



UNIVERZITET SINGIDUNUM
FAKULTET ZA INFORMATIKU I RAČUNARSTVO

Primena teorije informacija u procesu generisanja kriptoloških ključeva

- diplomski rad -

Mentor:

prof. dr *Mladen Veinović*

Kandidat:

Nikola Lazić

Beograd, 2016.



FAKULTET ZA INFORMATIKU I RAČUNARSTVO

UNIVERZITET SINGIDUNUM

FAKULTET ZA INFORMATIKU I RAČUNARSTVO

Beograd, Danijelova 32

Kandidat: Nikola Lazić

Broj indeksa: 2012/200460

Smer: Informatika i računarstvo

Tema: Primena teorije informacija u procesu generisanja kriptoloških ključeva

MENTOR

Datum odobrenja rada:
Beograd, _____.____.____.

Prof. dr. Mladen Veinović

DEKAN

Prof. dr. Dragan Cvetković

Primena teorije informacija u procesu generisanja kriptoloških ključeva

Sažetak:

U ovom radu bavićemo se fenomenom slučajnosti u domenu teorije informacija i primene u kriptološkim mehanizmima, koja je od suštinskog značaja za informacionu bezbednost. Savremeni sistemi zaštite ne mogu uvek da se oslanjaju na pseudo-generatore slučajnosti - PRNG i to je jedan od glavnih razloga zašto ćemo se fokusirati na istinito slučajne generatore - TRNG. Rigoroznom informacionom analizom procenićemo kvalitet nekog informacionog izvora i stepena neodređenosti koji zadovoljava uslove za generisanje ključeva visoke entropije i neprediktibilnosti. Imajući na umu osobine stacionarnosti informacionih izvora napravićemo pregled najpoznatijih metoda za postdigitalnu obradu generisanih sekvenci.

Ključne reči: teorija informacija, slučajnost, entropija

The usage of information theory in the process of generation cryptographic keys

Abstract:

In this thesis we will concentrate on the phenomenon of randomness in the domain of information theory and implementation in cryptologic mechanisms, which are of essential value for information security. Modern protection systems cannot always rely on pseudo-random generators - PRNG and that is one of the main reasons why we are going to focus on true random generators - TRNG. With extensive information analysis we are going to estimate the quality of a information source and the degree of indetermination which will satisfy the conditions for high entropy key generation and unpredictability. Having in mind the traits of information source immobility, we are going to compose a review of most noted methods for post-digital distillation of generated sequences.

Keywords: information theory, randomness, entropy

Sadržaj

1	Uvod	- 3 -
1.1	Predmet istraživanja	- 3 -
1.2	Doprinos rada	- 3 -
1.3	Struktura rada	- 3 -
2	Pregled u oblasti istraživanja	- 4 -
2.1	Slučajni ključevi	- 4 -
2.2	Generatori kriptoloških ključeva	- 4 -
2.2.1	Linearni kongruentni generatori	- 5 -
2.2.2	Linearni pomerački registri sa povratnom spregom	- 5 -
2.2.3	ANSI X9.17	- 6 -
2.2.4	FIPS 186	- 6 -
2.2.5	RSA	- 7 -
2.2.6	Blum, Blum i Shub	- 7 -
2.2.7	Aditivni generatori	- 8 -
2.2.8	AT&T 7001	- 9 -
2.2.9	Intel TRNG	- 9 -
2.2.10	Oscilacije vazduha u disku računara kao generator slučajnosti	- 9 -
2.3	Testovi slučajnosti – standardi za testiranje slučajnosti	- 10 -
2.3.1	Ispitivanje učestalosti u nizu	- 10 -
2.3.2	Ispitivanje učestalosti u bloku	- 11 -
2.3.3	Ispitivanje uzastopnih ponavljanja istih bitova u nizu	- 12 -
2.3.4	Ispitivanje najdužeg uzastopnog ponavljanja jedinica u bloku	- 12 -
2.3.5	Ispitivanje stanja binarne matrice	- 14 -
2.3.6	Ispitivanje diskretne Furijerove transformacije	- 15 -
2.3.7	Ispitivanje nepreklapajućih uzoraka	- 15 -
2.3.8	Ispitivanje preklapajućih uzoraka	- 16 -
2.3.9	Maurerov univerzalni statistički test	- 17 -

2.3.10 Ispitivanje linearne kompleksnosti	- 18 -
2.3.11 Serijski test	- 19 -
2.3.12 Ispitivanje približne entropije	- 20 -
2.3.13 Ispitivanje slučajnih zbirova	- 20 -
2.3.14 Ispitivanje slučajne digresije	- 21 -
2.3.15 Ispitivanje slučajne promenljive digresije	- 22 -
2.4 Postdigitalna obrada generisanih sekvenci	- 23 -
2.4.1 Mapiranje prelaza	- 23 -
2.4.2 Paritet niza	- 24 -
2.4.3 Operacija XOR	- 24 -
2.4.4 Brza Furijerova transformacija	- 24 -
2.4.5 Kompresija	- 24 -
2.4.6 Kompresija sa Jednosmernim funkcijama	- 25 -
3 Teorijske osnove istraživanja	- 26 -
3.1 Entropija	- 26 -
3.2 Sigurnost kriptosistema	- 27 -
3.3 Ispitivanje slučajnosti	- 28 -
3.4 Generatori slučajnosti	- 29 -
3.5 Pravi slučajni generator (TRNG)	- 30 -
3.6 Pseudo slučajni generator (PRNG)	- 30 -
3.7 Uporedna analiza generatora slučajnosti	- 31 -
3.8 Postdigitalna obrada	- 32 -
3.9 Oblast primene	- 33 -
4 Pregled predloženog rešenja	- 34 -
4.1 Vršenje testa slučajnosti računanjem uzajamne informacije	- 34 -
5 Zaključak	- 37 -
5.1 Predlog za budući rad	- 37 -
6 Literatura	- 38 -
7 Appendix	- 39 -

1 Uvod

Sigurnost savremenih informacionih tehnologija se bazira na izrazito jakom kriptografskom algoritmu, međutim osnova svakog kriptografskog algoritma je ključ. Mnogi sigurnosni protokoli zasnivaju svoju snagu na ključu. Izvori slučajnosti predstavljaju početnu tačku u generisanju kriptoloških ključeva. Od neodređenosti informacionog izvora zavisi kvalitet kriptološkog ključa, a samim tim i kompletan kriptosistem.

Većina šifarskih algoritama je uz pravilnu upotrebu dokazano sigurna. Obično se više pažnje posvećuje dokumentaciji i implementaciji ovih algoritama, dok se manje pažnje posvećuje generatorima slučajnosti. Upravo su generatori slučajnosti ključni u određivanju sigurnosti sistema. Drugim rečima, kvalitet generatora slučajnosti ogleda se u mogućnosti kriptanalitičara da izvrši napad.

Upravo su moderni kriptografski algoritmi i protokoli zasnovani na Kerkohovim (*Auguste Kerckhoffs*) principima koji govore da sigurnost sistema zavisi isključivo od materijala ključa, a ne od strukture sistema. Ovde generatori imaju ključnu ulogu, jer se oni koriste za generisanje materijala ključa. Snaga algoritma je narušena ukoliko napadač uspešno izvrši ekstrakciju bitova ključa iz izlaznih podataka.

1.1 Predmet istraživanja

U ovom radu predstavljen je fenomen slučajnosti kao preduslov za sigurnost informacionog sistema. Rigoroznim analizama procenjuje se kvalitet nekog izvora slučajnosti i određuje se upotrebljivost u kriptografskim sistemima.

1.2 Doprinos rada

Doprinosi ovog rada se ogledaju u skretanje pažnje na neophodnost kvalitetnog izvora slučajnosti kao osnovu za postizanje savršene tajnosti. Predloženi metod za ispitivanje slučajnosti koji određuje nivo predvidivosti generisane sekvence primenom teorije informacija.

1.3 Struktura rada

U prvom delu rada predstavljen je značaj kriptološkog ključa kao osnova sigurnosti kriptosistema. Detaljno su predstavljeni i analizirani generatori kriptoloških ključeva, kao i relevantni testovi slučajnosti. Opisane su najzastupljenije metode postdigitalne obrade koja je ključna u poboljšavanju slučajnosti nekog informacionog izvora.

U drugom delu rada predstavljene su teorijske osnove istraživanja, pojam entropije kao mere slučajnosti u generisanoj sekvenci, postavka savršenog kriptosistema i određivanje uslova za postizanje bezuslovne sigurnosti. Opisani su generatori slučajnosti i oblast njihove primene. Detaljno je opisana uloga i princip rada postdigitalne obrade.

U trećem delu opisan je predloženi metod za testiranje slučajnosti računanjem uzajamne informacije kod slučajno generisanih sekvenci. Izvršena su merenja iz različitih izvora i utvrđena je preciznost predloženog metoda.

2 Pregled u oblasti istraživanja

U ovom poglavlju će biti predstavljeni složeniji generatori kriptoloških ključeva i njihova primena, kao i najpoznatiji metodi postdigitalne obrade i preporučeni testovi slučajnosti. Posebna pažnja se skreće na njihovu neophodnost u ostvarivanju kriptološke sigurnosti u informacionim sistemima. Sigurnost šifarskog algoritma počiva na ključu. Ako je u upotrebi loš algoritam za generisanje ključeva, ceo sistem se dovodi u pitanje. Kriptoanalitičar ne mora vršiti napad na ceo algoritam za šifrovanje već je dovoljno da se usresredi samo na algoritam za generisanje ključeva da bi kompromitovao ceo sistem. Zbog takvih razloga obraća se pažnja na sam proces generisanja ključeva i njihov pravilan odabir.

2.1 Slučajni ključevi

Pojam slučajnosti prilikom generisanja kriptoloških ključeva je osnovni preduslov za siguran kriptosistem, zato su jedni od najboljih ključeva nizovi bitova generisani na slučajan način nekim automatskim postupkom. To znači da ako je dužina ključa 128 bita, svaki mogući 128-bitni ključ mora da bude podjednako verovatan.[4] Da bi se ostvarilo tako nešto, bitovi ključa se moraju generisati iz pouzadnog slučajnog izvora ili korišćenjem generatora pseudoslučajnih bitova koji ispunjava uslove kriptografske sigurnosti. Za proveru slučajnosti generisane sekvence koriste se statistički testovi slučajnosti kako bi smo zaključili da ispitivani generator poseduje željena svojstva i da bi kao takav bio pogodan generator za korišćenje.

2.2 Generatori kriptoloških ključeva

Generator slučajnih sekvenci koji je pogodan za korišćenje u kriptografiji ispunjava uslov da i pored svih poznatih činjenica vezanih za sam generator (šema, algoritam, itd.) reprodukuje potpuno nepredvidive vrednosti. Puno su kavlitetniji od generatora koji se koriste za računarske simulacije. Zbog složenosti algoritma znatno su sporiji od standardnih generatora ali zbog njihove namene u kriptografiji, gde je potrebno generisati mali broj vrednosti, brzina nije od presudnog značaja. Tu spadaju dva osnovna tipa generatora PRNG i TRNG čiji principi rada su detaljno opisani u nastavku ovog poglavlja. Da bi generator bio dovoljno dobar za generisanje ključeva i sve ostale kriptografske primene mora da ispuni sledeća tri uslova:

- Generator deluje kao da je slučajan. Verovatnoća pojave jedne generisane vrednosti treba biti ista za sve vrednosti u prostoru u kome se generiše niz. Što znači da prolazi sve statističke testove slučajnosti.
- Sekvenca je nepredvidiva. To znači da je matematički nemoguće predvideti sledeći slučajan bit jer ne sme da postoji zavisnost između generisanih sekvenci podniza.
- Generator se ne može reprodukovati. Ako se dva puta aktivira generator dobijaju se dve međusobno potpuno nezavisne slučajne sekvence. Generator je dobro konstruisan ukoliko je broj stanja kroz koja on prolazi prilično veliki.

