Fraktální a Neuro-Fraktální Kryptologie

Fractal and Neuro-Fractal Cryptology

Bc. Jaroslav Popelka

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jaroslav POPELKA**
Osobní číslo:	**A08468**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Fraktální a Neuro-Fraktální Kryptologie**

Zásady pro vypracování:

1. **Vypracujte literární rešerši na dané téma.**
2. **Vytvořte návrh systému šifrování pomocí neuronových sítí a fraktální geometrie.**
3. **Navržený systém otestujte a zaměřte se na možnost prolomení šifrovacího systému (např. metodou frekvenční analýzy).**
4. **Vytvořte aplikace (např. v prostředí Mathematica a WebMathematica) a taktéž prezentace obsahující popis problematiky, analýzu a grafické ukázky.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **ZELINKA, I.: Aplikovaná Informatika. Zlín. UTB, 1999. 183 s. ISBN 80-214-1423-5.**
2. **GURNEY, K.: An Introduction to Neural Networks. 1st edition. CRC Press, 1997. 234 s. ISBN 978-1857285031.**
3. **ZELINKA I, VČELAŘ F., ČANDÍK M.,: Fraktální geometrie-- principy a aplikace. BEN, Praha, 2006, 160 s. ISBN 80-7300-191-8.**
4. **KATZ, Jonathan. Introduction to Modern Cryptography: Principles and Protocols. Chapman & Hall, 1 edition. 2007. 552 s. ISBN 978-1584885511.**
5. **BÍLA J.: Umělá inteligence a neuronové sítě v aplikacích, ČVUT, 1996, ISBN 80-01-01275-1.**
6. **NOVÁK M., FABER J., KUFUDAKI O.: Neuronové sítě a informační systémy živých organismů, Grada, 1993, ISBN 80-58424-95-9.**
7. **MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: Umělá inteligence 4., Academia, 2003, ISBN 80-200-1044-0.**

Vedoucí diplomové práce: **Ing. Zuzana Oplatková, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **8. června 2010**

Ve Zlíně dne 19. února 2010

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Tématem této práce je fraktální a neurofraktální šifrování. Jedná se o moderní a stále se rozvíjející obor kryptologie. Čtenář je nejprve seznámen se základní teorií potřebnou pro další porozumění látky. Je obeznámen se základními principy šifrování, jsou mu nastíněny základy fraktální geometrie i neuronových sítí. Taktéž je stručně představeno programové vybavení ve formě software Mathematica. Praktická část se pak ve dvou samotných kategoriích věnuje jednotlivým částem zadání. Jsou objasněny principy, na kterých jsou fraktální a neurofraktální kryptologie založeny a zároveň jsou představeny vytvořené demonstrativní programy. Součástí práce je pak také několik grafických ukázek.

Klíčová slova: fraktály, neuronové sítě, kryptologie, šifrování, Mathematica

ABSTRACT

The theme of this work is fractal and neuro-fractal encryption. It is a modern and still developing field of cryptology. The reader is first introduced to the basic theory needed for further understanding of the substance. Is familiar with basic principles of encryption, as he outlined the basics of fractal geometry and neural networks. It is also briefly introduced the software in the form of Mathematica software. The practical part in two categories devoted themselves to individual parts of the assignment. They explained the principles upon which the fractal and neurofraktální based cryptography, while being introduced to set up demonstration programs. Part of this work is also some graphic examples.

Keywords: fractals, neural networks, cryptology, coding, Mathematica

Rád bych na tomto místě poděkoval vedoucí mé diplomové práce paní Ing. Zuzaně Oplatkové Ph.D. za její vstřícnost, odbornou pomoc, rady, návrhy i připomínky, které mi poskytla během vytváření této diplomové práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně dne …….

podpis diplomanta

OBSAH

ÚVOD

Kryptologie je vědní obor zabývající se šifrováním dat. A svým způsobem se nejedená o moderní obor. První zmínky o snaze skrýt nějaký text jsou několik tisíc let staré a lze je hledat na území starého Egypta. Následovali Řekové a zřejmě nejslavnější „starou“ šifru používal pro zprávy z bojiště Julius Caesar. Poté následovalo několika století útlumu, přičemž potřeba skrytí textu stále existovala. V 16. století nastala jakási šifrovací renesance a bylo vytvořeno několik nových kryptologických principů. Dalším významným bodem v historii šifrování bylo vynalezení telegrafu. Ve 20. století se pak stále více prosazovaly mnohdy velmi složité kryptografické stroje. Složitost šifer se přenesla i do století současného. Používané šifry založené na nejrůznějších principech používají objemné klíče, které zaručují jejich nerozluštitelnost. Tedy alespoň prozatím.

Všechny historické šifry totiž mají jednu společnou vlastnost – dříve, či později byly prolomeny. A stejně na tom jednou budou i šifry současné. Zvláště pak, pokud bude nadále platit Moorův zákon, podle kterého se výkon procesorů každé dva roky zdvojnásobí. Je tedy nutné stále současné šifry inovovat a vylepšovat, případně je nahrazovat šiframi novými. Mezi nejvíce diskutované možnosti budoucnosti šifrování patří:

- kvantová kryptografie - využití přírodních zákonů kvantové mechaniky
- kvazarová kryptografie – využití náhodných energetických toků z hvězd
- neurofraktální kryptologie – skloubení fraktálů a neuronových sítí

Tato diplomová práce má za úkol seznámit čtenáře právě se základy neurofraktální kryptologie a to včetně ukázek v software Mathematica.

I. TEORETICKÁ ČÁST

1 KRYPTOLOGIE

Již od pradávna je komunikace jednou ze základních potřeb lidstva. S postupem času však lidé zjistili, že mnohdy může tato jejich komunikace být kontrolována, či dokonce narušována. Tyto skutečnosti tak vedly k počátkům teorie šifrování, dešifrování a utajené komunikace vůbec…

Kryptologie je věda zabývající se šifrováním ze všech úhlů pohledu. Obecně ji lze rozdělit na dvě hlavní disciplíny, kterými jsou kryptografie a kryptoanalýza.

Kryptografie neboli šifrování, je nauka pojednávající o metodách utajování smyslu zpráv převodem do podoby, která je čitelná jen se znalostí dané šifry. Název je složeninou dvou slov pocházejících z řečtiny - kryptós znamená skrytý a gráphein lze přeložit jako psát. Kryptografie se tak jako lidská civilizace po staletí vyvíjela k větší složitosti a mnohokrát ovlivnila běh dějin.

Kryptoanalýza je svým způsobem protikladem ke kryptografii. Zabývá se metodami získávání obsahu šifrovaných informací bez přístupu k tajným informacím, které jsou za normálních okolností potřeba, tzn. především k tajnému klíči. Název této nauky je opět složen ze dvou řeckých slov - kryptós a analýein (uvolnit, rozvázat).

1.1 Základy kryptologie

Při návrhu každého šifrovacího systému je důležité zohlednit určitá kritéria, která mohou pomoci při výběru správné šifry. Dle Shannona jsou to pro posouzení kvality šifrovacího systému následující kritéria: [1]

- spolehlivost a odolnost
- objem klíče
- složitost šifrování a dešifrování
- šíření chyb
- transformace statických charakteristik

Během provozu šifrovacího systému je pak potřeba dodržet několik pravidel:

- neodesílat stejný text zašifrovaný různými klíči
- omezit používání frekventovaných slov
- omezit používání typických kombinací písmen, interpunkce, mezer

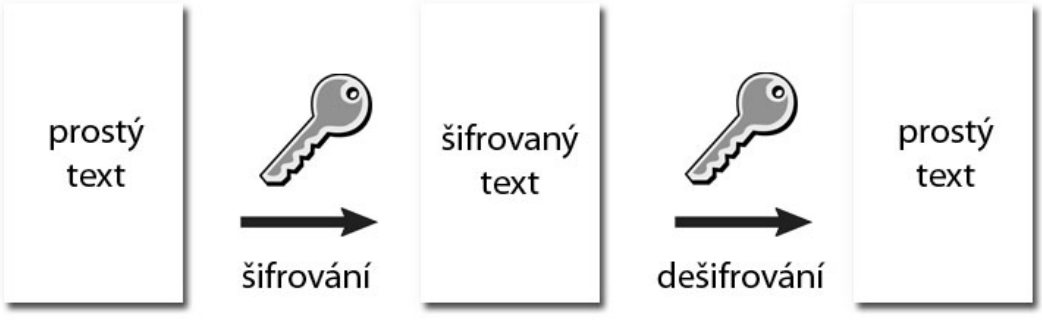
V oblasti šifrování se dále používá několik obecných pojmů. Prvním z nich je text, který dělíme na šifrovaný a otevřený. Ten je napsán běžným jazykem, kde každý takový jazyk má svou charakteristickou četnost jednotlivých písmen, čehož se využívá především při analýze šifer. V případě jednoduchých šifer tak lze zprávu dešifrovat pouze na základě této statické charakteristiky jazyka.

Samotný text je tvořen symboly – nejčastěji se jedná o písmena, číslice a interpunkční znaménka. Souhrn všech použitých symbolů tvoří abecedu textu, přičemž abeceda otevřeného a šifrovaného textu mohou a nemusí být stejné.

Důležitým pojmem je také šifrovací algoritmus. Jedná se o proces transformace, která převede otevřený text na šifrovaný text a naopak. Při procesu zašifrování i odšifrování se zpravidla používá klíče, což je tajná informace, bez níž nelze šifrový text přečíst. [2]

Dle způsobu samotného šifrovaní, práce s abecedou a použitého klíče můžeme šifrovací systémy rozdělit do 4 následujících skupin:

- Transpoziční systémy – přeskupování jednotlivých znaků zprávy
- Transkripční systémy – nahrazování znaků obecného textu jinými znaky
- Polyalfabetické šifry – stejnému znaku abecedy obecného textu je vždy přiřazen jiný znak abecedy šifrovaného textu
- Systémy s veřejným klíčem – použití veřejně známého šifrovacího systému a klíče



Obr. 1 - Obecný postup šifrování

1.2 Historie kryptologie

Počátky kryptologie lze hledat ve starém Egyptě. Okolo roku 1900 před naším letopočtem používali tamní obyvatelé pro zápis citlivých dat atypické hieroglyfy. Použitím takovýchto značek měli zajištěno, že k důvěrným informacím bude mít přístup pouze předem určená skupina lidí. Zároveň je nutno poznamenat, že i použití běžných hieroglyfů je možné považovat za jistou formu šifrování, jelikož mnoho lidí neumělo hieroglyfy číst, natož je pak třeba i psát.

Velkou měrou se o rozvoj v oblasti šifrování zasloužili staří Řekové. Již kolem roku 350 před naším letopočtem navrhl vojevůdce Aeneus Tacticus okolo dvaceti šifrovacích klíčů rozdělených do dvou skupin. Řecký historik Plutarchos zdokumentoval vznik prvního transpozičního šifrovacího systému (přeskupení písmen otevřeného textu podle předem určených pravidel) zvaného SCYTALE. Tento jednoduchý systém používali sparťanští vojevůdci již kolem roku 500 před naším letopočtem. Na dřevěnou hůl o daném průměru se navinula tenká kožená nebo pergamenová páska a napsala se na ni tajná zpráva. Poté byla páska opět sejmuta a doručena příjemci, který vlastnil podobnou dřevěnou hůl o stejném průměru. Opětovným namotáním pásky tak mohl přečíst šifrovanou zprávu. Při pokusu o přečtení zprávy na holi o jiném průměru získal útočník pouze nesmyslnou změť písmen.

Obr. 2 - SCYTALE

Dalším velkým objevem byla tzv. Polybiova šifrovací mřížka, jejíž princip byl, stejně jako u většiny antických šifer, velice jednoduchý. Abeceda byla vepsána do obdélníkové mřížky a každé písmeno reprezentováno dvojicí čísel, na jejichž průsečíku řádku a sloupce se dané písmeno nacházelo. Namísto čísel byly později v praxi použity také písmena a jiné nestandardní symboly.

Velmi slavnou se stala Caesarova šifra pocházející z roku 63 před naším letopočtem. Jedná se o klasický substituční systém, kdy jsou znaky otevřeného textu nahrazovány jinými znaky, a to dle předem dohodnutého systému. Julius Caesar popsal tuto šifru ve svém nejproslulejším díle *Zápisky o válce galské* a kromě vojenské komunikace ji například používal i při dopisování s egyptskou královnou Kleopatrou. Princip byl opět velmi jednoduchý: během šifrování byl každý znak nahrazen znakem, který je v abecedě o 3 pozice před ním. Substituční klíč tedy vypadal takto:

- otevřený text
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
- zašifrovaný text
  X Y Z A B C D E F G H I J K L M N O P Q R S T U V W

Algoritmus pro dešifrování byl obdobný: každý znak se nahradil znakem, který je o tři pozice za ním. Později byla tato šifra zdokonalena proměnlivou hodnotou posunutí.

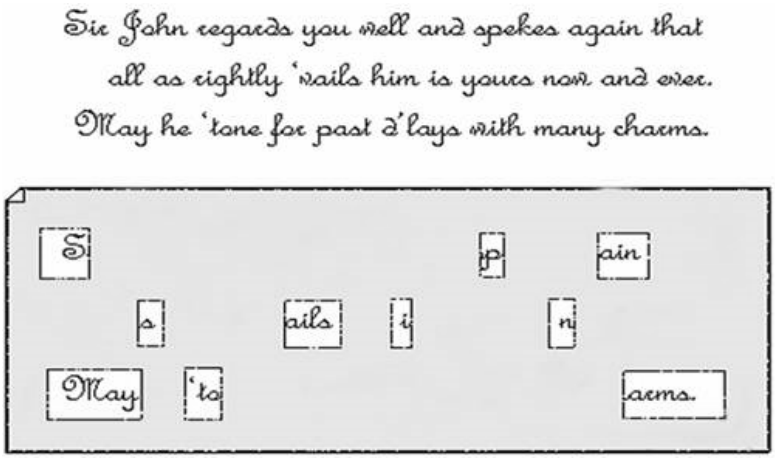
Je možné napsat, že s Caesarovou šifrou končí první velké kryptologické období. Několik následujících století totiž čeká tento obor útlum. V Byzantské říši byla pro tamního vládce napsána kniha pojednávající o základních šifrovacích metodách a postupech, okolo roku 855 bylo ve stejné oblasti publikováno několik šifrovacích klíčů, které však byly určeny pouze pro účely magie. V arabských zemích byl zkoumáním frekvence výskytu jednotlivých hlásek ve slovech položen základ kryptoanalýzy.

K opětovnému rozvoji kryptologie došlo na počátku 16. století. Roku 1518 napsal benediktinský opat ze Spanheimu Johannes Tritheim knihu *Steganographia* popisující stenografickou šifru, jejímž principem je nahrazení každého písmene slovem v předem určené tabulce. Téhož roku vydal první tištěnou knihu zabývající se popisem některých známých šifrovacích algoritmů nazvanou *Polygraphiae libri sex.* Ta byla několikrát opakovaně vydávána a v letech 1561 a 1564 byla vydána také její francouzská verze. Tehdejší panovnické rody se však obávaly, že by mohl vyzradit příliš mnoho tajemství a proto jej pro jistotu označili za čarodějníka spolčeného s ďáblem.

Ve stejné době byly vydány také další knihy věnující se známým šifrovacím metodám. Příkladem může být kniha *Opus novum…principibus maxime vtilissimum pro cipharis*, jejímž autorem je Jacopo Silvestri, jenž v ní popsal šest šifrovacích metod včetně tehdy stále velice oblíbené Caesarovy šifry. Kniha byla psána pro praktické použití a mohla tak oslovit širokou čtenářskou obec. [3]

Významnou osobností doby byl také italský filozof Giovanni Battista Della Porta, který roku 1563 vydal čtyřdílný spis s názvem *De furtivis literarum notis*. V knihách popsal rozdělení šifer na transpoziční, substituční a substituční symbolové, jejichž dominantou je použití nestandardní abecedy. Zároveň také navrhl použití různých „nadbytečných" znaků, které měli ztížit práci při případném útoku na šifrovaný text.

