VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 Lenka Procházková

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

GENERÁTOR PRAVÝCH NÁHODNÝCH ČÍSEL
TRUE RANDOM NUMBER GENERATOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Lenka Procházková

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Vlastimil Člupek, Ph.D.

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Informační bezpečnost**
Ústav telekomunikací

Studentka: Lenka Procházková ***ID:*** 195812
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Generátor pravých náhodných čísel

POKYNY PRO VYPRACOVÁNÍ:

V bakalářské práci proveďte analýzu možností realizace generátorů pravých náhodných čísel a určete hardwarové nároky jednotlivých řešení. Na základě provedené analýzy vyberte vhodné způsoby realizace generátoru pravých náhodných čísel na PC a na zařízeních využívaných v internetu věcí. Popište způsoby testování funkčnosti generátorů pravých náhodných čísel. Vytvořte návrh realizace generátoru pravých náhodných čísel na platformě PC. Realizujte navržený generátor pravých náhodných čísel a ověřte jeho funkčnost. Navrhněte rozšíření generátoru, které umožní jeho provoz i na smartphonech.

DOPORUČENÁ LITERATURA:

[1] STIPČEVIĆ, Mario; KOÇ, Çetin Kaya. True random number generators. In: Open Problems in Mathematics and Computational Science. Springer, Cham. 2014, 275-315.

[2] BALATTI, Simone, et al. True random number generation by variability of resistive switching in oxide-based devices. IEEE Journal on Emerging and Selected Topics in Circuits and Systems. 2015, 5(2), 214-221.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 27.5.2019

Vedoucí práce: Ing. Vlastimil Člupek, Ph.D.
Konzultant:

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

ABSTRAKT

Tato bakalářská práce si klade za cíl seznámit čtenáře s generátory pravých náhodných čísel a jejich možnou realizací. Nejprve je věnována pozornost náhodným číslům a generátorům pravých náhodných čísel, u kterých jsou představeny jejich principy a vlastnosti. Dále je uvedena vhodnost použití generátorů na počítači a na zařízení využívaném v internetu věcí. Práce obsahuje popis možností testování výsledných náhodných sekvencí. Součástí práce je návrh a realizace generátoru pravých náhodných čísel, který jako zdroj entropie využívá pohyb kurzoru uživatelem a návrh generátoru pro mobilní telefony. K otestování výsledné sekvence byla použita sada testů NIST STS. Práce obsahuje porovnání výsledků s výsledky testů NIST STS u sekvencí z internetových zdrojů.

KLÍČOVÁ SLOVA

Náhodná čísla, generátor náhodných čísel, generátor pravých náhodných čísel, TRNG, testování, NIST STS.

ABSTRACT

The goal of this bachelor thesis is to apprise with true random number generators and their realisation. At first, random numbers and true random number generators are described and principles, details and properties are explained. The thesis also describes the use of generators in computers and in IoT devices and random sequences testing. The part of this thesis is description and realisation of computer generator and description of smartphone generator. The source of entropy used in this work is cursor move. Generated sequences were tested by the NIST STS and compared with sequences from other generators.

KEYWORDS

Random numbers, random number generator, true random number generator, TRNG, testing, NIST STS.

PROCHÁZKOVÁ, Lenka. *Generátor pravých náhodných čísel*. Brno, Rok, 68 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Vlastimil Člupek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Generátor pravých náhodných čísel“ jsem vypracovala 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 autorka uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma 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 autorky

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu bakalářské práce panu Ing. Vlastimilovi Člupkovi, Ph.D. za čas, který mi věnoval, odborné vedení, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autorky

Obsah

Úvod **12**

1 Náhodná čísla **13**
1.1 Generátor pseudonáhodných čísel 14
1.2 Generátor pravých náhodných čísel 14
1.2.1 Generátory založené na šumu 16
1.2.2 Generátory založené na chaosu 18
1.2.3 Generátory založené na kruhových oscilátorech 19
1.2.4 Generátory založené na radioaktivním rozpadu 21
1.2.5 Kvantové generátory 21
1.2.6 Generátory založené na chování uživatele 22
1.3 Postproces 23

2 Základní vlastnosti generátorů **25**
2.1 Rychlost generování 25
2.2 Hardwarové nároky 26
2.3 Kryptografické zabezpečení 27

3 Užití náhodných čísel **28**
3.1 Užití náhodných čísel v kryptografii 28
3.1.1 Symetrická kryptografie 28
3.1.2 Asymetrická kryptografie 29
3.2 Generátory vhodné pro použití v PC 29
3.3 Generátory používané v internetu věcí 30

4 Testování generátorů náhodných čísel **32**
4.1 NIST 32
4.2 Dieharder 35

5 Návrh generátoru pravých náhodných čísel **36**
5.1 Základní technické parametry 36
5.2 Struktura aplikace 37

6 Realizace generátoru **41**
6.1 Generování jednotlivých bitů 41
6.2 Převod bitů na heslo 42
6.3 Potíže při realizaci 43
6.4 Spustitelný JAR soubor 45

7 Testování aplikace a vygenerovaných sekvencí bitů **46**
7.1 Časová náročnost generování sekvencí 46
7.2 Výpočetní náročnost aplikace 48
7.3 Testování vygenerovaných sekvencí bitů 50

8 Návrh rozšíření aplikace na smartphony **55**
8.1 Základní technické parametry 55
8.2 Návrh struktury aplikace 55

9 Závěr **59**

Literatura **61**

Seznam symbolů, veličin a zkratek **66**

Seznam příloh **67**

A Obsah přiloženého CD **68**

Seznam obrázků

1.1 Graf znázorňující rozdělení generátorů. 13
1.2 Graf závislosti entropie E zdroje binárních symbolů na pravděpodobnosti p_0 výskytu bitu s hodnotou 0. 15
1.3 Proces tvorby náhodných čísel pomocí generátorů založených na šumu. 17
1.4 Schéma generátoru využívajícího teplotní šum rezistoru [2]. 18
1.5 Třístupňový kruhový oscilátor sestavený z invertorů (i_1, i_2 a i_3) [1]. . 20
1.6 Schéma zapojení TRNG s větším množstvím kruhových oscilátorů [12]. 20
1.7 Schéma kvantového generátoru [15]. 22
2.1 Graf znázorňující rychlost jednotlivých generátorů v kBps. 26
5.1 Grafické uživatelské rozhraní aplikace. 37
5.2 Schéma znázorňující proces generování náhodných čísel pomocí aplikace. 38
5.3 Schéma znázorňující proces generování náhodných čísel a testování výsledné sekvence. 38
5.4 Vývojový diagram znázorňující generování náhodných čísel. 39
5.5 Výsledný výpis obsahující vygenerovanou sekvenci, heslo a dobu potřebnou ke generování. 40
7.1 Graf znázorňující dobu generování sekvencí o různých délkách. 47
7.2 Graf znázorňující dobu generování jednoho bitu v sekvencích o různých délkách. 47
8.1 Návrh grafického uživatelské rozhraní aplikace pro mobilní telefony. . 56
8.2 Vzhled tlačítek v mobilní aplikaci. 56
8.3 Vývojový diagram znázorňující algoritmus pro generování náhodných sekvencí na mobilních zařízeních (smartphonech). 58

Seznam tabulek

2.1 Srovnání PRNG a TRNG 25
2.2 Rychlost generování 26
2.3 Hardwarové nároky 27
3.1 Vhodnost TRNG a PRNG pro aplikace 28
6.1 Ukázka převodu bitů na heslo 42
7.1 Časová náročnost generátoru 46
7.2 Využití systémových prostředků během chodu aplikace 48
7.3 Využití systémových prostředků během mazání vygenerovaných sekvencí 48
7.4 Využití systémových prostředků během nastavování parametrů generátoru 49
7.5 Využití procesoru během generování sekvencí 49
7.6 Využití paměti RAM během generování sekvencí 49
7.7 Využití grafického procesoru během generování sekvencí 50
7.8 Výsledek testu NIST STS vygenerované sekvence 51
7.9 Porovnání výsledků testu NIST STS se sekvencemi vygenerovanými pomocí internetových zdrojů 53
7.10 Porovnání výsledků testu NIST STS se sekvencemi vygenerovanými pomocí internetových zdrojů 54

Seznam výpisů

6.1 Generování bitů pomocí pohybu kurzoru. 41
6.2 Ukázka kódu pro převod bitů na heslo. 42
6.3 Vytvoření nového vlákna. 44
6.4 Ukázka kódu, který proběhne po úspěšném vygenerování sekvence. . . 44

Úvod

V dnešní době se rozšiřuje množství lidí využívající mobilní sítě nebo internet pro komunikaci. Proto se zvyšuje potřeba zabezpečení šířeného obsahu. Je důležité, aby data předávaná pomocí mobilních sítí nebo internetu byla chráněna před zneužitím. Mezi chráněná data patří osobní údaje a fotografie, firemní informace, informace o kreditních kartách, apod.

Zabezpečení dat je důležité i u IoT (Internet of Things - Internet věcí) technologií. IoT zařízení tvoří vlastní síť, ve které si jednotlivá zařízení předávají informace, které získávají ze zabudovaných senzorů. Mezi získaná data se řadí záznamy z kamer, informace o vstupech do budov nebo hodnoty získané z lékařských senzorů. I tato data je potřeba chránit.

Pokud by neexistovalo žádné zabezpečení dat, bylo by snadné data získat a zneužít. K ochraně dat se využívají kryptografické metody jako je šifrování. Pro zašifrování dat je potřeba dlouhý a náhodně vygenerovaný klíč. Náhodně vygenerovaným klíčům a jejich zdrojům se věnuje tato bakalářská práce.

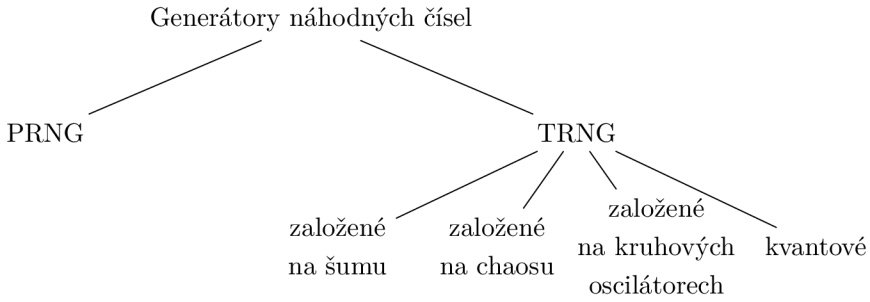
Předmětem práce je návrh dvou generátorů, jeden pro PC, druhý pro smartphone[1] a realizace generátoru pro PC. Práce v první kapitole definuje pojem náhodná čísla a popisuje generátory pravých náhodných čísel i jejich rozdělení. V následující kapitole jsou rozebrány vlastnosti generátorů, rychlost generování a kryptografické zabezpečení generátorů. V kapitole tři je uvedena vhodnost jednotlivých generátorů pro použití v PC a v zařízeních využívajících v internetu věcí. Ve čtvrté kapitole jsou popsány možnosti testování náhodných vygenerovaných sekvencí pomocí statistických testů.

Hlavním výstupem práce je návrh a realizace generátoru pravých náhodných čísel určeného pro použití na platformě PC. Návrh generátoru je uveden v kapitole pět. Realizace generátoru podle návrhu a potíže vzniklé při realizaci jsou popsány v kapitole šest. Kapitola sedm obsahuje čas potřebný pro generování náhodných sekvencí o různých délkách, výpočetní náročnost aplikace, postup testování vygenerovaných sekvencí a výsledky testování bitových sekvencí. V kapitole jsou dále porovnány výsledky testování generátoru s výsledky testování sekvencí z internetových zdrojů. Kapitola osm se věnuje rozšíření generátoru, které umožní jeho provoz i na smartphonech (tzv. chytrých mobilních telefonech). V deváté kapitole je sepsán závěr této práce.

[1] Smartphone neboli chytrý telefon je telefon s operačním systémem a pokročilými funkcemi, mezi které patří možnost instalovat a upravovat aplikace.

1 Náhodná čísla

Náhodná čísla se dělí na pseudonáhodná a pravá náhodná čísla, která se dále dělí podle typu generátoru a použitého algoritmu. V angličtině se pro generátory pseudonáhodných čísel používá označení PRNG (Pseudo Random Number Generator) a pro generátory pravých náhodných čísel TRNG (True Random Number Generator). Rozdělení graficky znázorňuje obrázek 1.1, který je převzat z [1]. Třetím typem je entropický generátor, který je kombinací obou výše zmíněných generátorů [1, 2].



Obr. 1.1: Graf znázorňující rozdělení generátorů.

S novými technologiemi se zvyšuje důležitost generátorů pravých náhodných čísel a fyzikálních nedeterministických generátorů náhodných čísel. Generátory náhodných čísel patří mezi nejaktuálnější témata výzkumu v posledním desetiletí. Každý rok vznikne okolo 80 nových patentů. Celkový počet patentů od roku 1970 přesahuje číslo 1400 [1].

Na počátku moderní éry počítačů se otázkou náhodných čísel zabýval John von Neumann, který přišel s tvrzením, že deterministický Turingův stroj není schopen generovat náhodná čísla. Náhodná čísla jsou nezbytná v číselných simulacích, statistických výzkumech, loterii a kryptografii. Dnes jsou náhodná čísla nejvíce využívaná v kryptografii, kde jsou všechny části zabezpečovacích protokolů zveřejněné až na klíč, nebo jinou tajnou informaci. Tato informace musí být náhodná a u symetrické kryptografie ji zná pouze příjemce a odesílatel. Zabezpečovací protokoly se používají pro mobilní komunikace, přístup do mailu, online platby, elektronické bankovnictví, internetový obchod, chod bankomatů a bezkontaktní klíče, atd. Další odvětví využívající náhodná čísla jsou loterie. V tomto případě má většina států striktně dané požadavky na náhodná čísla užívaná v herních a loterijních automatech. Tato opatření mají za úkol zajistit spravedlivou hru a zamezit možnosti předpovídat výsledky [1].

Mezi příklady užití náhodných čísel v kryptografii patří protokol BB84 [3], u kterého by bylo nebezpečné, kdyby byl útočník schopen vypočítat nebo předpovědět tajné heslo Alice a Boba. Protokol BB84 je nejstarší protokol kvantové kryptografie,

který slouží k distribuci klíčů. Kdyby byl útočník schopen vypočítat tajná čísla, kryptografický potenciál daného protokolu by byl roven nule [1].

1.1 Generátor pseudonáhodných čísel

Generování náhodných čísel je proces, který má nepředvídatelný výsledek. Pseudonáhodná čísla jsou generována softwarově nebo pomocí předvypočítaných tabulek, které jsou určené pro tvorbu číselných sekvencí. Základem pro generování čísel je matematický výpočet, který vytváří deterministickou periodickou řadu čísel. V praxi má vysokou míru korelace[1] a nulové zkreslení, počet jedniček je totožný s počtem nul [1, 4].

Mezi vlastnosti pseudonáhodných čísel patří opakovatelnost. Ta by měla být za co nejdelší časový úsek. Pokud známe počáteční čísla, lze dopočítat výstupní číselnou posloupnost. Výslednou posloupnost bitů lze v těchto případech i předpovědět. Kladnou vlastností jsou nízké náklady, rychlost a snadná implementace. Pro realizaci stačí počítač, není potřeba dodatečný hardware [1, 4]. V kryptografii se nejčastěji používá kombinace PRNG a TRNG. Tato kombinace je popsána v kapitole 2.3.

1.2 Generátor pravých náhodných čísel

Podle Kerckhoffsova principu musí být generátor náhodných čísel schopen generovat náhodné bity, i když bude každá část generátoru zveřejněná (schéma, algoritmy, apod.) [1].

Základní odlišností od PRNG je způsob realizace generátoru. TRNG jsou založeny na fyzikálních jevech, které jsou nedeterministické. Proto jsou vhodnější pro generování pravých náhodných čísel. Implementace je složitější než u PRNG, protože je potřeba dodatečný hardware, který je často mimo počítač. Obvykle je k PC připojený pomocí USB nebo PCI sběrnice [1].

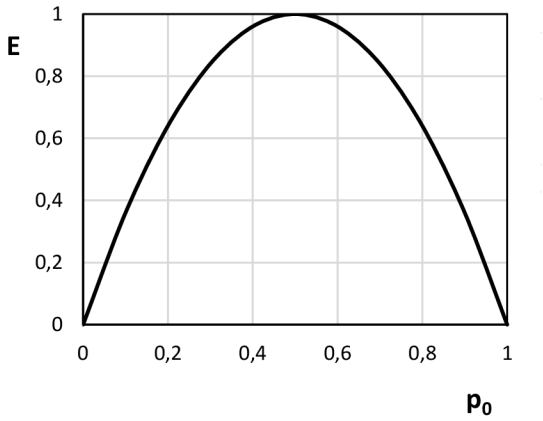
Entropie je veličina, která popisuje míru neurčitosti systému. Systém se může s pravděpodobností p_1 až p_N nacházet v N různých stavech X_1 až X_N. Při generování náhodných posloupností v binární soustavě mohou nastat 2 stavy proto $N = 2$ [2].

První stav nastane s pravděpodobností p_0, když je výstup generátoru $X = 0$. Druhý stav nastane s pravděpodobností $p_1 = 1 - p_0$. Tento stav má na výstupu hodnotu $X = 1$. Entropie je definována rovnicí 1.1. Jednotkou entropie je bit (binární symbol) [2],

$$E_x = (-log_2 p_x). \tag{1.1}$$

[1]Korelace je vzájemný vztah mezi dvěma veličinami, v našem případě vztah jedniček a nul.

Závislost entropie ilustruje obrázek 1.2 převzatý z [2]. Z grafu vyplývá, že pokud je p_0 rovno 0 nebo 1 je entropie E nulová. Nulová entropie nastane, pokud má zdroj entropie v binární soustavě na svém výstupu samé jedničky nebo nuly. V tomto případě dokážeme odhadnout celou posloupnost s pravděpodobností úspěchu $1/N = 0,5$. Pokud platí, že pravděpodobnost $p_1 = p_0 = 0,5$ je entropie maximální a posloupnost nelze odhadnout [2].