2.2.1 Linearni kongruentni generatori

Linearni kongruentni generatori (engl. *linear congruential generators*) predstavljaju jedne od osnovnih i najpoznatijih algoritama pseudoslučajnih sekvenci. Mogu se predstaviti na sledeći način:

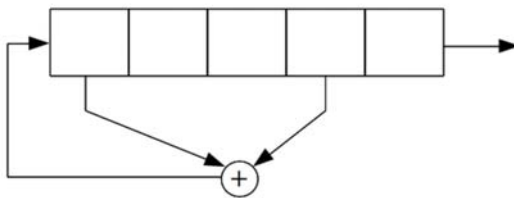
$$X_n = (aX_{n-1} + b) \bmod m$$

gde X_n predstavlja n -ti broj u sekvenci, X_{n-1} predstavlja prethodni broj u sekvenci. Vrednosti a, b, m čine specifične konstante za generator. Tako da a bude množilac, b je priraštaj, a m predstavlja moduo. Ako su vrednosti za a, b, m pravilno izabrane, ovaj generator će biti generator sa najvećom periodom i to ne većom od m . Iako su se pokazali efikasnim i uspevaju da daju dobre rezultate na velikom broju empirijskih testova slučajnosti, ovi generatori ne mogu da se koriste u kriptografske svrhe zbog svoje osobine predvidivosti. [4]

2.2.2 Linearni pomerački registri sa povratnom spregom

Osnovne dve funkcije pomeračkih registara sa povratnom spregom čine funkcija pomeračkog registra i funkcija povratne sprege. Pomerački registar predstavlja sekvencu dužine n bita i kada god je potreban jedan bit, svi bitovi u pomeračkom registru se pomeraju za jedno mesto u desno. Funkcija povratne sprege određuje na osnovu određenih bitova u registru koji bit će se upisati na prvo mesto. Izlazni bit je određen na osnovu početnog stanja i povratne sprege, uglavnom bit najmanje težine. Perioda pomeračkog registra je dužina niza bitova izlazne sekvence pre nego što ona počne da se ponavlja.

Najprostiji tip pomeračkih registara sa povratnom spregom je linearni pomerački registar sa povratnom spregom (engl. *linear feedback shift register*, LFSR). Funkcija povratne sprege kod LFSR generatora je XOR operacija nad pojedinim bitovima u registru. Lista tih bitova se naziva Fibonačijeva konfiguracija (engl. *Fibonacci configuration*) ili tap sekvencu (engl. *tap sequence*). Pomerački registar dužine n može da ima $2^n - 1$ početnih stanja, a ne 2^n da se ne bi desilo da postoji stanje u kojima mogu biti sve nule. Izlazni niz pokazuje osobinu periodičnosti pa se smatra pseudoslučajnim.



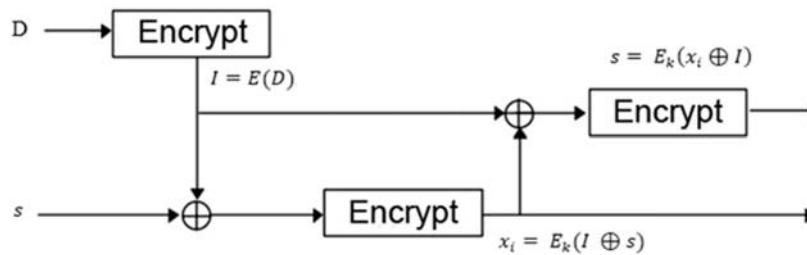
Slika 1 - Linearni pomerački registar sa povratnom spregom

Generisane sekvence pomeračkih registara nalaze svoju primenu u kriptologiji i u teoriji kodovanja. Koristi se uglavnom kao generator ključeva u šiframa toka.

2.2.3 ANSI X9.17

ANSI X9.17 je složeni generator pseudoslučajnih sekvenci. Koristi se za generisanje simetričnih ključeva kod DES (engl. *Data Encryption Standard*) algoritma i inicijalnih vektora za CBC (engl. *Cipher Block Chaining*¹) režim. ANSI X9.17 generator vrši šifrovanje pomoću trostrukog DES algoritma sa dva ključa. Početna 64-bitna vrednost generatora, s , generiše se na osnovu trenutnog vremena i datuma, a zatim se obavlja šifrovanje: $I = E_k^3(s)$

U algoritmu se dalje obavljaju sledeće operacije, računa se vrednost $I = E(D)$, gde D predstavlja 64-bitni zapis trenutnog vremena. $x_i = E_k(I \oplus s)$ i $s = E_k(x_i \oplus I)$. Izlaz iz generatora je 64-bitni niz slučajnih brojeva. Algoritam se ponavlja dok se ne dobije sekvenca željene dužine.



Slika 2 - Standard ANSI X9.17 za generisanje ključeva

Ovaj generator se pokazao kao siguran u većini primena ali je potrebno napomenuti da nije formalno dokazano da je kriptološki siguran. Prednost korišćenja ANSI X9.17 generatora je što ima dosta veću brzinu u odnosu na ostale generatore pseudoslučajnih sekvenci.[2]

2.2.3.1 ANSI X9.17 preporuke za generisanje ključeva

DoD (*Department of Defense*) – Američko ministarstvo odbrane preporučuje korišćenje algoritma DES u režimu rada OFB za generisanje slučajnih ključeva. Ključ za DES algoritam se generiše pomoću vektora sistemskih prekida, registra rada sistema i sistemskih brojača. Inicijalni vektor se generiše na osnovu radnog takta, sistemskog ID-a, datuma i vremena. Za otvoreni tekst (engl. *plain text*) koristi se 64-bitni znakovni niz generisan van sistema.

2.2.4 FIPS 186

FIPS 186 je pseudoslučajni generator privatnih ključeva kod DSA algoritma (*Digital Signature Algorithm*) koji koristi FIPS 186 jednosmernu funkciju $G(t, c)$. DSA algoritam ne predstavlja algoritam za šifrovanje već se koristi za digitalno potpisivanje. Algoritam kojim je definisana funkcija:

¹ Režim ulančavanja blokova

- Ulazni nizovi bitova t i c dužine 160 bitova rastavljaju se na po pet 32-bitnih podnizova $t_0, t_1, \dots, t_4$ i $c_0, c_1, \dots, c_4$, a zatim se vrši operacija ekskluzivno ILI za svako $i = 0, \dots, 4$, $x_i = t_i \oplus c_i$
- U petlji se za svako i radi sledeće:
 - o $b_1 = c_{(i+4) \bmod 5}$, $b_2 = c_{(i+3) \bmod 5}$
 - o $a_1 = x_i$, $a_2 = x_{(i+1) \bmod 5} \oplus x_{(i+4) \bmod 5}$
 - o Određuje se nadovezivanje nizova $A = a_1, a_2, B = b_1^{(LSB24)} b_2$, gde $b_1^{(LSB24)}$ predstavlja 24 najniža bita niza b_1
 - o Niz A se šifrue nizom B pomoću DES algoritma: $y_i = E_B(A)$
 - o Niz y_i se rastavlja u dva 32-bitna podniza: L_i, R_i
- U petlji se za $i = 0, \dots, 4$ određuje: $z_i = L_i \oplus R_{(i+2) \bmod 5} \oplus L_{(i+3) \bmod 5}$
- Izlaz iz funkcije $G(t, c)$ je 160-bitni niz koji se dobija nadovezivanjem podnizova $z_0 z_1 z_2 z_3 z_4$

Kao ulazni podatak FIPS 186 generatora koristi se 160-bitni prost broj. Za unutrašnje potrebe generatora definiše se parametar t . Vrednost t je heksadecimalna. Zatim se generiše početna vrednost s pomoću nekog izvora slučajnosti, a nakon toga se u petlji $= 1, \dots, n$, gde je n određeno dužinom sekvence, obavljaju sledeće operacije:

- $z_i = (s + y_i) \bmod 2^b$ – parametar y_i služi za prikupljanje slučajnosti, a parametar b dužinu bitova
- $a_i = G(t, z_i) \bmod q$ – primena FIPS 186 jednosmerne funkcije, parametar q predstavlja ulazni 160-bitni prost broj
- $s = (1 + s + a_i) \bmod 2^b$

Izlazna sekvenca iz generator predstavlja niz od n brojeva $a_1, \dots, a_m$ u intervalu $0 \leq a_i \leq q$ [2][3]

2.2.5 RSA

RSA je generator pseudoslučajnih brojeva i radi tako što se izračuna moduo N koji je proizvod dva velika prosta broja p i q , zatim se računa ceo broj e koji je prost u odnosu na $(p-1)(q-1)$. Bira se slučajna vrednost x_0 koja predstavlja seme (*seed*), gde je x_0 manje od N .

$$x_{i+1} = x_i^e \bmod N$$

Na izlazu ovog generatora naći će se bit najmanje težine od x_i . Ovom generatoru sigurnost pruža pouzdanost RSA algoritma jer ga je teško razbiti. Sigurnost mu obezbeđuje veliki broj N . [4]

2.2.6 Blum, Blum i Shub

Generator Blum, Blum i Shub naziv je dobio po svojim tvorcima. Spada u grupu najefikasnijih generatora zasnovanih na teoriji složenosti, poznat je i pod nazivom generator kvadratnih ostataka jer u osnovi ima veze sa kvadratnim ostacima po modulu n .

Najpre se generišu dva velika prosta broja p i q , koji su kongruentni sa 3 po modulu 4 (prilikom deljenja sa 4 daju ostatak 3). Odredi se vrednost n kao proizvod p i q . $n = pq$

što predstavlja Blumov ceo broj. Zatim se bira seme (*seed*) za generator koje je uzajamno prosto sa n (najmanji zajednički delilac za s i n je 1). Izračunavanjem: $x_0 = x^2 \bmod n$. Dalje se izračunavaju bitovi, i -ti pseudoslučajan bit je bit najmanje težine od x_i , gde je:

$$x_i = x_{i-1}^2 \bmod n.$$

Jedno od svojstva ovog generatora je to da ako su poznati parametri p i q moguće je izračunavanje i -tog bita. b_i je bit najmanje težine x_i i to $x_i = x_i^{(2^i) \bmod ((p-1)(q-1))}$. Sigurnost se garantuje na osnovu težine faktorisanja broja n . To znači da ako je n javno svako može da generiše bitove korišćenjem ovog generatora ali ako kriptanalitičar ne uspe da faktoriše n nikada ne može predvideti izlaz iz generatora. Sigurnost generatora čini matematička osnova faktorisanja broja n i to na način da je data sekvencija koju je generisao ovaj generator nepredvidiva ulevo i nepredvidiva udesno. To znači da kriptanalitičar ne može predvideti sledeći bit u sekvenci, ni prethodni bit u sekvenci. Algoritam ovog generatora je relativno spor, međutim primenu ima na mestima gde je bezbednost od velikog značaja, kao što je generisanje kriptoloških ključeva.[4]

2.2.7 Aditivni generatori

Aditivni generatori (engl. *additive generators*), poznatiji i pod nazivom Fibonačijevi generatori sa kašnjenjem (engl. *lagged Fibonacci generators*). Koriste se kao gradivni elementi za sigurne generatore. Efikasnost ovih generatora leži u tome što umesto slučajnih bitova daju slučajne reči.

