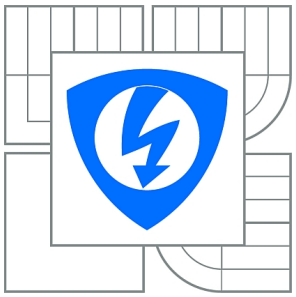

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY


FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS


# ENTROPICKÝ GENERÁTOR NÁHODNÝCH ČÍSEL

ENTROPIC RANDOM NUMBER GENERATOR

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS


AUTOR PRÁCE — MARCEL KUKKO

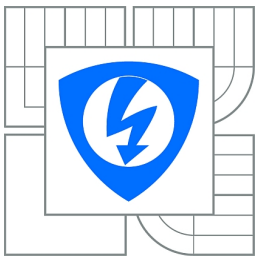
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. KAREL BURDA, CSc.

SUPERVISOR



BRNO 2014



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**Fakulta elektrotechniky a komunikačních technologií**

**Ústav telekomunikací**

# Bakalářská práce

bakalářský studijní obor
**Teleinformatika**

| | | | |
|---|---|---|---|
| ***Student:*** | Marcel Kukko | ***ID:*** | 145940 |
| ***Ročník:*** | 3 | ***Akademický rok:*** | 2013/2014 |

**NÁZEV TÉMATU:**

## Entropický generátor náhodných čísel

**POKYNY PRO VYPRACOVÁNÍ:**

Nastudujte a popište problematiku entropických generátorů náhodných čísel. Na tomto základě navrhněte řešení entropického generátoru náhodných čísel pro běžný osobní počítač. Svůj návrh zdůvodněte, prakticky zrealizujte a vlastnosti navrženého generátoru otestujte. Získané výsledky zhodnoťte.

**DOPORUČENÁ LITERATURA:**

[1] Burda K.: Aplikovaná kryptografie. VUTIUM, Brno, 2013.
[2] Rukhin A., aj.: A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications. NIST SP 800-22. National Institute of Standards and Technology, Gaithersburg, 2010.

| | | | |
|---|---|---|---|
| ***Termín zadání:*** | 10.2.2014 | ***Termín odevzdání:*** | 4.6.2014 |

***Vedoucí práce:*** doc. Ing. Karel Burda, CSc.
***Konzultanti bakalářské práce:***

**doc. Ing. Jiří Mišurec, CSc.**
*Předseda oborové rady*

**UPOZORNĚNÍ:**

## ABSTRAKT

Práce se zaměřuje na problém generování náhodných čísel osobním počítačem. V rámci studie jsou popsány nejpoužívanější typy generátorů náhodné posloupnosti a způsoby jejich testování. Druhá část práce se zaměřuje na návrh entropického generátoru a provedení analýzy jeho testování.

## KLÍČOVÁ SLOVA

GNP, GPNP, entropie, statistický test

## ABSTRACT

The paper is focused on the random number generation by personal computer. It describes the most widely used types of random sequence generators and methods for their testing. The second part is focused on the design of the generator and an analysis of its testing.

## KEYWORDS

RNG, PRNG, entropy, statictical test

KUKKO, Marcel *Entropický generátor náhodných čísel*: bakalářská práce. BRNO: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 39 s. Vedoucí práce byl doc. Ing. Karel Burda, CSc.

# PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Entropický generátor náhodných čísel“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

BRNO ..... .....

(podpis autora)

## PODĚKOVÁNÍ

Děkuji vedoucímu práce panu doc. Ing. Karlu Burdovi, CSc. za velmi užitečnou metodickou pomoc a cenné rady při zpracování bakalářské práce.


BRNO . . . . . .

. . . . . .

(podpis autora)

# OBSAH

# SEZNAM OBRÁZKŮ

# SEZNAM TABULEK

# ÚVOD

Bakalářská práce se zaměřuje na problematiku náhodných čísel a jejich generování pomocí osobního počítače. Jedná se o aktuální téma, neboť generátory náhodných čísel se využívají v řadě běžně používaných počítačových programů, či hrách. Důležitou roli pak generátory hrají při tvorbě kryptografických klíčů. Mnoho kryptografických algoritmů během řady let jejich využívání potvrdilo své přednosti při odolávání útokům. Některé algoritmy využívají pro svou činnost výpočet složitých diskrétních logaritmů, či součinu dostatečně velkých prvočísel. Nicméně společným jmenovatelem pro všechny kryptografické algoritmy je používání generátoru náhodných posloupností. A ten také bývá achilovou patou těchto algoritmů.

Pokud lze odhadnout klíč, používaný algoritmem, vešeré zabezpečení poskytované tímto algoritmem ztrácí smysl. V případě, že utočník zjistí, že při generování klíče je použit predikovatelný zdroj náhodných čísel, výrazně se tím zvýší šance na prolomení tohoto klíče. Problém nastává právě při generování náhodného klíče pomocí počítače, který je už ve své podstatě predikovatelný. John von Neumann v roce 1951 tento problém popsal, když řekl: „Každý, kdo používá aritmetiku ke generování náhodných čísel, je hříšníkem“ [11]. Je tedy důležité, aby použitý generátor splňoval i jisté statistické vlastnosti. Z tohoto důvodu je práce zaměřena také na způsob testování generátorů náhodných čísel. Nedílnou součástí práce je také návrh a implementace vlastního programu, sloužícího pro generování náhodných čísel.

# 1 GENEROVÁNÍ NÁHODNÝCH POSLOUPNOSTÍ

Může se zdát poněkud zvláštní, že počítač, jedno z nejpřesnějších a nejdeterminističtějších zařízení sestrojených člověkem, má být schopno generovat náhodná čísla. Může se to jevit dokonce jako nemožné. Vždyť každý program vytváří výstupy, které jsou plně predikovatelné a tedy nejsou náhodné.

Nicméně osobní počítače používají běžně při své činnosti generátory náhodných čísel. Termín, který se pro tyto generátory používá je generátor pseudonáhoných čísel. Právě slovíčko „pseudo“ vyjadřuje skutečnost, že produkovaný výstup takového generátoru je deterministický a není tedy zcela náhodný.

Definice, která i když ne zcela dokonale vystihuje podstatu počítačového generátorů náhodných čísel by mohla znít asi takto. Deterministický program produkující náhodnou sekvenci čísel se musí lišit a být statisticky nekorelovaný s programem využívajícím jeho výstup [9]. Jinými slovy tedy dva různé generátory náhodných čísel musí produkovat statisticky stejné výsledky. Pokud tomu tak není, tak alespoň jeden z nich je špatný generátor.

Způsob jakým je generátor náhodných čísel implementován zcela závisí na pohledu programátora. Proces náhodný pro jednu aplikaci nemusí být dostatečně náhodný pro aplikaci druhou. Ač se zdá, že tento problém nelze kvantifikovat, existuje řada statistických testů, které zvládnou prověřit vlastnosti generátoru. Dobrý generátor by měl projít všemi testy. V případě, že tomu tak není a generátor v některém z testů selže, musí uživatel uvážit do jaké míry je testované hledisko podstatné a zda vůbec lze tento generátor použít.

V zásadě rozlišujeme tři typy generátorů náhodné posloupnosti. Jedná se o:

- Generátor náhodných posloupností
- Generátor pseudonáhodných posloupností
- Entropický generátor náhodných posloupností

## 1.1 Generátor náhodných posloupností

U GNP[1] se ke generování náhodné posloupnosti využívá fyzikálních procesů, které obsahují šum. Ten je dále vzorkován a uložen jako náhodná posloupnost jedniček a nul. Aby tento systém správně fungoval, musí být zabezpečeno několik požadavků. Šum musí být nahodný a těžce odhadnutelný. Takovými zdroji jsou zejména fyzikální procesy jako změna teploty, rozpad radioaktivních jader, a elektromagnetický šum. Tyto jevy jsou závislé na tolika faktorech, že je prakticky nemožné je zcela

[1] generátor náhodných posloupností

odhadnout. Častokrát se využívá například teplotní šum rezistorů, tzv. Johnsonův šum. Tyto zdroje jsou sice dostatečně náhodné, ale signály jsou velice slabé a je potřeba je zesílit před vlastním navzorkováním [9]. Navíc pro praktické využití je rychlost generování pomalá. Posloupnost bitů odvozená z náhodného procesu bývá ještě pomocí vhodných algoritmů upravována k odstranění případných statistických závislostí (např. von Neumannův korektor) a případně i statisticky kontrolována na náhodnost [4].