Další italský renesanční učenec Gerolamo Cardano věnující se především matematice, fyzice a astronomii, vydal během svého života přes 130 knih. Kryptologické teorii se přitom alespoň z části věnoval hned ve dvou z nich: *De subtilitate* a *De Rerum Varietate.* Popisuje v nich také šifru založenou na mřížce vyrobené z pevného materiálu s nepravidelně umístěnými otvory, skrze které lze na list papíru zapsat zprávu, která se po odstranění mřížky a dopsání písmen do vzniklých mezer stává zprávou skrytou. Oprávněný příjemce vlastnící stejnou mřížku poté pouze tuto přiloží na list papíru a v jednotlivých otvorech si původní zprávu bez obtíží přečte. Tato v 16. a 17. století velmi oblíbená šifra se tedy dle svého objevitele jmenuje Cardanova mřížka. [3]



Obr. 3 - Princip Cardanovy mřížky

V roce 1586 vydal francouzský diplomat Blaise de Vigenére svou 600. stránkovou knihu *Traicté des chiffres*, ve které mimo jiné popsal několik šifrovacích algoritmů včetně mnoha praktických ukázek. Známý je však také díky jednoduché a oblíbené šifře založené na tabulce složené z abecedy, kdy ta je vždy na každém řádku záměrně posunuta o jeden znak. Kombinací písmen z textu a hesla pak dochází v tabulce k určení znaku, přičemž množina těchto znaků vytváří šifrovaný text. Příjemce znalý hesla poté postupuje opačně.

Důležitým milníkem lidstva byl vynález telegrafu. Ve skutečnosti se však nejedná o vynález několika posledních století. Slovo telegrafovat řecky znamená psát na dálku, což již okolo roku 450 před naším letopočtem splňoval vynález řeckých filosofů Kleoxenese a Demokritose. Ti zapsali jednotlivá písmena do tabulky 5 na 5 znaků, načež vojáci na jednotlivých stanovištích pomocí zapálených pochodní určovali pozici posílaných písmen. Tento jednoduchý ohňový telegraf však nebyl příliš spolehlivý. Na několik století pak telegraf přešel do ústraní. Až v období Renesance se o vzkříšení pokusil dnes známý vynálezce Robert Hook. V roce 1684 představil zařízení ve tvaru dřevěné brány, k jejímuž břevnu pomocí provázků a kladek vytahoval trojúhelníkový terčík, jehož jednotlivým polohám poté přiřadil písmena. Jeho myšlenka se sice v té době neujala, výrazně však pomohla následujícímu vynálezci. Roku 1792 nabídl mladý Francouz Claude Chappe projekt optického telegrafu, který pracoval na principu semaforu složeného z vahadla a dvou kratších ramen, přičemž dle jejich poloh bylo možné určit až 92 různých znaků. Princip se velmi osvědčil a na začátku 19. století byly semafory rozesety po celém území Francie. Objev elektrického proudu však pro tento systém znamenal konec a semafory se přestěhovaly na železnici. S objevem elektřiny se však začaly objevovat nové telegrafní systémy. Thomas Sommering využil poznatku, že elektřina rozkládá slanou vodu na vodík a kyslík a sestrojil komplikovaný a nepříliš spolehlivý bublinkový telegraf. Vztahy mezi elektřinou a magnetismem se zase snažili pochopit pánové Charles Wheatstone a William Cook, kteří sestrojili magnetický telegraf s pěti magnetkami, které se kombinací přenášených signálů otáčely, přičemž jejich průsečíky ukazovaly na jednotlivá písmena. Zlomem byl až Morseův telegraf. Původně malíř Samuel Finley Morse přeměnil svůj ateliér na dílnu, ve které v roce 1837 vynalezl přístroj umožňující zapisovat abecedu složenou z teček a čárek. Telegraf však nebyl příliš dokonalý a po ukázce odborníkům z newyorské univerzity se Morse dočkal posměchu. Během čtyř měsíců přístroj zdokonalil a stal se jedním z nejuznávanějších vynálezců své doby. [4]

Z hlediska kryptologie byla důležitá právě jeho abeceda složená z teček a čárek. Jednalo se o první šifrovací systém, ve kterém odesilatel a příjemce mohli rychle komunikovat na dálku bez potřeby třetí strany k přenosu šifrované informace. Tím odpadli starosti s bezpečností přenosu. Do této doby bylo vždy potřeba zajistit bezpečný přenos šifrované informace a šifrovacího klíče. Ne zřídkakdy se tak stávalo, že posel vyslaný s takovýmto materiálem byl přepaden a fyzicky donucen k prozrazení tajných informací.

Roku 1917 objevil Gilbert Vernam nový druh šifry, která je i v současné době považována za bezpečnou a je známa pod názvem Vernamova šifra. K telegrafní pásce přiložil ještě jednu, která obsahovala náhodně generované heslo. Otevřený text se spolu s heslem načítají do šifrovaného textu, přičemž je zde uplatněna binární operace XOR. V průběhu čtyřicátých let se matematikům podařilo dokázat, že jde o bezpečný systém, což je hlavní důvod, proč se s ním často setkáváme i v současné době.

Během 20. století se s rozvojem techniky začaly objevovat také stále složitější kryptografické stroje. Zřejmě nejznámějším z nich byla Enigma, kterou si již roku 1918 nechal patentovat německý inženýr Arthur Scherbius a ještě téhož roku ji nabídl německému námořnictvu. To tento šifrovací stroj začalo používat, přičemž jej v následujících letech několikrát zdokonalilo. Enigma svým vzhledem připomíná klasický psací stroj, před jehož klávesnicí je umístěno 26 konektorů, které slouží k propojení jednotlivých písmen. Za klávesnicí se pak nachází svítící deska, která obsahuje taktéž 26 písmen. Vlastní šifrovací mechanismus se skládá z prostoru, na jehož stranách jsou dvě ozubená kola, mezi která se vkládají další tři, přičemž se vybírají z pěti možných. Po stisku písmene na klávesnici se první kolo otočilo vždy o jednu pozici. Poté, co se první okolo otočí 26x, otočí se kolo druhé a nakonec i třetí. Bylo tak možno dosáhnout 17576 různých stavů. Pro ztížení práce kryptoanalytikům byla navíc délka zprávy omezena na 250 znaků, aby se nemohly opakovat sekvence, což by útočníkovi velice pomohlo při luštění kódu. [5]

Obr. 4 - Enigma

V 70. letech minulého století se kryptologie stala vědeckou disciplínou a byla teoreticky navržena asymetrická kryptografie. Realizována však byla až o několik let později. Od této doby již kryptografie pokračuje ve stejném duchu až dodnes.

1.3 Moderní šifrovací systémy

Moderní šifrovací systémy lze dle použitého klíče rozdělit do dvou skupin.

Symetrické šifrovací systémy, někdy též nazývané konvenční, používají k šifrování i dešifrování jediný klíč. Výhodou takovýchto šifer je jejich nízká výpočetní náročnost, nevýhodou je pak nutnost sdílení tajného klíče, tudíž se odesílatel i příjemce tajné zprávy musí na tomto klíči předem domluvit. Dle způsobu zpracování otevřeného textu lze dále tuto skupinu dělit na šifry proudové, které zpracovávají text po jednotlivých bitech a šifry blokové, které otevřený text rozdělí na bloky o stejných velikostech.

Druhou skupinou jsou asymetrické šifrovací systémy, což je soubor metod, které pro šifrování i dešifrování používají rozdílné klíče – volně přístupný veřejný klíč je určený pro zašifrování textu, soukromý klíč je dostupný pouze tomu kdo má danou zprávu dešifrovat. Tato dostupnost jednotlivých klíčů je také největších výhodou asymetrického šifrování. [6]

1.3.1 DES

Symetrický algoritmus DES šifruje 64 bitů otevřeného textu na 64 bitů šifry. Každý osmý bit je však kontrolní, efektivní délka klíče je tedy pouze 56 bitů. Kvůli této relativně nízké délce klíče byl v roce 1998 přijat standard známý pod názvem Triple DES, který se od klasického DES liší tím, že stejná data projdou algoritmem třikrát, čímž se zvýší efektivní délka klíče (168b).

Počátky algoritmu sahají do roku 1973, kdy ministerstvo obchodu USA vyhlásilo soutěž na vytvoření šifrovacího standardu pro ochranu důvěrných dat v informačních systémech. Vysokým nárokům však nevyhověl žádný z navrhovaných algoritmů, a proto se soutěž konala v roce 1974 znovu. Vítězem se stala firma IBM, která pouze zdokonalila svůj stávající algoritmus Lucifer. Následně byl DES přijat jako šifrovací standard pro zabezpečení neutajovaných dat v civilním a vládním sektoru s předpokládanou délkou používání deset až patnáct let, s podmínkou, že jeho bezpečnost bude každých pět let kontrolována. Již v roce 1975 se však začalo spekulovat o bezpečnosti tohoto algoritmu. Podle mnohých odborníků totiž k rozluštění šifry stačilo vyzkoušet všechny možné kombinace klíčů. V roce 1997 vypsala agentura RSA kryptoanalytickou soutěž, v níž bylo cílem tuto šifru prolomit. Po necelých pěti měsících byla šifra za pomoci Internetu prolomena.

1.3.2 IDEA

Tento algoritmus byl vyvinut v roce 1991 ve Švýcarsku jako alternativa k šifrovacímu standardu DES. Jedná se o symetrickou šifru pracující po 64 bitových blocích za použití 128 bitového klíče. Skládá se z řady osmi identických transformací a vstupní transformace. Procesy šifrování a dešifrování jsou podobné. Algoritmus odvozuje velkou část své bezpečnosti ze střídání operací z různých skupin – modulární sčítání a násobení a bitové nonekvivalence (XOR), které jsou v jistém smyslu algebraicky neslučitelné. Do dnešní doby není znám úspěšný útok na prolomení této šifry a ta je tedy považována za bezpečnou. Z důvodu existence rychlejších algoritmů a pokroků v kryptografii se však šifra příliš nepoužívá.

1.3.3 AES

Stejně jako TRIPLEDES, vznikla také šifra AES jako reakce na prolomení šifry DES. Jedná se o symetrickou šifru s délkou klíče 128, 192 nebo 256b, přičemž data šifruje postupně po blocích s pevnou délkou 128b. Velkou výhodou šifry je její rychlost. Stejně jako u předchozí šifry, ani u AES nebyl dosud zaznamenán úspěšný útok na prolomení. Šifra je tedy považována za bezpečnou, varianta s klíčem 256b je hojně používána.

1.3.4 RSA

Objev tohoto prvního v praxi reálně použitelného šifrovacího systému s veřejným klíčem oznámili jeho tvůrci Rivest, Shamir a Adleman v dubnu roku 1977. RSA představuje systém, který je relativně bezpečný, ale značně pomalý. V hardwarové implementaci šifruje zhruba tisíckrát pomaleji než DES, v softwarové pak téměř stokrát.

Veřejný klíč je generován s použitím velkých prvočísel a celá bezpečnost RSA je založena na problému faktorizace velkých čísel na prvočísla. Základem je výběr dvou velkých (řádově stovky cifer) prvočísel, která se vynásobí. Na základě jejich součinu je vygenerován jak veřejný, tak privátní klíč. Bez znalosti původních prvočísel je prakticky nemožné součin rozložit zpět na počáteční prvočísla.

Při použití klíče s délkou 1024 a více bitů lze o šifře RSA hovořit i v dnešních dnech jako o šifře bezpečné, a proto je tento algoritmus využíván například také pro digitální podpis.

2 FRAKTÁLNÍ GEOMETRIE

Významná umělecká díla dvacátého století nabízejí mnoho příkladů využití geometrických útvarů. Tvarově jednoduché a často izolované okouzlují diváka iluzí dokonalosti. Fraktální umění, vytvářející atraktivní vzory na základě rovnic, se může stát novým přístupem k využití matematického aparátu pro design a tvorbu v umění. Počínaje devadesátými lety dvacátého století mnoho umělců před monitory svých počítačů a pouze s pomocí příslušného programového vybavení a s myší jako jediným nástrojem manipuluje barvami, profily a stínováním za účelem vytvoření dekorativního designu, krásného jen ze své podstaty. Fraktály mají nicméně využití i v jiných oblastech než v umění. [7]

2.1 Historie vzniku

Na přelomu 19. a 20. století se v mnoha vědních oborech začaly objevovat nové jevy a konstrukce, které se od těch ideálních značně lišily a byly tak přijímány se značným odporem. Karl Weierstrass objevil v roce 1872 spojitou funkci, která nemá v žádném svém bodě derivaci, Helge von Koch přišel roku 1904 s nekonečně dlouhou křivkou ohraničující konečnou plochu. Stejně tak ve fyzice začala klasická mechanika narážet na vážné problémy, kdy byly při výpočtech pohybu tří hmotných těles objeveny podivné nestability směřující k chaotickému chování v situaci, která sama o sobě nijak chaotická nebyla.

V 60. letech počítal americký meteorolog Edward Lorenz vývoj zjednodušeného modelu počasí obsahujícího pouze tři proměnné měnící se v čase. Svými výsledky byl velmi překvapen, jelikož daný model vykazoval značnou nestabilitu a chaotický vývoj. K výraznému zpřesnění výsledků nedocházelo ani změnou výchozích dat, Lorenz si však povšimnul, že vypočtená cesta vývoje se pohybuje po jistém podivném útvaru.

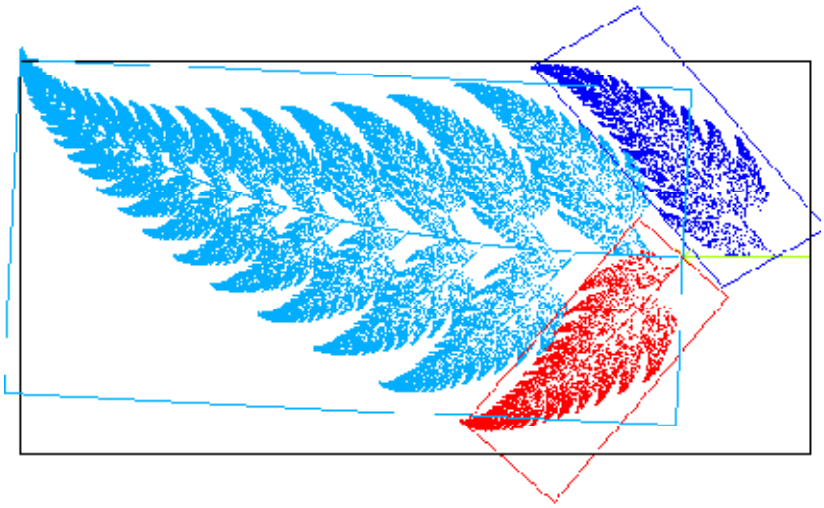
Obdobně byl svými výsledky překvapen také francouzský matematik Benoit Mandelbrot. Při studiu chyb u přenosu signálů v telekomunikačních sítích, kdy se v signálech střídaly intervaly bez chyb s chybnými, si povšiml jisté pravidelnosti. Po každém zvýšení přesnosti měření se totiž intervaly, které se dříve jevily jako bezchybné, rozpadly na množinu intervalů chybných a bezchybných. Při studiu kolísání cen na burze se později setkal s obdobným případem, když zjistil, že jednotlivé průběhy cen jsou si až nápadně podobné. Od té doby se teorii soběpodobnosti stále více věnoval, dokud nedospěl k domněnce, že v přírodě existuje určitý skrytý fraktální řád. [1]

Dnes víme, že všechny tyto jevy, které si badatelé v první polovině 20. století nedokázali vysvětlit, byly prvními náznaky fraktální geometrie a vedly k samotnému vzniku tohoto mladého odvětví matematiky. Již meziválečné generaci matematiků se teoreticky podařila prokázat existence fraktálního světa, avšak až Mandelbrotova neochvějná tvrdošíjnost vedla v 70. letech minulého století k vzniku pojmu fraktál a s ním spojenému oboru fraktální geometrie.

2.2 Fraktály a jejich principy

Pod pojmem fraktál si lze představit objekt, jehož geometrická struktura se v základním útvaru opakuje až do nekonečna. Pojem nekonečno je však nutno brát v matematickém slova smyslu, jelikož ve fyzikálním světě vždy existuje nějaká hranice, za kterou takové opakování sekvencí končí.

Obecně lze objekty ve fraktální geometrii rozdělit do dvou skupin, a to na fraktály soběpodobné a fraktály soběpříbuzné. Do první skupiny lze zařadit struktury, se kterými je možné se setkat jen v matematických konstrukcích. Charakteristickým znakem je pro ně to, že původní motiv se v nich neustále opakuje a tak je jakýkoliv výsek fraktálu vždy přesnou kopií původního objektu. Do druhé skupiny pak patří struktury, se kterými se setkáváme dnes a denně. Příkladem jsou mraky, lesy, hory a další objekty z blízkého i vzdáleného okolí. Charakteristická je pro ně vlastnost, kdy kterýkoliv výsek není přesnou, ale jen podobnou kopií původního útvaru. [8]



Obr. 5 - Soběpodobný fraktál

V současné době se pro konstrukci fraktálů používá především dvou algoritmů a to algoritmu IFS a algoritmus TEA.