Za početno stanje ovog generatora uzima se niz od n -bitnih reči. $X_1, X_2, X_3, \dots, X_m$. Izračunavanje i -te reči generatora se vrši po formuli:

$$X_i = (X_{i-a} + X_{i-b} + X_{i-c} + \dots X_{i-m}) \bmod 2^n$$

Najmanja perioda ovog generatora je $2^n - 1$ i to pod uslovom da su koeficijenti $a, b, c, \dots, m$ ispunili zahtev, da bit najmanje težine formira najduži LFSR² (engl. *linear feedback shift register*).

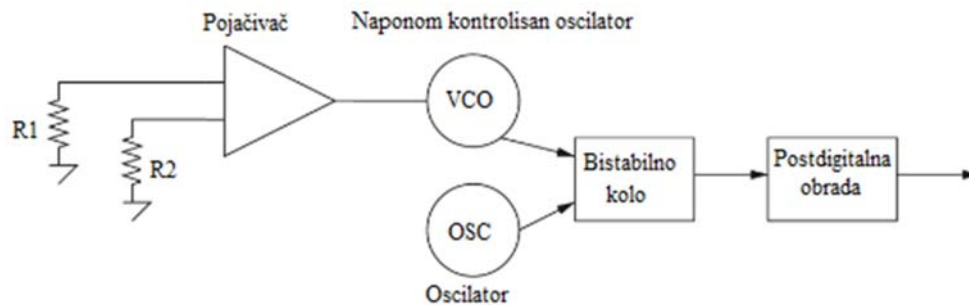
² Linearni pomerački registar sa povratnom spregom

2.2.8 AT&T 7001

Komercijalni čip kompanije AT&T je pravi generator slučajnih sekvenci i koristi fizički šum kao entropiju izvora. Generiše tok slučajnih bitova na osnovu nestabilnosti frekvencije oscilatora.[4] Izlazni niz bitova je dalje procesuiran ekskluzivnim sabiranjem - XORovanjem tako što se vrši ekskluzivno sabiranje izlaznih bita sa prethodnim bitovima.[9]

2.2.9 Intel TRNG

Za izvor slučajnosti (entropiju izvora) Intelov pravi generator slučajnih sekvenci (TRNG) koristi termički šum na čvoru. Na samom ulazu se nalaze dva otpornika da bi ovaj generator učinili mnogo snažnijim i otpornijim na ulazni napon i varijacije u samom šumu. Zatim se koristi diferencijalni pojačivač šuma koji šalje signal na ulaz na Naponom kontrolisan oscilator (engl. *voltage-controlled oscillator*) čiji signal se kombinuje sa još jednim oscilatorom. Na izlazu se dobija sekvenca koja se destiluje korišćenjem metode postdigitalne obrade Mapiranje prelaza. Na kraju se destilovani niz hešuje jednosmernom funkcijom SHA-1.



Slika 3- Intel TRNG

Nakon implementacije NIST statističke baterije testova, smatra se da je metod Mapiranje prelaza ključan u otklanjanju odstupanja u generisanoj sekvenci. [8]

2.2.10 Oscilacije vazduha u disku računara kao generator slučajnosti

Ovaj generator koristi varijacije u brzini obrtanja diska koje je prouzrokovano turbulencijom vazduha što je ujedno i izvor slučajnosti. Proces generisanja slučajnih bitova počinje tako što generator meri vreme potrebno za čitanje jednog bloka diska, a zatim koristi varijacije u tom izmerenom intervalu kao izvor slučajnosti. Da bi uklonio nastala odstupanja i korelacije primenjuje metod brze Furijerove transformacije. Na izlazu slučajan niz bitova se dobija korišćenjem spektralnih uglova za frekvencije u intervalu $(0, \pi)$ normalizovanih na jedinični interval.

2.3 Testovi slučajnosti – standardi za testiranje slučajnosti

NIST (*National Institute of Standards and Technology*) je neregulatorno telo koje pripada odeljenju trgovine Sjedinjenih Američkih Država, osnovano 1901. godine. Njegova misija je da promoviše inovacije SAD-a, industrijsku kompetitivnost u cilju unapređivanja nauke, standarda i tehnologije, kako bi se poboljšala ekonomska sigurnost i kvalitet života.

NIST paket statističkih testova sadrži 15 testova, koji su razvijeni kako bi se uspešno izvršilo testiranje slučajnosti binarnih sekvenci proizvedenih od strane fizičkih ili softverskih generatora slučajnosti. Glavni fokus testova je na raznovrsnosti tipova determinističnosti koji mogu postojati u sekvenci. Pojedini testovi se sastoje od podtestova. NIST paket se sastoji od sledećih testova:

- Ispitivanje učestalosti u nizu
- Ispitivanje učestalosti u bloku
- Ispitivanje uzastopnih ponavljanja istih bitova u nizu
- Ispitivanje najdužeg uzastopnog ponavljanja jedinica u bloku
- Ispitivanje stanja binarne matrice
- Ispitivanje diskretne Furijerove transformacije
- Ispitivanje nepreklapajućih uzoraka
- Ispitivanje preklapajućih uzoraka
- Maurerov univerzalni statistički test
- Ispitivanje linearne kompleksnosti
- Serijski test
- Ispitivanje približne entropije
- Ispitivanje slučajnih zbirova
- Ispitivanje slučajne digresije
- Ispitivanje slučajne promenljive digresije

2.3.1 Ispitivanje učestalosti u nizu

Fokus ovog testa je na ispitivanju odnosa jedinica i nula kompletne sekvence. Cilj testa predstavlja određivanje da li je broj nula i jedinica približno isti kao što se očekuje kod istinski slučajne sekvence. Test procenjuje da bi broj nula i jedinica trebalo da bude isti, svi podtestovi zavise od uspešnosti ovog testa. Pokretanje testa se vrši pozivanjem metoda $\text{Frequency}(n)$, gde n predstavlja dužinu bitnog uzorka.

Primer:

Ulazni podaci

```
ε = 11001001000011111101101010100010001000010110100011
    00001000110100110001001100011001100010100010111000
n = 100
```

n – dužina bitova u stringu

ϵ – sekvenca bitova generisanih od strane PRNG ili TRNG generatora

Obrada

$$S_{100} = -16$$

$$S_{\text{obs}} = 1.6$$

S_{obs} – konačna vrednost sume X_i ($X_i = 2\varepsilon - 1 = \pm 1$) sekvence podeljena sa kvadratnim korenom dužine sekvence

Izlaz

$$P = 0.109599$$

Zaključak

Vrednost $P \geq 0.01$, pa se sekvenca smatra slučajnom.

Ukoliko je vrednost parametra P manja od 0.01 sekvenca se smatra determinističkom.

Ukoliko ima previše nula ili previše jedinica, rezultat statističkog testa će biti veći od nule.

2.3.2 Ispitivanje učestalosti u bloku

Fokus testa je na ispitivanju odnosa jedinica u n -bitnim blokovima. Cilj ovog testa je utvrđivanje da li je učestanost jedinica u n -bitnom bloku jednak $n/2$, kao što se očekuje kod prave slučajnosti. Pokretanje testa se vrši pozivanjem metoda $\text{BlockFrequency}(M, n)$, gde n predstavlja dužinu bitnog uzorka, M predstavlja dužinu svakog bloka.

Primer:

Ulazni podaci

$$\varepsilon = \begin{array}{l} 11001001000011111101101010100010001000010110100011 \\ 00001000110100110001001100011001100010100010111000 \end{array}$$

$$n = 100$$

$$M = 10$$

M – dužina svakog bloka

Obrada

$$N=10$$

$$\chi^2 = 7.2$$

$\chi^2(\text{obs})$ – mera koja određuje koliko se posmatrana raspodela jedinica unutar n -bitnog bloka poklapa sa očekivanom raspodelom ($1/2$)

Izlaz

$$P = 0.706438$$

Zaključak

Vrednost $P \geq 0.01$, pa se sekvenca smatra slučajnom.

Ukoliko je vrednost parametra P manja od 0.01 sekvenca se smatra determinističkom.

Vrednosti manje od 0.01 označava veliku devijaciju od jednake raspodele jedinica i nula, u bar jednom bloku.

2.3.3 Ispitivanje uzastopnih ponavljanja istih bitova u nizu

Fokus ovog testa je ispitivanje ukupnog broja uzastopnih ponavljanja jednog broja u sekvenci, gde ponavljanje predstavlja neprekidnu sekvencu identičnih bitova. Cilj ovog testa je utvrđivanje da li je broj uzastopnih ponavljanja jedinica i nula različitih dužina očekivan kao za slučajnu sekvencu. Pažnja se obraća i na brzinu oscilacije između nula i jedinica. Pokretanje testa se vrši pozivanjem metoda $\text{Runs}(n)$, gde n predstavlja dužinu bitnog uzorka.

Primer:

Ulazni podaci

Preporučuje se da svaka sekvenca koja se testira sadrži barem 100 bita ($n \geq 100$).

$\varepsilon = 11001001000011111101101010100010001000010110100011$
00001000110100110001001100011001100010100010111000
 $n = 100$
 $\tau = 0.02$

Obrada

$\pi = 0.42$
 $Vn(obs) = 52$

$Vn(obs)$ – konačan broj uzastopnih ponavljanja (zbir broja ponavljanja nula i jedinica)

Izlaz

$P = 0.500798$

Zaključak

Vrednost $P \geq 0.01$, pa se sekvenca smatra slučajnom.

Ukoliko je vrednost parametra P manja od 0.01 sekvenca se smatra determinističkom.

Velika vrednost konačnog broja uzastopnih ponavljanja $Vn(obs)$ označava da je oscilacija velike brzine i obrnuto.

2.3.4 Ispitivanje najdužeg uzastopnog ponavljanja jedinica u bloku

Fokus ovog testa je ispitivanje najdužeg uzastopnog ponavljanja jedinica u n -bitnim blokovima. Cilj testa je utvrđivanje da li je dužina najdužeg uzastopnog ponavljanja jedinica unutar testirane sekvence poklapa sa dužinom najdužeg uzastopnog ponavljanja jedinica

koja bi se očekivala u slučajnoj sekvenci. Neregularnost očekivane dužine najdužeg uzastopnog ponavljanja jedinica implicira da postoji neregularnost u očekivanoj dužini najdužeg uzastopnog ponavljanja nula. Prema tome neophodan je samo test za jedinice. Pokretanje testa se vrši pozivanjem metoda LongestRunOfOnes(n), gde n predstavlja dužinu bitnog uzorka.

Sekvenca bitova je generisana od strane TRNG ili PRNG generatora. Postoje tri moguće vrednosti parametra M ($8, 128, 10^4$) u skladu sa dužinom bitnog uzorka n (128, 6272, 750.000).

Primer testa:

Ulazni podaci

Preporučuje se da svaka sekvenca koja se testira sadrži barem 100 bita ($n \geq 100$).

$\varepsilon = 11001100000101010110110001001100111000000000001001$
 $00110101010001000100111101011010000000110101111100$
 $1100111001101101100010110010$

n = 128

Obrada

Podblok	Maksimalna uzastopna ponavljanja	Podblok	Maksimalna uzastopna ponavljanja
11001100	(2)	00010101	(1)
01101100	(2)	01001100	(2)
11100000	(3)	00000010	(1)
01001101	(2)	01010001	(1)
00010011	(2)	11010110	(2)
10000000	(1)	11010111	(3)
11001100	(2)	11100110	(3)
11011000	(2)	10110010	(2)

$v0 = 4; v1 = 9; v2 = 3; v4 = 0; \chi^2 = 4.882457$

$X^2(\text{obs})$ – predstavlja meru koliko dobro posmatrano najduže uzastopno ponavljanje unutar n-bitova bloka odgovara očekivanoj dužini. Velike vrednosti ovog parametra označavaju da testirana sekvenca ima grupe jedinica.

