VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2019 Bc. Martin Šelinga

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

SOFTWARE PRO HODNOCENÍ ZDROJŮ ENTROPIE
SOFTWARE FOR ENTROPY RESOURCE EVALUATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Martin Šelinga**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Karel Burda, CSc.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Martin Šelinga ***ID:*** 164412
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Software pro hodnocení zdrojů entropie

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte a popište problematiku zdrojů entropie a jejich hodnocení. Na tomto základě vytvořte program pro hodnocení zdrojů entropie, který bude založen na standardu NIST SP 800-90B. Následně pomocí svého programu otestujte běžně využívané zdroje entropie a získané výsledky analyzujte.

DOPORUČENÁ LITERATURA:

[1] Turan M. S. aj.: Recommendation for the entropy sources used for random bit generation. NIST SP 800-90B. Gaithersburg: National Institute of Standards and Technology, 2018. Dostupné na: https://bit.ly/2NrBGI8

[2] Burda K.: Kryptografické generátory. Sdělovací technika, 2016, č. 6.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 16.5.2019

Vedoucí práce: doc. Ing. Karel Burda, CSc.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Táto diplomová práca je zameraná na skúmanie zdrojov entropie. Obsahuje popis generátorov náhodnych čísel a testov používaných na hodnotenie kvality entropie.Súčasťou je vytvorenie programu na jej hodnotenie a generátory pre operačné systémy Linux a Windows. Následne bolo vykonané meranie entropie na fyzických pracovných staniciach a Cloudových prostrediach.

KĽÚČOVÉ SLOVÁ

Entropia, generovanie náhodných čísel, hodnotenie entropie, NIST, C++, AWS, Azure

ABSTRACT

This thesis is focused on exploring the sources of entropy. It includes a description of random number generators and tests used to evaluate entropy quality. Random number generator for Windows and Linux OS was created together with software for entropy evaluation. Subsequently, measurement of entropy was performed on physical workstations and Cloud environments.

KEYWORDS

Entropy, generation of random numbers, entropy evaluation, NIST, C++, AWS, Azure

ŠELINGA, Martin. *Software pro hodnocení zdrojů entropie*. Brno, Rok, 83 s. Diplomová práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: doc.Ing. Karel Burda, CSc.

VYHLÁSENIE

Vyhlasujem, že som svoju diplomovú prácu na tému „Software pro hodnocení zdrojů entropie“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád by som sa poďakoval pánovi docentovi Ing. Karlovi Burdovi CSc. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Brno ....

podpis autora

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výzkum popsaný v tejto diplomovej práci bol realizovaný v laboratóriách podporených projektom SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno ....

podpis autora

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

Obsah

Úvod **13**

1 Entropia **14**
1.1 Entropia v informačnej teórii 15
1.1.1 Použitie entropie v kryptografii 17
1.1.2 Pojmy súvisiace s entropiou 17
1.2 Generátory náhodných čísel 18
1.2.1 Náhodné generátory 19
1.2.2 Deterministické generátory 20
1.2.3 Hybridné generátory 21

2 Zdroje Entropie **26**
2.1 Model zdroja entropie 26
2.1.1 Zdroj šumu 26
2.1.2 Operačné testy 27
2.2 Príklady zdrojov entropie 27
2.2.1 Klávesnica a myš 27
2.2.2 Sieťová karta 27
2.2.3 Teplota CPU, GPU, HDD 28
2.2.4 RAM 28
2.2.5 Dátum a čas 28
2.2.6 Zmeny obrazu 28
2.3 Entropia v systéme Windows - CryptoAPI 29
2.3.1 Vnútorná štruktúra Windows RNG 30
2.4 Entropia v systéme Linux - /dev/random 31
2.5 Známe útoky na RNG 33
2.5.1 Netscape SSL 33
2.5.2 Kerberos 4.0 33
2.5.3 Virtuálny stroj JAVA 34
2.5.4 Linux RNG 34

3 Operačné testy **35**
3.1 Typy operačných testov 35
3.1.1 Operačné testy pri štarte 35
3.1.2 Kontinuálne operačné testy 36
3.1.3 Operačné testy na vyžiadanie 36
3.2 Požiadavky na operačné testy 36

3.3 Test počtu opakovaní . . . 37
3.4 Adaptívny proporčný test . . . 38

4 Validácia zdroja entropie 40
4.1 Overovací proces . . . 40
4.1.1 Zber dát . . . 40
4.2 Požiadavky na overovacie testy . . . 41
4.2.1 Požiadavky na zdroj entropie . . . 41
4.2.2 Požiadavky na zdroj šumu . . . 41
4.2.3 Požiadavky na zber dát . . . 42
4.3 Nezávislé a identicky distribuované zdroje šumu . . . 43
4.3.1 Permutačné testovanie . . . 43
4.3.2 Dodatočné Chí kvadrát testy . . . 50

5 Stanovanie minimálnej entropie 54
5.1 Estimačné testy . . . 54
5.1.1 Odhad najčastejšej spoločnej hodnoty . . . 54
5.1.2 Test kolízii . . . 55
5.1.3 Markov model . . . 55
5.1.4 Odhad t triedy . . . 57
5.1.5 Najdlhší opakovaný subreťazec . . . 57
5.1.6 Odhad najčastejšie opakujúceho sa okna . . . 58
5.1.7 Viacnásobná MMC predikcia . . . 58
5.1.8 LZ78Y predikcia . . . 58

6 Praktická časť 60
6.1 Generátor pre operačný systém Windows . . . 60
6.2 Generátor pre operačný systém Linux . . . 63
6.3 Program na vyhodnocovanie entropie . . . 64

7 Meranie a analýza zdrojov entropie 69

8 Záver 70

Literatúra 71

Zoznam symbolov, veličín a skratiek 74

Zoznam príloh 75

A Výsledky meraní **76**
A.1 Windows 76
A.1.1 Windows 76
A.1.2 WindowsAWS 77
A.1.3 WindowsAzure 78
A.2 Linux 79
A.2.1 Linux 79
A.2.2 LinuxAWS 80
A.2.3 LinuxAzure 81

B Obsah priloženého CD **83**

Zoznam obrázkov

1.1 Schéma tepelného motora [2]. 15
1.2 Schéma všeobecného komunikačného systému [17]. 16
1.3 Náhodný generátor procesora Intel Pentium III [15]. 19
1.4 Bloková schéma deterministického generátora Hashgen [15]. 21
1.5 Schematické znázornenie hybridného generátora [15]. 22
1.6 Vývojový diagram generátora Hash_DRGB [15]. 23
1.7 Prevodná funkcia F(Str,M) pre Hash_DRGB [15]. 24
2.1 Model zdroja entropie [21]. 26
2.2 Fotka steny s lávovými lampami ktoré slúžia ako zdroj entropie[7]. . . 29
2.3 Implementácia zdrojov entropie OS Windows[20]. 31
2.4 Schéma generátora v operačnom systéme linux [9]. 32
6.1 Generátor pre operačný systém Windows. 60
6.2 Generátor pre operačný systém Linux. 63
6.3 Algoritmus pre hodnotenie entropie[21] 65
6.4 Program pre hodnotenie entropie. 66

Zoznam tabuliek

3.1 Príklady medzných hodnôt ore Adaptatívny proporčný test[16] 39
4.1 Výsledky testu nezávislosti 51
4.2 Alokácia párov pre nezávislé testy 51
7.1 Výsledky meraní výsledok H_bitstring 69

Zoznam výpisov

2.1 CryptGenRandom API [8] 29
2.2 Príklad využitia CryptGenRandom [8] 30

Úvod

Entropia je veličina ktorá popisuje mieru neurčitosti systému. Kryptografické generátory náhodných bitov vyžadujú zdroj šumu ktorý bude poskytovať digitálny výstup s určitou mierov nepredvídateľnosti.

Dôvodom pre túto nepredvídateľnosť je aby potencionálny útočník nedokázal v krátkom čase nejakým spôsobom vyrátať alebo uhádnuť kryptografické kľúče.

V tejto práci sa zamerám na teoretický popis entropie a generátorov náhodných čísel. Ďalej popíšem zdroje entropie a testy na hodnotenie kvality entropie.

Následne bude navrhnuté a naprogramované generátory náhodných čísel ktorých výstup bude otestovaný vytvoreným programom na testovanie a hodnotenie zdrojov entropie vychádzajúci z odporučení NIST.

Generátory budú použité na vygenerovanie súborov s náhodnými dátami na rôznych platformách (Windows, Linux) vo virtuálnych cloudových prostrediach.

Tieto výstupy budú otestované navrhnutým program na hodnotenie entropie a vzájomne porovnané a vyhodnotené.

1 Entropia

V tejto kapitole bude vysvetlený pojem Entropia a pojmy s ňou súvisiace z fyzikálneho pohľadu a zároveň to ako je chápana a aké má využite z pohľadu kryptografie.

Entropia je veličina ktorú v roku 1850 zaviedol nemecký fyzik Rudolf Clausius. Slovo entropia má pôvod v gréčtine predpona *ex-* znamená v rámci, zatiaľ čo koreň *-trop* znamená transformácia[1].

Entropia patrí medzi základné a veľmi dôležité pojmy v oblasti fyziky,matematiky, teórie pravdepodobnosti a teórie informácie. Táto veličina umožňuje sledovať nevratnosť zmeny deja. Entropia ostáva pri samovoľných zmenách izolovaných systémov iba bezo zmeny prípadne môže rásť. Popisuje degradáciu tepla ku ktorej dochádza pri nenávratných zmenách. Entropia je stavová veličina, jej zmena je závislá iba od stavu na počiatku a na konci deja. Pri všetkých premenách energie dochádza ku strate a postupnej degenerácii energie a nárastu entropie. Degeneráciu je možné chápať ako stratu schopnosti konať prácu.

Claudius zaviedol pojem entropia na to aby bolo možné kvantitatívne meranie spontánnej zmeny. Entropia podľa Claudiusa je presný spôsob vyjadrenia druhého termodynamického zákona. Claudiusova forma druhého termodynamického zákona hovorí že spontánna zmena pre nezvrátiteľný proces v izolovanom systéme vždy vedie k nárastu entropie. Izolovaným systémom chápe taký systém ktorý si nevymieňa teplo alebo prácu s jeho okolím. Ako príklad je možné uviesť kocku ľadu ktorá je položená na rozpálenú platňu. Táto platňa a kocka ľadu tvoria izolovaný systém ktorého entropia sa zvyšuje pri topení ľadu [2].

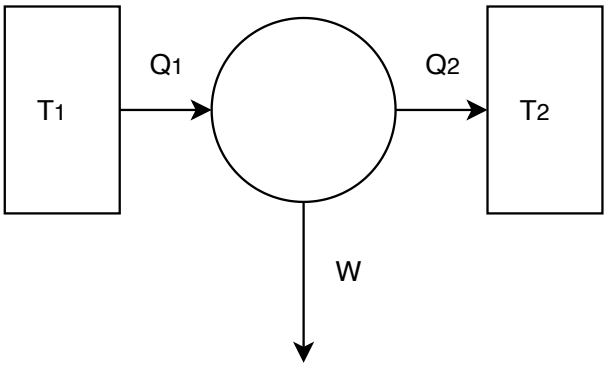
Podľa Claudiusovej definície entropie ak teplo Q je dodané do tepelného zásobníka s teplotou T ktorá je väčšia ako absolútna nula, potom sa entropia zvýši o $\Delta S = Q/T$. Z tejto rovnice dokážeme odvodiť vzťah ktorý je ekvivalentný s obvyklou definíciou.

Predpokladajme že máme dva tepelné zásobníky R_1 a R_2 ktoré majú teplotu T_1 a T_2 (kocka ľadu a rozpálená platňa). Ak množstvo tepla Q prúdiace z R_1 do R_2 potom sa celková zmena entropie rovná:

$$\Delta S = Q\left(\frac{1}{T_1} - \frac{1}{T_2}\right) \tag{1.1}$$

čo dokazuje že $T_1 > T_2$. Z tohoto pozorovania je možné utvoriť záver že teplo sa nikdy spontánne nevytvorí bez dodania energie respektíve z chladnej látky teplá. Ekvivalentne je možné usudzovať že celková zmena entropie musí byť pozitívna pre spontánne prúdenie tepla. Ak sa $T_1 = T_2$ zásobníky tepla sú v rovnováhe, neprebieha prúdenie tepla a $\Delta S = 0$

Podmienka $\Delta S \geq 0$ stanovuje maximálnu účinnosť tepelných motorov, preto systémy ako benzínový alebo parný motor pracujú v cykloch.



Obr. 1.1: Schéma tepelného motora [2].

Prepokladajme že motor absorbuje teplo Q_1 z R_1 a vylúči teplo Q_2 do R_2 pre každý dokončený cyklus. Úspora energie vykonaná za jeden cyklus je $W = Q_1 - Q_2$ a celková zmena entropie je:

$$\Delta S = \left(\frac{Q_1}{T_1} - \frac{Q_2}{T_2} \right) \tag{1.2}$$

aby bola vykonaná práca W čo najväčšia, Q_2 musí byť v pomere ku Q_1, nesmie však byť nulové pretože to by znamenalo že ΔS by dosiahlo negatívnu hodnotu čo by bolo porušenie druhého termodynamického zákona. Najmenšia možná hodnota Q_2 korenšponduje s podmienkou $\Delta S = 0$ a teda:

$$\left(\frac{Q_1}{Q_2} \right)_{min} = \frac{T_2}{T_1} \tag{1.3}$$

Rovnica 1.3 vyjadruje základné obmedzenie efektivity všetkých tepelných strojov. Proces pri ktorom je $\Delta S = 0$ je reverzibilný a aj najmenšia zmena by spôsobila spustenie tepelného stroja na spätný chod ako chladničku[2].

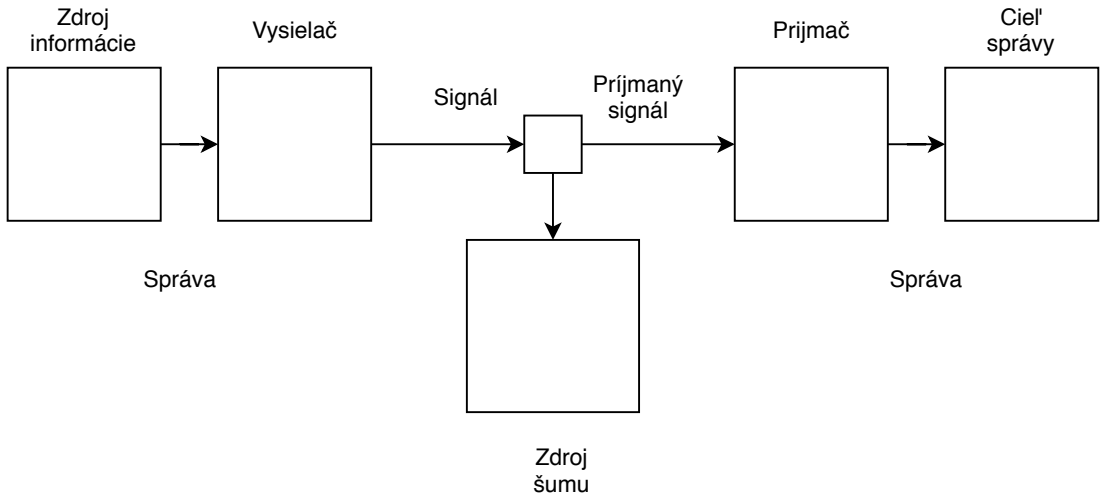
1.1 Entropia v informačnej teórii

V nasledujúcej kapitole je entropia rozobratá z pohľadu teórie informácie. Shanon publikoval v roku 1948 článok v ktorom definoval entropiu pre použitie v informačnej teórii. Predstavil v ňom komunikačný model viz. Obr.1.2 v ktorom je zdroj správy entita ktorá ju pôvodne vytvorila. Zdrojom správy je obvykle človek ale v tomto prípade to môže byť aj zviera, počítač alebo iný objekt. Enkóder je objekt ktorý správu premieňa na signál ktorý sa prenáša ku príjemcovi správy. Je niekoľko ciest ktorými sa dá tento model aplikovať na ľudí majúcich konverzáciu po telefóne. Napríklad zvuk ktorý produkujú ústa sa dá považovať za správu a telefónne slúchadlo za enkóder. Tento enkóder prekladá zvuk z úst na elektrický signál ktorý putuje

elektrickou sieťou. Alternatívne môže niekto za zdroj správy považovať mysel a ústa spolu so slúchadlom za enkóder[2].

Kanál je médium ktoré prenáša správu. Kanál môžu byť káble v prípade drôtového spojenia alebo vzduch v prípade bezdrôtového spojenia či optické káble.

Za šum sa považuje čokoľvek čo koliduje respektíve ruší vysielaný signál. V prípade telefonného hovoru to môžu byť šum spôsobený statickou elektrinou na linke, ozvena z inej linky prípadne ruch z pozadia. Signály prenášané opticky môže trpieť z nevhodných poveternostných podmienok alebo nadmernej vlhkosti. Zdroje šumu sa líšia od prípadu na prípad podľa komunikačného systému. Jeden komunikačný systém môže mať niekoľko zdrojov šumu avšak za určitých podmienok je s nimi možné počítať iba s ako jedným zdrojom šumu. [17].



Obr. 1.2: Schéma všeobecného komunikačného systému [17].

Dekóder je objekt ktorý konvertuje prijatý signál do formy ktorú dokáže príjemca spracovať. V prípade telefónneho hovoru to môže byť slúchadlo alebo elektronické obvody. Opäť záleží na perspektíve akou sa na problematiku budeme pozerať a niektoré zdroje uvádzajú ako súčasť dekóderu aj sluchový aparát človeka.

Príjemca správy je objekt ktorý dostane správu. Analogicky k odosieľatelovi to môže byť človek, zviera, počítač alebo iný objekt.

Shanonová teória sa zaoberá primárne enkodérom, kanálom, zdrojom šumu a dekóderom. Ako vyplýva z príkladov vyššie cielom teórie je sústrediť sa na signály a to ako ich preniesť presne a efektívne a nezaoberať sa tým čo sa chápe pod jednotlivými zložkami systému[2].

1.1.1 Použitie entropie v kryptografii

Kryptografia je nevyhnutná na zabezpečenie dát či už v prípade ich prenosu cez internet alebo ich uloženia na pamätové zariadenia. V minulosti sa v kryptografii využívali algoritmy a kódy ktoré boli tajné. Toto však bolo veľmi nepraktické v prípade že strany ktoré sa navzájom nepoznali vopred chceli spolu komunikovať bezpečne.

V súčastnosti bezpečnosť komunikácie závisí predovšetkým od toho ako silné tajné kryptografické kľúče sa využívajú. Kľúč je silný len do takej miery ako ťažké je ho uhádnuť, inými slovami ako moc je náhodný. Bezpečnostné štandarty vyžadujú aspoň 112 bitovú silu kryptografických kľúčov čo znamená že pokus o prelomenie by sa rovnal minimálne ekvivalentu 2^{112} možný kombinácii kľúčov. Hoci iné faktory ako spolahlivosť kryptografickeho algoritmu a jeho bezpečná implementácia hrajú veľkú úlohu v bežpečnosti kryptografie táto práca je zameraná ná kvalitu náhodnosti generácie kľúčov.

1.1.2 Pojmy súvisiace s entropiou

Náhodnosť

Pod náhodnosťou rozumieme malý počet určitých vzorcov a predpovedateľnosti udalostí. Náhodná sekvencia udalostí, symbolov alebo krokov ktorá nevytvorí rozpoznateľný vzorec alebo kombináciu. Ojedinelé náhodné udalosti sú z definície nepredvídateľné. Avšak v prípade ich že sa často opakujú s určitou frekvenciou počas veľkého množstva pokusov dokážeme s určitou pravdepodobnosťou predpovedať výsledok. Na príklade hodu dvoma kockami vieme konštatovať že výsledok jedného hodu je náhodný avšak pri sérii opakovaní súčet 7 sa bude opakovať dva krát častejšie ako súčet 4. Z tohoto pohľadu náhodnosť je merítko pravdepodobnosti výsledku a aplikuje sa v pravdepodobnosti a informačnej entropii.

Relativita entropie

Za užitočné merítko entropie môžeme považovať takzvanú relativitu entropie. Relativita entropie funguje rovnako dobre v diskrétnom tak spojitom prípade distribúcie relatívnej entropie. Definíciou je Kullback-Leiblerova odchýlka od rozdelenia na referenčnú mieru p a je absolútne spojitá vzhľadom na k opareniam m teda p formulácia $(dx) = f(x)m(dx)$ pre nezáporné m integrované funkcie f z m prvého integrálu potom relatívna entropia môže byť definovaná ako:

$$D_{KL}(p||m) = \int log(f(x))pdx = \int f(x)log(f(x))mdx \tag{1.4}$$

V takejto podobe relatívna entropia zovšeobecňuje ako diskréna entropia je. V prípade že opatrenie m je samo o sebe rozdelením pravdepodobnosti, relatívna entropia je nezáporná, nulová v prípade že $p{=}m$. Pre každé opatrenie je definovaný priestor a preto koordinácia nezávislej a nemennej re-parametrizácie v prípade že jedna rada zohľadňuje transformáciu opatrenia m. Relatívna, implicitná a diferenciálna entropia závisí na referenčnom opatrení m [14].

Hustota informácie

Pri šifrovaní je entropia používaná ako miera nepredvídateľnosti kryptografického kľúča. Útočník môže uhádnuť kľúč aj na prvý pokus a teda entropia nedokáže popísať počet pokusov nutných k rozšifrovaniu kľúča hrubou silou. V prípade 128bitového kľúča ktorý je náhodne vygenerovaný má 128bitov entropie čo je 2^{128-1} nutných kombinácii k prelomeniu hrubou silou. Ak budeme uvažovať 1000 miestne binárne číslo v prípade že má 1000 bitov entropie dá sa entropia považovať za výbornú. Tak isto je možné entropiu považovať za dobrú v prípade že má 999 bitov entropie rovnomerne rozložených. Avšak v prípade kde je 999 bitov entropie pričom prvá číslica je pevná a zvyšok je náhodný, prvá číslica šifrovaného textu je neentropická [6]. S hustotou informácie súvisí kompresia dát. Poznáme stratovú a bezstratovú kompresiu. Pri bezstratovej znižujeme bity súboru a odstraňujeme štatistickú nadbytočnosť dát. Stratová kompresia znižuje bity súboru odstránením zbytočnej informácie. V prípade maximálne skomprimovaného súboru dosahujeme entropie ideálneho zdroja.