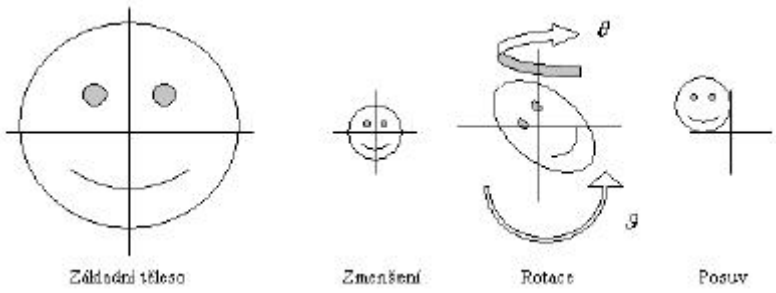
2.2.1 IFS

V případě algoritmu IFS je konstrukce fraktálů možná pomocí afinních transformací, které provádí s daným objektem několik základních operací – rotaci, zmenšování a posuv. Matematický popis daných transformací je dán těmito vztahy:

$$w(x) = w\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} r_1 cos\varphi & -r_2 sin\vartheta \\ r_1 sin\varphi & r_2 cos\vartheta \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \tag{1}$$

$$w(x) = w\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$$

Jednotlivé parametry výše uvedených transformací mají následující význam. Úhel φ určuje otočení osy x, v jejímž směru je útvar přeškolován parametrem r_1 a současně υ určuje úhel otočení osy y, v jejímž směru je útvar přeškolován parametrem r_2. Parametry e a f určují translaci útvaru podle jednotlivých (neotočených) os. Z těchto interpretací plyne, jakým způsobem jsou realizována zejména zkosení či zrcadlení. [1]



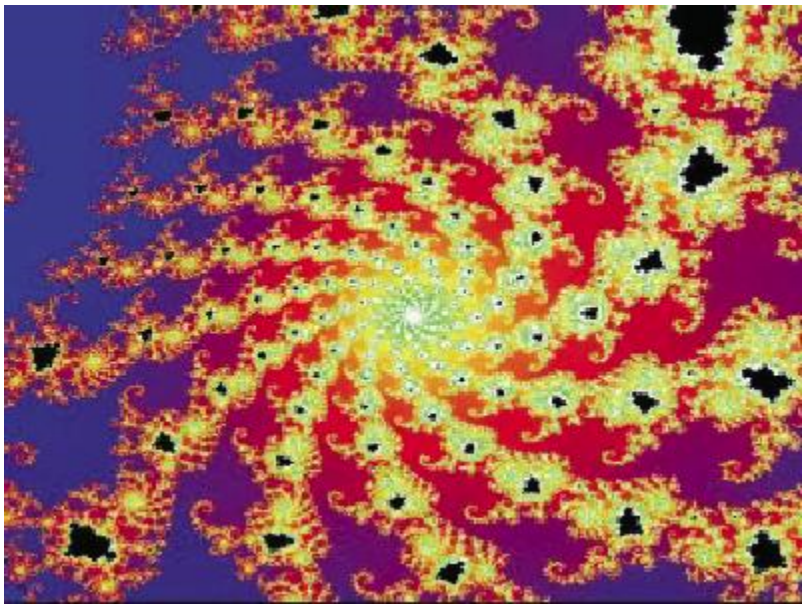
Obr. 6 - Základní operace algoritmu IFS

Opakovaným aplikováním afinní transformace nebo její skupiny se dosáhne toho, že se z vlastního tělesa začne vytvářet fraktální struktura. To, jak tyto afinní transformace budou používány, ovlivní výsledný charakter fraktálu. Jestliže se budou používat všechny transformace rovnoměrně, pak se získá fraktál soběpodobný. Pokud budou používány s ohledem na uživatelsky zadané pravděpodobnosti pro každou transformaci, pak bude výsledkem fraktál soběpříbuzný.

2.2.2 TEA

V případě algoritmu TEA se jedná rovněž o iterační algoritmus, přičemž iterace se provádí jen do uživatelsky definované hranice. TEA (Time Escape Algorithms) se používá v komplexní (definiční) oblasti a pracuje na principu, kdy bere bod po bodu a jako inicializační hodnoty pro start použije komplexní souřadnice jednotlivých bodů. Po této inicializaci proběhne příslušná transformace a vyhodnotí se, zda modul výsledného komplexního čísla přesahuje hranice zadané oblasti. Pokud ne, nově vypočítané komplexní číslo se použije pro iteraci v kroku dalším a opět se kontroluje překročení hranice. To se neustále opakuje, dokud není vyčerpán uživatelsky zadaný počet cyklů. [8]

Pokud trajektorie, vzniklá těmito iteracemi, zůstává uvnitř hranic oblasti, pak se danému startovnímu bodu přiřadí černá barva. Dojde-li k jejich překročení, iterační proces se zastaví a příslušnému startovnímu bodu se přiřadí barva "úměrná" počtu iterací, které byly potřebné pro překročení hranice.



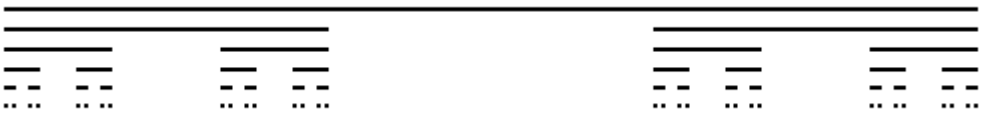
Obr. 7 - Fraktál vytvořený algoritmem TEA

2.3 Známé fraktály a jejich objevitelé

2.3.1 Cantorova množina

Georg Ferdinand Ludwig Philipp Cantor byl významný německý matematik a logik. Kromě matematiky se v pozdějším věku velmi věnoval také teologii, zejména ve vztahu k vlastní práci týkající se nekonečna. Je znám především tím, že teorii množin rozšířil o nekonečná čísla, označovaná jako ordinální a kardinální čísla.

Cantorova množina byla poprvé prezentována roku 1883 a jedná se o nejprozkoumanější a nejjednodušší IFS fraktál, jehož sestavení je velmi triviální záležitostí. Celá množina je tvořena úsečkou na intervalu [0,1]. Konstrukce spočívá ve vynechání prostřední třetiny tohoto intervalu, čímž vzniknou dvě úsečky třetinové délky, na kterých tento proces dále opakujeme.



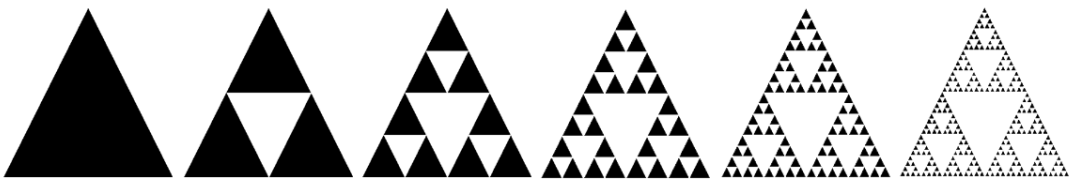
Obr. 8 - Cantorova množina

Cantor tak získal důkaz pro své tvrzení *„množina všech podmnožin dané množiny obsahuje více prvků než původní množina“* a dokázal tak existenci více než jednoho nekonečna.

2.3.2 Sierpinskeho trojúhelník

Wacław Franciszek Sierpiński byl významný polský matematik, který pracoval zejména na teorii množin, teorii čísel a topologii. Je považován za otce klasického fraktálu a znám je především díky veleznámým fraktálům, které nesou jeho jméno.

Základním objektem Sierpinskeho trojúhelníku je nejčastěji rovnoramenný trojúhelník. Sestrojením středních příček dojde k rozdělení na čtyři shodné trojúhelníky, přičemž prostřední z nich je vynechán. Tento postup se dále opakuje na zbylé trojúhelníky. Obdobným způsobem je možné sestrojit také Sierpinského čtverec.

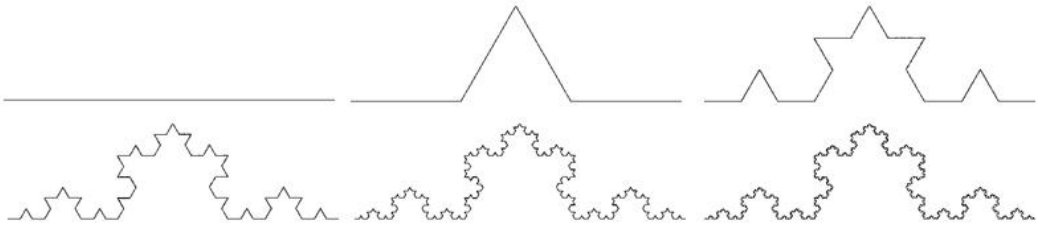


Obr. 9 - Sierpinskeho trojúhelník – 6 iterací

2.3.3 Kochova křivka

Niel Fabian Helge Von Koch byl přední švédský matematik, který roku 1904 představil ve své knize komplikovanou křivku nesoucí později jeho jméno. Jedná se o křivku, pro jejíž inicializaci je zapotřebí tzv. inicializátoru (nejčastěji úsečka) a generátoru (tvar, jímž se nahradí inicializátor). [9]

Klasická Kochova křivka jako generátor používá trojúhelník. Konstrukce probíhá způsobem, kdy je po každé výměně každá strana generátoru v dalším kroku považována za součást inicializátoru, neboli každá rovná strana je nahrazena zmenšenou kopií generátoru.

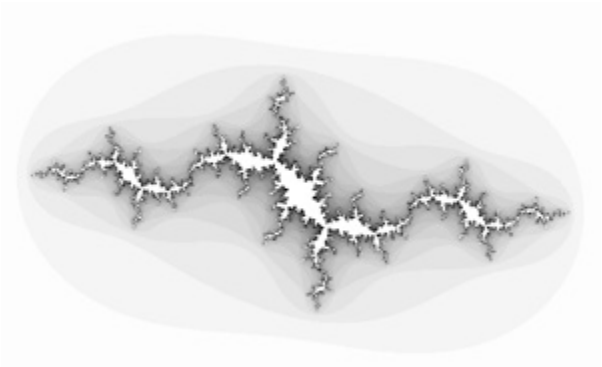


Obr. 10 - Kochova křivka – 6 iterací

2.3.4 Juliovy množiny

Gaston Maurice Julia byl francouzský matematik, který se již v mládí velmi zajímal o matematiku a hudbu. Jeho studia bohužel přerušila první světová válka, která jej doživotně poznamenala ztrátou nosu. Po návratu do Francie publikoval pro několik významných časopisů. Jeho články byly mezi tehdejšími matematiky velmi populární a Gaston Julia byl za svou práci oceněn také francouzskou akademií věd.

Juliova množina je množina všech bodů v komplexní rovině, pro které posloupnost $z_{n+1} = z_n^2 + c$, kde c je libovolné komplexní číslo, nediverguje. Hranice takovéto množiny poté tvoří fraktál.

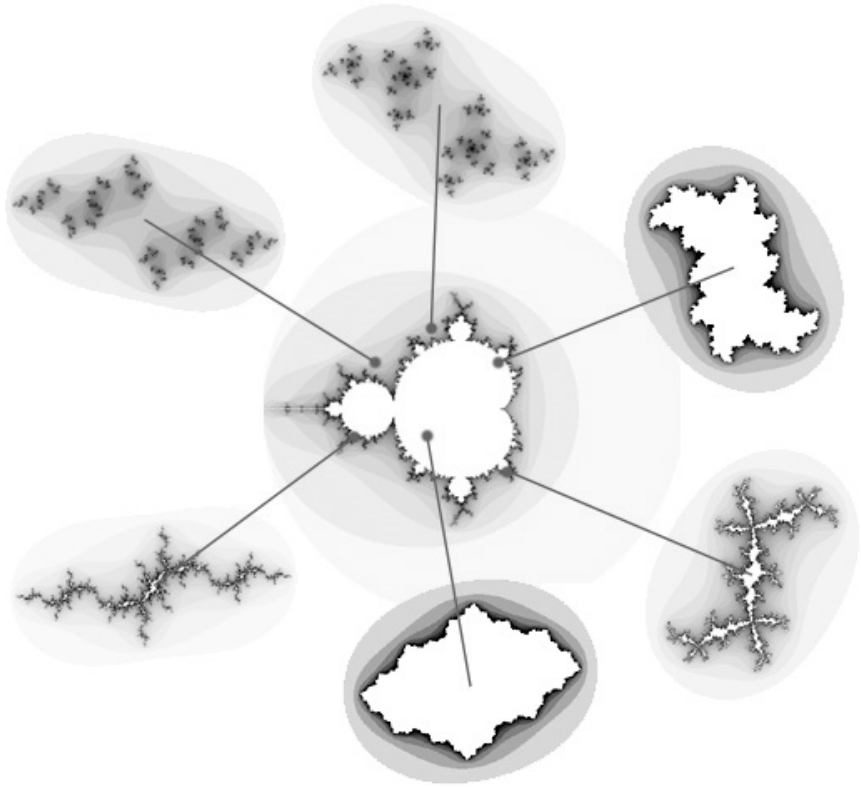


Obr. 11 - Juliova množina pro c nabývající hodnoty -1.15 + 0.27i

2.3.5 Mandelbrotova množina

Zakladatel fraktální geometrie, francouzský matematik Benoît Mandelbrot, je považován za jednoho z nejpopulárnějších matematiků 20. století. Od roku 1951 publikoval práce z oblastí teorie informace, ekonomiky a dynamiky kapalin. V roce 1975 vytvořil pojem fraktál a popsal jeho struktury ve své knize *Les objets fractals, forme, hasard et dimension*, ve které navázal například také na článek *Přírodní dualita statistického rozložení* českého geografa, demografa a statistika Jaromíra Korčáka z roku 1938. V roce 1982 Mandelbrot rozšířil a aktualizoval své myšlenky v knize *The Fractal Geometry of Nature*. Toto vlivné dílo přivedlo fraktály do hlavního proudu profesionální i populární matematiky. [9]

Mandelbrot se snažil vytvořit určitý katalog Juliových množin. I ke svému překvapení nakonec došel k množině, kde ke každému jejímu bodu je přiřazena právě jedna Juliova množina.



Obr. 12 - Mandelbrotova množina

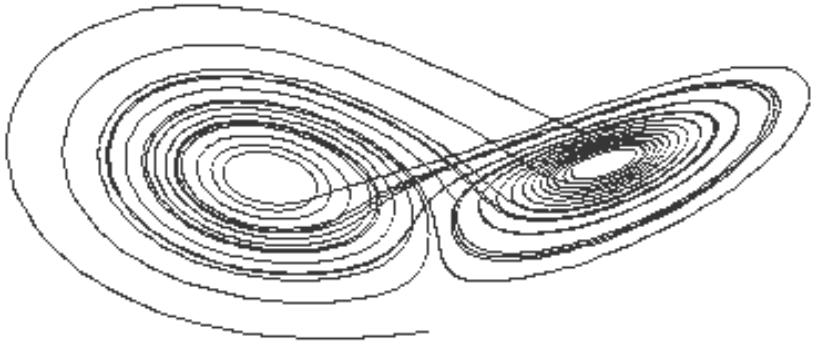
Juliovy množiny jsou přiřazeny tak, že bod Mandelbrotovy množiny je parametrem c pro Juliovu množinu. Vzhled Juliovy množiny závisí na poloze c v Mandelbrotově množině. Pokud c leží uvnitř Mandelbrotovi množiny, pak bude Juliova množina spojitá. Pokud leží mimo Mandelbrotovu množinu, pak bude Juliova množina "rozpadlá" a pokud leží na hranici Mandelbrotovy množiny pak bude Juliova množina na hranici spojitosti.

2.3.6 Lorenzův atraktor

Edward Norton Lorenz byl americký matematik a meteorolog, působící v oblasti teorie chaosu. Vystudoval matematiku na Harvardské univerzitě v Cambridge, během druhé světové války sloužil jako meteorolog pro letecké jednotky USA. Po svém návratu se rozhodl studovat meteorologii na Massachusetts Institute of Technology, kde poté působil mnoho let jako profesor.

Při svém studiu modelů počasí zjistil, že počasí se ne vždy chová podle předpovědi. I malá odchylka v počátečních hodnotách proměnných v jeho počítačovém modelu počasí měla za následek veliké rozdíly v chování počasí. Tato citlivá závislost na počátečních podmínkách se později stala známou jako motýlí efekt. Lorenz prozkoumal tento model z hlediska matematiky a publikoval své závěry v práci z roku 1963, ve které popsal relativně jednoduchý systém rovnic vedoucí k modelu nekonečné složitosti.