Obr. 1.2: Graf závislosti entropie E zdroje binárních symbolů na pravděpodobnosti p_0 výskytu bitu s hodnotou 0.

Zdroje entropie je potřeba řídit a sledovat faktory, které by mohly ovlivnit kvalitu vytvořené řady. Mezi tyto faktory patří provozní teplota, náchylnost k elektronickému šumu a rušení, změna napájecího napětí a stáří součástek. Z těchto důvodů jsou generovaná čísla statisticky testována, aby se ověřila jejich náhodnost. Testování je uvedeno v kapitole 4. V případě, že generátor neprojde statistickými testy, nelze ho označit jako kvalitní. Zároveň ale nelze prokázat, že generátory, které těmito testy projdou, jsou skutečně kvalitní a náhodné, mohou obsahovat jiné slabiny [5].

Pravděpodobnost zvolení 1 nebo 0 popisuje rovnice 1.2, kde b je odchylka a p jsou jednotlivé pravděpodobnosti [1],

$$b = \frac{p(1) - p(0)}{2}. \tag{1.2}$$

TRNG jsou tvořené tak, aby korelace mezi jednotlivými bity byla malá [1].

Existuje velké množství konstrukcí TRNG a stále nové se vyvíjejí. Obecně je lze rozdělit na 4 hlavní typy: kvantové, založené na šumu, založené na chaosu a založené na kruhových oscilátorech [1]. Jednotlivé detektory a jejich vlastnosti jsou popsány níže v kapitole 2.

Náhodné sekvence lze získat i pomocí systémových hodin, stisku kláves na klávesnici, zátěže operačního systému, zvuku snímaného mikrofonem, obrazu z webkamery nebo pohybu myší. Protože tyto generátory mohou mít opakující se periodu, je lepší

používat generátory využívající šum, chaos, kruhové oscilátory nebo kvantové generátory [6].

Mnoho aplikací využívá TRNG jako základ, který se pak upravuje v PRNG. Důvodem je nižší časová náročnost PRNG. Kombinací těchto dvou druhů generátorů lze získat velké množství velmi kvalitních náhodných čísel během krátké doby [7].

1.2.1 Generátory založené na šumu

Atmosférický šum vzniká v důsledku elektromagnetických poruch. Tento šum se může šířit na tisíce kilometrů. Je snadné ho získat, ale je nutné vyvarovat se zdrojům, které by vytvářely periodičnost. Šum lze rozlišit na vnější a vnitřní, nebo-li uměle vytvořený. Vnější šum vzniká v přírodě. Jeho zdrojem může být bouřka, vítr, moře, paprsky slunce, nebo vesmírný prach dopadající na Zem. Vnitřní, nebo-li uměle vytvořený je nechtěně vyvolaný pomocí elektronických obvodů [8].

Další dělení je na bílý, barevný, praskavý, blikavý a tepelný šum. Pro TRNG je nejvhodnější bílý a tepelný šum. Bílý šum je náhodný s rovnoměrnou výkonovou spektrální hustotou, v praxi se používá k testování kmitočtové odezvy a generování náhodných čísel. Tepelný šum využívá velké množství volných iontů a elektronů ve vodičích, které vytváří teplo a vibrace. Jejich pohyb vytváří proud, který má náhodný charakter. Zdroj odporu ve vodiči je vhodný pro TRNG [8].

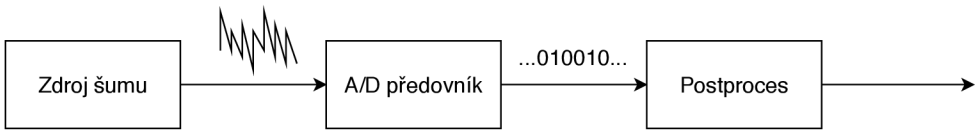
Nejlépe se ke generování náhodných čísel hodí bílý šum, který má rovnoměrnou výkonovou spektrální hustotu a nemá žádnou autokorelaci[2] [8].

Hlavním principem generátoru založeného na šumu je pravidelné odebírání náhodného analogového napětí. Toto napětí se porovnává s určitou předem stanovenou prahovou hodnotou. Je-li hodnota napětí vyšší než prahová hodnota, je vygenerována jednička, v opačném případě se vygeneruje nula. Prahová hodnota musí být nastavena tak, aby pravděpodobnost 1 a 0 byla stejná. Nastavování prahové hodnoty je časově náročný problém. Ladění odchylky na hodnotu $0,1$ vyžaduje 10 sekund, ladění na 10krát nižší hodnotu $(0,01)$ bude trvat 100krát delší dobu. Dalším problémem je stabilita. Protože i malá změna průměrné hodnoty vede ke zkreslení [1].

U všech generátorů využívajících šum je potřeba následné zpracování. Mezi používané úpravy výsledných sekvencí patří jednoduché zpracování pomocí Ad hoc korektoru, jako je XOR několika bitů [4]. Výsledná úprava neboli postproces je popsána v kapitole 1.3. Celý proces generování náhodných čísel znázorňuje obrázek 1.3.

Příkladem generátoru náhodných čísel používající atmosférický šum je webová stránka **random.org**. Tento generátor využívá radiostanice zaznamenávající atmosférický tlak. Důvodem využití tohoto fyzikálního jevu jsou složité a málo přesné

[2]Nelze odhadnout další hodnoty.



Obr. 1.3: Proces tvorby náhodných čísel pomocí generátorů založených na šumu.

předpovědi. Data získaná pomocí radiostanic se používají jako „surovina“ pro generátory, které jsou dostupné na `https://www.random.org/`. V současné době se data poskytované touto službou využívají pro vědecké účely, loterie a online hry. Tato služba existuje od roku 1998, kdy ji zprovoznil doktor Mads Haahr [4].

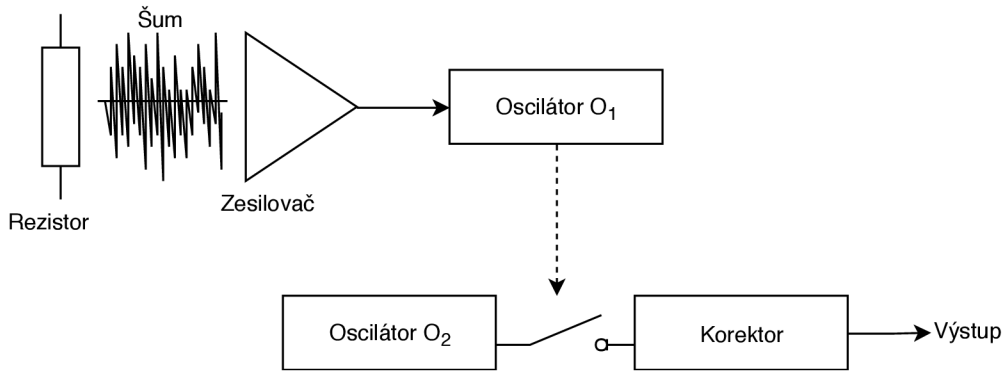
Aranous Alea II je kompaktní TRNG využívající polovodičové křižovatky pro generování Gaussova bílého šumu. Tento šum je zesílen pomocí A/D (analogově-digitálního) převodníku a zpracován vestavěným mikroprocesorem. Tento generátor se k PC připojuje pomocí USB portu, přes který je napájen [9].

Generátory založené na teplotním šumu rezistoru

Šum na analogových součástkách je náhodný signál, který narušuje zpracování a přenos signálu. Základní rozdělení šumu je na externí a interní. Interní vzniká uvnitř analyzovaného obvodu a externí mimo obvod [8].

U tohoto typu generátoru je jako náhodný proces využit teplotní šum rezistoru. Důsledkem tepelného pohybu elektronů v rezistoru dochází k nepatrným a náhodným změnám napětí. Tento šum se zesiluje a ovládá napětím řízený oscilátor O_1. Kmitočet tohoto oscilátoru závisí na hodnotě napětí, které do oscilátoru vstupuje. Náhodnou veličinou je doba periody T. Impulzy oscilátoru slouží ke vzorkování impulzu druhého oscilátoru O_2. Frekvence oscilátoru O_2 je stabilizovaná a mnohonásobně vyšší než u prvního oscilátoru. Protože je doba periody T náhodná, nelze předpovědět, zda bude na O_2 impulz nebo nulové napětí. Bity vybrané vzorkováním jsou po dvojicích vedeny na vstup von Neumannova korektoru. V případě shody obou bitů, dochází k jejich odstranění. V opačném případě kolektor přenese hodnotu z O_2 na svůj výstup. Posloupnost vygenerovaná kolektorem je konečná [2]. Schéma generátoru využívající teplotní šum rezistoru znázorňuje obrázek 1.4.

U teplotního šumu rozlišujeme výstřelový šum, inverzní šum, okolní šum a kvantizační šum. Výstřelový šum vzniká rychlým pohybem elektronů v zařízeních, které jsou napájeny elektrickým napětím. Inverzní šum tvoří nečistoty ve vodivostním pásu polovodičů. Tento šum lze potlačit zvýšením frekvence. Kvantizační šum vzniká při zpracování analogového signálu a jeho převáděním do digitální formy. Okolní šum má nejvíce zdrojů, mezi nejčastější patří okolní zvuky, teplota, další šumy, nebo kombinace více faktorů [8].



Obr. 1.4: Schéma generátoru využívajícího teplotní šum rezistoru [2].

Šum lze softwarově i hardwarově redukovat. Hardwarovou redukcí může být analogové stínění a softwarovou redukcí výpočet průměrného signálu v určité oblasti [8].

Příkladem generátoru využívající tepelný šum rezistoru je **Intel RNG**. Tento generátor je implementovaný jako část počítačových procesorů s 64 bitovou architekturou. Vygenerované bity nejsou příliš náhodné a vyžadují následné zpracovaní. V tomto případě se používá modifikovaná von Neumannova metoda [1, 10].

1.2.2 Generátory založené na chaosu

Generátory založené na chaosu využívají opakované měření fyzických systémů. Měření fyzických vlastností v chaosu nepatří mezi nejpříznivější. Chaos nezaručuje náhodnost systému. Důvodem užívaní těchto generátorů je přesvědčení, že těžko popsatelné systémy se chovají náhodným způsobem. Toto tvrzení není prokázáno. Nejčastěji se používají generátory využívající optické, elektrické nebo optoelektrické chaotické systémy [1].

Nejrychlejší generátory jsou realizovány pomocí zpětnovazebního laseru. Světlo chaoticky kolísavé amplitudy je detekováno rychlou fotodiodou. Získanou amplitudu zpracovává 8bitový A/D převodník, který je schopen produkovat až 300 gigabitů za sekundu [1, str. 13]. Díky možnostem přidání drobných laserů, rezonátorů a aktivních a pasivních optických prvků na čip, lze tyto generátory zcela integrovat a snížit jejich spotřebu energie [1].

Pro generování náhodných čísel lze použít lasery využívající UWB (Ultra-Wideband - Ultraširoké pásmo). Kromě laserů generátor obsahuje vzorkovač a komparátor. Náhodný generátor využívající UWB je tvořen dvěma zpětnovazebními lasery L_1 a L_2. L_2 narušuje zpětnovazební smyčku laseru L_1. Tím se zvýší šířka pásma. Výstupní intenzita světla se odvádí pomocí rozdělovače paprsků a přesouvá se do optického vzorkovače s konstantní vzorkovací frekvencí. Vzorkovač porovnává vzorkovanou hodnotu intenzity světla s prahovou hodnotou. Na výstupu znamená nízká

intenzita světla „0“, zatímco vysoká intenzita znázorňuje „1“. Všechny části generátoru jsou na optické úrovni. V případě potřeby lze výstup transformovat do digitální podoby pomocí fotodiody a zesilovače [1].

Mezi kladné vlastnosti těchto generátorů patří rychlost generování bitů, možnost snadné integrace a nízká spotřeba energie. Negativní vlastností je neprokazatelnost, že těžko popsatelné systémy jsou náhodné. Bity získané pomocí těchto generátorů bývají zkreslené. K odstranění zkreslení se používá XOR dvou výstupních bitů ze dvou nezávislých generátorů. Výsledné sekvence bitů prochází relevantními statistickými testy [1].

ChaosKey TRNG je hardwarový generátor obsahující zdroj šumu a ARM SoC[3]. Tento generátor má všechen hardware, ovladače a další software dostupný pod volně šiřitelnou licencí [11].

1.2.3 Generátory založené na kruhových oscilátorech

Tato skupina generátorů se označuje jako FRO generátory (Free Running Oscillator - Volně běžící oscilátor). Tyto generátory lze považovat za speciální případ generátorů založených na šumu protože využívají tepelný šum oscilátoru, který způsobuje časové odchylky hrany signálu oscilátoru tzv. jitter[4]. Tyto odchylky způsobují nepředvídatelné chovaní generátoru [1, 12].

Generátory používají dva klopné obvody, na jejichž vstup je převeden signál z oscilátorů O_1 a O_2. Na první obvod je převeden signál z rychlého oscilátoru O_1 a na druhý obvod z pomalejšího oscilátoru O_2. Poměr rychlostí oscilátorů tvoří nepředvídatelnost. Perioda rychlosti O_1 se musí vejít do časového okna, které tvoří jitter O_2. Střídání signálu O_1 na datovém vstupu je $\frac{1}{2}$[5] [1, 12].

Tyto vlastnosti lze získat pomocí kruhového oscilátoru, tento oscilátor má dostatečnou rychlost a saturovaný výstup. Možností implementace kruhového oscilátoru je pomocí třístupňového invertoru[6], kdy výstup z posledního invertoru je vstup prvního (obrázek 1.5). Každý prvek přidává do obvodu pracovní zpoždění, které prodlužuje celkovou periodu [12].

Složitější implementací je spojení více kruhových oscilátorů. Toto spojení se skládá pomocí stromové struktury a hradel XOR[7]. Výstupem je signál obsahující

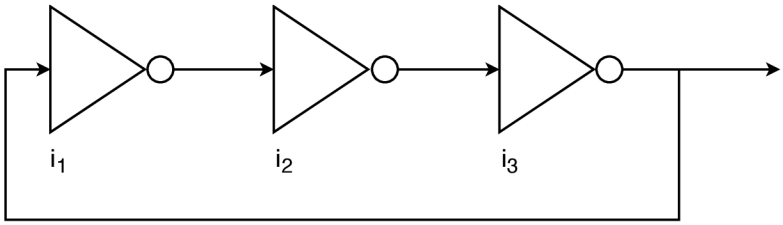
[3]ARM je označení pro architekturu procesoru. Tato architektura má nízkou spotřebu. SoC (System on Chip) znamená systém na čipu.

[4]Jitter je odchylka charakteristik periodického signálu. Tato odchylka je nežádoucí.

[5]Na datovém vstupu je stejná šance získat jedničku nebo nulu.

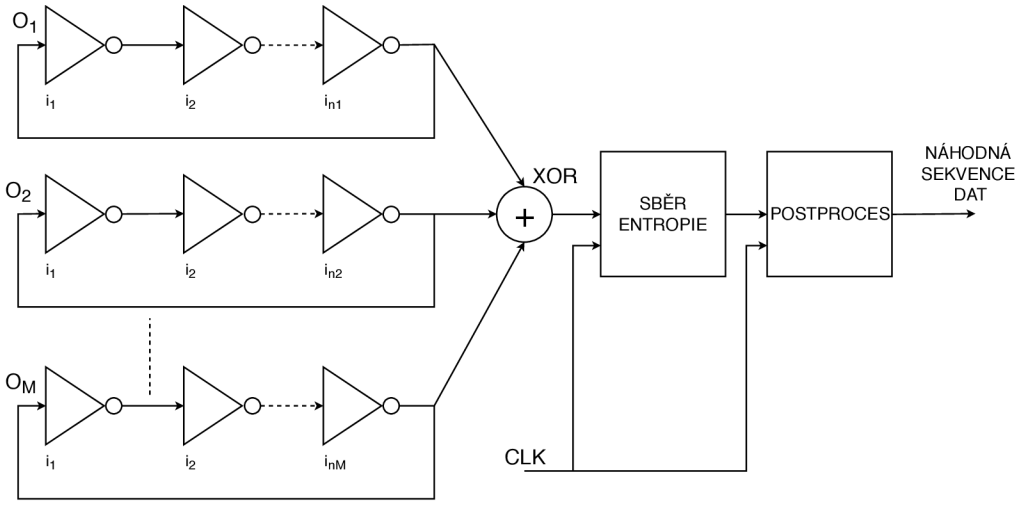
[6]Invertor je logický obvod, který realizuje funkci logické negace.

[7]Hradlo XOR je logický digitální elektronický obvod, jehož výstup je logický součet všech vstupů modulo počet vstupů.



Obr. 1.5: Třístupňový kruhový oscilátor sestavený z invertorů (i_1, i_2 a i_3) [1].

požadovanou entropii. Tento signál je vzorkován nekorelovanou frekvencí pro získání náhodných bitů - „sběr entropie“. Míra entropie záleží na návrhu oscilátorů, struktury XOR a vzorkování. U výsledné sekvence většinou dochází k vylepšení statistických vlastností pomocí postprocesu [12]. Postproces je popsán v kapitole 1.3. Generátor obsahující více kruhových oscilátorů znázorňuje obrázek 1.6.



Obr. 1.6: Schéma zapojení TRNG s větším množstvím kruhových oscilátorů [12].

Generátory založené na kruhových oscilátorech nejsou nákladné řešení. Vygenerovaná sekvence má nízkou úroveň entropie. Pozitivní vlastnost je snadná implementace na čipy, které se používají v různých řešeních kybernetické bezpečnosti. Negativní vlastností je, že nemohou nabídnout prokazatelnou náhodnost [1].