Entropia je často používaná ako charakteristika informačného obsahu zdroja dát, tento informačný obsah nie je absolútny. Zdroj , ktorý vždy generuje ten istý symbol má mieru entropie 0. Definícia významu symbolu záleží od abecedy tj. zdroj ktorý produkuje reťazec A-B-A-B-A-B ... pričom A a B nasleduje periodicky. Pravdepodobnostný model považujeme za jednotlivé písmena ako nezávislý, miera entropie je 1bit na znak. Ak sa za postupnosť považuje AB-AB-AB-AB..., teda symbol je dvojmiestny blok, miera entropie potom predstavuje 0 bitov na znak [6].

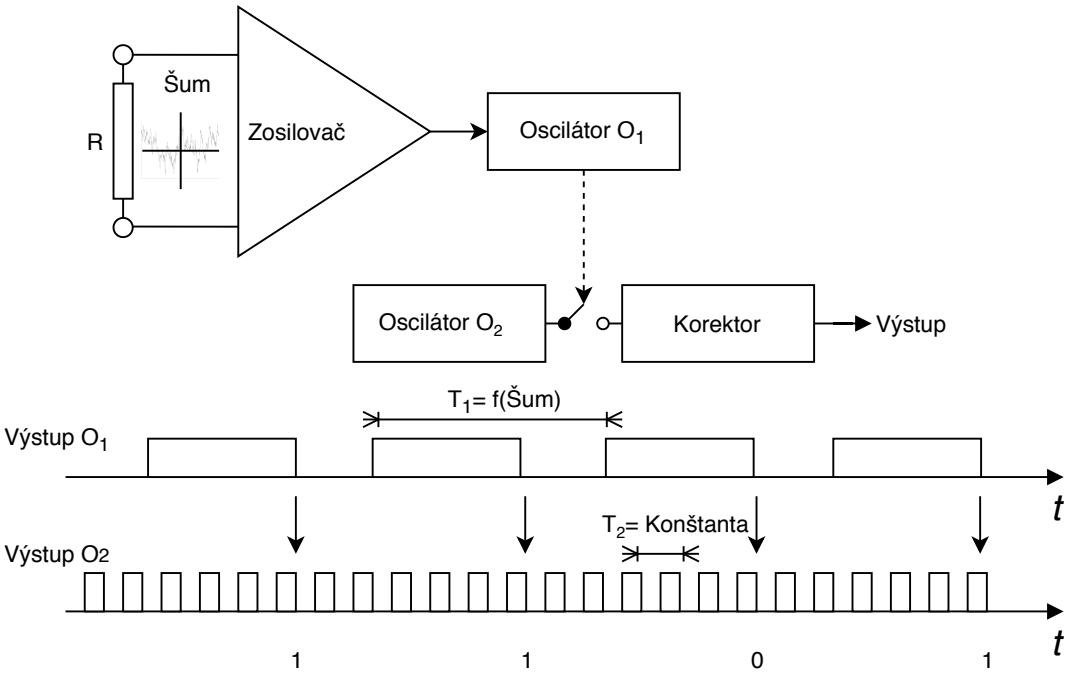
1.2 Generátory náhodných čísel

Generátory náhodných čísel majú využitie v hazarde, štatistike, vede a kryptografii. Použitie v kryptografii zahŕňa napríklad využitie v šifrovacích protokoloch typu SSL *(Secure Socket Layer)*. Pri šifrovacích kľúčoch týchto protokolov je dôležité aby nebolo možné tieto kľúče predom odhaliť či už celé alebo ich časť. Odvodenie generovaného kľúča by útočníkovi poskytlo výhodu pri pokuse o kompromitáciu spojenia keďže by podstatne mohol skrátiť dobu potrebnú na prelomenie kryptosystému. Pri

predchádzaní tohoto problému sa pre generátory stanovujú parametre volené náhodne podľa rovnomerného rozdelenia.

K získavaniu nepredvídateľných hodnôt kľúčov a ďalších kryptografických parametrov sa používajú gerátory nepredvítateľných bitov. Tieto generátory je možné rozdeliť na:

- Náhodné generátory
- Deterministické generátory
- Hybridné generátory



Obr. 1.3: Náhodný generátor procesora Intel Pentium III [15].

1.2.1 Náhodné generátory

Náhodné generátory označované aj ako generátory náhodných postupností *(Non-deterministic Random Bit Generator alebo aj True Random Bit Generator)* využívajú kvantové deje ako napríklad atómu rádioaktívneho prvku. Medzi ďalšie deje využiteľné pre generovanie môžu využívať šumy prípadne fluktuácie. Napríklad teplotný šum rezistorov alebo kolísanie kmitočtu oscilátora. Postupnosti bitov odvodené z náhodného procesu sa často upravujú dodatočnými algoritmami kvôli odstráneniu štatistických závislostí. Ako príklad literatúra uvádza generátor procesoru Intel III ktorý možno vidieť na Obr. 1.3 kde je využitý Neumannov korektor na odstránenie štatictických závilostí [15].

Náhodným procesorom využívaným v tomto generátore je teplotný šum rezistora R, ktorý vzniká nepatrnými zmenami v dôsledku tepelného pohybu elektronov v materiále rezistoru. Šum rezistoru je zosilený o a následne je ním ovládaný napätením riadený oscilátor O_1. U tohoto oscilátoru závisí kmitočet na hodnote privedeného napätia čiže doba periódy oscilátora O_1 je v dôsledku náhodného šumu rezistoru náhodnou veličinou. Súčasťou procesoru je je vysoko stabilný oscilátor O_2. Tento oscilátor generuje krátke impulzy ktorých dĺžka je rovnaká ako dĺžka medzier medzi impulzami. Na výstupe oscilátoru O_2 je vzorkovaný zostupnou hranou impulzu O_1 čo je na Obr. 1.3 vyjadrené spínačom. V prípade že sa zostupná hrana vyskytne v dobe impulzu O_2 tak sa získaný vzorok stane pre korektor vstupným bitom b s hodnotou 1. Vzhľadom k náhodnej dĺžke T sa zostupná hrana môže zostupná hrana vyskytnúť i medzera medzi impulzami O_2. Hodnota bitu b potom bude 0. Ako dôsledok nežiaduceho kolísania frekvencie oscilátoru O_2 môže sa vyskytnúť situácia že počet núl a jednotiek v bitoch b nebude rovnaká. Z tohoto dôvodu sú dvojice bitov vedené do Neumannoveho korektoru ktorého účelom je vyrovnať počet jednotiek a núl. V prípade že sa vyskytne rovnaká hodnota oboch bitov korektor odstráni takúto dvojicu. Inak korektor z danej dvojice prenesie na svoj výstup druhý bit zo vstupnej dvojice. Na príklade z literatúry sa na vstupe nachádza dvojica (1,1) a (0,1) pričom prvá dvojica bude korektorom odstránená a z druhej sa prenesie na výstup druhý bit teda bit s hodnotou 1. Táto postupnosť korektorom odfiltrovaných bitov je výstupom popisovaného generátora [15].

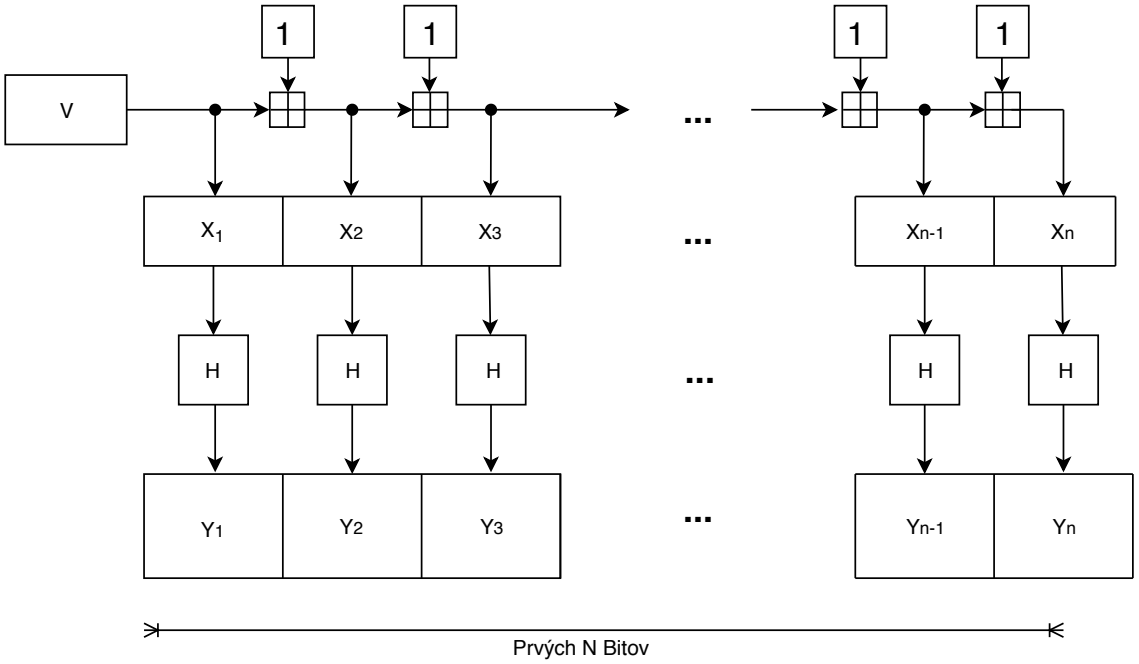
1.2.2 Deterministické generátory

Deterministické generátory *(Deterministic Random Bit Generator alebo Pseudorandom Bit Generator)* odvodzujú výstupné bity výpočetným postupom teda deterministicky z tajnej východzej hodnoty. Nazývajú sa tiež generátory pseudonáhodnej postupnosti. Pod pseudonáhodnosťou sa rozumie skutočnosť že výstupnú postupnosť tohoto generátoru nedokážeme od postupnosti náhodnej.

Mozgom deterministických generátorov je automat ktorý má konečný počet stavov. Východzí stav tohoto generátora určuje hodnota ktorá sa nazýva semeno (seed). Hodnota musí byť utajená pretože v prípade že sa ju dozvie útočník mohol by jednoducho predvídať výstupy generátoru.

Automat potom deterministickým spôsbom prechádza z jedného stavu do nasledujúceho pričom z aktuálnej hodnoty jeho stavu sa odvodzujú hodnoty bitov na výstupe. Ako príklad generátoru tohoto typu uvádza literatúra generátor Hashgen. Automatom je v tomto prípade čítač, na odvodenie nepredvídateľných bitov sa používa hashovacie funkcia. Bloková schéma je zobrazená na Obr.1.4 .

Generátor Hashgen je prakticky funkcia $G(V, N)$ kde V je semeno o dĺžke 440



Obr. 1.4: Bloková schéma deterministického generátora Hashgen [15].

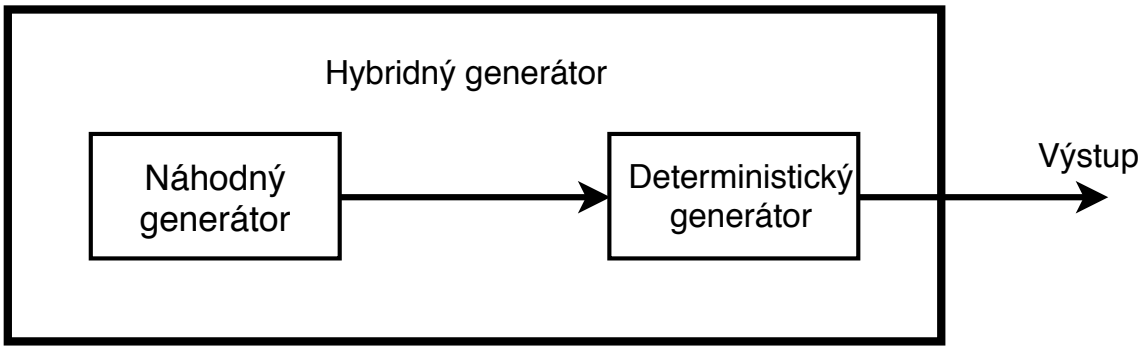
bitov a N je počet bitov ktoré sa musia vygenerovať. Semeno V slúži ako počiatočná hodnota X_1 čítača ktorý je v ďalšom cykle inkrementovaný o hodnotu 1 v aritmetike modulo 2^{240}. Na Obr.1.4 to predstavuje symbol $\boxplus$. To znamená že $X_{i+1} = (X_i + 1) mod^{440}$. Hodnota X_i je vedená na vstup hešovacej funkcie H kde získa blok bitov $Y_i = H(X_i)$ o dĺžke výstupu zvolenej hešovacej funkcie. Napríklad v prípade SHA-256 bude mať výstupný blok dĺžku 256 bitov. Avšak v prípade že potrebujeme nepredvídateľné číslo o dĺžke N = 384 bitov tak podľa Obr. 1.4 nám budú stačiť 2 hodnoty čítača tj. dve hešovania. Získame tak reťazec o dĺžke 2x256 = 512 bitov z ktorého použijeme 384 bitov.

Hešovacie funkcie sú konštruované ako funkcie s mnohými nelineárnymi iteráciami, z čoho vyplýva že z tohoto množstva iterácii a nelineárnych väzieb sa vplyv každého bitu zo vstupu X_i mnohonásobne a veľmi zložitým spôsobom uplatňujev hodnote na výstupe každého bitu Y_i. Preto ak zmeníme hodnotu čítača X_i iba v jedinom bite tak analýzou blokov Y_i Y_{i+1} nezistíme medzi týmito blokmi nejaké štatistické súvislosti. Oba bloky sa budú javiť ako náhodné a navzájom nezávislé [15].

1.2.3 Hybridné generátory

Ako vyplýva zo schematického zobrazenia hybridných generátorov na Obr.1.5. Z názvu je zrejmé že tento generátor je kombinácia predchodzích dvoch typov. Náhodný generátor slúži ako zdroj semien pre deterministický generátor ktorého výstup

je zároveň výstup hybridného generátora. Kombinácia oboch generátorov predstavuje kompromis ktorý poskytuje vysokú rýchlosť generovania čo je prednosť deterministických generátorov a zároveň dostatočnú úroveň bezpečnosti keďže semeno deterministického generátora sa totiž mení tak často že známe útoky na tento typ generátora nie sú efektívne.



Obr. 1.5: Schematické znázornenie hybridného generátora [15].

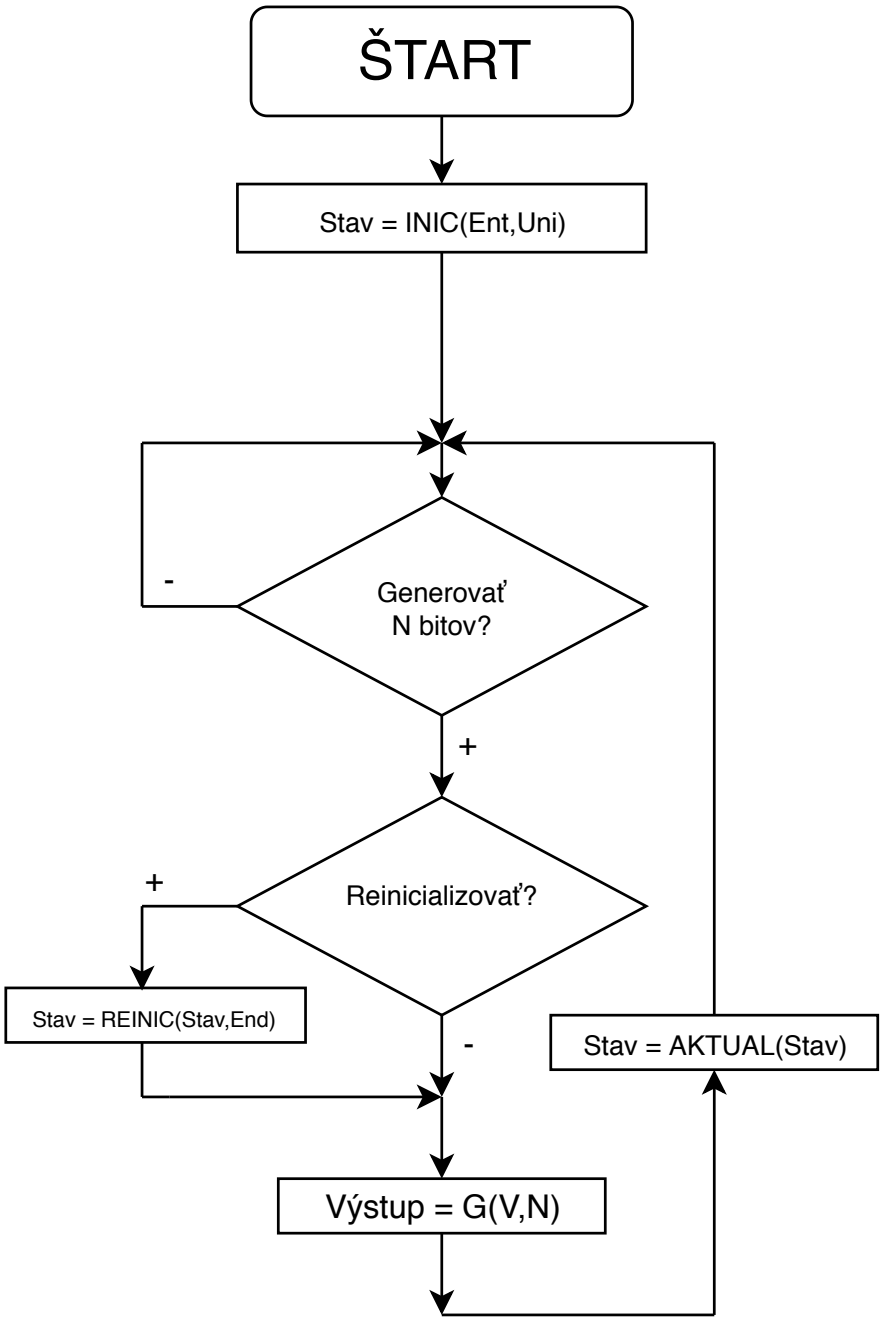
Náhodné generátory majú k dispozícii zdroje entropie s vysokou hodnotou entropie E. Tieto zdroje nemajú stále časové parametry, sú pomalé a nákladné. Toto iniciovalo snahu využiť bežné zdroje entropie ako je napr. šum na výstupe zvukovej karty či binárne záznamy o aktivitách užívateľa. Entropia týchto zdrojov je síce nízka ale kompresiou je vieme zvýšiť. V prípade že máme reťazec R o dĺžke M = 2560 bitov zo zdroja s E = 0.1 Sh/bit tak tento reťazec obsahuje entropiu o veľkosti S = M x E = 256 Sh. Hašovanie reťazca R pomocou funkcie SHA-256 môžeme entropiu skomprimovať do výstupného reťazca r o dĺžke m = 256bitov. Bity tohoto reťazca majú najvyššiu možnú hodnotu entropie $e = S/m$ = 1Sh/bit a reťazec r tak môžeme použiť ako semeno pre deterministický generátor.

Na Obr.1.6 je znázornený diagram hybridného generátora Hash_DRGB. Po zapnutí generátora (START) sa prevedie inicializácia generátora (INIC). Počas tejto inicializácie sa na základe reťazca *Ent* reťazec s požadovaným množstvom entropie ktorý je vyžiadaný od zdroja entropie a reťazca *Uni* - unikátny reťazec ktorým sa inicializuje daný generátor alebo daný štart a definuje počiatočný stav generátoru.

Tento stav je daný trojicou Stav = (V,C,J) kde V je 440 bitov dlhé semeno, C je 440 bitov dlhá konštanta určená k inicializácii a J poradové číslo najbližšej procedúry Generovať N bitov. Ako ďalší krok v slučke testuje či mu bol zadaný príkaz generovať N bitov. V kladnom prípade sa testuje či nie je potrebné previesť reinicializáciu. V prípade že je stavová premenná J väčšia ako stanovená hodnota J_{max}, tak je procedúra Generovať N bitov bola spustená toľko krát že je z bezpečnostných dôvodov žiaduce aby sa hodnota semena V modifikovala novou dávkou entropie. Po prípadnej reinicializácii deterministickým generátorom G(V,N) ktorý je znázornený

na Obr. 1.4 je ním vygenerované N nepredvítateľných bitov. Hybridný generátor následne zaktualizuje svoj stav (AKTUAL) a vráti sa do stavu v ktorom sa testuje či bol vydaný príkaz Generovať N Bitov.

Jadrom procedúr INIC (Inicializácia) REINIC (Reinicializácia) je prevodná funkcia F(Str,M) ktorej diagram je znázornený na Obr. 1.7. Touto funkciou sa entropia zo vstupného reťazca *Str* prevedie respektíve skomprimuje do výstupného reťazca o požadovanej dĺžke M = 440 bitov. Prevodná funkcia F funguje tak že sa najprv



Obr. 1.6: Vývojový diagram generátora Hash_DRGB [15].

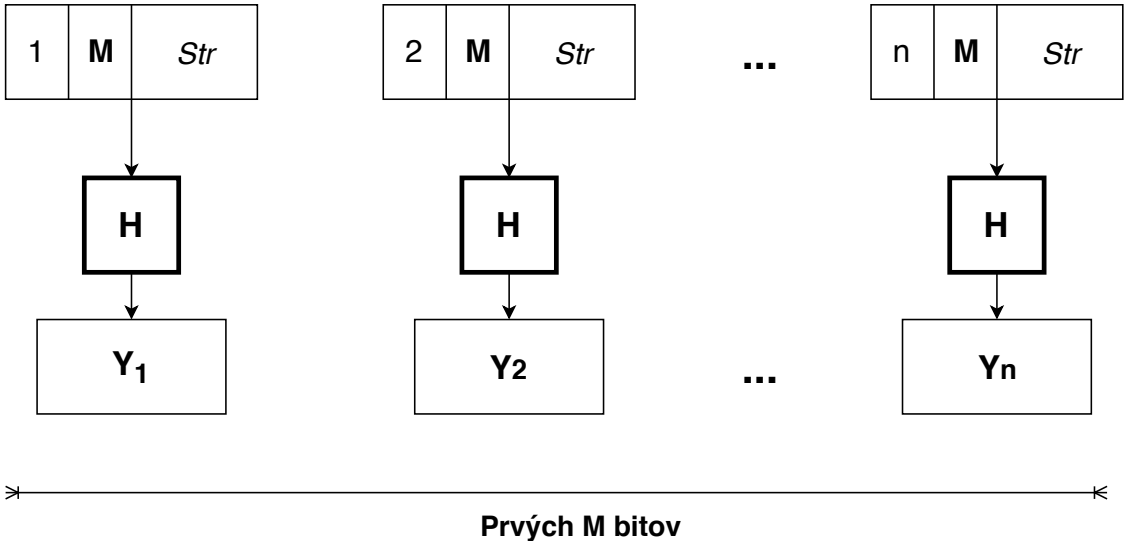
nastaví bajt i čítača na hodnotu 1 zaň sa pripoja 4 bajty ktoré vyjadrujú hodnotu M a za ne sa pripojí vstupný reťazec *Str*. Takto vzniknutý reťazec sa zahešuje čím sa získa blok Y_1. Hodnota čítača sa podľa potreby ďalej inkrementuje a reťazce $(i||M||Str)$ sa postupne hešujú do podoby blokov Y_1 do tej doby než sa ich celková dĺžka dostane k hodnote potrebnej na získanie požadovaných M výstupnych bitov.