## 1.2 Generátor pseudonáhodných posloupností

GPNP[2] je deterministický program vytvářející řetězce náhodně uspořádaných bitů, které jsou při vhodném nastavení dostatečně odolné i pro použití v kryptografii. Generátory vyžadují pro svou činnost inicializační semínko, náhodné číslo, od kterého se dále odvíjí rozvoj pseudonáhodné řady. Výhoda těchto generátorů je jejich snadná implementace a rychlost generování posloupnosti. Mezi nevýhody například patří perioda opakování posloupnosti, nebo nerovnoměrnost rozdělení pro velké množství generovaných čísel. Navíc požadavek na náhodné inicializační semínko nás přivede opět k původnímu problému získání náhodného čísla. Příkladem takového generátoru může být často využívaný lineární kongruentní generátor, který je definován vstahem 1.1 [13].

$$x_i = (ax_i + c)\mathrm{mod}m \tag{1.1}$$

Operace *mod* značí zbytek po celočíselném dělení, $a$,$c$ a $m$ jsou vhodně zvolené konstanty. Prvotní hodnota $x_0$ je inicializační semínko. Generátor je schopen generovat celá čísla v rozsahu $0 \leq x_i \leq m$, s maximální periodou opakování $m$. Existuje celá řada nastavení konstant $a$, $c$ a $m$ pro tento generátor, který se využívá jako základní fuknce generování náhodných čísel ve většině programovacích jazyků.

## 1.3 Entropický generátor náhodných posloupností

Posledním typem generátoru je entropický generátor náhodných posloupností. Pojem entropie se používá pro veličinu popisující míru neurčitosti systému, který se může s pravděpodobnostmi $p_1$ až $p_N$ nacházet v $N$ různých stavech $X_1$ až $X_N$. Entropie stavu s hodnotou $X$ je definována vztahem 1.2 [4].

$$E_x = (-\log_2 p_x) \tag{1.2}$$

Entopický generátor je tvořen kombinací GNP a GPNP. GNP zde slouží jako zdroj pro inicializační semínko. Základem GNP je právě zdroj entropie. Jako zdroj

[2]generátor pseudonáhodných posloupností

entropie se v praxi používájí například události spojené s psaním na klávesnici, nahodilým pohybem myši, či náhodné bity na výstupu zvukové karty. GNP v sobě akumuluje posloupnost z jednotlivých zdrojů entropie a po dosažení stanoveného objemu je posloupnost přenesena do GPNP, kde slouží jako inicializační semínko. GPNP je tedy základem entropického generátoru, který produkuje výslednou posloupnost. GPNP na základě inicializačního semínka vyprodukuje výslednou posloupnost o maximální stanovené délce. Během doby generování GPNP se v akumulátoru GNP vytvoří nové inicializační semínko a celý proces se cyklicky opakuje. Blokové schéma takového generátoru je zachyceno na obrázku 3.1.

Příkladem entropického generátoru může být generátor kryptografické knihovny BSAFE [3]. Tento generátor využíva kryptografických hešovacích funkcí MD5 nebo SHA1. Generátor je dále charakterizován stavem $S_j$ podle vzorce 1.3

$$S_j = H(S_{j-1}||X_j), \tag{1.3}$$

kde operace $||$ vyjadřuje zřetězení dvou posloupností, $X_j$ je inicializační semínko a $H$ je operace daná výběrem hashovací funkce. Výstupní posloupnost tohoto generátoru je pak dána fukncí 1.4.

$$Y_{j,i} = H(S_{j,i}), S_{j,i} = (S_j + C \cdot i) \mathrm{mod} 2^L \tag{1.4}$$

Entropické generátory se jeví jako vhodná cesta vývoje v problematice generování náhodných čísel pro účely kryptografie [4]. Výše uvedený generátor však využívá hešovací funkce MD5 a SHA1, které se v dnešní době již nedoporučuje používat.

# 2 TESTOVÁNÍ GENERÁTORŮ NÁHODNÝCH ČÍSEL

Existuje celá řada testů, které lze aplikovat na generátory náhodných čísel a rozhodnout tak o tom, zda se je generovaná sekvence náhodná. Výstup statistického testu, aplikovaný na náhodnou sekvenci, je a priori známý a lze jej popsat pomocí pravděpodobnosti. Existuje nespočet testů, které hledají opakující se vzory v sekvenci a rozhodnou tak o tom, zda je opravdu náhodná. Přestože však existuje tolik testů, žádný z nich není testem kompletním, který by rozhodl o náhodnosti sekvence ze všech hledisek.

NIST[1] vyvinul soubor 15ti statistických testů, které slouží k rozhodnutí o náhodnosti binární posloupnosti generované harwarovými či softwarovými generátory náhodných čísel. Tyto testy se zaměřují na aspekty, které mohou nastat u nenahodilé sekvence. Jedná se o tyto testy[13]:

- Test četnosti
- Test četnosti v bloku
- Test shodné série
- Test nejdelší série v jednom bloku
- Test nezávislosti
- Test spektra
- Test nepřekrývajících se šablon
- Test překrývajících se šablon
- Maurerův „univerzální statistický“ test
- Test lineární zpětné vazby
- Test četnosti série
- Test entropie
- Test kumulativních součtů
- Test náhodných návštěv
- Test variací náhodných návštěv

## 2.1 Test četnosti

Tento test je zaměřen na celkový počet vygenerovaných jedniček a nul v sekvenci. Cílem tohoto testu je určit, zda počet jedniček a nul odpovídá předpokládanému počtu očekávaného u náhodné sekvence, tedy je přibližně shodný. Pro výpočet statistiky se v tomto testu využívá normální rozdělení. Všechny následné testy jsou závisle na použití tohoto testu.

[1] National Institute of Standards and Technology

## 2.2 Test četnosti v bloku

Tento test se zaměřuje na podíl jedniček a nul v blocích o velikosti $M$ bitů. Test rozhodne, zda je četnost jedniček v každém bloku přibližně rovna $M/2$, jak je očekáváno od náhodné sekvence. Referenční rozdělení, které určuje míru shody pozorovaného podílu v bloku délky $M$ s očekávaným podílem (1/2) je rozdělení pravděpodobnosti $\chi^2$.

## 2.3 Test shodné série

Test se zaměřuje na celkový počet shodných sérií v sekvenci, kde shodnou sérii můžeme definovat jako nepřerušenou řadu stejných bitů. Shodná série délky $k$ obsahuje přesně $k$ shodných bitů a je ohraničena na začátku i na konci bitem opačné hodnoty. Cílem tohoto testu je rozhodnout, zda počet shodných sérií jedniček a nul různých délek $k$ je očekávatelný od náhodné sekvence. Jinými slovy tedy tento test rozhodne, zda frekvence změny jedniček a nul je příliš vysoká, či nízká.

## 2.4 Test nejdelší série v jednom bloku

Test je zaměřen na nejdelší sérii jedniček z bloků o velikosti $M$ bitů. Cílem tohoto testu je určit, zda délka nejdélší série jedniček v testovací sekvenci je očekávatelná u náhodné sekvence. Neočekávané hodnoty délky nejdelší série jedniček indikují také neočekávané hodnoty nejdelší série nul, a proto stačí tento test provádět pouze pro jedničky. Pro výpočet statistiky se zde využívá rozdělení $\chi^2$.

## 2.5 Test nezávislosti

Test je zaměřen na určení hodnosti submatic v rámci celé sekvence. Sekvenci délky $n$ rozdělíme na matice $M \cdot Q$, kde počet těchto matic bude roven $N = \frac{n}{M \cdot Q}$. Přebytečné bity neuvažujeme. Cílem tohoto testu je zhodnotit lineární závislosti mezi řadami konstantní délky v rámci celé vstupní sekvence.

## 2.6 Test spektra

Tento test využívá diskrétní Fourierovy transformace pro vypočtení spektra sekvence. V rámci tohoto testu se prověřují ve vypočteném spektru periodicity, které by poukazovaly na odchylky od náhodné sekvence. Tedy zda jsou ve spektru všechny složky zastoupeny stejně s maximální odchylkou 5 %.

## 2.7 Test nepřekrývajících se šablon

Test je zaměřen na počet výskytů předem stanovených sekvencí bitů v celkové generované sekvenci. Cílem tohoto testu je odhalit generátory, které produkují příliš mnoho stejných aperiodických sekvencí. Používá se zde okno délky $m$ bitů, které odpovídá hledané šabloně délky $m$. Pokud bity se v okně hledaná šablona nenachází, okno se posune o jeden bit dále. Pokud však je hledaná šablona nalezena, okno se posune na bit následující po nalezené šabloně.

## 2.8 Test překrývajících se šablon

Tento test, stejně jako test nepřekrývajících se šablon, zkoumá počet výskytů předem stanovených sekvencí v celkové sekvenci. Rozdíl od předešlého testu je v tom, že v případě, že je hledaná sekvence nalezena, okno se posouvá pouze o jeden bit dále.

## 2.9 Maurerův „univerzální statistický“ test

Cílem tohoto testu je určit, zda lze sekvenci zkomprimovat tak, aniž by tím došlo ke ztrátě informace. Pokud lze sekvenci značným způsobem takto komprimovat, je považována za náhodnou.