TRNG-IP-76 je hardwarový nedeterministický generátor vytvořený společností Inside Secure. Jako zdroj náhodnosti se používá výstřelový šum oscilátoru, který má nepředvídatelný výstup. Generátor obsahuje 8 FRO. Výstupy z oscilátorů jsou sloučeny pomocí hashovací funkce. Generátor je určený pro zabezpečení mobilních zařízení a IoT. TRNG-IP-76 je bezpečný a má nízkou spotřebu energie. Tato vlastnost je klíčová pro použití v IoT. Při generování náhodné sekvence není procesor napájen, protože generování entropie probíhá na pozadí, což snižuje spotřebu energie a šetří výdrž baterie [13].

1.2.4 Generátory založené na radioaktivním rozpadu

Vhodným fyzikálním jevem pro využití v TRNG je radioaktivní rozpad. Časové okamžiky, kdy se radioaktivní zdroj rozpadá jsou zcela nepředvídatelné a mohou být snadno detekovány a přeneseny do počítače [4].

Tyto generátory využívají chybu obsaženou v kvantových mechanických přírodních zákonech, které probíhají během radioaktivního rozpadu. Generovaná posloupnost je skutečně náhodná [14].

Vhodným příkladem generátoru využívajícího radioaktivní rozpad je **HotBits**. Jedná se o internetový zdroj náhodných čísel, který využívá jako zdroj záření cesium. Čísla jsou generována po sobě jdoucími páry radioaktivních rozpadů propojených s počítačem pomocí Geiger-Müllerovy trubice. Náhodnou sekvenci čísel lze objednat vyplněním formuláře na webové stránce `https://www.fourmilab.ch/hotbits/` s požadavkem na množství potřebných bytů. Požadavek je poslán na server, který odpovídá pomocí posloupnosti náhodných čísel. Tato čísla si žadatel ukládá. Komunikace se serverem využívá protokol HTTPS[8], který zajišťuje bezpečný přenos. Po přijetí všech bytů se čísla okamžitě zlikvidují, aby nemohla být opětovně poslána dalšímu žadateli [14].

Alternativou k HotBits může být Java balíček randomX. Počítačový software využívající balíček randomX může volit mezi různými pseudonáhodnými generátory a HotBits [14].

Nevýhodou tohoto typu generátoru je, že práce s radioaktivním materiálem je nebezpečná a vyžaduje vysokou míru opatrnosti, aby se předešlo nehodám. Výhodou je nemožnost předpovědět, kdy dojde k rozpadu částic [14].

1.2.5 Kvantové generátory

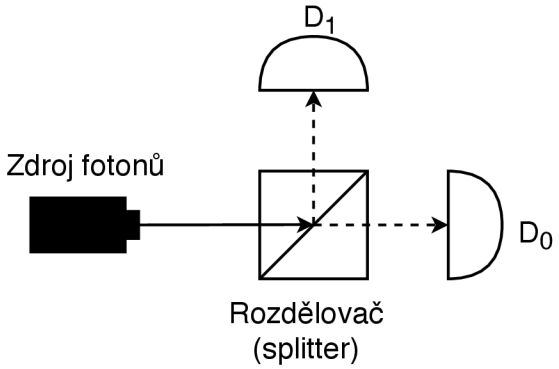
Kvantová mechanika je součást teoretické fyziky, která matematicky popisuje vesmír na atomové a subatomové úrovni. Generátory využívají skutečnost, že se při určitých okolnostech subatomové částice chovají náhodně. Stejně jako u generátorů založených na chaosu, které jsou popsané v části 1.2.2, se z důvodu neobjasnitelného jevu předpokládá, že je jev nedeterministický a náhodný [4].

K tvorbě náhodných bitů se využívají parametry z kvantových stavů světla. Jako zdroje světla se používají světelné diody, laser nebo fotonový zdroj. Tyto zdroje jsou cenově dostupnější a bezpečnější v porovnání s radioaktivním materiálem. Samotný optický generátor tvoří zdroj světla, rozdělovač (splitter) a dva detektory (D_0, D_1). Rozdělovač obsahuje polopropustný materiál, který polovinu vstupního paprsku odrazí a druhou polovinu propustí. V případě fotonů[9] polopropustný materiál foton

[8] Internetový protokol umožňující zabezpečenou komunikaci v počítačové síti.

[9] Foton je elementární nedělitelná částice.

propustí nebo odrazí s pravděpodobností 50 %. Tím dochází ke směrování světla na detektory D_0 a D_1. Detektor je zařízení citlivé na světlo, které detekuje fotony. V případě že dojde k propuštění fotonu rozdělovačem, zachytí ho detektor D_0 a na výstup se zapíše jednička. Pokud dojde k odrazu fotonu, zaznamená foton detektor D_1. Tento detektor zapíše na výstup nulu. Oba detektory mají rozdílnou efektivitu[10] spínání, tato vlastnost způsobí, že jeden zapisuje častěji než druhý, čímž dochází ke zkreslení náhodnosti vygenerované sekvence [15]. Generátor znázorňuje obrázek 1.7.



Obr. 1.7: Schéma kvantového generátoru [15].

Quantis TRNG je hardwarový generátor využívající elementární proces kvantové optiky. Generátor obsahuje polopropustné zrcátko, na které jsou posílány fotony. Fotony zrcátko odrazí nebo propustí. Dojde k detekci příslušným detektorem, který zapíše na výstup jedničku nebo nulu. Quantis TRNG je dostupný ve formě PCI Express karty nebo jako USB zařízení. Zařízení je průběžně monitorováno a v případě poruchy je generování zastaveno [16].

1.2.6 Generátory založené na chování uživatele

Pro generování náhodných čísel lze použít také lidský faktor. Tato softwarová řešení spoléhají na náhodnost zainteresovaných komponent, mezi které patří pohyb kurzoru po monitoru nebo stisk kláves na klávesnici [17].

Tyto generátory jsou náchylné na stereotypní chování uživatele. Čísla generovaná jedním uživatelem se časem mohou stát pseudonáhodnými. Pokud se uživatelé na zařízení budou střídat, lze získat zcela náhodná čísla [18].

Tento typ generátoru používá program **Randomgen**, který využívá znalost o poloze myši, velikost volné paměti a swapovací oblasti, ze kterých pomocí algoritmu získá číslo [19].

[10]Pravděpodobnost, že detektor foton opravdu zaznamená.

Při použití klávesnice pro generování hodnot lze použít hodnotu frekvence stisku kláves. Tyto hodnoty jsou zkreslené, protože nejdříve jsou stisky ukládány do bufferu a až poté se hromadně odesílají k vyhodnocení. Stlačení jednotlivých kláves program vyhodnotí, jako kdyby byly provedeny s minimálním rozestupem, což způsobuje zmíněné zkreslení [4].

V praxi se více používá metoda založená na pohybu kurzoru ve vyznačeném poli na monitoru. Tato metoda se používá například v bankách při generování bezpečnostních klíčů [4, 19].

U chytrých telefonů lze pro generování náhodné posloupnosti využít akcelerometr. Akcelerometr měří zrychlení při pohybu, naklonění zařízení. Toto generování může probíhat na pozadí při pohybu uživatele. Takto získaná náhodná čísla lze následně použít v případě potřeby [20].

VeraCrypt je bezplatný šifrovací open source program určený pro zabezpečení disků. Tento program obsahuje generátor náhodných čísel určený ke generování šifrovacích klíčů, soli[11] a klíčových souborů. Náhodné bity jsou získány z pohybu kurzoru, stisku kláves nebo ze statistik síťového rozhraní. VeraCrypt pro ukládání bitů používá „pool“ o velikosti 320 bytů. Před přidáním hodnoty do „poolu“ je získaná hodnota rozdělena na jednotlivé byty a je provedena operace modulo. Poté se hodnota zapíše na pozici kurzoru v „poolu“ a pozice se posune o jeden byte. V případě, že je kurzor na posledním místě v „poolu“ přesouvá se na pozici první. Po přidání šestnácti bytů do „poolu“ dochází k míšení všech hodnot v „poolu“. Před exportem výstupních dat dochází ke XORu hodnot a obrácení každého bitu (0 je změněna na 1 a 1 na 0) [17].

1.3 Postproces

Generátory náhodných čísel nejsou nikdy dokonalé. Proto je potřeba posloupnost vygenerovaných bitů upravit pomocí vhodných algoritmů. Tato úprava vede ke snížení nebo odstranění případné statistické závislosti. Myšlenkou postprocesu je obětovat určité procento bitů, za účelem vzniku menší, ale náhodnější sekvence. Zatímco získávání bitů pomocí TRNG je hardwarová záležitost, algoritmy pro zpracovávání jsou obvykle komplikované a je potřeba použít počítač, mikrokontrolér[12] nebo FPGA (Field Programmable Gate Array - Programovatelné hradlové pole). Mezi nejznámější a nejpoužívanější metody patří použití kryptografických hashovacích funkcí,

[11] Sůl je v kryptografii několik náhodných bitů, které slouží jako doplnění vstupu jednosměrných funkcí do požadované velikosti.

[12] Mikrokontrolér je integrovaný obvod, který představuje mikropočítač neboli jednočipový počítač [21].

extrakčních algoritmů a jednoduchých korektorů, které jsou určené pro úpravu náhodné bitové posloupnosti [1].

Hashovací funkce je jednosměrná[13] matematická funkce. Převádí vstupní množinu o n prvcích do binárního čísla, které má přesně daný počet bitů. Výstupní posloupnost se nazývá hash. Hash má nejčastěji od 128 do 512 bitů. Hlavní nároky na hashovací funkce jsou jednosměrnost a bezkoliznost 1. a 2. řádu [1].

Jednou z nejpoužívanějších technik pro úpravu náhodné posloupnosti je „bělení“ výstupu generátoru pomocí kryptografické hashovací funkce jako je MD5, SHA-1, SHA-2 a SHA-3. U většiny generátorů probíhá hashování pomocí počítače. Mezi výjimky, kdy je hash tvořen pomocí hardwarového čipu patří **Intel RNG** a **VIA C3**, oba generátory využívají funkci SHA-1 [1].

Extrakční algoritmus převádí dlouhou, slabě náhodnou sekvenci, do sekvence kratší s téměř dokonalou náhodností. Existuje mnoho druhů extraktorů, ale u žádného není prokázáno, že funguje podle požadovaných představ. Problém spočívá v paměťové náročnosti a vysokém využití procesoru, který vede ke zpomalení výstupní rychlosti generování bitů. Účelem těchto algoritmů je eliminovat změny TRNG způsobené stárnutím, teplotními změnami nebo útoky [1].

Ad hoc korektory jsou jednoduché korektory určené pro úpravu náhodných bitových posloupností. Příkladem je XOR dvou nebo více sousedních bitů z jednoho generátoru, XOR bitů z více generátorů, vynechávání bitů a von Neumannovo schéma. Tyto úpravy vždy nevedou k větší náhodnosti a mohou tvořit statistické nedostatky. Příkladem je von Neumannovo schéma, kde je bitová sekvence rozdělena na dvojice. Dvojice obsahující bity „11“ a „00“ se odstraňují. Dvojice „01“ je převedena na „0“ a „10“ na „1“. V případě, že generátor vygeneruje posloupnost „1010101010“, korektor tuto posloupnost změní v „11111“. Upravená posloupnost je zkreslená a statisticky závislá [1].

[13] U jednosměrných funkcí nelze dopočítat vstupní data z výstupních bitů. Jediným řešením je náhodný odhad.

2 Základní vlastnosti generátorů

Následující tabulka 2.1 porovnává vlastnosti generátorů pravých a pseudonáhodných čísel. Tabulka je převzata z [4].

Tab. 2.1: Srovnání PRNG a TRNG.

Vlastnost	**PRNG**	**TRNG**
Účinnost	Výborná	Slabá
Deterministický X nedeterministický	Deterministický	Nedeterministický
Opakovatelnost	Periodická	Neperiodická

Jak je vidět v tabulce 2.1, PRNG jsou schopny vyprodukovat více čísel v kratší době. Tato vlastnost je dobrá pro případy, kdy je potřeba velké množství čísel. Typickou vlastností je opakovatelnost stejné sekvence. U moderních generátorů, je tato vlastnost zanedbatelná, protože doba jedné periody je dostatečně dlouhá. Pravé náhodné generátory jsou pomalejší, protože získaná data se ještě upravují pomocí algoritmů. TRNG jsou oproti pseudonáhodným generátorům neperiodické. Tato vlastnost ovšem neplatí u všech typů TRNG. Například generátory založené na lidském chování se mohou po určité době změnit na periodické, protože lidské jednání je stereotypní. Konstrukce TRNG je složitější o dodatečnou hardwarovou část. Generátory jsou pomalejší, protože je potřeba dodatečné zpracování hodnot.

V následujících podkapitolách jsou popsány vlastnosti pouze pro TRNG.

2.1 Rychlost generování

Rychlost generování je schopnost generátoru vyprodukovat určité množství náhodných bitů za jednotku času. Rychlost závisí na složitosti generátoru. Čas potřebný k získání sekvence nelze snadno snižovat. Snižování složitosti hardwaru vede ke změnám kvality vygenerovaných čísel [22].

Tabulka 2.2 ukazuje rychlosti TRNG generátorů. U všech generátorů je uvedena rychlost generování bitů. Po vygenerování dochází k postprocesu, který je časově náročnější než generování. Postproces je popsán v kapitole 1.3 a doba postprocesu záleží na požadovaných vlastnostech výstupní sekvence bitů.

ChaosKey TRNG má maximální rychlost generování omezenou na přenosovou rychlost dat pomocí USB [23].

U **random.org** je uvedeno množství vygenerovaných bitů pouze jednou z radiostanic, které tato společnost vlastní [4].

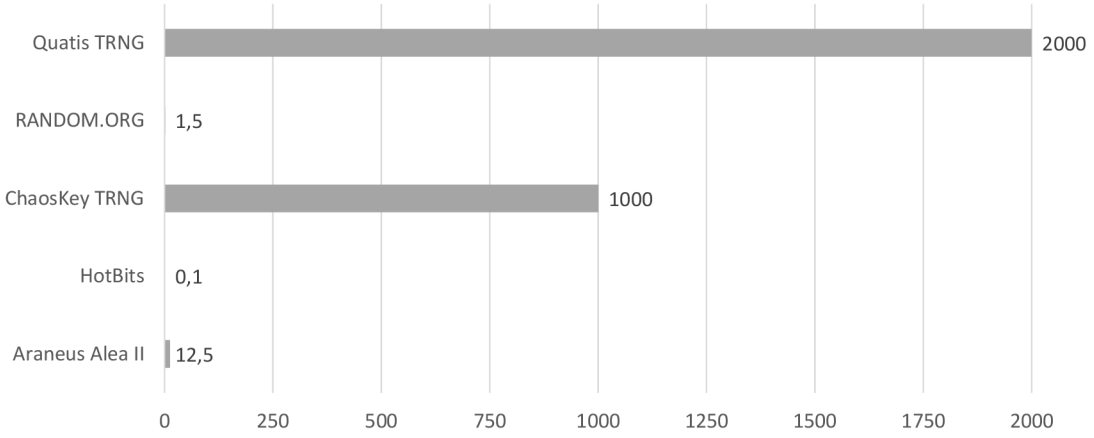
Quantis TRNG je schopen generovat proud bitů o rychlosti 4 Mbps a 16 Mbps. Rychlost záleží na připojení generátoru k zařízení. U připojení přes USB port se

Tab. 2.2: Rychlost generování náhodných čísel podle použité technologie.

Generátor	**Fyzikální princip**	**Rychlost**
Araneus Alea II	bílý šum	100 kbps [9]
HotBits	radioaktivní rozpad	100 Bps [14]
ChaosKey TRNG	chaos vytvořený šumem	1 MBps [23]
Intel RNG	teplotní šum rezistoru	neuvedeno
Random.org	atmosférický šum - meteorologický jev	12 000 bps [4]
Quatis TRNG	elementární proces kvantové optiky	až 16 Mbps [16]
TRNG-IP-76	šum vytvořený pomocí kruhových oscilátorů	neuvedeno

jedná o 4 Mbps. Při připojení pomocí PCI Express dochází k proudu o 4 Mbps a 16 Mbps [9].

Obrázek 2.1 znázorňuje hodnoty z tabulky 2.2. Všechny hodnoty byly převedeny na kiloByty za sekundu (kBps).



Obr. 2.1: Graf znázorňující rychlost jednotlivých generátorů v kBps.

2.2 Hardwarové nároky

TRNG pro svoji funkci potřebují speciální hardware připojený k zařízení, nebo internetové spojení. Jednotlivé nároky jsou uvedeny v tabulce 2.3.

HotBits používá radiační detektor od firmy Aware Electronics model RM-80, který je k počítači připojen pomocí sériového portu. Nyní se jedná o třetí generaci HotBits generátorů. Tato verze je spuštěná od roku 2006 a využívá Linux s distribucí Fedora 5. Předchozí dvě generace z roku 1996 a 1998 využívaly systém Windows [14].

Tab. 2.3: Ukázka hardwarových nároků vybraných generátorů.

Generátor	**Transport dat**	**Nároky (potřebný hardware)**
Araneus Alea II	USB port	zdroj šumu
HotBits	internetová služba	radiační detektor
ChaosKey TRNG	USB port	zdroj šumu a ARM SoC
Intel RNG	součástí vybraných počítačových procesorů	zdroj šumu
Random.org	internetová služba	radiostanice zaznamenávající atmosférický tlak
Quatis TRNG	PCI Express port USB port	měřič kvantového rozpadu
TRNG-IP-76	zabudovaný v zařízení	FRO tvořící šum

2.3 Kryptografické zabezpečení

U dokonalého kryptograficky bezpečného generátoru by nemělo jít žádným způsobem výstupní čísla předvídat.

U PRNG nelze docílit, aby nebylo možno výslednou posloupnost předpovědět. Generátory pseudonáhodných čísel, které jsou dostatečně bezpečné se označují jako CSPRNG (Cryptographically secure pseudorandom number generator - Kryptograficky bezpečný pseudonáhodný generátor čísel) [24].