V rámci inicializácie sa nastavuje počiatočný stav generátora (V,C,J) = INIC (Ent,Uni). Toto nastavenie je dané vzťahmi:

$$V = F(Str, 440), \text{ kde } Str = (Ent||Uni)$$

$$C = F(Str, 440), \text{ kde } Str = (0||V)$$

$$J = 1$$



Obr. 1.7: Prevodná funkcia F(Str,M) pre Hash_DRGB [15].

pričom tu čísla x a tak isto v nasledujúcich rovniciach s operátorom || majú význam bajtu s hodnotou x. Reťazec *Ent* je reťazec s požadovaným množstvom entropie, ktorá pri štarte vyžadovaná od zdroja a reťazec *Uni* je unikátny reťazec ktorým sa individualizuje daný generátor alebo daný štart. Obvykle sa jedná o aktuálny časový údaj.

Cieľom reinicializácie (REINIC) je modifikovať stav generátoru na základe jeho predchádzajucého stavu(V) a na základe novej dávky entropie (*Ent*):

$$V = F(Str, 440), \text{ kde } Str = (1||V||Ent)$$

$$C = F(Str, 440), \text{ kde } Str = (0||V)$$

$$J = 1$$

Pri aktualizácii (AKTUAL) sa stav generátoru iba upravuje na základe predchádzajucého stavu[15].

$$V = (V + h + C + J) mod^{440} \text{ kde } h = H(3||V$$

$$C = C$$

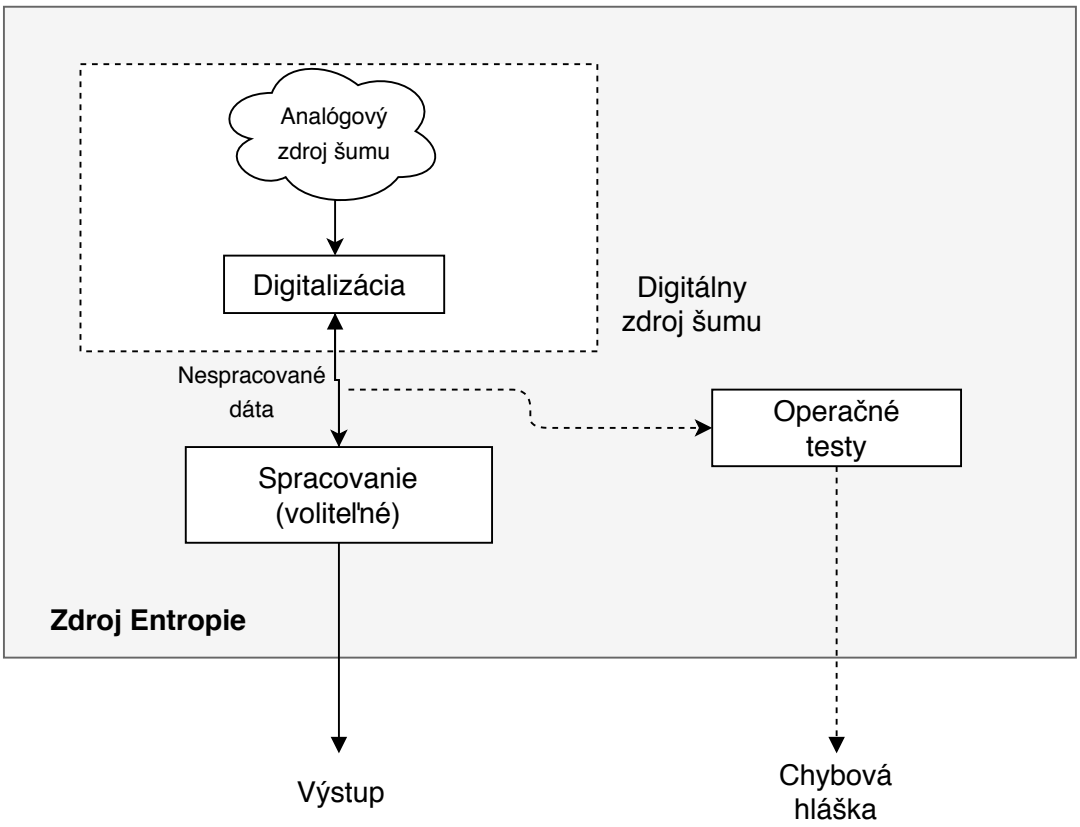
$$J = J + 1$$

2 Zdroje Entropie

Táto kapitola je zameraná na popis zdrojov entropie ktoré je možno nájsť v osobnom počítači. Táto entropia poslúži ako zdroj inicializačného semienka *(seed)* pre generátor náhodných čísel. Tieto zdroje entropie sú obvykle založené na dejoch ktoré generujú analógový šum, ten je zdigitalizovaný a ďalej sa spracúváva a testuje.

2.1 Model zdroja entropie

Táto sekcia popisuje do detailu model zdroja entropie. Na Obr. 2.2 je možné vidieť ilustráciu zdroja spolu s komponentami ktoré obsahuje. Pozostáva zo zdroja šumu voliteľného komponenta na spracovanie a komponentu na operačné testy.



Obr. 2.1: Model zdroja entropie [21].

2.1.1 Zdroj šumu

Zdroj šumu je koreňom bezpečnosti pre zdroj entropie a pre generátor náhodných čísel ako celku. Je to súčasť ktorá obsahuje nedeterministický entropiu poskytujúci

proces ktorý je zodpovedný za náhodnosť spojenú s reťazcom na výstupe zdroje entropie.

V prípade že nedeterministická aktivita ktorá je vzorkovaná tvorí iné ako binárne dáta vzorkovací proces zahŕňa aj digitalizačný proces ktorý konvertuje výstupný vzorok. Výstup z digitalizovaného zdroja šumu sú nespracované dáta.

NIST odporúčanie predpokladá že vzorky šumu získané zo zdroja pozostávajú z bitového reťazca konštantnej dĺžky. Zdroje šumu je možné rozdeliť do dvoch kategórii. Fyzické zdroje šumu ktoré využívajú dedikovaný hardware na generovanie náhodnosti alebo nefyzické zdroje šumu ktoré využívajú výstup systémových dát. Sú to napríklad výstupi API (Application programming interface), dáta z pamäte RAM, systémový čas alebo vstupné dáta od užívateľa (pohyb myši).

2.1.2 Operačné testy

Operačné testy sú integrálnou súčasťou návrhu zdroja entropie a slúžia k overovaniu že zdroj šumu a celý zdroj entropie pracuje tak ako sa od neho očakáva. Pri testovaní zdroja entropie je cieľom zaistiť že zlyhanie zdroja entropie sú zachytené rýchlo a s vysokou spolahlivosťou. Ďalším aspektom operačného testovania je odhad ktorá časť zdroja entropie môže s najväčšou pravdepodobnosťou zlyhať pre konkrétny zdroj entropie. Operačné testy by mali zahŕnať tieto testy na overenie týchto podmienok. Viac v kapitole 3.

2.2 Príklady zdrojov entropie

2.2.1 Klávesnica a myš

Pri zachytávaní pohybu myši je možné snímať súradnice v danom časovom intervale. Tieto informácie je takmer nemožné zopakovať rovnako (pri pohybe myši) pri stisku kláves môže užívateľ úmyselne zadať rovnaký reťazec čo môže kompromitovať bezpečnosť avšak pri kombinácii viacerých vstupov toto riziko zaniká. Nevýhodou generácie entropie pomocou

2.2.2 Sieťová karta

Vhodnou možnosťou na získanie entropie sú pakety na vstupe sieťovej karty. Pri zachytení paketov zo sieťovej karty je možné je možné získať dáta z tela paketu, kombináciu IP adresies a ďalších dát. Výhodou použitia entropie je nezávislosť na zásahu alebo zadania vstupných informácii od užívateľa.

2.2.3 Teplota CPU, GPU, HDD

Ďalšia kategória zdrojov entropie je teplota jednotlivých komponentov osobného počítača. Výsledky nebudú úplne náhodné avšak za desatinnou čiarkou je priestor pre nepredvídateľné vplyvy ako šum keďže meranie je analógové a podľa presnosti merania teploty sa môže využiť hodnoty za desatinnou čiarkou. S teplotou súvisia aj otáčky ventilátoru. Ak tieto nie sú nastavené na pevno. Otáčky sa zaokrúhlujú na desiatky takže sa využívajú desiatky keďže jednotky stoviek nie sú dostatočne premenlivé.

2.2.4 RAM

Pri využití fyzickej RAM pamäte je možné získať dobré hodnoty entropie v prípade že nie je nastavené pevné stankovanie ako napríklad v operačnom systéme linux. V tom prípade je alokácia pamäte predvídateľná do takej miery že je možné považovať ju za determnistickú.

2.2.5 Dátum a čas

Čas v systémoch je meraný s presnosťou na stovky nanosekúnd takže je tu veľký priestor pre získavanie náhodných čísel s tým že túto postupnosť nie je možné zopakovať keďže čas beží len dopredu. Útočník môže poznať sekundy ale presné hodnoty nanosekúnd nemá ako zistiť. Na druhú stranu je v prípade použitia času podstatná periodicita a teda ľahká odvoditeľnosť.

2.2.6 Zmeny obrazu

Ďalšou možnosťou ako získať dobré hodnoty entropie je sledovanie vizuálnych zmien obrazu napríklad na pixeloch monitora. Príkladom zaujímavej implementácie je takzvaná "Wall of randomness"spoločnosti CloudFlare. Fyzikálne deje prebiehajúce v lávovej lampe sú z podstaty náhodné a tieto sú zachytávané kamerou a na základe nich sa generujú náhodné čísla.

Obr. 2.2: Fotka steny s lávovými lampami ktoré slúžia ako zdroj entropie[7].

2.3 Entropia v systéme Windows - CryptoAPI

Na rozdiel od operačný systémov BSD a Linux či MacOS nie je priamo prístupný generátor náhodných čísel.Implementácia zdrojov entropie v operačnom systéme Windows využíva kryptografické rozhranie Cryptography Next Generation API. Toto API poskytuje sadu rozhraní s najviac používanými kryptografickými funkciami. Funkcia pre generovanie náhodných čísel je `CryptGenRandom`. Pomocou tejto funkcie je možné generovať priamo kryptografické kľúče do zásobníka ako náhodné dáta[8]. Ďalšou funkciou je `CryptGenKey` ktorá generuje kľúče pre špecifické kryptografické algoritmy obe však vnútorne používajú rovnaký generátor náhodných čísel.

Crypto API umožňuje vybrať si CSP (Cryptographic Service Provider) čo je softwarová knižnica implementujúca funkcie CryptoAPI. Niektoré knižnice sú súčasťou systému Windows iné môžu byť importované či vytvorené užívateľom. Mnou vytvorený generátor popísaný v kapitole 6.1 využíva Microsoft Strong Cryptografy Provider. Táto knižnica je v základe súčasťou každej inštalácie Windows a v súčastnosti poskytuje dostatočne silnú kryptografiu.

```
1 BOOL WINAPI CryptGenRandom(
2 HCRYPTPROV hProv ,
3 DWORD dwLen ,
4 BYTE* pbBuffer
5 );
```

Výpis 2.1: CryptGenRandom API [8]

Ako je naznačené v pseudokóde `CryptGenRandom` pracuje s troma parametrami: vý-

stup zásobníka, dĺžka a prístup k CSP. Na zavolanie týchto funkcií tak ako je naznačené v príklade kódu C++ je potrebné aby mal užívateľ nainštalované Windows SDK ktoré sú dostupné na stránkach spoločnosti Microsoft.

```
#include <windows.h>
#include <wincrypt.h>
#define SIZE 160
void main() {
            HCRYPTPROV hProv = 0;
            BYTE data[SIZE];
            CryptAcquireContext(&hProv,NULL,NULL,PROV_RSA_FULL , 0);
            CryptGenRandom(hProv , SIZE , data);
            }
```

Výpis 2.2: Príklad využitia CryptGenRandom [8]

2.3.1 Vnútorná štruktúra Windows RNG

Náhodný generátor operačného systému Windows je rozdelený do troch častí:

Hlavná slučka

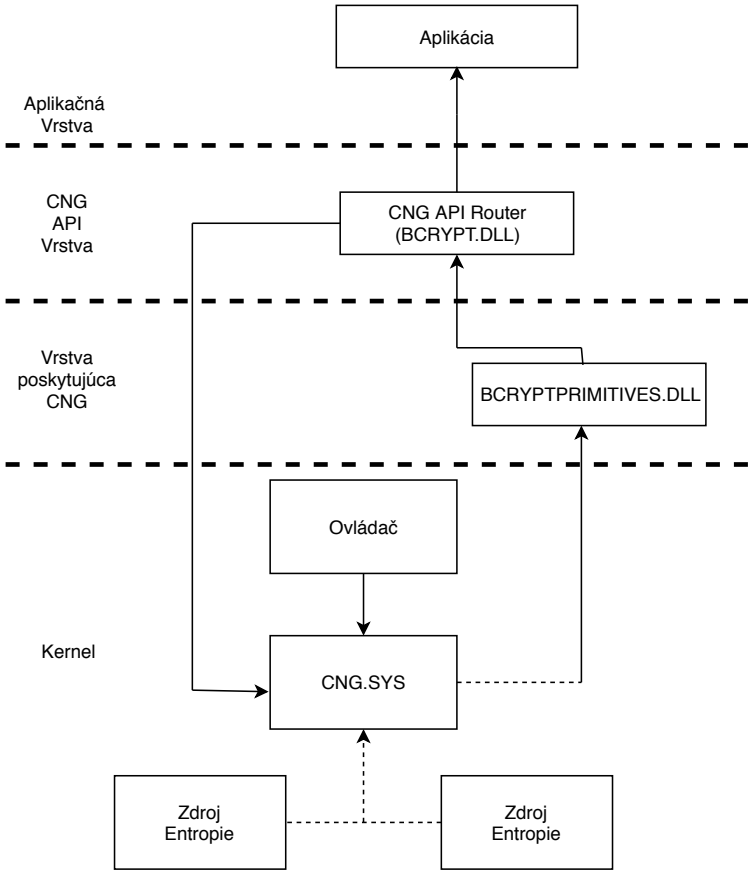
Náhodný generátor operačného systému Windows hešuje časť jeho stavu algoritmom SHA1 ktorý vyprodukuje 20 bajtov na výstupe v slučke. Aktualizuje svoj stav použitím matematických operácii a kombináciou výstupu z interného generátora.

Interný generátor

Vnútorný generátor používa niekoľko inštancií prúdovej šifry RC4. Po inicializácii alebo po generácii preddefinovaného počtu výstupu vyvolá spojenie s vnútorným zberačom entropie na vytvorenie nových RC4 inštancií.

Zberač entropie

Zberač entropie zbiera niekoľko tisícok bajtov entropie zo systémových dát, na ne potom aplikuje kryptografické transformácie na zamiešanie a použije výsledok ako vstup pre RC4 šifru vnútorného generátora [8].



Obr. 2.3: Implementácia zdrojov entropie OS Windows[20].

2.4 Entropia v systéme Linux - /dev/random

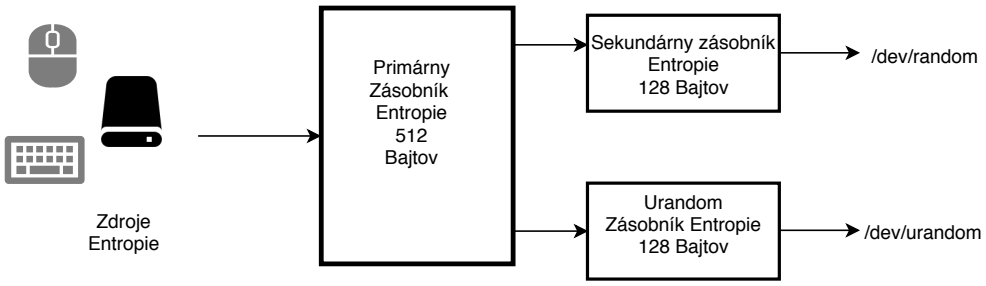
Generovanie náhodných čísel v operačnom systéme linux pozostáva z troch asynchrónnych procedúr. Počas prvej procedúry sa zozbiera entropia z operačného systému na základe viacerých dejov prebiehajúcich v jadre operačného systému. V druhom kroku sa entropia pridá do zásobníka kde sa upravuje hešovacou funkciou. Tretia procedúra predstavuje vyžiadanie náhodných čísel a následnú aktualizáciu zásobníka. Jediná nelineárna funkcia použitá týmito procedúrami je hešovacia funkcia SHA-1.

Zásobník a počítadlá

Na Obr.2.4 je znázornený schematický priebeh generácie náhodných bitov. Vnútry generátora sú 3 zásobníky : primárny, sekundárny a urandom ktorých veľkosti sú 512 a 128 bajtov. Výstupy zo zdrojov entropie sú pridávané do primárneho zásobníka, výstup zo primárneho zásobníka je extrahovaný a pridaný do sekundárneho a urandom zásobníka. Počas extrakcie zo zásobníka je vnútorný stav modifikovaný na základe spätnej väzby.

Každý zo zásobníkov má svoje vlastné počítadlo entropie. Toto počítadlo nadobúda celočíselnú hodnotu medzi 0 a veľkosťou zásobníka v bitoch čo indikuje súčasnú hodnotu entropie v zásobníku. Keď sa vyextrahuje entropia na výstup počítadlo zníži svoju hodnotu na naopak keď sa získa entropia zo zdroja, hodnota počítadla sa inkrementuje. Počítadlo sa vždy zníži o počett extrahovaných bitov. Inkrementovanie je žložitejšie. V prípade že sa bit pridáva z jedného zo zdrojov potom sa vyráta odhadne hodnota entropie a na základne nej sa inkrementuje počítadlo. Odhad hodnoty entropie využíva časovanie posledných udalostí toho istého zdroja entropie. Ak sa bity prevádzajú z primárneho zdroja entropie, počítadlo sa inkrementuje na základe počtu prenesených bitov.

Počítadlo má zásadnú úlohu keď sa získava entropia blokovacím rozhraním `/random/dev` jeho úlohou je rozhodnúť či je jej v zásobníku dostatok na dodanie požadovaných náhodných dát. V prípade že je odpoveď negatívna pokúsy sa preniesť bity z primárneho zásobníka do sekundárneho a ak sa ani to nepodarí zablokuje zdroj a vyčká pokiaľ sa nezíska dostatočný výstup zo zdroja entropie do zásobníka a tým sa nezvýši hodnota počítadla.



Obr. 2.4: Schéma generátora v operačnom systéme linux [9].

Pridanie fyzickej entropie

Bity entropie sa pridávajú do primárneho zásobníka z externých zdrojov. Tieto zdroje sú rôzne a môžu sa líšiť pre stolné stanice a servere. Medzi hlavné zdroje patria myš a klávesnica, vstupy a výstupy na disku, špecifické prerušenia. Vždy keď nastane požadovaný jav vygeneruje sa 32bitové slovo ktoré reprezentuje jeho časovanie a 32 bitové slovo ktoré kóduje jeho atribúty(napríklad ktorá klávesnica bola stlačená). K tomu sa pridáva rozdiel medzi časovaním po sebe idúcich udalostí rovnakého typu na odhad koľko entropie bolo poskytnuté konkrétnym výskytom.

Vzhľadom na asynchrónnosť tohoto systému zaznamenaná entropia sa nedá jednoducho pridať do zásobníka miesto toho je zbieraná a dávkovaná. Niekoľkokrát za minútu je vždy nová dávka dát pridaná do primárneho zásobníka. V prípade že je zásobník plný tj. jeho hodnota dosiahla 4096 entropia sa začne pridávať do

sekundárneho zásobníka. Po naplnení sekundárneho zásobníka sa proces vráti späť a tak ďalej. Entropia sa nikdy nepridáva do urandom zásobníka. Proces pridávanie entropie zvyšuje hodnotu počítadla daného zásobníka pre odhad hodnoty entropie v ňom.

Generovanie výstupu

Náhodné bity sa získavajú z jedného z troch zásobníkov. Získavajú sa keď používateľ použije `/dev/urandom` a keď kernel zavolá `get_random_bytes`, zo sekundárneho zásobníka keď používateľ použije `/dev/random` a z primárneho zásobníka v prípade že ostatné dva zásobníky nemajú dostatočný počet entropie a potrebujú naplnenie. Proces získavania entropie zahŕňa tri kroky: aktualizácia obsahu zásobníka, extrahovanie náhodných bitov na výstup a zníženie počítadlo entropie zásobníka. Tento proces zahŕňa ďalej hešovanie obsahu zásobníka použitím SHA-1 hešovacej funkcie a pridanie výsledkov do zásobníka[9].

2.5 Známe útoky na RNG

2.5.1 Netscape SSL

V roku 1996 bol demonštrovaný útok na RNG na SSL protokol prehliadača Netscape. SSL je široko akceptovaný sieťový šifrovací protokol. SSL malo chrániť privátnu komunikáciu od odpočúvania a impersonáciou - šifrovanie hesiel a čísel kreditných karriet. Netscape nikdy nezverejnil špecifikáciou implementácie SSL ale útočníkom sa podarilo reverzným inžinierstvom získať kľúče.

Netscape využíval vlastný RNG na generovanie šifrovacích kľúčov. Útočníci preskúmali PRNG algoritmus a zistili že semeno používa veľmi nízky počet slabých parametrov na generovanie. V najlepšom prípade boli ako zdroje entropie použité systémový čas, identifikátor procesu prehliadače a identifikátor rodičovského procesu prehliadača. Hoci vstupy boli zašifrované pomerne silnou hešovacou funkciou (MD5) útočníkom sa podarilo uhádnuť originálne hodnoty hrubou silou [11].

