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 NÁHODNÝCH ČÍSEL VYUŽÍVAJÍCÍ BEZDRÁTOVÉ SÍTĚ

RANDOM NUMBER GENERATOR USING WIRELESS NETWORKS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Jan Frolka

AUTHOR

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

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Jan Frolka ***ID:*** 206591

Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Generátor náhodných čísel využívající bezdrátové sítě

POKYNY PRO VYPRACOVÁNÍ:

V diplomové práci proveďte analýzu možností realizací generátorů náhodných čísel. Zaměřte se na možné využití bezdrátových sítí (na využití RSSI, bitové chybovosti paketů, …) ke generování náhodných čísel. Popište způsoby testování náhodných čísel. Navrhněte a implementujte generátor náhodných čísel využívající bezdrátové sítě, jenž bude ovládán pomocí webové stránky. Webová stránka bude umožňovat generování náhodných čísel o požadované délce, jejich otestovaní na náhodnost pomocí různých testů a export vygenerovaných čísel do souboru. Přehledně prezentujte dosažené výsledky.

DOPORUČENÁ LITERATURA:

[1] LATIF, Rabia; HUSSAIN, Mukhtar. Hardware-based random number generation in wireless sensor networks (WSNs). In: International Conference on Information Security and Assurance. Springer, Berlin, Heidelberg, 2009. p. 732-740.

[2] FRANCILLON, Aurélien; CASTELLUCCIA, Claude. TinyRNG: A cryptographic random number generator for wireless sensors network nodes. In: 2007 5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops. IEEE, 2007. p. 1-7.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 19.5.2023

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

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Tato diplomová práce seznamuje čtenáře s typy generátorů náhodných čísel a možnostmi jejich realizace. Jsou zde popsány způsoby testování náhodných čísel a různé typy souborů statistických testů. Součástí práce je návrh a realizace vlastního generátoru náhodných čísel využívající bezdrátové sítě. Jako zdroj náhodnosti je zde využita síla signálu antény násobená hodnotou času od posledního přijatého paketu. Generátor je ovládán pomocí webové aplikace, která umožňuje uživateli vygenerovat, uložit a otestovat náhodná čísla pomocí sady testů NIST STS. Vygenerované sekvence z generátoru byly dále otestovány pomocí sady ENT. Výsledky jsou dále porovnány s generátorem využívající atmosférický šum z webové stránky.

KLÍČOVÁ SLOVA

Generátor náhodných čísel, bezdrátové sítě, python, testování, NIST STS.

ABSTRACT

This master's thesis introduces the reader to the types of random number generators and the possibilities of their implementation. The methods of testing random numbers and different types of statistical test sets are described. The thesis includes the design and implementation of a custom random number generator using wireless networks. The value of the signal strength multiplied by the difference in time since the previous packet was received is used as the source of randomness. The generator is controlled by a web application that allows the user to generate, store and test random numbers using the NIST STS test suite. The generated sequences from the generator were further tested using the ENT suite. The results are further compared with the generator using atmospheric noise from the website.

KEYWORDS

Random number generator, wireless networks, python, testing, NIST STS.

FROLKA, Jan. *Generátor náhodných čísel využívající bezdrátové sítě*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 68 s. Diplomová práce. Vedoucí práce: Ing. Vlastimil Člupek, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Jan Frolka
VUT ID autora:	206591
Typ práce:	Diplomová práce
Akademický rok:	2022/23
Téma závěrečné práce:	Generátor náhodných čísel využívající bezdrátové sítě

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Vlastimilovi Člupkovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

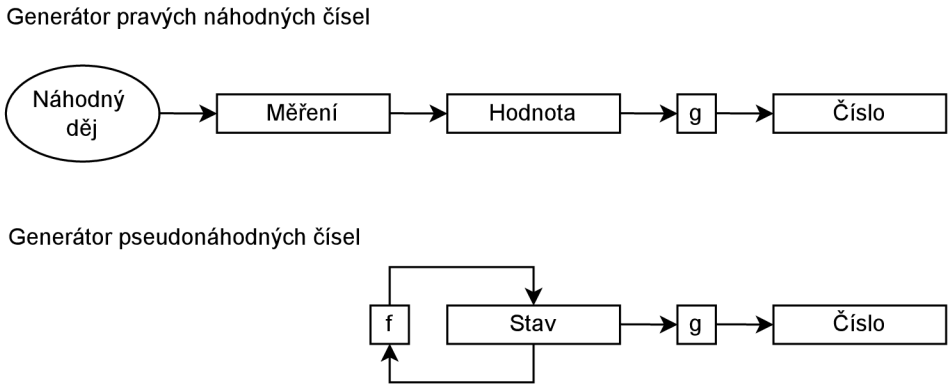
Tato diplomová práce se věnuje problematice generování skutečně náhodných kryptograficky bezpečných čísel a jejich využití v kryptografii, kde hraje důležitou roli například v asymetrické kryptografii, kde slouží například pro generování privátních a veřejných klíčů. Cílem práce je poskytnout systém, který by umožňoval generování takových čísel za využití dat z bezdrátových sítí.

Tato práce se v první kapitole zabývá různými způsoby generování náhodných čísel, jejich dělení a fyzikální jevy, na kterých jsou tyto generátory založeny. Jsou zde popsány techniky využité v již existujících generátorech náhodných čísel. Druhá kapitola se dopodrobna zabývá testováním náhodných čísel a popisem vybraných statistických testů. Některé tyto testy se poté využívají v této práci. Ve třetí kapitole je popsán standard IEEE 802.11, režimy síťových karet a možné využitelné parametry ke generování náhodných čísel. Je zde také uvedena struktura rámců a obsah hlaviček. Ve čtvrté kapitole je popsán způsob zachytávání paketů a vyhodnocení vhodných parametrů z komunikace, které by se daly použít v návrhu a implementaci v pozdějších kapitolách. Pátá kapitola obsahuje návrh vlastního generátoru náhodných čísel a způsob zobrazení uživateli. Jsou zde popsány technologie a principy využité při implementaci vlastního generátoru. Šestá kapitola popisuje realizaci vlastního generátoru náhodných čísel, zachytávání provozu a způsob zobrazení uživateli za pomoci webové stránky. Webová stránka umožňuje generování náhodných čísel o požadované velikosti, jejich zobrazení, stažení do souboru a následné otestování vybranými statistickými testy. Sedmá kapitola pojednává o výsledcích testování testy NIST a ENT.

1 Náhodná a pseudonáhodná čísla

Problematika generování náhodných čísel stojí na jejich opravdové náhodnosti. Jako nejjednodušší příklady generování náhodných čísel můžeme uvést například hod kostkou v rozmezí čísel 1 až 6 nebo hod mincí.

Pro dnešní využití je hod kostkou v digitálních systémech nevhodné a primitivní. V současné době dělíme systémy na pseudonáhodné generátory náhodných čísel označované anglicky jako PRNG (Pseudo Random Number Generator) a generátory pravých náhodných čísel TRNG (True Random Number Generator) [1].



Obr. 1.1: Rozdíl fungování generátorů [2].

Náhodné číslo je takové, které je vylosováno z posloupnosti čísel z nichž je každá hodnota zastoupena se stejnou pravděpodobností, tedy odpovídá rovnoměrnému rozdělení. Takové náhodné číslo je tedy statisticky nezávislé na ostatních [3].

1.1 Generátor pseudonáhodných čísel

Generátor pseudonáhodných čísel má ve svém jádru automat, který z matematické funkce, vytváří deterministickou a periodickou posloupnost čísel. Počátečním stavem funkce je semínko (seed). Takové generátory mají dokonalou rovnováhu výstupních hodnot a odpovídají skutečným náhodným hodnotám, nicméně po určitém čase vykazují korelaci funkce, tzn. že po čase generované posloupnosti začnou jevit určitou závislost. Výhodou je rychlé generování čísel, nevýhodou perioda opakování posloupností [4, 5].

Tyto generátory dělíme na lineární a nelineární. Liší se vlastnostmi a oblastmi využití. Lineární generátory mají výhodné statické vlastnosti, zatímco nelineární jsou vhodnější z hlediska kryptografické bezpečnosti [5].

1.1.1 Lineární kongruentní generátor - LCG

Jedná se o nejjednodušší lineární generátor, anglicky Linear Congruential Generator. Využíván byl už na počátku padesátých let dvacátého století a je definován vztahem:

$$x_{i+1} = (ax_i + b) \bmod m, \tag{1.1}$$

kde

- m je modul,
- a je multiplikativní konstanta,
- b je aditivní konstanta,
- x_i udává nám periodu,
- x_0 je počáteční hodnota semínka (seed).

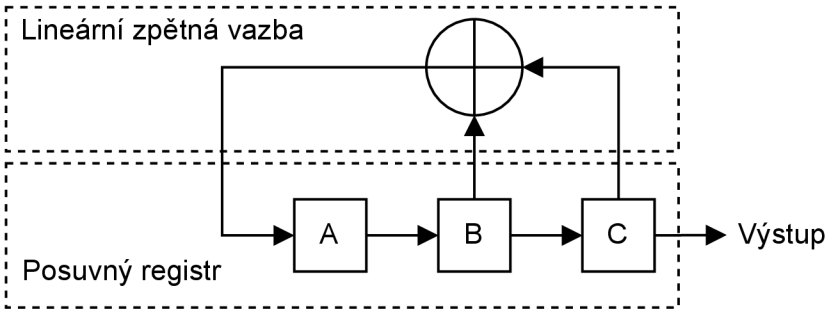
Přičemž musí platit podmínky:

- c a m nesmí být soudělná čísla,
- $a - 1$ je násobek každého prvočinitele m,
- $a - 1$ je násobek 4, pokud m je násobek 4.

Generátor produkuje pouze celá čísla a to v rozsahu $< 0, m - 1 >$. Počet hodnot je omezený a po m vygenerovaných číslech se nám posloupnost začne opakovat. x_0 udává plnou periodu a musí platit, $x_0 \in < 0, m-1 >$. Pokud bychom zvolili konstanty vhodně, potom se dostaneme na tzv. plnou periodu a maximální hodnota může být m [5, 6].

1.1.2 Lineární zpětnovazební posuvný registr - LFSR

Tento generátor můžeme vidět na obr. 1.2. Jeho struktura se skládá z posuvných registrů a lineární zpětné vazby. Každý průchod buňkami A, B nebo C ovlivní hodnoty bitů, přičemž výstup z buňky B a C se XORuje a vrací se nám jako vstup do buňky A. Zároveň hodnota z buňky C je vyvedena jako výstup celého bloku [4].



Obr. 1.2: Posuvný registr s lineární zpětnou vazbou [7].

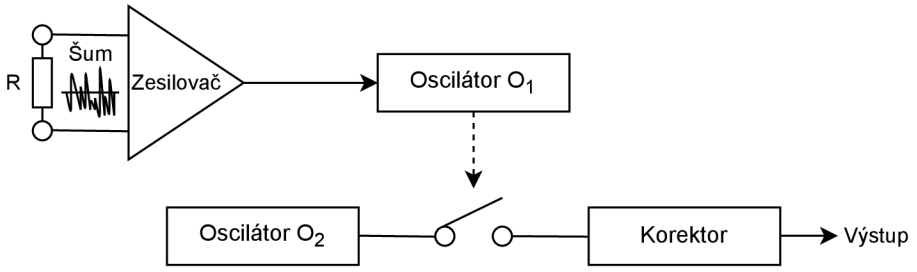
1.2 Generátor pravých náhodných čísel

Generátory pravých náhodných čísel jsou založeny na nedeterministickém procesu, zpravidla na fyzikálních jevech. Tento proces je zpětně nerekonstruovatelný a zároveň nepředvídatelný. Tyto generátory bývají hůře implementovatelné, protože je většinou potřeba k počítači připojit další zařízení, které nám bude dávat zdroje náhodných dat. Těmi můžou být například měření atmosférického šumu, rozpad atomových částic, počítadlo času mezi blesky, atd. Využít lze také šum na konektorech zvukové karty, šum mikrofonu, časování operací při práci pevného disku, intervaly mezi stlačením kláves, délky stlačení klávesy, změny napětí a proudu při průchodu součástkou na desce plošného spoje, teplota součástek, teplotní šum součástek [1, 3].

Celkově tyto generátory mají vynikající vlastnosti v generování opravdových náhodných posloupností, avšak jejich největší nevýhodou je jejich pomalá rychlost oproti PRNG [3, 8].

1.2.1 Generátory založené na šumu

Schéma generátoru náhodných čísel v procesorech Intel Pentium III využívající teplotní šum rezistoru můžeme vidět na obr. 1.3. Tento **teplotní šum** (také označován jako Johansonův šum), vzniká kvůli malým a náhodným změnám napětí, které je způsobeno tepelným pohybem elektronů v materiálu. Šum z rezistoru je přiveden na zesilovač, jehož signál zesílí pro následné zpracování. Napětí je poté přivedeno na oscilátor O_1, který kmitá náhodně dle velikosti napětí na vstupu. Oscilátor O_2 nám poté udává hodinové impulzy. Výsledné bitové hodnoty jsou přivedeny na vstup von Neumannova korektoru, který zahazuje dvojice stejných bitů a na výstup dává druhý bit z dvojice rozdílných bitů [2].



Obr. 1.3: Generátor binární posloupnosti [2].

Atmosférický šum je přírodním zdrojem náhodného děje způsobeném bleskovými výboji při bouřkách a je vhodný jako zdroj pro generování náhodných čísel. Každou vteřinu dojede po celém světě ke 40 záblesků, to je $3,5$ milionu záblesků

za celý den. Tyto záblesky se snadno šíří v poměrně velkých vzdálenostech. Součástí šumu jsou ale také jiné doprovodné jevy jako jsou vítr, sluneční paprsky nebo vesmírný prach dopadající na Zem [3, 9].

1.2.2 Generátory založené na chaosu

Generátory využívající chaos nezaručují náhodnost generovaných posloupností. Fungují na principu snímání kvantově mechanické neurčitosti. Příkladem může být detekce fotonu, nicméně se jedná o pomalé rychlosti generování náhodných čísel [1, 10].

Dalším příkladem může být využití chaotických polovodičových laserů nebo supermřížek. Zde se využívá chaotické oscilace proudu v materiálech. Teplotní šum součástek nám zde také vkládá určitou část chaotických změn. Polovodičové supermřížky jsou charakteristické vytvářením formací domén elektrického pole. Polovodičové součástky obsahují příměsy různých prvků, ty ovlivňují chování VA (Voltampérové) charakteristiky. To nám způsobí nelineární chování prvků při tunelování svou zápornou diferenciální vodivostí v závislosti na typu proudu [11].

1.2.3 Generátory založené na radioaktivním rozpadu

Dalším skvělým zdrojem náhodnosti je rozpad radioaktivních jader, protože jejich vlastností je samovolný rozpad. Poločas rozpadu můžeme definovat jako dobu, za kterou se aktivita jádra zmenší na polovinu hodnoty. Poločas rozpadu není předem znám a probíhá bez vnějších vlivů. Může se pohybovat v rozmezí několika sekund až po několik milionů let, což je závislé na typu radioaktivní látky. Jedná se tedy o zcela náhodný jev a nelze jej předvídat [1, 12, 13].

Protože je radioaktivní rozpad přítomen všude okolo nás, nemusíme mít k dispozici nutně radioaktivní prvek. Lze využít detektory, které zachycují přírodní rozpad thoria a uranu v zemské kůře nebo kosmické záření. Zde platí, čím blíže jsme okraji atmosféry, tím větší množství kosmického záření bude k dispozici [14].

1.2.4 Generátory založené na kruhových oscilátorech

Tyto generátory využívají transformaci elektrického šumu do volně běžících oscilátorů, kde nám přináší časové odchylky na vzestupné a sestupné hrany hodinového signálu. Ve frekvenčním spektru je můžeme označit za fázový šum nebo jako zpožďovací prvek v časové oblasti. Náhodnost z těchto elektrických šumů lze převést na náhodná čísla. Ty získáme jako řetězec bitů po navzorkování hodinového signálu v určitém intervalu [15].

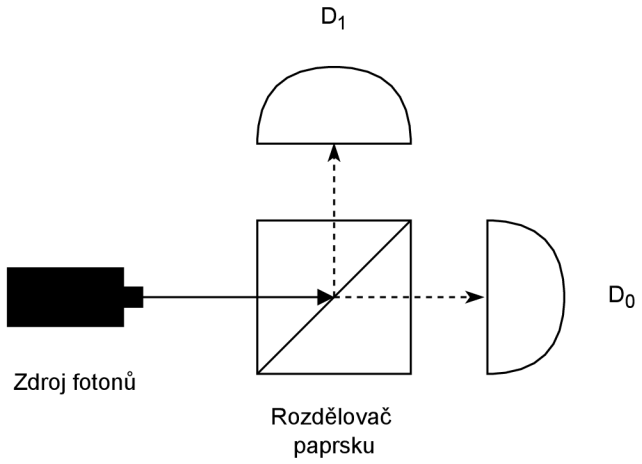
1.2.5 Generátory založené na chování uživatele

Ke generování lze využít uživatelova chování. Nejčastěji se snímá poloha kurzoru myši na monitoru. K tomu se využívají souřadnice X a Y, které se dále převádějí na přesnou polohu kurzoru.