## 2.10 Test lineární zpětné vazby

Test je zaměřen na délku posuvného registru s lineární zpětnou vazbou[2]. Cílem testu je určit, zda sekvence je dostatečně složitá na to, abychom ji mohli označit za náhodnou. Náhodné sekvence jsou charakterizovány větší délkou bitů LFSR. Pokud je tedy výsledná bitová délka LSFR u zkoumané sekvence příliš krátká, nejedná se o náhodnou sekvenci.

## 2.11 Test četnosti série

Cílem tohoto testu je určit, zda počet výskytů $2^m$ $m$bitových vzorů je očekávatelný od náhodné sekvence. Náhodná čísla mají rovnoměrné rozložení, tedy každý $m$bitový vzor má stejnou pravděpodobnost výskytu jako všechny ostatní $m$bitové vzory. Pokud budeme uvažovat $m = 1$, pak je tento test shodný s testem četnosti z kapitoly 2.1.

[2]Linear Feedback Shift Register, LFSR

## 2.12 Test entropie

Stejně jako u testu četnosti série z kapitoly 2.11, je i tento test zaměřen na výskyt $m$bitových vzorů v sekvenci. V rámci tohoto testu se však sleduje četnost vzorů sousedních délek, tedy $m$ a $m + 1$.

## 2.13 Test kumulativních součtů

Test je zaměřen na střední vzdálenost od počátečního bodu průběhu náhodné procházky[3], která je definována součtem sousedních bitů v sekvenci. Cílem testu je rozhodnout, zda kumulativní součet úseků testované sekvence je příliš malý, či velký vzhledem k očekávanému chování náhodné sekvence. Tento kumulativní součet se dá považovat za náhodnou procházku. Tedy pro náhodné procesy bude střední vzdálenost od počátečního bodu blízká nule. U procesů nenáhodných však bude dosahovat větších hodnot.

## 2.14 Test náhodných návštěv

Test je zaměřen na počet cyklů, které se během kumulativního součtu náhodné procházky navštíví přesně $K$krát. Kumulativní součet náhodné procházky je odvozen od částečných součtů poté, co je sekvence (0,1) transformována na sekvecni (-1,1). Cylkus náhodné procházky se skládá ze sekvence kroků jednotné, náhodné délky, která začíná a končí v počátku. Cílem tohoto testu je určit, zda počet návštěv jednotlivých stavů v rámci cyklu lze považovat za náhodný. Tento test je ve své podstatě série osmi testů, kde jednotlivé testy jsou prováděny pro stavy: $-4$, $-3$, $-2$, $-1$, $+1$, $+2$, $+3$ a $+4$.

## 2.15 Test variací náhodných návštěv

Tento test se zaměřuje na celkový počet návštěv jednotlivých stavů při kumulativním součtu náhodné procházky. Cílem testu je pak určit odchylku od očekávaného počtu návštěv náhodné sekvence a rozhodnout tak, zda se jedná o náhodnou sekvecni. Test je složen ze série osmnácti testů pro stavy: $-9$, $-8$, ..., $-1$ a $+1$, $+2$, ..., $+9$.

[3] random walk

# 3 NÁVRH ENTROPICKÉHO GENERÁTORU NÁHODNÝCH ČÍSEL

## 3.1 Výběr zdroje entropie

Při generování náhodných čísel pomocí počítače se často využívá klávesnice, nebo myš jako vstup pro generování inicializačního semínka. Tento způsob se uplatňuje řadu let u mnoha systémů. Pokud software vyžaduje náhodné číslo pro inicializační semínko, oznámí uživateli, aby náhodně stiskal klávesy na klávesnici, nebo pohyboval myší. Zaznamenáním těchto vstupních hodnot a času výskytu získal program velice těžce predikovatelné inicializační semínko. Tato metoda má ovšem svá negativa. Prvním problémem je, že uživatel nemusí být přítomen. Počítač může být umístěn v serverovně, nebo na uzavřeném místě, nebo nemusí mít přímo připojenou klávesnici či myš. Dalším problémem je také to, že na počítači může běžet současně více procesů, které vyžadují náhodný vstup. Tato situace si může vyžádat více vstupů, než je uživatel schopen zadat, nebo také více, než by se uživateli líbilo. Tímto se dostáváme k dalšímu problému a to laxnímu přístupu uživatele k zadávání náhodného vstupu. Uživatel totiž může být otráven zadáváním náhodných vstupů a bude mačkat, nebo držet pouze jednu klávesu. Tímto způsobem se znehodnocuje náhodnost celého procesu.

Jinou možností je generovaní náhodného semínka z těžce odhadnutelných dat počítače. Mezi ty spadá například datum a čas, zpracování ID souborů, obsah jednotlivých disků, atd. Tyto data jsou vetšinou ještě mezi sebou kombinována tak, aby byla zvýšena náhodnost. Problém u této metody je ten, že pro určení náhodného vstupu jsou používána data, která jsou zjistitelná. Pokud se útočník naboural do počítače, dostane se mu přístupu ke vstupním hodnotám, které se využívají ke generování. Druhým problémem je to, že náhodnost bývá zabezpečena známými metodami zpracování dat. Pokud jsou tyto metody veřejně dostupné, útočníkovi stačí zadat vstupní hodnoty a dostane se ke generovaným „náhodným“ hodnotám.

Jako další možností se pak jeví využití náhodných fyzikálních procesů, jako například šum na vstupu zvukové karty. Využití takového zdroje entropie se zdá jako vhodné řešení, jelikož se jedná o zdroj náhodného procesu. Navíc výstup takového zdroje se neopakuje a není zcela závislý na vstupu uživatele.

## 3.2 Výběr generátoru pseudonáhodné posloupnosti

Při volbě vhodného generátoru pseudonáhodné posloupnosti je potřeba hledět na několik faktorů. Při výběru GPNP bychom se měli vyvarovat několika bodům[7].

- Nepoužívat generátory založené na LKG[1]. Tento bod bude ještě rozebrán dále.
- Nepoužívat generátory s periodou menší než $\approx 2^{64} \approx 2 \cdot 10^{19}$.
- Nepoužívat generátory, u kterých je známo, že bity nižšího řádu nemají náhodnou posloupnost (LKG).
- Nepoužívat generátory, které jsou v knihovnách programovacích jazyků. Např. funkce jako `rand`, nebo `srand` jsou většinou nevhodně navrženy a nemají žádnou standardizovanou implementaci.

Nyní se podíváme podrobně na rovnici 1.1 LKG a vyšetříme, proč není vhodné využívat zabudované funkce pro generování náhodných čísel v programovacích jazycích. U tohoto generátoru se výstupní posloupnost bude opakovat s periodou maximálně $m$ a to pouze při vhodně zvolených konstantách. LKG se hojně využíval již v padesátých letech minulého století [11]. První známky o nevhodnosti tohoto generátoru se oběvovali až v letech šedesátých. Pokud náhodná čísla použijeme pro vyplnění $k$-rozměrného prostoru, měla by jej při dostatečném počtu vyplnit zcela. U LKG tomu tak však není a čísla se rozloží do $(k-1)$rozměrných ploch. Těchto ploch bude maximálně $m^{1/k}$. Například pokud budeme uvažovat trojrozměrný prostor, tak při klasické volbě $m$ jako maximální hodnoty čísla integer na počítači, tedy většinou $m = 2^{32}$ bude vytvořeno zhruba 1600 ploch, na kterých budou rozprostřena všechna generovaná čísla. Nevhodnou volbou $a$ a $m$ však můžeme dostat mnohem méně těchto ploch, jako napříkal u IBM mainframových počítačů nechvalně známá funkce `RANDU` s parametry $a = 65539$ a $m = 2^{31}$. U této funkce se dalo vygenerovat čísla, která byla rozložena pouze v 11 plochách.

Další nevýhodou tohoto generátoru je to, že pokud je $m$ zvoleno jako mocnina 2, tak bity nižšího řádu nejsou náhodné. Nejméně významný bit (LSB) má periodu maximálně 2, druhý nejméně významný bit pak má periodu maximálně 4, třetí maximálně 8 atd. Ale pokud se $m$ nezvolí jako mocnina dvou, bude pro vykonávání fukncí násobení a modulo potřeba v počítači podpora dvojnásobné přesnosti.

Ozázkou je, jak tedy zvolit vhodný GPNP. Takový generátor by měl kombinovat alespoň dvě (ideálně nekorelované) metody výpočtu. Tyto metody se musí vypočítávat nezávisle na sobě. Pro kombinaci těchto metod by se měla použít jednoduchá operace, která neznehodnotí náhodnost operandů.