2.5.2 Kerberos 4.0

Podobný útok ako na SSL implementáciu v Netscape bol demonštrovaný na MIT implementáciu protokolu Kerberos verzie 4.0. Generácia kľúčov pre spojenie používané ako dôkaz identity a autentizáciu správ využívalo PRNG ktorého semeno využívalo parametre ako čas, ID procesu , ID hosta , počítadlá a podobne. Táto implementácia nemala dostatočnú entropiu a preto bolo možné kľúče uhádnuť hrubou silou a hádaním podobne ako v predchádzajúcom prípade[12].

2.5.3 Virtuálny stroj JAVA

Ďalším z možných útokov preukázala analýza RNG využívaného serverovým appletom Tomcat využívajúci virtuálny stroj JAVA. HTTP je bezostavové spojenie mnoho webových stránoch používa mechanizmy ako cookies alebo prepisovanie URL na udržanie stavu relácie na užívateľovej strane.

Stavové relácie sa používajú na udržiavanie a zisťovanie stavu napríklad nákupného košíka, užívateľových preferencíi, predchádzajúcich transakcíi a podobne.

Aby sa zabránilo únosu relácie server generuje token ID relácie ktorý je reprezentovaný veľkým náhodnym číslom a toto je zakódované pomocou cookie v URL prehliadača. Vychádza sa z predpokladu že útočník by len ťažko dokázal uhádnuť URL token kvôli veľkému náhodnému číslu. Toto však platí len v prípade že RNG ktorý generuje token je dostatočne silný a teda používa vhodné zdroje entropie.

Literatúra uvádza príklad kde útočníci demonštrovali únos relácie použitej Tomcat serverom na generovanie ID tokenov. Programovací jazyk JAVA poskytujen dve API pre generovanie náhodných čísiel: `java.util.Random` ktorý je LCG a `java.security.Secure.Random` ktorý je silnejší PRNG so 160bitovým zásobníkom a SHA-1 funkciou.Útok je založený na tom že oba generátory využívajú ako semeno veľmi málo zdrojov entropie ako napríklad systémový čas v milisekundách a heš objektu v pamäti ktorý dosahujú iba niekoľko hodnôt. Pri skúšaní všetkých možných kombinácii hrubou silou a porovnávaní výsledkov boli útočníci schopný uniesť reláciu prehliadača a tak získať napríklad ich transakcie. Autori odhadujú že vzhľadom na veľkosť vyhladávacieho priestoru ktorý je 2^{42} je útok možné vykonať na osobnom počítači.[13]

2.5.4 Linux RNG

Popis linuxového generátoru bol uvedený v podkapitole 2.4. Literatúra uvádza že generátor náhodných čísel využívajúci `/dev/random` a `/dev/urandom` mal problém s počiatočnou inicializáciou semena. Autori demonštrovali útok ktorý spočíval v odopretí služby kde sa exesívne čítal výstup zo zdroja entropie a tým sa spôsobilo pernamentné zablokovanie zdroja.[9]

3 Operačné testy

Operačné testy *(Health Tests)* sú dôležitou súčasťou testovania zdroja entropie. Ich cieľom je detekovať odchýlky a abnormálnosti od očakávaného chovania zdroja šumu čo najrýchlejšie ako a podobná udalosť vyskytne s vysokou spolahlivosťou. Zdroje šumu sú citlivé na zmeny podmienok v ktorých pracujú a teda zmena veličín ako teplota, vlhkosť či zmena v elektrickom poli môže ovplyvniť výstup a spôsobiť zmeny v chovaní zdroja.Operačné testy použijú entropiu ako vstup a charakterizujú očakávané chovanie zdroja šumu na základe tejto hodnoty.

Operačné testovanie zdroja šumu bude závislé na použitej technológii zdroja. Keďže vo väčšine prípadov zdroje šumu nebudú produkovať nezávislé binárne dáta neovplyvnené vonkajšími vplyvmi procedúry popísané v NIST SP 800-22 takmer vždy zlyhajú a nedajú sa použiť pre monitorovanie zdroja šumu. Operačné testy musia byť starostlivo navrhnuté tak aby brali do úvahy očakávané štatistické chovanie korektne pracujúceho zdroja šumu. Operačné testovanie zdroja entropie bude typicky navrhnuté tak aby zachytilo zlyhania zdroja šumu založené na výstupe očakávaného výstupu počas zlyhania alebo na zachytenie odchýľku na výstupe počas korektnej operácie zdroja šumu. Operačné testy v troch prípadoch:

1. V prípade že je značný pokles entropie na výstupe zdroja.
2. Keď zlyhá zdroj šumu.
3. V prípade hardwérovej poruchy následkom ktorej prestane implementácia fungovať korektne.

Operačné testy sa aplikujú na výstupy zdrojov šumu predtým než sa vykonaním akýchkoľvek úprav.

3.1 Typy operačných testov

3.1.1 Operačné testy pri štarte

Testy pri štarte (*Start-up Health Tests*) sú navrhnuté tak že sa vykonajú vždy po zapnutí alebo reštarte zarariadenia a pred prvým použitím zdroja entropie. Poskytujú základné uistenie že komponenty zdroja entropie sú funkčné a budú fungovať tak ako sa od nich očakáva pred ich skutočným použitím. Tak isto poskytujú informáciu o tom že od posledného použitia zdroja entropie nedošlo k žiadnemu zlyhaniu. Vzorky ktoré sú použité počas testov pri štarte nesmú byť získané počas normálneho prevozu. Tieto vzorky sa môžu zahodiť alebo následne použiť za predpokladu že počas testov sa nevyskytli žiadne chyby a zlyhania.

3.1.2 Kontinuálne operačné testy

Kontinuálne operačné testy (*Continuos Health Tests*) sú spustené v nekonečnej slučke na výstupe zdroja šumu počas aktívneho prevozu zdroja. Kontinuálne testy sa sústreďujú na správanie zdroja entropie s cieľom zachytiť anomálie na výstupe zdroja šumu. Zmysel týchto testov je umožniť zdroju entropie identifikovať akékoľvek možné zlyhanie v jeho zdroji šumu. Tieto testy bežia kontinuálne na všetkých digitalizovaných vzorkoch získaných zo zdroja šumu a preto musia mať veľmi nízku šancu na falošne pozitívny výsledok počas normálneho používania zdroja šumu. V mnohých systémoch preto pri počítaní s možnosťou falošne pozitívneho výsledku môže ovplyvniť šancu na detekovania chyby počas celého životné cyklu zariadenia. Kontinuálne testy pracujú s obmedzenými zdrojami čo limituje ich schopnosť detekovať chyby zdroja šumu, sú navrhnuté tak aby zachytili iba veľké a vážne zlyhania.

Kontinuálne testy operujú nad prúdom hodnôt. Tieto hodnoty môžu byť výstup zo samotného zdroja entropie po spracovaní, nie je potrebné blokovať hodnoty zo zdroja šumu alebo zdroja entropie počas chodu testu. Je dôležité počítať s možnosťou že na to môže spôsobiť nízke hodnoty entropie pretože chyba je signalizovaná až po určitom čase keď sa nahromadí dostatočný dôkaz o chybe. Tieto hodnoty už mohli byť predtým doručené na výstup zdroja entropie. Na druhej strane je ale potrebné nastaviť hranicu ktorá sa považuje za falošne pozitívny výsledok na akceptovateľnú úroveň. Hranica stanovená odporúčaním je že jedna chyba v 2^{20} vzorkoch generovaných zdrojom šumu je akceptovateľná avšak vzorce na stanovanie akceptovateľnej hranice sú rôzne a môžu sa prispôsobiť v závislosti na požiadavkach. [16].

3.1.3 Operačné testy na vyžiadanie

Operačné testy na vyžiadanie (*On-demand Health Tests*) podľa odporúčania NIST nie sú vyžadované počas operácie avšak je vyžadované aby bolo možné vykonať test na výstupe zdroja šumu. Za test výstupu šumu je možné považovať test pri štarte v prípade ak sa vykoná reštart zariadenia.

3.2 Požiadavky na operačné testy

Operačné testy sú požadovanou súčasťou zdroja entropie. Operačné testy musia obsahovať testy pri štarte a kontinuálne testy.

- Kontinuálne testy by mali obsahovať aspoň jedno z nasledovných:
  - Testy počtu opakovaní alebo Adaptívny a proporčné testy
  - Test definovaný vývojárom za predpokladu že spĺňa rovnaké podmienky ako testy predchádzajúceho bodu.

  - Testy môžu obsahovať ďalšie testy ktoré vývojár navrhne.
- V prípade pozitívne výsledku operačného testu to znamená zachytenia chyby, zdroj entropie musí upozorniť aplikáciu ktorá ho využíva (napríklad zdroj náhodných bitov). Aplikácia dostane okno na rozhodnutie na ďalší postup v závislosti na type poruchy.
- Optimálna hodnota hranice na zachytenie falošne pozitívneho výsledku môže závisieť na hodnote ktorú zdroj entropie poskytuje na výstupe. Pre Adaptívne a proporčné testy prípadne pre Testy počtu opakovaní je hodnota stanovená medzi 2^{-20} a 2^{-40}. Nižšia tolerancia je akceptovateľná. V prípade požiadaviek na vyššiu toleranciu je potrebné zdôvodniť prečo je to potrebné pre konkrétnu aplikáciu.
- Testy pri štarte by mali prebehnúť aspoň na 1024 po sebe idúcich vzorkoch. Tieto vzorky sa môže ďalej použiť v prípade že prešli testami bez nájdenia chyby alebo sa môžu zahodiť.
- Považuje sa za dostatočné ak sa miesto testov na vyžiadanie inicializujú testy pri štarte reštartovaním zariadenia.
- Testy by sa mali vykonávať pred spracovaním spracovacím členom. Je možné vykonať ďalšie testy po spracovaní.
- Musí byť definované kedy sa zdroj šumu dostane do chybového stavu. To znamená že začne produkovať napríklad periodické výstupy ako 101...01.
- Musí byť stanovená tolerancia vzhľadom na vonkajšie podmienky a vplyvy v ktorých zdroj pracuje.

3.3 Test počtu opakovaní

Test počtu opakovaní má za úlohu zachytiť katastrofické zlyhanie pri ktorom sa zdroj šumu ostane zaseknutý na rovnakej hodnote výstupnej hodnoty po dlhý čas. Tento test má za úlohu zachytiť úplne zlyhanie zdroja šumu.

Pri predpoklade že minimálna entropia H zdroja šumu pravdepodobnosť že zdroj sa zasekne na generovaní rovnakých n identických vzorkov je najviac $2-H(n-1)$. Test oznámi chybu v prípade že vzorok sa opakuje viac ako C krát. Hodnota C je determinovaná z konštanty pre akceptovateľnú chybovosť α a odhadu entropie za použitia nasledovného vzorca:

$$C = 1 + [\frac{-log_2}{H}] \tag{3.1}$$

Hodnota C je najmenšie celé číslo spĺňajúce podmienku $\alpha \geq 2^{-H(C-1)}$, táto podmienka zaručí že získanie sekvencie rovnakých hodnôt C z po sebe idúcich vzorkov zo zdroja šumu bude maximálne rovná α. Napríklad pre hodnotu $\alpha = 2^{-20}$ bude

medzná hranica testu počtu opakovaní pre zdroj entropie s entropiou H=2.0 bitu pre vzorok rovná $1 + [20/2.0] = 11$

Postup pri získavaní vzorky zo zdroja šumu je nasledovný:

1. A=next()
2. B=1
3. X=next()
4. - If(X=A),
   - B=B+1
   - If(B $\geq C$) ohlás chybu.
   - else:
   - A=X
   - B=1
5. Opakuj od kroku 3.

Medznú hodnotu tohoto testu je možné aplikovať na akýkoľvek odhad hodnoty entropie H vrátane odhadov pre veľmi malé hodnoty ako aj veľmi veľké hodnoty. Je dôležité podotknúť že tento test zachytí iba veľmi vážne zlyhania zdroja šumu. Napríklad v prípade že zdroj šumu ktorý má 8 bitov entropie na jeden vzorok s medznou hodnotou šiestich opakovaní a toleranciou pre falošne pozitívne hodnoty 10^{12} pre generované vzorky. V prípade že by došlo k zlyhaniu zdroja šumu pričom by šanca bola 1/16 že vzorok bude rovnaký ako predchádzajúci vzorok a entropia by dosahovala hodnotu iba 4 bity na vzorok bolo by potrebné minimálne milión vzorkov na to aby test počtu opakovaní zistil problém.

3.4 Adaptívny proporčný test

Adaptívny proporčný test je navrhnutý na detekovanie veľkej straty entropie ktorá môže nastať z dôvodu fyzického zlyhania zdroja alebo silnej zmeny podmienok okolia zdroja. Test pravidelne meria lokálny výskyt zmien frekvencie hodnoty vzorku v sekvencii vzorkov na detekovanie opakovania rovnakej hodnoty vzorku. Čiže test dokáže detokovať či sa nejaká hodnota opakuje častejšie než sa predpokladalo vzhľadom na požadovanú entropiu zdroja v porovnaní s detekciou úplneho zlyhania zdroja ktoré poskytuje len test počtu opakovaní.

Test vezme vzorok zo zdroja šumu a potom zráta počet koľko krát sa rovnaká hodnota opakuje počas ďalších W-1 vzorkov. Ak tento počet opakovaní dosiahne medznú hranicu C test vyhlási chybu. Veľkosť okna W je závisí od veľkosť abecedy, v prípade že zdroj šumu je binárny tj. produkuje iba dve rozdielne hodnoty musí mať hodnotu 1024 a v prípade že zdroj je nebinárny hodnotu 512. Pre po sebe idúci

počet vzorkov šumu ak budeme uvažovať meznú hodnotu C a veľkosť okna W postup pri získavaní vzorky zo zdroja šumu je nasledovný:

1. A=next()
2. B=1
3. X=next()
4.
   - Pre i = 1 pre W-1
   - If (A = next()) B= B+1
   - If(B $\geq C$) ohlás chybu.
5. Opakuj od kroku 1.

Medzná hodnota C je vybratá tak aby pravdepodobnosť získania C alebo viacerých rovnakých hodnôt v okne veľkosti W dosiahne maximálne hodnotu α. Matematicky hodnotu C môžeme popísať nasledovnou rovnicou:

$$Pr(B \geq C) \leq \alpha \tag{3.2}$$

Pre binárne zdroje je možné rozšíriť test overovaním či W-B $\geq$ C čo zaručí že v prípade že sekvencie binárnych hodnôt opakujúce sa príliš často budú zachytené už prvým testovacím oknom.

V tabuľke 3.1 sú uvedené príklady medzných hodnôt pre rôzne odhady minimálnej entropie na vzorok a s veľkosťou okna $\alpha = 2^{-20}$ [16].

Binárne dáta W=1024		Nebinárne dáta W=512	
Entropia	Medzná hodnota C	Entropia	Medzná hodnota C
0,2	941	0,5	410
0,4	840	1	311
0,6	748	2	177
0,8	664	4	62
1	589	8	13

Tab. 3.1: Príklady medzných hodnôt ore Adaptatívny proporčný test[16]

4 Validácia zdroja entropie

V tejto kapitole bude popísané validácia zdroja entropie podľa odporúčania NIST SP-800-90B. Táto špecifikácia záhrňa požiadavky na hodnotenie zdrojov entropie okrem iného aj na komerčné účely a popisuje postup potrebený na to aby bolo možné zdroj entropie považovať za vhodný.

Validácia je nevyhnutná na uistenie že zdroj spĺňa všetky požiadavky podľa odporúčania NIST. Validácia spočíva v porovnaní testov z akreditovaného labratória NVLAP oproti požiadavkám SP-800-90B. Validácia poskytuje potvrdenie že entropia poskytovaná zdrojom je dostatočná a samotná validácia zdroja treťou stranou môže byť požiadavkou na impelentáciu podľa rôznych legislatív prípadne štandartov organizácii alebo firiem.

Validácia zdroja entropie predstavuje mnoho výziev pretože žiadna iná časť náhodného generátora náhodných čísiel nie je tak závislá od technických detailov implementácie ako zdroj entropie. Zároveň správna implementácia zdroja entropie je kľúčová pre bezpečnosť generátora náhodných čísiel. Je potrebné zaistiť aby zdroj entropie poskytoval konzistentné hodnoty bitových reťazcov ktoré sú dostatočne vysoké na to aby spĺňali alebo prekonali minimálne stanovenú hodnotu.

Pri návrhu zdroja entropie ktorý spĺňa požiadavky a teda generuje dostatočný prúd bitových reťazcov je potrebné správne odhadnúť entropiu generovanú zdrojom správnym vzorkovaním digitalizovaného šumu ktorý generuje.

4.1 Overovací proces

4.1.1 Zber dát

1. Sekvenčný súbor dát o počte aspoň 1 000 000 získaný priamo zo zdroja šumu by mal byť zachytený pre validáciu. V prípade že generácia 1 000 000 po sebe idúcich vzoriek nie je možná, generácia menších po sebe idúcich zdrojov za použitia rovnakého zdroja šumu je povolená. Menšie súbory dát by mali obsahovať aspoň 1 000 vzorkov. Zoskupený súbor dát by mal obsahovať aspoň 1 000 000 vzorkov.
2. Výstup kondicionovaného komponentu by mal byť zoradený v poradí v akom bol generovaný a musí s ním byť zaobchádzané s ako binárnym reťazcom. Za pozornosť stojí uviesť že dáta získané zo zdroja šumuna overenie sa môžu použiť ako vstup ku kondicionačnému komponentu na zber upravených výstupných hodnôt.
3. Pre testy reštartu sa musí zdroj entropie reštartovať aspoň 1000, pre každý reštart je potrebné zozbierať 1000 po sebe idúcich vzorkov priamo zo zdroja

šumu. Dáta je potrebné extrahovať vždy keď je zdroj šumu pripravený a schopný poskytnúť údaje, vďaka ktorým je možné možné vygenerovať výstup zo zdroja entropie. Tieto dáta sú uložené v matici 1000 x 1000 M,kde M reprezentuje j^tí vzorok z i^té*ho* reštartu.

4.2 Požiadavky na overovacie testy

V tejto podkapitole sú popísané požiadavky na zdroj entropie, šumu a spôsob zberu dát. Zmysel týchto požiadaviek je poskytnutie akéhosi vodítka pre návrh implementácie a návrhu zdroja entropie podľa NIST [21].

4.2.1 Požiadavky na zdroj entropie

V tejto podkapitole sú uvedené požiadavky na zdroj entropie:

1. Zdroj entropie musí mať stanovené presné operačné podmienky v ktorých je možné garantovať jeho korektné fungovanie. Medzi tieto podmienky patrí rozsah operačných teplôt,elektrického napätia, aktivita operačného systému a podobne.
2. V prípade že v zdroji entropie nie je použitý člen na spracovanie , výstupom zdroja entropie je výstup zdroja šumu a žiadne ďalšie rozhranie nie je potrebné. V tomto prípade je výstup zdroja šumu výstupom zdroja entropie.
3. V prípade že sa používa člen na spracovanie entropie, výstup zdroja šumu musí byť prístupný cez ďalšie rozhranie.
4. Prístup k zdroju entropie môže byť obmedzený počas testovania za podmienok že by tento prístup mohol ovplyvniť výsledky merania.

4.2.2 Požiadavky na zdroj šumu

Zdroj entropie nebude mať viac entropie ako poskytuje zdroj šumu. Preto je potrebné testovaniu a požiadavkám na zdroj šumu venovať zvýšenú pozornosť. Zdroj šumu má fundamentálny význam pre zdroj entropie - tj. ak nefunguje správne zdroj entropie nedosiahne dostatočné množstvo entropie.

1. Operácie zdroja šumu musia zdokumentované. V dokumentácii zdroja šumu je uvedené ako zdroj šumu pracuje, na akom princípe funguje jeho náhodnosť a uviesť prečo sa považuje za akceptovateľný zdroj pre výstup entropie.
2. Správanie zdroja šumu musí byť statické. To znamená že sa pravdepodobnosť distribúcie výstupu zo zdroja šumu nebude meniť v čase. Dokumentácia zdroja

šumu by mala uviesť prečo sa predpokladá že výstup sa radikálne nezmení v čase počas normálnej prevádzky.

3. Zdroj šumu musí mať jasne definované vyjadrenie prečo sa očakáva že sa poskytne dostatočná entropia a že zdroj šumu je schopný dodať požadovanú úroveň entropie. Za technicky jasné vyjadrenie sa považuje stochastický model výstupov zdroja šumu a odhad entropie založený na tomto stochastickom modele.
4. Zdroj šumu musí byť chránený pred útočníkom najvyššou možnou mierou. To zahŕňa možné ovplyvnenie zdroja šumu ako aj znalosti o ňom. Spôsob akým sa chráni zdroj musí byť zdokumentovaný vrátane popise koncepčnej ochrannej hranice ktorá má chrániť zdroj pred pozorovaním útočníkom alebo ovplyvňovaniu zdroja útočníkom.
5. Hoci zdroj šumu nemusí poskytovať neovplyvnené a nezávislé výsledky musí mať náhodný výstup. To znamená že výstup nesmie byť definovateľný žiadnym známym algoritmickým pravidlom. Musí byť jasne uvedené či sú produkované IID alebo non-IID dáta. Podľa tohoto sa rozhodne testovacia cesta v hodnotení entropie. V prípade že sa má jednať o IID testy je potrebné toto tvrdenie podložiť odôvodnením resp. dôkazom tohoto tvrdenia.
6. Zdroj šumu musí generovať bitový reťazec s pevnou dĺžkou. Dokumentácia zdroja šumu musí uviesť dĺžku symbolu v bitoch a zoznam alebo rozsah možných výstupov pre každý zdroj šumu.
7. Je možné použiť ďalšie zdroje šumu na zvýšenie bezpečnosti zdroja entropie. Popis implementácie týchto zdrojov musí byť uvedený v dokumentácii [21].

4.2.3 Požiadavky na zber dát

V nasledujúcej sekcii sú uvedené požiadavky pri zbere dát.

1. Dáta ktoré sa budú testovať by mali byť čisté dáta bez úpravy (raw data).
2. Zber dát musí prebiehať za normálnych operačných podmienok.
3. Dáta musia byť zachytené zo zdroja entropie ktorý je považovaný za validný. O validite dát rozhoduje či boli predtým vykonané predtým spomenuté testy s dobrým výsledkom.
4. Každá hardwerová alebo aktualizácia softwérovu musí byť zaznamená v dokumentácii alebo uvedená ako poznámka k zachyteným dátam [21].