Dále můžeme využít délku stisknutí klávesy v milisekundách. Zde ovšem vstupuje mnoho faktů. Pokud vezmeme příliš krátkou posloupnost čísel, zjistíme, že se nám některé hodnoty opakují. Dalším důležitým faktorem je, že konkrétní uživatel bude mít přibližně stejnou techniku psaní, tudíž i jeho rychlost bude stejná a proto bude generovat podobné hodnoty. To lze částečně eliminovat, pokud by se u počítače uživatelé střídali [1].

1.2.6 Generátory založené na kvantových stavech

Tyto generátory využívají náhodného chování subatomárních částic za určitých okolností. Z našich dosavadních znalostí fungování vesmíru se předpokládá, že události tohoto typu jsou nedeterministické. Pokud by tomu tak nebylo, znamenalo by to, že veškeré události od počátku velkého třesku jsou předem určené. Náhodnost je součástí kvantové mechaniky [3, 10].



Obr. 1.4: Schéma kvantového generátoru [10].

Na obrázku 1.4 můžeme vidět schéma kvantového generátoru. Skládá se ze světelného zdroje kterým může být světelná dioda, laser nebo zdroj fotonů. Dále z rozdělovače paprsku a detektoru. Rozdělovač má v sobě umístěno polopropustné zrcadlo, které zajišťuje buďto průchod paprsku do detektoru D_0 nebo odraz do detektoru D_1. Reflektivita tohoto rozdělovače je rovna $0,5$, což nám zajišťuje stejnou pravděpodobnost rozdělení paprsku. Detektor D_0 zapisuje na výstup logickou nulu a detektor D_1 logickou jedničku [16].

Pokud budeme mít k dispozici přirozené světlo, tak polovina světelného proudu projde na detektor D_0 a polovina světelného proudu bude odražena a vyslána na detektor D_1. Pokud však vyšleme jeden foton, tak nemáme dostatek energie, která by byla rozdělena a dostáváme se tak do superpozice. To znamená, že se elektron nenachází v žádné z energetických hladin. Pokud ovšem provedeme měření, tak se jeho stav změní a dostaneme konkrétní hodnotu energie, která vystupuje v superpozici. Tyto stavy jsou zcela náhodné a nelze je předvídat. Každé měření tedy bude generovat zcela odlišné hodnoty [16, 17].

2 Testování generátorů náhodných čísel

K testování určité posloupnosti náhodných čísel se využívají různé statistické testy. Těchto testů je velké množství a porovnávají, zda je získaná sekvence dat skutečně náhodná. Ty vyhledávají opakující se části dat a porovnávají, zda mají nějakou souvislost. Pokud tento test odhalí návaznost, pak daným testem neprojde. Samotný jeden test nám nemůže určit, zda jsou data opravdu náhodná. Proto se vždy provádí určitá sada testů, která rozhodne o náhodnosti dané sekvence [5].

2.1 Testování nulovou hypotézou

Před samotným testování je vhodné si uvést pár základních pojmů. Prvním pojmem je **statistická hypotéza**, kterou můžeme definovat jako tvrzení, které se týká neznámých parametrů. Testování **nulovou hypotézou** si označíme námi požadovaný výstup, tj. zda je sekvence dat náhodná. Opakem tvrzení H_0 je alternativní hypotéza označována jako H_1. Obě hypotézy se navzájem nevylučují, tzn. potvrdíme–li hypotézu H_1, neznamená automatické zamítnutí hypotézy H_0 [18, 19].

V tabulce 2.1 můžeme vidět logiku rozhodování testování nulovou hypotézou.

Tab. 2.1: Tabulka rozhodování testování nulovou hypotézou [19].

Správnost	H_0 přijímáme	H_0 nepřijímáme
H_0 je pravda	Správně	Chyba 1. druhu α
H_0 je nepravda	Chyba 2. druhu β	Správně

Dalším důležitým pojmem je **hladina významnosti** testu. Existuje totiž pravděpodobnost, že zamítneme hypotézu H_0 i přes to, že platí. Je potřeba brát v potaz fakt, že potvrzení nebo zamítnutí testované hypotézy provádíme na základě testování náhodného výběru dat. Může tedy dojít k nesprávnému určení, tyto chyby označujeme jako chyba prvního druhu α a chybu druhého druhu β [18, 19].

2.2 Sada testů NIST - Statistical Test Suite

Protože generování opravdu náhodných čísel je značně důležité, byl vytvořen standardizovaný soubor statistických testů určených pro otestování aspektů, které by se mohli vyskytovat u nenahodilé sekvence. Tento soubor vytvořil Národní institut standardů a technologie (NIST) v Americe a skládá se z patnácti testů. Každý test se zaměřuje na odlišné vlastnosti [18]. Tyto testy jsou vypsány níže:

1. Frekvenční (monobitový) test,
2. Frekvenční test v rámci bloku,
3. Test běhů,
4. Test nejdelšího běhu v bloku,
5. Test hodnot binární matice,
6. Spektrální test (diskrétní Fourierovy transformace),
7. Test shody nepřekrývajících se šablon,
8. Test shody překrývajících se šablon,
9. Maurerův „univerzální statistický“ test,
10. Test lineární složitosti,
11. Sériový test,
12. Test přibližné entropie,
13. Test kumulativních součtů,
14. Test náhodných výběrů a
15. Variantní test náhodných výběrů.

2.2.1 Frekvenční (monobitový) test

Test je zaměřen na podíl jedniček a nul v celé posloupnosti. Test porovnává, zda počet jedniček a nul v posloupnosti je přibližně stejný jako by se dalo očekávat ze skutečně náhodně sekvence. Protože se jedná o porovnání pouze 2 stavů (0 a 1), porovnává blízkost k hodnotě pravděpodobnosti $p_0 = 0,5$.

Nejprve test převede vstupní sekvenci na hodnoty -1 a $+1$ a jsou sečteny jako $S_n = X_1 + X_2 + \ldots + X_n$, kde $X_i = 2\varepsilon_i - 1$.
Poté se vypočítá $s_{obs} = \frac{|S_n|}{\sqrt{n}}$. Nakonec se vypočítá P-hodnota $= \mathrm{erfc}(\frac{s_obs}{\sqrt{2}})$ [20].

2.2.2 Frekvenční test v rámci bloku

Test se zaměřuje na rozdělení dat do bloků velikosti M a poté testování rozdílu jedniček v tomto bloku. Předpokládá se, že hodnota jedniček bude přibližně rovna $M/2$, což odpovídá rozdělení χ^2 [20].

2.2.3 Test nejdelší sekvence

Tento test se zaměřuje na sekvence stejných bitů které jsou v posloupnosti obsaženy do přerušení jinou hodnotou bitu. Délka k udává délku dané sekvence, která je ohraničena vpředu a vzadu hodnotami rozdílných bitů. Výstupem je hodnota, která nám říká, zda frekvence změn jedniček a nul je v pořádku, případně zda je příliš pomalá nebo rychlá [20].

2.2.4 Test nejdelší sekvence v bloku

Test se zaměřuje na sekvence stejných bitů rozdělených do bloků o velikosti M. Účelem testu je určit, zda v testované posloupnosti délka nejdelší sekvence jedniček v bloku je podobná hodnotě jaká by se dala očekávat v náhodné posloupnosti. Tento test se provádí jen u jedniček, protože se předpokládá, že pokud se zde objeví neočekávané hodnoty u jedniček, pak se projeví stejným způsobem u nul. Pro statistické výpočty se zde využívá χ^2 rozdělení [20].

2.2.5 Test hodnot binární matice

Test zkoumá hodnoty podmatic v rámci celé posloupnosti. Sekvence se rozdělí na matice $M * Q$ a počet těchto bloků bude roven: $N = (n/MQ)$, kde n je počet bitů v posloupnosti, M je délka segmentu a Q je počet vytvořených matic.

Výsledkem testu je ověření lineární závislosti mezi jednotlivými řadami s pevnou délkou a celou sekvencí. Tento test je také součástí v Diehard baterii testů [20].

2.2.6 Spektrální test (diskrétní Fourierovy transformace)

Tento test se snaží odhalit periodické rysy, které by naznačovaly odchylku od předpokládaných hodnot náhodnosti. Využívá se zde diskrétní Fourierovy transformace, kdy hodnoty by měly mít prahovou hodnotu 95 %. Poté se kontroluje počet vrcholů přesahujících tuto hodnotu a zda se významně liší od 5 %. Referenčním rozdělením pro testování je normální rozdělení [20].

2.2.7 Test shody nepřekrývajících se šablon

Pro tento test se využívá rozdělení sekvencí na M–bitové okno a vyhledává určitých cílových vzorů. Protože generátory mají tendenci tyto vzory v posloupnosti opakovat, snažíme se detekovat, zda tyto vzory existují. Pokud test odhalí v okně vzor, posune se o jeden bit za nalezený vzor a hledání se opakuje. Pokud se vzor v bloku neobjeví, okno se posunuje o jeden bit dále a zkouší se hledání vzoru znovu. Pro každý blok se vypočítávají výsledné hodnoty a ty se poté vyhodnocují [20].

2.2.8 Test shody překrývajících se šablon

Test funguje na stejném principu jako je předchozí test shody nepřekrývajících se šablon. S tím rozdílem, že pokud je nalezen vzor, okno se posouvá pouze o jeden bit dále a ne až za nalezený vzor. Jak už z názvu plyne, jedná se o překrývající se šablony [20].

2.2.9 Maurerův univerzální statistický test

Test počítá počet bitů mezi shodnými vzory, podle které jsme schopni poznat schopnost komprimace. Test je schopen říci, zda je možné sekvenci významně komprimovat bez ztráty informace. Pokud je možné sekvenci významně zkomprimovat, je považována za nenáhodnou. V opačném případě je sekvence náhodná [20].

2.2.10 Test lineární složitosti

Test se zkoumá délku posuvného registru s lineární zpětnou vazbou (LFSR). Test určuje, zda je posloupnost dostatečně komplexní na to, aby mohla být označena jako náhodná. Pokud máme dostatečně dlouhou délku registru LFSR, je posloupnost označena jako náhodná. Pokud je délka LFSR registru malá, lze tvrdit, že je testovaná posloupnost nenáhodná [20].

2.2.11 Sériový test

Tento test počítá četnost všech překrývajících se 2^m n–bitových vzorů v celé posloupnosti. Protože v náhodné posloupnosti má každý m-bitový vzor stejnou pravděpodobnost výskytu, test porovnává zda se tyto vzory vyskytují opravdu náhodně. Pokud bychom měli $m = 1$, tak by se jednalo o první, tj. Frekvenční monobitový test [20].

2.2.12 Test přibližné entropie

Test stejně jako u předchozího testu zkoumá četnost všech překrývajících se m–bitových vzorů v posloupnosti. V tomto testu se porovnává četnost překrývajících se bloků o velikosti m a $m + 1$ [20].

2.2.13 Test kumulativních součtů

Tento test se zaměřuje na zjištění, zda součet dílčích posloupností v sekvenci není příliš velký nebo příliš malý vzhledem k očekávaným hodnotám dané posloupnosti. Pro nenáhodné sekvence budou hodnoty střední vzdálenosti od počátečního bodu vysoké, naopak u náhodných se budou blížit k nule [20].

2.2.14 Test náhodných výběrů

V tomto testu se zaměřujeme na počet cyklů, které se v součtu navštíví přesně k–krát. Tento test je odvozen z jednotlivých součtů potom co je převedena sekvence

(0, 1) na sekvenci (−1,+1). Celý tento test se skládá z posloupnosti náhodně zvolených kroků, které začínají na jejím začátku a postupně se vrací. Testujeme zde počet návštěv určitého stavu v rámci cyklu a odchylek od očekávaných hodnot dané náhodnou posloupností. Jedná se o sérii osmi testů, kdy se testy liší v hodnotě stavu pro které se vypočítávají závěry, ty jsou: $-4, -3, -2, -1$ a $+1, +2, +3, +4$ [5, 20].

2.2.15 Variantní test náhodných výběrů

Tento test se zaměřuje na celkový počet návštěv jednotlivých navštívených stavů při součtu náhodné procházky. Výstupem testu je odhalení odchylky od náhodného pohybu, který je očekávaná v náhodné sekvenci. Celý test se skládá ze série 18 testů, pro stavy $-9, -8, -7, -6, -5, -4, -3, -2, -1$ a $+1, +2, +3, +4, +5, +6, +7, +8, +9$ [5, 20].

2.3 Sada testů ENT

Jedná se o soubor statistických testů, které se skládají z 6 rozdílných testů. Tyto testy jsou vypsány níže [18, 21]:

1. Entropie,
2. Možnosti komprese,
3. Chí–kvadrát test dobré shody,
4. Aritmetický průměr,
5. Monte Carlo hodnota pro π,
6. Koeficient sériové korelace.

2.3.1 Entropie

Tento test udává počet bitů informace obsažené v jednom bajtu dat. Jedná se o výpočet entropie v datovém toku, který se vypočítává z hustoty informace. Pro skutečné náhodné generátory by hodnoty informace měly být vyšší než $7,9$ bitů na bajt [21, 22, 23, 24].

2.3.2 Možnosti komprese

Z hodnoty entropie v předešlém testu se také vypočítává odhad, jak velké komprese lze u dané posloupnosti dosáhnout. Z daného souboru dat by měly být hodnoty komprese rovny ideálně 0, protože pro opravdový náhodný generátor platí, že je nekomprimovatelný [21, 22, 23, 24].

2.3.3 Chí–kvadrát test dobré shody

Tento test rozhoduje, zda jsou testovaná data rovnoměrně rozdělena. Udává hodnotu zkreslení testovaných dat. Jedná se o nejčastější test, který je velmi citlivý na chyby v generátorech. Pro dobré generátory jsou výsledky umístěny ve středu pravděpodobnostního rozdělení. Pro krajní hodnoty pravděpodobnosti větší než 99 % a menší než 1 % je nutné opakování testu na jiném souboru dat. Pokud ani tento opakovaný test neuspěje, je téměř jisté, že sekvence není náhodná. Pro hodnoty pravděpodobnosti 99–95 % a 1–5 % je sekvence označena jako podezřelá. Hodnoty pravděpodobnosti 95–90 % a 5–10 % je sekvence značena jako téměř podezřelá [21, 22, 23, 24].

2.3.4 Aritmetický průměr

Výpočet aritmetického průměru testovaných dat nám udává výsledek součtu všech bajtů dělený délkou souboru. Pro dostatečně náhodná data by se měl průměr bajtů rovnat přibližně hodnotě $127,5$ a pro bity $0,5$, což je hodnota 50% z celkového souboru. Pokud se hodnota průměrů výrazně liší od těchto hodnot, jsou hodnoty posloupnosti trvale nízké nebo vysoké [21, 22, 23, 24].

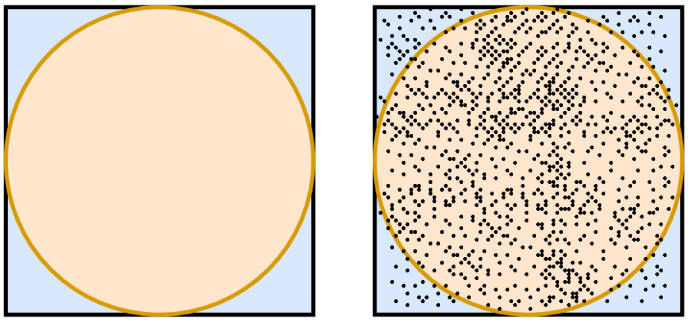
2.3.5 Monte Carlo hodnota pro π

V tomto testu je šestice po sobě jdoucích bajtů použita jako souřadnice bodu ve čtverci. Dále je vepsána kružnice a body, které leží uvnitř poloměru kružnice jsou považovány za zásah.

Procento bodů ležících uvnitř kružnice je poté použito pro výpočet hodnoty π. Pro velké soubory dat se tato hodnota blíží opravdové hodnotě π, pokud se jedná o opravdu náhodné čísla. Počítá se tedy jako procentuální rozdíl od reálného π. Zanesení bodů do souřadnic můžeme vidět na obr. 2.1 [21, 22, 23, 24].

2.3.6 Koeficient sériové korelace

Koeficient nám udává, jak moc je každý bajt v souboru závislý na bajtu předchozím. Pro náhodné sekvence by tato hodnota měla být co možná nejblíže nule a měla by být menší než $0,005$. Nenáhodné sekvence mají hodnotu sériové korelace kolem $0,5$. Pro silně předvídatelná data mají hodnotu sériového korelačního koeficientu blížící se 1 [21, 22, 23, 24].



Obr. 2.1: Monte Carlo π [25].

2.4 Diehard

Diehard je baterie obsahující sérii patnácti statistických testů vytvořených Georgem Marsagliem, který tyto testy publikoval již v roce 1995. Každý z těchto testů vrací p-hodnotu, který by měla být rovnoměrně rozdělena mezi 0 a 1. Testy by se měly opakovat na jiných souborech dat generovaných stejným generátorem, abychom prokázaly spolehlivost výsledku [21, 22, 23, 24]. Test byl nahrazen robustnější sadou zvanou TestU01, která je popsaná blíže v [26]. Seznam testů je vypsán níže:

1. Birthday Spacings,
2. Overlapping Permutations,
3. Ranks of 31x31 and 32x32 Matrices,
4. Ranks of 6x8 Matrices,
5. Monkey Tests on 20-bit Words,
6. Monkey Tests OPSO, OQSO, DNA,
7. Count the 1's in a Stream of Bytes,
8. Count the 1's in Specific Bytes,
9. Parking Lot Test,
10. Minimum Distance Test,
11. Random Sphere Test,
12. The Squeeze Test,
13. Overlapping Sums Test,
14. Runs Test,
15. The Craps Test.