## Výběr algoritmu

Jako vhodný algoritmus pro generátor pseudonáhodné posloupnosti počítačem se jeví např. algoritmus zmiňovaný v pramenu [11]. Tento algoritmus je navržen tak,

[1] lineární kongruentní generátor

aby splňoval všechny předešlé zásady. Úspěšně také zvládá veškeré testy zmiňované v kapitole 2. O kvalitě algoritmu také jistě svědčí to, že jeho autoři nabídli pozornému čtenáři, který dokáže, že algoritmus nevyhovuje některému z běžně používaných statistických testů odměnu ve výši 1000$. Tato odměna nebyla doposud nikomu vyplacena. Algoritmus je definován v hlavičkovém souboru `ran.h` a generuje náhodná čísla s periodou $\approx 3,138 \cdot 10^{57}$. Následuje krátký výpis kódu algoritmu.

```
struct Ran {
 Ullong u,v,w;
 Ran(Ullong j) : v(4101842887655102017LL), w(1) {
        u = j ^ v; int64();
        v = u; int64();
        w = v; int64();
 }
 inline Ullong int64() {
        u = u * 2862933555777941757LL + 7046029254386353087LL;
        v ^= v >> 17; v ^= v << 31; v ^= v >> 8;
        w = 4294957665U * (w & 0xffffffff) + (w >> 32);
        Ullong x = u ^ (u << 21); x ^= x >> 35; x ^= x << 4;
        return (x + v) ^ w;
 }
 inline Doub doub() { return 5.42101086242752217E-20 * int64(); }
 inline Uint int32() { return (Uint)int64(); }

};
```

Základním předpokladem u tohoto generátoru je to, že se jedná o objekt, v tomto případě strukturu. Generátor si tak udržuje informaci o vnitřním stavu mezi jednotlivými voláními funkcí.

Konstruktor `Ran()` požaduje vstupní hodnotu, která se stává inicializačním semínkem pro generátor. Tato vstupní hodnota může být jakýmkoliv číslem datového formátu integer. Jakmile je vytvořena instance generátoru, tak jsme schopni generovat náhodná čísla v různých datových formátech. Pro funkce `int32()`, `int64()` a `doub()` vrací následnou hodnotu náhodného čísla v datovém formátu odpovídajícímu názvu funkce.

Jiným typem vhodného algoritmu se jeví aplikace hešovacích funkcí tak, jako je tomu například u generátoru popsaného v kapitole 1.1. Hešovací funkci chápeme jendosměrnou funkci, která ze vzoru libovolné velikosti vytvoří určitý obraz stanovené délky. Ze získaného obrazu však nelze, nebo lze jen velice obtížně zjistit použitý vzor.

Hešovací funkce se využívají zejména v různých autentizačních kryptosystémech [4], ale lze ji využít i pro generování náhodných čísel. Vstupem takového generátoru je inicializační semínko, na které se aplikuje vybraná hešovací funkce v kombinaci s dalšími metodami výpočtu. Výhodou tohoto principu generování je zejména skutečnost, že výsledný generátor splňuje nejen statistické vlastnosti, ale je také odolný vůči napadení útočníkem. Takový generátor je pak vhodný například pro generování kryptografických klíčů.

## 3.3 Návrh řešení entropického generátoru

Jako zdroj entropie pro navrhovaný generátor jsem zvolil zvukovou kartu. Ta je dnes součástí prakticky všech osobních počítačů. Jedná se tedy o vhodné a laciné řešení. Navíc tento zdroj není závislý na vstupech uživatele, a lze tak zajistit určitou autonomnost generátoru, na rozdíl od zadávání vstupů pomocí klávesnice, či nahodilého pohybu myší. Podle [9] lze využít zvukovou kartu i s připojeným mikrofonem. Osobní počítač, nacházející se v reálném prostředí, je totiž vystavován neustálému působení nahodilých jevů. Kombinací těchto jevů vzniká šum na vstupu mikrofonu, který bude sloužit jako zdroj entropie.

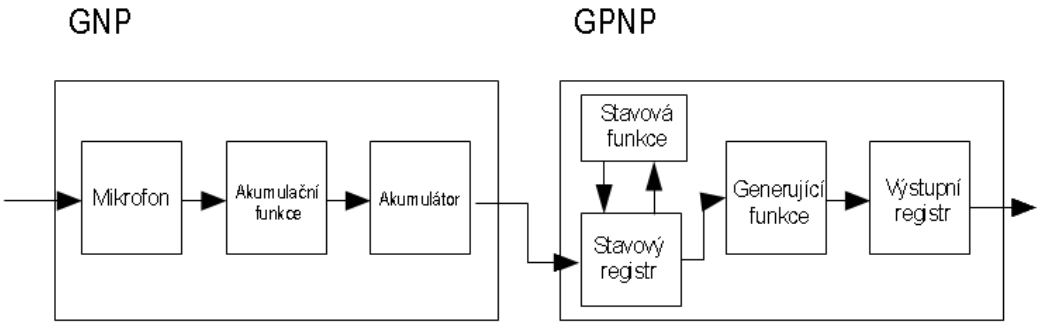
Mikrofon snímá samozřejmě veškeré zvuky z okolí, tedy i takové, které by mohli ovlivnit nahodilost vstupu. Například hluk způsobený otáčkami ventilátorů počítače. Tyto ventilátory mají stálou frekvenci otáčení a způsobují tedy periodicity na vstupu mikrofonu. Z tohoto důvodu bych využil pouze nejméně významný bit vzorkovaného signálu, který bude ovlivněn zejména šumem. K odstranění případných statistických závislostí tohoto zdroje můžeme použít von Neumannův korektor. Tato metoda, kterou ve svých generátorech používá například firma Intel [8] aplikuje dvojici bitů jako vstup funkce a výstupní bit je pak dán tabulkou 3.1.

Tab. 3.1: von Neumannův korektor

| Vstupní bity | Výstupní bit |
|---|---|
| 0,0 | žádný výstup |
| 0,1 | 1 |
| 1,0 | 0 |
| 1,1 | žádný výstup |

Výstupní bity von Neumannova korektoru budou sloužit pro tvorbu inicializačního semínka generátoru pseudonáhodné posloupnosti.

Blokové schéma generátoru je zachyceno na obrázku 3.1. Blok *GNP* obsahuje bloky *Mikrofon*, *Akumulační funkce* a *Akumulátor*. Blok *Mikrofon* je reprezentován




Obr. 3.1: Blokové schéma enropického generátoru [4]

mikrofonem, vstupem a výstupem zvukové karty a von Neumannovým korektorem. Vstupem tohoto bloku je tedy zvuk z reálného prostředí, který je snímán mikrofonem. Po následném zpracování signálu zvukovou kartou dostaneme na výstupu vzorky o stanovené velikosti. Z jednotlivých vzorků se vždy použije pouze nejméně významný bit. Dvojice těchto bitů bude tvořit vstup von Neumannova korektoru. Výstupem korektoru je již náhodná posloupnost bitů. Tyto náhodné bity vstupují do bloku *Akumulační funkce*. Jakmile bude v *Akumulační funkci* dosaženo požadované hodnoty entropie, *Akumulátor* vytvoří z daného počtu bitů inicializační semínko a jako pole bitů jej odešle do *GPNP*.

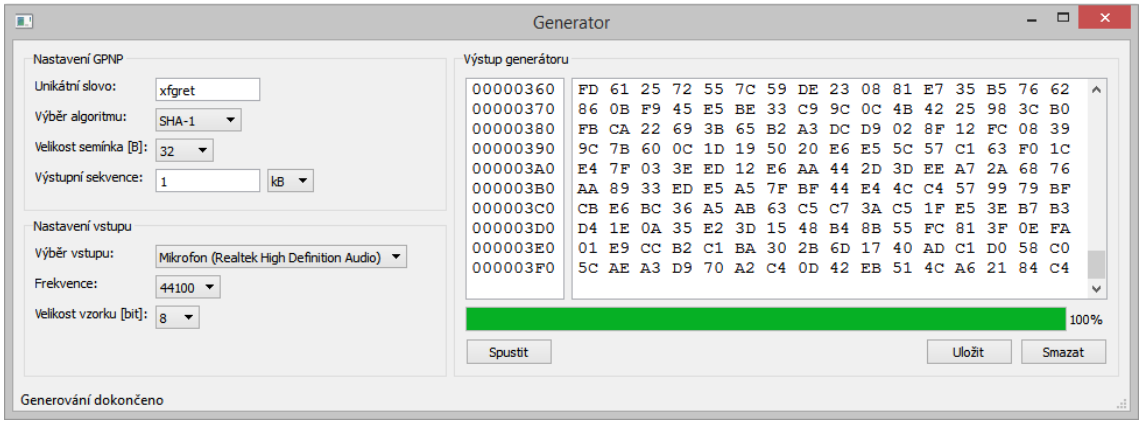
Do bloku *GPNP* tedy vstupuje pole bitů. V bloku stavová fuknce se za pomocí inicializačního semínka provede aktualizace stavu generátoru a generující fuknce vytvoří novou posloupnost čísel. Výstupní registr ukládá generovanou posloupnost do souboru a kontroluje překročení maximálního počtu generovaných bitů Nmax. Po jeho překročení se zastaví načítání mikrofonem a generátor se ukončí.