4.3 Nezávislé a identicky distribuované zdroje šumu

Vzorky zo zdroja šumu sú nezávislé a identicky distribuované *(independent and identically distributed - IID)* v prípade že každý vzorok má rovnakú šancu na výskyt ako iné vzorky a zároveň sú všetky vzorky na sebe nezávislé. V prípade že vzorky nie sú identicky distribuované alebo nie sú nezávislé distribuované alebo zároveň nespĺňajú obe podmienky hodnotenie entropie je náročnejšie a vyžaduje si viacero rozličných metód.

V tejto podkapitole budú uvedený popis štatistických testov pomocou ktorých je možné nájsť dôkaz že vzorok nie je IID. V prípade že sa nenájde dôkaz že vzorok nie je IID považuje sa za IID.

V týchto testov sa bude uvažovať sekvencia dát S kde $S = (S_1, ...SL)$ pričom $S_i \in A = X_1, ...X_k$ ako vstup a testovacia hypotéza predpokladá že hodnoty S sú IID. V prípade že je testovacia hypotéza vyvrátená ktorýmkoľvek testom hodnoty S sú považované za neIID. Štatistické testy založené na permutačnom testovaní sú popísané v nasledovnej podkapitole a v podkapitole 4.3.2 sú uvedené dodatočné Chí kvadrát testy.

4.3.1 Permutačné testovanie

Permutačné testovanie je spôsob ako testovať štatistickú hypotézu v ktorej je skutočná hodnota testovaného vzorku porovnaná k referenčnej distribúcii ktorá je odvodená zo vstupných dát.

Vo všeobecnosti postup pri permutačnom testovaní spočíva v generovaní 10 000 permutácii dátovej sady. Vyrátaním testovej štatistiky pre každú permutáciu a porovnanie výsledka so štatistikou na originálnej dátovej sade. Toto sa zopakuje pre každý ďalší test uvedený v nasledovnom texte.

Všeobecná štruktúra permutačného testu:

Vstup:$S = (S_1, ...S_L)$

1. Pre každý test i
   1.1. Nastavenie počítadla $C_{i,0}$ a $C_{i,1}$ na nulu
   1.2. Vypočítaj štatistický test T_i na dátovom vzorku S
2. pre j= 1 až 10 000
   2.1. Vykonaj permutácie na dátovom vzorku S za použitia Fisher-Yatesovho algoritmu.
   2.2. Pre každý test i
      i. Vypočítaj štatistiku T na permutovaných dátach.
      ii. Ak $(T > T_i)$ inkrementuj $C_{i,0}$. Ak $(T = T_i)$ inkrementuj $C_{i,1}$

3. Ak (($C_{i,0}$+$C_{i,1} \leq 5$) alebo ($C_{i,0} \geq 9995$)) pre akékoľvek i odmietni predpoklad IID inak predpokladaj že výstup zdroja šumu je IID.

Výstup: Rozhodnutie o IID alebo neIID

V prípade že vzorky sú IID permutácie datasetu by nemali spôsobiť dramaticky veľké zmeny v testovanej štatistike. V zásade originálny súbor dát a permutovaný súbor dát je získaný z rovnakej distibúcie a teda testovaná štatistika by mala byť podobná. V prípade že testované dátové súbor nie sú IID potom originálne a permutované testové štatistiky budú mať značné rozdiely. Počítadlá $C_{i,0}$ a $C_{i,1}$ sa používajú na určenie rozdielu medzi pôvodným dátovým súborom a permutovaným dátovým súborom. Extrémne hodnoty počítadiel sú znakom toho že dáta nie sú IID. Ak je súčet počítadiel $C_{i,0}$ a $C_{i,1}$ viac ako 5 znamená to že testy originálneho súboru dát majú vysokú hodnotu. Naopak ak je $C_{i,0}$ väčšie ako 9995 znamená to že testty pôvodných hodnôt mali veľmi nízku hodnotu.

Fisher-Yattesov algoritmus sa dá popísať nasledovne:

Vstup:$S = (S_1, ...S_L)$

1. pre i z L až po 1 vykonaj:
   1.1. Vygeneruj náhodné celé číslo j ktoré spĺňa $1 \leq j \leq i$
   1.2. Vymeň S_j a S_i

Výstup: Premiešané $S = (S_1, ...S_L)$

Testy je možné aplikovať na binárne ako aj nebinárne dáta. Avšak pre niektoré z testov počet konkrétnych hodnôt vzorkov značne ovplyvňuje distribúciu testovanej štatistiky a teda možnosť chyby. Pre tieto testy sa používa jedna z nasledujúcih dvoch konverzíi v prípade že vstup je binárny napríklad k=2.

- *Konverzia I* spočíva v rozdelení sekvencie do osembitových neprelínajúcich sa blokov a v zrátaní čísel v každom bloku. Nuly sa pridajú na koniec v prípade že posledný blok má menej ako 8 bitov. Napríklad 20bitový vstup (1,0,0,0,1,1,1,0,1,1,0,1,1,0,1,1,0,0,1,1). Prvý a posledný 8bitový blok obsahuje štyri a šesť jednotiek. Posledný blok ktorý nie je kompletný obsahuje dve jednotky. Výstupná sekvencia je (4, 6, 2).
- *Konverzia II* spočíva v rozdelení sekvencie do osembitových neprelínajúcich sa blokov a vo vypočítaní celočíselnej hodnoty každého bloku. Napríklad pri vstupe máme hodnotu (1,0,0,0,1,1,1,0,1,1,0,1,1,0,1,1,0,0,1,1) potom celočíselná hodnota prvých dvoch blokov je 142 a 219. Nuly sa pridávajú k poslednému bloku v prípade že má menej ako 8bitov. Potom sa z poslednej sekvencie stáva (0,0,1,1,0,0,0,0) s celočíselnou hodnotou 48. Výstupná sekvencia má hodnotu (142, 219, 48).

Exkurzný štatistický test

Exkurzný štatistický test *(Excursion Test Statistic)* meria ako moc sa súčet hodnoty vzorkov odlišuje od primernej hodnoty vzorku v každom bode súboru dát. Ak máme $S = (S_1, ...S_L)$ a testová štatistika T predstavuje najväčšiu odchýlku od priemeru vyrátame ju nasledovne:

1. Vypočítanie priemernej hodnoty vzorku: $\bar{X} = (S_1 + S_1 + S_L)/L$
2. Pre i=1 do L
   Vypočítanie $d_i = |\sum_{j=1}^{i} S_j - \times \bar{X}|$
3. T=max $(d_1, ...d_L)$

Príklad: V prípade že je vstupný set dát S = (2, 15, 4, 10, 9)priemerná hodnota vzorku je 8 z čoho vyplýva: $d_1 = |2 - 8| = 6; d_2 = |(2 + 15) - (2 \times 8)| = 1; d_3 = |(2 + 15 + 4) - (3 \times 8)| = 3; d_4 = |(2 + 15 + 4 + 10) - (4 \times 8)| = 1$; a $d_5 = |(2 + 15 + 4 + 10 + 9) - (5 \times 8)| = 0$. Potom T = max(6, 1, 3, 1, 0) = 6.
Test je možné aplikovať na binárne dáta bez použitia korekcie.

Počet cirkulárnej štatistiky

Test počtu cirkulárnej štatistiky *(Number of Directional Runs)* určuje počet behov zostavených použitím vzťahu po sebe idúcich vzorkov. Ak máme $S = (S_1, ...S_L)$ T je vypočítané nasledovne:

1. Zostav sekvenciu $S' = (S'_1, ...S'_{L-1})$

$$S'_i = \begin{cases} -1, & \text{ak} \quad S_i > S_{i+1}. \\ +1, & \text{ak} \quad S_i \leq S_{i+1}. \end{cases} \tag{4.1}$$

   pre i=1,...,L-1
2. Výsledok T je počet behov S'.

Príklad:V prípade že začiatočná sekvencia S = (2, 2, 2, 5, 7, 7, 9, 3, 1, 4, 4); potom S'= (+1, +1, +1, +1, +1, +1, -1, -1, +1, +1). Sú tu tri behy: (+1, +1, +1, +1, +1, +1), (-1, -1) a (+1, +1), takže T = 3.
Na testovanie binárnych dát je potrebné uplatiť prvú korekciu zmienenú v predchádzajúcom texte tejto podkapitoly.

Dĺžka cirkulárnej štatistiky

Testy dĺžky cirkulárnej štatistiky *(Length of Directional Runs)*vyrátajú najdlhší beh použitím vzťahov medzi po sebe idúcimi vzorkami.Ak máme $S = (S_1, ...S_L)$ T je

vypočítané nasledovne:

1. Zostav sekvenciu $S' = (S'_1, ...S'_{L-1})$

$$S'_i = \begin{cases} -1, & \text{ak} \quad S_i > S_{i+1}. \\ +1, & \text{ak} \quad S_i \leq S_{i+1}. \end{cases} \tag{4.2}$$

   pre i=1,...,L-1
2. Výsledok T je najdlhší beh v množine S'.

Príklad:V prípade že začiatočná sekvencia S = (2, 2, 2, 5, 7, 7, 9, 3, 1, 4, 4); potom S'= (+1, +1, +1, +1, +1, +1, -1, -1, +1, +1). Sú tu tri behy: (+1, +1, +1, +1, +1, +1), (-1, -1) a (+1, +1), takže T = 6.
Na testovanie binárnych dát je potrebné uplatiť prvú korekciu zmienenú v predchádzajúcom texte tejto podkapitoly.

Počet zvýšení a znížení hodnoty

Test zameraný na štatistické meranie maximálneho počtu zvýšení a znížení hodnoty v za sebou idúcich vzorkoch*(Number of Increases and Decreases)*. Ak máme $S = (S_1, ...S_L)$ T je vypočítané nasledovne:

1. Zostav sekvenciu $S' = (S'_1, ...S'_{L-1})$

$$S'_i = \begin{cases} -1, & \text{ak} \quad S_i > S_{i+1}. \\ +1, & \text{ak} \quad S_i \leq S_{i+1}. \end{cases} \tag{4.3}$$

   pre i=1,...,L-1
2. Je potrebné vyrátať počet -1 a +1 v množine S' čo znamená že T bude maximálny počet výskytu týchto hodnôt.

Príklad:V prípade že začiatočná sekvencia S = (2, 2, 2, 5, 7, 7, 9, 3, 1, 4, 4); potom S'= (+1, +1, +1, +1, +1, +1, -1, -1, +1, +1). Je tu osem +1 a dve -1 v množine S'takže z toho vyplýva že T=max (počet +1 a -1) =max(8,2)=8. Na testovanie binárnych dát je potrebné uplatiť prvú korekciu zmienenú v predchádzajúcom texte tejto podkapitoly.

Počet behov založený na mediáne

Tento štatistický test založený na počte behov v závisloti na mediáne vstupných dát.*(Number of Runs Based on the Median)*.Ak máme $S = (S_1, ...S_L)$ T je vypočítané nasledovne:

1. Nájdenie mediánu $\bar{X}$ zo súboru $S = (S_1, ...S_L)$

2. Zostavenie sekvencie $S' = (S'_1, ...S'_1)$ kde

$$S'_i = \begin{cases} -1, & \text{ak} \quad S_i > \bar{X}. \\ +1, & \text{ak} \quad S_i \leq \bar{X}. \end{cases} \tag{4.4}$$

   pre i=1,...,L-1
3. T je počet behov v množine S'.

Príklad:V prípade že začiatočná sekvencia S = (5, 15, 12, 1, 13, 9, 4); Medián je 9. Potom S'= (-1, +1, +1, -1, +1, +1, -1). Behy sú (-1), (+1, +1), (-1), (+1, +1), a (-1). Máme tu 5 behov a teda T=5. Na testovanie binárnych dát nie je potrebná žiadna korekcia keďže hodnota mediánu týchto dát je uvažovaná ako 0.5.

Dĺžka behu založená na mediáne

Tento štatistický test *(Length of Runs Based on Median)* sleduje dĺžku najdlhšieho behu v závislosti od mediánu vstupných dát. Ak máme $S = (S_1, ...S_L)$ T je vypočítané nasledovne:

1. Nájdenie mediánu $\bar{X}$ zo súboru $S = (S_1, ...S_L)$

2. Zostavenie sekvencie $S' = (S'_1, ...S'_1)$ kde

$$S'_i = \begin{cases} -1, & \text{ak} \quad S_i > \bar{X}. \\ +1, & \text{ak} \quad S_i \leq \bar{X}. \end{cases} \tag{4.5}$$

   pre i=1,...,L-1
3. T je dĺžka najdlhšieho behu množine S'.

Príklad:V prípade že začiatočná sekvencia S = (5, 15, 12, 1, 13, 9, 4); Medián je 9. Potom S'= (-1, +1, +1, -1, +1, +1, -1). Behy sú (-1), (+1, +1), (-1), (+1, +1), a (-1). Najdlhší beh má dĺžku 2 a teda T=2. Na testovanie binárnych dát nie je potrebná žiadna korekcia keďže hodnota mediánu týchto dát je uvažovaná ako 0.5.

Test priemerného počtu kolízii

Test priemerného počtu kolízii *(Average Collision Test Statistic)* je test ktorý počíta počet po sebe idúcich vzorkov kým sa nevyskytne duplikát. Test funguje nasledovne:

1. Majme pole C. C predstavuje počet vzorkov s rovnakou hodnotou v súbore hodnôt $S = (S_1, ...S_L)$. Na začiatku je C prázdne.
2. i=1
3. Kým je i=1 < L
   3.1. Nájdi najmenšiu hodnotu j tak že $S = (S_i, ...S_{i+j-1})$ obsahuje dve identické hodnoty. Ak také j neexistuje preruš slučku.
   3.2. Pridaj j do C
   3.3. i=i+j
4. T je priemerná hodnota všetkých hodnôt v poli hodnôt C.

Príklad:V prípade že začiatočná sekvencia S = (2, 1, 1, 2, 0, 1, 0, 1, 1, 2) . Prvá kolízia nastáva pre j=3, keďže druhá a tretia hodnota sú rovnaké. 3 sa pridá do pola C. Prvé tri vzorky sa zahodia a pokračuje sa v testovaní sekvencie (2, 0, 1, 0, 1, 1, 2). Kolízia nastáva pre j=4. Tretia sekvencia na zanalyzovanie je s (1,1,2) a kolízia nastáva pre j=2. Pre poslednú sekvenciu nie sú žiadne kolízie keďže sa jedná o (2). Pole C má hodnoty C = [3,4,2]. Priemerná hodnota C je 3 a teda T=3. Na testovanie binárnych dát je potrebné použiť konverziu II spomenutú v predchádzujúcom texte tejto podkapitoly.

Test maximálneho počtu kolízii

Test maximálneho počtu kolízii *(Maximum Collision Test Statistic)* funguje obdobne ako predchádzajúci test. Opäť sa počíta počet po sebe idúcich vzorkov kým sa nevyskytne duplikát avšak výsledná hodnota je maximálna hodnota pola C. Test funguje nasledovne:

1. Majme pole C. C predstavuje počet vzorkov s rovnakou hodnotou v súbore hodnôt $S = (S_1, ...S_L)$. Na začiatku je C prázdne.
2. i=1
3. Kým je i=1 < L
   3.1. Nájdi najmenšiu hodnotu j tak že $S = (S_i, ...S_{i+j-1})$ obsahuje dve identické hodnoty. Ak také j neexistuje preruš slučku.
   3.2. Pridaj j do C
   3.3. i=i+j
4. T je maximálna hodnota z hodnôt v poli C.

Príklad:V prípade že budeme uvažovať rovnaký súbor hodnôt ako v predchádzajúcom teste S = (2, 1, 1, 2, 0, 1, 0, 1, 1, 2) potom hodnota pola C = [3,4,2] a teda T=max(3,4,2)=4. Na testovanie binárnych dát je potrebné použiť konverziu II spomenutú v predchádzujúcom texte tejto podkapitoly.

Test periodicity

Cieľom testu periodicity *(Periodicity Test Statistic)* je identifikovať počet periodických štruktúr v súbore dát. Test bude uvažovať parameter p ako parameter prerušenia pričom p<L a T sa vyráta nasledovne:

1. Inicializácia T a nastavenie T na 0
2. Pre i=1 do L-p
   Ak ($S_i = Si + p$) inkrementuj hodnotu T o 1

Príklad: Ak bude súbor dát na vstupe (2, 1, 2, 1, 0, 1, 0, 1, 1, 2) hodnota p=2. Keďže hodnota $S_i = Si + p$ pre päť hodnôt i((1, 2, 4, 5 a 6) T=5.
Na testovanie binárnych dát je potrebné uplatiť prvú korekciu zmienenú v predchádzajúcom texte tejto podkapitoly.

Test kovariancie

Test kovariance *(Covariance Test Statistic)* meria silu oneskorenej korelácie. Uvažujeme hodnotu parametru prerušenia $p <$ L ako vstup. Test potom bude fungovať nasledovne:

1. Inicializácia T a nastavenie T na 0
2. Pre i=1 do L-p
   T=T+($S_i \times Si + p$) inkrementuj

Príklad: Ak sú dáta na vstupe (5, 2, 6, 10, 12, 3, 1) a parameter p=2. T sa vyráta ako $(5 \times 6) + (2 \times 10) + (6 \times 12) + (10 \times 3) + (12 \times 1) = 164$. Na testovanie binárnych dát je potrebné uplatiť prvú korekciu zmienenú v predchádzajúcom texte tejto podkapitoly.

Kompresné testy

Účelom kompresných testov *(Compression Test Statistic)* je dobrá adaptácia odstraňovaní redundancie v bitových reťazcoch. Vstupné dáta pre test sú dĺžka dátových sád a podsád ktoré sú zakódované do reťazca znakov a spracované všeobecným komprimačným algoritmom. Tento test sa vykonáva nasledovne:

1. Zakódovanie dát tak že reťazec znakov obsahujúci hodnoty bude oddelený jedinou medzerou. To znamená že zo súboru dát S= (144, 21, 139, 0, 0, 15) sa stane 144 21 139 0 0 15.
2. Kompresia reťazca znakov algoritmom bzip2.

3. T je dĺžka komprimovaného reťazca v bajtoch.

Test je možné aplikovať na binárne dáta bez použitia korekcie.

Testy ktoré boli popísané v predchádzajúcich sekciách slúžia na overenie či je daný dátový súbor IID alebo nie. Niektoré z testov napríklad kompresný test sú efektívne pri detekovaní opakujúcich sa vzorcov (reťazce sa opakujú častejšie ako by sa očakávalo keby vzorky boli IID). Iné testy (smerový test alebo testy založené na mediáne) sa sústreďujú na súvislosť medzi numerickými hodnotami nasledujúcich vzorkov pričom sa snažia nájsť trend alebo inú koreláciu napríklad vzorky s vysokou hodnotou ktoré sú nasledované vzorkami s nízkou hodnotou.

4.3.2 Dodatočné Chí kvadrát testy

Táto podkapitola obsahuje popis dodatočný Chí kvadrát štatistických testov. Tieto postupy slúžia na testovanie nezávislosti a zhodu modelu s pozorovaním (*goodness-of-fit*). Testy nezávislosti sa snažia odhaliť závislosti medzi po sebe idúcimi vzorkami dát. Testy zhody modelu s pozorovaním testujú či je rozloženie vstupnej sekvencie rovnaké pre v subsete dát určenom pre testovanie.

Testovanie nezávislosti pre nebinárne dáta

Majme vstup $S = (S_1, ...S_L)$ kde $S_i \in A = X_1, ...X_k$, nasledovné kroky sa musia vykonať odhadnutie počtu zásobníka n_{bin} potrebných pre Chí kvadrát testy.

1. Nájdenie pomeru P_i pre každé X_i v S pre $P_i = \frac{počet X_i v S}{L}$
   Vyrátanie očakávaného počtu výskytu pre každý možný pár(Z_i, Z_j) v S ako $e_(i, j) = P_i, P_j L/2$
2. Alokácia možných párov (Z_i, Z_j) začínajúca od najmenšieho $e_(i, j)$ do zásobníka podľa očakávanej hodnoty každého zásobnika je aspoň 5.Očakávaná hodnota zásobníka je rovná sume $e_(i, j)$ párov ktoré sú zahrnuté v zásobníku. Po alokácii všetkých párov ak je výsledná hodnota posledného páru menej ako 5 zlúč posledné dva páry. Počet n_{bin} bude počet zásobníkov vytvorených v tejto procedúre

Po vytvorení zásobníku sa vykoná Chí kvadrát test nasledovným postupom:

1. Premenná o bude predstavovať zoznam počtu n_{bin} každý inicializovaný pre 0. Od j=1 do L-1.
   1.1. Ak pár S_j, S_{j+1} je v zásobníku i inkrementuj O_i o 1

1.2. j=j+2

2. T je vypočítané ako T=$\sum_{i=1}^{n_{bin}} \frac{(o_i - E(Bin_i))^2}{E(Bin_i}$

Test zlyhá v prípade že T dosiahne väčšiu hodnotu ako kritická hodnota Chí kvadrát testu s $(n_{bin} - 1) - (k - 1) = n_{bin} - k$

Príklad: Ak S bude (2, 2, 3, 1, 3, 2, 3, 2, 1, 3, 1, 1, 2, 3, 1, 1, 2, 2, 2, 3, 3, 2, 3, 2, 3, 1, 2, 2, 3, 3, 2, 2, 2, 1, 3, 3, 3, 2, 3, 2, 1, 3, 2, 3, 1, 2, 2, 3, 1, 1, 3, 2, 3, 2, 3, 1, 2, 2, 3, 3, 2, 2, 2, 1, 3, 3, 3, 2, 3, 2, 1, 2, 2, 3, 3, 3, 2, 3, 2, 1, 2, 2, 2, 1, 3, 3, 3, 2, 3, 2, 1, 3, 2, 3, 1, 2, 2, 3, 1, 1).
Abeceda pozostáva z k=3 hodnoty 1, 2, 3 $p_1, p_2 ap_3$ dosiahnu 0,21 0,41 a 0,38. S L=100 očakávané výsledky budú vyzerať nasledovne:

(z_i, z_j)	(1,1)	(1,3)	(3,1)	(1,2)	(2,1)	(3,3)	(2,3)	(3,2)	(2,2)
$e_{i,j}$	2.21	3.99	3.99	4.31	4.31	7.22	7.79	7.79	8.41

Tab. 4.1: Výsledky testu nezávislosti

Páry budú alokované do $n_{bin} = 6$ zásobníkov

Zásobník	Páry	E(Bini)
1	(1,1), (1,3)	6.2
2	(3,1), (1,2)	8.3
3	(2,1), (3,3)	11.53
4	(2,3)	7.79
5	(3,2)	7.79
6	(2,2)	8.41

Tab. 4.2: Alokácia párov pre nezávislé testy

Testovanie zhody modelu s pozorovaním pre nebinárne dáta

Test slúži na kontrolu či distribúcia vzorkov je identická pre rôzne časti výstupu. V prípade že máme $S = (S_1, ...S_L)$ kde $S_i \in A = X_1, ...X_k$ budú prevedené nasledovné kroky na získanie zásobníka n_{bin} dát.