V dnešní době jsou většinou kryptograficky bezpečné TRNG a CSPRNG. Zároveň ale neplatí, že každý TRNG je bezpečný. Bezpečnost je potřeba začlenit do návrhu hardwaru generátoru. Je potřeba odstranit rušivé elementy, které mohou udělat výstupní sekvenci předvídatelnou.

CSPRNG se používají ve většině aplikací, které využívají TRNG pro zisk entropií, který nemusí být kryptograficky bezpečný. V případě, že je TRNG kryptograficky bezpečný nesmí být možné předpovědět výstupní sekvenci ani při znalosti vstupních hodnot [24].

3 Užití náhodných čísel

Vlastnosti uvedené výše dělají TRNG vhodné pro hazard, loterii a kryptografii. Naopak snadná implementace a rychlost dělá PRNG vhodnější na simulace a modelování. Následující tabulka 3.1 ukazuje, který generátor se více hodí pro dané případy [4].

Tab. 3.1: Vhodnost TRNG a PRNG generátorů pro dané aplikace [4].

Použití	**Vhodný generátor**
Loterie	TRNG
Hry a gamblerství	TRNG
Simulace a modelování	PRNG
Bezpečnost (generování klíčů, ...)	TRNG
Umění	TRNG i PRNG

3.1 Užití náhodných čísel v kryptografii

Kryptografie vznikla s potřebou utajit informace před neoprávněnými osobami. Jedná se o systém algoritmů, který slouží k zajištění důvěrnosti přenášených zpráv. Šifrování v počítačové technice je realizované s použitím náhodných čísel. Existují dva typy šifrování: **symetrické** a **asymetrické** [2].

3.1.1 Symetrická kryptografie

U symetrických šifrovacích algoritmů se používá sdílený klíč pro zašifrování na straně odesílatele a dešifrování na straně příjemce. Výhodou je rychlost šifrování i pro velké množství dat. Nevýhodou je složitá distribuce klíče a nutnost utajení klíče. Existují dva typy symetrické šifry: proudová a bloková [2].

Proudová šifra zpracovává zprávu po bitech. Základní operací je XOR jednotlivých bitů zprávy s jednotlivými bity hesla. Příkladem proudové šifry je Vernamova šifra [2].

Bloková šifra rozděluje zprávu na bloky o délce 64, 128 nebo 256 bitů. Konstrukce těchto šifer vychází z konceptu kaskádové šifry, kdy dochází ke zřetězení. Výstupní blok jedné šifry je vstupním blokem pro následující šifry. Příkladem blokové šifry je AES [2, 25].

3.1.2 Asymetrická kryptografie

Asymetrické kryptografické algoritmy využívají veřejný a soukromý klíč. Výhodou je jednoduchá distribuce veřejných klíčů a nepopiratelnost autora zprávy. Nevýhodou je pomalé šifrování. Asymetrické kryptosystémy využívají problém faktorizace velkých čísel a problém řešení diskrétních logaritmů [2].

Mezi asymetrické kryptosystémy patří RSA. Tento systém je pojmenován po tvůrcích Rivest, Shamir a Adleman. RSA se používá pro šifrování a elektronický podpis [2].

Elektronický podpis slouží k nahrazení klasického podpisu. Elektronický podpis se připojuje k elektronickým datům a slouží k jednoznačnému ověření identity osoby, která data podepsala. Při podepisování se využívá privátní klíč k podpisu a veřejný klíč k ověření elektronického podpisu [2].

3.2 Generátory vhodné pro použití v PC

Většina příkladů generátorů náhodných čísel uvedených v části 1.2 je vhodná pro připojení k počítači. Generátory lze zabudovat do hardwaru počítače nebo přípojit pomocí USB nebo PCI sběrnice.

U generátorů založených na šumu je nevýhodou rušení zdroje entropie dalšími komponenty PC, které vytváří šum. Tento šum může zkreslovat výslednou sekvenci. Příkladem generátoru založeného na šumu je **Intel RNG**. Tento generátor je součástí vybraných počítačových procesorů [8, 10].

Další možností jsou generátory založené na chaosu a na kruhových oscilátorech. Výhodou u obou typů generátorů pro použití v PC je možnost zabudování na čip. Nevýhodou generátorů využívajících kruhové oscilátory je nízká úroveň entropie [1].

Bez dodatečného hardwaru jsou generátory založené na chování uživatele. Tyto generátory nejsou vždy zdrojem kvalitní náhodné sekvence. Hlavním důvodem je stereotypní jednání uživatelů [18].

Poslední možností je použití internetových serverů poskytující náhodné sekvence. V tomto případě je potřeba internetové připojení do veřejné sítě. Mezi webové stránky poskytující náhodná čísla patří **random.org**, **HotBits** a **LavaRand**. HotBits poskytují náhodné sekvence se zaručenou vysokou úrovní entropie. HotBits využívá radioaktivní rozpad. LavaRand jako zdroj náhodné sekvence používá snímky lávových lamp [4, 14, 26].

Pro zabezpečenou komunikaci se serverem lze použít šifrované spojení pomocí protokolu HTTPS. Zabezpečenou komunikaci umožňuje server random.org a HotBits, kde je šifrován požadavek uživatele na server a náhodná vygenerovaná data posílaná zpět k uživateli [14, 27].

3.3 Generátory používané v internetu věcí

IoT je propojení různých zařízení pomocí internetu bez účasti člověka. V této síti mohou být fyzická zařízení, vozidla, domácí spotřebiče, atd. IoT nachází uplatnění ve všech oblastech lidského života jako je maloobchod, zdravotnictví, průmysl, energetika, logistika, atd. IoT zařízení jsou vybavena softwarem, senzory, pohyblivými částmi a síťovou konektivitou, která těmto zařízením umožňuje navzájem si vyměňovat data. Získaná data jsou předávána za účelem dalšího zpracování a vyhodnocení. Dnes lze díky IoT náramkům zachraňovat nemocné pacienty, ovládat centrální systém domů nebo jednotlivé prvky zabezpečení budov jako jsou alarmy, požární čidla, kamerový systém a další zařízení [28].

V dnešní době tvoří IoT miliardy výpočetně omezených zařízení, které jsou bezdrátově propojené se vzdálenými systémy [29].

Zařízení, která jsou součástí IoT, potřebují zabezpečení, které ochrání uživatele před ztrátou dat a sníží rizika související s ukládáním hesel nebo jejich ověřováním. U generátorů náhodných čísel v internetu věcí je potřeba, aby měl malou spotřebu energie pro generování náhodné sekvence a malé rozměry. U IoT zařízení se nejčastěji setkáváme s generátory využívající šum [5, 30, 31].

Generátory založené na šumu jsou vhodné, protože je lze snadno zabudovat na čip. Nevýhodou u IoT mohou být jiné zdroje šumu, které u těchto generátorů vedou ke zkreslení. Další nevýhodou je potřebný postproces [1, 8].

Kromě zabudovaného generátoru lze náhodnou sekvenci získat pomocí webových stránek mezi které patří **random.org** a **HotBits** [4, 14].

Další z generátorů uvedených v kapitole 1.2 jsou generátory založené na chaosu. Pro použití v IoT je vhodná snadná implementace drobných laserů na čip, malá spotřeba energie a rychlost generování. Hodnoty vygenerované pomocí těchto generátorů jsou zkreslené a potřebují dodatečnou úpravu [1].

Příkladem generátorů využívající FRO je **TRNG-IP-76**, který je navržený přímo pro použití v internetu věcí. TRNG-IP-76 více popisuje kapitola 1.2.3. Mají malou spotřebu, jsou snadno zabudovatelné na čip a levné. První dvě vlastnosti jsou pro IoT vhodné. Nevýhodou je nízká entropie výstupní sekvence [12, 13].

Quantis QRNG Chip je kvantový generátor navržený pro IoT. Tento čip vyrobený firmou ID Qutantique je malý, rychlý, má malou spotřebou energie a je jednoduché ho přidat do IoT zařízení. Rozměr čipu je 4,2 x 5,0 x 1,1 mm, rychlost generování náhodných bitů je 4,9 Mbps [32].

Náhodná čísla v IoT mohou být kromě výše uvedených generátorů také generována pomocí snímaných dat. Množství snímaných dat je závislé na senzorech, které detektor vlastní [30].

U nositelných nebo implantovatelných zařízení lze kromě fyzikálních jevů snímat také zrychlení, úhlovou rychlost, lidské chování nebo magnetické pole za účelem generování pravých náhodných čísel. Generátory založené na chování uživatele nejsou u nositelných IoT zařízení vhodné řešení, protože lidé mají stereotypní chování. Není vhodné používat hodnoty vygenerované pomocí chůze [31].

4 Testování generátorů náhodných čísel

V průběhu používání generátoru se mohou měnit jeho statistické vlastnosti. Z tohoto důvodu se sekvence výstupních bitů statisticky testují. Testování se provádí ve stanovených časových intervalech, v případě vysokých nároků na bezpečnost se provádí před každým generováním náhodné sekvence. Existuje velké množství statistických testů, které se v bitové sekvenci snaží najít periodu [2].

Pro testování se často využívají předpřipravené softwarové balíky. Nejznámější balíky testů jsou testy **NIST STS** (The NIST Statistical Test Suite - Soubor statistických testů vytvořených NIST) nebo **Diehard**. Testy Diehard byly dříve velmi využívané, dnes jsou zastaralé. NIST STS je soubor zaměřený hlavně na kryptografickou bezpečnost [33]. Více o testech NIST je v kapitole 4.1.

Při testování se výsledná hodnota porovnává s kritickou hodnotou, která je definovaná pro daný test. Ve většině případů se jedná o hodnotu 0,01. Výsledná hodnota zkoumané testované sekvence by měla být menší nebo rovna kritické hodnotě. Pokud je kritická hodnota překročena, znamená to, že testovaná sekvence bitů není náhodná [34].

I když generátor projde všemi známými statistickými testy, není zaručeno, že vygenerovaná sekvence bitů je náhodná. Pokud dnes uspěje ve všech testech, neznamená to, že daný výsledek se bude opakovat i zítra. Většina testů náhodnosti kontroluje jednu nebo více statistických vlastností sekvencí náhodných čísel. Některé testy jsou více zaměřené na PRNG, jiné na TRNG, některé mají obecnou povahu. Příkladem obecného testu je NIST STS. Podle hardwarových vlastností počítače, délky sekvence a zvoleného testu se odvíjí doba testování. Testy jsou časově náročnější než samotné generování čísel. Testování sekvence o délce 1×10^9 bitů pomocí NIST STS s využitím jednojádrového procesoru trvá asi 6 hodin [1].

4.1 NIST

NIST (National Institute of Standards and Technology - Mezinárodní instituce standardů a technologií v USA) vytvořila statistické testy pro testování náhodných čísel vytvořených generátory. Definuje konstrukční a zkušební kritéria, která musí RNG (Random Number Generator - Generátor náhodných čísel) splňovat, aby byl považován za vhodný pro kryptografické aplikace. Požadavky jsou popsané v SP 800-22 a SP 800-90 [5].

NIST STS je soubor 15 testů, který byl vyvinut pro testování náhodnosti u libovolně dlouhých binárních souborů vytvořenými softwarovými nebo hardwarovými generátory. Jednotlivé testy mohou mít více částí [33].

1. **Frekvenční test (Frequency Test)** [33]
   Zaměřuje se na poměr počtu jedniček a nul v celé sekvenci. Počet jedniček by se v poměru k celé sekvenci měl blížit k $\frac{1}{2}$. Všechny následující testy závisí na splnění tohoto testu.
2. **Blokový frekvenční test (Frequency Test within a Block)** [33]
   Podobný jako je *Frekvenční test*. Rozdílem je, že se netestuje celá sekvence, ale M-bitové bloky. Účelem testu je určit, zda je četnost jedniček v jednotlivých M-blocích rovna $\frac{M}{2}$, kde M je počet bitů v jednom bloku.
3. **Test sekvencí bitů (Run Test)** [33]
   Zaměřuje se na celkový počet běhů[1] v pořadí. Účelem testu je určit, zda střídání po sobě jdoucích sekvencí nul nebo jedniček je různorodé a není příliš rychlé nebo příliš pomalé.
4. **Test na nejdelší sekvenci jedniček v jenom bloku (Test for the Longest Run of Ones in a Block)** [33]
   Účelem je zjistit, zda je délka nejdelší sekvence po sobě jdoucích jedniček v M-bitových blocích stejná, jako délka očekávaná v náhodné sekvenci.
5. **Test sérií binárních matic (Binary Matrix Rank Test)** [33]
   Je zaměřen na lineární závislosti mezi částmi sekvence, které mají pevnou délku.
6. **Test diskrétní Fourierovy transformace (Discrete Fourier Transform Test)** [33]
   Určuje, zda má sekvence pravidelné rysy[2] napříč celou posloupností. Test se provádí pomocí diskrétní Fourierovy transformace. Každý bit má přiřazenou výšku (pík). Pokud počet píků se stejnou výškou nepřekročí prahovou hodnotu 5 %, lze sekvenci nazvat za náhodou.
7. **Test nepřekrývajících se vzorů (Non-overlapping Template Matching Test)** [33]
   Zaměření testu je počet výskytů předem určených cílových vzorů o m-bitové délce. Vzor je řetězec bitů, který je obsažen v pracovním okně, které má velikost M-bitů. Toto okno vyhledává konkrétní vzor v sekvenci. Pokud se řetězec v sekvenci nevyskytuje, okno se posune o jeden bit dále. V případě, že se vzor vyskytuje, okno se posune o jeden bit po nalezeném vzoru a hledání se obnoví.
8. **Test překrývajících se vzorů (Overlapping Template Matching Test)** [33]
   Pracuje na stejném principu jako *Test nepřekrývajících se vzorů*. Pokud je hledaný vzor nalezen, celé okno se však místo posunu o řetězec posouvá o jeden bit.

[1] Běh je nepřerušená sekvence identických bitů.
[2] Pravidelné rysy jsou opakující se vzory blízko sebe.

9. **Mauerův „univerzální statistický“ test (Maurer's „Universal Statistical“ Test)** [33]
   Zaměřuje se na počet bitů mezi shodnými vzory. Kontroluje, zda je možné sekvenci komprimovat bez ztráty informace. Pokud je možná velká komprimace, je sekvence považována za nenáhodnou.
10. **Test lineární složitosti (Linear Complexity Test)** [33]
    Zaměřuje se na délku LFSR (Linear-feedback shift register - Posuvný registr s lineární zpětnou vazbou[3]). Účelem tohoto testu je zjistit, zda je sekvence dostatečně složitá.
11. **Test sérií (Serial Test)** [33]
    Zaměřením testu je frekvence všech možných překrývajících se M-bitových vzorů v celé sekvenci. Zjišťuje se, zda je počet výskytů překrývajících se vzorků přibližně stejný, jak se u náhodných sekvencí očekává.
12. **Přibližný test entropie (Approximate Entropy Test)** [33]
    Stejně jako u *Testu sérií* je tento test zaměřený na frekvenci všech možných překrývajících se vzorů o délce m-bitů. Porovnává se frekvence dvou po sobě jdoucích bloků, které se překrývají. Výsledek se porovnává s očekávaným výsledkem pro náhodnou sekvenci.
13. **Test kumulativních součtů (Cumulative Sums Test)** [33]
    Zaměřuje se na maximální odchylku od nuly pro kumulativní součet všech bitů. Bity s hodnotou 1 reprezentuje kladná jednička, bity s hodnotou 0 záporná jednička. První součet tvoří první bit. Každý následující součet je tvořen všemi předcházejícími bity a aktuálním bitem. Cílem testu je určit, zda kumulativní součet testované sekvence není příliš malý nebo velký vzhledem k očekávanému chování náhodné sekvence. Pokud bude výsledek blízký nule, jedná se o náhodnou sekvenci.
14. **Test náhodných návštěv (Random Excursions Test)** [33]
    Zaměřuje se na počet cyklů, které jsou během kumulativního součtu s náhodným průchodem navštíveny Kkrát. Každý cyklus má náhodně vybranou délku. Kumulativní součet je odvozen z částečných součtů poté, co jsou jednotlivé bity sekvence transformovány z (0,1) na (-1,1). Cílem je určit, zda se počet návštěv jednotlivých stavů liší od očekávaných hodnot pro náhodné sekvence. Tento test představuje sérii 8 testů. V každé části se testuje jeden ze stavů: -4, 4, -3, 3, -2, 2, -1 a 1. Sekvence musí obstát ve všech osmi částech.

[3]Posuvný registr s lineární zpětnou vazbou má lineárně závislý výstup na předchozích výstupech a na počátečním stavu. Využívá se pro generování PRNG a je součástí proudových šifer.

15. **Test náhodných variant návštěv (Random Excursions Variant Test)** [33]
    Zaměřuje se na celkový počet návštěv jednotlivých stavů při kumulativním součtu. Test určuje odchylku od očekávaného počtu návštěv během náhodného průchodu sekvencí. Test se skládá z 18 testů. Testují se stavy: -9, 9, -8, 8, . . . , -2, 2, -1, 1. Sekvence musí obstát ve všech osmnácti částech.

4.2 Dieharder

Dieharder je určený pro testování RNG. Tento test by měl ulehčit testování TRNG využívaných v kryptografii a výzkumu. Obsahuje 3 základní části: 17 Diehard testů, 3 testy převzaté z NIST STS (testy NIST STS jsou popsány v kapitole 4.1) a 8 nových testů vytvořených Robertem G. Brownem [35].

5 Návrh generátoru pravých náhodných čísel

Součástí této práce je návrh a realizace vlastního generátoru pravých náhodných čísel. Realizace generátoru je uvedena v kapitole 6. Testování výsledné sekvence, výpočetní náročnost aplikace a časová náročnost generování je v kapitole 7. Jako zdroj entropie je zvolen pohyb kurzoru. Dnes je kurzor součástí operačních systémů obsahujících grafické uživatelské rozhraní, jedná se tedy o laciné a vhodné řešení.