# 4 REALIZACE NAVRŽENÉHO GENERÁTORU

Pro vlastní naprogramování entropického generátoru jsem využil programovacího jazyku C++ spolu s multiplatformní knihovnou Qt [12]. Tato knihovna je jednou z nejpopulárnějších knihoven pro vytváření programů s grafickým uživatelským rozhraním. Označení multiplatformní je zde zcela na místě, neboť zdrojový kód programu stačí pouze přeložit na jiném operačním systému a je zde plně funkční. Lze jej tedy použít pro všechny běžné operační systémy osobních počítačů, jako jsou MS Windows, Linux a OS X. S novou verzí Qt 5 lze program použít i na mobilních operačních systémech jako Android, iOS a Windows Phone. Pro tuto knihovnu je vytvořeno i vývojové prostředí Qt Creator, které umožňuje rychlým a pohodlným způsobem využít veškeré vymoženosti Qt knihovny.

## 4.1 Uživatelské rozhraní

Uživatelské rozhraní programu je zobrazeno na obrázku 4.1. Celé rozhraní je rozděleno na tři základní celky: *Nastavení GPNP*, *Nastavení vstupu* a *Výstup generátoru*.




Obr. 4.1: Uživatelské rozhraní generátoru

V celku *Nastavení GPNP* se nastavují vstupní parametry vlastního generátoru pseudonáhodné posloupnosti. Je zde možnost zadat unikátní slovo, které se podílí na aktualizaci stavu generátoru viz kapitola 4.2. Dále je zde možnost výběru použité hešovací fuknce (MD4, MD5, SHA-1, SHA-2, SHA-3) s různou délkou výstupu. Maximální velikost semínka je nastavena na 1024 B. Minimální velikost semínka se odvíjí podle vybraného algoritmu tak, aby byla shodná, nebo vetší než velikost výstupu hashovací funkce. Poslední nastavení tohoto celku definuje cekovou velikost

generované posloupnosti v jednotkách B, kB a MB. Maximální velikost generované posloupnosti byla stanovena na 512 MB.

Druhým celkem je celek *Nastavení vstupu*. Na tomto místě se nastavují parametry vstupního zařízení. Jako výběr vstupu tedy slouží veškeré připojené zařízení pro záznam zvuku. Dále je zde možnost nastavení vzorkovací frekvence a velikost vzorku v bitech pro vybrané zařízení.

Největší plochu uživatelského rozhraní zabírá celek *Výstup generátoru*. Značnou plochu zabírají dvě textová pole. Větší z nich zobrazuje výpis generované posloupnosti v hexadecimálním zápisu a s 16 byty na řádek. Do menšího textového pole se pak zapisuje aktuální počet generovaných bytů, opět v hexadecimálním vyjádření. Celek také obsahuje tři tlačítka. Tlačítko *Spustit*, slouží ke spouštění vlastního procesu generování, jehož průběh můžeme sledovat na ukazateli průběhu umístěným pod textovým polem. Tlačitko *Uložit* slouží k uložení generované posloupnosti do souboru ve formě jednotlivých bytů a tlačítko *Smazat* vymaže generovanou posloupnost z paměti.

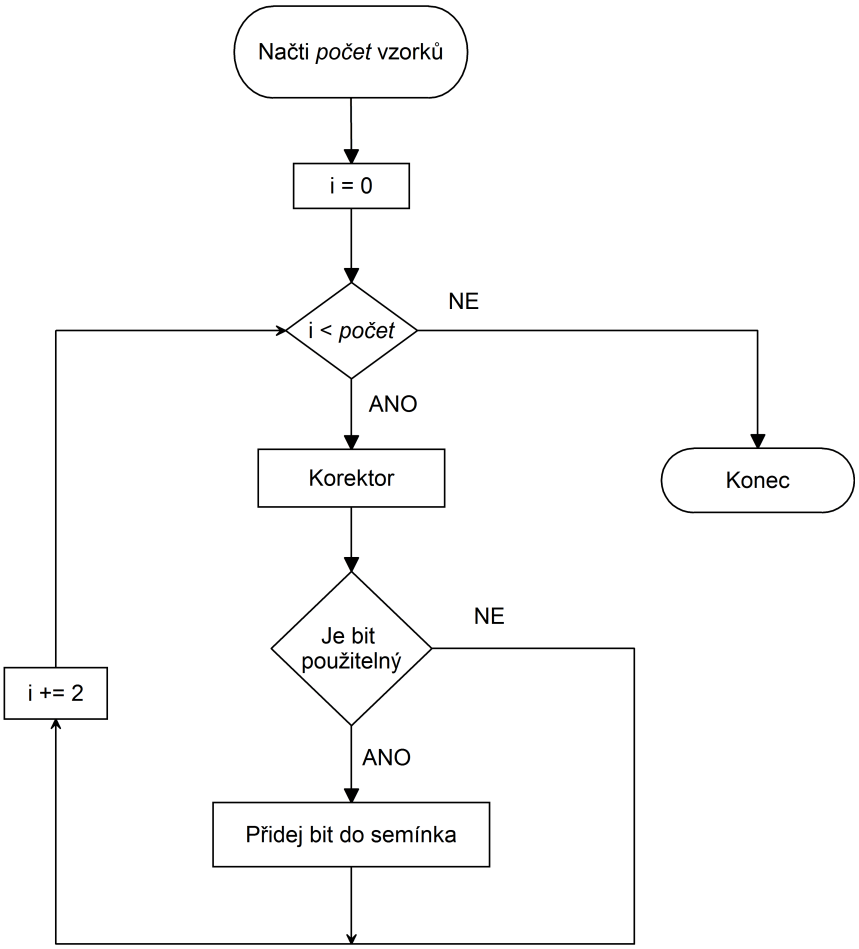
## 4.2 Konstrukce programu

Jeden ze základních prvků multiplatformní knihovny Qt je mechanismus spojování signálů a slotů. Každý objekt v Qt může mít své signály a sloty. Signály, jak už z jejich názvu vyplývá, jsou zprávy, které objekt generuje v případě jeho změny, nebo při nějaké důležité události (typickým příkladem signálu je například kliknutí myší na tlačítko). Slotem pak označujeme klasickou funkci, která pokud je připojena k signálu, je zavolána a umožňuje nějakým způsobem měnit objekt. Signál může volaným slotům předávat také parametry. K jednomu signálu lze připojit i více slotů, nebo další signály.

Základní kostra programu je tvořena třemi třídami. Třídou *Listener*, která ovládá vstupní zvukové zařízení, načítá vzorkovaný signál a vytváří inicializační semínka. Třída *GPNP* má za úkol vlastní generování posloupnosti a třída *MainWidget*, která se stará o tvorbu uživatelského rozhraní a další podpůrné fuknce (např. ukládání do souboru). Komunikace mezi jednotlivýma třídama i v rámci nich probíhá pomocí mechanismu signálů a slotů. Mezi hlavní metody programu patří metoda *readAudioBuffer()* třídy *Listener* a metoda *generate()* třídy *GPNP*, které budou následně podrobněji rozebrány.

Metoda *readAudioBuffer()*, která je zároveň také slotem, je volána vždy, když načítací buffer vstupního zařízení obsahuje vzorky signálu. Tyto vzorky jsou tedy předány do metody, kde probíhá jejich zpracování. Vývojový diagram metody je zobrazen na obrázku 4.2. Jednotlivé vzorky jsou po dvojicích načítány korektorem,

kde se z každého vzorku využije vždy jen nejméně významný bit a zpracuje pomocí metody von Neumann viz. kapitola 3.3. V případě, že výsledkem je náhodný bit, tento bit se zapíše do semínka. Zápis do semínka probíhá tak, že se nejprve z jednotlivých bitů sestaví byte, který se následně ukládá do pole bytů. Po dosažení stanovené velikosti semínka se toto pole v rámci spojení signál – slot přenese jako vstupní parametr metody *generate()*.




