** *Primjer rezultata izvršavanja vlastitog generatora*

Iz prikazanih primjera rezultata može se primijetiti da na prvi pogled nije vidljiv obrazac ponavljanja. Iako se kao početni parametar za generiranje svakog broja uzima broj sekundi koji je protekao od ponoći 1. 1. 1970., što je dovoljno nasumičan podatak, programsko se rješenje sastoji od poprilično jednostavnih funkcija i nakon nekog bi se vremena pojavio vidljiv uzorak ponavljanja, što nije dobra karakteristika generatora. Na prvu se čini da ima bezbroj mogućnosti za generiranje slučajnih brojeva, no potrebno je vremena i truda kako bi se osmislilo kvalitetno programsko rješenje koje zadovoljava uvjete nasumičnosti.

## 5.3. Analiza testiranja oba rješenja

Testiranje implementacije LCG generatora kroz skupove parametara A-E pokazalo je da je izbor početnih parametara ključan pri radu LCG generatora. Odabrane su različite skupine parametara, od većih do manjih vrijednosti, pri čemu su dobiveni različiti rezultati generiranih brojeva. Kod skupova parametara A i B, parametri su relativno dobro odabrani što se može zaključiti prema slikama 5.1. i 5.2. na kojima nisu vidljivi uzorci generiranja brojeva. U ostalim slučajevima, kod skupova parametara C, D i E, jasno su vidljivi uzorci generiranja brojeva, što znači da brojevi nisu nasumični, odnosno da su početni parametri loše odabrani.

Implementacija vlastitog rješenja također je izrađena u JavaScript skriptnom jeziku. Iako na prikazanim rezultatima (slika 5.6. i 5.7.) nisu odmah vidljivi uzorci generiranja brojeva, vlastito rješenje ne generira toliko nasumične brojeve kao LCG. Funkcije koje se koriste nisu dovoljno razrađene, odnosno nakon nekog bi vremena uzorci generiranja brojeva bili vidljivi.

Iz analize i testiranja može se zaključiti da je vrlo zahtjevno osmisliti kvalitetni vlastiti generator koji bi generirao slučajne brojeve. Čak i kod generatora koji su već dobro razrađeni i općepoznati, nije jednostavno odabrati početne uvjete, a bez dobro odabranih početnih uvjeta, generirani brojevi neće biti dovoljno nasumični.

# 6. ZAKLJUČAK

U prvom dijelu završnog rada opisan je pojam slučajnih brojeva općenito i podjela na „istinske“ i pseudoslučajne brojeve. „Istinski“ slučajni brojevi generiraju se na temelju prirodnih pojava ili fizikalnih procesa nepredvidljive prirode, dok su pseudoslučajni brojevi generirani računalno koristeći razne algoritme i formule.

Prikazan je razvoj metoda i izvora generatora kroz povijest. Ljudi su koristili slučajne brojeve u razne svrhe od davnina, počevši od bacanja kockice, novčića pa sve do razvoja prvih tablica slučajnih brojeva. Razvojem računalne tehnologije došlo je i do razvoja raznih metoda za generiranje slučajnih brojeva koje se temelje na algoritmima i matematičkim formulama (metoda sredine kvadrata, LCG, LFG, Mersenne Twister). U tim je metodama važno dobro odabrati početne parametre kako ne bi došlo do vidljivog uzorka prilikom generiranja brojeva.

U današnjem se društvu slučajni brojevi koriste za igre na sreću, simulacije, lutrije, računalne igrice, itd. Zaključeno je da najvažniju primjenu imaju u kriptografiji i sigurnosti računalnog sustava. Slučajni su brojevi temelj osiguravanja telekomunikacijskih mreža, tehnologija poput *blokchain*-a i općenito tajnosti komunikacije. Kao najbolji primjer generatora za svrhu kriptografije, naveden je kvantni generator Quantis koji koristi kvantnu optiku kao izvor istinske slučajnosti. Važno je spomenuti i moderni *state-of-the-art* generator „*Ranger*“ kao primjer kriptografski primjenjivog generatora i servis *Random.org* koji pruža razne alate za generiranje slučajnosti.

Programsko se rješenje sastoji od dva dijela. Prvi je dio implementacija dobro poznatog LCG generatora, a drugi implementacija vlastitog generatora. Analizom i testiranjem utvrđeno je da je za pouzdanost LCG generatora izrazito važan odabir početnih parametara. Ukoliko nisu dobro odabrani, brojevi nisu dovoljno nasumični. Implementacija vlastitog izvora izrađena je u JavaScript skriptnom jeziku, testirana i analizirana. Zaključeno je da ni vlastita metoda nije zadovoljavajuća i da nije jednostavno stvoriti potpuno novu metodu.

# LITERATURA


[1] I. Urbiha, „*Generiranje niza pseudoslučajnih brojeva*“, Matematičko-fizički list, 2011.

[2] S. Jurić, „*Generatori pseudo-slučajnih brojeva*“, FER, Zagreb, 2001., dostupno na: http://sigurnost.zemris.fer.hr/random/2001_juric/#CSPRNG, pristup ostvaren 30. kolovoza 2019.

[3] C. Hoffman, „*How computers generate random numbers*“, How-to Geek, 2019., dostupno na: https://www.howtogeek.com/183051/htg-explains-how-computers-generate-random-numbers/, pristup ostvaren 30. kolovoza 2019.

[4] C. Tashian, „*A brief history of random numbers*“, freeCodeCamp, 2017., dostupno na:

https://www.freecodecamp.org/news/a-brief-history-of-random-numbers-9498737f5b6c/, pristup ostvaren 30.kolovoza 2019.