2.5 Dieharder

Dieharder je nástroj navržený jak pro testování samotných generátorů náhodných čísel tak i vygenerovaných posloupností. Obsahuje některé zmodifikované testy ze sady Diehard a STS (Statistical Test Suite). Účelem vytvoření těchto testů byla možnost testování velké posloupnosti náhodných čísel. Proto se také upřednostňuje testování přímo generátorů namísto pouze generované posloupnosti. Je tu i možnost testování generovaných posloupností, je tu však podmínka dostatečného vzorku dat, kde dostáváme soubory o několika Gigabytech. Díky volné dostupnosti se jedná o poměrně variabilní nástroj. Umožňuje nám vkládání vlastních testů a generátorů.

Dieharder je poměrně silný nástroj, který dokáže i poměrně slabý generátor dotlačit k jednoznačnému selhání. V rámci přesnosti se zde bavíme o jednotkách desetitisíciny procenta, namísto původních 1–5 % možného selhání [24, 27, 28].

3 Standard IEEE 802.11

Tento standard vznikl v roce 1997 pracovní skupinou zvanou 802.11 mezinárodní instituce IEEE (Institute of Electrical and Electronics Engineers) za účelem zbavení Ethernetu drátů. První pokusy vznikaly již od roku 1986, kdy se narazilo na problémy se vzájemnou kompatibilitou. Od té doby se zvyšovala potřeba vytvoření společného standardu. Prvotní označení neslo název „bezdrátový Ethernet“ a starala se o testování a dodržování standardu organizace zvaná WECA (Wireless Ethernet Compatibility Alliance), která se později přejmenovala na Wi–Fi Alliance. Odtud známe dnešní označení Wi–Fi (Wireless Fidelity) [31, 32].

V tabulce 3.1 můžeme vidět, kdy který standard vznikl, jejich značení, pásma a maximální rychlosti. Zároveň za zmínku stojí standard 802.11be neboli Wi–Fi 7, který slibuje rychlost až 40 Gbit/s a jeho vydání je naplánováno na rok 2024 [33].

Tab. 3.1: Tabulka standardů, vznik a podporované rychlosti [31].

Standard	**Označení**	**Rok vydání**	**Pásmo**	**Maximální rychlost**
			[GHz]	**[Mbit/s]**
802.11	Wi–Fi 0	1997	2,4	2
802.11b	Wi–Fi 1	1999	2,4	11
802.11a	Wi–Fi 2	1999	5	54
802.11g	Wi–Fi 3	2003	2,4	54
802.11n	Wi–Fi 4	2009	2, 4 5	600
802.11ac	Wi–Fi 5	2013	5	3 466,8
802.11ax	Wi–Fi 6, Wi–Fi 6E	2019	2,4 5	600–9 608
802.11be	Wi–Fi 7	2024	2,4 5 6 7	40 000

3.1 Režimy síťových karet

Síťové karty se mohou nacházet v několika rozdílných režimech a určují nám, jak se dané zařízení bude chovat. Vše záleží na podpoře čipsetu konkrétní bezdrátové síťové karty a použitém ovladači.

Tyto režimy jsou: **Master** (zařízení se chová jako přístupový bod - AP (Access Point)), **Managed** (zařízení je schopné se připojit a komunikovat jen s asociovaným AP), **Ad–hoc** (zařízení je schopno komunikovat jen s ostatními zařízeními typu Ad-hoc ve svém dosahu), **Mesh** (v podstatě síť typu Ad–hoc s rozdílem, že zařízení jsou schopna komunikovat s ostatními napřímo), **Repeater** (zařízení přijme na vstup signál, zesílí ho a pošle ho dál), **Monitor** (zařízení přijímá veškerou komunikaci na daném kanálu), **Promiscuous** (zařízení zachytává veškeré pakety v síti ke které je asociováno) [31, 34].

3.1.1 Promiskuitní režim

Zařízení v tomto režimu je schopno zachytávat veškeré pakety v dané síti. Je potřeba být připojen ke konkrétní síti. Všechna zařízení nepodporují tento režim. V promiskuitním režimu karty odstraňují Frame header, takže se ke zpracování dostanou i pakety, které nejsou určené danému zařízení [31].

3.1.2 Monitorovací režim

Zařízení v tomto režimu pasivně naslouchá na vybraném kanále. Karta není připojena k síti a nevysílá žádná data. Karta zachytává i poškozené pakety, protože ignoruje pole kontrolního součtu rámce (CRC). Zachycená komunikace obsahuje i datové rámce, ale jejich obsah je zašifrován [31, 35].

3.2 RSSI - Received Signal Strength Indication

Na základě měření výkonu přijatého signálu, vlivu okolního prostředí a vzdálenosti od vysílače nám udává sílu přijatého signálu. Vliv prostředí zde hraje velkou roli a významně nám ovlivňuje hodnoty přijatého signálu. Udává se v decibelech na miliwattu [36, 37].

$$RSSI = 10 \cdot \log\left(\frac{P}{0,001}\right) \qquad [\text{dBm}] \tag{3.1}$$

3.3 Struktura rámců

Rámce ve standardu 802.11 jsou obohaceny o 2 typy hlaviček. Jednou z nich je Radiotap header a druhou IEEE 802.11 radio information. Každá z nich přenáší jiný typ informací [39].

3.3.1 Radiotap header

Představuje nástroj pro předání informací z ovladače síťové karty pro uživatelské aplikace a naopak. Informace, které tato hlavička přenáší se dělí do dvou kategorií na povinné a volitelné informace. To přináší určitou variabilitu v předávaných informací. Oproti hlavičce Prism nebo AVS, které mají předem stanovenou délku a systémy tak s fixní délkou počítají. Například v hlavičce Prism není dostatek místa pro kontrolní součet FCS (Frame Check Sequence), protože má danou fixní délku 144 B a není ji tak možné rozšířit o toto pole. To lze až u Radiotap hlavičky u níž se dá prakticky definovat libovolný počet polí za pomoci příznakových bitů, ty lze vidět na obr.4.2 [38, 39].

Povinná pole Jsou přítomná vždy a jejich velikost je dána. Na obr.3.1 lze vidět rozmístění povinných položek Radiotap hlavičky.

Obr. 3.1: Hlavička Radiotap [39].

Výpis jednotlivých polí je uveden níže:

- **version** - udává verzi hlavičky, v současné době je použita hodnota 0, přidáním polí se číslo verze nemění, má velikost 1 B,
- **pad** - v současné době se nevyužívá, má velikost 1 B,
- **len** - udává celkovou délku radiotap dat spolu s radiotap hlavičkou, má velikost 2 B,
- **present** - obsahuje informaci o přítomných příznacích následované po radiotap hlavičce, má velikost 4 B.

Nepovinná pole Obsahují dodatečné informace, které daná síťová karta podporuje. Je na výrobci síťové karty, se kterými poli bude umět daný hardware pracovat. Je zde nutné dodržet souslednost daných polí, s ohledem na jejich velikost. Uvádí se pole v pořadí jak jsou definována v present rámci. Jak můžeme vidět na obrázku 4.2, v poli *present* jsou nejprve uvedeny hodnoty jako Flags, Rate, Channel, dBm Antenna Signal, Antenna, a RX flags [38].

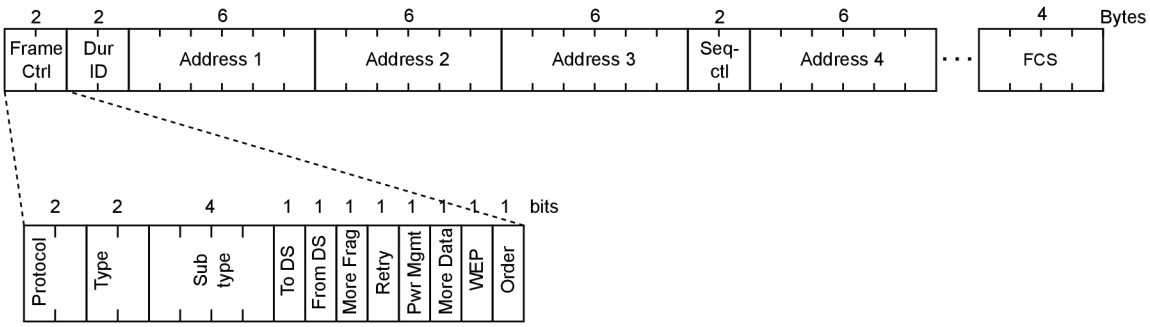
- **Flags** - obsahuje informace o vysílaných a přijatých rámců jako je například zda rámec obsahuje WEP šifrování, zda je rámec fragmentován, zda obsahuje pole FCS nebo zda neprošel kontrolou FCS, je o velikosti 8 bitů,
- **Rate** - udává hodnotu datové rychlosti pro vysílaných nebo přijímaných rámců, je o velikosti 8 bitů,

- **Channel** - skládá se ze dvou hodnot, první je vysílací nebo přijímací frekvence a druhá jsou příznaky, které nesou informaci o použité modulaci, má velikost 2 x 16 bitů,
- **dBm Antenna Signal** - udává velikost signálu antény od referenční hodnoty, toto pole je velikosti 8 bitů,
- **Antenna** - udává identifikátor vysílací nebo přijímací antény, první anténa je vždy 0, pole o velikosti 8 bitů,
- **RX flags** - informace o nastavení rámců, indikuje poškození FCS CRC, je o velikosti 16 bitů.

Těchto polí je mnohem více, ale výše jsou uvedena pouze ta pole, která jsou přítomna na obrázku 4.2. Více informací ohledně ostatních polí, lze nalézt v literatuře [38].

3.3.2 802.11 radio information

V této hlavičce se přenášejí informace nezbytné pro fungování komunikace bezdrátových sítí. Na obrázku 3.2 je možné vidět typy polí obsažených v této hlavičce. Rámce dělíme do několika skupin na managementové, řídící a datové [39].



Obr. 3.2: Hlavička 802.11 radio information [39].

Management rámce Jsou rámce pro správu sítě. Přenášejí informace k vytvoření spojení mezi koncovým a přístupovým bodem, informace o bezdrátových sítí a další. Popis jednotlivých rámců je uveden níže [39].

- **Beacon** - jsou vysílány periodicky přibližně každých 100 ms, aby informovaly o přítomnosti bezdrátové sítě a pro synchronizaci klientů.
- **Probe request** - koncové zařízení vysílá požadavek na přístupový bod pro zjištění dostupných sítí.
- **Probe response** - přímá odpověď na Probe request, přístupový bod odesílá možné SSID pro připojení.

- **Association request** - použije koncová stanice pro připojení k vybranému přístupovému bodu.
- **Association response** - pokud koncová stanice splní požadavky přístupového bodu vytvoří se Association ID a odpovídá spolu se zprávou o úspěchu pro připojení.
- **Reassociation request** - vysílá se pokud koncová stanice opustí nebo se přesune v rámci obsluhované oblasti, zahrnuje informaci o starém přístupovém bodu, aby nový přístupový bod mohl kontaktovat ten starý a mohli si předat informace o přidružení.
- **Reassociation response** - je vysílána jako odpověď na Reassociation request a na základě informací poskytnutých v requestu přístupový bod buďto povolí nebo nepovolí připojení k novému přístupovému bodu.
- **Disassociation** - slouží k ukončení navázaného spojení, například protože stanice se snaží použít neplatné parametry nebo je neaktivní.
- **Authentication** - rámce pro kontrolu kompatibility nebo šifrování, snaží se ověřit typ zařízení.
- **Deauthentication** - rámec pro ukončení spojení navázané v rámci autentizace.

Řídící rámce Pomáhají doručování datových a managementových rámců. Slouží k řízení přístupu k médiu. Neobsahují tělo, jen hlavičku a návěští [39, 40].

- **Request to Send** (RTS) - rámce slouží k rezervaci média pro přenos datového rámce, pokud chce stanice zaslat datový rámec. musí počkat na odpověď CTS.
- **Clear to Send** (CTS) - potvrzení požadavku RTS, značí, že je médium rezervováno pro přenos datového rámce.
- **Acknowledgment** (ACK) - slouží k potvrzení o úspěšně přeneseném datovém rámci, vysílá se vždy, pokud byl datový rámec přenesen bez poškození. V opačném případě se ACK neodešle a odesílatel zašle datový rámec znovu.
- **Power–Save Poll** (PS–Poll) - je zasláno koncovou stanicí po probuzení z úsporného režimu. Dává signál přístupovému bodu, že je připravena přijmout rámce uložené v mezipaměti v čase, kdy byla v režimu spánku.

Datové rámce Slouží k přenosu dat nebo k vyvolání události. Máme 4 základní typy zpráv, ty jsou uvedeny níže [40].

- **Data** - slouží ke komunikaci mezi stanicemi nepodporující QoS.
- **QoS Data** - slouží ke komunikaci mezi QoS stanicemi.
- **Null Data** a **QoS Null Data** - přenášejí se prázdné rámce, může indikovat probouzení nebo ukládání do úsporného režimu.

4 Zachytávání provozu

Pro zachytávání bezdrátového datového provozu bylo vybráno prostředí Kali Linux verze 6.0.0 pro jeho skvělé uzpůsobení právě pro analýzu datového provozu. Pro zachytávání provozu v monitorovacím režimu byl vybrán USB adaptér Alfa Awus 1900. Veškerá práce byla prováděna přes virtualizační nástroj VMware Workstation 17 Player.

4.1 Zachytávání paketů

Prvně bylo nutné provést aktualizaci jádra kernelu na nejnovější verzi. Nejprve byly staženy aktuální informace o balíčcích.

```
sudo apt update
```

Dalším příkazem proveden upgrade balíčků a hlavně jádra systému Kali Linux. Jedná se o zdlouhavý proces, délka trvání závisí na rychlosti připojení lokální sítě.

```
sudo apt upgrade
sudo apt dist-upgrade
```

Dále bylo nutné provést instalaci linuxových hlavičkových zdrojových kódů, vložení odkazu pro závislosti, které se nainstalují pomocí `dkms` balíčku.

```
apt install linux-headers-$(uname -r)
apt install build-essential bc dkms git libelf-dev
```

V dalším kroku byl stažen ovladač přes odkaz na github. Nejprve bylo nutné naklonovat složku, zkompilovat a nainstalovat ovladač. Poté už stačilo operační systém restartovat.

```
mkdir -p ~/driver
cd ~/driver
git clone https://github.com/aircrack-ng/rtl8814au
cd rtl8814au
make dkms_install
reboot
```

Po restartu systému bylo vhodné zkontrolovat verze nainstalovaného ovladače a jádra systému Kali Linux za pomoci příkazu `dkms status`. Pomocí `airmon-ng` bylo možné přepnout síťovou kartu do monitorovacího režimu. Před přepnutím bylo nutné najít a ukončit všechny procesy, které kartu využívají a mohli by bránit v přepnutí do jiného módu. Dalším příkazem byla karta přepnuta do monitorovacího režimu.

```
airmon-ng check kill
```

```
airmon-ng start wlan0 1
```

Nyní je karta schopna zachytávat pakety s označení Dot11 na kanále 1. Karta v tomto režimu dokáže monitorovat jen na konkrétním kanále. Pokud by byl zadán příkaz `airmon-ng start wlan0` bez definice kanálu, karta začne defaultně monitorovat kanál číslo 10.