Obr. 4.2: Vývojový diagram metody *readAudioBuffer()*

Metoda *generate()* je tedy zavolána s parametrem inicializační semínko. Nejprve se provede aktualizace stavu generátoru. Tato operace probíhá podle vzorce 4.1, kde $S_i$ značí stav generátoru a $x$ je inicializační semínko.

$$S_i = H\left(S_{i-1}||x\right) \tag{4.1}$$

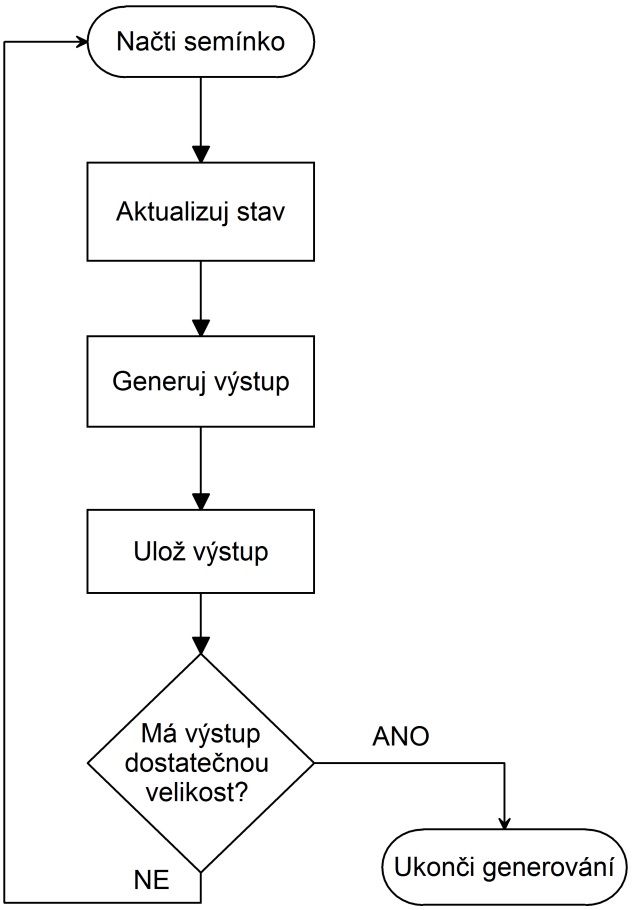
Operace $||$ vyjadřuje zřetězení proměnných a $H()$ vybranou hešovací funkci. Inicializace stavu $S_0$ probíhá při inicializaci celé třídy kdy počáteční stav $S_0 = H(W_0)$,

kde $W_0$ je unikátní slovo z nastavení generátoru reprezentované jako pole bytů. Následující proces s názvem *Generuj výstup* probíhá ve dvou krocích.

$$W_i = W_{i-1} \cdot i \tag{4.2}$$

$$Y_i = H\left(S_i \otimes Wi\right) \tag{4.3}$$

Nejprve se podle vzorce 4.2 vypočte nové unikátní slovo $W_i$. Operace násobení probíhá ovšem nad každým bytem. Následně podle 4.3 proběhne exkluzivní součet a vybraná hešovací funkce a výsledkem je výstupní posloupnost $Y_i$.




Obr. 4.3: Vývojový diagram metody *generate()*

# 5 TESTOVÁNÍ NAVRŽENÉHO GENERÁTORU

K testování navrženého generátoru jsem využil program NIST test suite[10], který používá testy popsané v kapitole 2. Pro úspěšné použití všech testů je potřeba dodržet několik zásad, jako například minimální velikost vstupních dat. Některé testy vyžadují, aby jedna sekvence obsahovala alespoň $10^6$ bitů, další testy pak stanovují minimální počet těchto sekvecní. Z těchto důvodů byla stanovena velikost dat na 16 MB tak, aby bylo možné provést všechny testy v rámci jednoho testování. Tyto data budou použita jako 128 sekvencí po 128 kB. Navrhovaný generátor ukládá data jako řetězec po jednotlivých bytech a data tak splňují i podmínku typu vstupního souboru. Pro hladinu významnosti byla zvolena hodnota $\alpha = 0,01$. Provedl jsem celkem tři měření pro různé vlivy okolního prostředí na generátor. Nastavení generátoru bylo v každém měření shodné podle tabulky 5.1.

Tab. 5.1: Nastavení generátoru

| Parametr | Hodnota |
|---|---|
| Unikátní slovo | xfrget |
| Výběr algorimtu | SHA3-512 |
| Velikost semínka | 64 B |
| Výstupní sekvence | 16 MB |
| Frekvence | 96000 Hz |
| Velikost vzorku | 8 bit |

## 5.1 Interpretace empirických výsledků

Při důkladné analýze zjištěných výsledků můžeme dojít ke třem závěrům:

- Analýza *P-hodnot*[1] nepoukazuje na významné odchýlení od náhodných hodnot.
- Analýza jasně poukazuje na významné odchýlení od náhodných hodnot.
- Analýza není průkazná.

Interpretace dosažených výsledků testování může být různorodá. NIST test suite si pro danou interpretaci zvolili dvě kritéria[13]. Jedná se o podíl úspěšných sekvencí v testech a rovnoměrné rozdělení *P-hodnot*.

[1] *P-hodnota* určuje pravděpodobnost, že testovací kritérium dosáhlo své hodnoty

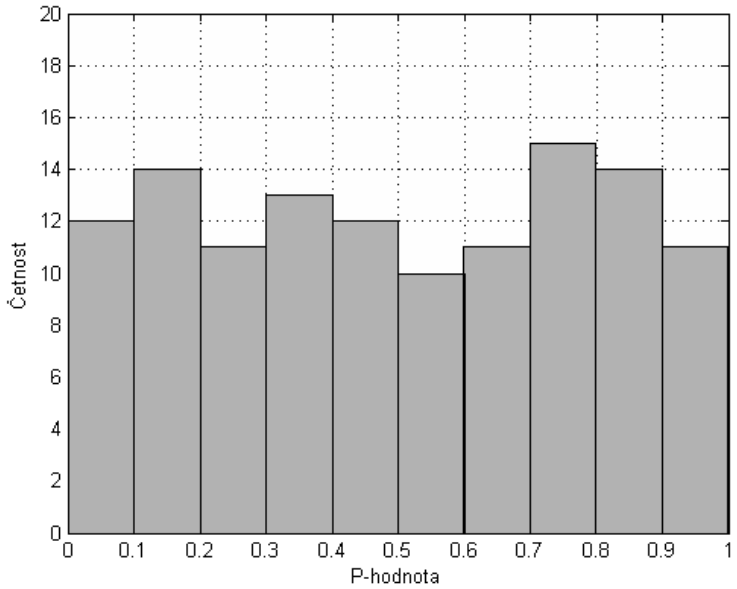
## Podíl úspěšných sekvencí

Pro každý statistický test se vypočítává určitý počet *P-hodnot* podle počtu sekvencí. Pro konstantní hladinu významnosti se očekává, že jisté procento *P-hodnot* bude indikovat, že test nebyl úspěšný. Například při hladině významnosti rovne $\alpha = 0,01$ se očekává, že přibližně 1 % sekvencí v testu neuspěje. Tedy v případě, že testujeme 128 sekvencí ($m = 128$), hladina významnosti $\alpha = 0,01$ a pro 127 sekvencí platí, že *P-hodnota* $\geq 0,01$, tak podíl úspěšných sekvencí bude $127/128 = 0,9922$.

Rozhodnutí určíme pomocí intervalu spolehlivosti jako $\hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}}$, kde $\hat{p} = 1 - \alpha$ a $m$ je počet testovaných sekvecní. Pokud výsledný podíl padne mimo tento interval, jedná se o důkaz, že data nejsou náhodná. Pro uvedený příklad bude tedy interval spolehlivosti roven $0,99 \pm 3\sqrt{\frac{0,99 \cdot 0,01}{128}} = 0,99 \pm 0,026384$ (tedy minimálně 123 úspěšných sekvencí). Výsledek tohoto kritéria je uveden ve sloupci *Úspěšnost* v tabulkách 5.2, 5.3 a 5.4.

## Rovnoměrné rozdělení *P-hodnot*

V rámci analýzy výsledků testování také zkoumáme, zda rozdělení *P-hodnot* je rovnoměrné. Tuto zkutečnost můžeme zkoumat například graficky pomocí histogramu 5.1. Interval hodnot 0 – 1, do kterého spadá velikost *P-hodnot*, rozdělíme na 10 tříd.




Obr. 5.1: Ukázka histogramu

Četnosti *P-hodnot* v jednotlivých třídách jsou pak znázorněny pomocí sloupcových grafů. Zda se jedná o rovnoměrné rozdělení lze zjistit také pomocí aplikace statistiky

$\chi^2$ a určit novou *P-hodnotu* pomocí testu dobré shody na jednotlivé *P-hodnoty* (tedy *P-hodnotu* z *P-hodnot*). Výpočet je dán vzorcem 5.1

$$\chi^2 = \sum_{i=1}^{10} \frac{(F_i - s/10)^2}{s/10} \tag{5.1}$$

kde $F_i$ je četnost *P-hodnot* ve třídě $i$ a $s$ je počet sekvencí. *P-hodnota* se pak vypočítá pomocí neúplné gama funkce $P\text{-}hodnota_T = \text{igamc}(9/2, \chi^2/2)$. Pokud $P\text{-}hodnota_T \geq 0,0001$, pak lze rozdělení sekvencí považovat za rovnoměrné. Aby daná statistika měla význam, je pořeba uvažovat alespoň 55 sekvencí.

## 5.2 Analýza testování

Nejen interpretace výsledků je důležitá při vlastním testování. Pozornost je potřeba věnovat také nastavení vstupních parametrů jednotlivých testů a také vhodnost použití daného testu na zkoumaná data. Za tímto účelem byla NIST[13] vydána některá doporučení ohledně vhodného nastavení vstupních parametrů a nastavení testů.