Izlaz

P = 0.180609

Zaključak

Vrednost $P \geq 0.01$, pa se sekvenca smatra slučajnom.

Ukoliko je vrednost parametra P manja od 0.01 sekvenca se smatra determinističkom.

2.3.5 Ispitivanje stanja binarne matrice

Drugi pristup pri testiranju slučajnosti jeste utvrđivanje linearne zavisnosti između podnizova fiksne dužine. Podnizovi pripadaju originalnoj sekvenci. Postupak se sastoji od sledećih koraka:

- konstituisanje matrice uzastopnih nula i jedinica iz sekvence
- provera linearne zavisnosti između redova i kolona

Devijacija od očekivane vrednosti matrice (*rank deficiency*) daje potrebnu statistiku.

Test je primena jednog od testova iz *Diehard*³ baterije testova.

Test se poziva preko metode Rank(n).

n – dužina bitnog uzorka

Izračunavanje verovatnoće zavisnosti podnizova:

$$p_r = 2^{r(Q+M-r)-MQ} \prod_{i=0}^{r-1} \frac{(1-2^{i-Q})(1-2^{i-M})}{1-2^{i-r}}$$

M – broj redova u matrici

Q – broj kolona u matrici

r = 0, 1, 2, ..., m

m ≡ min (M, Q)

Primer:

Ulazni podaci

ε – binarno proširenje do 1000000 bita

ε = 11001001000011111101101010100010001000010110100011
00001000110100110001001100011001100010100010111000

n = 100

Obrada

N₁ = 46

N₀ = 47.5

d = -1.376494

Izlazni parametar

P = 0.168669

Vrednost P ≥ 0.01, pa se sekvenca smatra slučajnom.

³ Diehard baterija statističkih testova koristi se za merenje kvaliteta generatora nasumičnih brojeva.

2.3.6 Ispitivanje diskretne Furijerove transformacije

Test je zasnovan na diskretnoj Furijerovoj transformaciji. Test pripada skupu procedura poznatih kao „spektralne metode”. Svrha Furijerovog testa je otkrivanje periodičnih funkcija u seriji bitova. Koristi se za prikazivanje devijacije od pretpostavke o slučajnosti.

Test se izračunava po sledećoj formuli:

$$f_j = \sum_{k=1}^n x_k \exp(2\pi i (k-1)j/n)$$

Test se poziva preko metode DiscreteFourierTransform(n).
n – dužina bitnog uzorka

Primer:

Ulazni parametri

ε – binarno proširenje do 1000000 bita
n = 100000
M = Q = 32
M – broj redova u matrici
Q – broj kolona u matrici

Obrada

N = 97
 $F_M = 23, F_{M-1} = 60, N - F_M - F_{M-1} = 14$
 $X^2 = 1.2619656$

Izlazni parametar

P = 0.532069
Vrednost $P \geq 0.01$, pa se sekvenca smatra slučajnom.

2.3.7 Ispitivanje nepreklapajućih uzoraka

Test odbacuje sekvence koje imaju previše ili premalo pojava datog neperiodičnog šablona. Potrebno je otkriti da li broj pojava uzoraka koji se ne poklapaju približno jednak broju koji se očekuje za niz slučajnih brojeva.

Fiksirana sekvenca nula i jedinica (šablon):

$$B = \{j, 1 \leq j \leq m-1, \varepsilon_{j+k}^0 = \varepsilon_k^0, k = 1, \dots, m-j\}$$

m – dužina bita u šablonu

Izračunavanje sume nasumičnih varijabli:

$W = W(m, M)$ - Broj pojava u datom šablonu B

$$W = \sum_{i=1}^{n-m+1} I(\varepsilon_{i+k-1} = \varepsilon_k^0, k = 1, \dots, m)$$

Test se poziva preko metode NonOverlappingTemplateMatching(m, n).

m – dužina bita u šablonu

n – dužina bitnog uzorka

Primer:

Ulazni parametri

$\varepsilon = 2^{20}$ bitova dobijenih od G-SHA-1 generatora

$n = 2^{20}$

B = 000000001

Obrada

$\mu = 255.984375$

$\sigma^2 = 247.499999$

W1 = 259; W2 = 229; W3 = 271; W4 = 245; W5 = 272; W6 = 262; W7 = 259; W8 = 246;

$X^2(\text{abs}) = 5.999377$

Izlazni parametar

P = 0.110434

Vrednost $P \geq 0.01$, pa se sekvenca smatra slučajnom.

2.3.8 Ispitivanje preklapajućih uzoraka

Test odbacuje sekvence koje prikazuju previše ili premalo pojava u m – nizu jedinica, ali sa lakoćom se može modifikovati za otkrivanje neregularnih pojava za bilo koju periodičnu šemu B. Za implementaciju ovog testa parametri M i N su određeni tako da $n = MN$, gde n predstavlja blokove u okviru originalnog niza a svaki blok ima dužinu M.

$\tilde{W}_j = \tilde{W}_j(m, n)$ predstavlja broj mogućih preklapanja nizova jedinica dužine m u j-tom bloku

Asimptotička distribucija W_j je sastavni deo Poisson distribucije:

$$E \exp\{t \tilde{W}_j\} \rightarrow \exp\left\{\frac{\lambda(e^t - 1)}{2 - e^t}\right\}$$

Kada važi $(M - m + 1)2^{-m} \rightarrow \lambda > 0$

t – realna varijabla

Odgovarajuća verovatnoća se može izraziti kroz Kumerovu funkciju $\Phi = {}_1F_1$

$$P(U = u) = \frac{e^{-\eta}}{2^u} \sum_{l=1}^u \binom{u-1}{l-1} \frac{\eta^l}{l!} = \frac{\eta e^{-2\eta}}{2^u} \Phi(u+1, 2, \eta)$$

Ukoliko U označava nasumičnu varijablu sa sastavnim delom $\eta = \lambda/2$ Poisson asimptotičke distribucije onda važi $u \geq 1$

Test se poziva preko metode OverlappingTemplateMatching(m, n).

m – dužina bita u šablonu

n – dužina bitnog uzorka

Primer:

Ulazni parametri

ε = binarno proširenje do 1000000 bita

n = 1000000

B = 111111111

Obrada

v0 = 329; v1 = 164; v2 = 150; v3 = 111; v4 = 78; v5 = 136;

X^2 (abs) = 8.965859

Izlazni parametar

P = 0.110434

Vrednost $P \geq 0.01$, pa se sekvenca smatra slučajnom.

2.3.9 Maurerov univerzalni statistički test

Ispituje mogućnosti kompresije niza bez gubitka informacije. Za niz koji se može značajno kompresovati smatra se da nije niz slučajnih bitova, zbog činjenice da je kompresija niza efikasnija ukoliko niz pokazuje periodična svojstva.

Test zahteva dugačku sekvencu bitova (reda $10 * 2^L + 1000 * 2^L$ gde je $6 \leq L \leq 16$) koji su podeljeni u dva dela L – tog bloka ($6 \leq L \leq 16$), Q ($\geq 10 * 2^L$), K ($\sim 1000 * 2^L$).

Q – broj blokova u inicijalnoj sekvenci

K – test blokovi

Test nije prilagođen za velike vrednosti L, jer inicijalizacija uzima vremenski eksponencijal u vrednosti L.

Test pregleda celu sekvencu dok prelazi kroz test segment L-tog bloka tražeći najbližu prethodnu istu L-tu šemu i beleži rastojanja kao broj blokova.

Formula koja izračunava pre koliko blokova je bilo ponavljanje:

$$f_n = \frac{1}{K} \left[\sum_{i=Q+1}^{Q+K} \log_2 \right]$$

Algoritam izračunava $\log_2$ za sve L – te šeme u test segmentu, zatim računa prosek za sve proširive dužine sa brojem test blokova.

i – trenutni blok koji se izračunava

Test se poziva preko metode Universal(L, Q, n).

L – dužina svakog bloka

Q – broj blokova u inicijalnoj sekvenci

n – dužina bitnog uzorka

Primer:

Ulazni parametri

ε = binarno proširenje do 1000000 bita

n = 1000000

B = 111111111

Obrada

v0 = 329; v1 = 164; v2 = 150; v3 = 111; v4 = 78; v5 = 136;

X^2 (abs) = 8.965859

Izlazni parametar

P = 0.110434

Vrednost $P \geq 0.01$, pa se sekvenca smatra slučajnom.

2.3.10 Ispitivanje linearne kompleksnosti

Ovaj test koristi linearnu kompleksnost za testiranje slučajnosti. Linearna kompleksnost je sastavni deo linearnih pomeračkih registara. Test se koristi kako bi se odredila dužina linearnog pomeračkog registra, ispituje se kompleksnost sekvence i ukoliko sekvenca sadrži potrebnu kompleksnost, smatra se slučajnom.

Test se poziva preko metode LinearComplexity(M, n).

M – dužina svakog bloka

n – dužina bitnog uzorka

Primer:

Ulazni podaci

ε – binarno proširenje do 1000000 bita

n = 1000000 = 10⁶

M=1000

Obrada

$V_0 = 11$; $V_2 = 31$; $V_3 = 501$; $V_4 = 258$; $V_5 = 57$; $V_6 = 26$;
 $X^2(\text{abs}) = 2.700348$

Izlaz

$P = 0.845406$
Vrednost $P \geq 0.01$, pa se sekvenca smatra slučajnom.

2.3.11 Serijski test

Serijski test utvrđuje učestalost svih mogućih preklapanja n -bitnog niza u kompletnoj sekvenci.

Test se poziva preko metode $\text{Serial}(m, n)$.

m – dužina svakog bloka

n – dužina bitnog uzorka

Primer:

Ulazni podaci

ε – binarno proširenje do 1000000 bita

$m = 2$

$n = 1000000$

Obrada

$\#0s = 499971$

$\#1s = 500029$

$\#00s = 250116$

$\#01s = \#10s = 249855$

$\#11s = 250174$

$\psi^2_2 = 0.343128$; $\psi^2_1 = 0.003364$; $\psi^2_0 = 0.000000$;

Izlazni parametri

$\nabla\psi^2_2 = 0.339764$

$\nabla^2\psi^2_2 = 0.336400$

$P_1 = 0.843764$

$P_2 = 0.561915$

Vrednosti P_1 i $P_2 \geq 0.01$, pa se sekvenca smatra slučajnom.

2.3.12 Ispitivanje približne entropije

Posmatra se učestalost pojave svih mogućih preklapajućih n-bitnih uzoraka u nizu. Cilj ovog testa je poređenje učestalosti preklapajućih blokova sa očekivanim rezultatima.

Test se poziva preko metode `ApproximateEntropy(m, n)`.

m – dužina svakog bloka

n – dužina bitnog uzorka

Primer:

Ulazni podaci

ε – binarno proširenje do 1000000 bita

$\varepsilon = 11001001000011111101101010100010001000010110100011$
00001000110100110001001100011001100010100010111000

m = 2

n = 100

Obrada

$ApEn(m) = 0.665393$

$X^2(obs) = 5.550792$

Izlazni parametri

P = 0.235301

Vrednost $P \geq 0.01$, pa se sekvenca smatra slučajnom.