4.2 Výpis zachycených hodnot

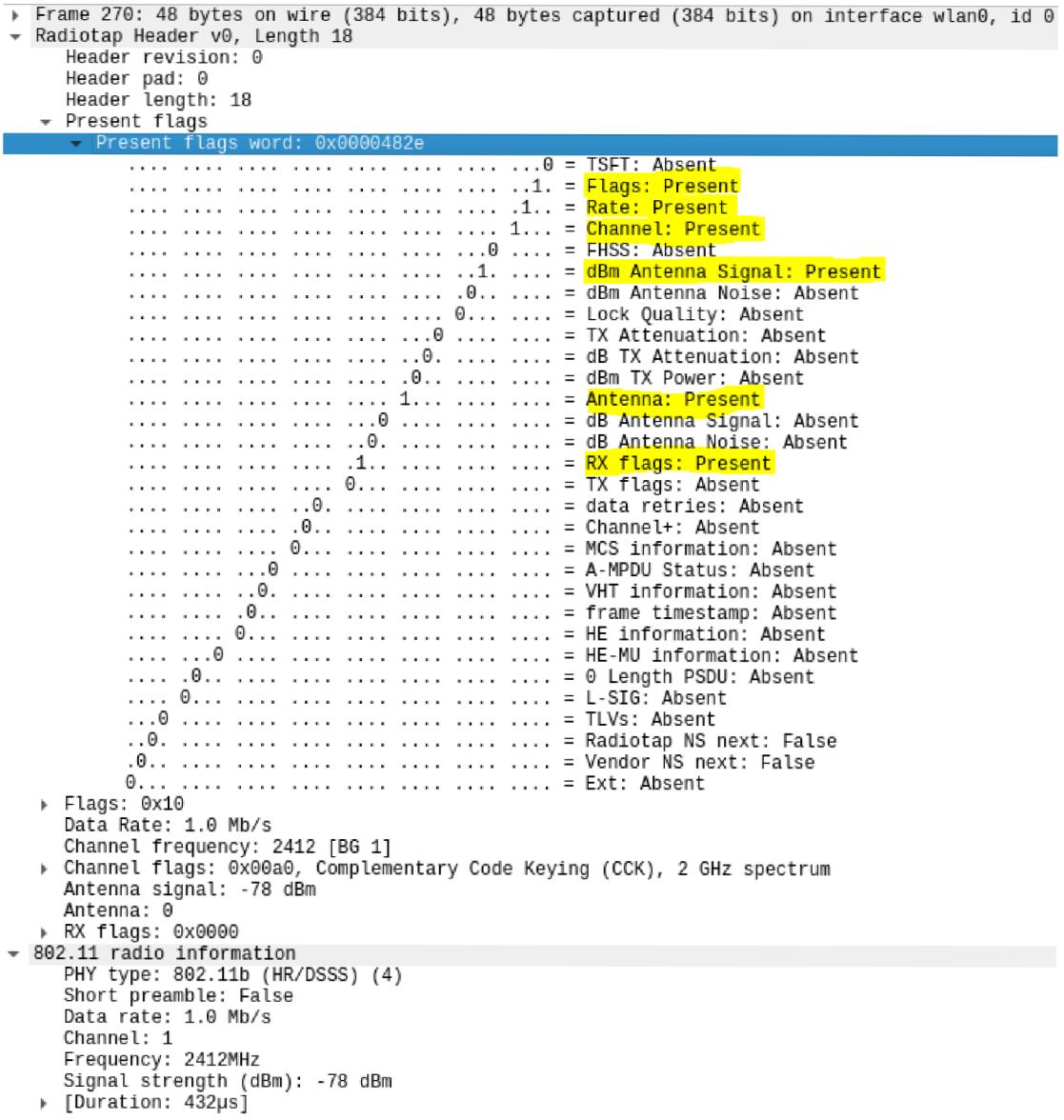
K analýze datového provozu bylo využito programu Wireshark. Na obrázku 4.2 můžeme vidět příznaky, které říkají co za informace paket obsahuje. V tomto paketu lze vyčíst:

- Příznaky - hodnota 0x10 značí, že na konci je kontrola rámce FCS,
- Přenosová rychlost - značí přenosovou rychlost,
- Kanál - frekvenci a číslo kanálu,
- Anténní signál - hodnota RSSI,
- Antény - počet antén,
- Přijaté příznaky - informace, že příznak je součástí paketu.

Jak lze vidět na obr. 4.1, parametry se budou nacházet v *Radiotap Header* a 802.11 *radio information*. Bohužel ze zachycených hodnot vyšlo najevo, že jsou parametry sítě předem definované (například frekvence kanálů, přenosová rychlost) a tak nejsou vhodným zdrojem náhodnosti. Jako jediný parametr, který vykazuje náhodné hodnoty je Antenna signal, neboli hodnota RSSI.

```
▸ Frame 295: 260 bytes on wire (2080 bits), 260 bytes captured (2080 bits) on interface wlan0, id 0
▾ Radiotap Header v0, Length 18
    Header revision: 0
    Header pad: 0
    Header length: 18
  ▸ Present flags
  ▸ Flags: 0x10
    Data Rate: 1.0 Mb/s
    Channel frequency: 2412 [BG 1]
  ▸ Channel flags: 0x00a0, Complementary Code Keying (CCK), 2 GHz spectrum
    Antenna signal: -74 dBm
    Antenna: 0
  ▸ RX flags: 0x0000
▾ 802.11 radio information
    PHY type: 802.11b (HR/DSSS) (4)
    Short preamble: False
    Data rate: 1.0 Mb/s
    Channel: 1
    Frequency: 2412MHz
    Signal strength (dBm): -74 dBm
  ▸ [Duration: 2128µs]
▸ IEEE 802.11 Beacon frame, Flags: ........C
▸ IEEE 802.11 Wireless Management
```

Obr. 4.1: Výpis zachyceného paketu.



Obr. 4.2: Příznaky dostupné ve vybraném paketu.

5 Návrh generátoru

Pro vytvoření generátoru náhodných čísel je využito programovacího jazyku Python (verze 3.10). Na obr. 5.1 lze vidět schéma navrženého generátoru náhodných čísel. Nejprve jsou vyčítány hodnoty RSSI a čas od posledního přijatého paketu. Hodnota času od posledního přijatého paketu však není součástí a musí být počítána pomocí vytvořené funkce.

Tyto hodnoty budou dále vstupovat do hešovací funkce, která zajistí velkou změnu na výstupu funkce oproti malým změnám na vstupu. Výstupem bude vytvořený polynom, který bude dále vstupovat do generátoru v podobě LFSR registru. Výstupní sekvence generovaných bitů se bude porovnávat, zda splňuje minimální určenou délku bitů. Pokud ne, bude se celý proces opakovat. V opačném případě bude vygenerovaná posloupnost vyslána jako výsledek generování a zobrazena na webové stránce.

Na obrázku 5.2 je zobrazen diagram zobrazení na webové stránce. Uživatel provede definici vstupních parametrů jako je délka požadované sekvence. Po stisknutí tlačítka Generate, odešle webová stránka požadavek na vyčtení hodnoty RSSI a doby od posledního přijatého paketu v mikrosekundách. Tyto hodnoty poté přecházejí do generátoru náhodných čísel, kde se provedou operace a vytvoří se výstupní sekvence, která porovnává výstupní délku zadanou uživatelem. Pokud by délka nebyla dostatečně dlouhá, provede se generování znovu. Pokud výstupní sekvence bitů splňuje požadavek na minimální délku, předá se dále webové stránce, která zobrazí výsledek uživateli.

5.1 Python

Je programovací jazyk vyšší úrovně navržený Guidem van Rossumem již v roce 1991. Jedná se o multiplatformní nástroj a je populární díky své jednoduché syntaxi, která je vhodná pro začínající programátory. Komunita kolem Pythonu je velká a díky její vysoké aktivitě lze jazyk rozšířit o množství funkcionalit, které v základu nejsou součástí jeho jádra. Jsou součástí vývoje jazyka, mezi hlavní výhody široké komunity je psaní dokumentací, tvorba knihoven a frameworků a také snadno přístupná fóra kde komunita řeší své problémy. Python se v dnešní době těší velké oblibě. V posledních letech si drží hlavní místo v TIOBE a PYPL indexech [41].

TIOBE index (The Importance of Being Earnest) udává popularitu jednotlivých programovacích jazycích, které uživatelé nejčastěji vyhledávají ve vyhledávačích Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube a Baidu [41, 42].

PYPL index (PopularitY of Programming Language) přináší data přímo z Google Trends a zaměřuje se na celosvětovou popularitu vyhledávání [43].

Tab. 5.1: TOP 5 nejvyhledávanějších prog. jazyků za poslední rok [43].

Rank	Language	Share	Trend
1	Python	27,27 %	, −0,5 %
2	Java	16,35 %	−1,6 %
3	JavaScript	9,52 %	+0,2 %
4	C#	6,62 %	−0,3 %
5	C/C++	6,55 %	−0,4 %

S rozvojem strojového učení a datové analýzy je v těchto odvětvích hojně vyhledáván kvůli jednoduché manipulaci s daty a vytváření složitějších struktur. Je také využíván mezi populárními aplikacemi jako je například YouTube, Spotify, Instagram, Dropbox a jiné [41].

V této práci se bude využívat výhradně jazyk Python z důvodu snadného rozšíření o dostupné knihovny a jednoduchost nasazení na zařízeních Raspberry Pi.

Numpy Numpy představuje základní balíček pro vědecké výpočty. Umožňuje vytvářet vícerozměrná pole, matematické a logické operace, manipulaci a třídění, základní lineární algebru, diskrétní Fourierovy transformace, základní statistické opera a mnoho další [44]. Tato knihovna je licencovaná jako BSD (Berkeley Software Distribution) neboli zcela volně šiřitelná. Stejně jako MIT (Massachusetts Institute of Technology) je kód svobodně šiřitelný a využitelný při uvedení původního autora.

Scapy Je program určen pro práci s pakety v Pythonu. Dokáže pakety přijímat, odesílat, odposlouchávat i pozměňovat. Za jeho pomoci lze pracovat i s pakety, které nejsou určené danému zařízení za pomoci využití monitorovacího režimu síťové karty. Tento program dokáže nahradit funkcemi programy jako jsou arpspoof, arping ale také Nmap, tcpdump a tshark. Mezi speciální funkcionality oproti ostatním programům patří injektování vlastních rámců v bezdrátových sítích, odesílání neplatných rámců nebo kombinací technik jako je VLAN hopping a ARP cache poisoning [45]. V této práci je využito scapy knihovny pro příjem a vyčítání parametrů bezdrátové komunikace, konkrétně hodnot RSSI z Radiotap hlavičky.

Flask Je webový micro framework určený pro programovací jazyk Python. Umožňuje vkládání rozšíření pro další funkcionality. Flask je vhodný pro implementaci funkcionalit pokud chceme v aplikaci využívat například vstupních dat z webové aplikace. V takovém případě dochází totiž k podobným funkcionalitám jako by se jednalo o desktopovou aplikaci. Flask umožňuje poskytování statických souborů. Tyto soubory je nutné vložit do složky zvané `static`. Jednotlivé vstupní body pro

tento framework jsou vytvářeny dekorováním odpovídající funkce pomocí `route`. V parametrech této dekorace je možné specifikovat cestu a metodu dotazu, které bude tento koncový bod přijímat. V takto dekorované metodě je poté možné získat parametry pomocí proměnné request. Lze získávat parametry předávané v URL adrese nebo také z dat v těle dotazu ve formátu JSON. Návratová hodnota z této metody je automaticky serializována také do formátu JSON. Flask je ideální pro malé projekty pro jeho jednoduchost a rychlost vytvoření, následné testování a úpravu webových aplikací [46, 47].

Jinja2 Je systém pro tvorbu šablon ve značkovacím jazyce HTML. Tento systém je hojně využíván v ekosystému jazyka Python. Výsledné soubory nemusí být pouze v jazyce HTML ale je také možné použít jiné značkovací jazyky jako je JSON, XML nebo jiné textové soubory. Tento systém umožňuje využití různých řídících sekvencí, které mohou modifikovat strukturu souboru. Tyto řídící sekvence mohou být cykly, podmínky a jiné. V rámci šablon lze využít proměnné, které jsou ve výsledném souboru nahrazeny odpovídajícími hodnotami. Celkový systém tak efektivně kombinuje šablonu a vstupní data pro vytvoření výsledného souboru. Proměnné nemusí být pouze primitivní typy, ale je možné použít i komplexní datové typy jako jsou slovníky [48].

V rámci šablon je také možné definovat makra. Mohou obsahovat předdefinovanou strukturu dat. Takto vytvořená makra je možné zavolat a tato struktura bude vložena na místo volání makra. Makra mohou obsahovat parametry volání. Data z těchto parametrů je možné využít v rámci definované struktury [48].

Některé šablony mohou nabývat obrovských velikostí a systém tedy umožňuje rozdělení do několika menších souborů. Ty lze následně propojovat pomocí klíčových slov Include a Import. Include umožňuje propojení vytvářených textů zatímco Import zprostředkovává přístup k definovaným makrům. Tento šablonový systém byl využit při tvorbě uživatelského rozhraní aplikace ve frameworku Flask [48].

Výpis 5.1: Příklad šablony systému Jinja2 [48]

```
<h1>{{ test_name }} Results</h1>
<ul>
{% for student in students %}
<li>
  <em>{{ student.name }}:</em>
  {{ student.score }}/{{ max_score }}
</li>
{% endfor %}
</ul>
```

5.2 Hešovací funkce

Vytváří z libovolné délky bitové posloupnosti sadu znaků pevné délky, tzv. heš. Tyto funkce se vyznačují značnou **diverzifikací** při změně malého množství bitů. Už při změně jednoho bitu, dochází ke změně až poloviny bitů původní zašifrované zprávy. Další důležitou vlastností je **jednosměrnost** funkce, což znamená, že případný útočník není schopen z výsledné hodnoty heše získat původní vzor. S ohledem na libovolnou délku vstupních bitů není počet vzorů ničím limitován. Naopak s předem danou délkou heše jsou omezené i počty vzorů, tudíž existují vzory se stejnou hodnotou heše. Hešovací funkce zaručuje tzv. **bezkoliznost**. To znamená, že nelze nalézt dva různé vzory [49, 50].

Rozlišujeme několik druhů hešovacích algoritmů. Dělí se dle délky výstupního heše. Například pro variantu SHA–256 (Secure Hash Algorithm) má výsledný heš délku 256 bitů.

Odolnost hešovací funkce O odolnosti hešovací funkce hovoříme, pokud je velmi obtížné případně současnými prostředky nemožné, nalézt zprávy, které odpovídají svému vzoru nebo dvě zprávy se stejným vzorem. Kryptograficky bezpečné hešovací funkce se nazývají takové funkce, které jsou odolné vůči kryptoanalýze, tzn. bez znalosti klíče nelze zjistit původní zprávu. V kryptografii se za bezpečná v současné době považují hešovací funkce SHA–256 až SHA–512. Právě varianta SHA–256 je využita v generátoru náhodných čísel v této práci.

Hashlib Knihovna využívající bezpečnostních algoritmů k hešování a zpracování zpráv. Jsou zde definované hešovací algoritmy jako SHA–1, SHA–224, SHA–256, SHA–384, SHA–512 a RSA MD5 [51]. Knihovna je v této práci využívána ke změně stejnorodých dat, kdy nám na vstup přijdou stejné hodnoty a potřebujeme zajistit, aby nikdy nebyla tato data stejná.

5.3 JavaScript

Je skriptovací jazyk využívající se na webových stránkách. Zapisuje se přímo do HTML kódu a spouští se v prohlížeči na straně klientského počítače. Mezi hlavní charakteristiky JavaScriptu patří [52]:

- Objektově orientovaný - využívá objektů prohlížeče,
- Case sensitive - netoleruje záměnu velkých a malých písmen,
- Interpretovaný - není nutné ho kompilovat,
- Závislý na prohlížeči - v různých verzích prohlížeče nemusí správně fungovat.

Ajax - Asynchronous JavaScript and XML Je označení pro asynchronní odesílání nebo získávání požadavků pro zobrazení informací bez nutnosti načítání kompletní HTML stránky. Na základě požadavku o získání/odeslání informace se aktualizuje pouze obsah, který se změní namísto celé stránky. Odpadá tak nepříjemný efekt překreslení celé stránky. Při správné implementaci dokáží snížit zatížení na webové stránky, případně na celou síť. Tím lze docílit například menšího zatížení databázových serverů, protože nedochází k výměně takového množství dat [53].

Asynchronní programování Technika používaná pro spouštění dlouho trvajících procesů při zachování responsivity aplikace. Takto spuštěné asynchronní procesy neběží na hlavním vlákně aplikace. Hlavní vlákno tedy může obsluhovat uživatelské rozhraní. Všechny tyto procesy běží na vláknech spuštěných na pozadí.