Lorenzův atraktor je nelineární trojdimenzionální deterministický dynamický systém odvozený ze zjednodušených rovnic vynucené konvekce v atmosféře. Pro jistou množinu parametrů systém vykazuje chaotické chování a zobrazuje to, co se dnes nazývá podivný atraktor. [1]



Obr. 13 - Lorenzův atraktor

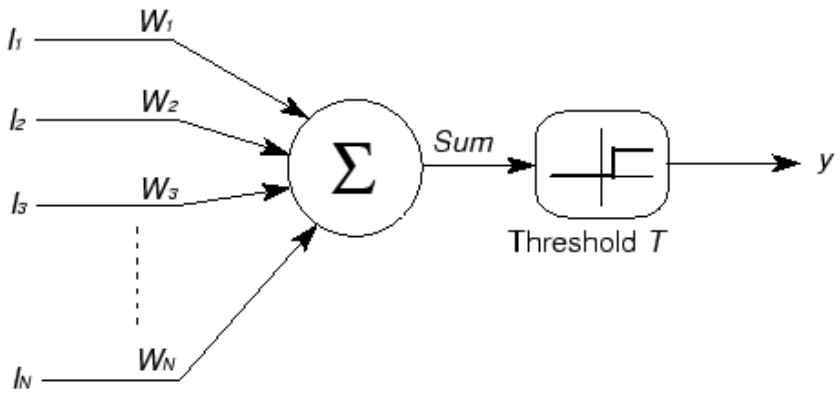
3 NEURONOVÉ SÍTĚ

Tradiční počítač se velice dobře hodí k řešení řady běžných úloh. Je rychlý a dělá přesně to, co mu řekneme. Naneštěstí nám však nemůže pomoci v situaci, když sami docela přesně nerozumíte problému, který chceme řešit. Navíc standardní algoritmy nepracují dobře s porušenými nebo nekompletními daty. Jenže v reálném světě je právě tento druh dat často tím jediným dostupným. Řešením je tedy použití umělé neuronové sítě, výpočetního systému, který se může učit sám. [10]

3.1 Historie

Historie vzniku vývoje neuronových sítí spadá do první poloviny 20. Století, kdy Američan Warren Sturgis McCulloch publikoval vůbec první práci o neuronech a jejich modelech. Ve 40. letech pak společně se svým studentem (W. Pitts) vypracoval model neuronu, který se prakticky používá dodnes. Na základě jejich výsledků vytvořil v roce 1958 další Americký vědec Frank Rosenblatt první funkční perceptronovou síť. Byla však schopna řešit pouze problémy, které byly lineárně reparabilní. Na tento nedostatek bylo roku 1969 upozorňováno v knize *Perceptron*, díky čemuž přestal být o neuronové sítě zájem. [11]

V polovině 80. let se však naštěstí našlo několik průkopníků, kteří pozvedli princip neuronových sítí opět do popředí. Začaly vycházet publikace věnující se tomuto oboru, byly vynalezeny nové typy sítí. Dnes již můžeme říci, že význam a uplatnění neuronových sítí od té doby rostl společně s rozvojem PC.

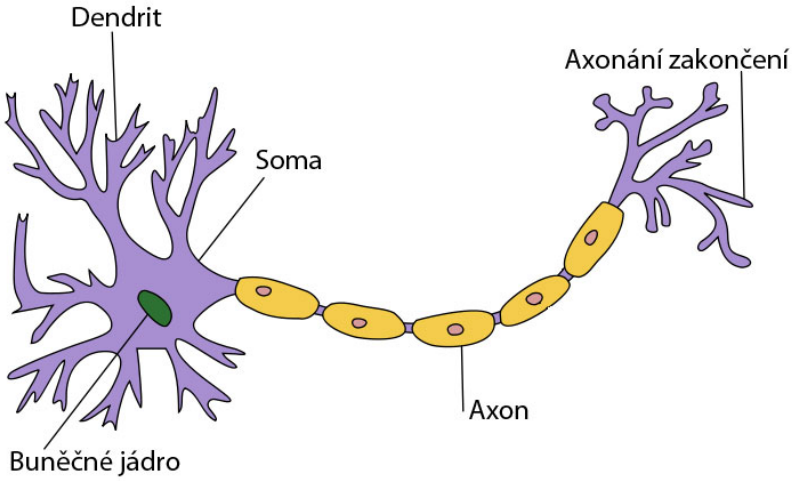


Obr. 14 - McCulloch-Pittsův model neuronu

3.2 Biologická inspirace

Umělé neuronové sítě jsou velmi hrubou a přibližnou napodobeninou základních topologií vyskytujících se v lidském mozku. Tam se nachází přibližně $10^{13\text{-}15}$ neuronů, přičemž jich každý den odumře asi 10.000, což za celkový život člověka činí zhruba 0.00025% celkového počtu neuronů. [12]

Základním stavebním prvkem nervové soustavy živých organismů je tedy nervová buňka, neuron. Jde o samostatnou živou buňku specializovanou na co nejúčelnější zpracování, uchování a přenos informací. Fyziologové objevili a popsali v živých organismech řadu druhů neuronů, všechny ale mají stejnou strukturu. Základem neuronu je tedy tělo s buněčným jádrem zvané *soma*, ze kterého vychází jediný, avšak aktivní a rozvětvený výstup zvaný *axon*. Ten se na svém vzdálenějším konci větví a napojuje (chemická synapse) se zvláštními přísavnými tělísky na pasivní vlákna jiných neuronů. Těmito vstupy jsou *dendrity*, které působí jako vstupní kanály neuronů. [12] [13]



Obr. 15 - Bilogický neuron

Technický neuron je tedy primitivním modelem této specializované jednotky. Má velmi jednoduchý princip: ohodnotí všechny své vstupy jejich vahami, takto získané hodnoty sečte a vzniklou hodnotu dosadí do přenosové funkce daného neuronu. Výstup z funkce je pak také výstupem neuronu, který dále může sloužit jako vstup do dalších neuronů.

3.3 Klasifikace sítí

Propojením určitých vstupů a výstupů neuronů vzniká neuronová síť. Ne všechny neuronové sítě jsou však stejné. Podle několika kritérií je můžeme rozdělit do více samostatných skupin.

Podle počtu vrstev rozeznáváme sítě jednovrstvé a vícevrstvé. Sítě s jednou či dvěma vrstvami bývají většinou speciální sítě, které mají svůj speciální učící algoritmus a topologii. Tří a vícevrstvé sítě pak nejčastěji využívají klasického učícího algoritmu Blackpropagation. Pro topologii sítí také obvykle platí pravidlo, že každý neuron bývá spojen s každým neuronem ve vyšší vrstvě. Každý takový spoj je pak ohodnocen vahami, které mohou nabývat různých hodnot a vyjadřují, jaký význam tento spoj pro neuron má.

Podle algoritmu učení dělíme sítě na učení s učitelem a bez učitele. V prvním případě to znamená, že síť se snaží přizpůsobit svou odezvu na vstup informace tak, aby se její momentální výstup co nejvíce podobal požadovanému originálu. Učitelem je v takovém případě myšlen princip předložení vzoru a vyžadování jeho zopakování. Existují však i případy sítí, u kterých dochází k učení bez učitele, což je proces, ve kterém síť vychází z informací, které jsou obsaženy ve vstupních vektorech. U těchto sítí tedy nejsou stanoveny počáteční podmínky učitele.

Neuronové sítě můžeme klasifikovat podle stylu jejich učení a to na sítě deterministické a stochastické. Styl učení v podstatě znamená, jakým způsobem se přistupuje ke zjištění vah dané sítě. V případě, že se jedná o zjištění výpočtem, mluvíme o deterministickém učení. Pokud jsou však váhy získávány pomocí generátoru náhodných čísel, mluvíme o stochastickém stylu učení. Tento způsob se však obvykle používá jen při startu sítě. [11]

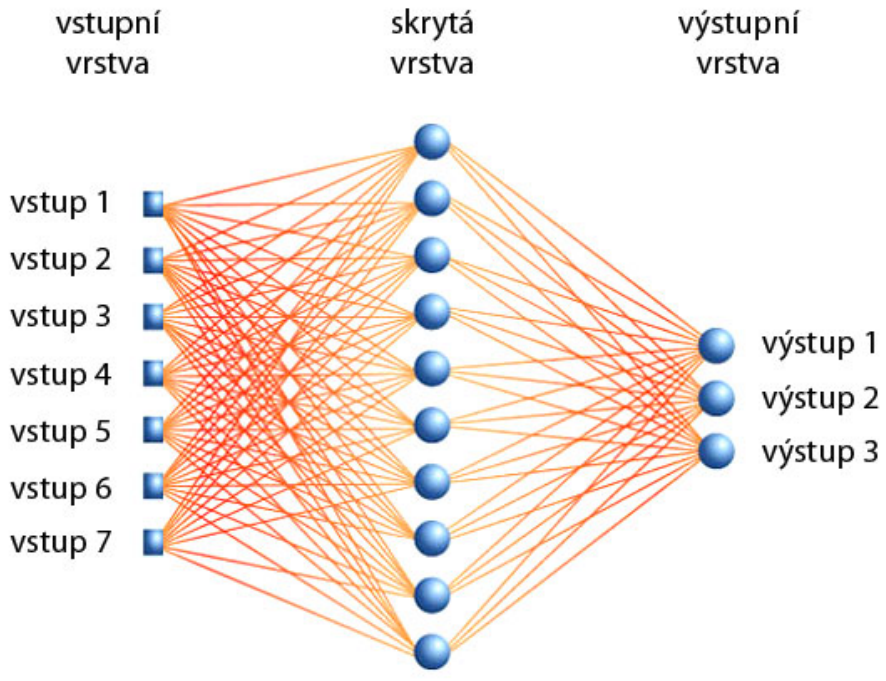
3.4 Obecná neuronová síť

V technickém slova smyslu si tedy můžeme neuronovou síť představit jako síť složenou z několika vrstev, přičemž každá taková vrstva se skládá z „libovolného“ počtu neuronů. Libovolného však jen teoreticky, jelikož jsme stále limitování technickými podmínkami.

Problém vhodného počtu vrstev u vícevrstvých sítí byl vyřešen ve druhé polovině 20. století, kdy Kolmogorovův teorém o řešení třináctého Hilbertova problému aplikovaného na neuronové sítě vedl k poznatku, že k aproximaci libovolné funkce neuronovou sítí stačí, aby tato měla minimálně tři vrstvy s odpovídajícím počtem neuronů v každé vrstvě. [13]

U vícevrstvých sítí platí, že první vrstva je vždy větvící, což znamená, neurony ve vstupní vrstvě pouze distribuují vstupní hodnoty do další vrstvy. Vzhledem k tomu, že se jedná obecně o vícebodový vstup do sítě, mluvíme o vstupních vektorech informací. Každý ze vstupů má navíc přiřazen tzv. váhu, což je bezrozměrné číslo, které udává, jaký význam daný vstup pro příslušný neuron má.

Učící schopnost neuronových sítí spočívá právě v možnosti měnit tyto váhy v síti podle vhodných algoritmů na rozdíl od sítí biologických, kde je schopnost se učit založena na možnosti tvorby nových spojů mezi neurony. Fyzicky jsou si tedy tyto dvě sítě založeny na rozdílných principech učení, logicky však ne.



Obr. 16 - Neuronová síť

3.5 Fungování sítě

Každou nově vytvořenou, ale také jakoukoliv nenaučenou neuronovou síť lze považovat za síť, která nic neumí – neumí rozeznávat, neumí klasifikovat. Aby bylo možné danou síť používat, musí být tato naučena. Z toho důvodu byly vyvinuty algoritmy, pomocí kterých se příslušná síť dokáže naučit na danou množinu informací.

Tento algoritmus se zpravidla dělí na dvě fáze, a sice fázi aktivační a fázi adaptační. Ty ke své činnosti potřebují trénovací množinu, což je skupina vektorů obsahujících informace o daném problému pro učení. V případě učení s učitelem se jedná o dvojice vektorů vstup/výstup, v případě učení bez učitele pak trénovací množina obsahuje pouze vektor vstupů. Cyklickým střídáním obou fází dochází k samotnému učení.

Aktivační fáze (nazývána také jako vybavovací) je proces, při kterém se předložený vektor informací na vstup sítě přepočítá přes všechny spoje včetně jejich ohodnocení vahami až po výstup, kde se objeví odezva sítě na tento vektor ve formě výstupního vektoru. Při učení se tento vektor porovnává s vektorem originálním, přičemž rozdíl mezi těmito vektory se označuje jako lokální chyba.

Adaptační fáze (také označovaná jako učící) je pak proces, při kterém je minimalizována tato lokální chyba sítě tak, že se přepočítávají váhy jednotlivých spojů směrem z výstupu na vstup za účelem co největší podobnosti výstupní odezvy s originálním vektorem.

Takto se obě fáze střídají, přičemž jednotlivé lokální odchylky se postupně sčítají. Po proběhnutí celé trénovací množiny je hotova jedna epocha. Suma lokálních odchylek vzniklá během této epochy je označována jako globální odchylka. V případě, že tato nabývá hodnoty menší než je povolená odchylka, proces učení končí. [1]

Proces učení neuronové sítě tedy není nic jiného, než přelévání informací ze vstupu na výstup a naopak. Při učení se ve vstupním prostoru vytvářejí shluky bodů, které představují jednotlivé členy tříd, přičemž každý shluk reprezentuje jinou třídu. To zda se síť naučí správným odezvám na dané předměty, závisí na více okolnostech – na množství vektorů, jejich velikosti, topologii sítě, odlišnostech jednotlivých tříd, trénovací množině a jiných. Samotná schopnost přiřazení vstupů jednotlivým třídám je založena na přepočítávání vzdálenosti daného členu od členů již přiřazených.

4 MATHEMATICA

Mathematica je počítačový program široce používaný ve vědeckých, technických a matematických kruzích. Program byl původně vytvořen Stephenem Wolframem a následně vyvíjen týmem matematiků a programátorů, který vytvořil a vede. Je prodáván firmou Wolfram Research se sídlem v Champaign, Illinois, která byla založena v roce 1987 a dnes je považována za jednoho z průkopníků v oblasti výpočetní vědy a zároveň za jednu ze světově nejuznávanějších softwarových společností věnujících se výrobě matematických programů.

4.1 Přednosti programu

- Notebookový dokumentační systém - Mathematica zajišťuje kompletní technický dokumentační systém zahrnující sazbu matematických výrazů, formátovaného textu, zvuku, grafiky, animací a hypertextových odkazů.
- Programovací jazyk – Mathematica poskytuje silné programové vývojové prostředí. Programový kód Mathematica dokáže odrážet specifičnost problému, což jej dělá kratším a snadněji čitelným. Tato ojedinělá pružnost činní přechod z jiného programovacího jazyka jednodušším a efektivním a tak není ani pro dříve neprogramující uživatele obtížně brzy začít tvořit kvalitní programy.
- Interaktivní nápověda – Nápověda v software Mathematica zahrnuje kompletní dokumentaci pro všechny funkce v tomto software obsažené. Navíc je vybavena pokročilými vyhledávacími schopnostmi včetně množství hypertextových odkazů. Nápověda také obsahuje interaktivní příklady, které demonstrují použití funkcí, jejich hlavní schopnosti a nejlepší způsob, jak je využít.
- Grafika - Mathematica obsahuje velké množství vestavěných grafických vzorů pro vizualizaci výsledků. Kromě 2D a 3D grafiky nabízí uživateli také vrstevnicové grafy, grafy hustoty a navíc také specializované ekonomické či statistické grafy.
- Symbolické a numerické výpočty - Každá funkce, která je v prostředí Mathematica zavedena může pracovat jak s numerickými tak se symbolickými vstupy. [14]

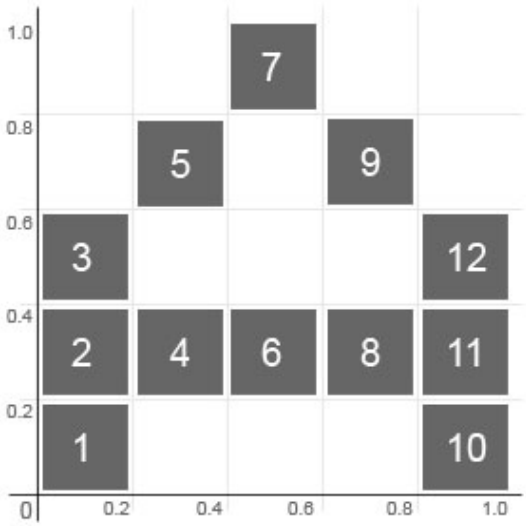
II. PRAKTICKÁ ČÁST

5 FRAKTÁLNÍ KRYPTOLOGIE

5.1 Návrh systému šifrování

Fraktální geometrii lze pro šifrování využít tím způsobem, že jednotlivá písmena abecedy popíšeme několika afinními transformacemi. Pro tyto účely je nejvhodnějším tělesem čtverec, pomocí kterého lze použitím několika transformací ve čtvercové síti přiměřené velikosti vytvářet grafické podoby jednotlivých písmen.