2.3.13 Ispitivanje slučajnih zbirova

Test utvrđuje da li je kumulativni zbir parcijalnih sekvenci, koje se javljaju u testiranom nizu, suviše veliki ili suviše mali u odnosu na očekivano ponašanje tog kumulativnog zbira slučajnih sekvenci. Velike vrednosti ukazuju da postoji veliki broj jedinica ili nula na početku sekvence. Male vrednosti ukazuju da su jedinice i nule previše jednako raspoređene.

Test se poziva preko metode `CumulativeSums(mod, n)`.

mod = 0 ili mod = 1

n – dužina bitnog uzorka

Primer:

Ulazni podaci

ε – binarno proširenje do 1000000 bita

$\varepsilon = 11001001000011111101101010100010001000010110100011$

00001000110100110001001100011001100010100010111000
 $n = 100$

$\text{mod} = 0 \parallel \text{mod} = 1$

Obrada

$z = 1.6 \parallel z = 1.9$

Izlazni parametar

$P = 0.219194 \parallel P = 0.114866$

Vrednost $P \geq 0.01$, pa se sekvenca smatra slučajnom.

2.3.14 Ispitivanje slučajne digresije

Test utvrđuje da li je broj određenih stanja u okviru ciklusa odstupa od onoga što bi se očekivalo od slučajne sekvence. Sastoji se od osam testova i zaključaka, po jedan test i zaključak za svako stanje: -4, -3, -2, -1, +1, +2, +3, +4.

Test se poziva preko metode RandomExcursions(n).
 n – dužina bitnog uzorka

Primer:

Ulazni podaci

ε – binarno proširenje do 1000000 bita
 $n = 1000000 = 10^6$

Obrada

$J = 1490$

Stanje = x	X^2	P	Zaključak
-4	3.835698	0.573306	slučajan
-3	7.318707	0.197996	slučajan
-2	7.861927	0.164011	slučajan
-1	15.692617	0.007779	nije slučajan
+1	2.485906	0.778616	slučajan
+2	5.429381	0.365752	slučajan
+3	2.404171	0.790853	slučajan

+4	2.393928	0.792378	slučajan
----	----------	----------	----------

Za x, u sedam stanja važi da je $P \geq 0.01$, pa se sekvenca smatra slučajnom.

2.3.15 Ispitivanje slučajne promenljive digresije

Test utvrđuje da li je ukupan broj stanja u kojima se niz nalazi, između [-9, -1] i [1, 9] premašuje očekivane vrednosti čisto slučajne sekvence. Sastoji se od osamnaest testova i zaključaka, po jedan test i zaključak za svako stanje: -9, -8, -7, ..., -1, +1, +2, +3, ..., +9.

Test se poziva preko metode RandomExcursionsVariant(n).

n – dužina bitnog uzorka

Primer:

Ulazni podaci

ε – binarno proširenje do 1000000 bita

n = 1000000 = 10⁶

Obrada

J = 1490

Stanje = x	X ²	P	Zaključak
-4	3.835698	0.573306	slučajan
-3	7.318707	0.197996	slučajan
-2	7.861927	0.164011	slučajan
-1	15.692617	0.007779	nije slučajan
+1	2.485906	0.778616	slučajan
+2	5.429381	0.365752	slučajan
+3	2.404171	0.790853	slučajan
+4	2.393928	0.792378	slučajan

Za svako od 18 stanja za x, važi da je $P \geq 0.01$, pa se sekvenca smatra slučajnom.

2.4 Postdigitalna obrada generisanih sekvenci

Glavni problem generatora slučajnosti je to što bi mogla da se javi predvidivost u generisanoj sekvenci iako je sekvenca generisana na potpuno slučajan način. Da bi smo obezbedili uniformnost generisanog niza, gde izlaz prevazilazi oređena odstupanja (engl. *bias*) ili statističku zavisnost izazvanu od hardverskog izvora slučajnosti moramo uvesti proces postdigitalne obrade generisanih binarnih sekvenci.[5]

Naknadno procesiranje generisanih sekvenci ili postdigitalna obrada ekstrakta entropije izvora se vrši kombinovanjem različitih entropija izvora i postdigitalne obrade koja rezultuje sa binarnim sekvencama koje poseduju veliku verovatnoću nepredvidivosti, čak iako izvor binarne sekvence ne poseduje osobine nepredvidivosti. U nastavku će biti opisani najzastupljeniji metodi destilacije generisanih sekvenci.

2.4.1 Mapiranje prelaza

Mapiranje prelaza (engl. *transition mapping*) je algoritam koji je osmislio Džon fon Nojman (engl. John von Neumann) i služi za potpuno uklanjanje pojave dugih nizova 0 ili 1. Algoritam radi tako što se analiziraju po dva bita zajedno i u slučaju da postoji prelaz između dva bita (01-10) samo jedan od njih se prihvata kao slučajni i to ako je 01 uzima se 0, a ako je 10, uzima se 1. Ako nema prelaza (00-11) bitovi se odbacuju kao da nisu slučajni.[5] Pretpostavimo da je verovatnoća za vrednost 1 je $0,5 + e$ i verovatnoća za vrednost 0 je $0,5 - e$. Gde e predstavlja odstupanje izvora odnosno neuniformnu raspodelu nula i jedinica u generisanom nizu. Verovatnoća za svaki par je prikazana u sledećoj tabeli. [7]

Par	Verovatnoća
00	$(0,5 - e)^2 = 0,25 - e + e^2$
01	$(0,5 - e) * (0,5 + e) = 0,25 - e^2$
10	$(0,5 + e) * (0,5 - e) = 0,25 - e^2$
11	$(0,5 + e)^2 = 0,25 + e + e^2$

Ovaj algoritam će eliminisati svako odstupanje izvora ali će takođe odbaciti neodređen broj ulaznih bitova sekvence za svaki željeni broj izlaznih bitova. Verovatnoća za svaki poseban par koji će biti odbačen je $0,5 + 2e^2$, tako da očekivan broj ulaznih bitova koji će dovesti do izlaznih X bitova je:

$$\frac{X}{(0,25 - e^2)}$$

2.4.2 Paritet niza

Paritet niza (engl. *stream parity*) radi tako što se nizovi dele na parove, a zatm se određuje paritet svakog para. Parovi kod kojih je paritet različit odbacuju se, a kod kojih je isti zadržava se samo prvi bit. Bitovi koji nisu odbačeni čine niz koji prelazi u sledeći korak. U zavisnosti od kvaliteta dobijene kompresije entropije izvora određuje se stopa eksponencijalnog skraćanja polaznog niza.[5]

2.4.3 Operacija XOR

Jedan od načina da se eliminiše odstupanje jeste primena operacije XOR nad nekoliko bitova. Ako neki bit odstupa prema nuli za faktor e , tada se verovatnoća za vrednost 0 može predstaviti kao:

$$P(0) = 0,5 + e$$

Primenom operacije XOR na dva takva bita:

$$P(0) = (0,5 + e)^2 + (0,5 - e)^2 = 0,5 + 2e^2$$

Primena operacije XOR na m bitova će se uzajamno približiti ka jednakoj verovatnoći za 0 i za 1. [4]

2.4.4 Brza Furijerova transformacija

Činjenica da generator slučajnosti ima odstupanje ne mora da znači da je neupotrebljiv, on i dalje sadrži korisnu količinu slučajnosti. Ta slučajnost može biti ekstraktovana kroz korišćenje diskretne Furijerove transformacije ili nadograđene varijante, brze Furijerove transformacije (engl. *Fast Furier Transform*, FFT). Korišćenjem Furijerove transformacije binarnih sekvenci, jake korelacije se odbacuju. Ako su odgovarajući podaci procesuirani i ukoliko su ostale korelacije otklonjene, spektralne linije koje opisuju statističku nezavisnost i normalna distribuirana slučajnost može biti proizvedena.[7]

2.4.5 Kompresija

Reverzibilne ili povratne tehnike kompresije takođe omogućavaju metod eliminisanja odstupanja generisane sekvence. S obzirom da je kompresija reverzibilna, ista količina informacije mora biti prisutna u komprimovanim podacima kao što je bila i u stanju pre komprimovanja. Po Šenonovoj teoriji informacija, ovo je jedino moguće ukoliko je, u proseku, verovatnoća različitih komprimovanih sekvenci uniformno distribuirana u odnosu na verovatnoću nekomprimovanih sekvenci. Zapravo, komprimovane sekvence moraju biti razrešene od odstupanja, u odnosu na sekvence nad kojima nije vršena kompresija.[7]

2.4.6 Kompresija sa Jednosmernim funkcijama

Kriptografske heš (engl. *hash*) funkcije obezbeđuju uniformnost izlaznih podataka, pa su takođe pogodne za upotrebu u postdigitalnoj obradi. Samo pod uslovom da binarna sekvenca koja je u procesu destilacije ima visoku entropiju. Najzastupljeniji je SHA (engl. *Secure Hash Algorithm*) algoritam koji spada u grupu SHS (engl. *Secure Hash Standard*) standarda.

3 Teorijske osnove istraživanja

3.1 Entropija

Sa stanovišta savremene teorije informacija nemoguće je objasniti pojavu slučajnosti i generatore slučajnosti bez entropije. Entropiju kao informacionu meru prvi je uveo Kłod Elmvud Šenon (*Claude Elmvood Shannon*) 1948. godine u svom radu "Matematička teorija komunikacija". Šenon je u svom radu pokazao da izvor informacija poseduje kompleksnost ispod koje se informacija iz ovog izvora ne može dalje kompresovati bez gubitka. Upravo tu kompleksnost nazvao je entropija. Ona postavlja granicu najbolje moguće kompresije bez gubitka informacije. To znači da ako je entropija izvora veća od kapaciteta kanala prenosa neke poruke, takva poruka se ne može preneti bez gubitka informacije.

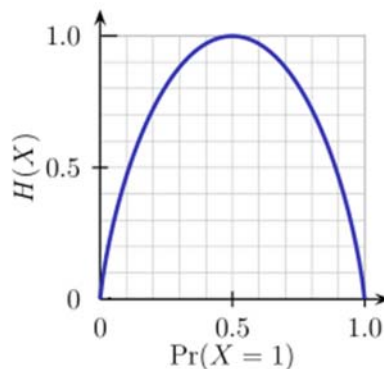
Entropija predstavlja meru neodređenosti koju nosi izlazni podatak generisan iz nekog izvora. Teorija informacija definiše količinu informacije koju nosi neka poruka. Količina te informacije u poruci se meri entropijom poruke $H(M)$. Ako entropiju poruke merimo brojem bitova, pod uslovom da je svaki simbol poruke jednako verovatan onda ona iznosi $H = \log_2 n$, gde n predstavlja broj mogućih simbola. Dakle za binarni niz iz informacionog izvora, entropija zavisi od verovatnoća nula i jedinica. $P_{(0,1)} = 0.5$. Entropiju informacionog izvora možemo predstaviti na sledeći način:

$$H = - \sum_{i=0}^{L-1} p_i \log p_i$$

Entropija binarnog informacionog izvora :

$$H = -p \log_2 p - (1-p) \log_2 (1-p)$$

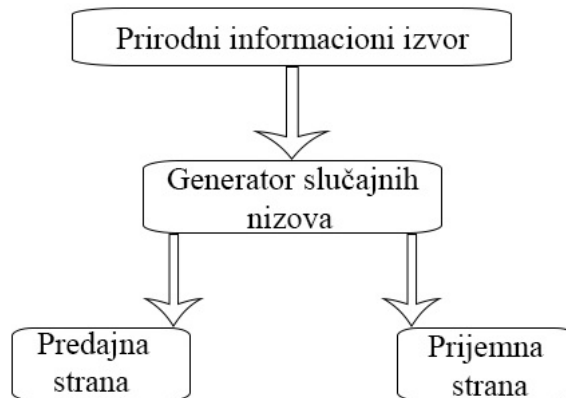
Gde entropija H predstavlja količinu informacije po jednom bitu i može da se kreće između 0 i 1. Za binarnu sekvencu kažemo da je slučajna ako ima maksimalnu entropiju po bitu $H = 1$