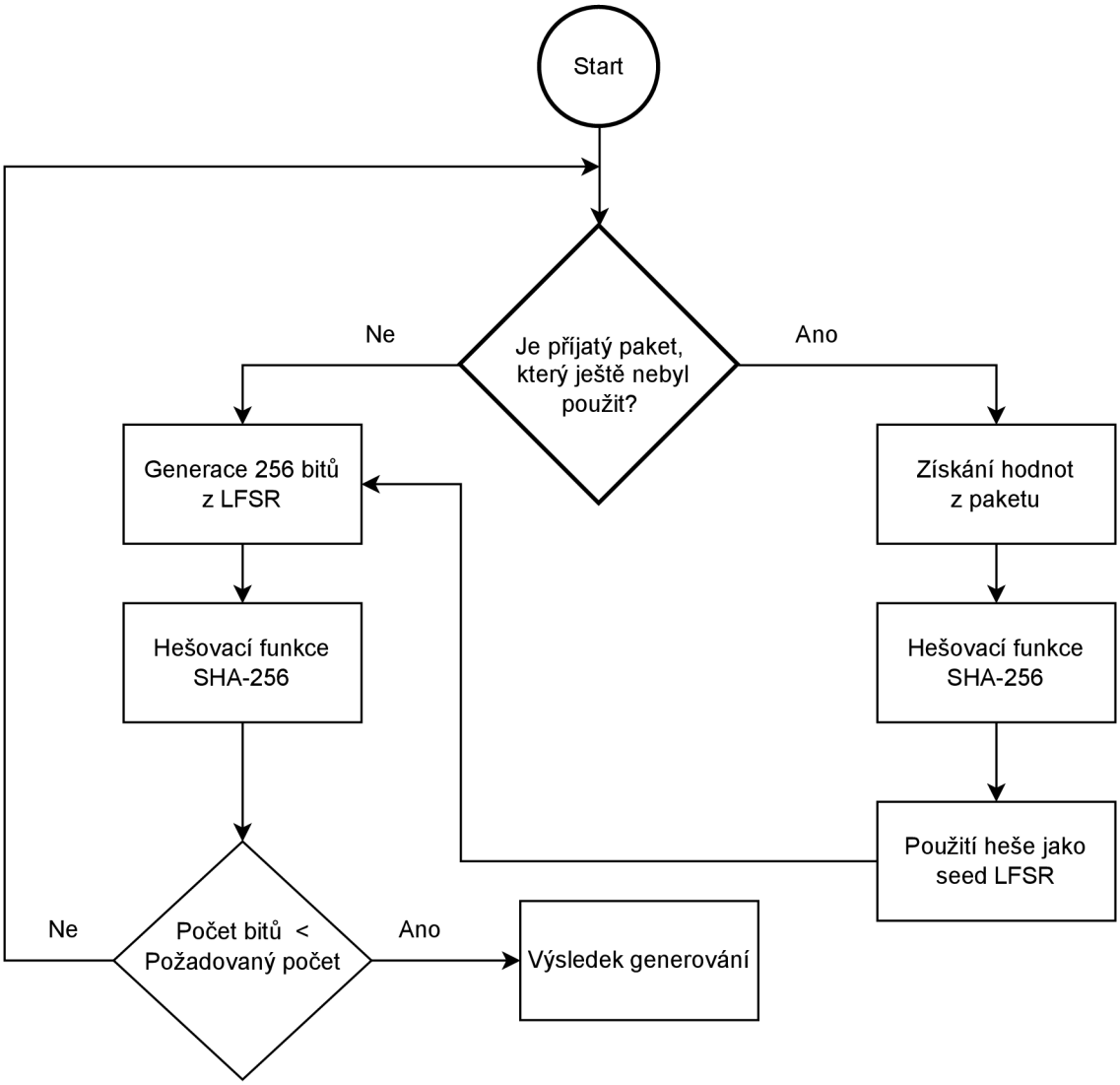
Synchronizace takto spuštěných procesů může probíhat pomocí dvou technik. První z nich je zpětné volání (callback). Ty jsou volány při důležitých událostech dlouho běžícího procesu. Druhou technikou je použití speciální syntaxe `Async/Await`. Ta umožňuje spuštění procesu libovolné části aplikace a čekání na výsledek pomocí klíčového slova `Await` [54].

Asynchronní programování v jazyce JavaScript JavaScript umožňuje programování pomocí obou těchto technik. Novější rozhraní používání techniku techniku Async/Await, zatímco starší knihovny používají stále zpětné volání. Jedním z příkladů funkcionality využívající tento novější přístup je „Fetch API“ [54].

Fetch API Jedná se o programové rozhraní pro provádění dotazů v REST API. Umožňuje specifikaci typu metody, nastavení vlastních hlaviček nebo také nastavení těla zprávy. Při zavolání metody Fetch s tímto nastavením je možné pomocí klíčového slova `Await` čekat na odpověď. V této odpovědi je následně možné získat navrácená data a případně je deserializovat z formátu `JSON` [55].

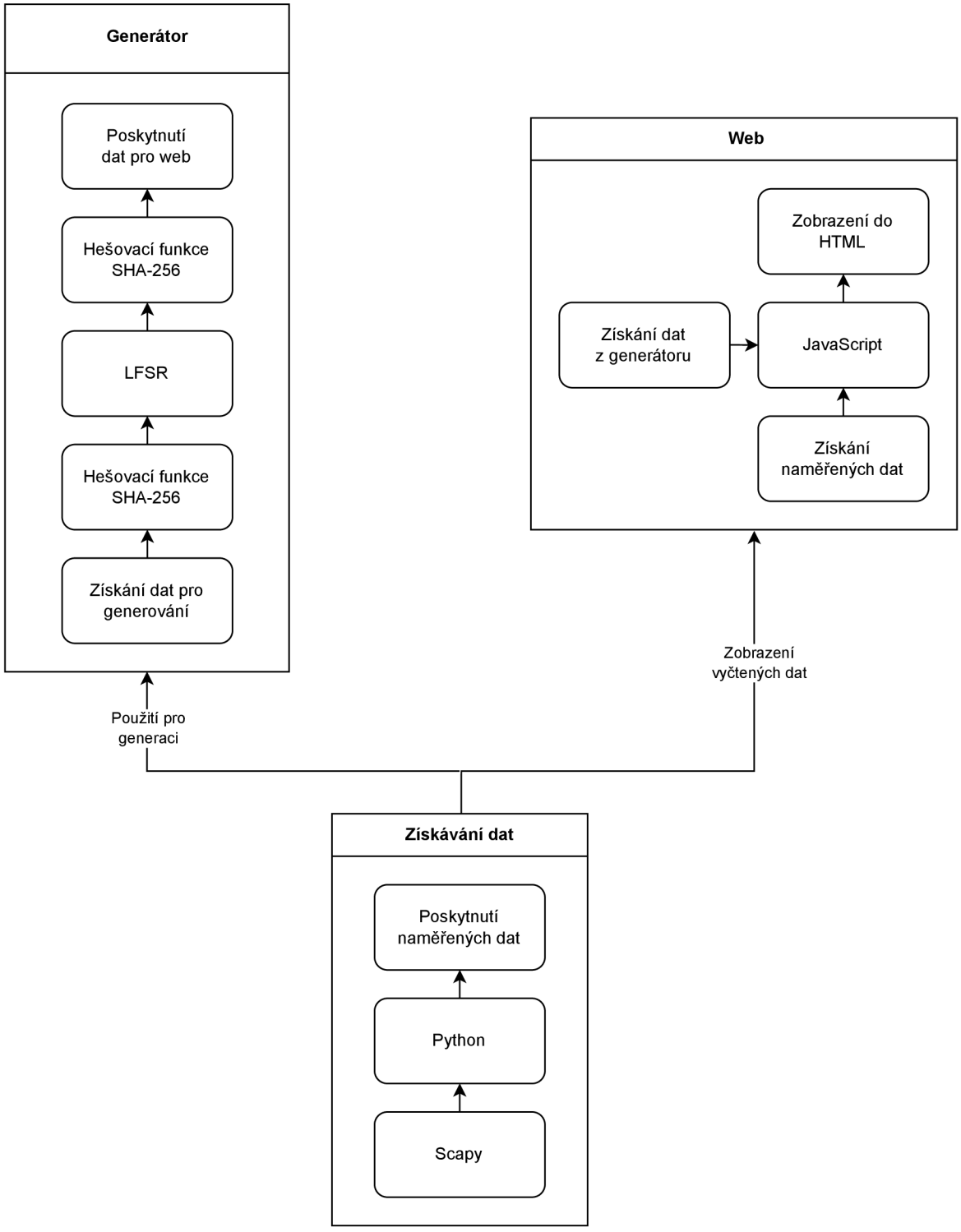
Bootstrap Je soubor komponent pro tvorbu webových stránek. Obsahuje sadu kaskádových stylů (CSS), které umožňují modifikovat vzhled HTML elementů. Jednotlivé komponenty jsou identifikovány pomocí názvu třídy. Tuto třídu stačí přidat do HTML elementu pro aplikování odpovídajícího stylu. Tato knihovna umožňuje vytvářet vzhledné webové stránky s minimálním úsilím a znalostí CSS. Tato knihovna byla využita pro tvorbu klientské webové aplikace [56].

5.4 Schéma generátoru



Obr. 5.1: Blokové schéma navrženého generátoru.

5.5 Diagram zobrazení na webové stránce



Obr. 5.2: Vývojový diagram zobrazení na webové stránce.

6 Realizace navrženého generátoru

Generátor je rozdělen na zachytávací část, která je označena jako Sniffer a samotnou aplikaci určenou k definici vstupních podmínek s možností zobrazení výsledků. Tyto definice a ovládání aplikace probíhá na vytvořené webové stránce.

6.1 Vyčítání hodnot z paketů

Vyčítání hodnot z paketů zajišťuje soubor s názvem `Sniffer.py`. Algoritmus použitý v tomto souboru je vidět ve výpisu 6.1. Ten má za úkol přepnout kartu z promiskuitního do monitorovacího režimu, pokud se v něm nenachází. Dále zachytává pakety s označením Dot11 a počítá čas od posledního přijatého paketu. Poslední zachycený paket je uchováván pro účely generace.

Výpis 6.1: Zachytávání provozu a výpočet času od posledního paketu

```
class Sniffer(Thread):
    def __init__(self, interface="wlan0"):
        super().__init__()
        self.packet_delay = 0
        self.last_packet = None
        self.interface = interface

    def run(self):
        os.system(f"airmon-ng check kill")
        os.system(f"airmon-ng start {self.interface}")
        sniff(iface=self.interface, prn=self.callBack)

    def setChannel(self, channel: int):
        os.system(f"iw dev wlan0 set channel {channel}")
        print(f"Changing channel to {channel}")

    def callBack(self, pkg):
        if pkg.haslayer(Dot11):
            if pkg.type == 0 and pkg.subtype == 8:
                if self.last_packet is not None:
                    self.packet_delay =
                    pkg.time - self.last_packet.time
                self.last_packet = pkg
```

6.2 Implementace generátoru

V souboru `app.py` je uvedena hlavní logika generátoru a také jeho rozhraní, které využívá webová aplikace. Při spuštění aplikace je nejprve vytvořen polynom, který je využíván v LFSR. Dále je také vytvořen výchozí stav, pomocí kterého je LFSR inicializováno. Tento stav se skládá z náhodně vygenerovaných bitů poskytnutých výchozím generátorem náhodných čísel v jazyce Python. Následně je inicializován a spuštěn sniffer, který začne zachytávat pakety.

Při požadavku o vygenerování náhodného čísla je nejprve kontrolováno, zda sniffer zachytil nějaký paket. Pokud byl paket zachycen, ale ještě nebyl použit, budou z tohoto paketu vyčteny hodnoty síly signálu a čas od posledního přijatého paketu. Jejich kombinace následně tvoří nový seed pro LFSR.

Vytvoření seedu pro LFSR Ze služby sniffer je získána hodnota mezičasu mezi přijatými pakety. Z této hodnoty je získán řád nanosekund, které i ve velice zarušeném prostředí poskytuje dostatečnou variabilitu. Tento čas je následně násoben s hodnotou anténního signálu z posledního zachyceného paketu. Tato položka je v jednotkách decibel na miliwatt. Výsledek násobení je převeden na řetězec a použit jako vstup do hešovací metody SHA–256. Výsledek hešování je nově vytvořený seed pro LFSR. Kód je možné vidět ve výpisu 6.2.

Výpis 6.2: Výpis kódu pro vytvoření seedu LFSR

```
1  def getLfsrSeed(s: Sniffer):
2      hash = hashlib.sha256()
3      randomDelay =
4          s.packet_delay * 100000000000000 % 1000000000
5      hash.update(bytearray(str(randomDelay *
6          s.last_packet.dBm_AntSignal), 'utf-8'))
7      bits = []
8      digest = hash.digest()
9      for i in range(32):
10         byte = digest[i]
11         for y in range(8):
12             bits.append((byte & (1 << y)) >> y)
13     return bits
```

Postup generování Jak už bylo dříve zmíněno, je při požadavku na generování snaha použít nový seed pro LFSR. Ne vždy je to možné, aby bylo možné pokračovat v generování i pokud nepřichází nové pakety, je využita generační funkce LFSR.

Je vždy vygenerováno 256 bitů, které jsou následně použity jako vstup do hešovací funkce SHA–256. Ta generuje již výsledné náhodné hodnoty. Jelikož samotné LFSR neposkytuje dostatečnou náhodnost pro použití v kryptografii, byla využito vlastností hešovací funkce. Tímto by měly být náhodně generované čísla odolné vůči kryptoanalýze. Pokud je požadováno více než 256 bitů, je provedena nová generace. Takto vygenerované bity jsou následně řazeny do sekvence z níž je vybrán pouze požadovaný počet bitů.

6.2.1 Komunikační rozhraní generátoru

Komunikační rozhraní generátoru je realizováno jako REST API. Toto API se skládá z několika přístupových bodů. Jejich popis je uveden níže.

index Index je hlavním vstupním bodem webové aplikace. Stará se o zobrazení šablony vzhledu hlavní stránky. Jedná se o hlavní rozcestník, který následně pomocí uživatelského rozhraní umožňuje volat ostatní koncové body. Kód je možné vidět ve výpisu 6.3.

Výpis 6.3: Výpis kódu pro zobrazení HTML stránky

```
@app.route("/")
def hello():
    return render_template("index.html")
```

getBitmap Pomocí této metody, která je uvedena ve výpisu 6.4, je možné získat rastrový obrázek, který obsahuje pixely s barvou. Ta je udávána náhodnou posloupností vygenerovanou pomocí generátoru. Velikost takto vygenerovaného obrázku je 128 x 128 pixelů. Obrázek je následně navrácen jako PNG soubor. Podrobnější popis způsobu generování bitmapy je uveden v kapitole 7.5.

Výpis 6.4: Výpis kódu pro vygerování bitmapy

```
@app.route("/getBitmap")
def getBitmap():
    pixels = []
    number = int.from_bytes(getRandom(s, 128 * 128),
        byteorder="little")
    for i in range(128 * 128):
        pixels.append
            (np.uint8(((number & 1 << i) >> i) * 255))
    pixels = np.array(pixels).reshape((128, 128))
    pi = Image.fromarray(pixels)
```

```
11      pi.save("bitmap.png")
12      return send_file("bitmap.png", mimetype='image/png')
```

getRandom Tato část rozhraní umožňuje získávání náhodného čísla o konkrétní bitové délce. Parametrem je právě tato délka jako celé číslo. Požadavek je nutné volat pomocí metody POST. Výsledkem je celé číslo o dané délce v podobě řetězce. Odpověď není nijak formátovaná. Výpis kódu je uveden v 6.5.

Výpis 6.5: Výpis kódu pro získání náhodného čísla

```
1 @app.route("/getRandom", methods=["POST"])
2 def randomNumberRequest():
3     random = getRandom(s, request.json["numberOfBits"])
4     return str(int.from_bytes(random, byteorder="little"))
```

downloadRandomNumbers Tato metoda je zodpovědná za vygenerování určitého počtu čísel o předem dané velikosti do souboru. Tento textový soubor je následně umožněno uživateli stáhnout. Vstupními parametry jsou: počet vygenerovaných čísel a jejich velikost v bitech. Výpis kódu je uveden v 6.6.

Výpis 6.6: Výpis kódu pro stažení souboru s náhodnými čísly

```
1 @app.route("/downloadRandomNumbers", methods=["GET"])
2 def downloadRandomNumbers():
3     numberSize = int(request.args["numberSize"])
4     numberCount = int(request.args["numberCount"])
5     return Response(numberGenerator(numberCount,
6         numberSize), mimetype="text/plain",
7                 headers={"Content-Disposition":
8                     "attachment;filename=numbers.txt"})
```

runSts Úkolem této metody je spouštění souboru testů Národního institutu standardu a technologií. Tato metoda vytváří binární soubor, který obsahuje náhodnou posloupnost bitů. Velikost této sekvence je dána vstupním parametrem. Před spuštěním testů je vygenerován soubor s parametry testovací sady, který je následně přesměrován na standardní vstup tohoto programu. Po dokončení testování jsou výsledky testů překopírovány do složky `results`. Tato složka je před každým testováním vyčištěna. Výsledkem volání této metody je zpráva ve formátu `JSON` obsahující textový výstup jednotlivých testů a také standardní a chybový výstup programu.

Tento koncový bod je možné volat pouze metodou POST. Parametry této metody jsou předávány v adrese URL. Kód je možné vidět ve výpisu 6.7.

Výpis 6.7: Výpis kódu pro otestování čísel

```
def runSts(bitCount):
    stsFileName = "sts-input.bin"
    generateRandomBinaryFile(stsFileName, bitCount)

    cmd = ["./assess", str(math.floor((bitCount / 10)))]
    createStsInputFile(stsFileName)
    popen = subprocess.Popen
        (cmd, stdin=open("sts.input", "r"),
        stdout=subprocess.PIPE, stderr=subprocess.PIPE,
        universal_newlines=True, cwd="sts-2.1.2")
    out, err = popen.communicate()
    return out, err
```

Spuštění aplikace Po stažení a rozbalení přílohy je nutné otevřít terminál ve složce s aplikací. Následně provést import LSFR logiky z adresáře github z URL: `https://github.com/mkazmier/linear-register/blob/master/lfsr.py` Pro spuštění webové aplikace a fungování generátoru je nutné mít nainstalované tyto závislosti pomocí příkazů:

```
pip3 install numpy
pip3 install scapy
pip3 install flask
pip3 install imageio
```

Dále je nutné mít staženou sadu testů NIST STS a vloženou do kořenového adresáře s aplikací. Tuto sadu lze stáhnout z adresy: `https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software`. Spuštění webové stránky je následně provedeno za pomoci příkazu:

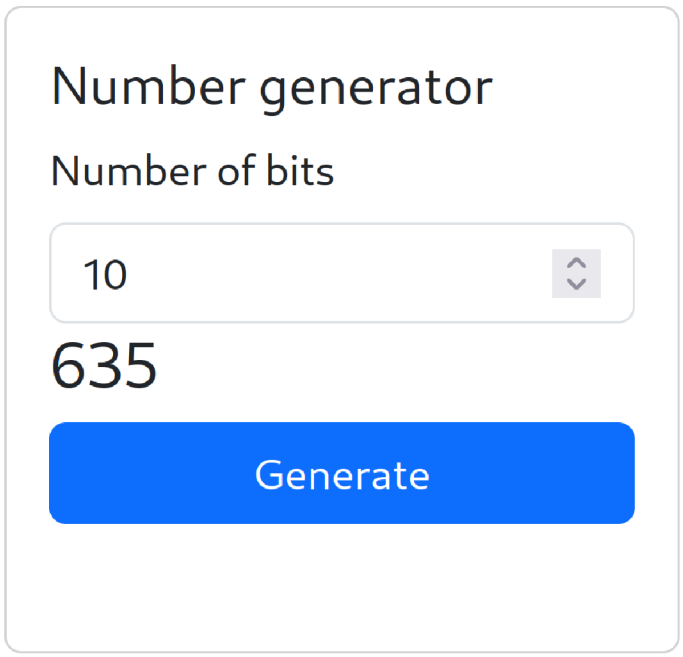
```
sudo flask run
```

Následně na adrese `http://127.0.0.1:5000/` je možné nalézt stránku jako je na obrázku 6.6.