Z provedených pokusů jsem usoudil, že pro účely této práce bude nejlepší volbou síť o rozměru 5x5 prvků. Pokud tuto graficky znázorníme na intervalu [0,1] v obou osách, získáme celkem 25 polí o velikosti každého z nich 0.2x0.2. Do takto vzniklé sítě je pak možné formou menších čtverců zakreslit jednotlivá písmena.



Obr. 17 - Grafické znázornění písmene A

Grafická ukázka zobrazuje písmeno A za využití 12 čtverců, vyobrazení zbylých písmen je součástí přílohy PI. Pro uvedené písmeno platí dle matice transformace (1) následující tabulka parametrů *e* a *f*:

čtverec	1	2	3	4	5	6	7	8	9	10	11	12
parametr e	0	0	0	0.2	0.2	0.4	0.4	0.6	0.6	0.8	0.8	0.8
parametr f	0	0.2	0.4	0.2	0.6	0.2	0.8	0.2	0.6	0	0.2	0.4

Tab. 1 - Tabulka parametrů pro písmeno A

Tyto parametry by již nyní bylo možné odeslat jako zašifrovanou zprávu a to ve formě vektoru {e f}:

{0 0 0 0.2 0.2 0.4 0.4 0.6 0.6 0.8 0.8 0.8 0 0.2 0.4 0.3 0.6 0.2 0.6 0 0.2 0.4} (2)

V případě jednoho písmene by nebyl problém jej na straně příjemce dešifrovat, v případě více různých písmen by to však problém již byl. Každé písmeno je totiž ve své grafické podobě vyjádřeno jiným počtem čtverců a tedy i jiným počtem parametrů. Před odesíláním šifrované zprávy je tedy nutné zajistit stejný počet parametrů pro všechna písmena. Nejvíce parametry je popsáno písmeno B. To se ve své grafické podobě skládá z 16 čtverců (viz. PI) a je tak popsáno celkem 32 parametry. Každé z písmen je tak potřeba doplnit právě na tento počet parametrů. Pro doplnění jsem zvolil číslo 0 a v případě (2) vznikne následující vektor:

{0 0 0 0.2 0.2 0.4 0.4 0.6 0.6 0.8 0.8 0.8 0 0 0 0 0 0.2 0.4 0.3 0.6 0.2 0.6 0 0.2 0.4 0 0 0 0} (3)

Nyní by bylo možné odesílat také zprávy o velkém počtu písmen, které jsou ve své šifrované podobě zapsány jedním dlouhým vektorem hodnot. Tento vektor však stále obsahuje čísla ve formě ne-příliš vhodné pro odesílání a tak je potřeba jej dále upravit.

V první řadě jsem každé číslo vektoru vynásobil 10. Tato hodnota byla zvolena především pro zajištění celočíselnosti. Následně jsem ještě ke každé hodnotě ve vektoru přičetl 1. Tuto úpravu jsem zvolil za účelem zjednodušení odesílání šifrovaného textu. Vektor v tuto chvíli obsahuje pouze čísla od 1 do 9 a jednoduchou úpravou tak z vektoru mohu vytvořit jedno velké číslo, které je pro odeslání mnohem vhodnější. V případě, kdy by nedošlo k přičtení, mohla by na přední pozici vektoru zůstat 0, která by při převedení vektoru na číslo zmizela a celou šifru tak znehodnotila. Vektor (3) tak po úpravě vypadá následovně:

11133557799911111353739371351111 (4)

Na straně příjemce je pak postup opačný. Číslo je rozděleno po 32 znacích a takto vzniklé hodnoty jsou zapsány do vektoru. Každá hodnota ve vektoru je upravena odečtením 1 a vydělením 10. Odstraněním přebytečných parametrů pak získá příjemce opět původní hodnoty v rozmezí [0,1]. Tyto pak použije pro vyobrazení jednotlivých písmen ve formě fraktálního textu.

5.2 Aplikace návrhu

Vytvořený návrh bylo potřeba následně aplikovat a otestovat. Pro tyto účely jsem se rozhodl využít software Mathematica, se kterým mám již zkušenosti z jiných předmětů vyučovaných na naší fakultě. Připravil jsem tedy 2 samotné programy – jeden pro šifrování, druhý pro dešifrování. Zdrojové kódy obou programů jsou součástí přílohy této práce (PII a PIII), v této části jsou pouze podrobněji popsány důležité části těchto kódů.

5.2.1 Fraktální šifrování

Celý program je uzavřen v jedné funkci `Module`, která nemá nadefinovanou žádnou lokální proměnnou. Pomocí `InputString` je nejprve načten text určený k zašifrování. Tento text může být tvořen libovolnými ASCII znaky včetně písmen české abecedy. Následuje převedení do vektoru, malá písmena jsou převedena na velká, specifické české znaky jsou nahrazeny znaky anglické abecedy. Je určen počet znaků.

```
sifrovani[] := Module[{},
      text = InputString["Zadejte text urceny k zasifrovani"];
      a = ToUpperCase[Characters[text]];
      a = StringReplace[a, {"Á" -> "A", "Č" -> "C", … ,"ž" -> "Z];
      b = StringLength[text];
```

Následuje cyklus beroucí znak po znaku. Pakliže nalezne shodu (tj. znak anglické abecedy nebo mezera), zapíše do předpřipravené matice určené hodnoty koeficientů. Ostatní znaky přehlíží. Zároveň dochází k doplnění jednotlivých částí matice na 32 hodnot.

```
For[i = 1, i < b + 1, i++,
      Switch[a[[i]],
        "A", c = {{0, 0,…, 0.8,0.8}, {0, 0.2,…, 0.2, 0.4}};d = 12,
        "B", c = {{0, 0,…, 0.8, 0.8}, {0, 0.2,…, 0.2, 0.6}}; d = 16,
        …
        …
        …
        " ", c = {{}, {}}; d = 0,(* mezera*)
        _, c = {{}, {}}; d = 16(* vsechny ostatni znaky *)
      ];

      If[d < 16,
        For[j = 1, j < 17 - d, j++,
        AppendTo[c[[1]], 0];
        AppendTo[c[[2]], 0];
        ]
      ];
      If[h == 1, h = 0,
        e = AppendTo[e, c[[1]]];
        e = AppendTo[e, c[[2]]];
      ];
];
```

Takto připravenou matici je následně možné převést na vektor čísel a ten upravit do podoby vhodné pro odeslání.

```
f = Flatten[e];
f = Round[(f*10)+1];

For[i = 1, i < Length[f] + 1, i++,
 g = g + f[[i]]*10^(Length[f] - i);
];

Print["Zasifrovany text: ", g]
```

Program je poté možné volat příkazem `sifrovani[]`

5.2.2 Fraktální dešifrování

Před samotným dešifrováním je nutné si nejprve příkazem `Needs` načíst 2 důležité knihovny určené pro práci s fraktály.

```
Needs["ProgrammingInMathematica`AffineMaps`"]
Needs["ProgrammingInMathematica`IFS`"]
```

Poté již následuje samotný program. Opět se jedná o funkci `Module`. Zašifrovaný text ve formě čísla je nyní načten pouze funkcí `Input`. Číslo je okamžitě rozděleno do vektoru a jednotlivé hodnoty jsou převedeny do správného tvaru na intervalu [0,1].

```
desifrovani[] := Module[{},
      text = Input["Zadejte zasifrovany text"];

      a = {}; c = {{}, {}};
      i = 1;
      While[i == 1,
        a = PrependTo[a, (Mod[text, 10] - 1)/10];
        text = Quotient[text, 10];
        If[text < 1, i = 0]
      ];
      b = Length[a];
```

Následuje rozčlenění na jednotlivé subvektory o velikosti 32 znaků, přičemž je myšleno na možnost ztráty několika posledních znaků.

```
      While[b > 31,
        c[[1]] = AppendTo[c[[1]], Take[a, 16]];
        c[[2]] = AppendTo[c[[2]], Take[a, {17, 32}]];
        a = Drop[a, 32];
        b = b - 32;
        b1 = b1 + 1;
      ];
```

Během šifrování jsou vektory *e* a *f* většiny písmen doplněny na celkových 32 hodnot číslem 0. Tento doplněk je nyní odstraněn a to procházením jednotlivých subvektorů směrem od jejich zadních pozic. Žádné z písmen totiž nemá na koci svých originálních vektorů dvě hodnoty 0. Výjimkou je mezera, za kterou jsou následně považovány všechny znaky, které dosáhnou 15. iterace.

```
i=1;f={};g={};
While[i<Dimensions[c][[2]]+1,
  d=c[[1]][[i]]; e=c[[2]][[i]];
  For[j=16,j>0,j--,
   If[d[[j]]!=0 ,
    f=AppendTo[f,Take[d,j]]; g=AppendTo[g,Take[e,j]];
    j=0,
    If[j==1,
       f=AppendTo[f,{0.9}]; g=AppendTo[g,{0.9}];
       j=0;
    ];
   ];
  ];
  i++;
];
```

Následuje úprava jednotlivých vektorů do podoby určené k vykreslení. Vzniká vektor f1 obsahující jednotlivé subvektory, které určují zmenšení základního čtverce (0.195 zvoleno z důvodu viditelnosti mezer), jeho náklon a posun vůči počátečnímu bodu (e,f). Zároveň dochází k určení maximálního počtu písmen, která budou na jednom řádku.

```
For[i = 1, i < b1 + 1, i++,
  h = f[[i]];k = g[[i]];
  For[j = 1, j < Length[h] + 1, j++,
   If[h[[j]] < 0.9,
     AppendTo[f1, {0.195, 0, 0, 0.195, h[[j]] + l, k[[j]] + m}],
     l = l - 0.5;
   ];
  ];
  l = l + 1.1;
  If[l > 21,
   l = 0;m = m - 1.5;
  ];
];
```

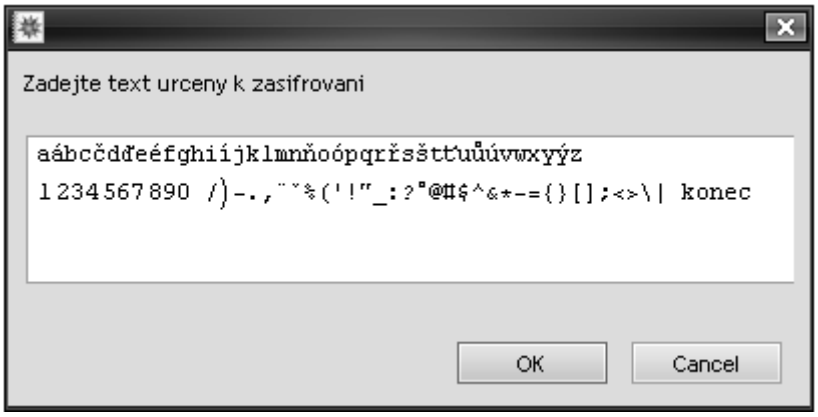
Program končí samotným vykreslením jednotlivých fraktálů. Jako základní tvar je určen čtverec 1x1, počáteční bod se nachází na souřadnicích [0,0]. Spuštění: `desifrovani[]`

```
ctvr=Show[Graphics[{RGBColor[0,0,0],Polygon[
      {{0,0},{1,0},{1,1},{0,1}}]}], Axes->False,Frame->False,
      PlotRange->All];
AM[{a_, b_, c_, d_, e_, f_}] :=map[{{a,-b,e},{c,d,f}}];
f2=AM/@f1;ifs0=IFS[f2];

Print["Desifrovany text:"]; Show[Nest[ifs0,ctvr,1]]
```

5.3 Otestování návrhu

Jako první jsem program otestoval na většině znaků ASCII tabulky:



Obr. 18 - Zadání testovacího ASCII textu

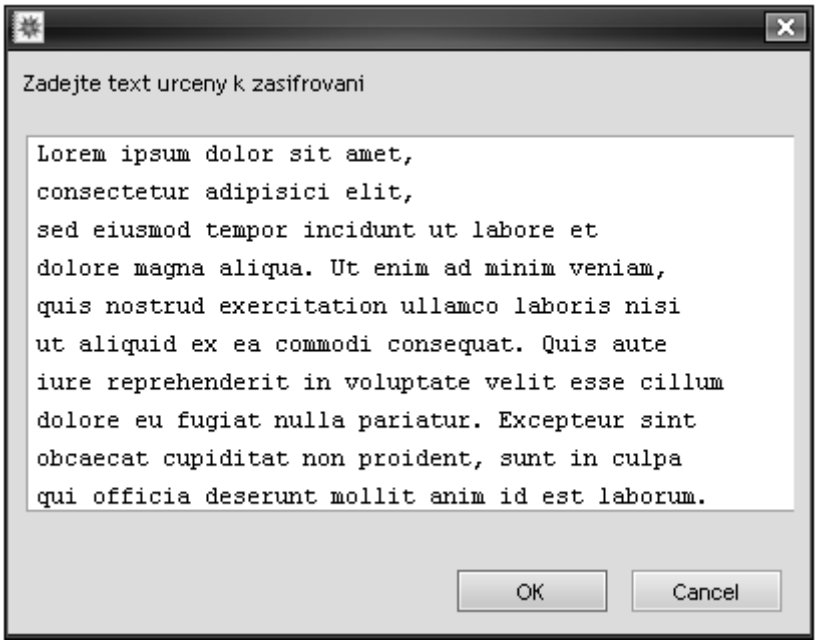
Šifrovaným textem na výstupu bylo následující číslo:

```
11133557799911111353739371351111111335577999111113537393713511111111133335
55777991357915915915937111335577991111135719191937111111113355577991111135
71919193711111111113355779991113579191919357111111133557799911135791919193
35711111113335557799113579159159191911111333555779911357915915919191111111
13355791111113579595999111111113355577799111357191491493711111111357999991
11113579555135791111133555557799111191913579191911111335555577991111919135
57919191111357999911111111311135791111111111111355779911111357954637191111
11111135791111111135791111111111111111357999991111357975713579111111113357
99999111135797531357911111111357999991111357975313579111111335577999111135
57191919357111111133557799911113571919193571111111113355779111113579595959
97111111133557799911113571919391571111111113355577799113579595935925917111
11335557779911357959359259171111333555777999116815915915924911113335557777
99911681591591592491135555579111111199135799911111111355555791111111991357
79991111111111135799991111135791113579111111111357999911111357911135791111
11111135799991111135791113579111111111357999111111157931357911111111111357
99991111135791313579111111133577991111111193753719111111113555791111111111
97135791111111111355579111111111971357911111111111333555777991111913915911
79191111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111135577991111135795463719111111133557799
91111357191919357111111111357999991111357975313579111111113335557799113571
91591591919111133557799111113571919193711111
```

Po dosazení čísla do programu pro dešifrování, jsem získal následující fraktální text:

Obr. 19 - Dešifrovaná verze testovacího ASCII textu

Pro druhý pokus jsem zvolil standardní pseudolatinský text užívaný v grafickém designu a navrhování jako demonstrativní výplňový text, známý jako Lorem ipsum.