1. Premenná C_i bude počet výskytu X_i v celom súbore dát S a $e_i = c_i/10$ pre $1 \leq i \leq k$. C_i sa delí 10 pretože dataset S bude rozdelený na 10 častí.
2. Majme zoznam List[i] pričom budeme priraďovať i od najmenšieho e_i čiže do List[*1*] priradíme najmenšiu hodnotu e_i do List[*2*] druhú najmenšiu a tak ďalej.

3. Alokujme vzorky dát do zásobníka. Priradíme hodnoty List[i] zásobníku pokiaľ nedosiahneme sumu e_i pokiaľ nedosiahne počet vzoriek v zásobníku aspoň hodnoty 5 potom priradíme vzorky v zásobníku do ďalšieho List[i]. V prípade že hodnota posledného zásobníka je menej ako 5 zlúčime posledné 2 zásobníky. Po tejto procedúre bude vytvorené n_{bin} počet zásobníkov.
4. Uvažujme E_i ako očakávanú hodnotu vzoriek v zásobníku Bin$i$. E_i bude suma hodnôt e_i čiže ak Bin 1 obsahuje x_1, $x_1 0$ a $x_5 0$ potom $E_1 = e_1 + e_1 0 + e_5 0$

Po získaní zásobníka n_{bin} pre hodnoty E_i Chí kvadratický test pre zhodu s modelom sa vykoná nasledovne:

1. Rozdelenie S do 10 neprekrývajúcich sa sekvencií dĺžky $|\frac{L}{10}|$kde $S_d = (S_{d|L/10|+1}, ..., S_{d+1|L/10|})$Súbor dát sa vhodne zaokrúhli ak L nie je deliteľné 10 zvyšné vzorky sa nepoužijú.
2. T=0
3. Pre d = 0 až 9
   3.1. Pre i do n_{bin}
   3.2. Premenná O_i bude celkový počet výskytu v zásobníku Bin$i$ v S_d
   3.3. T=T+$\frac{(o_i - E_i)^2}{E_i}$

Test zlyhá ak výsledok T bude väčší ako medzná hranica Chí kvadrát 9(n_{bin}-1)

Testovanie nezávislosti pre binárne dáta

Tento test overuje nezávislosť binárnych dát. Chí kvadratický test pre nezávislosť medzi bitmi by sa mohol použiť tiež ale jeho sila je limitovaná malým výstupom. Vhodnejší test spočíva porovnávaní frekvencií m-tíc s ich očakávanými hodnotami ktoré sú vyrátané vynásobením pravdepodobnosti každého nasledujúceho bitu za predpokladu že vzorky sú nezávislé. V prípade že bity nie sú nezávislé očakávané hodnoty pravdepodobnosti m-tíc odvodené od ich bitových hodnôt pravdepodobnosti ovplyvnia celý súbor dát a Chí kvadratický test bude oveľa väčší než predpoklad. Za predpokladu že máme binárny datátový súbor hodnôt S = $S_1, ..., S_L$ dĺžka m-tíc bude definovaná nasledovne:

1. Premenné p_0 a p_1 bude podiel núl a jednotiek v S, $p_0 = \frac{\#0vS}{L}$ a $p_1 = \frac{\#1vS}{L}$
2. Nájdenie maximálneho celého čísla pre m ktoré $\min(p_0 p_1)^m \ \frac{L}{m} \geq 5$ Ak je m viac ako 11 bude mať vždy m=11 Ak m je 1 potom test zlyhá.
3. Test sa aplikuje ak m ≥ 2
4. Nastavenie T = 0
5. Rozdelenie súbora S do neprekrývajúcich sa m-bitových blokov označených ako B = $(B_1, ..., B_{\frac{L}{m}})$ Ak L nie je deliteľné m , zvyšné bity sa zahodia.

6. Pre každú možnú m-ticu $(a_1, a_2, ..., a_m)$
   6.1. Premenná o bude počet koľkokrát sa postupnosť $(a_1, a_2, ..., a_m)$ opakoval vo vstupe B
   6.2. Premenná w bude počet jednotiek v $(a_1, a_2, ..., a_m)$
   6.3. Premenná $e = p_1^w (p_0)^{m-w} \frac{L}{m}$
   6.4. T = T+$\frac{(o-e)^2}{e}$

Test zlyhá ak je T viac ako $2^m - 2$.

Testovanie zhody modelu s pozorovaním pre binárne dáta

Tento test kontroluje distribúciu počtu jednotiek v neprekrývajúcich sa intervaloch vstupných dát na zistenie či distribúcia ostavá nezmenená v celom rozsahu sekvencie. Za predpokladu že máme binárny datátový súbor hodnôt S = $(S_1, ..., S_L)$ dĺžka m-tíc bude definovaná nasledovne:

1. Nech je premenná p podiel jednotiek v celej sekvencii S. p =počet 1 / L
2. Rozdelenie S do 10 neprekrývajúcich sa sekvencií dĺžky $|\frac{L}{10}|$kde $S_d = (S_{d|L/10|+1}, ..., S_{d+1|L/10|})$Súbor dát sa vhodne zaokrúhli ak L nie je deliteľné 10 zvyšné vzorky sa nepoužijú.
3. T=0
4. Nech je očakávaný počet núl a jednotiek v sekvencii rovný $e_0 = (1-p)\frac{L}{10}$ a $e_1 = p\frac{L}{10}$
5. Pre d = 0 až 9
   5.1. Nech o_1 a o_0 sú počet núl a jednotiek v S_d
   5.2. T=T+ $\frac{(o_0-e_0)^2}{e_0} + \frac{(o_1-e_1)^2}{e_1}$

5 Stanovanie minimálnej entropie

Jedným z hlavných požiadaviek na zdroj entropie je schopnosť spoľahlivo vytvárať náhodné reťazce dát. Na uistenie že požadované hodnoty entropie na výstupe sú dosiahnuté je potrebné tieto výstupy overiť. Overenie výstupov záleží od toho či sú dané dátové sady IID alebo nie sú IDD.

Postup a testy na overenie IID sa nachádza v predchádzajúcej kapitole. Najvhodnejším testom pre IID sadu je estimátor najčastejšej spoločnej hodnoty. Ostatné testy sa používajú pre neIID zdroje. Kolízne, Markove a Compresné testy sa používajú iba na binárne vstupy. Každý z nasledovných estimátorov použije na vstupe sekvenciu S=$(S_1, ..., S_L)$ z ktorej sa dosiahne výstup A=$\{X_1, ..., X_k\}$.

5.1 Estimačné testy

V nasledovnej podkapitole sú popisané estimačné testy. Tieto testy sú implementované v praktickej časti práce. Tam kde to dáva zmysel je uvedený krátky popis prípadne podrobnejší popis algoritmu. Pre testy ktoré využívajú zložitejšie algoritmy je uvedený slovný popis a odkaz na príslušnú špecifikáciu NIST.

5.1.1 Odhad najčastejšej spoločnej hodnoty

Táto metóda založená na odhade najčastejšej spoločnej hodnote *(The Most Common Value Estimate)* nájde pomer $\hat{p}$ najčastejšej spoločnej hodnoty vo vstupnej dátovej sade. Následne sa vytvorí interval spolahlivosttti pre tento pomer. Horná hranica tohoto intervalu sa považuje za hranicu minmálnej entropie na vzorok zo zdroja. V prípade že za vstup budeme uvažovať S=$(S_1, ..., S_L)$ kde $s_i \in A = \{X_1, ..., X_k\}$:

1. Nájdenie pomeru najčastejšej hodnoty v datasete $\hat{p}$

$$\hat{p} = max\frac{\#X_i vS}{L}$$

2. Vypočítame hornú hranice pravdepodobnosti najčastejšie spoločnej hodnoty p_u ako

$$p_u = min\left(1, \hat{p} + 2,576\sqrt{\frac{\hat{p}(1-\hat{p})}{L-1}}\right)$$

3. Odhadovaná minimálna entropia tak bude $-log_2(p_u)$

5.1.2 Test kolízii

Test kolízií podľa Hagertyho a Drapera (*The Collision Estimate*)meria priemerný počet vzoriek do prvej kolízie v sete dát, kde sa za kolíziu považuje hocijaká opakujúca sa hodnota. Cieľom tejto metódy je odhad pravdepodobnosti najpravdepodobnejšej výstupnej hodnoty založenej na čase kolízie. Táto metóda poskytuje nízku spolahlivosť odhadu entropie pre zdroje šumu ktoré môžu byť ovplyvnené výstupné hodnoty. Tým je myslené že čas do kolízie je relatívne krátky. Naopak zdroje šumu ktoré majú vyššie časy medzi kolíziami poskytnú lepšiu hodnotu entropie.

V prípade že za vstup budeme uvažovať S=$(S_1, ..., S_L)$ kde $s_i \in A = \{0, 1\}$:

1. Nastavenie v=0 index=1.
2. Začatie s S_{index} krok po kroku kým sa nepozoruje hodnota ktorá sa opakuje. Tj. nájde najmenšie j pre S_i=S_j pre hocijaké i s indexom $\leq$ i<j
3. Nastav v=v+1 t_v=j-index+1 a index=j+1.
4. Opakuj kroky 2 a 3 kým sa neprejde celá dátová sada.
5. Vypočítaj priemer vzorky $\bar{X}$ a štandartnú odchýlku $\hat{\sigma}$ ako

$$\bar{X} = \frac{1}{v}\sum_{i=1}^{v} t_i, \quad \hat{\sigma} = \sqrt{\frac{1}{v-1}\sum_{i=1}^{v}(t_j - \bar{X})^2}$$

6. Použitím binárneho hľadania sa vyrieši nasledovná rovnica pre parameter p:

$$\bar{X}' = pq^{-2}(1 + \frac{1}{2}(p^{-1} - q^{-1})F(-pq^{-1}\frac{1}{2}(p^{-1} - q^{-1}))$$

7. V prípade že binárne hladanie má riešenie potom minimálne entropia je negatívna hodnota logaritmu pre parameter p:

$$min - entropy = -log_2(2)$$

ak hľadanie nemá riešenie potom sa minimálna hodnota entropie nájde ako :

$$min - entropy = log_2(2) = 1$$

5.1.3 Markov model

V Markovom modele (*The Markov Estimate*) prvého rádu výsledok nasledujúcej hodnoty záleží iba na poslednej pozorovanej hodnote v Markovom modele n-tého rádu hodnota nasledujúceho vzorku záleží na hodnotách predchádzajúcich n-pozorovaných hodnôt. Preto Markov odhad poskytuje šablónu na testovanie zdrojov so závisloťami. Markov model poskytuje odhad minimálnej entropie meraním hodnoty zo vstupného

súboru údajov. Odhad minimálnej entropie je založený na prítomnosti subsekvenicie výstupných dát na rozdiel od odhadu minimálnej entropie pre výstup.

Vzorky sú zozbierané zo zdroja šumu a špecifikované ako reťazce dĺžky d. Z týchto dát sa pravdepodobnosť pre začiatočný stav ako aj tranzičný stav vypočíta medzi akýmikoľvek dvoma stavmi . Tieto pravdepodobnosti sú použité na určenie najväčšej pravdepodobnosti hocijakého reťazca dĺžky d. Zodpovedajúca maximálna pravdepodobnosť sa použije na určenie minimálnej entropie prítomnej vo všetkých takýchto reťazcoch generovaných zdrojom šumu. Táto metóda sa používa iba na binárne vstupy. V prípade že za vstup budeme uvažovať S=$(S_1, ..., S_L)$ kde $s_i \in A = \{0, 1\}$:

1. Odhadnutie počiatočné pravdepodobnosti pre každú výstupnú hodnotu $P_0 = \frac{\#0vS}{L}$ a $P_1 = 1 - P_0$
2. T bude transformačná matica o rozmeroch 2×2

	0	1
0	$P_{0,0}$	$P_{0,1}$
1	$P_{1,0}$	$P_{1,1}$

   Kde sa pravdepodobnosti vypočítajú ako:
   $P_{0,0}=\frac{\#00vS}{\#00vS+\#01vS}$, $P_{0,1}=\frac{\#01vS}{\#01vS+\#01vS}$
   $P_{1,0}=\frac{\#10vS}{\#10vS+\#11vS}$, $P_{1,1}=\frac{\#01vS}{\#10vS+\#11vS}$

3. Nájdenie pravdepodobnosti najpravdepodobnejšieho výstupu sekvencie o dĺžke 128 je vypočítané nasledovne:

Sekvencia	Pravdepodobnosť
00...0	$P_0 \times P_{0,0}^{127}$
0101...01	$P_{0,0} \times P_{0,1}^{64} \times P_{1,0}^{63}$
0111...1	$P_{0,0} \times P_{0,1} \times P_{1,1}^{126}$
0100...0	$P_1 \times P_{1,0} \times P_{0,0}^{126}$
1010...10	$P_1 \times P_{1,0}^{64} \times P_{0,1}^{63}$
11...1	$P_1 \times P_{1,1}^{127}$

4. $\hat{p}_{max}$ bude maximálna pravdepodobnosť vychádzajúca z vyššie uvedenej tabuľky minimálna entropia bude negatívny logaritmus pravdepodobnosti najčastejších výstupov $\hat{p}_{max}$

$$min - entropy = min(\text{-}\log_2\hat{p}_{max}\ /128,1)$$

5.1.4 Odhad t triedy

Odhad t triedy *(t-Tuple Estimate)* je metóda založená na skúmaní frekvencie počtu t-tíc(páry, dvojice trojice) ktoré sa objavia v súbore dát a tvoria odhad entropie pre vzorok, založenom na výskyte týchto t-tíc. Frekvencia t-tice $(r_1, r_2, ...r_t)$ v S=$(S_1, ..., S_L)$ je počet i kde $s_i = r_1, s_{i+1} = r_2, ..., s_{i+t-1}$. Tieto t-tice sa môžu prekrývať. V prípade že za vstup budeme uvažovať S=$(S_1, ..., S_L)$ kde $s_i \in A = \{X_1, ..., X_k\}$:

1. Nájdi najväčšie t pre ktoré je počet výskytu v najviac opakujúcej sa t-tici v S aspoň 35.
2. Do pola Q[i] naplň počet opakovaní najčastejšej t-tice v S pre i=1,...,t.
3. Pre $i = 1$ nech $P[i] = Q[i]/(L - j + 1)$ a vypočítaj odhad maximálnej hodnoty vzorky ako $P_{max}[i] = P[i]^{1/i}$ nech $\hat{p} = max(P_max[1], ..., P_max[t])$
4. Vypočítame hornú hranice pravdepodobnosti najčastejšie spoločnej hodnoty p_u ako

$$p_u = min\left(1, \hat{p} + 2,576\sqrt{\frac{\hat{p}(1-\hat{p})}{L-1}}\right)$$

5. Odhadovaná minimálna entropia tak bude $-log_2(p_u)$

5.1.5 Najdlhší opakovaný subreťazec

Test najdlhšieho opakovaného subraťazca *(Longest Repeated Substring (LRS) Estimate)* je metóda kde sa odhadujú kolízie zdroja entropie založené na opakovaní subreťazcov medzi vstupnou sadou dát. Táto metóda sa používa na reťazce ktoré sú príliš dlhé pre odhad t-triedy. V prípade že za vstup budeme uvažovať S=$(S_1, ..., S_L)$ kde $s_i \in A = \{X_1, ..., X_k\}$:

1. Nájdenie najmenšej hodnoty u tak aby sa počet výskytu najčastejšieho u reťazca súbora v S bolo menej ako 35.
2. Nájdenie najväčšej hodnoty v tak aby sa počet výskytu vo u reťazcoch súbora S bolo aspoň 2 a najčastejšia (v+1)tica sa vyskytla v súbore S aspoň raz. Inými slovami v nadobudne najväčšiu dĺžku pre opakované reťazce. Ak $v<u$ odhad sa nevypočíta.
3. Pre W=u do w vypočítaj pravdepodobnosť kolízie reťazca W.

$$P_w = \frac{\sum_i \binom{C_i}{2}}{\binom{L-W+1}{2}}$$

4. Vypočítame hornú hranice pravdepodobnosti najčastejšie spoločnej hodnoty p_u ako

$$p_u = min\left(1, \hat{p} + 2,576\sqrt{\frac{\hat{p}(1-\hat{p})}{L-1}}\right)$$

5. Odhadovaná minimálna entropia tak bude $-log_2(p_u)$

5.1.6 Odhad najčastejšie opakujúceho sa okna

Odhad najčastejšie opakujúceho sa okna *The Multi Most Common in Window - MultiMCW)* obsahuje niekoľko subpredpovedí pričom každá z nich sa sústredí na uhádnutie správneho výstupu. Tieto predpovede sú založené na posledný w výstupoch. Každá subpredpoveď počíta hodnotu ktorá sa vyskytla najčastejšie v okne predchádzajúcich W výstupov. MultiMCW predikcia má tabuľku hodnôt podľa ktorej zistí koľkokrát bola ktorá subpredpoveď pravdivá a použije túto subpredpoveď pre predikovanie ďalšej hodnoty. V prípade že je výsledok rovnaký vzorka ktorá sa vyskytla najčastejšie a zároveň bola posledná je použitá na predikciu. Prediktor bol navrhnutý pre prípady kedy sa najčastejšia spoločná hodnota mení v závislosti na čase ale zároveň ostáva rozumne stabilná počas dlhotrvajúcej periódy.Podrobný popis obsahuje špecifikácia NIST [21].

5.1.7 Viacnásobná MMC predikcia

Viacnásobná MMC predikcia *(The MultiMMC Prediction Estimate)* pozostáva z viacerých Markovových modelov s počítaním subpredpovedí. Každý MMC predpoveď zaznamenáva dáta a pozorované frekvencie pre tranzíciu z jedného výstupu na druhý až do následného výstupu. V typickom Markovovom modele sa počíta s pravdepodobnosťou tranzície od začiatku) a vytvorí predpoveď založenú na najčastejších tranzíciach zo súčasného výstupu. Multi MMC obsahuje D subpredpoveď ktorá pracuje paralelne pre každú hĺbku 1 až D. Napríklad MMC s hĺbkou 1 vytvorí model prvého poradia zatiaľ čo MMC s dĺžkou D vytvorí model Dtého poradia. Multi MMC si ukladá hodnoty počtu správnych predpovedí do tabuľky a používa subprediktor s najčasttejšími správnymi predveďami pre ďalšie predpovede. Podrobný popis obsahuje špecifikácia NIST [21].

5.1.8 LZ78Y predikcia

Predikcia LZ78Y(*The LZ78Y Prediction Estimate*) je založená na LZ78 enkódovaní s Berstein Yabbahovej schéme pre pridávanie reťazca do slovníka. Prediktor má slovník reťazcov ktorý sú pridávané do slovníka a pokračuje v pridávaní nových reťazcov pokiaľ slovník nedosiahne maximálnu kapacitu. Vždy keď je spracovaný

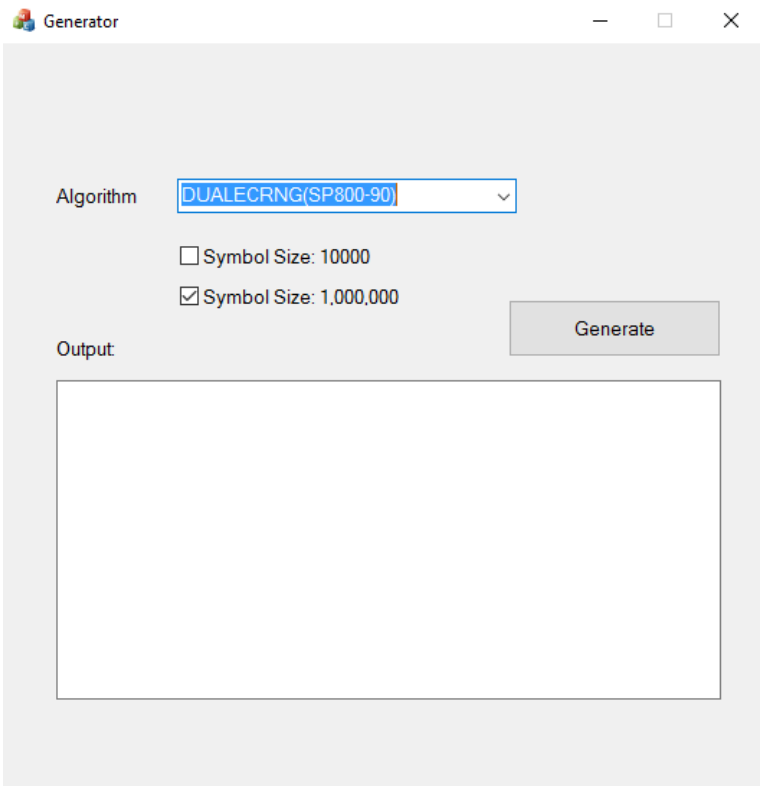
vzorok každý podreťazec v posledných B vzorkov slovník sa aktualizuje a vzorok je zároveň pridaný do slovníka. Podrobný popis obsahuje špecifikácia NIST [21].

6 Praktická časť

Praktická časť tejto diplomovej práce sa skladá z vytvorenia programu pre vyhodnocovanie kvality entropie založenom na štandarte SP800-90B, generátorov pre platformy Windows a Linux a samotného merania na týchto platformách.

6.1 Generátor pre operačný systém Windows

Generátor pre platformu Windows využíva CryptoAPI ktoré je popísané v kapitole 2.3. Samotná implementácia je v jazyku C++ vývojového prostredia Visual Studio 2017 za použitia frameworku MFC pre grafické rozhranie. V generátore sú imple-



Obr. 6.1: Generátor pre operačný systém Windows.

mentované 3 algoritmy pomocou ktorých je možno generovať náhodné dáta.
Sú to :

- **RNG-FIPS186-2, FIPS140-2**
- **DUALECRNG SP800-90**
- **FIPS186DSARNG**

Ďalej má užívateľ možnosť zvoliť veľkosť generovaného súboru. Keďže podľa NIST sa za spoľahlivé meranie považuje iba také kde dátový súbor obsahuje aspoň 1 000 000

znakov , užívateľ má možnosť vybrať generovanie súboru 10 000 a 1 000 000. Zmysel generovania menších súborov predstavuje časovú úsporu pri testovaní a ladení hlavnej aplikácie.