5.1 Základní technické parametry

- Platforma:
  - Microsoft Windows 10.
- Zobrazení:
  - Na šířku.
- Jazyk:
  - Čeština.
- Programovací jazyk:
  - Java.
- Vývojové prostředí:
  - Eclipse s vizuálním nástrojem JavaFX Scene Builder.

Pro tvorbu aplikace, která pomocí chování uživatele generuje náhodná čísla, byl zvolen programovací jazyk Java. Hlavním důvodem volby je možnost rozšíření aplikace na jiné platformy, než je OS Windows. Druhý důvod je osobní. Mé zkušenosti s programováním jsou nejrozšířenější u programovacího jazyka Java.

Pro tvorbu aplikace byla použita open source vývojová platforma Eclipse, která je určena k programování v jazyce Java. Základní funkce platformy lze rozšířit pomocí modulů o další funkce - nové typy editorů, podpora pro další programovací jazyky (C/C++, PHP, HTML, XML, JavaScript nebo Python), nové ladicí nástroje, návrh grafického uživatelského rozhraní, napojení na aplikační servery atd. Platforma byla vytvořena firmou IBM, která ji v roce 2001 vydala pod licencí EPL[1] pro volné použití. Eclipse lze použít na operačních systémech Windows, Linux, Solaris a dalších [36]. Eclipse je zadarmo ke stažení na webové stránce `https://www.eclipse.org/downloads/`.

Uživatelské rozhraní je vytvořeno pomocí vizuálního nástroje JavaFX Scene Builder. Tento nástroj je určený k vytváření grafického uživatelského rozhraní v aplikacích využívající JavaFX.

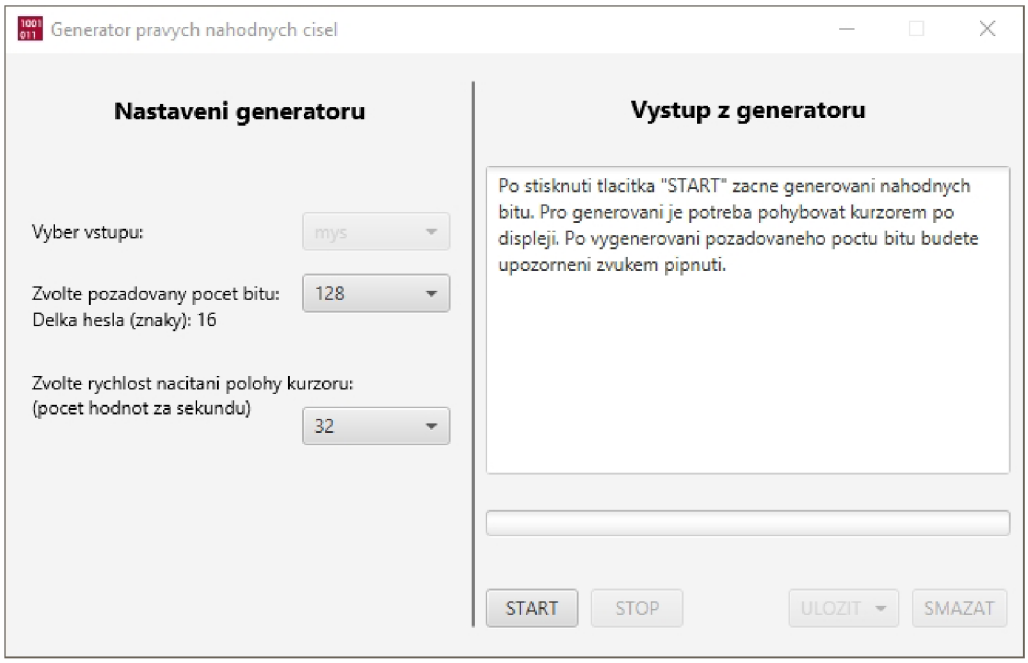
[1]EPL je Eclipse Public Licence dostupná na stránce `https://www.eclipse.org/legal/epl-v10.html`.

Hlavní myšlenkou návrhu je použití lidského faktoru jako zdroje náhodného chování. Stejnou myšlenku využívá VeraCrypt. Návrh byl inspirován generátory, které jsou popsány v kapitole 1.2.6.

5.2 Struktura aplikace

Aplikace pro generování se skládá ze dvou částí. První část načítá a zpracovává data získaná pomocí uživatele. Mezi zpracovávaná data patří souřadnice umístění kurzoru a čas od zapnutí aplikace po spuštění generování náhodných čísel.

Druhá část aplikace má GUI (Graphical User Interface - Grafické uživatelské rozhraní). Návrh uživatelského rozhraní znázorňuje obrázek 5.1. GUI se skládá ze dvou částí, pravé a levé půlky.



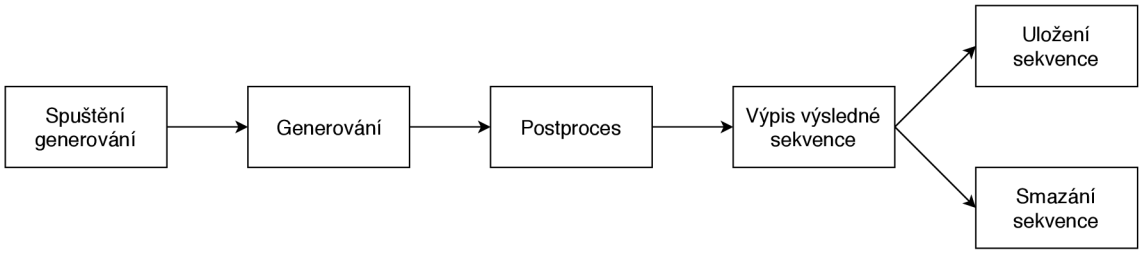
Obr. 5.1: Grafické uživatelské rozhraní aplikace.

Levá část „Nastavení generátoru“ umožňuje uživateli nastavit, jakou rychlostí se mají daná data generovat, jak velká má být výsledná vygenerovaná sekvence a velikost vygenerovaného hesla. Poslední volitelnou položkou je výběr vstupu. Tato položka zatím není aktivní, protože návrh obsahuje pouze jednu možnost vstupu dat. Tato položka umožní rozšíření aplikace.

Pravá půlka „Výstup z generátoru“ obsahuje velké pole, kde jsou pokyny pro uživatele během generování. Po vygenerování sekvence se do pole vypíše výsledná posloupnost, vygenerované heslo a doba, za kterou byla sekvence vygenerovaná. Kromě

pole tato část obsahuje i ProgressBar a čtyři tlačítka. ProgressBar ukazuje, kolik bitů se již vygenerovalo z celkového požadovaného množství. Tlačítka „START“ a „STOP“ slouží ke spuštění a zastavení generování čísel. Tlačítko „ULOŽIT“ slouží k uložení výsledné posloupnosti do TXT souboru. Po kliknutí na „ULOŽIT“ má uživatel možnost zvolit, zda chce uložit vygenerovanou sekvenci i heslo, nebo jenom heslo popřípadě jenom sekvenci. Poslední je tlačítko „SMAZAT“. Po stisku tohoto tlačítka dochází ke smazání celé vygenerované sekvence a hesla.

Proces generování pomocí aplikace zobrazuje obrázek 5.2. Náhodné generování se spouští stisknutím tlačítka „START“. Po spuštění se pomocí pohybu kurzoru generují náhodná data. Každé číslo je vytvořeno ze získaných hodnot pomocí algoritmu, který znázorňuje vývojový diagram na obrázku 5.4.



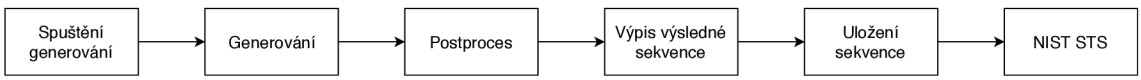
Obr. 5.2: Schéma znázorňující proces generování náhodných čísel pomocí aplikace.

Po vygenerování dvou bitů dochází k jejich úpravě. Důvodem je odstranění případných statistických závislostí. Bity sekvence se upraví pomocí von Neumannova korektoru. Tento korektor je popsán v kapitole 1.3. Převádí dvojici bitů na jeden bit (jedničku nebo nulu) nebo dvojici odstraní.

Po vygenerování požadovaného počtu bitů dojde k převedení bitů na heslo a poté se výsledná sekvence a heslo vypíše na displej vpravo do pole pro výpis.

Vygenerovanou posloupnost a heslo lze uložit do TXT souboru nebo smazat. O této akci a možnostech uložení rozhodne uživatel pomocí uživatelského rozhraní.

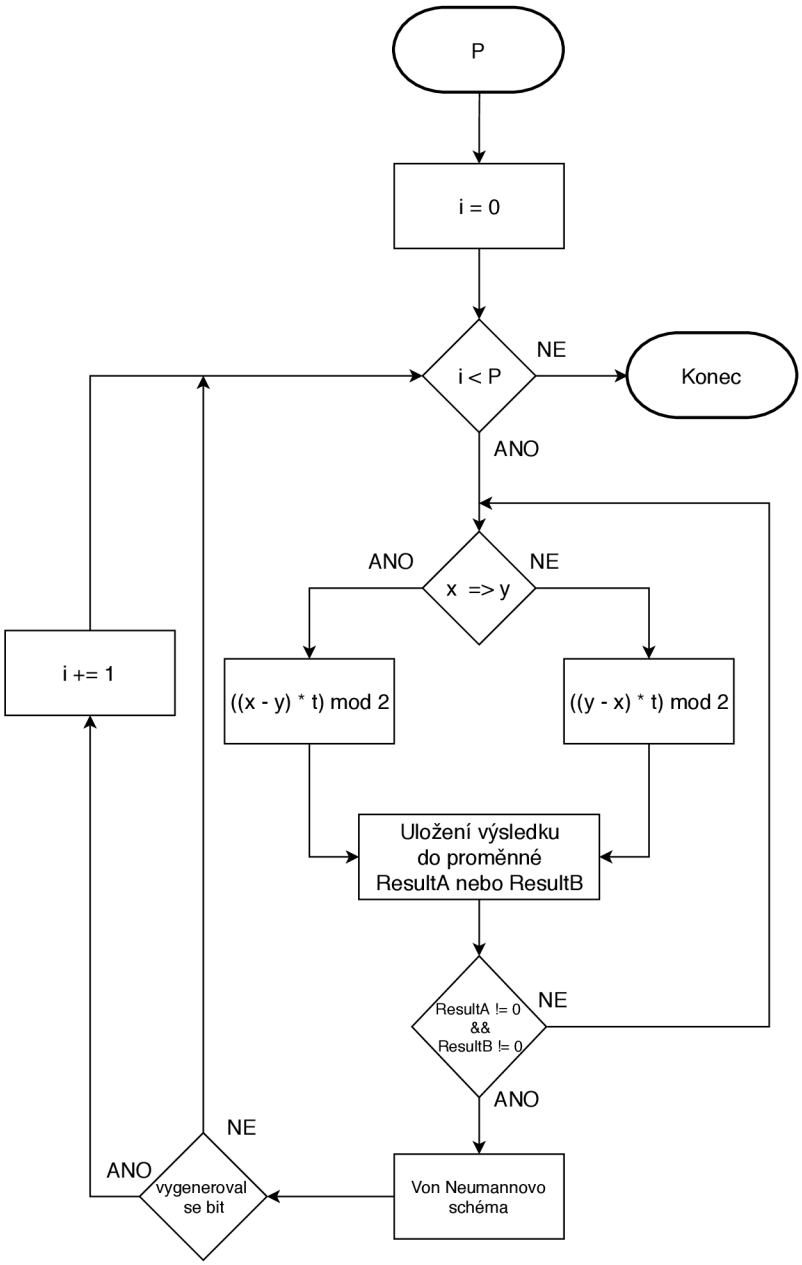
Pro testování aplikace je schéma rozšířené o testování výsledné sekvence pomocí NIST STS. Schéma generování i s testováním znázorňuje obrázek 5.3.



Obr. 5.3: Schéma znázorňující proces generování náhodných čísel a testování výsledné sekvence.

Aplikace je vytvořena v programovacím jazyce Java. Tento programovací jazyk podporuje snímání pohybu a umístění kurzoru. Kurzor je snímán pomocí rozhraní `MouseMotionListener` [37].

Další veličinou, se kterou se pracuje je rozdíl času od zapnutí aplikace po spuštění generování čísel. Čas v nanosekundách se získá pomocí příkazu `System.nanoTime()`.



Obr. 5.4: Vývojový diagram znázorňující generování náhodných čísel.

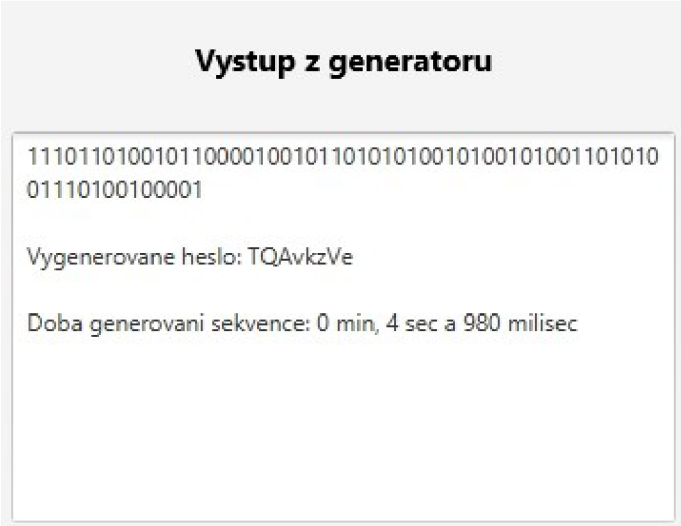
Obrázek 5.4 znázorňuje proces generování náhodných bitů. P znázorňuje počet požadovaných vygenerovaných bitů. Tento počet je zadán uživatelem pomocí levé strany uživatelského rozhraní „Nastavení generátoru“. Počet bitů lze vybrat z možností 16, 32, 64, 128, 256, 512, 1024 nebo 2048. Defaultně je nastavena možnost 128 bitů. Proces generování probíhá do té doby, než se vygeneruje požadovaný počet bitů.

Počet již vygenerovaných bitů znázorňuje i. Tato hodnota se po odstranění předchozí sekvence nastaví na hodnotu 0. Po vygenerování každého bitu se zvýší o 1.

Generátor pracuje se třemi proměnnými. První je čas od spuštění aplikace po kliknutí na tlačítko „START“. Tuto proměnnou v diagramu zastupuje písmeno t. Další dvě proměnné jsou souřadnice polohy kurzoru $[x, y]$. Rychlost načítání polohy kurzoru je možno nastavit v levé části aplikace v poli „Rychlost generování [bit/s]“.

Po načtení hodnot x a y dochází k jejich porovnání. V případě, že je hodnota x větší nebo rovna hodnotě y dochází k odečtení hodnoty y od hodnoty x. Pokud platí, že $x < y$ dochází k operaci $y - x$. Následně se výsledek tohoto rozdílu vynásobí hodnotou t.

Výsledný výsledek se upravuje operací modulo 2. Úpravou modulo 2 nám vzniká jeden bit s hodnotou 0 nebo 1. Po vygenerování dvou bitů dochází k úpravě pomocí von Neumannova schématu. Výsledný bit se zapíše na konec vygenerované sekvence a i se zvýší o 1. Poté se algoritmus opakuje do doby, než i je rovno P. V případě že $i = P$, dochází k ukončení generování a k převodu vygenerovaných bitů na heslo a následnému výpisu výsledné vygenerované sekvence, hesla a doby, během které byla sekvence vygenerovaná do pole v části „Výstup z generátoru“. Výsledný výpis po generování sekvence o velikosti 64 bitů a rychlosti načítání polohy kurzoru 128 hodnot za sekundu znázorňuje obrázek 5.5.



Obr. 5.5: Výsledný výpis obsahující vygenerovanou sekvenci, heslo a dobu potřebnou ke generování.

6 Realizace generátoru

Tato kapitola popisuje realizaci aplikace pro generování pravých náhodných čísel, která je popsaná v kapitole 5. Pro tvorbu aplikace bylo použito vývojové prostředí Eclipse IDE for Java EE Developers 2018-09 a vizuální nástroj JavaFX Scene Builder. Eclipse IDE for Java EE Developers je edice Eclipse, která obsahuje nástroj pro vývoj na platformě Java EE a vývoj webových aplikací.

6.1 Generování jednotlivých bitů

Generování náhodných bitů je realizované pomocí pohybu kurzoru po displeji. Algoritmus použitý pro generování bitů je popsán ve výpisu 6.1. Výsledné vygenerované hodnoty jsou vkládány do `ArrayList<Byte> listOfBites`, se kterým se dále pracuje. Jako postproces je použito von Neumannovo schéma, které je popsané v kapitole 1.3.

Výpis 6.1: Generování bitů pomocí pohybu kurzoru.

```
// pozice kurzoru X
mouseX = (int)MouseInfo.getPointerInfo().getLocation().getX();
// pozice kurzoru Y
mouseY = (int)MouseInfo.getPointerInfo().getLocation().getY();
// porovnani hodnot X, Y a nasledny vypocet bitu
if (mouseX > mouseY) {
    //a = cas zapnuti aplikace, b = cas spusteni generovani
    result = (int) (((mouseX - mouseY) * (b - a)) % (2));
} else {
    result = (int) (((mouseY - mouseX) * (b - a)) % (2));
    if (result < 0) {
        result *= (-1);
    }
}
// von Neumannovo schema
if (resultA == 2 && resultB == 2) {
    resultA = result;
} else if (resultA != 2 && resultB == 2) {
    resultB = result;
} else if (resultA != 2 && resultB != 2) {
    if (resultA == 1 && resultB == 0) {
        finalResult = 1;
        listOfBites.add(finalResult);
```

```
    } else if (resultA == 0 && resultB == 1) {
        finalResult = 0;
        listOfBites.add(finalResult);
    }
}
```

6.2 Převod bitů na heslo

Převod bitů na heslo je realizován pomocí `while` cyklu. Jeden znak hesla je vytvořen z 8 vygenerovaných bitů, jejichž hodnota je z binární soustavy převedena do decimální. Výsledná hodnota v decimální soustavě je upravena operací modulo 64. Výsledek je index znaku v `ArrayList<String> znakyNaHeslo` (například bity 00000011 se převedou na hodnotu 3, index 3 má v poli znak „d“). Algoritmus pro převod bitů na heslo znázorňuje výpis 6.2. Tabulka 6.1 zobrazuje výsledek převod sekvence o velikosti 48 bitů na heslo, kde vygenerovaná sekvence je 111011010010110000100101101010100101001010011010 a heslo je TQAvkz.