Obr. 20 - Zadání testovacího textu Lorem ipsum

Tento text o 429 znacích je ve své šifrované podobě popsán 13472 hodnotami a výstupem je tedy skutečně velké číslo. Po jeho dosazení do programu pro dešifrování je výsledek tento (pro účel této ukázky byl upraven počet znaků na řádek):

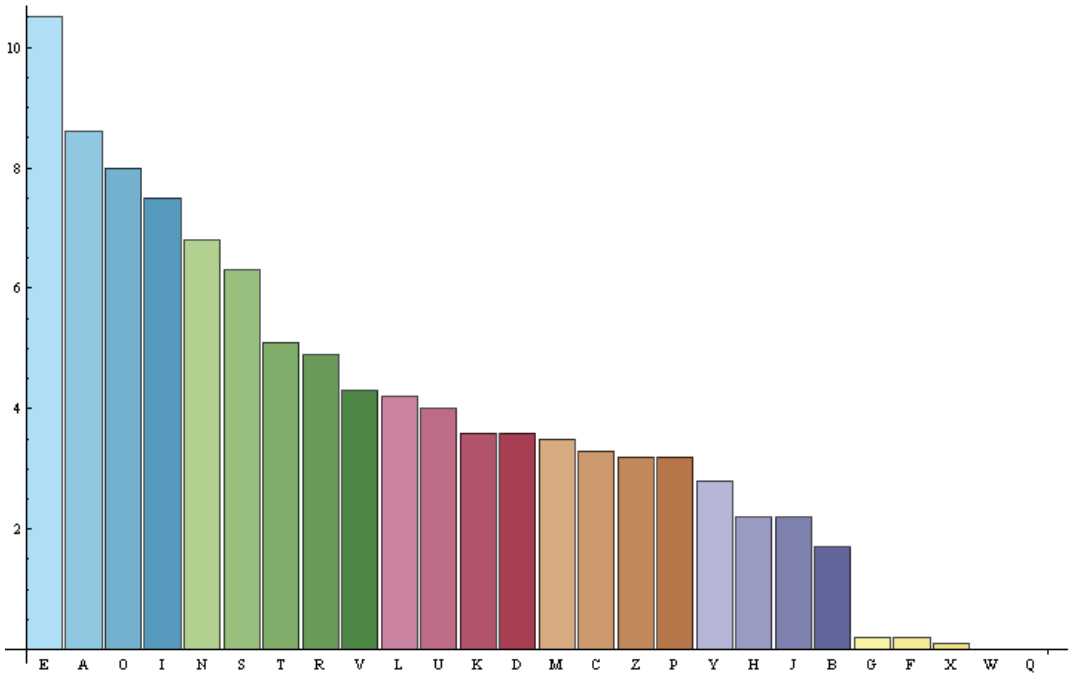
```
LOREM IPSUM DOLOR SIT AMET CONSECTETUR A
DIPISICI ELIT SED EIUSMOD TEMPOR INCIDU
NT UT LABORE ET DOLORE MAGNA ALIQUA UT EN
IM AD MINIM VENIAM QUIS NOSTRUD EXERCITA
TION ULLAMCO LABORIS NISI UT ALIQUID EX
EA COMMODI CONSEQUAT QUIS AUTE IURE REPR
EHENDERIT IN VOLUPTATE VELIT ESSE CILLU
M DOLORE EU FUGIAT NULLA PARIATUR EXCEPT
EUR SINT OBCAECAT CUPIDITAT NON PROIDEN
T SUNT IN CULPA QUI OFFICIA DESERUNT MOL
LIT ANIM ID EST LABORUM
```

Obr. 21 - Dešifrovaná verze testovacího textu Lorem ipsum

5.3.1 Frekvenční analýza

Frekvenční analýza je dešifrovací metoda vycházející ze skutečnosti, že každé písmeno se v běžném textu vyskytuje různě často. Každý si jistě dokáže představit, že v jakémkoli delším českém textu bude výskyt písmena A mnohem častější než písmena X. [15]

Následující graf zobrazuje procentuální výskyt jednotlivých znaků v českém textu:

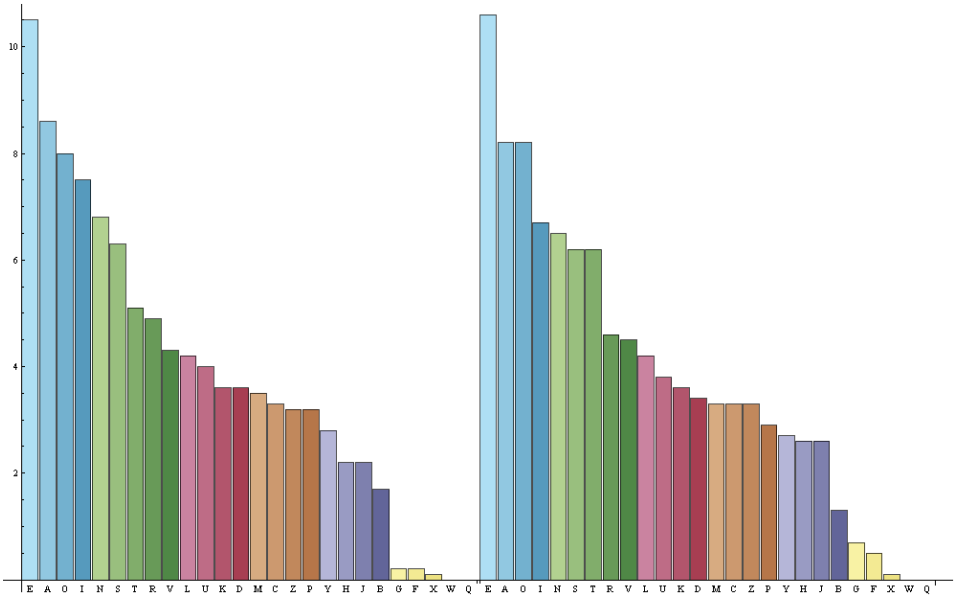


Obr. 22 - Statická charakteristika českého jazyka

Pro test prolomení šifry metodou frekvenční analýzy jsem vybral text dostupný na [15]. Jedná se celkem o 2150 znaků, které jsou v šifrované podobě popsána 55.904 čísly. Testovaný text začíná slovy „Všech deset příkladů“ a končí před tabulkou „výskyt znaků takový:“.

Jelikož není v analýze zmiňována mezera, upravil jsem šifrovací program tak, aby šifra obsahovala pouze písmena anglické abecedy. Stejně tak jsem v Mathematice vytvořil program, který pro vložené číslo určí všechny jeho dělitele. Pro každého z nich poté číslo rozdělí na vektory, pro které určí jednotlivé statické charakteristiky. Během své činnosti postupně vyloučí nevhodné možnosti (více možných vektorů než 26), až na svém konci vypíše pouze ty vhodné. Těchto je často velmi málo a během všech testů se mezi výsledky objevovalo rozdělení po 32 znacích.

I přes fakt, že analýza vybere jako vhodné rozdělení po 32 znacích, není následné rozšifrování stále otázkou chvilky. Vytvořený program pro analýzu obsahuje druhou část, která zajistí náhradu vektorů vhodnými písmeny dle dříve uvedené statické charakteristiky českého jazyka, přičemž vzniklý text nedosahuje ani 50% přesnosti s originálem.

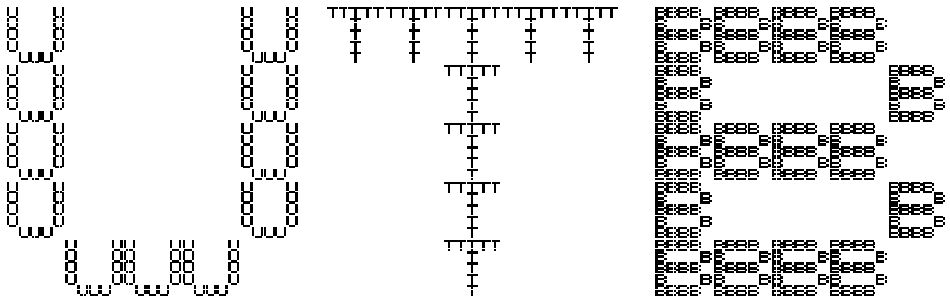


Obr. 23 - Porovnání statických charakteristik

vlevo český jazyk, vpravo testovaný text

5.3.2 Možná vylepšení

Program tak jak je napsán plně splňuje zadání práce. Jistě by jej však šlo dále rozvíjet. Bylo by možné do programu přidat další ASCII znaky či číslice. Stejně tak by bylo možné dále ovlivňovat odesílaný text (opakování jednotlivých písmen). Na straně příjemce by bylo zase možné program upravit do podoby, kdy by vypisoval skutečný textový fraktál, tedy písmena složená ze sebe samých. Tyto a další úpravy jsou však již na samotných uživatelích programu.



Obr. 24 - Textový fraktál

6 NEUROFRAKTÁLNÍ KRYPTOLOGIE

6.1 Návrh systému šifrování

Aby mohla být využita neuronová síť k zašifrování textu, musí být písmena vyjádřena pomocí čísel. Buď se jednotlivým písmenům přiřadí přímo vektory čísel (potom se jedná o neuronové šifrování), nebo můžeme využít numerický popis písmen z šifrování pomocí fraktální geometrie, přičemž se bude jednat o dvojité zašifrování. Text bude nejprve zašifrován pomocí fraktální geometrie a poté naučením sítě na tuto matici, a to tak, že na vstupu budou předem dohodnuté vektory skládající se z 0 a 1 a k nim na výstupu budou přiřazeny vektory afinních transformací. Jako šifrovaný text potom budou sloužit váhy a prahy jednotlivých vrstev neuronové sítě a také informace o vstupních vektorech, bez kterých by nebylo možné dešifrovat text.

Obr. 25 - Princip neurofraktálního šifrování

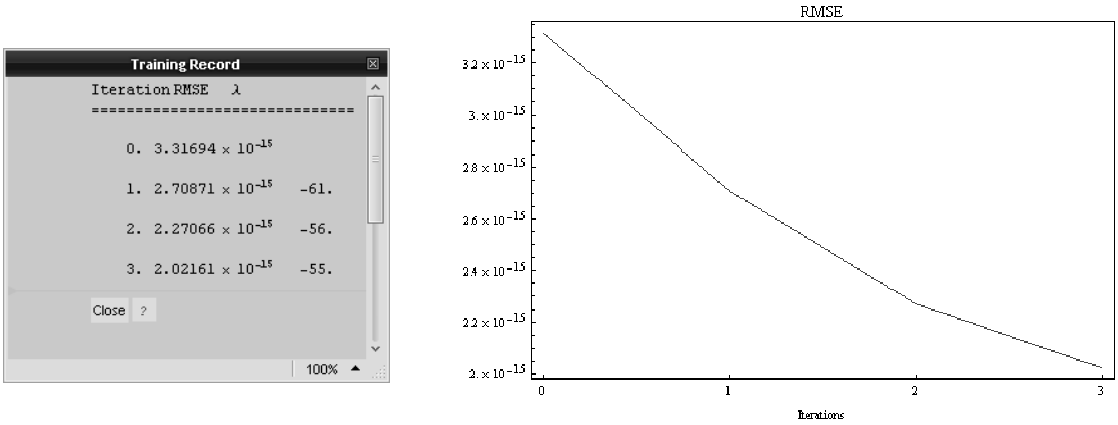
Základní princip neurofraktální kryptologie je tedy objasněn. Během návrhu systému je však nutné vyřešit několik problémů. Prvním z nich je velikost vektorů, pomocí kterých budou jednotlivá písmena předložena neuronové síti. K dispozici mám celkem 26 písmen plus mezeru. Číslo 27 jde v binárním kódu vyjádřit jako 11011. Mohlo by se tedy zdát, že budou postačovat vektory, jejichž velikost je 5. Toto však není pravda. Jak bude dále uvedeno, během učení sítě je zobrazována globální chyba, která v případě použití takovýchto vektorů dosahuje pro nás nepoužitelných hodnot. Provedl jsem tedy několik pokusů (vektor značí velikost vektoru):

vektor	chyba	vektor	chyba	vektor	chyba
9	1.6697	*14*	1.3260	*19*	0.7169
10	1.4675	*15*	1.1025	*20*	0.6614
11	1.5096	*16*	1.0636	*21*	0.4443
12	1.2853	*17*	0.7568	*22*	0.4142
13	1.2458	*18*	0.7214	*23*	$4.58 * 10^{-14}$

Tab. 2 - Přehled výsledků pokusu pro určení velikosti vektoru

Tabulka na předešlé straně naznačuje, že za vhodné je možné považovat až vektory o velikosti 23 prvků a více. Z toho důvodu jsem se pro program vytvářený v rámci této práce rozhodl pro vektory o velikosti 25 prvků. Vytvoření jednotlivých vektorů je popsáno dále.

Druhým oříškem se během návrhu systému následně stala samotná neuronová síť. Pro software Mathematica naštěstí existuje dostupné řešení v podobě doplňku NeuralNetworks. Jedná se o modelovací balíček, který je specializovaný přímo na neuronové sítě. Umožňuje vytváření neuronových sítí, během učení vypisuje hodnotu globální chyby, kterou následně také zhodnotí graficky.



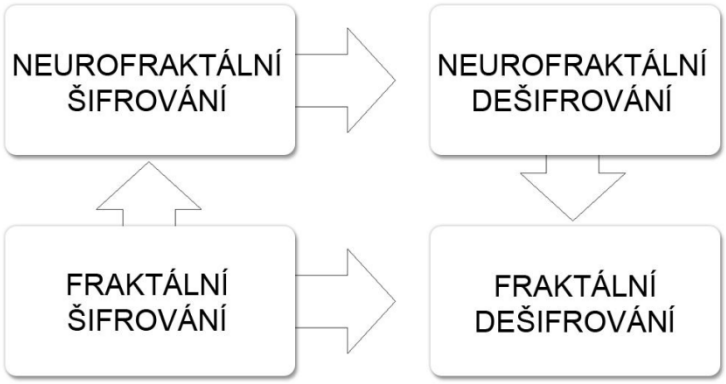
Obr. 26 - Grafické zhodnocení učení

Pro účely této práce tak byla zvolena neuronová síť se třemi vrstvami:

- vstupní vrstva – 25 vstupních neuronů
- skrytá vrstva – 28 skrytých neuronů
- výstupní vrstva – 32 výstupních neuronů

6.2 Aplikace návrhu

Obdobně jako u fraktálního šifrování, také zde bylo potřeba navrhnutý systém aplikovat. Na výběr jsem v tuto chvíli měl ze dvou možností. Mohl jsem upravit již vzniklý program pro fraktální šifrování, nebo vytvořit jakousi nadstavbu tohoto programu. Z hlediska lepší čitelnosti kódu jsem si vybral druhou možnost. Program pro neurofraktální šifrování tak jako vstup využívá výstup z programu pro fraktální šifrování. Stejně tak výstup programu pro neurofraktální dešifrování lze použít jako vstup pro program dešifrování fraktálového.



Obr. 27 - Princip nadstavby programu

Zdrojové kódy obou programů jsou součástí přílohy této práce (PIV a PV), v následující části jsou pouze podrobněji popsány důležité části obou kódů.