Slika 4- Entropija slučajne promenljive sa dva ishoda

3.2 Sigurnost kriptosistema

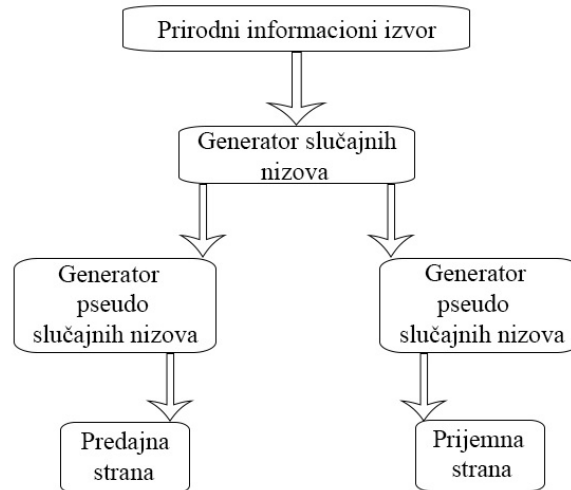
Prema Šenonu koji je definisao precizan matematički model značenja sigurnosti kriptosistema, postoje dva tipa sigurnosti. Prvi tip definiše bezuslovnu sigurnost, što znači da se za kriptografski sistem može dokazati da je siguran iako kriptanalitičar raspolaže neograničenim vremenskim i računarskim resursima. Kriptosistem koji dostiže bezuslovnu sigurnost je sistem u kome šifrat ne daje nikakve moguće informacije o otvorenom tekstu. Po Šenonovoj teoriji to je ostvarivo samo ako mogućih ključeva ima koliko i mogućih poruka odnosno dužina ključa mora biti najmanje iste dužine koliko i sama poruka i nijedan ključ se ne može ponovo upotrebiti. Šifra koja se bazira na ovom kriptosistemu se naziva jednokratna beležnica (engl. *One-Time Pad*) i predstavlja Šenonov ideal bezuslovne sigurnosti ili savršenu tajnost.



Slika 5- Šenonov ideal

Na slici je dat prikaz generatora slučajnosti koji generiše kriptološke ključeve na osnovu nekog prirodnog informacionog izvora. Dobijeni slučajni niz predstavlja kriptološki ključ koji se koristi za šifrovanje kod jednokratne beležnice. Problemi koji se javljaju su previše zahtevni, potrebno je obezbediti kopije ključa na obe strane i ključevi moraju biti iste dužine kao i poruka.

Drugi tip koji je Šenon definisao se naziva dokaziva računarska sigurnost, što znači da se za kriptosistem može dokazati nominovana sigurnost u odnosu na kriptanalitičara koji poseduje ograničene vremenske i računarske resurse. U ovakve kriptosisteme spadaju i sekvencijalne šifre iz domena simetričnih šifara i asimetirični kriptografski algoritmi. I po Šenonu predstavljaju pseudo ideal računarske sigurnosti.



Slika 6 - Šenonov Pseudo ideal

Na slici je prikazan prirodni informacioni izvor koji predstavlja generator početnih stanja kod pseudoslučajnih generatora koji se koriste za generisanje kriptološkog ključa. Kada su u pitanju kriptosistemi prednost pripada pseudo idealu u odnosu na Šenonov apsolutni ideal ali je upotrebom pseudo generatora slučajnosti uskraćena savršena tajnost.

3.3 Ispitivanje slučajnosti

Statistička ispitivanja slučajnosti mere kvalitet generatora slučajnih sekvenci, ispitivanja pronalaze slabosti generatora. Procedura ispitivanja se sastoji od propuštanja uzorka slučajne sekvence kroz seriju statističkih ispitivanja, gde je cilj pronaći attribute koji nisu karakteristični za slučajne sekvence. Neophodno je da generator ispuni sva ispitivanja, ukoliko generator ne ispuni barem jedno ispitivanje, smatra se da je pao test slučajnosti i posmatra se kao pseudoslučajni generator. Neophodnost upotrebe ispitivanja slučajnosti je neminovna, čovek ne može samostalno odrediti da li je sekvenca slučajna ili ne.

Teorija statistike pruža kvantativne mere za slučajnost. Statističko testiranje predstavlja tradicionalnu metodu testiranja slučajnosti. Upotreba jednog statističkog testa nije dovoljna, upravo iz razloga što sekvenca može proizvesti različite tipove neslučajnosti. Ukoliko generator prođe širok spektar testova, sigurnost u njegovu slučajnost raste. Iako je teško dokazati da je generator istinski slučajan, testovi slučajnosti pomažu da se pronađu slabosti koje generator potencijalno poseduje.

Svaki test pokušava da otkrije neku od osobina neslučajnosti. Može se reći da testovi nisu kompletno isključivi, što znači da postoji mogućnost da dođe do preklapanja. Statističko testiranje je najčešće korišćen način ispitivanja slučajnosti.

Postoji veliki broj testova za ispitivanje slučajnosti, u nastavku su nabrojani najrelevantniji testovi slučajnosti:

- **Knuth**
Knjiga Donalda Knuta (Donald Knuth) – “Umetnost kompjuterskog programiranja“ izdata 1969 godine je najčešće korišćena referenca o statističkom testiranju generatora. Njegova paleta testova je bila de facto standard za testiranje slučajnosti.
- **Diehard**
George Marsaglia je 1995 godine predstavio paletu testova strožijih od klasičnih Donaldovih testova. Strožiji su upravo zbog pomećenog praga prolaznosti. Iako se više ne koriste, njegovi testovi su široko prihvaćeni i prepoznati kao kvalitetni.
- **Crypt-X**
Paket statističkih testova razvijen od strane istraživača na informaciono bezbednosnom istraživačkom centru na Univerzitetu tehnologije u Kvinslendu (engl. Queensland University of Technology) u Australiji. Predstavlja komercijalni paket statističkih testova. Crypt-X testovi su primenjeni na osnovu tipa algoritma koji se testira i usmereni su ka testiranju pseudoslučajnih generatora brojeva. Podržavaju protočne šifre, blok šifre i generatore šifara toka.
- **NIST**
Objavljen 2001 godine, NIST statistički test sadrži 16 testova sposobnih da testiraju slučajnost proizvoljno dugih binarnih sekvenci proizvedenih od strane softverskih ili fizičkih generatora slučajnih ili pseudoslučajnih sekvenci. NIST paket statističkih testova predstavlja najzastupljeniji paket testova.
- **ENT**
ENT Program Džona Vokera je još jedan paket statističkih testova koji služe za testiranje slučajnosti. Korišćen od strane Random.org i Hotbits-a. ENT program je naznačen kao koristan za one koji ispituju pseudoslučajne generatore za šifrovanje i statističke aplikacije, algoritme kompresije i druge aplikacije gde je količina informacije datoteke od važnosti.
- **Previous MSISS Project**
Louise Foley je predložila paket od pet testova, koji bi mogli da zamene ENT program. Preporučeni su za testiranje sekvenci generisanih od strane Random.org.

3.4 Generatori slučajnosti

Zbog velike primene u oblasti kriptografije za generisanje ključeva, šifrovanja, sigurnosnih protokola, generisanja digitalnog potpisa javila se potreba za slučajnim i pseudoslučajnim vrednostima. Jedan od najtežih aspekata kriptografskih algoritama je generisanje slučajne informacije. Kvalitetan izvor slučajnosti je od fundamentalnog značaja jer direktno utiče na sigurnost kriptografskog sistema. To govori u prilog da ova oblast zahteva generatore koji su u stanju da proizvedu veliku količinu slučajnih vrednosti.

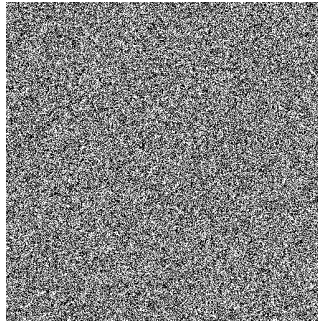
Generatori slučajnosti se dele prema načinu generisanja slučajne sekvence. Izlaz iz generatora slučajnosti je obično binarna sekvenca, postoje i generatori kod kojih izlaz nije u binarnom obliku ali posle je moguće konvertovanje u binarni kod. Postoje dva osnovna tipa generatora za reprodukciju slučajnih sekvenci. To su pravi slučajni generatori (*True Random Number Generators*) i pseudoslučajni generatori (*Pseudo Random Number Generators*).

Pravi slučajni generatori prikupljaju informacije iz stvarnog sveta da bi generisali sekvencu. Nazivaju se pravi ili istinski jer u normalnim okolnostima nije moguće predvideti sledeću vrednost u sekvenci. Pseudoslučajni generatori se uzdaju u algoritme sa nepredvidim izlazom, pod uslovom da nikome nisu poznata početna stanja, koji su dovoljni da zadovolje kriterijume slučajnosti.

Kada se govori o slučajnim nizovima, oni treba da poseduju osobinu uniformnosti. Uniformnost predstavlja raspodelu u nizu nula i jedinica, gde svaki broj ima jednaku verovatnoću da se pojavi. Ova osobina je značajna, jer sprečava vršenje predikcije budućih sekvenci.

3.5 Pravi slučajni generator (TRNG)

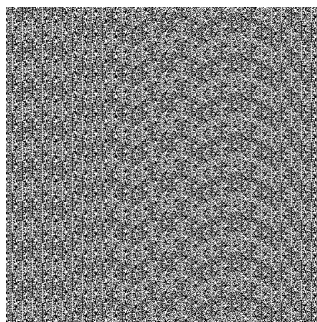
Pravi slučajni generator zahteva prirodan izvor slučajnosti tj. entropiju izvora kako bi uspešno generisao slučajnu sekvencu. Dobijanje slučajne sekvence nije lako, zbog toga pravi slučajni generator mora biti fizički implementiran. Pravi slučajni brojevi su po definiciji nedeterministički, nije ih moguće ponovo reprodukovati. Proces postdigitalne obrade je potreban kako bi se prevazišla bilo koja slabost u slučajnosti izvora koji je stacionaran (pojavljivanje nizova nula i jedinica velike dužine). Izvor slučajnosti kod pravih slučajnih generatora se sastoji iz neke fizičke vrednosti npr. atmosferski šum, emisija čestica tokom radioaktivnog raspada, kvantni efekti u poluprovodniku. Bitna karakteristika pravog slučajnog generatora ogleda se u izbegavanju periodičnosti, što znači da se dobijeni niz nikada neće ponoviti.



Slika 7- Bitmapa sekvence koja je generisana pravim generatorom slučajnosti

3.6 Pseudo slučajni generator (PRNG)

Pseudo slučajni generatori su mnogo više zastupljeniji od pravih slučajnih generatora. Nisu potpuno nasumični što znači da generisana sekvenca ne zavisi od entropije izvora. Predstavljaju determinističke generatore koji na osnovu matematičkog algoritma, uzimajući početno stanje generišu sekvencu brojeva. Upravo se osobina determinističnosti ogleda u mogućnosti predviđanja brojeva u nizu. Uslov za početno stanje je ispunjen ukoliko je ulazna vrednost generisana preko istinski slučajnog izvora. Ulazna vrednost predstavlja sekvencu (*initial seed*) ili ključ koji se koristi za pokretanje procesa generisanja. Generisana slučajna sekvenca je znatno duža od ključa. Ukoliko je ključ poznat moguće je izvršiti reprodukciju izlaznih nizova. Jedna od karakteristika PRNG-a jeste periodičnost, što znači da će se posle određenog vremena ponoviti ista sekvenca.