Tab. 6.1: Tabulka znázorňující převod vygenerovaných bitů na heslo.

11101101	00101100	00100101	10101010	01010010	10011010
237	44	37	170	82	154
T	Q	A	v	k	z

První řádek tabulky 6.1 znázorňuje vygenerovanou sekvenci rozdělenou na části po 8 bitech, druhý řádek zobrazuje převedení bitů z binární do dekadické soustavy. Ve třetím řádku je výsledné heslo.

Heslo může být vytvořeno z 64 různých znaků, mezi které patří malé a velké znaky abecedy bez diakritiky a čísla 0-9. Všechny možné znaky jsou zobrazeny v řádcích 17-24 ve výpisu 6.2. Abeceda a čísla 0-9 dávají dohromady 62 možností, proto je na konec přidán @ a &.

Další možností je použití 128 znaků místo 64. Klávesnice nemá 128 znaků, které by šlo napsat bez přepínání mezi jazyky klávesnice a nebo opakování znaků proto je zvoleno 64 znaků.

Výpis 6.2: Ukázka kódu pro převod bitů na heslo.

```
int cyklus = 0;
Byte aktualniBit;
while (cyklus < (pozadovanyPocet / 8)) {
      int konecneCislo = 0;
```

```
      int umisteniBituArray = cyklus * 8;                              5
      for (int j = 0; j < 8; j++) {                                    6
        aktualniBit = listOfBites.get(umisteniBituArray);              7
        // prevod 8 bitu na jedno decimalni cislo                      8
        if (aktualniBit == 0) {                                        9
            umisteniBituArray++;                                       10
        } else if (aktualniBit == 1) {                                 11
            konecneCislo += 1 * Math.pow(2, j);                        12
            umisteniBituArray++;                                       13
        }                                                              14
      }                                                                15
      cyklus++;                                                        16
      List<String> znakyNaHeslo = new ArrayList<String>                17
      (Arrays.asList("a", "b", "c", "d", "e", "f", "g", "h",           18
      "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s",           19
      "t", "u", "v", "w", "x", "y", "z", "0", "1", "2", "3",           20
      "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E",           21
      "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P",           22
      "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "@",           23
      "&"));                                                           24
      heslo.add(znakyNaHeslo.get(konecneCislo % 64));                  25
    }                                                                  26
  }                                                                    27
```

6.3 Potíže při realizaci

Při realizaci nastaly potíže s aktualizací uživatelského rozhraní během generování náhodných hodnot. Po stisknutí tlačítka „START“ došlo ke spuštění generování náhodných čísel, ale zároveň přestalo odpovídat GUI. Pro odstranění potíží bylo potřeba přesunout funkci pro generování bitů a aktualizování GUI na nové vlákno, které jede na pozadí aplikace. V uživatelském rozhraní probíhá na pozadí změna `ProgressBar`, textového pole pro výsledný výpis bitů a hesla, zpřístupnění a znepřístupnění „START“ a „STOP“ tlačítka.

Vlákno, které ukazuje výpis 6.3 se inicializuje při spuštění aplikace. Při stisknutí tlačítka „START“ se pomocí příkazu `thread.restart()` restartuje. Při zastavení generování tlačítkem „STOP“ dojde pomocí příkazu `thread.cancel()` k jeho zrušení.

Výpis 6.3: Vytvoření nového vlákna.

```
// tvorba noveho vlakna
final Service<Void> thread = new Service<Void>() {
  @Override
  public Task<Void> createTask() {
  return new Task<Void>() {
    @Override
    public Void call() throws InterruptedException {
      // generovani probiha do doby nez se vygeneruje
      // pozadovany pocet
      while (listOfBites.size() < pozadovanyPocet) {
        generator(listOfBites);
        // aktualizovani ProgressBar
        pBar.setProgress(listOfBites.size() /
        (double) pozadovanyPocet);
        try {
        // casova mezera mezi dvema nactenymi hodnotami
          TimeUnit.MILLISECONDS.sleep(1000 /
          pozadovanarychlostGenerovani);
        } catch (InterruptedException e) {
          // TODO Auto-generated catch blocks
          return null;
          }
        }
        return null;
      }
    };
  }
};
```

V případě, že generování proběhne úspěšně, změní se uživatelské prostředí, výsledná sekvence se vypíše (zavolá se funkce na výpis, která je popsaná v kapitole 6.2). Kód, který proběhne po úspěšném generování (generování, které nebylo zastavené tlačítkem „STOP“), znázorňuje výpis 6.4.

Výpis 6.4: Ukázka kódu, který proběhne po úspěšném vygenerování sekvence.

```
thread.setOnSucceeded((e) -> {
  // smazani textu v textovem poli pro vypis
  t.clear();
```

```
  // zpristupneni a znepristupneni tlacitek
  stopButton.setDisable(true);
  saveButton.setDisable(false);
  deleteButton.setDisable(false);
  // vypis vyslednych bitu
  for (byte bite : listOfBites) {
    if (bite == 0) {
      t.appendText("0");
    } else {
      t.appendText("1");
    }
  }
  // prevod bitu na heslo
  prevodNaHeslo(listOfBites);
});
```

6.4 Spustitelný JAR soubor

JAR (Java ARchiver) je komprimovaný archivní soubor s Java třídami a daty (obrázky, dokumentace, ...), který má příponu .jar. Jedná se o spustitelný soubor, který lze vytvořit ve vývojovém prostředí Eclipse.

V menu vybereme položku „File“ a poté „Export“. Otevře se nové okno, kde rozbalíme složku „Java“ a zvolíme „Runnable JAR file“. Klikneme na „Next“. V poli „Launch configuration“ zvolíme projekt který chceme exportovat a v poli „Export destination“ místo uložení. Nakonec klikneme na tlačítko „Finish“.

7 Testování aplikace a vygenerovaných sekvencí bitů

Aplikace byla spuštěna na počítači s procesorem Intel Core i3-5010U 2,10 GHz, pamětí RAM 4 GB a 64bitovým operačním systémem Windows 10 Home. Všechna data uvedená v této kapitole a testované sekvence byly získány na výše uvedeném zařízení.

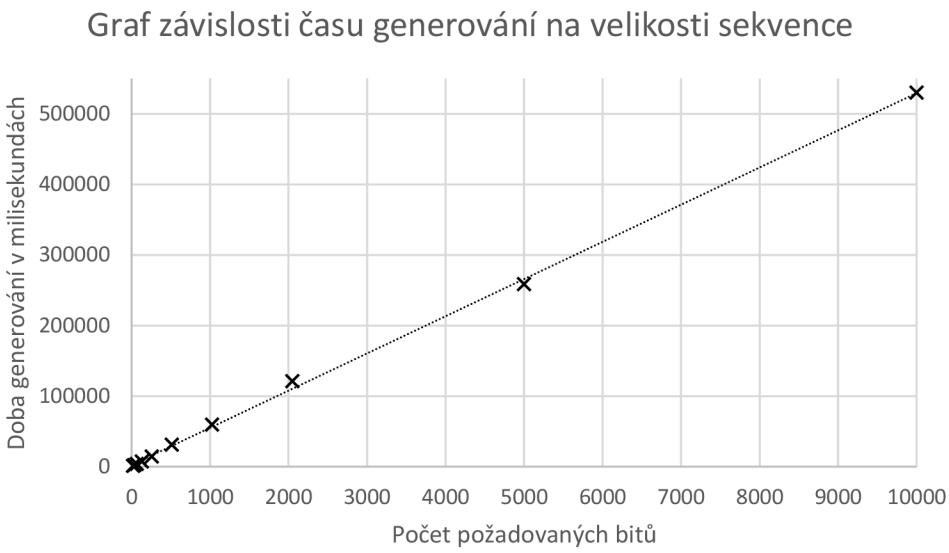
7.1 Časová náročnost generování sekvencí

Časovou náročnost generování sekvence lze zjistit v samotné aplikaci. Po vygenerování požadované sekvence se na displeji objeví doba, během které byla sekvence vygenerována. Následující tabulka 7.1 ukazuje dobu generování v milisekundách pro sekvence o různých délkách, při souvislém pohybu kurzoru. V prvním sloupci je délka sekvence, ve druhém sloupci doba generování celé sekvence, třetí sloupec obsahuje dobu generování jednoho bitu z dané sekvence. Tato hodnota vznikla podílem celkového času generování délkou sekvence. Hodnoty v druhém i třetím sloupci jsou průměrné hodnoty ze čtyř vygenerovaných sekvencí o dané délce.

Tab. 7.1: Časová náročnost generování sekvencí v milisekundách.

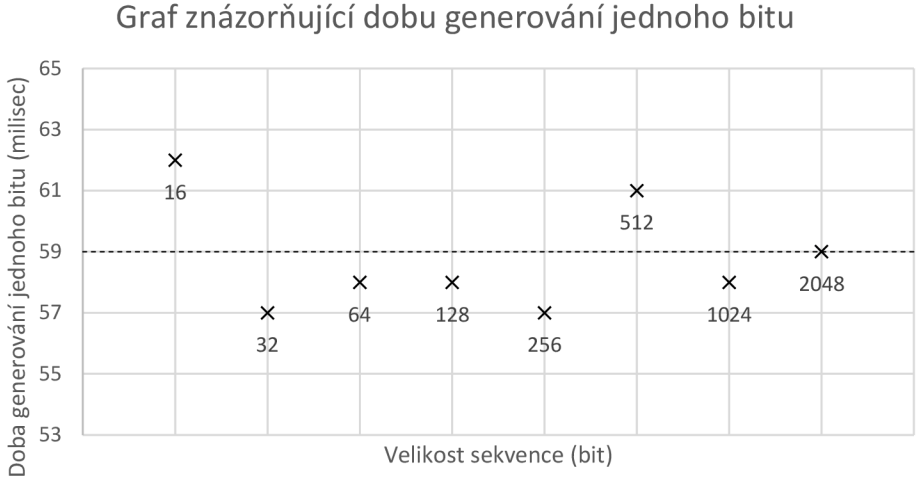
Délka sekvence	**Doba generování v milisekundách**	
	celá sekvence	**1 bit**
16	986	62
32	1838	57
64	3717	58
128	7423	58
256	14578	57
512	31114	61
1024	59588	58
2048	121109	59
5 000	258951	53
10 000	530227	52

Následující obrázek 7.1 znázorňuje tabulku 7.1. Z grafu lze vyčíst, že časová náročnost generování sekvencí se přímo úměrně odvíjí od požadované délky sekvence. Délka sekvence a doba generování jsou lineárně závislé.



Obr. 7.1: Graf znázorňující dobu generování sekvencí o různých délkách.

Obrázek 7.2 zobrazuje celkovou dobu generování převedenou na dobu potřebnou k vygenerování jednoho bitu. Přerušovaná čára zobrazuje průměrnou dobu generování jednoho bitu pro sekvence o délce 16 až 2048 bitů. Průměrná doba potřebná pro vygenerování jednoho bitu je 59 milisekund. Délky jednotlivých sekvencí jsou zobrazeny pod jednotlivými body. Z grafu lze vyčíst, že doba potřebná pro vygenerování jednoho bitu se příliš nemění v závislosti na délce sekvence.



Obr. 7.2: Graf znázorňující dobu generování jednoho bitu v sekvencích o různých délkách.

Výše uvedené časy byly získány nepřerušovaným pohybem kurzoru během generování. V případě generování sekvence na pozadí při běžné obsluze počítače je doba

generování delší. Generování sekvence o délce 2048 bitů při rychlosti 128 načtených hodnot za sekundu trvalo 7 minut a 48 sekund při generování během běžného používání zařízení. Konkrétní aktivitou bylo čtení mailů. Při souvislém pohybu kurzoru byla průměrná doba generování sekvence o stejné délce 2 minuty a 11 sekund (přibližně čtyřikrát méně).

7.2 Výpočetní náročnost aplikace

Využití systémových prostředků aplikací bylo zjištěno pomocí dat, která jsou dostupná ve Správci úloh v OS Windows. Během generování sekvencí a obsluhy aplikace byla zaznamenávána data o využití CPU (Central Processing Unit - Centrální procesorová jednotka), operační paměti RAM a GPU (Graphics Processing Unit - Grafický procesor). Zaznamenávání bylo spuštěno po dobu 3 minut a 34 sekund. Dohromady bylo vytvořeno 214 záznamů. Během zaznamenávané doby bylo vygenerováno 6 sekvencí o různé délce a různé rychlosti načítání hodnot. Tabulka 7.2 znázorňuje všechny získané hodnoty během zaznamenávání.

Tab. 7.2: Využití systémových prostředků během chodu aplikace.

	CPU	**RAM paměť**	**GPU**
Průměrná hodnota	5,6 %	126,2 MB	0,3 %
Maximální hodnota	22,0 %	159,1 MB	2,2 %
Minimální hodnota	0,0 %	96,7 MB	0,0 %

Využití systémových prostředků během mazání sekvencí zobrazuje tabulka 7.3. Systémové prostředky potřebné během nastavování požadovaných parametrů sekvence jsou v tabulce 7.4.

Tab. 7.3: Využití systémových prostředků během mazání vygenerovaných sekvencí.

	CPU	**RAM paměť**	**GPU**
Průměrná hodnota	10,0 %	127,9 MB	0,3 %
Maximální hodnota	15,3 %	144,2 MB	0,5 %
Minimální hodnota	9,0 %	116,3 MB	0,0 %

Procesor byl nejvíce vytížen během nastavování požadovaných parametrů a mazání výsledných sekvencí. Nejméně byl procesor použit při generování krátkých sekvencí s pomalým načítáním hodnot. Využití procesoru během generování náhodných sekvencí zobrazuje tabulka 7.5.

Tab. 7.4: Využití systémových prostředků během nastavování parametrů generátoru.

	CPU	**RAM paměť**	**GPU**
Průměrná hodnota	10,7 %	124,0 MB	0,3 %
Maximální hodnota	22,2 %	154,8 MB	1,1 %
Minimální hodnota	0,3 %	96,7 MB	0,0 %

Rychlost načítání uvedená v tabulkách 7.5, 7.6 a 7.7 je počet poloh kurzoru, které byly načteny během jedné sekundy. Sekvence uvedené v tabulkách byly vygenerovány během následujících časů. Sekvence o délce 1024 bitů byla vytvořena během 1 minuty, 11 sekund a 570 milisekund. Generování sekvence o délce 128 bitů trvalo 8 sekund a 701 milisekund. U sekvence o délce 64 bitů trvalo generování 8 sekund a 235 milisekund. Sekvence o délce 16 bitů byly vygenerovány během 2 sekund a 249 milisekund (64 načtených hodnot za sekundu), 21 sekund a 228 milisekund (4 načtené hodnoty za sekundu) a 46 sekund a 100 milisekund (2 načtené hodnoty za sekundu).

Tab. 7.5: Využití procesoru během generování sekvencí.

Délka sekvence (bit) **Rychlost načítání**	**1024** **128**	**128** **128**	**64** **64**	**16** **64**	**16** **4**	**16** **2**
Průměrná hodnota	4,6 %	9,2 %	3,8 %	4,6 %	1,1 %	0,9 %
Maximální hodnota	13,8 %	18,6 %	7,4 %	5,2 %	9,0 %	11,5 %
Minimální hodnota	0,6 %	4,8 %	1,9 %	4,2 %	0,0 %	0,0 %

Použití paměti RAM záleželo na procesu generování, kdy se zvyšovala hodnota použití v MB s množstvím již vygenerovaných náhodných bitů. Nejméně paměti bylo potřeba při generování sekvencí o délce 16 bitů, kde se hodnota využití paměti během generování změnila o 0,9 MB až 4,7 MB. Hodnoty využití paměti RAM během generování sekvencí jsou zobrazeny v tabulce 7.6.

Tab. 7.6: Využití paměti RAM během generování sekvencí.

Délka sekvence (bit) **Rychlost načítání**	**1024** **128**	**128** **128**	**64** **64**	**16** **64**	**16** **4**	**16** **2**
Průměrná hodnota (MB)	110,8	131,1	126,0	121,8	146,4	141,2
Maximální hodnota (MB)	122,8	159,1	146,8	146,8	146,9	141,4
Minimální hodnota (MB)	101,8	121,6	119,1	119,1	146,0	137,1

Použití grafického procesoru GPU bylo nejvyšší během nastavování parametrů. Během samotného generování byl využit minimálně. Při generování s pomalou rychlostí načítání dat bylo využití grafického procesoru po celou dobu téměř nulové. V tabulce 7.7 je znázorněno využití GPU během generování.

Tab. 7.7: Využití grafického procesoru během generování sekvencí.

Délka sekvence (bit) **Rychlost načítání**	**1024** **128**	**128** **128**	**64** **64**	**16** **64**	**16** **4**	**16** **2**
Průměrná hodnota	0,3 %	1,1 %	0,7 %	0,7 %	0,0 %	0,0 %
Maximální hodnota	2,2 %	1,7 %	0,1 %	0,9 %	0,2 %	0,2 %
Minimální hodnota	0,0 %	0,0 %	0,2 %	0,6 %	0,0 %	0,0 %

Využití systémových prostředků během používání aplikace a generování náhodných sekvencí je nízké. V případě generování náhodných sekvencí lze mít aplikaci spuštěnou na pozadí aniž by systémově omezovala použití jiných aplikací.