Popis základných funkcií kódu

V nasledujúcej podsekcii je popísaná funkcionalita a logika základných funkcií a metód programu. Neuvádza do detailu popis prvkov slúžiacich pre grafické rozhranie ale kód slúžiaci ako jadro generátora.

```
CFileDialog dlg(FALSE, _T("bin"), NULL, OFN_HIDEREADONLY |
   OFN_OVERWRITEPROMPT, szFilter, this);
```

Pomocou tejto funkcie otvárame dialogové okno pre uloženie dát.

```
AfxBeginThread(MyThreadProc, 0);
    destFileName = sFilePath;
    strStatus += sFilePath;
    m_ctlResult.SetWindowText(strStatus);
  }
```

Spustenie vlákna pre generovanie náhodných čísiel.

```
UINT MyThreadProc(LPVOID Param {
CEntroyGeneratorDlg *dlg = (CEntroyGeneratorDlg *) AfxGetApp()->
   GetMainWnd();
  ULONG rep = strtoul("20", NULL, 10);
  ULONG size = strtoul("125000", NULL, 10);
  if(!isLarge)
    size = strtoul("1250", NULL, 10);
  PUCHAR data = (PUCHAR)calloc(size + 1, 1);
  BCRYPT_ALG_HANDLE alg = NULL;
  LPCWSTR algorithm;
  }
```

V tejto časti sa nastavuje veľkosť symbolu.

```
switch (dlg->m_ctlAlgorithm.GetCurSel())
  {
  case 0:
    algorithm = BCRYPT_RNG_ALGORITHM;
    break;
  case 1:
    algorithm = BCRYPT_RNG_DUAL_EC_ALGORITHM;
```

```
    break;
  case 2:
    algorithm = BCRYPT_RNG_FIPS186_DSA_ALGORITHM;
    break;
  default:
    algorithm = BCRYPT_RNG_ALGORITHM;
    break;
  }
```

Pomocou podmienky `switch-case` si užívateľ volí želaný algoritmus.

```
NTSTATUS c = BCryptOpenAlgorithmProvider(&alg, algorithm, NULL, 0);
  if (c != 0) {
    std::cerr << "BCryptOpenAlgorithmProvider fail: ";
    switch (c) {

    case STATUS_INVALID_PARAMETER: std::cerr << "invalid parameter";
      break;
    case STATUS_NO_MEMORY: std::cerr << "memory allocation failure";
    }
    std::cerr << std::endl;
    return 0;
  }
```

Tu sa volajú knižnice poskytovateľa algoritmu.

```
for (ULONG i = 0; i < rep; i++) {
    memset(data, 0, size + 1);
    high_resolution_clock::time_point start = high_resolution_clock::
    now();
    c = BCryptGenRandom(alg, data, size, 0);
    high_resolution_clock::time_point end = high_resolution_clock::now
    ();

  }
```

Časť ktorá volá API funkciu pre generáciu náhodných dát. Náhodné číslo je uložené do premennej `data`.

```

CFile file;
    file.Open(destFileName, CFile::typeBinary | CFile::modeCreate |
    CFile::modeWrite);
    int nSize = 125000;
    if (!isLarge)
      nSize = 1250;
    for (int i = 0; i<nSize; i++)
    {
      char Binary[8] = { '\0' };
      char c = data[i];
```

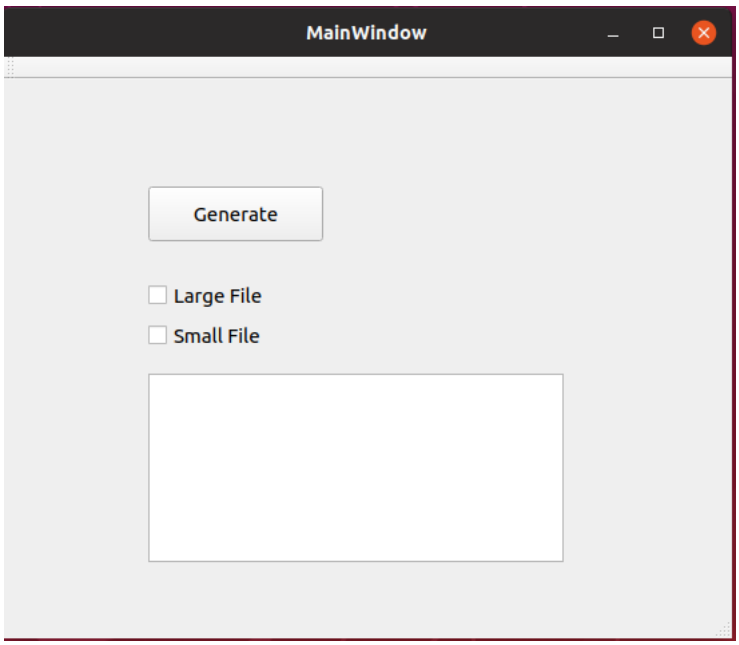
```
11          int ascii = (unsigned int)c;
12          ascii = abs(ascii);
13          dlg->Convert(ascii, Binary);
14          file.Write(Binary, 8);
15      }
16      file.Close();
```

Ukladanie súboru s dátami na disk a konverzia dát na binárneho formátu.

6.2 Generátor pre operačný systém Linux

Generátor pre operačný systém linux je napísaný v C++ a ako zdroj entropie využíva `/dev/urandom`. Teoretický popis linuxovej implementácie zdrojov entropie je uvedený v kapitole 2.4.

Uživateľ ma opäť analogicky možnosť vytvoriť veľký alebo malý súbor v závis-



Obr. 6.2: Generátor pre operačný systém Linux.

losti od jeho požiadaviek. Program využíva framework Qt pre vytvorenie grafického rozhrania.

Popis základných funkcií kódu

```
1 QString fileName = QFileDialog::getSaveFileName(this,
2           tr("Save to File"), "/home", tr("BIN Files (*.bin)"));
3       char *str=(char *)malloc(10);
```

Otvorenie dialógového okna pre uloženie súbora.

```
int nSize = 125000;
      if(ui->checkBoxSmall->isChecked())
         nSize = 1250;
      QByteArray ba=fileName.toLatin1();
      strcpy(str,ba.data());
      ui->textEdit->append("Generating...");
      using namespace std;
      char myRandomData[nSize];
      size_t size = nSize
```

Nastavenie veľkosti dát. Posielanie správy do konzolového okna o tom že sa generujú dáta a deklarácia veľkosti dát.

```
ifstream urandom("/dev/urandom", ios::in|ios::binary);
      if(urandom)
      {
          urandom.read(myRandomData, size);
          if(urandom) //Check if stream is ok, read succeeded
          {

            FILE * st = fopen(str, "w");
```

Táto sekcia kódu slúži k otvoreniu zásobníka `/dev/urandom` čítanie z neho a následný zápis.

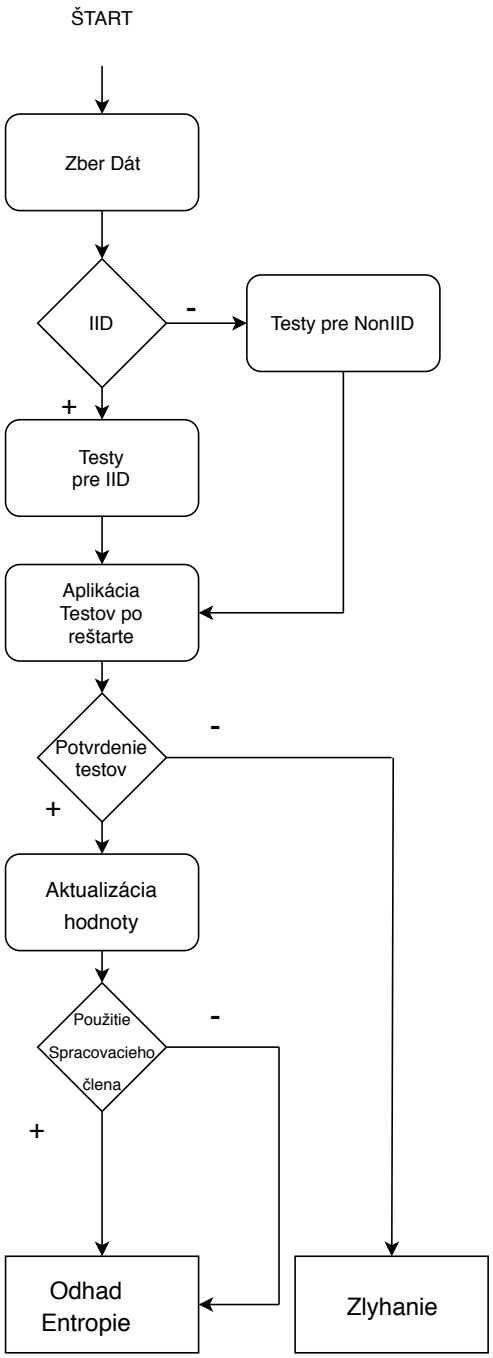
```
for (int i = 0; i<nSize; i++)
                {
                    char Binary[8] = { '\0' };
                    char c = myRandomData[i];
                    int ascii = (unsigned int)c;
                    ascii = abs(ascii);
                    Convert(ascii, Binary);
                     fwrite(Binary, 1, 8, st);
                }
              fclose(st);
          }
```

Zápis a konverzia dát.

6.3 Program na vyhodnocovanie entropie

Program na hodnotenie entropie je naprogramovaný v jazyku C++ a využíva knižnice MFC pre grafické rozhranie a `bzip2` pre prácu s komprimovaním bitov ktorá je potrebná v algoritmoch pre hodnotenie entropie popísaných v kapitole 5.1. Základná časť vychádza z repoziráru na github ktorý verejne poskytuje NIST spolu s odporúčaním na hodnotenie entropie [21].

Na Obr.6.3 je zobrazený algoritmus na základe ktorého prebieha hodnotenie entropie. Po zbere dát je potrebné rozhodnúť či sú dáta v súbore pochádzajú zo

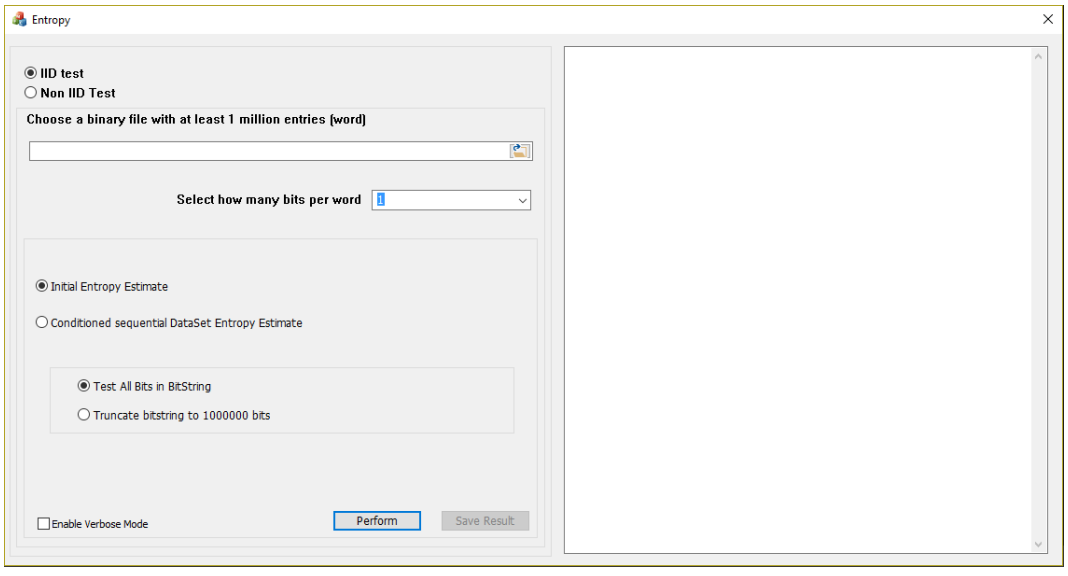


Obr. 6.3: Algoritmus pre hodnotenie entropie[21]

nezávislého a identicky distribuovaného zdroja šumu. Následne sa uplatnia testy po reštarte v prípade že zlyhajú zdroj sa nepovažuje za spoľahlivý a program sa ukončí. Ďalším krokom je dotaz na to či sa používa výstup priamo zo zdroja šumu alebo sa upravuje spracovávacím členom. Podľa toho sa uplatnia ďalšie algoritmy a vyhodnotí sa entropia vzorky dát.
V samotnom užívateľskom rozhraní užívateľ zvolí možnosť ktoré testy chce na vzorke

dát vykonať. Následne vyberie z adresárovej štruktúry cestu k dátam. Ostatné možnosti v užívateľskom rozhraní sú:

- **Voľba počtu bitov na slovo** Užívateľ má možnosť si vybrať pičet bitov na slovo od 1 až po 8.
- **Počiatočná entropia** Po vybratí možnosti *Initial Entropy Estimate* bude program rátať s tým že dáta boli priamo odobraté zo zdroja šumu a neboli spracovávané ďalším členom.
- **Vzorok po spracovaní** Pri zaškrtnutí možnosti *Conditioned sequential Dataset Entropy Estimate* bude program počítať s tým že súčasťou zdroja entropie bol člen pre spracovanie.
- **Testovanie všetkých bitov** Po vybraní možnosti *Test All Bits in BitString* budú otestované všetky dáta v dátovom súbore
- **Orezanie vzorku na 1 000 000 znakov** Možnosť *Truncate to 1 000 000* naopak oreže príliš veľké vzorky a tak ušetrí čas potrebný na dokončenie výpočtov. Počet vzorkov 1 000 000 je považovaný za dostatočný pre vykonanie testov. V prípade že sa testuje dátový súbor o menšom počte program upozorní chybovou hláškou ale dokončí testy.
- **Podrobný mód** Zaškrtnutie pola *Enable Verbose Mode* umožní zapnúť resp. vypnúť vypisovanie podrobností do konzolového okna o priebehu testov.



Obr. 6.4: Program pre hodnotenie entropie.

Popis základných funkcií kódu

Program je rozdelený na 2 hlavné vetvy podľa IID respektíve neIID testov.

IID

```
calc_stats(&data, rawmean, median);
```

Táto funkcia vypočíta základné štatistiky nájde priemerné hodnoty a medián bez ohľadu na to či sú dáta binárne alebo nie.

```
double H_min = most_common(data.symbols, sample_size, alphabet_size,
    verbose);
ubound = min(1.0,pmax + ZALPHA*sqrt(pmax*(1.0-pmax)/(len-1.0)));
```

Vyrátanie minimálnej entropie na základe testu najčastejšej spoločnej hodnoty (MCV)

```
bool chi_square_test_pass = chi_square_tests(data.symbols, sample_size,
     alphabet_size, verbose);
```

Funkcia ktorá ráta Chí kvadratické testy

```
bool len_LRS_test_pass = len_LRS_test(data.symbols, sample_size,
    alphabet_size, verbose);
```

Funkcia na vyrátanie najdlhšieho opakujúceho sa reťazca.

```
bool perm_test_pass = permutation_tests(&data, rawmean, median, verbose
    );
```

Permutačné testy, ich výsledkom je tlač na konzolu o tom či sa jedná alebo nejedná o IID dáta.

non-IID

```
ret_min_entropy = most_common(data.bsymbols, data.blen, 2, verbose);
```

Test entropie na základe najčastejšej spoločnej hodnoty.

```
ret_min_entropy = collision_test(data.bsymbols, data.blen, verbose);
```

Test kolízii - prebehne iba pre bitové reťazce.

```
ret_min_entropy = markov_test(data.bsymbols, data.blen, verbose);
```

Markov test pre binárne bitové reťazce.

```
ret_min_entropy = compression_test(data.bsymbols, data.blen, verbose);
```

Kompresný test pre bitové reťazce.

```
double bin_t_tuple_res = -1.0, bin_lrs_res = -1.0;
double t_tuple_res = -1.0, lrs_res = -1.0;
SAalgs(data.bsymbols, data.blen, 2, bin_t_tuple_res, bin_lrs_res,
    verbose);
```

```
4 SAalgs(data.symbols, data.len, data.alph_size, t_tuple_res, lrs_res,
      verbose);
```

Test pre T-tice.

```
1 ret_min_entropy = multi_mmc_test(data.bsymbols, data.blen, 2, verbose);
2 ret_min_entropy = multi_mmc_test(data.symbols, data.len, data.alph_size
      , verbose);
```

MMC test.

```
1 ret_min_entropy = LZ78Y_test(data.bsymbols, data.blen, 2, verbose);
2 ret_min_entropy = LZ78Y_test(data.symbols, data.len, data.alph_size,
      verbose);
```

LZ78Y test.

7 Meranie a analýza zdrojov entropie

Kompletné výsledky meraní sú uvedené v prílohe A. Meranie prebiehalo na fyzických počítačoch s operačným systémom Windows a distribúciou Linux Ubuntu verzie 18.04 . Postup pri meraní bol nasledovný:

1. Najprv sa vygenerovali dáta dostatočnej veľkosti za pomoci generátorov zmienených v kapitole 6.2 .
2. Následne bola zvolená možnosť dát ktoré nesplňujún požiadavky IID a bez úprav upravovacím členom.
3. Nebolo ich nutné orezávať keďže generátory vždy vyprodukujú presne dátovy set o 1 000 000 bitov.
4. Bol zvolený podrobný mód výpisu do konzoly.

Ďalšie merania prebiehali vo virtuálnych cloudových prostrediach Microsoft Azure a Amazon Web Services. Študenti VUT majú prístup ku kreditu 100$ a služby od Amazonu majú veľmi jednoduché rýchle a bezplatné možnosti nasadenia virtuálnych strojov. Toto boli hlavné dôvody pre zvolenie týchto dvoch platforiem. Cloudové prostredia zaznamenávajú stále väčší a väčší rozmach z dôvodu škálovateľnosti a šetrenia nákladov na tradičné dátové centrá. Literatúra uvádza možné zraniteľnosti a útoky na zdroje entropie vo virtuálnych prostrediach Amazon.[10] Útok na zásobnik entropie jeho tzv. otrávením nebol predmetom skúmania tejto diplomovej práce ale okrem toho sa v článku spomínal všeobecný problém s nedostatočnou entropiou z dôvodu princípu fungovania virtuálnych strojov a kopírovanie inštancíi medzi nimi.

	Fyzický stroj	AWS	Azure
Windows	0.567281	0.565893	0.563778
Linux	0.564351	0.562124	0.559642

Tab. 7.1: Výsledky meraní výsledok H_bitstring

Ako vyplýva z Tab.7.1 v súčastnosti takmer nie je rozdieľ medzi entropiou generovanou na fyzickom stroji a vo virtuálnom prostredí a tak možno považovať generovanie náhodných dát za bezpečné. Samozrejme v Cloudových prostrediach sú iné výzvy ako zabezpečenie prístupu či správna konfigurácia.

8 Záver

V tejto diplomovej práci boli popísané entropia,generátory náhodnych čísel a ich význam v kryptografii. Ďalej podrobný popis testov na určovanie hodnoty entropie zo zdrojov šumu. Meranie entropie sa odvíja od niekoľkých faktorov. Medzi hlavný patrí to či je vôbec zdroj entropie schopný poskytovať dostatočné náhodné dáta tj. či je vhodný a nedá sa jednoducho ovplyvniť či predpokladať ako bude fungovať.

Ďalej sa odhad entropie odvíja od toho či sú dáta generované nezávisle a identicky distribuovane. Na základe toho či sú dáta IID alebo nie sú IDD sa na nich vykonajú rôzne testy. Priemer týchto testov je potom považovaný za výsledok hodnoty entropie.

Súčasťou tejto diplomovej práce bolo vytvorenie 2 generátorov náhodnych čísel založených na dostupných zdrojoch entropie v operačnych systémoch Windows a Linux aplikovaním Microsoft CryptoAPI a `/dev/urandom`.

Ďalším programom ktorý bol vytvorený je samotný program na hodnotenie entropie založený na štandartoch a odporúčaniach NIST. Následne bolo vykonané meranie na súboroch dát ktoré boli získané na fyzických pracovných staniciach a na Cloudových prostrediach Amazon Webservices a Microsoft Azure. Keďže Cloud zažíva stále väčší a väčší rozmah predmetom ďalšieho možného skúmania môžu byť Cloudové služby od spoločnosti Google prípadne menej známe ako Alibaba, IBM či Oracle. Tieto služby sa nemusia skúmať len z pohľadu púheho generovania dát ale aj pokusy o ich ovplyvnenie popísaným v literatúre.

Literatúra

[1] Online slovník: *Vocabulary online definition* [online]. [cit. 23. 10. 2018]. Dostupné z URL:
<https://www.vocabulary.com/dictionary/entropy>.

[2] DRAKE,G.,: *Online vydanie encyklopédie Britannica* [cit. 23. 10. 2018]. Dostupné z URL:
<https://www.britannica.com/science/entropy-physics>.

[3] MARKOWSKI,G.,: *Online vydanie encyklopédie Britannica* [cit. 23. 10. 2018]. Dostupné z URL:
<https://www.britannica.com/science/information-theory/Classical-information-theory#ref214944>.

[4] WILEY, J.; *Applied cryptography: protocols, algorithms and source code in C.*. Vyd. 2. New York: 1996. 232 s. ISBN 0-471-11709-9.

[5] MASSEY, J,: *Guessing and Entropy.* Signal and Info. Proc. Lab., Swiss Federal Inst. Tech, CH-8092 Zurich, Switzerland [cit. 23. 10. 2018]. Dostupné z URL:
<http://www.isiweb.ee.ethz.ch/archive/massey_pub/pdf/BI633.pdf>.

[6] MALONE, D., WAYNE Sullivan: *Guessing is not a substitude for entropy.* Department of Mathematics, UCD, Dublin,Ireland. [cit. 23. 10. 2018]. Dostupné z URL:
<http://www.maths.tcd.ie/~dwmalone/p/itt05.pdf>.

[7] LIEBOW, J.: *Randomness 101: LavaRand in Production*

[cit. 21. 1. 2019]. Dostupné z URL:
<https://blog.cloudflare.com/randomness-101-lavarand-in-production/>.