6.2.2 Grafický vzhled aplikace

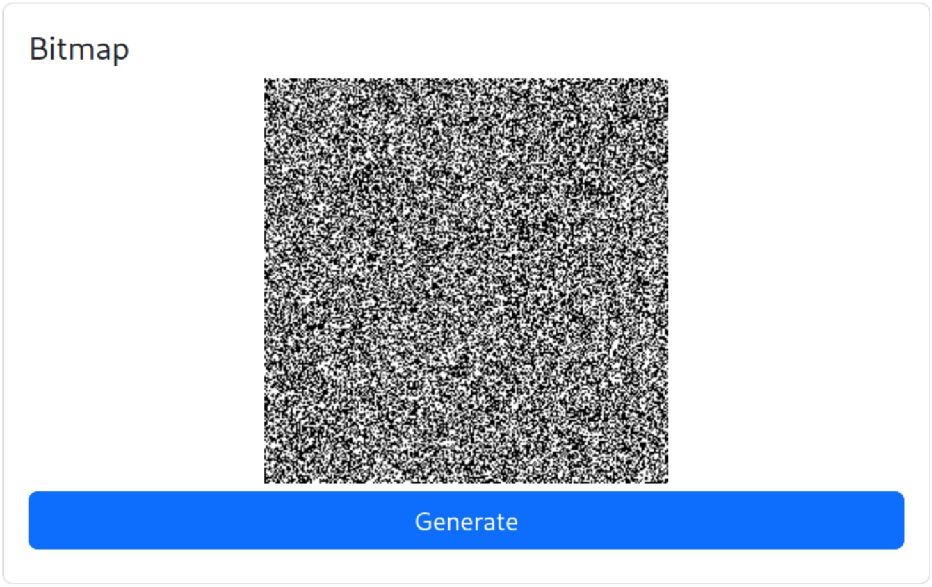
Samotné grafické rozhraní aplikace bylo tvořeno za pomoci značkovacího jazyka HTML a kaskádových stylů CSS. Byl použit základní styl frameworku Bootstrap. Samotné grafické rozhraní je rozděleno do sekcí. Každá z těchto sekcí poskytuje specifickou funkcionalitu v uživatelsky přehledném stylu. Tyto sekce jsou popsány níže.

Generátor náhodných čísel Tato sekce umožňuje generování náhodného čísla o předem zvolené velikosti. Tuto velikost je možné si zvolit v textovém poli. Následně pomocí tlačítka je uživateli umožněno vygenerovat náhodné číslo o této velikosti. Stisk tlačítka volá koncový bod webového rozhraní generátoru nazvaný `getRandom`. Výsledek tohoto dotazu je zobrazen mezi textové pole a tlačítko. Velikost čísla není omezena, ale generování obrovských velikostí může zabrat dlouhou dobu. Uživatelské rozhraní této sekce je možné vidět na obrázku 6.1.



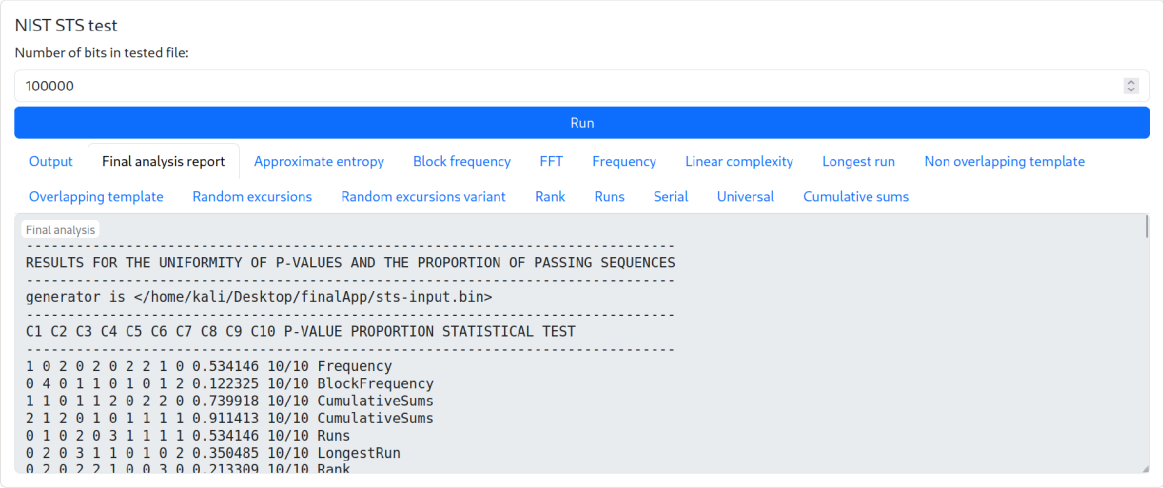
Obr. 6.1: Grafické rozhraní pro generování náhodných čísel.

Sekce bitmapy Zde je dostupná funkcionalita generování náhodné bitmapy. Pomocí tlačítka je volána metoda v generátoru nazvaná `getBitmap`. Výsledek této metody je obrázek, který je zde zobrazen. Opětovný stisk vygeneruje a zobrazí novou bitmapu. Toto uživatelské rozhraní je možné vidět na obrázku 6.2.



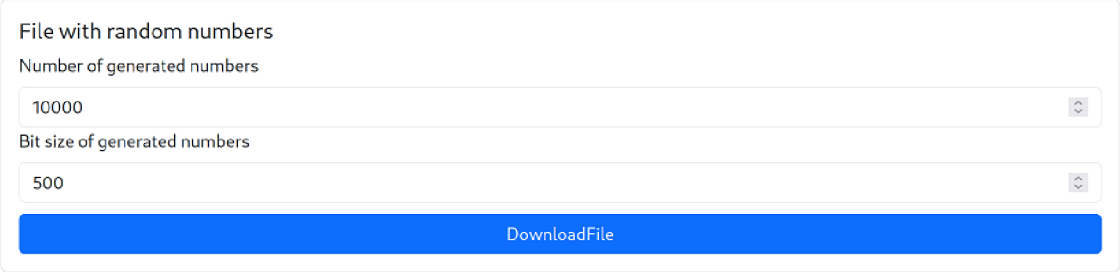
Obr. 6.2: Zobrazení bitmapy na webové stránce.

Spouštění NIST testů Tato část uživatelského rozhraní umožňuje spustit sadu testů od institutu NIST. Je zde dostupné nastavení velikosti bitové sekvence, která bude zapsána do souboru a použita pro testování. Po stisku tlačítka „Run“ je zavolána metoda `runSts`. Ta spustí testování. Také je uživateli zobrazena ikona načítaní. Po dokončení jsou zobrazeny výsledky jednotlivých testů v odpovídajících záložkách. Výsledky jsou zobrazeny v textové podobě. Formátování odpovídá výstupnímu textu z testovací sady. Rozhraní části pro spouštění testů je možné vidět na obrázku 6.3.



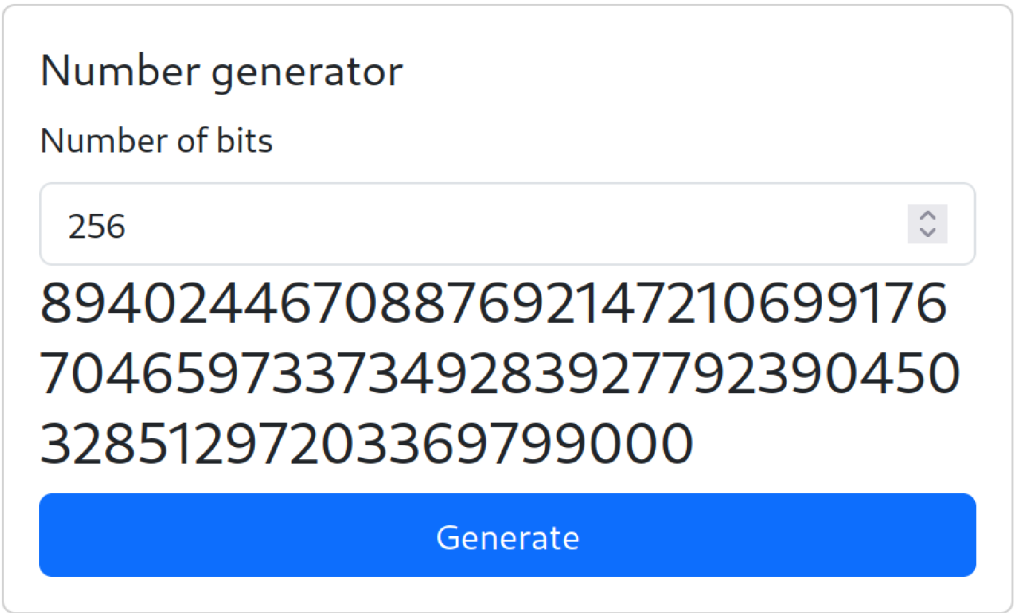
Obr. 6.3: Zobrazení NIST testů na webové stránce.

Generace souboru s čísly Tato sekce umožňuje generování souboru, jehož obsahem budou náhodně generovaná čísla v textové podobě. Je zde možné si v odpovídajících textových polích nastavit počet čísel, která budou generována a také jejich velikost. Po stisku tlačítka „Download File“ je započteno stahování a souběžná generace souboru. Generování je prováděno voláním metody `downloadRandomNumbers` z rozhraní generátoru. Vzhled tohoto rozhraní je možné vidět na obrázku 6.4.

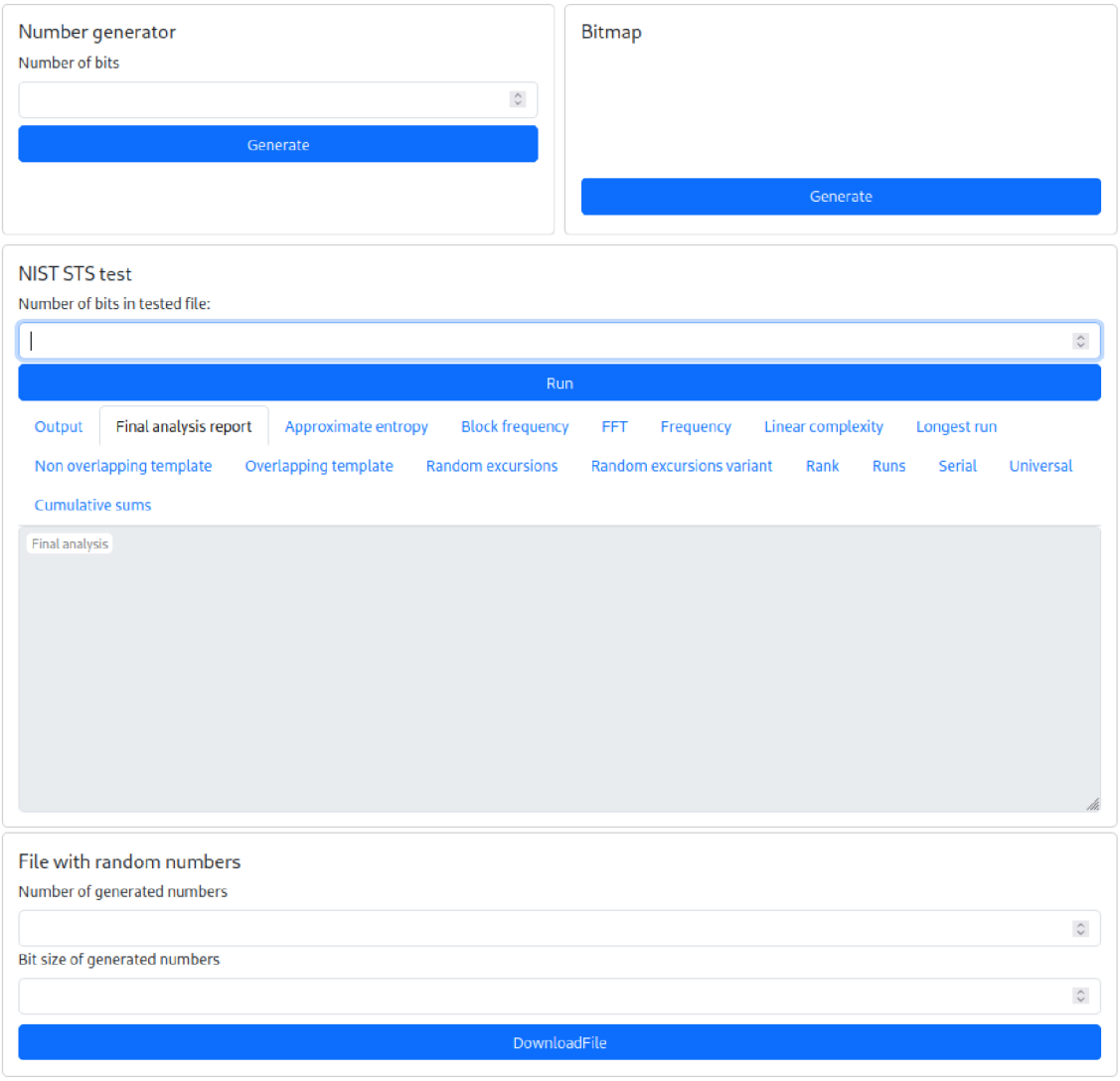


Obr. 6.4: Sekce generace souboru s náhodnými čísly.

Souhrnný přehled vzhledu webového rozhraní je možné vidět na obrázku 6.6. Na obrázku 6.5 je zobrazena ukázka 256 bitového čísla, které lze použít např. v symetrické šifře AES.



Obr. 6.5: Velikost generovaného čísla o délce 256 bitů.



Obr. 6.6: Vzhled webové stránky.

7 Testování vygenerovaných posloupností

Pro testování vygenerované posloupnosti byla zvolena testovací sada NIST STS (NIST Statistical Test Suite), kterou je možné stáhnout přímo ze stránek Národního institutu standardů a technologie na adrese: `https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software`. Tato sada disponuje množstvím testů, které byly podrobně probrány v kapitole 2.2.

Testování probíhalo v operačním systému Kali Linux. Protože je sada psaná v jazyce `C`, jako první po stažení a rozbalení bylo nutné provést kompilaci za pomocí příkazu `make` v rozbalené složce. Tímto program zkompilujeme a vytvoří nám soubor `assess` přes který lze program spustit.

Spuštění souboru se provádí zadáním příkazu do příkazové řádky operačního systému `assess <požadovaný počet bitů>`. Tato hodnota udává délku sekvence s označením n.

Program poté navede celým procesem, provede se výběr typu zdroje, kde zvolíme, že chceme testovat ze vstupního souboru. Poté vložíme cestu k souboru. Po zadání cesty vybereme, že chceme provést všechny testy a ponecháme parametry v základním nastavení. Dále se nás program zeptá na velikost datového toku. Tato hodnota udává velikost bloku M. Dále vybere zda se jedná o soubor s ASCII hodnotami nebo o binární soubor.

Výsledný soubor z testování s názvem `finalAnalysisReport.txt` je poté uložen ve složce `experiments\AlgorithmTesting`. Tento soubor obsahuje souhrn ze všech provedených testů. Detailní informace o konkrétních vypočtených proměnných k jednotlivým testům lze nalézt v jednotlivých složkách v souboru `stats.txt`.

V tabulce 7.1 lze vidět výsledky generované za pomoci generátoru navrženého v kapitole 5. Hodnoty v tabulce jsou uvedeny v procentech, označují procentuální úspěšnost daných testů. Pokud byly některé testy prováděny vícekrát, je zde uvedena průměrná hodnota. Byly testovány sekvence o velikostech 10 000, 100 000 a 1 000 000. Doporučená minimální velikost sekvence je 1 000 000, aby výsledky nebyly zkresleny. Test náhodných výběrů a Variantní test náhodných výběrů nejsou podporovány pro sekvence menší než 1 Mb, proto ani u sekvencí 10 000 a 100 000 neproběhnou. Minimální úspěšnost testů je stanovena na 80 %. U testů náhodných výběrů a Variantního testu náhodných výběrů je to potom 50 % pro sekvence o velikosti 1 Mb nebo větší.