7.3 Testování vygenerovaných sekvencí bitů

Pro testování sekvencí byl zvolen soubor patnácti testů NIST STS. Soubor testů NIST STS je popsaný v kapitole 4.1. Soubory potřebné k testování a dokumentaci k softwaru lze stáhnout ve formátu .zip na webové adrese: `https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software`.

Doporučené testování je v UNIXovém systému. Pro testování byl použit systém Ubuntu 18.04.2 (Bionic Beaver), který byl spuštěn ve virtuálním prostření VMware 15. Pro správný chod testu bylo potřeba doinstalovat kompilátor GCC. Kompilátor GCC se nainstaluje pomocí příkazu `$ sudo apt install gcc`. Dále bylo potřeba upravit soubor `makefile` a spustit ho, čímž se vytvoří soubor aplikace `assess`.

Aplikace se spouští zadáním příkazu `./assess <počet testovaných bitů>` do terminálu. Poté je uživatel vyzván ke zvolení požadovaného testu, zadání názvu a úložiště souboru s testovanou sekvencí a nastavení požadovaných parametrů. Po ukončení testování jsou do složky `experiments/algorithmTesting` uloženy výsledky v souboru `finalAnalysisReport.txt`. Podrobněji jsou pokyny a postup popsány v dokumentu [33, kapitola 5].

Otestovány byly sekvence vytvořené pomocí generátoru, který je popsán v kapitole 5. Výsledky jednotlivých dílčích testů znázorňuje tabulka 7.8. Testům byly podrobeny sekvence o délkách 10 000, 100 000 a 1 000 000 bitů. Doporučená minimální délka testované sekvence je 1 000 000 bitů, aby nedošlo ke zkreslení výsledků testu a všechny dílčí testy proběhly úspěšně [33].

Tab. 7.8: Výsledky jednotlivých částí testu NIST STS pro vygenerované sekvence o velikostech 10 000, 100 000 a 1 000 000 bitů.

Typ testu	**Velikost testované sekvence v bitech**		
	10 000	**100 000**	**1 000 000**
1. Frekvenční test	9/10 | 90 %	10/10 | 100 %	9/10 | 90 %
2. Blokový frekvenční test	9/10 | 90 %	10/10 | 100 %	10/10 | 100 %
3. Test sekvencí bitů	10/10 | 100 %	10/10 | 100 %	10/10 | 100 %
4. Test na nejdelší sekvenci jedniček v jenom bloku	10/10 | 100 %	9/10 | 90 %	9/10 | 90 %
5. Test sérií binárních matic	0/10 | 0 %	10/10 | 100 %	10/10 | 100 %
6. Test diskrétní Fourierovy transformace	10/10 | 100 %	10/10 | 100 %	10/10 | 100 %
7. Test nepřekrývajících se vzorů	nedokončen	10/10 | 100 %	10/10 | 100 %
8. Test překrývajících se vzorů	nedokončen	10/10 | 100 %	9/10 | 90 %
9. Mauerův „univerzální statistický“ test	0/10 | 0 %	9/10 | 90 %	9/10 | 90 %
10. Test lineární složitosti	10/10 | 100 %	10/10 | 100 %	10/10 | 100 %
11. Test sérií	10/10 | 100 %	10/10 | 100 %	9/10 | 90 %
12. Přibližný test entropie	10/10 | 100 %	8/10 | 80 %	8/10 | 80 %
13. Test kumulativních součtů	9/10 | 90 %	10/10 | 100 %	10/10 | 100 %
14. Test náhodných návštěv	-	-	2/2 | 100 %
15. Test náhodných variant návštěv	-	-	2/2 | 100 %

Každá buňka tabulky obsahuje dvě číselné hodnoty. První hodnota znázorňuje počet úspěšných proudů bitů a druhá je úspěšnost v procentech. Pro testování byla sekvence rozdělena na 10 proudů bitů (bitstreams), každý o velikosti 1 000, 10 000 nebo 100 000 bitů, v závislosti na celkové délce sekvence. Například sekvence o délce 100 000 bitů byla rozdělena na 10 proudů o 10 000 bitech.

Minimální úspěšnost pro projití dílčích testů kromě *Testu náhodných návštěv* a *Testu náhodných variant návštěv* je 8/10. Požadované minimum u *Testu náhodných návštěv* a *Testu náhodných variant návštěv* je 1/2. Minimum 1/2 je stanoveno pro sekvence o délce 1 000 000 bitů a více. Pro kratší sekvence není minimum definované a testy pro sekvence o délkách 10 000 a 100 000 neproběhnou.

V případech, kdy má jednotlivý test více výsledků, je v tabulce znázorněn průměr. Více výsledků má *Test sekvencí bitů*, *Test nepřekrývajících se vzorů*, *Test náhodných návštěv*, *Test náhodných variant návštěv* a *Test lineární složitosti*. Aby byl generátor uznán za vhodný, musí sekvence uspět ve všech 15 dílčích testech.

Jak bylo zmíněno v kapitole 4, pokud generátor projde všemi známými statistickými testy, neznamená to, že lze generátor označit za zcela náhodný a stejný výsledek se bude opakovat i u dalších sekvencí.

U sekvence o délce 10 000 nebyl dokončen *Test nepřekrývajících se vzorů* a *Test překrývajících se vzorů*. Sekvence neuspěla u *Testu sérií binárních matic* a *Mauerova „univerzálního statistického“ testu*. Důvodem uspění sekvence o délce 10 000 bitů pouze u devíti testů z 15 může být příliš krátká délka testované sekvence. Sekvence o délkách 100 000 a 1 000 000 bitů uspěly ve všech dílčích testech, u kterých pro ně byla stanovena minima pro projití.

Dvě následující tabulky zobrazují porovnání výsledků testování sekvence s výsledky sekvencí z webového zdroje. Tabulka 7.9 porovnává výsledky testování generátoru a webového generátoru random.org. V tabulce 7.10 jsou zobrazeny výsledky generátoru a internetového zdroje HotBits.

Sekvenci vygenerovanou pomocí **random.org** lze stáhnout ze stránky `https://archive.random.org/text`. Jedná se o archiv vygenerovaných sekvencí ve formátu TXT a BIN. Od roku 2006 se denně do archivu přidává jeden soubor v obou formátech obsahující jedničky a nuly, který je přístupný veřejnosti. Každý soubor obsahuje 8 388 608 náhodných bitů [4]. Pro porovnání byl použit první milion bitů sekvence ze dne 23.3.2019.

U **HotBits** je pro vygenerování sekvence potřeba vyplnit formulář obsahující počet bitů, formát zobrazení a zda vlastníte API klíč nebo ne. Formulář je dostupný na stránce `https://www.fourmilab.ch/hotbits/secure_generate.html`. API klíč je potřeba pro generování pravých náhodných bitů. V případě, že API klíč nevlastníte, vygenerují se pseudonáhodná čísla místo pravých náhodných čísel. O API klíč je možné požádat na webové stránce `https://www.fourmilab.ch/hotbits/apikey_request.html`. Maximální počet vygenerovaných bytů je 2048. Spojením více vygenerovaných sekvencí a převedením bytů na bity byla získána sekvence o velikosti 100 000 bitů. Tato sekvence byla otestována a výsledek je zobrazen v tabulce 7.10.

Testům byly podrobeny tři sekvence vygenerované generátorem podle návrhu v kapitole 5 a dvě sekvence z internetových zdrojů náhodných sekvencí. Z tabulek je viditelné, že výsledky testování generátoru mají menší procentuální úspěšnost než výsledky sekvencí z internetových zdrojů. Počet úspěšných dílčích testů je pro všechny sekvence srovnatelný a generátor vytvořený podle návrhu lze považovat za náhodný.

Tab. 7.9: Porovnání výsledků sekvence o velikosti 1 000 000 bitů vygenerovaných pomocí vlastního generátoru a random.org.

Typ testu	**Generátor podle vlastního návrhu**	**random.org**
1. Frekvenční test	9/10 | 90 %	10/10 | 100 %
2. Blokový frekvenční test	10/10 | 100 %	10/10 | 100 %
3. Test sekvencí bitů	10/10 | 100 %	10/10 | 100 %
4. Test na nejdelší sekvenci jedniček v jenom bloku	9/10 | 90 %	10/10 | 100 %
5. Test sérií binárních matic	10/10 | 100 %	9/10 | 90 %
6. Test diskrétní Fourierovy transformace	10/10 | 100 %	10/10 | 100 %
7. Test nepřekrývajících se vzorů	10/10 | 100 %	10/10 | 100 %
8. Test překrývajících se vzorů	9/10 | 90 %	10/10 | 100 %
9. Mauerův „univerzální statistický“ test	9/10 | 90 %	8/10 | 80 %
10. Test lineární složitosti	10/10 | 100 %	10/10 | 100 %
11. Test sérií	9/10 | 90 %	10/10 | 100 %
12. Přibližný test entropie	8/10 | 80 %	10/10 | 100 %
13. Test kumulativních součtů	10/10 | 100 %	9/10 | 90 %
14. Test náhodných návštěv	2/2 | 100 %	2/2 | 100 %
15. Test náhodných variant návštěv	2/2 | 100 %	2/2 | 100 %

Tab. 7.10: Porovnání výsledků sekvence o velikosti 100 000 bitů vygenerovaných pomocí vlastního generátoru a HotBits.

Typ testu	**Generátor podle vlastního návrhu**	**HotBits**
1. Frekvenční test	10/10 | 100 %	10/10 | 100 %
2. Blokový frekvenční test	10/10 | 100 %	10/10 | 100 %
3. Test sekvencí bitů	10/10 | 100 %	10/10 | 100 %
4. Test na nejdelší sekvenci jedniček v jenom bloku	9/10 | 90 %	10/10 | 100 %
5. Test sérií binárních matic	10/10 | 100 %	10/10 | 100 %
6. Test diskrétní Fourierovy transformace	10/10 | 100 %	10/10 | 100 %
7. Test nepřekrývajících se vzorů	10/10 | 100 %	10/10 | 100 %
8. Test překrývajících se vzorů	10/10 | 100 %	10/10 | 100 %
9. Mauerův „univerzální statistický“ test	9/10 | 90 %	9/10 |90 %
10. Test lineární složitosti	10/10 | 100 %	10/10 | 100 %
11. Test sérií	10/10 | 100 %	10/10 | 100 %
12. Přibližný test entropie	8/10 | 80 %	9/10 | 90 %
13. Test kumulativních součtů	10/10 | 100 %	10/10 | 100 %
14. Test náhodných návštěv	-	-
15. Test náhodných variant návštěv	-	-

8 Návrh rozšíření aplikace na smartphony

Pro použití aplikace v tzv. chytrých mobilních telefonech je potřeba nový vstup dat. Ke generování náhodných čísel pomocí smartphonu lze použít gyroskop, akcelerometr, snímač pohybu prstu po displeji apod.

8.1 Základní technické parametry

- Platforma:
  - Android.
- Zobrazení:
  - Na výšku.
- Jazyk:
  - Čeština.
- Programovací jazyk:
  - Java.
- Vývojové prostředí:
  - Eclipse s přidaným pluginem ADT.

Desktopová aplikace byla vytvořena v programovacím jazyce Java. Tento jazyk je podporován jak v počítačích tak i v tzv. chytrých mobilních zařízeních. Pro úpravu aplikace lze použít stejnou verzi Eclipse, v jaké byla aplikace vytvořena, s přidaným pluginem ADT (Android Development Tools). Plugin ADT přidá do prostředí Eclipse nabídky týkající se Androidu, které umožní tvorbu Android projektů, uživatelského rozhraní aplikací a export souborů do souborového formátu .apk (Android Package) podporovaného operačním systémem Android [38].

8.2 Návrh struktury aplikace

Mobilní aplikace bude mít stejnou funkčnost, jako desktopová aplikace popsaná v kapitole 5. Rozdílný bude vzhled uživatelského rozhraní a způsob získávání náhodných dat. Návrh uživatelského rozhraní znázorňuje obrázek 8.1.

Jednotlivé položky a tlačítka budou mít stejnou funkci jako u počítačové aplikace. Jedinou změnou v uživatelském rozhraní bude vzhled tlačítka „START“, „STOP“, „ULOŽIT“ a „SMAZAT“. Místo textu se použijí ikony, které potřebují méně místa. Ikony jsou znázorněny na obrázku 8.2. První ikona bude znázorňovat tlačítko „START“, druhá tlačítko „STOP“. Složka bude sloužit k uložení vygenerované sekvence a hesla. Tlačítko s ikonou koš bude určeno k odstranění vygenerované sekvence.

Obr. 8.1: Návrh grafického uživatelské rozhraní aplikace pro mobilní telefony.

Obr. 8.2: Vzhled tlačítek v mobilní aplikaci.

Náhodná čísla budou generovaná pomocí akcelerometru. Důvodem zvolení této možnosti je univerzální využití na smartphonech, kde je akcelerometr součástí základní výbavy. Plusem pro uživatele může být generování náhodných čísel na pozadí bez potřebné interakce.

Pro přístup k senzorům na platformě Android lze v programovacím jazyce Java použít předdefinovanou třídu `SensorManager`. Dále je nutností implementovat rozhraní `SensorEventListener`, které určuje, jak bude aplikace reagovat na změnu snímaných hodnot pomocí senzorů [39, 40].

Změny na senzorech telefonu lze zaznamenat pomocí třídy `SensorEvent`. Tato třída reaguje na události senzorů a uchovává informace, mezi které patří typ snímače, časové razítko, přesnost a hodnoty získané snímačem [41].

Pro získávání informací z akcelerometru se používá trojosý souřadnicový systém, který je definován vzhledem k obrazovce telefonu ve výchozí orientaci. Při změně orientace obrazovky zůstávají osy bez změny [41].

Využívají se tři osy, osa X, Y a Z. Osa X je vodorovná a směřuje zleva doprava, osa Y je svislá a vede ze spodní části telefonu nahoru a osa Z směřuje ze zadní strany telefonu do přední strany obrazovky. V tomto systému mají hodnoty souřadnice za obrazovkou zápornou hodnotu Z [41].

Ve třídě `SensorEvent` lze načíst pole hodnot získaných pomocí akcelerometru příkazem `values`, výsledné hodnoty jsou v m/s^2.

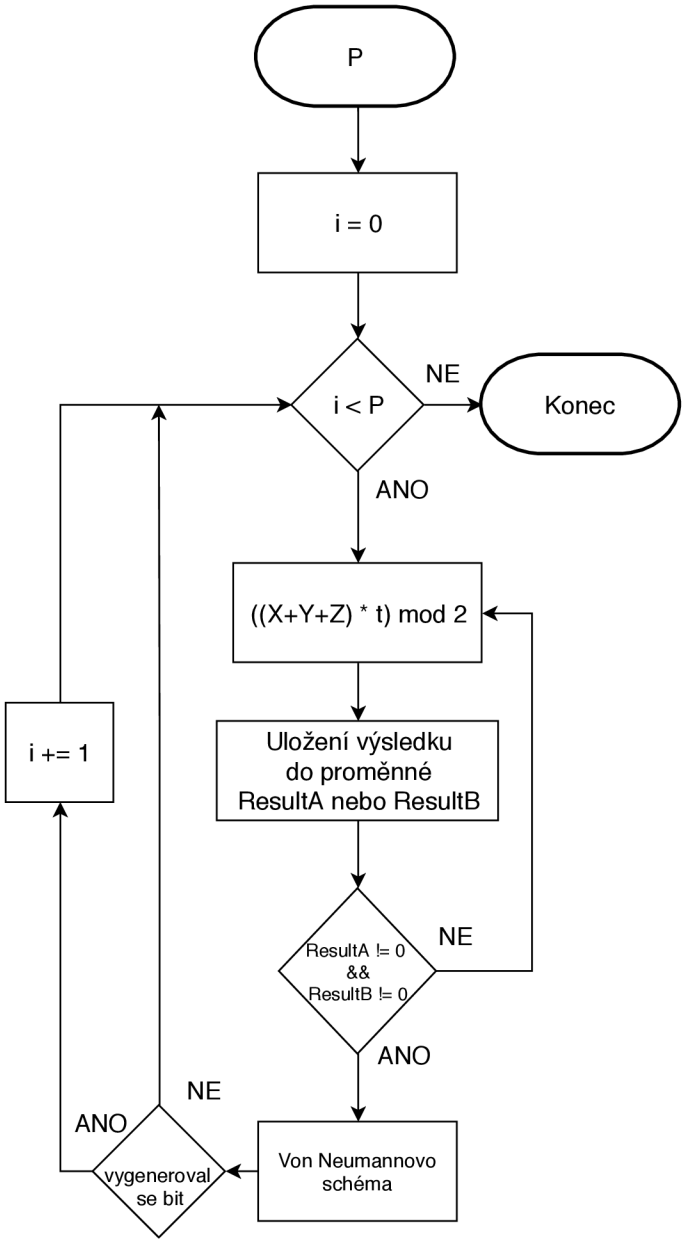
- `values[0]` - zrychlení na ose X mínus gravitace
- `values[1]` - zrychlení na ose Y mínus gravitace
- `values[2]` - zrychlení na ose Z mínus gravitace

Z hodnot zrychlení je odebrána gravitační síla z důvodu, že zařízení ležící na stole by mělo zrychlení o velikosti $g = 9,81\ m/s^2$ [41].

Pro generování lze použít upravený algoritmus popsaný v kapitole 5.2, který zobrazuje obrázek 5.4. Změna u mobilní aplikace bude v počtu načtených hodnot, kde by se místo hodnot x a y načítaly hodnoty X, Y a Z. Upravený algoritmus je zobrazen na obrázku 8.3.