[8] DORRENDORF, L.: *Cryptanalysis of the Windows Random Number Generator* School of Engineering and Computer Science, The Hebrew University of Jerusalem, Israel [cit. 21. 1. 2019]. Dostupné z URL:
<https://eprint.iacr.org/2007/419.pdf>.

[9] PINKAS, B.,GUTTERMAN,Z.: *Analysis of the Linux Random Number Generator* University of Haifa, Israel [cit. 21. 1. 2019]. Dostupné z URL:
<https://eprint.iacr.org/2006/086.pdf>.

[10] KERRIGAN, B.,CHEN,Y.: *A Study of Entropy Sources in Cloud Computers: Random Number Generation on Cloud Hosts* Dept. of Electrical and Computer

Engineering, SUNY - Binghamton [cit. 21. 1. 2019]. Dostupné z URL: <http://harvey.binghamton.edu/~ychen/chen-kerrigan.pdf>.

[11] GOLDBERG, I., WAGNER D.: *Randomness and the Netscape Browser.* [cit. 21. 1. 2019]. Dostupné z URL: <https://people.eecs.berkeley.edu/~daw/papers/ddj-netscape.html>.

[12] DOLE, B., LODIN S.: *Misplaced Trust: Kerberos 4 Session Keys.* [cit. 21. 1. 2019]. Dostupné z URL: <https://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=2331&context=cstech>.

[13] GUTTERMAN, Z., MALKHI D.: *Hold Your Sessions: An Attack on Java Session-Id Generation* School of Engineering and Computer Science, The Hebrew University of Jerusalem, Israel [cit. 21. 1. 2019]. Dostupné z URL: <http://leibniz.cs.huji.ac.il/tr/765.files%5CGuttermanMalkhi2005.pdf>.

[14] WATROUS, J,: *Theory of Quantum Information.*

[cit. 23. 10. 2018]. Dostupné z URL: <https://cs.uwaterloo.ca/~watrous/TQI/>.

[15] BURDA,K.,: *Kryptografické generátory. Sdělovací technika, 2016, č. 6.*

[16] BARKER,E.,ROGINSKY,A. : *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths* [cit. 23. 2. 2019]. Dostupné z URL: <https://csrc.nist.gov/publications/detail/sp/800-131a/archive/2011-01-13>.

[17] SHANNON, C.E.,: *A Mathematical Theory of Communication* The Bell System Technical Journal, Vol. 27, pp. 379–423, 623–656, July, October, 1948. [cit. 25. 10. 2018].Dostupné z URL: <https://sites.google.com/site/parthochoudhury/aMToC_CShannon.pdf? >

[18] HERBERT, H.,: *Digital random number generator using partially entropic data.* Phoenix: United States Patent US8489660B2, 2009. [cit. 25. 10. 2018].Dostupné z URL: <http://www.freepatentsonline.com/8489660.pdf >

[19] MÜLLER, S.,: *Linux Random Number Generator - A New Approach* August 12, 2018 [cit. 25. 10. 2018].Dostupné z URL: <http://www.chronox.de/lrng/doc/lrng.pdf >

[20] SCHONNING,N.,HALFIN,D.: *Federal Information Processing Standard (FIPS) 140 – Security Requirements for Cryptographic Modules* August 12, 2018 [cit. 25. 10. 2018].Dostupné z URL: <`https://docs.microsoft.com/en-us/windows/security/threat-protection/fips-140-validation` >

[21] TURAN,M., BARKER,E., KELSEY,J.: *Recommendation for the Entropy Sources Used for Random Bit Generation.* Phoenix: United States Patent US8489660B2, 2009. [cit. 25. 1. 2019].Dostupné z URL: <`https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf` >

Zoznam symbolov, veličín a skratiek

API	rozhranie pre programovanie aplikácií – Application programming interface
SSL	vrstva bezpečných soketov – Secure Sockets Layer
PRNG	Generátor pseudonáhodnych čísel – Pseudorandom Number Generator
DRBG	Deterministický generátor náhodnych čísel – deterministic random bit generator
CSP	Poskytovateľ kryptografickej služby – Crypto Service Provider
SHA	Bezpečné hešovacie algoritmy – Secure Hash Algorithm
RC4	Prúdová šifra – Rivest Cipher 4
IID	Nezávisle a identicky distribuované – Independent and identically distributed random variables
LRS	Najdlhší opakovaný subreťazec – Longest repeated substring
MCW	Najčastejšie opakujúce sa okno – Multi Most Common in Window
MMC	Viacnásobný Markov odhad – MultiMMC Prediction Estimate
AWS	Webové služby Amazon – Amanzon Webservices

Zoznam príloh

A Výsledky meraní **76**
A.1 Windows 76
A.1.1 Windows 76
A.1.2 WindowsAWS 77
A.1.3 WindowsAzure 78
A.2 Linux 79
A.2.1 Linux 79
A.2.2 LinuxAWS 80
A.2.3 LinuxAzure 81

B Obsah priloženého CD **83**

A Výsledky meraní

A.1 Windows

A.1.1 Windows

Meranie

```
opening file: 'C:\Users\Martin\Desktop\GenData\Windows.bin'
Number of Binary Symbols: 1000000
Symbol alphabet consists of 2 unique symbols
Running non-IID tests...
Running Most Common Value Estimate...
MCV Estimate: mode = 561902, p-hat = 0.56190200000000001, p_u = 0.56318000696577319
Most Common Value Estimate bit string = 0.828332 / 1 bits
Running Entropic Statistic Estimates bit strings only...
Collision Estimate: X-bar = 2.4991540355130142, sigma-hat = 0.4999999091329857,
p = 0.53796045595044006
Collision Test Estimate bit string = 0.894428 / 1 bits
Markov Estimate: P_0 = 0.56190200000000001, P_1 = 0.43809799999999999,
P_0,0 = 0.55539498950882804, P_0,1 = 0.44460501049117196, P_1,0 = 0.57024683974818424,
P_1,1 = 0.42975316025181576, p_max = 2.0611479835196668e-33
Markov Test Estimate bit string = 0.848283 / 1 bits
Compression Estimate: X-bar = 5.0986323815214964,sigma-hat = 1.0404843257325547,
p = 0.094490532628997981
Compression Test Estimate bit string = 0.567281 / 1 bits
Running Tuple Estimates...
t-Tuple Estimate: t = 17, p-hat_max = 0.56190200000000001, p_u = 0.56318000696577319
LRS Estimate: u = 18, v = 39, p-hat = 0, p_u = 0
T-Tuple Test Estimate bit string = 0.828332 / 1 bits
LRS Test Estimate bit string = inf / 1 bits
Running Predictor Estimates...
MultiMCW Prediction Estimate: N = 999937, Pglobal' = 0.56264158545881493
C = 561328 Plocal can't affect result r = 20
Multi Most Common in Window MultiMCW Prediction Test Estimate
bit string = 0.829712 / 1 bits
Lag Prediction Estimate: N = 999999, Pglobal' = 0.56328653533749695
 C = 562008 Plocal can't affect result r = 21
Lag Prediction Test Estimate bit string = 0.828059 / 1 bits
MultiMMC Prediction Estimate: N = 999998, Pglobal' = 0.56317313424371962
 C = 561894 Plocal can't affect result r = 20
Multi Markov Model with Counting MultiMMC Prediction Test Estimate
bit string = 0.828350 / 1 bits
LZ78Y Prediction Estimate: N = 999983, Pglobal' = 0.56317557151670328
C = 561888 Plocal can't affect result r = 20
LZ78Y Prediction Test Estimate bit string = 0.828343 / 1 bits
```

```
H_bitstring: 0.567281
```

A.1.2 WindowsAWS

Meranie

```
opening file: 'C:\Users\Martin\Desktop\GenData\WindowsAWS.bin'
Number of Binary Symbols: 1000000
Symbol alphabet consists of 2 unique symbols
Running non-IID tests...
Running Most Common Value Estimate...
MCV Estimate: mode = 562596, p-hat = 0.56259599999999998, p_u = 0.56387378266470856
Most Common Value Estimate bit string = 0.826556 / 1 bits
Running Entropic Statistic Estimates bit strings only...
Collision Estimate: X-bar = 2.5000125000625002, sigma-hat = 0.50000062484804564,
 p = 0.53181099997686943
Collision Test Estimate bit string = 0.911014 / 1 bits
Markov Estimate: P_0 = 0.56259599999999998, P_1 = 0.43740400000000002,
P_0,0 = 0.55576924786036142, P_0,1 = 0.44423075213963858, P_1,0 = 0.5713756618595166,
P_1,1 = 0.4286243381404834, p_max = 2.2480176911492453e-33
Markov Test Estimate bit string = 0.847304 / 1 bits
Compression Estimate: X-bar = 5.0972898033786862,sigma-hat = 1.042442092682526,
p = 0.095037539909293534
Compression Test Estimate bit string = 0.565893 / 1 bits
Running Tuple Estimates...
t-Tuple Estimate: t = 17, p-hat_max = 0.56259599999999998, p_u = 0.56387378266470856
LRS Estimate: u = 18, v = 41, p-hat = 0, p_u = 0
T-Tuple Test Estimate bit string = 0.826556 / 1 bits
LRS Test Estimate bit string = inf / 1 bits
Running Predictor Estimates...
MultiMCW Prediction Estimate: N = 999937, Pglobal' = 0.56332440968348352
C = 562011 Plocal can't affect result r = 25
Multi Most Common in Window MultiMCW Prediction Test Estimate
bit string = 0.827962 / 1 bits
Lag Prediction Estimate: N = 999999, Pglobal' = 0.56321055971316791 C = 561932
Plocal can't affect result r = 19
Lag Prediction Test Estimate bit string = 0.828254 / 1 bits
MultiMMC Prediction Estimate: N = 999998, Pglobal' = 0.56385591490777809
C = 562577 Plocal can't affect result r = 25
Multi Markov Model with Counting MultiMMC Prediction Test Estimate
bit string = 0.826602 / 1 bits
```

```
LZ78Y Prediction Estimate: N = 999983, Pglobal' = 0.56386835933238943
C = 562581 Plocal can't affect result r = 25
LZ78Y Prediction Test Estimate bit string = 0.826570 / 1 bits
H_bitstring: 0.565893
```

A.1.3 WindowsAzure

Meranie

```
opening file: 'C:\Users\Martin\Desktop\GenData\Windows_Azure.bin'
Number of Binary Symbols: 1000000
Symbol alphabet consists of 2 unique symbols
Running non-IID tests...
Running Most Common Value Estimate...
MCV Estimate: mode = 562090, p-hat = 0.56208999999999998, p_u = 0.56336794645493438
Most Common Value Estimate bit string = 0.827851 / 1 bits
Running Entropic Statistic Estimates bit strings only...
Collision Estimate: X-bar = 2.5002125180640355, sigma-hat = 0.50000057989031521,
 p = 0.53019886351245304
Collision Test Estimate bit string = 0.915395 / 1 bits
Markov Estimate: P_0 = 0.56208999999999998, P_1 = 0.43791000000000002,
P_0,0 = 0.55444778318024368, P_0,1 = 0.44555221681975632,
P_1,0 = 0.57189833527437139, P_1,1 = 0.42810166472562861, p_max = 1.6600019046178776e-33
Markov Test Estimate bit string = 0.850722 / 1 bits
Compression Estimate: X-bar = 5.0951981486367348,sigma-hat = 1.0419214150315916,
 p = 0.095877239689073357
Compression Test Estimate bit string = 0.563778 / 1 bits
Running Tuple Estimates...
t-Tuple Estimate: t = 17, p-hat_max = 0.56208999999999998, p_u = 0.56336794645493438
LRS Estimate: u = 18, v = 44, p-hat = 0, p_u = 0
T-Tuple Test Estimate bit string = 0.827851 / 1 bits
LRS Test Estimate bit string = inf / 1 bits
Running Predictor Estimates...
MultiMCW Prediction Estimate: N = 999937, Pglobal' = 0.56283653552491153
C = 561523 Plocal can't affect result r = 28
Multi Most Common in Window MultiMCW Prediction Test Estimate
bit string = 0.829212 / 1 bits
Lag Prediction Estimate: N = 999999, Pglobal' = 0.56239182046105296
 C = 561113 Plocal can't affect result r = 29
Lag Prediction Test Estimate bit string = 0.830352 / 1 bits
MultiMMC Prediction Estimate: N = 999998, Pglobal' = 0.56334407956030508
```

```
 C = 562065 Plocal can't affect result r = 28
Multi Markov Model with Counting MultiMMC Prediction Test Estimate
bit string = 0.827912 / 1 bits
LZ78Y Prediction Estimate: N = 999983, Pglobal' = 0.56336151481546537
 C = 562074 Plocal can't affect result r = 28
LZ78Y Prediction Test Estimate bit string = 0.827867 / 1 bits
H_bitstring: 0.563778
```

A.2 Linux

A.2.1 Linux

Meranie

```
opening file: 'C:\Users\Martin\Desktop\GenData\Ubuntu.bin'
Number of Binary Symbols: 1000000
Symbol alphabet consists of 2 unique symbols
Running non-IID tests...
Running Most Common Value Estimate...
MCV Estimate: mode = 561772, p-hat = 0.56177200000000005, p_u = 0.56305004869939212
Most Common Value Estimate bit string = 0.828665 / 1 bits
Running Entropic Statistic Estimates bit strings only...
Collision Estimate: X-bar = 2.5014195555978018, sigma-hat = 0.49999861021138581,
p = 0.51756969938077191
Collision Test Estimate bit string = 0.950175 / 1 bits
Markov Estimate: P_0 = 0.56177200000000005, P_1 = 0.43822799999999995,
P_0,0 = 0.55392579195830338, P_0,1 = 0.44607420804169662, P_1,0 = 0.57182921180118984,
P_1,1 = 0.42817078819881016, p_max = 1.4720130995050143e-33
Markov Test Estimate bit string = 0.852077 / 1 bits
Compression Estimate: X-bar = 5.0957853549326453,sigma-hat = 1.0449185817475659,
 p = 0.095649160834697433
Compression Test Estimate bit string = 0.564351 / 1 bits
Running Tuple Estimates...
t-Tuple Estimate: t = 17, p-hat_max = 0.56177200000000005, p_u = 0.56305004869939212
LRS Estimate: u = 18, v = 40, p-hat = 0, p_u = 0
T-Tuple Test Estimate bit string = 0.828665 / 1 bits
LRS Test Estimate bit string = inf / 1 bits
Running Predictor Estimates...
MultiMCW Prediction Estimate: N = 999937, Pglobal' = 0.56251761710740211
C = 561204 Plocal can't affect result r = 19
```

```
Multi Most Common in Window MultiMCW Prediction Test Estimate
bit string = 0.830030 / 1 bits
Lag Prediction Estimate: N = 999999, Pglobal' = 0.56342049228515023
C = 562142 Plocal can't affect result r = 23
Lag Prediction Test Estimate bit string = 0.827716 / 1 bits
MultiMMC Prediction Estimate: N = 999998, Pglobal' = 0.5630421760313975
C = 561763 Plocal can't affect result r = 19
Multi Markov Model with Counting MultiMMC Prediction Test Estimate
 bit string = 0.828685 / 1 bits
LZ78Y Prediction Estimate: N = 999983, Pglobal' = 0.56305060951988573
C = 561763 Plocal can't affect result r = 19
LZ78Y Prediction Test Estimate bit string = 0.828663 / 1 bits
H_bitstring: 0.564351
```

A.2.2 LinuxAWS

Meranie

```
opening file: 'C:\Users\Martin\Desktop\GenData\Ubuntu_AWS.bin'
Number of Binary Symbols: 1000000
Symbol alphabet consists of 2 unique symbols
Running non-IID tests...
Running Most Common Value Estimate...
MCV Estimate: mode = 562191, p-hat = 0.562191, p_u = 0.56346891386949438
Most Common Value Estimate bit string = 0.827592 / 1 bits
Running Entropic Statistic Estimates bit strings only...
Collision Estimate: X-bar = 2.4999250022499324, sigma-hat = 0.50000061935775308,
 p = 0.53249108795793632
Collision Test Estimate bit string = 0.909171 / 1 bits
Markov Estimate: P_0 = 0.562191, P_1 = 0.437809, P_0,0 = 0.55523933189846852,
P_0,1 = 0.44476066810153148, P_1,0 = 0.5711166284841106,
P_1,1 = 0.4288833715158894, p_max = 1.9900876595133069e-33
Markov Test Estimate bit string = 0.848678 / 1 bits
Compression Estimate: X-bar = 5.0935767311961229,sigma-hat = 1.046923760174453,
 p = 0.096539063290617455
Compression Test Estimate bit string = 0.562124 / 1 bits
Running Tuple Estimates...
t-Tuple Estimate: t = 17, p-hat_max = 0.562191, p_u = 0.56346891386949438
LRS Estimate: u = 18, v = 40, p-hat = 0, p_u = 0
T-Tuple Test Estimate bit string = 0.827592 / 1 bits
LRS Test Estimate bit string = inf / 1 bits
```

```
Running Predictor Estimates...
MultiMCW Prediction Estimate: N = 999937, Pglobal' = 0.56292051395314824
C = 561607 Plocal can't affect result r = 21
Multi Most Common in Window MultiMCW Prediction Test Estimate
 bit string = 0.828997 / 1 bits
Lag Prediction Estimate: N = 999999, Pglobal' = 0.56310459365995558
 C = 561826 Plocal can't affect result r = 26
Lag Prediction Test Estimate bit string = 0.828525 / 1 bits
MultiMMC Prediction Estimate: N = 999998, Pglobal' = 0.5634630414147791
C = 562184 Plocal can't affect result r = 21
Multi Markov Model with Counting MultiMMC Prediction Test Estimate
bit string = 0.827607 / 1 bits
LZ78Y Prediction Estimate: N = 999983, Pglobal' = 0.56346148425552134
C = 562174 Plocal can't affect result r = 21
LZ78Y Prediction Test Estimate bit string = 0.827611 / 1 bits
H_bitstring: 0.562124
```

A.2.3 LinuxAzure

Meranie

```
opening file: 'C:\Users\Martin\Desktop\GenData\Ubuntu_Azure.bin'
Number of Binary Symbols: 1000000
Symbol alphabet consists of 2 unique symbols
Running non-IID tests...
Running Most Common Value Estimate...
MCV Estimate: mode = 562474, p-hat = 0.56247400000000003, p_u = 0.56375182227917897
Most Common Value Estimate bit string = 0.826868 / 1 bits
Running Entropic Statistic Estimates bit strings only...
Collision Estimate: X-bar = 2.5001350074254085, sigma-hat = 0.50000060680852065,
p = 0.530833609214028
Collision Test Estimate bit string = 0.913668 / 1 bits
Markov Estimate: P_0 = 0.56247400000000003, P_1 = 0.43752599999999997,
P_0,0 = 0.55558479147480599, P_0,1 = 0.44441520852519401,
P_1,0 = 0.5713296383063825, P_1,1 = 0.4286703616936175, p_max = 2.1547493660463188e-33
Markov Test Estimate bit string = 0.847782 / 1 bits
Compression Estimate: X-bar = 5.0910593340925825,sigma-hat = 1.0463510452628035,
p = 0.097540749578755137
Compression Test Estimate bit string = 0.559642 / 1 bits
Running Tuple Estimates...
t-Tuple Estimate: t = 17, p-hat_max = 0.56247400000000003, p_u = 0.56375182227917897
```

```
LRS Estimate: u = 18, v = 48, p-hat = 0, p_u = 0
T-Tuple Test Estimate bit string = 0.826868 / 1 bits
LRS Test Estimate bit string = inf / 1 bits
Running Predictor Estimates...
MultiMCW Prediction Estimate: N = 999937, Pglobal' = 0.56318744513869834
C = 561874 Plocal can't affect result r = 21
Multi Most Common in Window MultiMCW Prediction Test Estimate
bit string = 0.828313 / 1 bits
Lag Prediction Estimate: N = 999999, Pglobal' = 0.56235383247342852 C = 561075
Plocal can't affect result r = 25
Lag Prediction Test Estimate bit string = 0.830450 / 1 bits
MultiMMC Prediction Estimate: N = 999998, Pglobal' = 0.56373795297764662
C = 562459 Plocal can't affect result r = 21
Multi Markov Model with Counting MultiMMC Prediction Test Estimate
bit string = 0.826903 / 1 bits
LZ78Y Prediction Estimate: N = 999983, Pglobal' = 0.56374239809975879
C = 562455 Plocal can't affect result r = 21
LZ78Y Prediction Test Estimate bit string = 0.826892 / 1 bits
H_bitstring: 0.559642
```

B Obsah priloženého CD

Do systému bol nahratý textový súbor s plným odkazom na GDrive autora keďže súbory presahovali limit stanovený informačným systémom VUT. Skrátený odkaz je uvedený tu URL: <`http://bit.do/selingadiplomka` >

V adresári sa nachádzajú v archívoch samostatne spustiteľné `.exe` súbory pre platformu Windows,zdrojové kódy ku všetkým vytvoreným program, súbor testovacích dát od NIST, vygenerované dáta autorom a výsledky testovania.

- `/` ..... Koreňový adresár
  - `Výsledky` ..... Výsledky testovania
    - `Ubuntu_vysledky.txt`
    - `UbuntuAWS_vysledky.txt`
    - `UbuntuAZURE_vysledky.txt`
    - `Windows_vysledky.txt`
    - `WindowsAWS_vysledky.txt`
    - `WindowsAZURE_vysledky.txt`
  - `TestDataNIST` ..... Súbor testovacích dát od NIST
    - `rand1.bin`
    - `rand4.bin.`
    - `rand8.bin`
    - `randData.bin`
    - `randData1.bin`
    - `truerand_1bit.bin`
    - `truerand_4bit.bin`
    - `truerand_8bit.bin`
    - `urand.bin`
  - `GenData` ..... Dáta vygenerované autorom
    - `Ubuntu_vysledky.bin`
    - `UbuntuAWS_vysledky.bin`
    - `UbuntuAZURE_vysledky.bin`
    - `Windows_vysledky.bin`
    - `WindowsAWS_vysledky.bin`
    - `WindowsAZURE_vysledky.bin`
  - `Linux_generator.zip_FEKT.pdf` ..... Zdrojový kód linuxového generátora
  - `Executable_Windows_Generator.rar` Archív so spustiteľným súborom generátora
  - `Executable_Evaluation.rar` Archív so spustiteľným SW pre hodnotenie entropie
  - `Evaluation_Source .rar` ..... Zdrojový kód SW pre hodnotenie entropie
  - `Windows_Generator_Source .rar` ..... Zdrojový kód Windows generátora