[5] N. Petrović, „*Alati za statističko testiranje nizova pseudoslučajnih brojeva*“, FERIT, Sveučilište Josipa Jurja Strossmayera, Osijek, 2018.

[6] M. Vidović, N. Bijelić, D. Popović, predavanje s kolegija *Objektno orijentisana simulacija*, smjer Logistika, Prometni fakultet, Beograd, 2019.

[7] B. Buchanan, „*For the love of computing: the Lagged Fibonacci Generator - where nature meet random numbers*“, Medium, 2018., dostupno na: https://medium.com/asecuritysite-when-bob-met-alice/for-the-love-of-computing-the-lagged-fibonacci-generator-where-nature-meet-random-numbers-f9fb5bd6c237, pristup ostvaren 1.rujna 2019.

[8] D. Marinčić, „*Generiranje pseudoslučajnih brojeva*“, Odjel za matematiku, Sveučilište Josipa Jurja Strossmayera, Osijek, 2019.

[9] G. Gravier, „*The crucial role of quantum random number generation in securing blockchains*“, ID Quantique SA, 2018., dostupno na: https://www.idquantique.com/the-crucial-role-of-quantum-random-number-generation-in-securing-blockchains/, pristup ostvaren 3. rujna 2019.

[10] „*What is the Q in QRNG?“,* ID Quantique SA, 2019., dostupno na: https://marketing.idquantique.com/acton/attachment/11868/f-0226/1/-/-/-/-/What%20is%20the%20Q%20in%20QRNG_White%20Paper.pdf, pristup ostvaren 3. rujna 2019.

[11] „*Quantis certifications*“, ID Quantique SA, dostupno na: http://marketing.idquantique.com/acton/attachment/11868/f-0117/1/-/-/-/-/Quantis%20Certifications%20Collection.pdf, pristup ostvaren 3. rujna 2019.

[12] H. Vučić, „*Kaos i Lorenzov sustav*“, Zavod za matematiku, Fakultet kemijskog inženjerstva i tehnologije, Sveučilište u Zagrebu, Zagreb, 2011.

[13] K. Miličević, „*Projektni tim s FERIT-a razvio „Ranger“ - Stand-alone generator slučajnih brojeva temeljen na teoriji kaosa*“, FERIT, Sveučilište Josipa Jurja Strossmayera, Osijek, 2019., dostupno na: https://www.ferit.unios.hr/fakultet/novosti-i-dogadanja/6926

[14] Random.org, dostupno na: https://www.random.org/, pristup ostvaren 5. rujna 2019.

[15] JSbin alat, dostupno na: https://jsbin.com/

# SAŽETAK

U završnom radu teorijski su opisani primjeri izvora slučajnih brojeva, njihova upotreba i važnost. Prikazan je vlastiti primjer izvora slučajnih brojeva. Opisana su svojstva slučajnih brojeva općenito te podjela slučajnih brojeva na „istinske“ i pseudoslučajne. Prikazano je korištenje slučajnih brojeva kroz povijest i razvoj metoda za generiranje slučajnih brojeva (metoda sredine kvadrata, LCG, LFG, Mersenne Twister). Kroz razne primjere primjene slučajnih brojeva zaključeno je da je najvažnija primjena u kriptografiji i da su za tu primjenu najpogodniji „istinski“ slučajni brojevi. Ukoliko se koriste pseudoslučajni brojevi, postojala bi sigurnosna slabost i mogućnost ugrožavanja sustava. Dane su dvije implementacije izvora slučajnih brojeva izrađene u JavaScript skriptnom jeziku, gdje je jedna od implementacija LCG generator slučajnih brojeva, dok je druga implementacija vlastite metode. Testirana su oba rješenja i zaključeno je da čak i dobro utemeljen i priznat generator poput LCG-a nije zadovoljavajući ako se ne odaberu dobri početni uvjeti. Također, zaključeno je da ni vlastita metoda nije zadovoljavajuća te da nije lako stvoriti potpuno novu metodu.

**Ključne riječi:** istinski slučajni brojevi, izvor slučajnih brojeva, kriptografija, kvantni generator, pseudoslučajni brojevi

## ABSTRACT

**Title:** Sources of random numbers and their use

In this bachelor thesis, different examples of random number generators, their use and importance were described. An own example of a random number generator was presented. The properties of random numbers, as well as the breakdown into “true” and pseudorandom numbers were described. The use of random numbers through the history and development of various methods for generating random numbers (mid-square method, LCG, LFG, Mersenne Twister) is presented. Through various examples of the use of random numbers, it has been concluded that the most important application is in cryptography and that "true" random numbers are the most suitable choice for this application. If pseudorandom numbers are used, there would be a security weakness and the potential for compromising the system. Two examples of random number generators were created in JavaScript, one being an implementation of LCG and the other being an own implementation of a custom method. Both solutions were tested, and it was concluded that even a well-established and recognized generator like LCG is not satisfactory unless good initial conditions are selected and that a completely new number generator is very hard to make.



**Keywords:** cryptography, quantum generator, pseudorandom numbers, random number generator, true random number

# ŽIVOTOPIS

Ana Babić rođena je 15. srpnja 1996. godine u Osijeku. U Osijeku završava OŠ „Vladimir Becić“ i nakon toga upisuje Prirodoslovno-matematičku gimnaziju. Tijekom srednjoškolskog školovanja aktivno djeluje kao volonter Gradskog društva Crvenog križa Osijek. 2015. godine ostvaruje upis na Fakultet elektrotehnike, računarstva i informacijskih tehnologija, smjer elektrotehnika.

Potpis:

_____

# PRILOG

Na optičkom disku priloženom uz ovaj završni rad nalazi se digitalna verzija rada u .docx i .pdf formatu. Također, nalazi se i izvorni kod u direktoriju Programski kod.