### Délka sekvence

Rozhodnutí o optimální délce sekvence není jednoduché. Pokud vezmeme v úvahu jednotlivé testy zjistíme, že Většina testů požaduje velikost vstupní sekvence v řádech $10^4$ bitů. Nicméně takto krátké sekvence jsou problémové pro výpočet statistiky jiných testů jako je Maurerův „univerzální statistický“ test, který vyžaduje sekvenci o délce v řádu $10^6$. Je tedy třeba dobře zvážit velikost testované sekvence podle aplikovaného testu.

### Počet sekvencí

Určení počtu sekvencí je těsně vázáno na volbu hladiny významnosti. Jestliže zvolíme hladinu významnosti $\alpha = 0,001$ očekává se, že zhruba 1 sekvence z 1000 bude zamítnuta. Pokud pak zvolíme počet sekvencí nedostatečně, například 100, výsledek testování nebude vyhodnocen adekvátně. Počet sekvencí by měl být vetší než reciproká hodnota hladiny významnosti.

### Velikost bloku

Příkladem tohoto problému může být nevhodně zvolená velikost bloku u testu entropie. Pro sekvenci o délce v řádu $10^6$ by se dalo očekávat, že vhodná délka bloku se blíží $m = \log_2 n$, jelikož obsahuje maximum entropie. Nicméně se ukázalo, že

pro $m > 14$ se pozorovaná statistika začíná lišit od očekávané. NIST tedy doporučuje, aby velikost bloku byla menší než $\log_2 n - 5$.

### Velikost šablon

U testů překrývajících se a nepřekrývajících se šablon je potřeba vzít v úvahu i délku těchto šablon. Pokud zvolíme délku nevhodně, například větší než $\log_2 n$ je jasné, že výskyt takovýchto šablon se bude téměř vždy blížit nule, což nám neposkytne žádnou užitečnou informaci.

## 5.3 Vlastní měření

Statistické testy NIST jsou navrženy a zobecněny tak, aby na vstupní data byly kladeny co nejmenší požadavky. Nicméně úplné zobecnění problému je velice težce interpretovatelné pomocí programového kódu. Z tohoto důvodu platí pro některé testy určitá omezení. Například u testu náhodných návštěv a testu variací náhodných návštěv je uměle vytvořena horní hranice počtu cyklů jako $max(1000, n/128)$. Aplikaci tohoto omezení můžeme pozorovat ve výsledkových tabulkách jako menší počet testovaných sekvencí u zmíněných dvou testů. První měření probíhalo s připojeným mikrofonem do mikrofonního vstupu a v tichém prostředí. Mikrofon tak mohl snímat pouze zvuky způsobené provozem počítače. Nicméně při testování byl mikrofon umístěn v dostatečné vzdálenosti od počítačové skříně tak, aby se tento vliv v co největší míře eliminoval. V tabulce 5.2 je uvedena část výstupu vyhodnocení generátoru. Kompletní vyhodnocení je k dispozici v příloze.

Tabulka 5.2 tedy zobrazuje výsledky jednotlivých testů. Celková *P-hodnota* testu je výsledkem testu dobré shody jednotlivých *P-hodnot* a musí být větší než 0,0001, abychom mohli test považovat za úspěšný. Druhým důležitým kritériem je pak sloupec *Úspěšnost*, který udává poměr úspěšných sérií v testu k celkovému počtu. Minimální počet úspěšných sérií pro zvládnutí testu byl stanoven na 75 u testu náhodných návštěv a testu variací náhodných návštěv. Pro zbylé testy byl stanoven minimální počet úspěšných sérií na 123.

Druhé měření probíhalo v silně zarušeném prostředí. Mikrofon byl umístěn přímo u reproduktorů, ze kterých hrála hlasitá hudba. Cílem tohoto měření byla snaha o zahlcení mikrofonu silným signálem. Výsledek testování je zachycen v tabulce 5.3. Minimální počet úspěšných sérií pro zvádnutí testu byl nastaven u testu náhodných návštěv a testu variací náhodných návštěv na 86. Pro ostatní testy byla určena minimální hranice 123 úspěšných sekvencí.

Poslední měření probíhalo se zapojeným mikrofonním vstupem, ale odpojeným mikrofonem. Použitý mikrofon je součástí náhlavní sestavy sluchátek s možností

Tab. 5.2: Test generátoru pro klidné prostředí

| Test | P-hodnota | Úspěšnost |
|---|---|---|
| Test četnosti | 0.043745 | 127/128 |
| Test četnosti v bloku | 0.392456 | 127/128 |
| Test kumulativních součtů | 0.066882 | 127/128 |
| Test shodné série | 0.949602 | 127/128 |
| Test nejdelší série v jednom bloku | 0.788728 | 126/128 |
| Test nezávislosti | 0.078086 | 127/128 |
| Test spektra | 0.671779 | 128/128 |
| Test nepřekrývajících se šablon | 0.046169 | 127/128 |
| Test překrývajících se šablon | 0.287306 | 128/128 |
| Maurerův „univerzální statistický“ test | 0.033288 | 128/128 |
| Test entropie | 0.585209 | 127/128 |
| Test náhodných návštěv | 0.548605 | 76/79 |
| Test variací náhodných návštěv | 0.238562 | 78/79 |
| Test četnosti série | 0.046169 | 126/128 |
| Test lineární zpětné vazby | 0.095617 | 127/128 |

odpojení tohoto mikrofonu. Na vstupu zvukové karty se tak budou vyskytovat pouze šumové signály. Výsledek tohoto měření opět přehledně zobrazuje tabulka 5.4. Minimální počet úspěšných sérií byl stanoven na 92 pro testy náhodných návštěv a variací náhodných návštěv a 123 pro zbylé testy.

### 5.3.1 Měření zdroje entropie

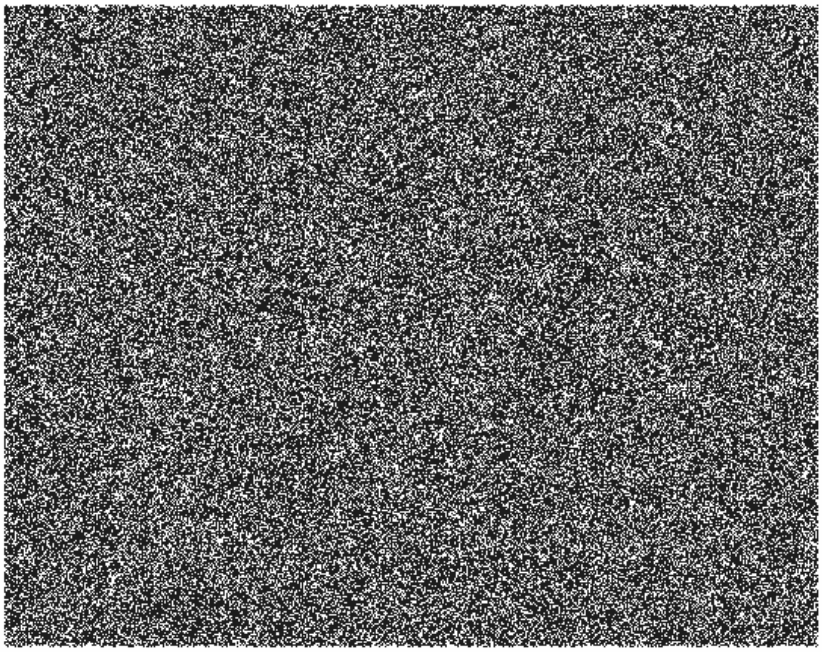
Předchozí trojici měření jsem provedl také pro data generovaná pouze mikrofonním vstupem. Z výsledků tohoto testování se ukázalo, že generovaná posloupnost uspěla pro všechny měření vždy pouze u testu nezávislosti a testu lineární zpětné vazby. U zbylých testů nastává problém zejména při výpočtu *P-hodnoty*, jelikož zde dochází k chybě *igamc: UNDERFLOW*. Tedy výsledek výpočtu funkce *igamc* je menší, než minimální hodnota, kterou je počítač schopen uložit v paměti. Pro bližší zkoumání problému jsem provedl ještě grafický test. Generovaná data jsem načetl do prostředí MATLAB ve formátu 32bitových celých čísel. Následně jsem dvojice po sobě jdoucích čísel vynesl do grafu jako souřadnice bodu $x$, $y$. Výstupy jsou zobrazeny na obrázcích 5.2 a 5.3. Prezentované grafické výstupy byly shodné pro všechny tři typy měření. Na prvním obrázku 5.2 je grafické rozložení dat generovaných navženým generátorem. Je patrné, že výstupní generovaná data jsou rozprostřeny rovnoměrně po celé ploše tak, jak bychom to očekávali u náhodných čísel. Naopak u ob-