Slika 8- Bitmapa sekvence koju je generisao pseudoslučajni generator

3.7 Uporedna analiza generatora slučajnosti

Pravi generatori slučajnosti

Prednost ovih generatora je ta da slučajna sekvenca dobijena iz ovih generatora je potpuno nedeterministična, nema periodičnosti, što je veoma bitno sa stanovništva kriptološke sigurnosti. Pogodni su za veći skup aplikacija kao što su kriptografski sistemi za šifrovanje, igre na sreću i na mestima gde je istinska slučajnost neophodna.

Mana im je da nisu toliko efikasni u odnosu na pseudoslučajne generatore, potrebno im je mnogo više vremena da proizvedu potrebnu količinu slučajnosti. Nepogodni su za simulacije i modelovanje aplikacija gde je potrebno proizvesti više podataka.[11]

Pseudoslučajni generatori

Prednost pseudoslučajnih generatora je ta što u nekim slučajevima generisana sekvenca može sadržati veću količinu entropije od pravih generatora slučajnosti (TRNG), ukoliko je inicijalno stanje generisano od strane TRNG generatora. Veoma oprezan odabir početnih stanja rezultuje prikrivanjem determinističkih osobina ovih generatora. Pseudoslučajni generatori su efikasni (brži) i lakši za implementaciju. Pogodni su za generisanje velikog broja sekvenci u kratkom vremenskom periodu.

Mana ovih generatora je determinističnost. Determinističke osobine (periodičnost) ovog generatora omogućavaju predviđanje sledećeg bita u sekvenci.[11]

Aplikacija	Najpogodniji generator
Igre lutrije	TRNG
Interaktivne igre i igre kockanja	TRNG
Uzimanje slučajnih uzoraka	TRNG
Simulacije i modelovanje	PRNG
Sigurnost (generisanje kriptoloških ključeva)	TRNG

Tabela 1-Oblasti primene generatora slučajnosti

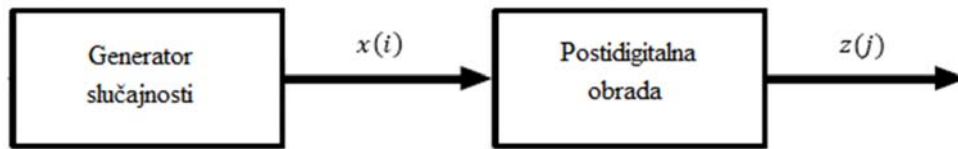
Karakteristike	TRNG	PRNG
Efikasnost	da	ne
Determinizam	da	ne
Periodičnost	da	ne

Tabela 2- Karakteristike generatora slučajnosti

3.8 Postdigitalna obrada

Generisane sekvence dobijene iz generatora slučajnosti koji koriste entropiju izvora, obično ne daju savršene rezultate. Dobijeni niz iako je na svim testovima slučajnosti davao dobre rezultate nije odmah zadovoljio kriterijum da je kriptološki upotrebljiv. Npr. Dobijeni niz ne zadovoljava uslove uniformne raspodele bitova u sekvenci ili pokazuje osobine stacionarnosti. Stoga ovakvi nizovi zahtevaju dodatnu obradu.

Osnovna svrha postdigitalne obrade je proces kreiranja pouzdanih slučajnih sekvenci iz nepouzdatih slučajnih izvora sekvenci.[5] Postoji veliki broj algoritama čija je svrha da uklone nesavršenosti iz sirovo generisane sekvence. Algoritam postdigitalne obrade kompresuje sekvencu dobijenu iz izvora tako da destiluje entropiju dobijene sekvence. Količina potrebne kompresije zavisi i od entropije izvora i od algoritma koji se koristi za destilaciju.



Slika 9- Princip postdigitalne obrade

Glavna ideja postdigitalne obrade je da “žrtvuje” određen broj bitova u sekvenci da bi dobio manji ali više slučajniji. Postoje četiri osnovne tehnike postdigitalne obrade:

- Ad hoc jednostavni korektori
- Poboljšavanje slučajnosti sa kriptografskim heš funkcijama
- Algoritmi za izdvajanje slučajnosti
- Funkcije za oporavak slučajnosti

Nakon dobijene postobrađene sekvence nemamo znanje o tome koji delovi niza su dobijeni postdigitalnom obradom, a koji od samog generatora slučajnosti. Podrazumevani prvi korak je ekstrakcija bitova iz generatora slučajnosti. To je jednostavan korak koji radi tako što se analogni ili digitalni signal dobijen iz entropije izvora konvertuje u digitalni binarni format. Drugi korak je dosta zahtevniji od prvog i dizajniran je tako da ukloni sva odstupanja (engl. *bias*) ili poremećaje (engl. *skew*) koji su prisutni posle prvog koraka. Dok se ekstrakcija bitova odnosno generisanje slučajne sekvence vrši hardverski, algoritmi postdigitalne obrade su mnogo kompleksniji pa moraju da se izvrše softverski ili pomoću mikro kontrolera iako najpogodniji metodi postdigitalne obrade su upravo oni koji mogu da se integrišu hardverski. Generalno metodi postdigitalne obrade zauzima mnogo resursa, pa dobri generatori slučajnosti ne bi trebali da koriste postdigitalnu obradu ili ako je već koriste,

da ograniče upotrebu ad hoc metoda postdigitalne obrade.[10] Upotrebom metoda postdigitalne obrade, značajno se poboljšava kvalitet slučajnosti nekog izvora što čini dobru početnu tačku za bolje slučajne ili pseudoslučajne sekvence.[5]

3.9 Oblast primene

Generatori slučajnosti i slučajne vrednosti kao njihova posledica su krucijalni sastojak u velikom spektru primene, uključujući kriptografiju, simulacije, interaktivne igre (engl. *gaming*), uzimanje slučajnih uzoraka za statističke testove, donošenje odluka i estetiku.[11]

Kriptografija

Kriptografija je umetnost ili nauka da smislenu sekvencu izokrene u naočigled slučajan šum na takav način da samo poznavaoac ključa može da dođe do originalne sekvence. Razvojem računara kriptografija je postala mnogo sofisticiranija, međutim svrha je ostala ista, da se preduhitri protivnik u sticanju prednosti kroz znanje sadržaja poruke. Ono što je potrebno ostvariti je neprediktibilnost generisanih sekvenci. I tako u samom centru kriptografskih sistema se nalazi generisanje tajnih, nepredvidih - slučajnih vrednosti. Koje su našle primenu u elektronskom poslovanju, sigurne komunikacije preko interneta.

Simulacija

Simulacija je proces ponovnog kreiranja nekog fenomena, okruženja ili iskustva koja pružaju korisniku šansu za novi nivo razumevanja. Kada se računari koriste za simulaciju nekog prirodnog fenomena, slučajno generisane vrednosti su potrebne da učine stvari mnogo realnijim. Simulacije pokrivaju veliku oblast primene, npr. nuklearnu fiziku gde se čestice sudaraju na slučajan način. Zahtevanjem sve veće preciznosti u sofisticiranim simulacijama kojima je potrebno sve više slučajnih vrednosti potreban je pouzdan generator slučajnosti.

Interaktivne igre

Igre kao što su bacanje kockica, igranje karata, vrtenje točka ruleta su fascinantna vrsta zabave za skoro svakoga. Slučajnost je ključna u igrama na sreću i od vitalnog je značaja industriji igara na sreću. Razvojem ove vrste igre na mreži, ove industrije podižu svoju novčanu vrednost. Stoga aplikacije ove vrste postaju veliki potrošači slučajnih vrednosti.

Uzimanje uzoraka

Često je nemoguće ispitati sve moguće slučajeve, ali slučajan uzorak pruža uvid u to šta predstavlja realno ponašanje. Vršenje statističkih eksperimenata sa slučajnim vrednostima daje jednaku šansu svakom članu skupa da bude izabran i zaobilazi probleme odstupanja.

Estetika

Upotreba slučajnih vrednosti u umetnosti postaje sve više popularnija. Neki se vode idejom muzike koja ne može da se predvidi i koriste slučajne vrednosti za stvaranje muzike. Slučajne vrednosti su dale umetnicima metod pomoću kojih mogu da se izdvoje od ostalih.

4 Pregled predloženog rešenja

4.1 Vršenje testa slučajnosti računanjem uzajamne informacije

Računanje uzajamne informacije predstavlja ključan korak u ispitivanju slučajnosti generisane sekvence. Metod se zasniva na izračunavanju uzajamne informacije podnizova unutar jedne sekvence. Količina uzajamne informacije između podnizova nam govori koliko su podnizovi povezani, odnosno odaju informaciju o generisanom nizu. Na ovaj način pokazuju prediktabilne osobine generatora.

Za testing uzorke generisane su sekvence iz tri različita izvora:

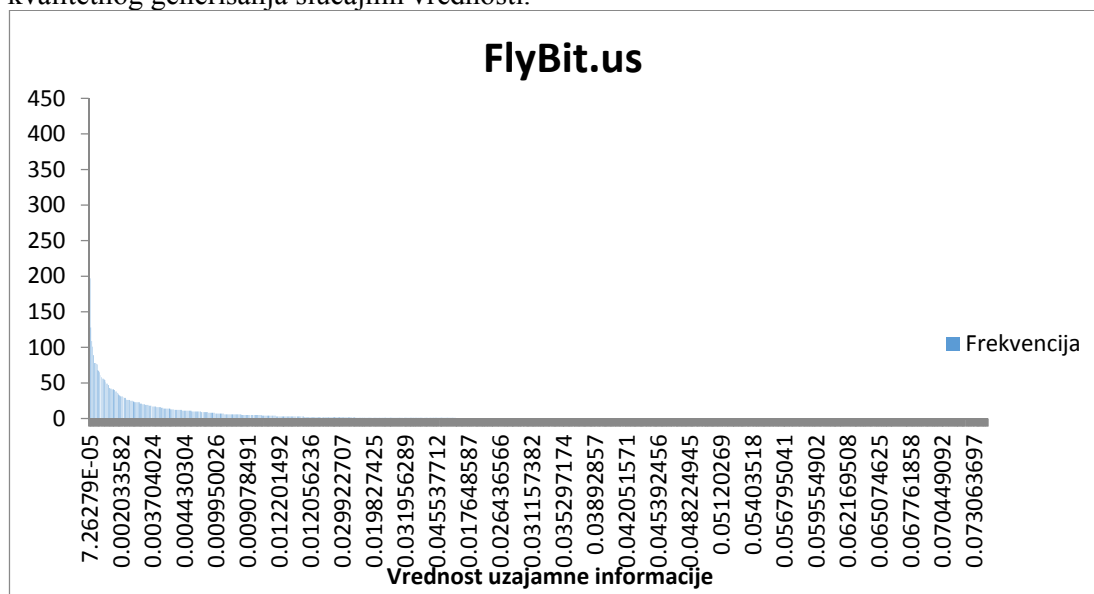
- FlyBit.us
- Random.org
- Bitmapa sekvence generisane pomoću php.rand()

Metod se primenjuje na podnizove generisanih sekvenci i ukoliko je vrednost uzajamne informacije između podnizova veća od vrednosti 0.01, smatra se da testirana sekvenca nije slučajna i generisani niz se odbacuje.