Pro sekvence bitů 10 000 a 100 000 neprošel Maurerův univerzální test, který počítá počet bitů mezi shodnými vzory, podle kterých jsme schopni poznat schopnost komprimace. Pokud sekvenci lze významně zkomprimovat, potom je sekvence považována za komprimovanou. Pro sekvenci 1 000 000 bitů již test prošel, což lze předpokládat, že pro menší sekvence nebyl dostatek dat potřebných pro přesnější vyhodnocení testu. Pro sekvenci 10 000 také vyšel nízko Test přibližné entropie,

který zkoumá četnost všech překrývajících se m-bitových vzorů v posloupnosti. To lze přisuzovat kvůli nedostatečnému množství dat.

Tab. 7.1: Tabulka výsledků testů NIST z generátoru navrženého v této práci.

Typ testu	**Velikost sekvence bitů**		
Typ testu	**10 000**	**100 000**	**1 000 000**
1. Frekvenční test	100%	100%	100%
2. Frekvenční test bloku	100%	100%	100%
3. Test běhů	100%	100%	100%
4. Test nejdelšího běhu v bloku	100%	100%	100%
5. Test hodnot binární matice	100%	100%	100%
6. Spektrální test	100%	100%	90%
7. Test shody nepřekrývajících se šablon	90%	100%	90%
8. Test shody překrývajících se šablon	100%	90%	100%
9. Maurerův univerzální test	0%	0%	100%
10. Test lineární složitosti	100%	100%	100%
11. Sériový test	100%	100%	100%
12. Test přibližné entropie	70%	100%	100%
13. Test kumulativních součtů	100%	100%	100%
14. Test náhodných výběrů	–	–	100%
15. Variantní test náhodných výběrů	–	–	90%

7.1 Testování sadou ENT

Další možností jak otestovat vygenerovanou sekvenci představuje ENT Test Suite dostupnou z: `https://www.fourmilab.ch/random/`. Tato sada vypisuje výsledek testu přímo do konzole. Lze ji stáhnout v zazipovaném souboru a stejně jako NIST STS, je nutné provést kompilaci příkazem `make`. Poté již stačí test spustit pomocí příkazu `ent -c <název souboru>`. Výsledek testu na stejnou sekvenci bitů jako v kapitole 7 lze vidět v tabulce 7.2.

Prvním testem je test Entropie, který udává počet bitů informace obsažené v jednom bajtu dat. Pro skutečně náhodně vygenerované bity by měla být tato hodnota vyšší než $7,9$ bitů na bajt, což tento test potvrdil. Dalším testem je test možnosti komprese. Jak již název napovídá, udává procentuální hodnotu o kolik je možné danou posloupnost zmenšit. Tato hodnota by se měla blížit nule, což tento test potvrdil a tak tedy sekvenci nelze komprimovat bez ztráty informace. Chí–kvadrát

Tab. 7.2: Tabulka výsledků testů ENT z generátoru využívající bezdrátové sítě.

Test	RNG bezdrátových sítí
Entropie	7,999567
Optimální komprese	0%
Chí–kvadrát test	599,65 (0,01%)
Aritmetický průměr	127,7306
Monte Carlo koeficient	3,131796527 (chyba 0,31%)
Koeficient sériové korelace	0,000273

test dobré schody rozhoduje, zda jsou testovaná data rovnoměrně rozdělena. Zde vykazují hodnoty velkou nepřesnost, resp. zde výsledek hodnot vychází na změnu testovacích dat, pokud by však hodnota vyšla stejně i u jiného souboru dat, jednalo by se o nenáhodné sekvence. Výsledek aritmetického průměru by se měl pohybovat kolem hodnoty $127,5$ což nám hodnota $127,7306$ dává odchylku o $0,18\,\%$. Monte Carlo koeficient udává počet procentuální rozdíl od reálné hodnoty π. Rozdíl je pouze $0,07\,\%$. Koeficient sériové korelace udává, jak moc je každý bajt v souboru závislý na předchozím bajtu. Hodnota $0,001430$ odpovídá náhodným hodnotám. Pokud by byly hodnoty nenáhodné, blížila by se tato hodnota $0,5$, pro silně předvídatelné hodnoty se blíží k 1.

7.2 Časová náročnost generování jednotlivých sekvencí

Testování se provádělo na stolním počítači s procesorem AMD Ryzen 5 2600x, který disponuje 6 jádry. Bohužel defaultně Python pracuje pouze na jednom jádře. Prvotní návrh generátoru byl koncipován na fungování na zařízení Raspberry Pi. Implementace na jiný programovací jazyk nebo přepsání kódu pro podporu více jader z důvodu časové tísni není možné.

V tabulce 7.4 můžeme vidět jak dlouho se která sekvence generovala a čas potřebný na jeden vygenerovaný bit. Čas generovaný celé sekvence je poté vynesen do grafu na obr.7.1. Můžeme vidět, že čas generování sekvencí je téměř lineární. Z toho lze poté také vypočítat potřebný čas pro vytvoření souboru určité velikosti.

Statistické testy NIST byly zvoleny z důvodu jejich uzpůsobení pro testování menšího vzorku dat. Například testy Dieharder vyžadují několikanásobně větší soubor dat. Některé zdroje uvádějí potřebnou hodnotu dat na 250 GB, aby byly testy

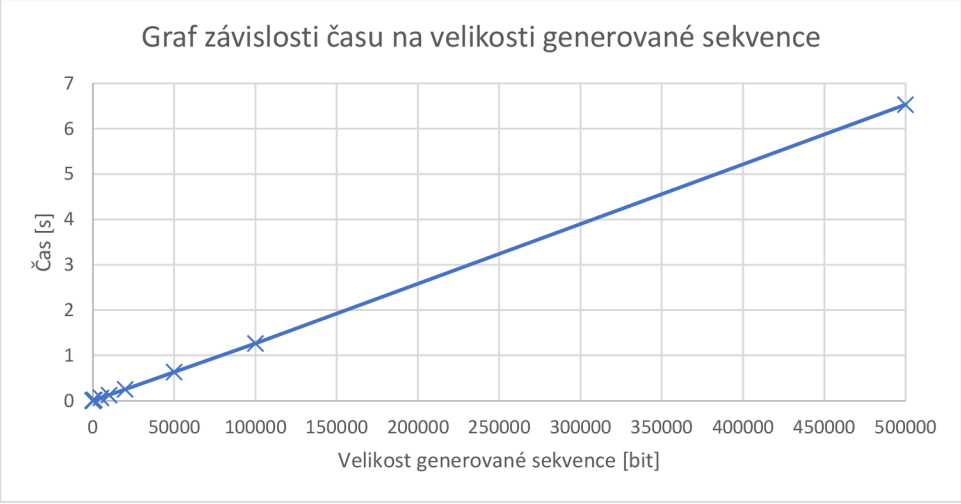
spolehlivé. V tabulce 7.3 lze vidět potřebný čas na vygenerování souboru o velikosti 250 GB tímto generátorem. Generátor by musel běžet celkem 325 dní aby byl schopen vygenerovat takové množství posloupností. To v této práci není možné realizovat.

Testům vytvořeným NIST stačí pro otestování podstatně menší soubor dat. Některým testům stačí data o velikosti několik kilobitů, jiné potřebují soubor o 1 Mb. Nutno podotknout, že i když generátory testy projdou, stále nezaručují, že se jedná o opravdu náhodné hodnoty.

Tab. 7.3: Tabulka časové náročnosti generování.

Potřebný soubor pro Dieharder test	250 **GB**
převedeno na bity	2 147 483 648 000 b
při rychlosti	76 522 b/s
potřebný čas	28 063 612 sekund
	467 727 minut
	7 795 hodin
	325 **dní**

Obr. 7.1: Graf časové závislosti na počtu bitů v sekvenci.



Tab. 7.4: Časová náročnost generování jednotlivých sekvencí.

Délka sekvence bitů [bit]	**Čas generování celé sekvence [ms]**	**Průměrný čas na bit [μs]**
1	7, 16	7155, 42
5	8, 12	1623, 87
10	7, 66	765, 92
50	7, 57	151, 30
100	7, 16	71, 62
200	7, 71	38, 54
500	12, 91	25, 82
1 000	22, 48	22, 48
5 000	65, 65	13, 13
10 000	129, 39	12, 94
20 000	255, 27	12, 76
50 000	632, 36	12, 65
100 000	1 265, 65	12, 66
500 000	6 534, 06	13, 07

7.3 Porovnání výsledků s generátorem využívající atmosférický šum

V tabulce 7.6 lze vidět porovnání hodnot ENT testů z generátoru využívající bezdrátové sítě s hodnotami vygenerovanými generátorem, který využívá jako zdroj dat atmosférický šum. Jsou použita data ze stránky: `https://archive.random.org/binary`. Zde jsou jednou denně uloženy náhodně vygenerované posloupnosti čísel. Archiv sahá až do roku 2006 a obsahuje předvygenerované soubory pravých náhodných čísel. Pro otestování byl vybrán soubor ze dne 1. 4. 2023 v binárním souboru.

V tabulce 7.5 lze vidět porovnání výsledků STS testů s generátorem využívající bezdrátové sítě a generátorem využívající atmosférický šum.

Tab. 7.5: Porovnání testů STS na posloupností z generátoru využívající bezdrátové sítě s generátorem využívající atmosférický šum.

Typ testu	RNG využívající bezdrátové sítě	RNG využívající atmosférický šum
1. Frekvenční test	100 %	100 %
2. Frekvenční test bloku	100 %	100 %
3. Test běhů	100 %	100 %
4. Test nejdelšího běhu v bloku	100 %	100 %
5. Test hodnot binární matice	100 %	100 %
6. Spektrální test	90 %	100 %
7. Test shody nepřekrývajících se šablon	90 %	90 %
8. Test shody překrývajících se šablon	100 %	100 %
9. Maurerův univerzální test	100 %	100 %
10. Test lineární složitosti	100 %	100 %
11. Sériový test	100 %	100 %
12. Test přibližné entropie	100 %	100 %
13. Test kumulativních součtů	100 %	100 %
14. Test náhodných výběrů	100 %	100 %
15. Variantní test náhodných výběrů	90 %	100 %

Tab. 7.6: Porovnání testů ENT na posloupností z generátoru využívající bezdrátové sítě s generátorem využívající atmosférický šum.

Test	RNG bezdrátových sítí	Random.org
Entropie	$7,999875$	$7,999834$
Optimální komprese	$0\,\%$	$0\,\%$
Chí–kvadrát test	2487,82 $(0,01\,\%)$	241,6 $(71,71\,\%)$
Aritmetický průměr	$127,6059$	$127,6096$
Monte Carlo koeficient	$3,139431792$ (chyba $0,07\,\%$)	$3,142880031$ (chyba $0,04\,\%$)
Koeficient sériové korelace	$0,001430$	$-0,001339$

7.4 Problémy při testování posloupností

V rámci této práce byla i snaha vytvořit a otestovat soubor o velikosti $2,5\,\mathrm{GB}$, který se podařilo vygenerovat. Celý proces trval přibližně 27 hodin. Bohužel se při následném zpracování ukázalo, že je soubor s daty poškozen a nelze jej tedy zpracovat a otestovat Dieharder testy. Menší soubory generované pro NIST STS, program Dieharder není schopen zpracovat. To je dáno charakteristickou vlastností testu, kdy se testuje obrovské kvantum dat právě proto, aby testy dokázaly odhalit poměrně těžce odhalitelné slabiny generátoru. U takto velkého množství dat, i na první pohled dobrý generátor, nakonec ukáže svou slabinu.

7.5 Hledání vzorů v bitmapě

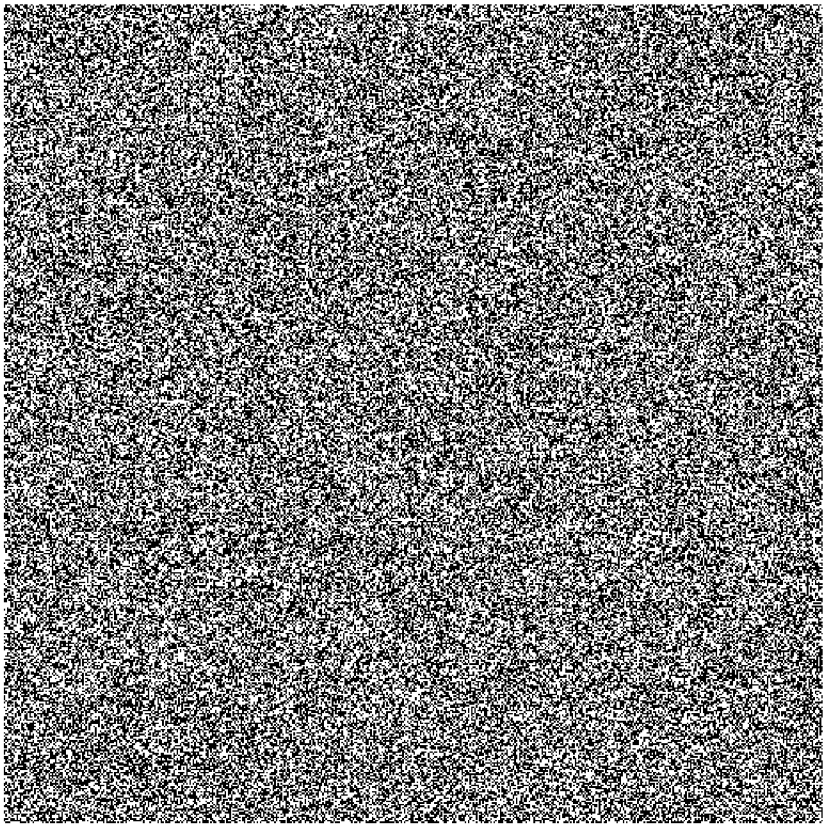
Pro lidský mozek je přirozené hledat ve věcech souvislosti. Proto je dobrý v hledání vzorů a nejjednodušší metodou detekování vzorů ve vygenerovaných posloupností je jejich zanesení do bitmapy [57].

Generování bitmapy je postaveno na principu generování posloupností náhodných čísel v podobě jednotlivých bitů. Tyto bity jsou poté zaneseny do pole o definované velikosti. Každý bit poté nese informaci o logické 0 a logické 1. Pokud generujeme pole o velikosti 64×64 bitů, je počet vygenerovaných bitů roven 4 096. Program lze upravit na generování větších souborů bitmapy. Příklad vygenerované sekvence vyobrazené v bitmapě z tohoto generátoru o velikost 512×512 bitů je zobrazen na obrázku 7.2.

Pro generování takto velké plochy je potřeba vygenerovat posloupnost o celkové délce 262 144 bitů. S ohledem na nízkou výpočetní rychlost generátoru je vygenerování takovéto posloupnosti časově náročnější a znamenalo by omezení webu.

Pokud by se v generátoru nacházely opakující se vzory na obrázku by byly patrné bloky podobných sekvencí. To by znamenalo, že se sekvence opakují a vytvářejí nám vzor. Jak je možné vidět na obrázku 7.2, se žádné vzory nenacházejí. Bližší informace, jak takovéto vzory vypadají jsou uvedeny v literatuře [57, 58].

Další vlastnost, kterou lze z bitmapy pozorovat, je shluk jedniček a nul. Pokud by se v jednom místě shlukovalo příliš mnoho jedniček nebo nul, nastaly by u obrazce velké prostory s převažující barvou. Tato vlastnost by se poté projevila na výsledcích statistickým testů [57, 59].



Obr. 7.2: Vygenerovaná bitmapa pro hledání vzoru v posloupnosti.

Závěr

V teoretické části této diplomové práce byla provedena analýza možností realizací generátorů náhodných čísel. Byly probrány rozdíly mezi pravými a pseudonáhodnými generátory náhodných čísel. Uvedeny byly příklady základních generátorů a některých jevů, které lze využívat jako zdroje náhodnosti. U symetrických kryptosystémů jako je AES, které slouží například pro zabezpečení Wi–Fi sítích nebo k zabezpečení dat, jsou náhodná čísla využívána pro tvorbu klíčů. Z asymetrických kryptosystémů lze uvést RSA, které slouží pro generování veřejných a soukromých klíčů nebo DSA určeného pro vytváření a ověřování digitálních podpisů. V další části se práce zabývá principy testování náhodných čísel i samotných generátorů. Uvedeno je několik druhů testů, z nichž vybrané jsou popsány podrobněji.

Dále je zde uveden standard 802.11, jeho historie, vznik a varianty. Jsou popsány rozdílné režimy síťových karet, které mohou být využívány, pokud jsou zařízením, respektive jeho ovladačem podporovány. Jsou uvedeny hlavičky a popis jednotlivých polí, dále rozdíly mezi rámci a jejich zprávy přenášené v bezdrátových sítích.

Na základě dostupných parametrů byl zvolen způsob fungování generátoru s hodnotou RSSI a následně kombinací s časem od přijetí předešlého paketu jako vstup do hešovací funkce pro dosažení velkého rozdílu na výstupu. Dále je posloupnost poslána do LFSR registru s lineární zpětnou vazbou kde se hodnoty bitů xorují. Následně je výstupní hodnota zahešována podruhé, kde se využila vlastnost jednosměrnosti hešovací funkce pro zajištění odolnosti vůči kryptografické analýze.