Hodnoty X, Y a Z znázorňují zrychlení jednotlivých os. Tyto tři parametry budou sečteny a provede se součin součtu s rozdílem časů t. Čas t je doba od zapnutí aplikace po spuštění generování pomocí tlačítka „START“. Výsledek se upraví operací modulo 2. Po vygenerování dvou bitů následuje jejich úprava pomocí von Neumannova schématu a výsledný bit se přidá na konec sekvence. Algoritmus se bude opakovat do doby, než počet vygenerovaných bitů i bude roven počtu požadovaných bitů P. Po vygenerování se vypíše sekvence v poli „Výstup z generátoru“.

Tento návrh nebyl otestován. V případě nesplnění požadavků by byl pozměněn.



Obr. 8.3: Vývojový diagram znázorňující algoritmus pro generování náhodných sekvencí na mobilních zařízeních (smartphonech).

9 Závěr

Cílem této bakalářské práce bylo porovnání generátorů pravých náhodných čísel, návrh a realizace generátoru pravých náhodných čísel pro PC a návrh rozšíření generátoru umožňující jeho provoz na smartphonech.

První kapitola se věnuje obecnému popisu generátorů pravých náhodných čísel. Po vysvětlení pojmu náhodná čísla následuje stručný popis pseudonáhodných čísel a charakteristika generátorů pravých náhodných čísel. U každého typu je uveden příklad konkrétního existujícího generátoru.

Druhá kapitola slouží k bližšímu pohledu na generátory náhodných čísel. Jsou zde popsány základní vlastnosti generátorů, mezi které patří rychlost generování a požadovaný hardware.

Ve třetí kapitole je uvedeno užití generátorů v kryptografii a vhodnost generátorů pro použití v počítači i internetu věcí. Pro PC existuje více vhodných generátorů než pro IoT. Důvodem jsou vyšší nároky na generátory pro IoT, které musí mít nízkou spotřebu energie a kompaktní rozměr. Možným řešením mohou být i internetové generátory náhodných sekvencí jako je random.org a HotBits.

Čtvrtá kapitola se zaměřuje na testování vygenerovaných sekvencí bitů. Kapitola především popisuje testovaní pomocí NIST Statistical Test Suite. Pro splnění tohoto testu je potřeba, aby sekvence vygenerované testovaným generátorem obstály ve všech 15 dílčích testech.

Pátá kapitola obsahuje návrh generátoru pravých náhodných čísel pro PC. Navržený generátor využívá jako zdroj entropie náhodné pohyby kurzoru obsluhou počítače. Generování náhodných sekvencí pomocí pohybu kurzoru je v praxi používáno poměrně často, využívá je například program VeraCrypt k tvorbě šifrovacích klíčů. Pole pro zisk náhodných hodnot je displej zařízení, v případě více monitorů jsou data snímána ze všech obrazovek. Podle souřadnic aktuálního umístění kurzoru a času od zapnutí aplikace po spuštění generování se vygeneruje příslušný bit, který se dále zpracuje a vloží do výsledné sekvence. Na závěr generování se sekvence převede na heslo obsahující malé a velké znaky abecedy bez diakritiky, čísla 0-9, & a @.

V další kapitole je popsána realizace aplikace podle návrhu z kapitoly pět. Obsahuje části zdrojového kódu, na kterých je popsán princip aplikace. V závěru kapitoly jsou zmíněny potíže při realizaci a tvorba spustitelného souboru.

Sedmá kapitola se věnuje testování vytvořeného generátoru. Implementovaný generátor byl podroben sadě patnácti testů NIST STS. Kapitola obsahuje výsledky testů společně s časovou a výpočetní náročností aplikace. Časová náročnost generování sekvencí se přímo úměrně odvíjí od požadované délky sekvence. Výpočetní náročnost aplikace je nízká zejména během generování sekvencí. Při nastavování po-

žadovaných parametrů a mazání vytvořených sekvencí jsou systémové nároky vyšší. Aplikaci lze zapnout a nechat běžet na pozadí během běžné obsluhy počítače, aniž by omezovala chod dalších aplikací.

Osmá kapitola obsahuje návrh generátoru pro smartphony. Zdrojem entropie v návrhu je akcelerometr, který je součástí základní výbavy každého smartphonu.

Hodnocení implementovaného generátoru lze provést prozkoumáním výsledků statistických testů. U sekvence o délce 10 000 bitů se daly očekávat špatné výsledky. Tato sekvence obstála pouze v devíti dílčích testech, pro dva testy neměla požadovanou minimální délku k provedení. Doporučená délka sekvence, aby data nebyla zkreslená a všechny testy proběhly úspěšně, je 100krát více, tj. 1 000 000 bitů. Sekvence o délkách 100 000 a 1 000 000 bitů prošly všemi testy, pro které měly stanovené minimum k projití.

Jak bylo popsáno ve čtvrté kapitole, tento výsledek se u všech sekvencí vygenerovaných generátorem nemusí opakovat. Dále je nutné podotknout, že generátor založený na lidském chování se po čase může stát periodický. V ideálním případě by každou vygenerovanou sekvenci měl vytvořit jiný uživatel. V závěru kapitoly sedm je porovnání výsledků generátorů. Pro srovnání výsledků byla zvolená sekvence bitů z generátoru random.org a HotBits. Internetové generátory měly vyšší procentuální úspěšnost v testech. Všechny srovnávané sekvence splnily požadované minima k projití a generátory můžeme označit za náhodné.

Na závěr je možno konstatovat, že výstupní sekvence vytvořené aplikací, až na jednu, prošly všemi NIST STS testy. Generátor realizovaný podle návrhu je plně funkční.

Literatura

[1] STIPČEVIĆ, Mario a Çetin Kaya KOÇ. *True Random Number Generators. Open Problems in Mathematics and Computational Science*. Cham: Springer International Publishing, 2014, 2014-11-11, s. 275-315. DOI: 10.1007/978-3-319-10683-0_12. ISBN 978-3-319-10682-3. Dostupné z URL: <http://link.springer.com/10.1007/978-3-319-10683-0_12>.

[2] BURDA, Karel. *Aplikovaná kryptografie* Brno: VUTIUM, 2013, 256 s. ISBN 978-80-214-4612-0.

[3] SHOR, Peter W. a John PRESKILL. *Simple Proof of Security of the BB84 Quantum Key Distribution Protocol*. Physical Review Letters [online]. 2000, 85(2), 441-444 [cit. 17. 11. 2018]. DOI: 10.1103/PhysRevLett.85.441. ISSN 0031-9007. Dostupné z URL: <https://link.aps.org/doi/10.1103/PhysRevLett.85.441>.

[4] HAAHR, Mads. *Introduction to Randomness and Random Numbers* [online]. 2009 [cit. 21. 10. 2018]. Dostupné z URL: <http://random.org/randomness/>.

[5] JONES, David A. *True random number generators for a more secure IoT* [online]. 24. 3. 2016 [cit. 21. 10. 2018]. Dostupné z URL: <http://www.techdesignforums.com/practice/technique/true-random-number-generators-for-more-secure-systems>.

[6] *Eknihovna MU: Generování náhodných čísel* [online]. [cit. 9. 11. 2018]. Dostupné z URL: <https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=7024>.

[7] NOVÁK, J. *Generování náhodných čísel : bakalářská práce*. Plzeň: Západočeská univerzita v Plzni, Fakulta elektrotechnická, 2016, 54 s. Dostupné z URL: <https://dspace5.zcu.cz/bitstream/11025/23167/1/BP_NOVAK_FINAL.pdf>.

[8] PETRŽELA, J. *Teorie elektronických obvodů : prezentace* [online]. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2010 [cit. 9. 11. 2018]. Dostupné z URL: <http://www.urel.feec.vutbr.cz/MTEO/mteo/sumy.pdf>.

[9] *Araneus Alea II: True Random Number Generator* [online]. [cit. 21. 10. 2018]. Dostupné z URL: <https://www.araneus.fi/products/alea2/en/>.

[10] JOHN, M. *Intel® Digital Random Number Generator (DRNG) Software Implementation Guide* [online]. 14. 5. 2014 [cit. 11. 11. 2018]. Dostupné z URL: <https://software.intel.com/en-us/articles/intel-digital-random-number-generator-drng-software-implementation-guide>.

[11] *ChaosKey* [online]. [cit. 9. 11. 2018]. Dostupné z URL: <https://altusmetrum.org/ChaosKey/>.

[12] ROBSON, Steward. *A Ring Oscillator Based Truly Random Number Generator : Master Thesis* [online]. Waterloo, Ontario, Canada, 2013 [cit. 9. 11. 2018]. Dostupné z URL: <https://core.ac.uk/download/pdf/144146883.pdf>.

[13] *TRNG-IP-76 / EIP-76 family of FIPS approved True Random Generators* [online]. [cit. 17. 11. 2018]. Dostupné z URL: <https://www.insidesecure.com/Products/Silicon-IP/Complex-Cryptographic-Accelerators/TRNG-IP-76>.

[14] WALKER, John. *HotBits: Genuine random numbers, generated by radioactive decay* [online]. 1996 [cit. 21. 10. 2018]. Dostupné z URL: <https://www.fourmilab.ch/hotbits/>.

[15] GARCIA-ESCARTIN, Juan Carlos. Quantum random number generators [online]. 2017, 89(1) [cit. 9. 11. 2018]. DOI: 10.1103/RevModPhys.89.015004. ISSN 0034-6861. Dostupné z URL: <https://link.aps.org/doi/10.1103/RevModPhys.89.015004>.

[16] *Quantis Random Number Generator: True random number generator exploiting the randomness of quantum physics* [online]. [cit. 9. 11. 2018]. Dostupné z URL: <https://www.idquantique.com/random-number-generation/products/quantis-random-number-generator/>.

[17] *Random Number Generator* [online]. [cit. 11. 11. 2018]. Dostupné z URL: <https://www.veracrypt.fr/en/Random%20Number%20Generator.html>.

[18] PECKA, Stanislav. *Návrh a implementace generátoru náhodných čísel: diplomová práce*. České Budějovice, 2018. 57 s. Vedoucí práce byl Ing. Petr Břehovský. Jihočeská univerzita v Českých Budějovicích: Přírodovědecká fakulta. Dostupné z URL: <https://theses.cz/id/sxuo5y/DP.pdf>.

[19] *True random number generator 1.0* [online]. 13. 10. 2009 [cit. 9. 11. 2018]. Dostupné z URL: <http://www.brothersoft.com/true-random-number-generator-169743.html>.

[20] *Senzory v mobilních telefonech od A do Z* [online]. 29. 11. 2014 [cit. 9. 11. 2018]. Dostupné z URL: <https://www.beryko.cz/blog/recenze/senzory-v-mobilnich-telefonech-od-a-do-z.html>.

[21] DUDÁČEK, Karel. *Mikrokontroléry: Doplňující text pro POS* [online]. Západočeská univerzita v Plzni, 2001. 7 s. [cit. 25. 10. 2018]. Dostupné z URL: <http://home.zcu.cz/~dudacek/Pot/mikrokontrolery.pdf>.

[22] COLESA, Adrian, Radu TUDIRAN a Sebastian BANESCU. Software random number generation based on race conditions. *2008 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. SYNASC 2008 [online]. 2008, 439-444 [cit. 9. 5. 2019]. DOI: 10.1109/SYNASC.2008.36. Dostupné z URL: <https://ieeexplore.ieee.org/document/5204851>.

[23] *ChaosKey true random number generator* [online]. [cit. 23. 11. 2018]. Dostupné z URL: <https://store.vikings.net/chaoskey>.

[24] YAO, Andrew C. Theory and application of trapdoor functions. *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)* [online]. IEEE, 1982, 80-91 [cit. 9. 11. 2018]. DOI: 10.1109/SFCS.1982.45. Dostupné z URL: <https://ieeexplore.ieee.org/document/4568378>.

[25] DAEMEN, Joan a Vincent RIJMEN. *The design of Rijndael: AES-the Advanced Encryption Standard*. New York: Springer, 2002. ISBN 978-3540425809.

[26] *What is LavaRnd?: LavaRnd is a Random Number Generator* [online]. [cit. 23. 11. 2018]. Dostupné z URL: <http://www.lavarand.org/what/index.html>.

[27] *Frequently Asked Questions (FAQ)* [online]. [cit. 01. 05. 2019]. Dostupné z URL: <https://www.random.org/faq/#Q2.4>.

[28] *Internet věcí (Internet of Things) - propojení různých zařízení díky internetu* [online]. [cit. 11. 11. 2018]. Dostupné z URL: <https://www.kodys.cz/technologie/internet-veci-internet-things>.

[29] LUETH, Kund Lasse. *State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating* [online]. 8. 8. 2018 [cit. 13. 5. 2019]. Dostupné z URL: <https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/>.

[30] WALLACE, Kyle, Kevin MORAN, Ed NOVAK, Gang ZHOU a Kun SUN. Toward Sensor-Based Random Number Generation for Mobile and IoT Devices. *IEEE Internet of Things Journal* 2016, 3(6), 1189-1201 [online]. [cit. 18. 11. 2018]. DOI: 10.1109/JIOT.2016.2572638. ISSN 2327-4662. Dostupné z URL: <https://ieeexplore.ieee.org/document/7477997>.

[31] SUN, Yingnan a B. LO. Random Number Generation Using Inertial Measurement Unit Signals for On-Body IoT Devices. *Living in the Internet of Things: Cybersecurity of the IoT - 2018* [online]. Institution of Engineering and Technology, 2018, 28 (9 pp.)-28 (9 pp.) [cit. 18. 11. 2018]. DOI: 10.1049/cp.2018.0028. ISBN 978-1-78561-843-7. Dostupné z URL: <http://digital-library.theiet.org/content/conferences/10.1049/cp.2018.0028>.

[32] *Quantis QRNG Chip: The world smallest QRNG for security, IoT & critical infrastructure applications.* 2018 [online]. [cit. 18. 11. 2018]. Dostupné z URL: <https://marketing.idquantique.com/acton/attachment/11868/f-025e/1/-/-/-/-/Quantis%20QRNG%20Chip_Brochure.pdf>.

[33] RUKHIN, Andrew, Juan SOTO a James NECHVATAL et al. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications* [online]. 2010. National Institute of Stand ards and Technology, 131 s. [cit. 25. 10. 2018]. Dostupné z URL: <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf>.

[34] BISKUP, Roman. *Vyhodnocování výsledků testování hypotéz na základě „p-value“* [online]. Jihočeská univerzita v Českých Budějovicích: Ekonomická fakulta, 2014, s. 3 [cit. 9. 11. 2018]. Dostupné z URL: <http://home.ef.jcu.cz/~birom/stat/cviceni/09/p_value.pdf>.

[35] BROWN, Robert G. Dieharder: A Random Number Test Suite: Version 3.31.1. *Robert G. Brown's General Tools Page* [online]. Duke University Physics Department, 2018 [cit. 9. 11. 2018]. Dostupné z URL: <http://webhome.phy.duke.edu/~rgb/General/dieharder.php>.

[36] *What is Eclipse?* [online]. [cit. 5. 4. 2019]. Dostupné z URL: <https://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Fint_eclipse.htm>.

[37] Interface MouseMotionListener. *ORACLE: Help Center* [online]. [cit. 12. 11. 2018]. Dostupné z URL: <https://docs.oracle.com/javase/7/docs/api/java/awt/event/MouseMotionListener.html>.

[38] *Android Development Tools for Eclipse* [online]. [cit. 12. 4. 2019]. Dostupné z URL: <https://marketplace.eclipse.org/content/android-development-tools-eclipse>.

[39] SensorManager. *Android Developers* [online]. [cit. 13. 4. 2019]. Dostupné z URL: <https://developer.android.com/reference/android/hardware/SensorManager>.

[40] SensorEventListener. *Android Developers* [online]. [cit. 13. 4. 2019]. Dostupné z URL: <https://developer.android.com/reference/android/hardware/SensorEventListener?hl=en>.

[41] SensorEvent. *Android Developers* [online]. [cit. 13. 4. 2019]. Dostupné z URL: <https://developer.android.com/reference/android/hardware/SensorEvent.html#values>.

Seznam symbolů, veličin a zkratek

A/D	Analogově digitální
BIN	Binární soubor
CPU	Central Processing Unit - Centrální procesorová jednotka
CSPRNG	Cryptographically secure pseudorandom number generator - Kryptograficky bezpečný pseudonáhodný generátor čísel
EPL	Eclipse Public Licence - Veřejná licence na Eclipse
FPGA	Field Programmable Gate Array - Programovatelné hradlové pole
FRO	Free Running Oscillator - Volně běžící oscilátor
GPU	Graphics Processing Unit - Grafický procesor
GUI	Graphical User Interface - Grafické uživatelské rozhraní
HTTPS	Hypertext Transfer Protocol Secure - Zabezpečený hypertextový přenosový protokol
IoT	Internet of Things - Internet věcí
JAR	Java ARchiver
LFSR	Linear-feedback shift register - Posuvný registr s lineární zpětnou vazbou
NIST	National Institute of Standards and Technology - Mezinárodní instituce standardů a technologií v USA
NIST STS	The NIST Statistical Test Suite - Soubor statistických testů vytvořených NIST
PRNG	Pseudo Random Number Generator - Generátor pseudonáhodných čísel
RNG	Random Number Generator - Generátor náhodných čísel
TRNG	True Random Number Generator - Generátor pravých náhodných čísel
TXT	Textový soubor
UWB	Ultra-Wideband - Ultraširoké pásmo

Seznam příloh

A Obsah přiloženého CD **68**

A Obsah přiloženého CD

- / ..... kořenový adresář přiloženého CD
  - xproch96_bakalarska_prace.pdf ..... elektronická verze bakalářské práce
  - generator_java ..... složka obsahující aplikaci
    - .settings
      - org.eclipse.core.resources.prefs
    - bin ..... složka obsahující zkompilované binární soubory
      - application
        - Gui.fxml
        - Main.class
        - RandomGenerator.class
      - resources
        - logo.png ..... logo aplikace
    - src ..... zdrojové soubory
      - application
        - Gui.fxml
        - Main.java
        - RandomGenerator.java
      - resources
        - logo.png ..... logo aplikace
    - .classpath
    - .projekt
    - build.fxbuild
    - generator.jar ..... spustitelný soubor