Generisane sekvence imaju dužinu od 12.800 bita, koje su podeljene u stotinu podnizova od po 128 bitova. Dužina podniza je određena sa predumišljajem, kako bi se koristila kao kriptološki ključ kod AES(engl. *Advanced Encryption Standard*) blokovske šifre. Izvršena su merenja maksimalne, minimalne i prosečne količine uzajamne informacije testiranih podnizova.

Testiranje generatora slučajnosti FlyBit.us

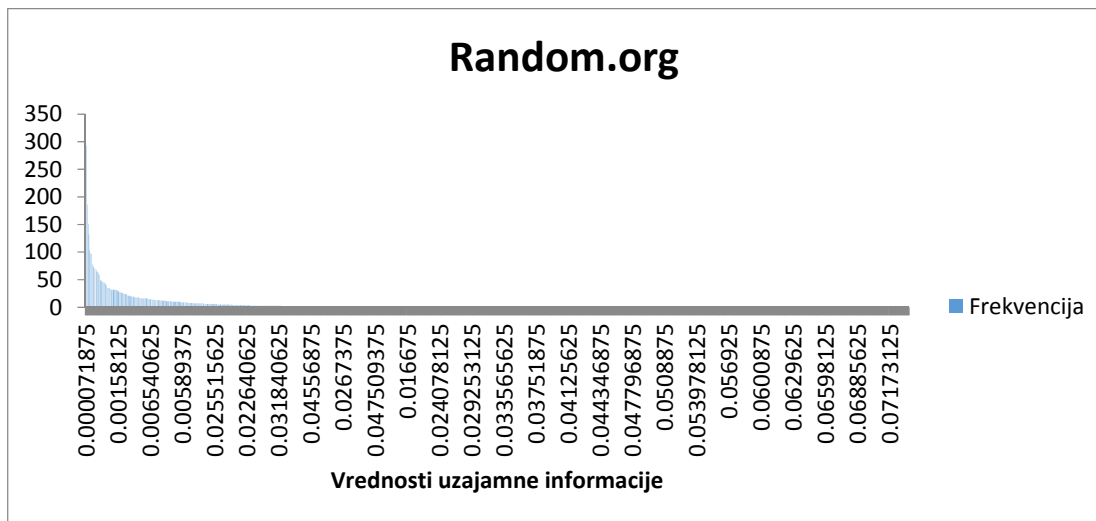
FlyBit je servis na mreži, predstavlja pravi generator slučajnosti koji generiše slučajne sekvence pomoću položaja civilnog avio saobraćaja u realnom vremenu. Vršena su merenja količine uzajamne informacije, uspešnost merenja je velika, upravo iz razloga kvalitetnog generisanja slučajnih vrednosti.



Slika 10- Prikaz frekvencije izmerene zajedničke informacije generatora flybit.us

Testiranje generatora slučajnosti Random.org

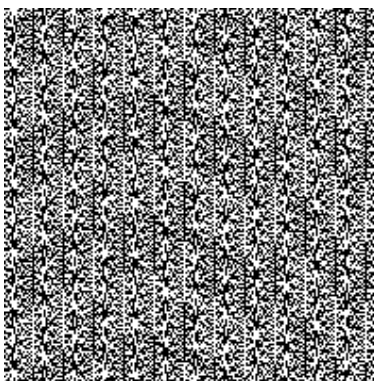
Random.org predstavlja pravi slučajni generator slučajnosti koji generiše slučajnost na osnovu atmosferskog šuma. Vršena su merenja količine uzajamne informacije, rezultati merenja pokazuju da je generisani niz istinski slučajan i ne pokazuje osobine predvidivosti.



Slika 11 - Prikaz frekvencije izmerene zajedničke informacije generatora Random.org

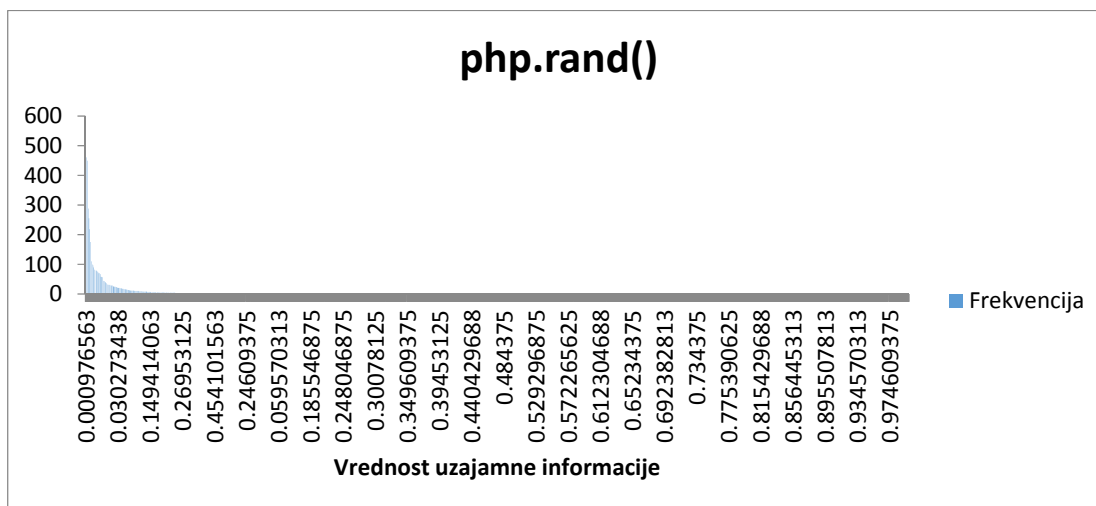
Testiranje slučajnosti upotrebom bitmapa sekvence generisane pomoću php.rand()

Funkcija php.rand() generiše pseudoslučajnu sekvencu. Korišćena bitmap sekvencu za test uzorak je predstavljena na sledećoj slici:



Slika 12 – Bitmap sekvencu generisana pomoću php.rand()

Ekstrakcija bitova se vršila konvertovanjem bitmap sekvence u binarni oblik. Merenja koja su vršena pokazuju da ovaj generator slučajnosti nije pogodan za kriptografsku upotrebu. Zbog velike količine uzajamne informacije izmerene u podnizovima ove sekvence, ovaj generator pokazuje osobine predvidivosti koje se mogu upotrebiti za kriptanalizu.



Slika 13-Prikaz frekvencije izmerene uzajamne informacije generisane bitmap sekvence php.rand()

Generator	Prosečna količina uzajamne informacije
flybit.us	0.005806
Random.org	0.005619
php.rand()	0.041977

Tabela 3- Prikaz prosečne količine zajedničke informacije iz različitih informacionih izvora

	flybit.us	Random.org	php.rand()
min	0.000000	0.000000	0.000000
max	0.074371	0.073600	1.000000

Tabela 4 – Prikaz opsega uzajamne količine infromacije iz različitih infromacionih izvora

Na osnovu podataka iz priloženih tabela možemo zaključiti da predloženi metod daje ključne informacije u vidu određivanja kvaliteta nekog izvora slučajnosti.

5 Zaključak

Razna sigurnosna rešenja oslanjaju se na upotrebu kriptografskih mehanizama, pa sama upotreba generatora slučajnosti pronalazi široku primenu. Zbog toga treba istaći da je generisanje slučajnosti od krucijalnog značaja za mnoge kriptografske mehanizme koji imaju veliki uticaj na sigurnost i skreće se pažnja u vidu estimacije kvaliteta slučajnosti kako bi se izvršio pravilan odabir kriptološki sigurnog generatora slučajnosti.

Generisanje visokog kvaliteta slučajnosti je najteži korak u postizanju sigurnosti. Pravljenje istinski slučajnih generatora je teško i izuzetno sporo, upravo zbog toga što se determinizam ogleda i u hardverskim i u softverskim implementacijama. Kako bi pronašli slučajnost, potrebno je potražiti je van klasičnog radnog okruženja.

Uvođenjem novog testa slučajnosti možemo sa većom sigurnošću zaključiti da li ispitivani generator poseduje željena svojstva i kao takvog ga okarakterisati da li je pogodan za upotrebu u kriptografiji.

5.1 Predlog za budući rad

Predlog za budući rad se ogleda u razvoju metoda za postdigitalnu obradu. Na ovaj način bi se poboljšala entropija izvora u vidu postizanja uniformnosti slučajne ili pseudoslučajne sekvence. To bi obezbedilo bolje rezultate na testovima slučajnosti i sigurniju upotrebu u vidu korišćenja generisanih sekvenci za kriptološke ključeve. Na taj način bi se poboljšala sigurnost celokupnog kriptografskog sistema.

6 Literatura

- [1] Coveyou, Robert. "Random Number Generation Is Too Important to Be Left to Chance," *Studies in Applied Mathematics*, III (1970), pp. 70-111
- [2] Milan Milosavljević, Miloš Milenković, Saša Adamović. "Generatori Pseudoslučajnih Brojeva", *Naučni skup Sinergija*, 2010
- [3] Dragan Pleskonjić, Nemanja Maček, Borislav Đorđević, Marko Carić. "Sigurnost Računarskih Sistema i Mreža", Mikro knjiga, 2007
- [4] Bruce Schneier. "Applied Cryptography", Wiley Publishing, 1996
- [5] Mladen Veinović, Saša Adamović. "Kriptologija 1", Univerzitet Singidunum, Beograd, 2013
- [6] NIST. "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications"
- [7] D.E. Eastlake, S.D Crocker, and J.I. Schiller, "Randomness Requirements for Security", RFC 1750, Dec 1994
- [8] Çetin Kaya Koç. "Cryptographic Engineering", Springer, 2009
- [9] David K. Gifford. "Natural Random Numbers", MIT, 1988
- [10] Mario Stipčević, Çetin Kaya Koç. "True Random Number Generators"
- [11] Charmaine Kenny. "Random Number Generators: An Evaluation and Comparison of Random.org and Some Commonly Used Generators", Trinity College Dublin, 2005
- [12] Milan Milosavljević, Saša Adamović. "Kriptologija 2", Univerzitet Singidunum, Beograd, 2014
- [13] Milan Milosavljević, Saša Adamović. "Osnovi Teorije Informacija i Kodovanja". Beograd, 2013

7 Appendix

Prilog izvornog koda za računanje uzajamne informacije

```
% integer values.
%
% INPUT:
% X --> first image
% Y --> second image (same size of X)
%
% OUTPUT:
% M --> mutual information of X and Y
%
% Written by GIANREGORIO Generoso.
% DATE: 04/05/2012
% E-MAIL: ggiangre@unisannio.it
%


---


X = double(X);
Y = double(Y);

X_norm = X - min(X(:)) + 1;
Y_norm = Y - min(Y(:)) + 1;

matAB(:,1) = X_norm(:);
matAB(:,2) = Y_norm(:);
h = accumarray(matAB+1, 1); % joint histogram

hn = h./sum(h(:)); % normalized joint histogram
y_marg=sum(hn,1);
x_marg=sum(hn,2);

Hy = - sum(y_marg.*log2(y_marg + (y_marg == 0))); % Entropy of Y
Hx = - sum(x_marg.*log2(x_marg + (x_marg == 0))); % Entropy of X

arg_xy2 = hn.*(log2(hn+(hn==0)));
h_xy = sum(-arg_xy2(:)); % joint entropy
M = Hx + Hy - h_xy; % mutual information
%M = h_xy; % ovo se koristi samo za racunanje zajednicke
```