Dále je zde uveden celkový návrh fungování a implementace vlastního generátoru náhodných čísel využívající parametrů z bezdrátových sítí. Pomocí webové stránky je uživatel schopen ovládat generátor, nastavovat parametry a zobrazení výstupu. Dále webová stránka umožňuje otestování náhodné posloupnosti čísel pomocí statistických testů NIST, které je popsáno v poslední kapitole. V poslední kapitole je provedeno testování posloupností za pomoci různých sad pro statistické testování náhodných čísel. Jsou zde uvedeny výsledky testů našeho generátoru a následné porovnání s generátorem náhodných čísel využívající atmosférický šum. Výsledky ukázaly, že navržený generátor v této práci splňuje většinu statistických testů.

V rámci pokračování práce by bylo vhodné vygenerování dostatečně velkého souboru pro otestování statistickými testy Dieharder. Ten by dokázal poměrně snadno určit slabiny generátoru a dle toho by mohlo dojít k úpravě algoritmu generátoru. Dalším krokem by mohla být optimalizace výkonu generátoru, aby se navýšila jeho rychlost generování posloupností.

Literatura

[1] PROCHÁZKOVÁ, Lenka. *Generátor pravých náhodných čísel.* Brno, 2019, 68 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Vlastimil Člupek.

[2] BURDA, Karel. *Kryptografie okolo nás.* Praha: CZ.NIC, z.s. p.o., 2019, 128 s. CZ.NIC. ISBN 978-80-88168-49-2.

[3] HAAHR, Mads. *Introduction to Randomness and Random Numbers* [online]. 2009 [cit. 13. 11. 2022]. Dostupné z URL: <`http://random.org/randomness/`>.

[4] BURDA, Karel. *Aplikovaná kryptografie.* Brno: VUTIUM, 2013, 255 s. ISBN 978-80-214-4612-0.

[5] KOLÁŘ, Michal. *Entropický generátor náhodných čísel.* Brno, 2015, 80 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Karel Burda.

[6] KAJAN, Michal. *Generátor náhodných čísel se zvoleným rozložením.* Brno, 2007, 38 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií, Ústav počítačových systémů. Vedoucí práce Jan Kořenek.

[7] BURDA, Karel. *Úvod do kryptografie.* Brno: Akademické nakladatelství CERM, 2015. ISBN 978-80-7204-925-7.

[8] KOÇ, Çetin Kaya. *Open Problems in Mathematics and Computational Science.* Cham: Springer International Publishing, 2014, 439s. [cit. 2. 12. 2022]. DOI: 10.1007/978-3-319-10683-0. ISBN 978-3-319-10682-3. Dostupné z URL: <`https://link.springer.com/chapter/10.1007/978-3-319-10683-0_12`>.

[9] KUMAR, Dhirendra, Chaitanya JADHAV, Prasanna MISRA a Manish GOSWAMI. *Opto-Radio Noise based True Random Number Generator.* 2020 24th International Symposium on VLSI Design and Test (VDAT), 2020, 1-5. [cit. 8. 12. 2022]. DOI: 10.1109/VDAT50263.2020.9190346. Dostupné z URL: <`https://ieeexplore.ieee.org/document/9190346`>.

[10] HERRERO-COLLANTES Miguel, Juan Carlos GARCIA-ESCARTIN. *Quantum random number generators* [online]. 2017, 89(1) [cit. 25. 11. 2022]. DOI: 10.1103/RevModPhys.89.015004. ISSN 0034-6861. Dostupné z URL: <<`https://link.aps.org/doi/10.1103/RevModPhys.89.015004`>>.

[11] UCHIDA, Atsushi, Kazuya AMANO, Masaki INOUE, et al. *Fast physical random bit generation with chaotic semiconductor lasers.* Nature Photon 2, 728–732 (2008). [cit. 10. 12. 2022]. Dostupné z URL: <https://www.nature.com/articles/nphoton.2008.227>.

[12] RÜSCHEN, Daniel, Moritz SCHREY, Johannes FREESE a Iris HEISTERKLAUS. *Generation of True Random Numbers based on Radioactive Decay.* [online]. Praha, 2017, 1-4 [cit. 2022-12-11]. Dostupné z URL: <http://poseidon2.feld.cvut.cz/conf/poster/proceedings/Poster_2017/Section_EI/EI_040_Ruschen.pdf>.

[13] KRÁLOVÁ, Magda. *Encyklopedie fyziky.* Techmania Science Center: EDUPORTÁL [online]. Plzeň [cit. 2022-12-11]. Dostupné z URL: <http://edu.techmania.cz/cs/encyklopedie/fyzika/atomy-castice/prirozena-radioaktivita/rozpadovy-zakon>.

[14] WALKER, John. *HotBits: Genuine random numbers, generated by radioactive decay.* [online] 1996 [cit. 29. 11. 2022]. Dostupné z URL: <https://www.fourmilab.ch/hotbits>.

[15] ALLINI, Ellie Noumon, Maciej SKÓRSKI, Oto PETURA, Florent BERNARD, Marek LABAN a Viktor FISCHER. *Evaluation and Monitoring of Free Running Oscillators Serving as Source of Randomness.* IACR Transactions on Cryptographic Hardware and Embedded Systems. 2018 [cit. 2022-12-11]. DOI: 10.13154/tches.v2018.i3.214-242. Dostupné z URL: <https://d-nb.info/1205104216/34>.

[16] MA, Xiongfeng, Xiao YUAN, Zhu CAO, et al. *Quantum random number generation.* npj Quantum Information 2, 16021 (2016). Dostupné z URL: <https://doi.org/10.1038/npjqi.2016.21>.

[17] REICHL, Jaroslav, Martin VŠETIČKA. *Encyklopedie fyziky.* Encyklopedie fyziky [online]. [cit. 2022-12-11]. Dostupné z URL: <http://fyzika.jreichl.com/main.article/view/744-superpozice>.

[18] MICHÁLEK, Tomáš. *Efektivní generátor náhodných čísel v nízko-výkonových zařízeních.* Brno, 2017, 66 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Radek Fujdiak.

[19] OTIPKA, Petr, Vladislav ŠMAJSTRLA. *PRAVDĚPODOBNOST A STATISTIKA: 12. Testování statistických hypotéz* [online]. [cit. 2022-12-11]. Dostupné z URL: <https://homel.vsb.cz/~oti73/cdpast1/KAP11/KAP12.HTM>.

[20] 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 Standards and Technology, 131 s. [cit. 10. 12. 2022]. Dostupné z URL: <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf>.

[21] WALKER, John. *ENT A Pseudorandom Number Sequence Test Program* [online]. 2008, 1 [cit. 2022-11-29]. Dostupné z URL: <http://fourmilab.ch/random/>.

[22] *Entropy and Random Number Generators* [online]. 2017 [cit. 2022-12-1]. Dostupné z URL: <https://calomel.org/entropy_random_number_generators.html>.

[23] ZEJDA, David. *ENT - Program pro testování sekvencí pseudonáhodných čísel* [online]. 2005 [cit. 2022-11-20]. Dostupné z URL: <https://www.root.cz/clanky/ent-program-pro-testovani-sekvenci-pseudonahodnych-cisel/>.

[24] ANDERSON, Walter. *A study of entropy: True random numbers* [online]. [cit. 2022-12-11]. Dostupné z URL: <https://sites.google.com/site/astudyofentropy/background-information/the-tests>.

[25] *The Monte-Carlo Method* [online]. 2020 [cit. 2022-12-10]. Dostupné z URL: <https://thatsmaths.com/2020/05/28/the-monte-carlo-method/>.

[26] P L'Ecuyer and R Simard. *TestU01: A software library in ANSI C for empirical testing of random number generators, User's Guide* [online]. 2013 [cit. 2022-12-10]. DIRO, University of Montreal. Dostupné z URL: <http://simul.iro.umontreal.ca/testu01/>.

[27] 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. 5. 12. 2022]. Dostupné z URL: <http://webhome.phy.duke.edu/~rgb/General/dieharder.php>.

[28] LIPTÁK, Juraj. *Nástroj pre testovanie súborov náhodných čísiel.* Brno, 2011, 55 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií, Ústav inteligentních systémů. Vedoucí práce Martin Hrubý.

[29] RAVINDRAN, Vish. *The Diehard Tests* [online]. [cit. 2022-12-10]. Dostupné z URL: <http://theurbanengine.com/blog//the-diehard-tests>.

[30] BELLAMY, James. *Randomness of D Sequences via Diehard Testing* [online]. 2013. [cit. 2022-12-10]. Dostupné z URL: <https://arxiv.org/abs/1312.3618>.

[31] BAKÓ, Zdeněk. *Analýza bezdrátového provozu Wi-Fi.* Brno, 2020, 104 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Jan Pospíšil.

[32] PETERKA, Jiří. Počítačové sítě, verze 4.0. *EArchiv: Archiv článků a přednášek Jiřího Peterky* [online]. 2015 [cit. 2022-11-19]. Dostupné z URL: <https://www.earchiv.cz/l226/slide.php3?l=16&me=20>.

[33] *EverythingRF: What is IEEE 802.11be?* [online]. [cit. 2022-11-30]. Dostupné z URL: <https://www.everythingrf.com/community/what-is-ieee-802-11be>.

[34] LOHNINGER, Hans. *Wireless Networking in the Developing World: IEEE 802.11 Wireless Networks* [online]. 2011. [cit. 2022-11-30]. Dostupné z URL: <http://www.vias.org/wirelessnetw/wndw_05_04.html>.

[35] *High on Wires: Difference - Promiscuous vs. Monitor Mode (Wireless Context)* [online]. 2008. [cit. 2022-11-30]. Dostupné z URL: <http://lazysolutions.blogspot.com/2008/10/difference-promiscuous-vs-monitor-mode.html>.

[36] POLOK, Zbyszek. *Lokalizace pro bezdrátové sítě malého dosahu.* Brno, 2013, 43 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Ondřej Hynčica.

[37] SAXENA, Mohit, Puneet GUPTA, Bijendra JAIN. *Experimental analysis of RSSI-based location estimation in wireless sensor networks* [online]. 2008. Proc. COMSWARE. 503 - 510. DOI: 10.1109/COMSWA.2008.4554465. Dostupné z URL: <https://www.researchgate.net/publication/4346354_Experimental_analysis_of_RSSI-based_location_estimation_in_wireless_sensor_networks>.

[38] BERG, Johannes. *Radiotap* [online]. [cit. 2023-05-15]. Dostupné z URL: <http://www.radiotap.org/>.

[39] ŠVANDA, Pavel. *Nástroj pro generování rámců podle standardu 802.11.* Brno, 2013, 60 s. Diplomová práce. Vysoké učení technické v Brně. Fakulta informačních technologií, Ústav inteligentních systémů. Vedoucí práce Matej Kačic.

[40] SHARP, Jeremy. *802.11 Frame Types and Formats - How I WI-FI* [online]. [cit. 2023-05-15]. Dostupné z URL: <https://howiwifi.com/2020/07/13/802-11-frame-types-and-formats/>.

[41] BLAŽKOVÁ, Tereza. *Jaké programovací jazyky pojedou v roce 2022?* [online]. [cit. 2023-05-05]. Dostupné z URL: <https://www.itnetwork.cz/jake-programovaci-jazyky-pojedou-2022>.

[42] CARBONNELLE, Pierre. *TIOBE Index for May 2023* [online]. [cit. 2023-05-10]. Dostupné z URL: <https://www.tiobe.com/tiobe-index/>.

[43] CARBONNELLE, Pierre. *PYPL PopularitY of Programming Language* [online]. [cit. 2023-05-10]. Dostupné z URL: <https://pypl.github.io/PYPL.html>.

[44] NumPy. *Numpy user guide* [online]. [cit. 2023-05-01]. Dostupné z URL: <https://numpy.org/doc/stable/user/index.html#>.

[45] BIONDI, Philippe. *Scapy 2.5.0 documentation* [online]. [cit. 2023-05-01]. Dostupné z URL: <https://scapy.readthedocs.io/en/latest/introduction.html>.

[46] MQ. *Lekce 1 - Úvod do frameworku Flask a webových aplikací v Pythonu* [online]. [cit. 2023-05-15]. Dostupné z URL: <https://www.itnetwork.cz/uvod-do-frameworku-flask-a-webovych-aplikaci-v-pythonu>.

[47] *Welcome to Flask — Flask Documentation (2.3.x)* [online]. [cit. 2023-05-15]. Dostupné z URL: <https://flask.palletsprojects.com/en/2.3.x/>.

[48] ACSANY, Philipp. *Primer on Jinja Templating - Real Python* [online]. [cit. 2023-05-15]. Dostupné z URL: <https://realpython.com/primer-on-jinja-templating/>.

[49] BURDA, Karel. *Bezpečnost komunikačních systémů [online]* Brno: Vysoké učení technické v Brně, 2022. ISBN 978-80-214-6044-7.

[50] ŠINDELÁŘOVÁ, Anna. *Kryptoměny.* Brno, 2019, 68 s. Bakalářská práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií, Ústav jazyků. Vedoucí práce Milan Smutný.

[51] *hashlib — Secure hashes and message digests* [online]. [cit. 2023-05-17]. Dostupné z URL: <https://docs.python.org/3.10/library/hashlib.html>.

[52] JANOVSKÝ, Dušan. *Javascript - úvod* [online]. [cit. 2023-05-17]. Dostupné z URL: <https://www.jakpsatweb.cz/javascript/javascript-uvod.html>.

[53] DAVIES, Alan. *Using Python Flask and Ajax to Pass Information between the Client and Server* [online]. [cit. 2023-05-17]. Dostupné z URL: <https://towardsdatascience.com/using-python-flask-and-ajax-to-pass-information-between-the-client-and-server-90670c64d688>.

[54] *Async function - JavaScript MDN* [online]. [cit. 2023-05-17]. Dostupné z URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function>.

[55] *Using the Fetch API - Web APIs MDN* [online]. [cit. 2023-05-17]. Dostupné z URL: <https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch>.

[56] *Get started with Bootstrap* [online]. [cit. 2023-05-17]. Dostupné z URL: <https://getbootstrap.com/docs/5.3/getting-started/introduction/>.

[57] REED, Nathan. *Quick And Easy GPU Random Numbers In D3D11 – Nathan Reed's coding blog* [online]. [cit. 2023-05-15]. Dostupné z URL <https://www.reedbeta.com/blog/quick-and-easy-gpu-random-numbers-in-d3d11/>.

[58] GOLDSTEIN, Michael. *Statistical Tests for Random Number Generators* [online]. [cit. 2023-05-15]. Dostupné z URL: <https://mgold.io/2015/02/17/statistical-tests-for-random-number-generators.html>.

[59] *RANDOM.ORG - Statistical Analysis* [online]. [cit. 2023-05-15]. Dostupné z URL: <https://www.random.org/analysis/>.

Seznam symbolů a zkratek

ACK	Potvrzovací zpráva
AES	Standard pokročilého šifrování
AJAX	Asynchronous JavaScript and XML
AP	Přístupový bod
API	Aplikační programové rozhraní
ARP	Adress Resolution Protocol
ASCII	Americký standardní kód pro výměnu informací
BSD	Berkeley Software Distribution
CRC	Cyklický redundantní součet
CSS	Kaskádové styly
CTS	Clear to Send
ENT	Program pro testování sekvencí pseudonáhodných čísel
FCS	Kontrolní sekvence rámce
HTML	Hypertextový značkovací jazyk
ID	Identifikátor
IEEE	Institut pro elektrotechnické a elektronické inženýrství
JSON	JavaScript Object Notation
LCG	Lineární kongruentní generátor
LFSR	Lineární zpětnovazební posuvný registr
MIT	Massachusetts Institute of Technology
NIST	Národní institut standardů a technologie v Americe
OPSO	Overlapping–Pairs–Sparse–Occupancy test
OQSO	Overlapping–Quadruples–Sparse–Occupancy test
PNG	Grafický formát pro bezeztrátovou kompresi rastrové grafiky

POST	Nástroj pro přeměnu stylů za pomoci JavaScriptu
PRNG	Generátor pseunáhodných náhodných čísel
PS	Úsporný režim
PYPL	PopularitY of Programming Language index
QoS	Quality of Service
REST	Representational State Transfer
RNG	Generátor náhodných čísel
RSA	Asymetrická šifra
RSSI	Indikátor síly přijímaného signálu
RTS	Ready to Send
RX	Přijaté rámce
SHA	Hešovací funkce
STS	Soubor statistických testů
TIOBE	The Importance of Being Earnest index
TRNG	Generátor pravých náhodných čísel
URL	Jednotný lokátor zdroje
USB	Univerzální sériová sběrnice
VA	Voltampérová charakteristika
VLAN	Virtuální lokální síť
WECA	Wireless Ethernet Compatibility Alliance
Wi–Fi	Wireless Fidelity
χ^2	Chí–kvadrát rozdělení pravděpodobnosti
π	Ludolfovo číslo

A Obsah přiloženého CD

- `/` ..... kořenový adresář přiloženého CD
  - `xfrolk03_diplomova_prace.pdf` ..... elektronická verze diplomové práce
  - `templates` ..... HTML stránka
    - `index.html`
  - `app.py` ..... Hlavní aplikace
  - `Sniffer.py` ..... Zachytávací aplikace