6.2.1 Neurofraktální šifrování

Před samotným spuštěním algoritmu je potřeba si načíst již zmíněnou knihovnu `NeuralNetworks`. Poté již začíná samotný program, který je opět součástí jedné funkce `Module`. Jako vstup tedy poslouží číslo získané programem pro fraktální šifrování. Toto je nejprve potřeba opět rozdělit na jednotlivé intervaly.

```
<< NeuralNetworks`

neurosifra[] := Module[{},
cislo = Input["Zadejte zasifrovane cislo"];

kopie = cislo;
a = IntegerLength[kopie];
c = {};
While[a > 31,
  c = PrependTo[c, Mod[kopie, 10^32]];
  kopie = Quotient[kopie, 10^32];
  a = a - 32;
];
```

Dále je potřeba připravit si síť, kterou budeme učit a samozřejmě také data kterými ji budeme učit. Jsou připraveny jednotlivé vektory z 0 a 1, a to tak, že je určen rozsah, který je rozdělen na potřebný počet dílů ze kterých jsou tyto vektory určeny. Nehrozí tak, že se objeví dvě písmena s téměř shodným vektorem. Stejně tak jsou připraveny výstupy pro neuronovou síť.

```
pocet = 25;
k1 = Table[i, {i, 0, (2^pocet) - 1, Quotient[(2^pocet) - 1, 26]}];
k2 = IntegerDigits[k1, 2];
For[i = 1, i < Length[k2] + 1, i++,
  If[Length[k2[[i]]] < pocet, PrependTo[k2[[i]], 0]; i--];
];

pismenoAin = k2[[1]]; pismenoBin = k2[[2]]; pismenoCin = k2[[3]];
pismenoDin = k2[[4]];…; pismenoZin = k2[[26]]; mezerain = k2[[27]];

pismenoAout =
 ToExpression[Characters["11133557799911111353739371351111"]];
pismenoBout =
 ToExpression[Characters["11111333555777991357915915915937"]];
…
…
…
pismenoZout =
 ToExpression[Characters["11333555777991111913915917919111"]];
mezeraout =
 ToExpression[Characters["11111111111111111111111111111111"]];
```

V dalším kroku jsou naplněny dva vektory, a to pouze podle toho, jaká písmena byla v zadaném textu rozeznána.

```
x={};y={};koef={};
For[i=1,i<Length[c]+1,i++,

  If[c[[i]]==11133557799911111353739371351111,
    AppendTo[koef,pismenoAin];If[MemberQ[x,pismenoAin]==False,
    AppendTo[x,pismenoAin];AppendTo[y,pismenoAout]]];
  If[c[[i]]==11111333555777991357915915915937,
    AppendTo[koef,pismenoBin];If[MemberQ[x,pismenoBin]==False,
    AppendTo[x,pismenoBin];AppendTo[y,pismenoBout]]];
  …
  …
  …
  If[c[[i]]==11333555777991111913915917919111,
    AppendTo[koef,pismenoZin];If[MemberQ[x,pismenoZin]==False,
    AppendTo[x,pismenoZin];AppendTo[y,pismenoZout]]];

  If[c[[i]]==11111111111111111111111111111111,
    AppendTo[koef,mezerain];If[MemberQ[x,mezerain]==False,
    AppendTo[x,mezerain];AppendTo[y,mezeraout]]];}
];
```

Poté následuje úprava těchto vektorů. Pro tak rozsáhlou síť, jakou je ta zde použitá, je potřeba, aby bylo při učení vedeno na vstup sítě dostatečné množství dat. Vstupní vektory jsou tak vhodně rozšířeny samy sebou.

```
delka = Length[x]; a = 1;
While[a == 1,
  For[i = 1, i < delka + 1, i++,
    x = AppendTo[x, x[[i]]];
    y = AppendTo[y, y[[i]]];
  ];
  delka = Length[x];
  If[delka > 50, a = 0];
];
```

Téměř na konci dochází k samotnému učení neuronové sítě.

```
iteraci = 10;
skryta = {Floor[Sqrt[Dimensions[x][[2]]*Dimensions[y][[2]]]]};
fdfrwrd2 = InitializeFeedForwardNet[x, y, Ceiling[skryta],
  Neuron -> SaturatedLinear, OutputNonlinearity -> None];
{fdfrwrd1skryta, fitrecord} = NeuralFit[fdfrwrd2, x, y, iteraci];
vahy = fdfrwrd1skryta[[1]];
```

Posledním krokem je výpis dat. Kromě celé matice vah, kterou je možné použít jako vstup pro neurofraktální dešifrování a je vypsáno také jedno velké číslo složené pouze z 0 a 1. Toto číslo je zápisem jednotlivých vektorů písmen upravených do pro odesílání srozumitelnější formy navíc doplněné na první pozici o číslo 1 (zamezuje ztrátě dat z důvodu reálné možnosti umístění vektoru začínajícím 0 na první pozici).

```
koef = Flatten[koef];
koef = PrependTo[koef, 1];

Print["Koeficient k odeslani: ", FromDigits[koef]];
Print["Matice vah k odeslani:"];
vahy
```

6.2.2 Neurofraktální dešifrování

Algoritmus neurofraktálního dešifrování je oproti tomu předešlému výrazně jednodušší a lze popsat jej ve dvou krocích. V první řadě dochází opět k aktivaci potřebné knihovny `NeuralNetworks`. Poté již následuje samotný program, zabalený do jedné funkce `Module`. V té jsou opět nejprve načteny potřebné hodnoty - vektor vah a číslo nahrazující vstupní vektory. To je okamžitě upraveno do potřebného tvaru. Zároveň je připravena struktura pro neuronovou síť.

```
<<NeuralNetworks`

neurodesifra[]:=Module[{},
  vahy=Input["Zadejte vahy"];
  koef=Input["Zadejte cislo znacici koeficienty"];

  fdfrwrd1skryta=FeedForwardNet[vahy,{AccumulatedIterations→0,
      Neuron→SaturatedLinear,FixedParameters→None,
      OutputNonlinearity→None,NumberOfInputs→25}];


  a=Drop[IntegerDigits[koef],1];
  b=Partition[a,25];
  c={};
```

Ve druhé části pak postupným přikládáním daných vektorů na vstup sítě dochází k jejich transformaci na odpovídající výstupní vektory, které jsou následně upraveny a je z nich opět vytvořeno jedno číslo, které je následně vypsáno.

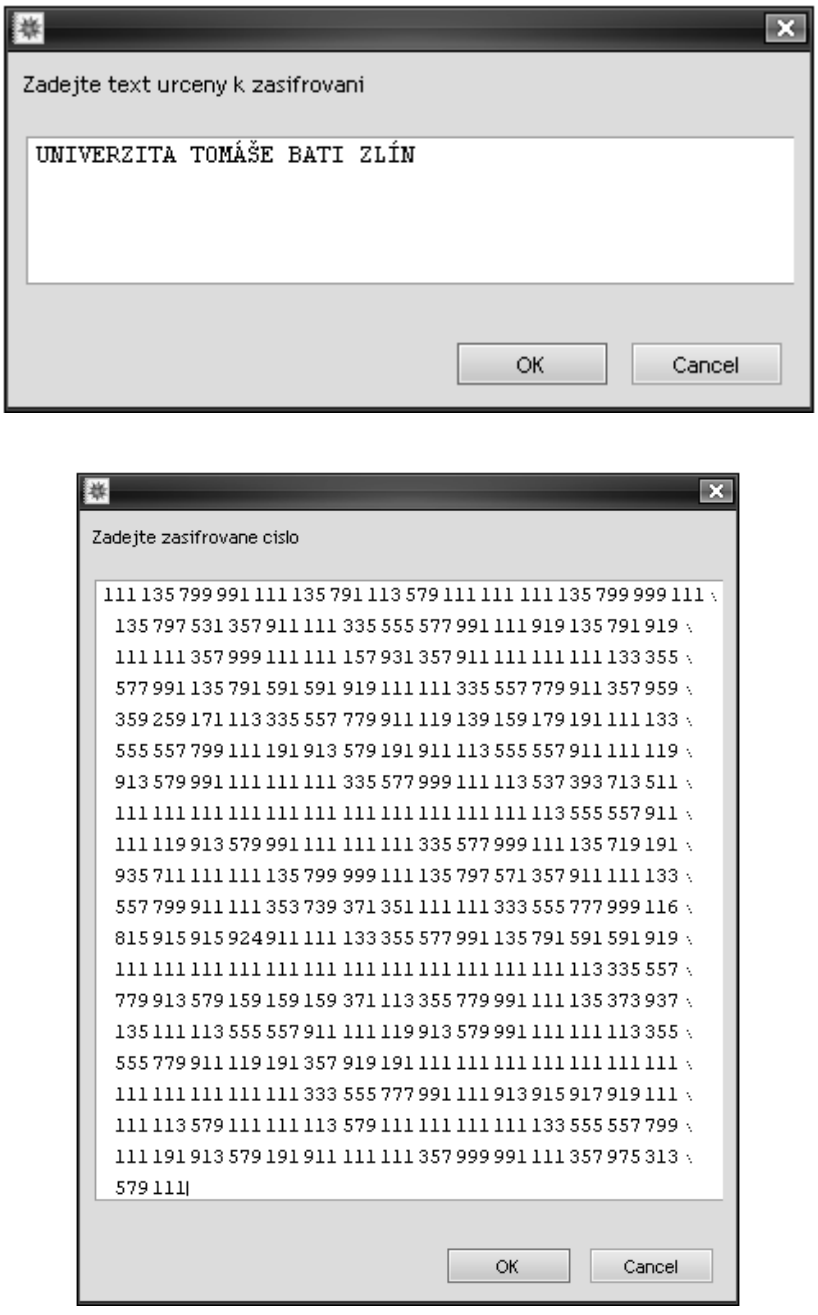
```
For[i = 1, i < Length[b] + 1, i++,
  f = fdfrwrd1skryta[b[[i]]];
  d = StringJoin[ToString[IntegerPart[f]]];
  e = ToExpression[d];
  c = AppendTo[c, e];
];

g = Flatten[c];
If[g[[1]] == 0, g = Drop[g, 1]; g = PrependTo[g, 1]];
Print["Desifrovane koeficienty: ", FromDigits[g]]
```

6.3 Otestování návrhu

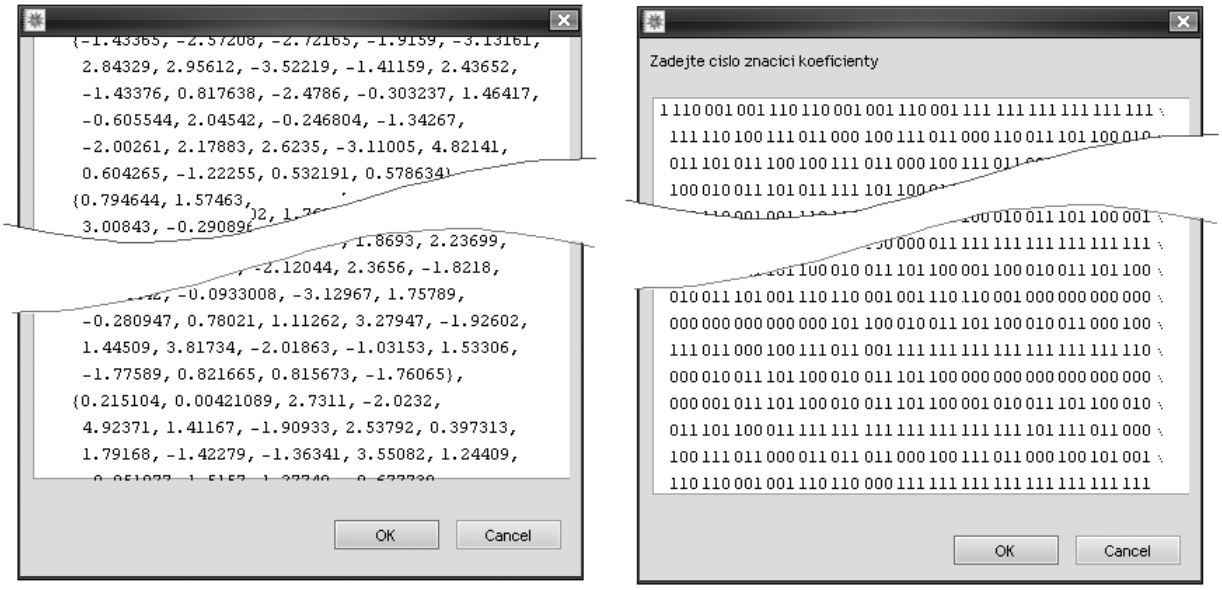
Neurofraktální šifrování je na rozdíl od šifrování fraktálního velmi náročné na čas. Samotné učení sítě je často otázkou několika minut, přičemž velmi záleží na délce šifrovaného textu. Program jsem tedy otestoval na několika jednoduchých příkladech.

Jako první jsem fraktálně zašifroval název univerzity. 27 znaků bylo v tomto případě zašifrováno 864 čísly.



Obr. 28 - Neurofraktální šifrování – zadání vstupů jednotlivým programům

Výstupem z neurofraktálního šifrování mi byla matice vah a číslo složené z 1 a 0. Ty jsem následně použil při neurofraktální dešifrování.

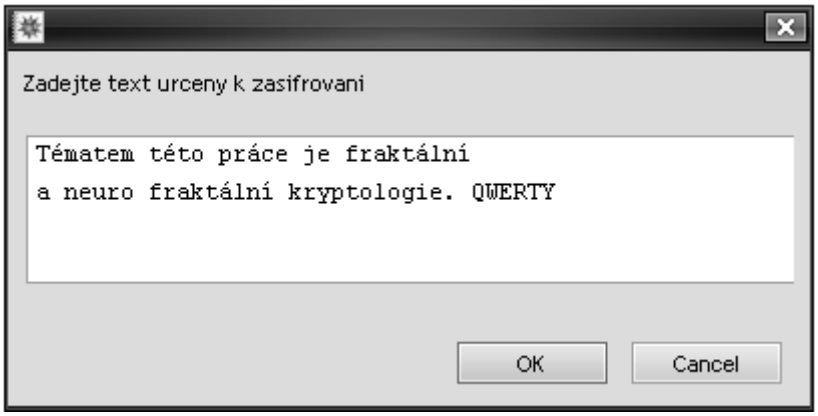


Obr. 29 - Neurofraktální dešifrování – zadání vstupů

Pomocí programu pro neurofraktální kryptologii jsem následně opět dostal číslo zahrnující jednotlivé koeficienty afinních transformací. Jejich dosazením do programu pro fraktální dešifrování jsem pak získal fraktální nápis, který je i přes evidentní poruchy stále čitelný.

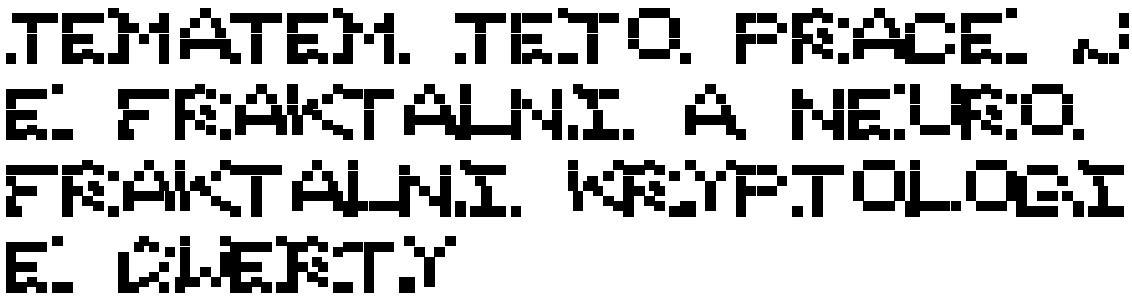
Obr. 30 - Neurofraktální dešifrování – dešifrovaný fraktální text

Pro druhý případ jsem zvolil o něco delší text (popsán 2176 čísly), přičemž doba učení neuronové sítě byla zhruba dvojnásobná oproti předešlému pokusu (cca 15 minut).



Obr. 31 - *Neurofraktální šifrování – text 2*

Výsledkem je opět fraktální text, který je i přes drobné chyby stále čitelný.



Obr. 32 - *Neurofraktální šifrování – dešifrovaný text 2*

6.3.1 Frekvenční analýza

Prolomení neurofraktálního šifrování pomocí frekvenční analýzy je ve své podstatě nemožné. Útočník může odchytit vektor vah, avšak pokud nebude vědět, že byla použita neuronová síť, nebude znát její parametry a strukturu, nebude schopný z těchto údajů nic vyčíst. Obdobně je tomu u druhého odesílaného čísla, které je pouze jakousi spletí 1 a 0, přičemž je v případě této práce navíc doplněno o další znak. I v případě, že by útočník tento fakt odhalil, je frekvenční analýza zbytku čísla velmi nepřesná a i při rozdělení na vektory o velikosti 25 prvků se podstatně liší od frekvenční analýzy českého jazyka.

7 PREZENTACE

Součástí zadání této práce bylo také vytvoření prezentace, která se tématem práce bude zabývat, přičemž by tato prezentace měla následně sloužit studentům kurzu Kryptologie. Jak název napovídá, tento kurz je zaměřen výhradně na aplikace šifrování. Je probírána jak teoretická část, tak část praktická, jenž je realizována pomocí software Mathematica. Během semestru jsou nejprve probírány šifry jednodušší – Caesarova šifra, šifra Play Fair, Vermanova šifra. Některé ze zmíněných jsou popsány i v teoretické části této práce. Později jsou probírány moderní šifry jako RSA, IFS či TEA. V blízké době by se studenti měli začít seznamovat také s principy šifer uvedenými v této práci.

Pomocí programu MS Office PowerPoint 2007 jsem tedy vytvořil odpovídající prezentaci, která je určitým shrnutím celé této práce. Je rozdělena do tří základních bloků:

- V první části je čtenář seznámen s teoretickým základem nutným pro pochopení principů fraktálního a neurofraktálního šifrování. Stručně jsou popsány zásady šifrování, vysvětleny neuronové sítě a popsány fraktály.
- Druhá část je věnována fraktálnímu šifrování. Čtenář je seznámen s principy této specifické kryptologické metody a také s pravidly, která je nutno dodržet při vytváření případného programu. Kapitola také obsahuje několik grafických ukázek.
- Obdobně zaměřená je třetí kapitola. Ta se věnuje neurofraktálnímu šifrování, jeho principům a pravidlům. Opět je doplněna několika příklady a vhodnými radami, které by mohly čtenářům pomoci při tvorbě programového vybavení.

Vzhledem ke svému rozsahu je tato prezentace pouze součástí přiloženého CD, a to jak ve formátu `pptx` (PowerPoint 2007 a novější), tak i formátu `ppt` (PowerPoint 97 – 2003) a přímo prezentujícím formátu `ppsx`.