Tab. 5.3: Test generátoru pro rušné prostředí

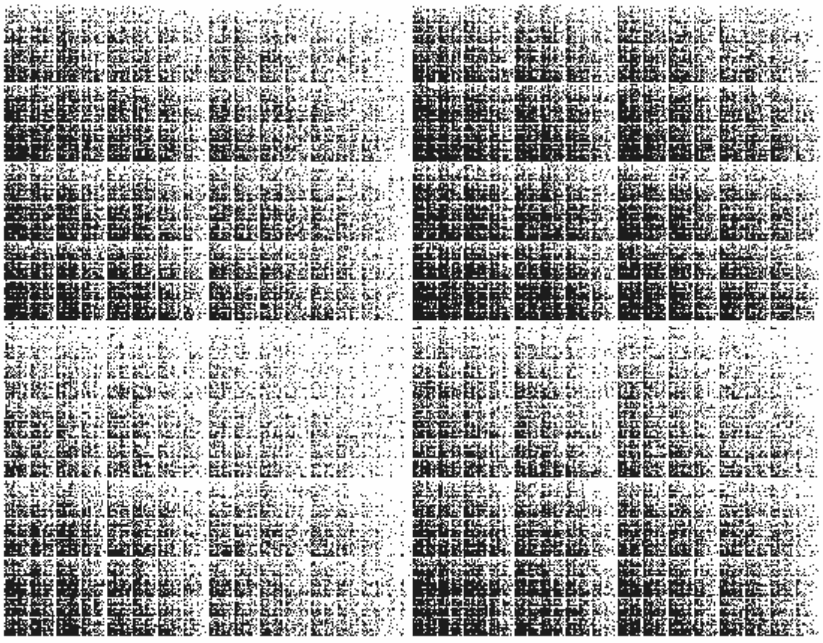
| Test | P-hodnota | Úspěšnost |
|---|---|---|
| Test četnosti | 0.004861 | 127/128 |
| Test četnosti v bloku | 0.082177 | 128/128 |
| Test kumulativních součtů | 0.021262 | 127/128 |
| Test shodné série | 0.602458 | 126/128 |
| Test nejdelší série v jednom bloku | 0.689019 | 127/128 |
| Test nezávislosti | 0.009998 | 128/128 |
| Test spektra | 0.350485 | 127/128 |
| Test nepřekrývajících se šablon | 0.155209 | 126/128 |
| Test překrývajících se šablon | 0.232760 | 127/128 |
| Maurerův „univerzální statistický“ test | 0.819544 | 128/128 |
| Test entropie | 0.534146 | 123/128 |
| Test náhodných návštěv | 0.996713 | 88/90 |
| Test variací náhodných návštěv | 0.602458 | 88/90 |
| Test četnosti série | 0.819544 | 128/128 |
| Test lineární zpětné vazby | 0.043745 | 127/128 |

rázku 5.3 lze pozorovat jasně nerovnoměrné rozložení. Výstupní data generovaná mikrofonem tedy nejsou rovnoměrně rozložená ve všech intervalech. Proto při využití generátoru pro generování kryptografických klíčů by bylo vhodné zvolit vyšší velikost inicializačního semínka tak, aby bylo dosaženo dostatečné akumulace entropie.

Navržený generátor tedy úspěšně prošel sadou statistických testů NIST pro testování generátorů náhoných čísel. Generátor je navíc díky použití kombinace zdroje entropie a hashovacích fukncí vhodný i pro generování bezpečnostních klíčů.



Obr. 5.2: Grafické rozložení dat generovaných generátorem



Obr. 5.3: Grafické rozložení dat generovaných zdrojem entropie

Tab. 5.4: Test generátoru s odpojeným mikrofonem

| Test | P-hodnota | Úspěšnost |
|---|---|---|
| Test četnosti | 0.772760 | 128/128 |
| Test četnosti v bloku | 0.848588 | 126/128 |
| Test kumulativních součtů | 0.957319 | 127/128 |
| Test shodné série | 0.023812 | 124/128 |
| Test nejdelší série v jednom bloku | 0.311542 | 128/128 |
| Test nezávislosti | 0.148094 | 126/128 |
| Test spektra | 0.392456 | 128/128 |
| Test nepřekrývajících se šablon | 0.311542 | 127/128 |
| Test překrývajících se šablon | 0.888137 | 125/128 |
| Maurerův „univerzální statistický“ test | 0.222869 | 127/128 |
| Test entropie | 0.862344 | 127/128 |
| Test náhodných návštěv | 0.148094 | 91/92 |
| Test variací náhodných návštěv | 0.625552 | 90/92 |
| Test četnosti série | 0.819544 | 127/128 |
| Test lineární zpětné vazby | 0.862344 | 127/128 |

# 6 ZÁVĚR

V rámci bakalářské práce byla popsána problematika generátorů náhodných čísel. První kapitola je věnována jednotlivým typům používaných generátorů. Mezi tyto generátory patří generátor náhodné posloupnosti, pseudonáhodné posloupnosti a také entropický generátor náhodné posloupnosti. Poslední jmenovaný generátor je také použit při vlastním návrhu generátoru náhodných čísel pro běžný počítač.

Druhá kapitola se věnuje testům generátorů náhodných čísel. Obsahuje popis souboru 15ti statistických testů, které jsou navrženy tak, aby zkoumaly veškeré situace, které mohou nastat u nenahodilé sekvence čísel.

Další část práce pojednává o vlastním návrhu entropického generátoru. V jednotlivých částech je postupně rozebrán teoretický popis návrhu, jeho praktická realizace a otestování funkčnosti generátoru. V rámci návrhu generátoru byl zvolen jako zdroj entropie mikrofonní vstup, který je dnes běžně dostupný u osobních počítačů. Kapitola zabývající se realizací generátoru obsahuje podrobný popis použitých algoritmů v programu, sloužících ke generování náhodných čísel. Poslední kapitola práce zhodnocuje testování navženého generátoru. Výstupní data vyprodukovaná programem úspěšně prošla veškerými testy. Lze tak konstatovat, že navržený generátor je plně funkční.

# LITERATURA


[1] BARKER, E. KELSEY, J. *Recommendation for Random Number Generation Using Deterministic Random Bit Generators* NIST SP 800-90A. Gaithersburg: National Institute of Standards and Technology, 2012.

[2] BARKER, E. KELSEY, J. *Recommendation for Entropy Sources Used for Random Bit Generation* NIST SP 800-90B. Gaithersburg: National Institute of Standards and Technology, 2012.

[3] BALDWIN, W. *Preliminary Analysis of the BSAFE 3.x Pseudorandom Number Generators* California: RSA Laboratorie´s Bulletin, 1998. Dostupné z: `ftp://ftp.rsasecurity.com/pub/pdfs/bulletn8.pdf`.

[4] BURDA, K. *Aplikovaná kryptografie.* Brno: VUTIUM, 2013. ISBN 978-80-214-4612-0.

[5] HERBERT, H. *Digital random number generator using partially entropic data* Phoenix: United States Patent US8489660B2, 2009. Dostupné z: `http://www.freepatentsonline.com/8489660.pdf`

[6] CHAICHANAVONG, P. *Entropy source for random number generation* Mountain View: United States Patent US8015224B1, 2007. Dostupné z: `http://www.freepatentsonline.com/8015224.pdf`

[7] KNUTH, E. *The art of computer programming* 2th ed. Addison-Wesley Publishing Company, 1981, 688 s. ISBN 0-201-03822-6.

[8] KOCHER, P. *The intel random number generator* Tech. Rep., Cryptography Research, Inc. White Paper Prepared for Intel Corporation, 1999.

[9] MORRISON, R. *Design of a True Random Number Generator Using Audio Input* Oregon: Department of Electrical and Computer Engineering, 2001. Dostupné z: `http://www.cs.ucsb.edu/~koc/cren/project/pp/morrison.pdf`

[10] NIST *Random Number Generation* NIST Information Technology Laboratory. [online]. 2014 [cit. 2014-04-22]. Dostupné z: `http://csrc.nist.gov/groups/ST/toolkit/rng/index.html`

[11] PRESS, W. *Numerical Recipies: The art of scientific computing.* 3rd ed. Cambridge: Cambridge University Press, 2007, xxi, 1235 s. ISBN 9780521880688.

[12] QT DIGIA. *Qt Project* [online]. 2013 [cit. 2013-12-30]. Dostupné z: `http://qt-project.org/`

[13] RUKHIN, A. *A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications.* NIST SP 800-22. Gaithersburg: National Institute of Standards and Technology, 2010.

# SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

GNP generátor náhodné posloupnosti

GPNP generátor pseudonáhodné posloupnosti

mod modulo, zbytek po celočíselném dělení

LFSR posuvný registr s lineární zpětnou vazbou

LKG lineární kongruentní generátor

H() aplikace hešovací funkce

NIST National Institute of Standards and Technology

$\otimes$ exkluzivní součet

*P-hodnota* pravděpodobnost, že testovací kritérium dosáhlo své hodnoty

$\chi^2$ rozdělení pravděpodobnosti chí kvadrát

$\alpha$ hladina významnosti

igamc neúplná funkce gama

# A OBSAH PŘILOŽENÉHO CD

- Text bakalářské práce ve formátu PDF.
- Zdrojové kódy navrženého generátoru.
- Přeložený kód pro MS Windows.
- Naměřená data.
- Výsledky testování generátoru podle NIST.