ZÁVĚR

Cílem práce bylo seznámit čtenáře se základními principy jednoho z moderních odvětví kryptologie. Fraktální a neurofraktální šifrování jistě patří mezi šifry naší budoucnosti. Zatím se však stále jedná o jakousi „exotickou“ formu šifrování.

V případě fraktálního šifrování jde o jistou verzi monoalfabetické šifry, kdy dochází k nahrazování znaků maticemi čísel. Jak je v práci naznačeno, je právě tato skutečnost jednou z nevýhod tohoto způsobu šifrování. Případný útočník sice přesně neví, na jakém principu jsou odesílaná data založena, avšak s potřebným programovým vybavením by pro něj zřejmě nebylo problémem šifrovaný text rozluštit. Případným řešením by mohlo být určité prohození odesílaných dat, možností by také bylo slučování jednotlivých znaků.

Neurofraktální šifrování je oproti tomu mnohem kvalitnější a propracovanější systém. Odesílanými daty jsou vstupní vektory a matice vah a prahů, přičemž tato data jsou pro útočníka bez znalosti neuronové sítě pro šifrování použité absolutně bezcenná. Nevýhodou této metody je však její náročnost, kdy učení neuronové sítě je záležitostí až několika minut.

Součástí práce bylo vytvoření programů v prostředí software Mathematica demonstrujících uvedené metody šifrování a dešifrování. Postup činnosti programů je vysvětlen v jednotlivých částech práce, zdrojové kódy jsou součástí přílohy a samotné programy jsou umístěny na přiloženém CD. Je pouze na uživateli jak s nimi naloží. Bylo by možné je dále rozvíjet a upravovat – v úvahu připadají výše uvedené možnosti přeskupování, zvýšení počtu iterací během vykreslování či připojení Hopfieldovy sítě za účelem zvýšení kvality rozpoznávání znaků složených z poškozených vzorů.

ZÁVĚR V ANGLIČTINĚ

The aim was to acquaint the reader with the basic principles of a modern cryptology industry. Fractal and neuro-fractal encryption ciphers certainly among our future. So far, however, still a kind of "exotic" form of encryption.

If fractal encryption on some version monoalphabetic ciphers, where there is a substitution matrix of numbers of characters. As indicated in the job, it is precisely this fact one of the drawbacks of this method of encryption. Although the attacker does not know exactly what the principles are the transmitted data based, but with the requisite software for him probably would not solve the problem of ciphertext. Possible solution might be an interjection outgoing data, the possibility would also merge the individual characters.

Neuro-fractal encryption is much better than a sophisticated system. Consigned data are input vectors and matrices weights and thresholds, and these data are for the attacker without the knowledge of neural networks used for the encryption totally useless. A disadvantage of this method is its complexity, the neural network learning is a matter to a few minutes.

Part of the work was to develop programs in an environment of Mathematica software, demonstrating that methods of encryption and decryption. The procedure is explained by the activity of programs in different parts of the work, source code is included in the Annex and the programs themselves are located on the CD. It is only on the user how to dispose of them. Would it be possible to further develop and modify - in consideration the above options seem rearrangement, increasing the number of iterations during the rendering of, or connected Hopfield network in order to improve the quality of character recognition of the damaged compound patterns.

SEZNAM POUŽITÉ LITERATURY

[1] ZELINKA, Ivan; VČELAŘ, František; ČANDÍK, Marek. *Fraktální geometrie: principy a aplikace*. Praha: BEN, 2006. 160 s. ISBN 80-7300-193-4.

[2] BITTO, Ondřej. *Fakulta informatiky Masarykovy univerzity* [online]. 2003 [cit.2010-06-01]. Historie Kryptologie. Dostupné z WWW: <http://www.fi.muni.cz/usr/jkucera/pv109/2003/xbitto.htm>

[3] KATZ, Jonathan. *Introduction to modern cryptography*. Boca Raton: Chapman & Hall/CRC, 2008. 534 s. ISBN 978-1-58488-551-1.

[4] TŮMA, Jan. Technický atlas: Spoje. *ABC*. 1992.

[5] Enigma. *Wikipedia: the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 11. 7. 2005, poslední úrava 11. 1. 2010 [cit. 2010-06-01]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Enigma>.

[6] MALANÍK, David. *Základy praktické kryptografie* [online]. Zlín: 2009. 36 s. Prezentace. UTB Zlín, FAI. Dostupné z WWW: <www.vyuka.fai.utb.cz>.

[7] MAŘÍK, Vladimír; ŠTĚPÁNKOVÁ, Olga; LAŽANSKÝ, Jiří. *Umělá inteligence (5)*. Praha : Academia, 2007. 544 s. ISBN 978-80-200-1470-2.

[8] ZELINKA, Ivan. *Aplikovaná informatika*. Zlín: Univerzita Tomáše Bati, 2005. 183 s. ISBN 80-7318-275-0.

[9] FALCONER, Kenneth. *Fractal geometry: mathematical foundations and applications*. Chichester: Wiley, 2003. 337 s. ISBN 0-471-95724-0.

[10] GURNEY, Kevin. *An introduction to neural networks*. Boca Raton: CRC Press, 1997. 234 s. ISBN 978-1-85728-503-1.

[11] ZELINKA, Ivan. *Umělá inteligence I : neuronové sítě a genetické algoritmy*. Brno: VUTIUM, 1998. 126 s. ISBN 80-214-1163-5.

[12] NOVÁK, Mirko. *Neuronové sítě a informační systémy živých organismů*. Praha: Grada, 1993. 265 s. ISBN 80-85424-95-9.

[13] BÍLA, Jiří. *Umělá inteligence a neuronové sítě v aplikacích*. Praha: ČVUT, 1996. 115 s. ISBN 80-01-01275-1.

[14] CHRAMCOV, Bronislav. *Základy práce v prostředí Mathematica*. Zlín: Univerzita Tomáše Bati, 2006. 122 s. ISBN 80-7318-510-5.

[15] *Libol.cz* [online]. 2007 [cit. 2010-06-01]. Jednoduché šifrování. Dostupné z WWW: <http://libol.wordpress.com/2007/06/10/>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AES Advanced Encryption Standard

ASCII American Standard Code for Information Interchange

CD Kompaktní disk

DES Data Encryption Standard

IBM International Business Machines Corporation

IDEA International Data Encryption Algorithm

IFS Iterated function systems

IT Informační technologie

MS Microsoft

RSA Rivest, Shamir, Adleman

TEA Time Escape Algorithms

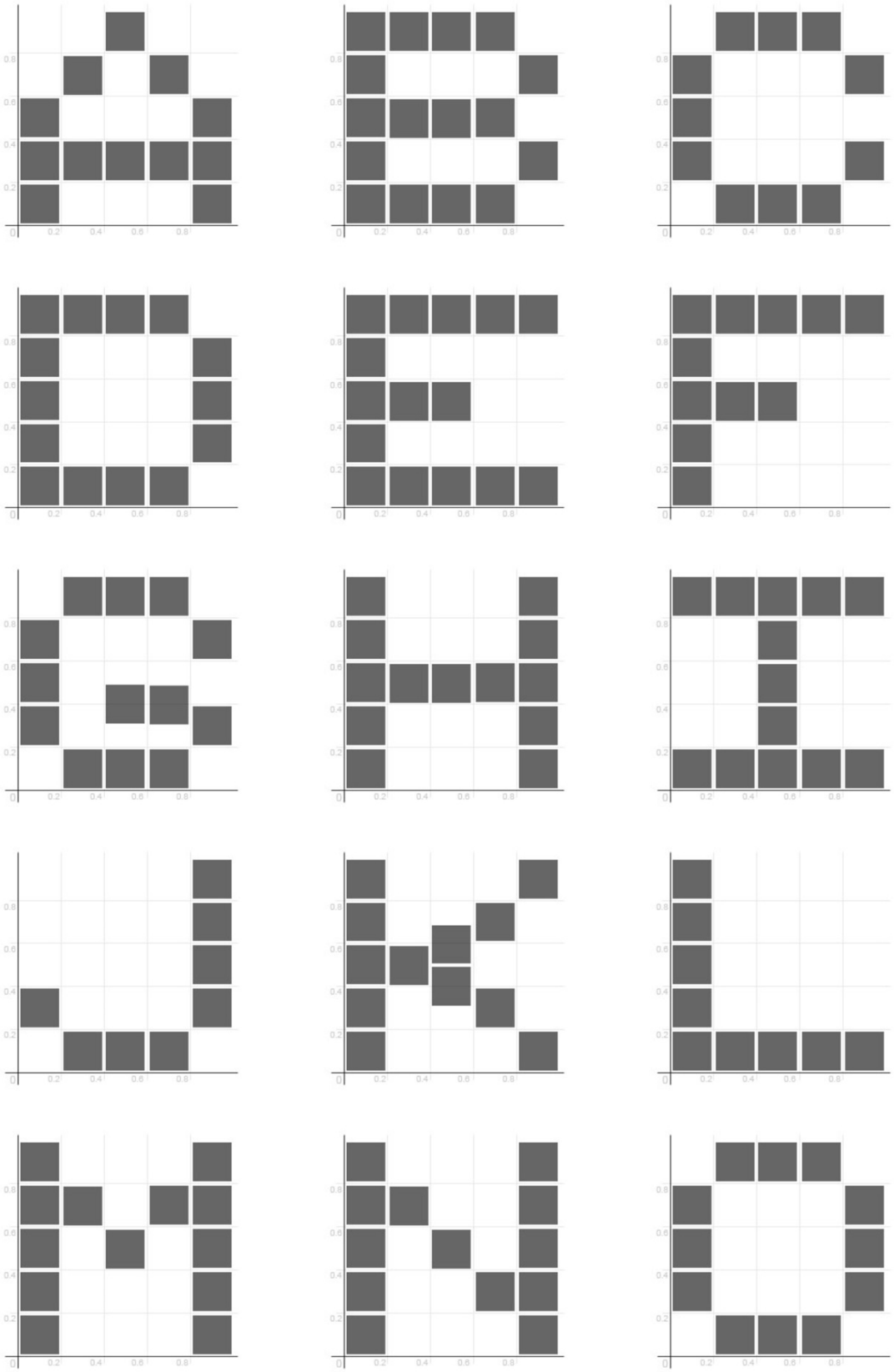
XOR Exkluzivní disjunkce – logická operace

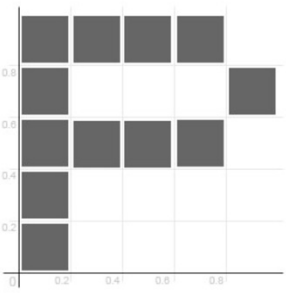
SEZNAM OBRÁZKŮ

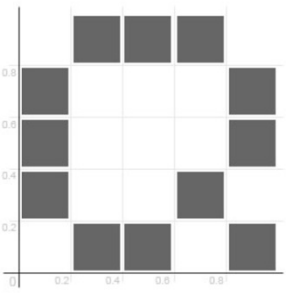
SEZNAM TABULEK

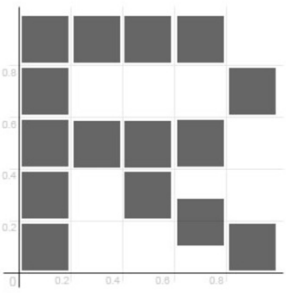
SEZNAM PŘÍLOH

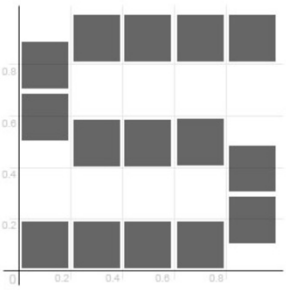
PŘÍLOHA P I: GRAFICKÁ PODOBA JEDNOTLIVÝCH PÍSMEN PRO POTŘEBY FRAKTÁLNÍ KRYPTOLOGIE

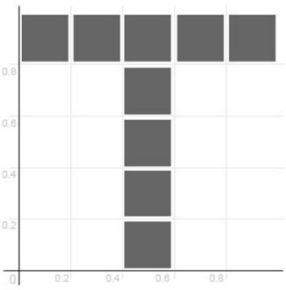


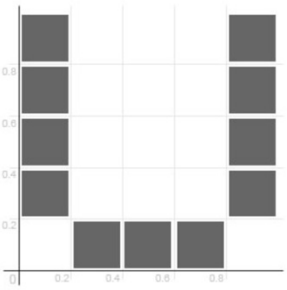


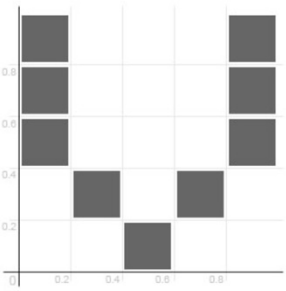


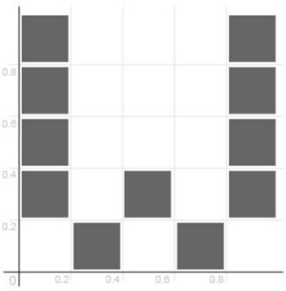


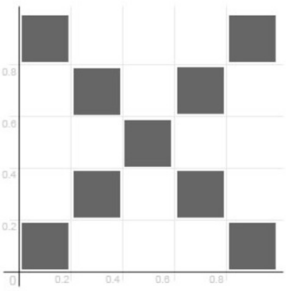


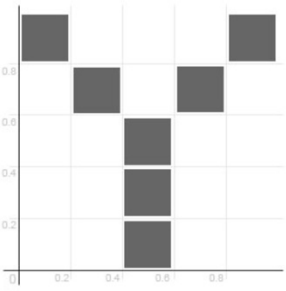


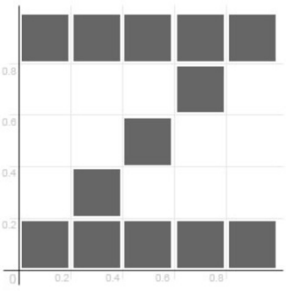


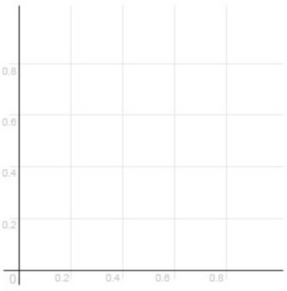












PŘÍLOHA P II: ALGORITMUS PROGRAMU PRO FRAKTÁLNÍ ŠIFROVÁNÍ

```
sifrovani[]:=Module[{},
  text=InputString["Zadejte text urceny k zasifrovani"];
  a=ToUpperCase[Characters[text]];
  a=StringReplace[a,
      {"Á"→"A","Č"→"C","Ď"→"D","É"→"E","Ě"→"E","Í"→"I","Ň"→"N","Ó"→"O",
      "Ř"→"R","Š"→"S","Ť"→"T","Ů"→"U","Ú"→"U","Ý"→"Y","Ž"→"Z","ď"→"D",
      "ě"→"E","ň"→"N","ř"→"R","ť"→"T","ů"→"U","ž"→"Z"}];
  b=StringLength[text];
  d=0;e={};
  f={};g=0;h=0;


  For[i=1,i<b+1,i++,
   Switch[a[[i]],

      "A",c={{0,0,0,0.2,0.2,0.4,0.4,0.6,0.6,0.8,0.8,0.8},{0,0.2,0.4,0.2,
      0.6,0.2,0.8,0.2,0.6,0,0.2,0.4}};d=12,
      "B",c={{0,0,0,0,0,0.2,0.2,0.2,0.4,0.4,0.4,0.6,0.6,0.6,0.8,0.8},{0,
      0.2,0.4,0.6,0.8,0,0.4,0.8,0,0.4,0.8,0,0.4,0.8,0.2,0.6}};d=16,
      "C",c={{0,0,0,0.2,0.2,0.4,0.4,0.6,0.6,0.8,0.8},{0.2,0.4,0.6,0,0.8,
      0,0.8,0,0.8,0.2,0.6}};d=11,
      "D",c={{0,0,0,0,0,0.2,0.2,0.4,0.4,0.6,0.6,0.8,0.8,0.8},{0,0.2,0.4,
      0.6,0.8,0,0.8,0,0.8,0,0.8,0.2,0.4,0.6}};d=14,
      "E",c={{0,0,0,0,0,0.2,0.2,0.2,0.4,0.4,0.4,0.6,0.6,0.8,0.8},{0,0.2,
      0.4,0.6,0.8,0,0.4,0.8,0,0.4,0.8,0,0.8,0,0.8}};d=15,
      "F",c={{0,0,0,0,0,0.2,0.2,0.4,0.4,0.6,0.8},{0,0.2,0.4,0.6,0.8,0.4,
      0.8,0.4,0.8,0.8,0.8}};d=11,
      "G",c={{0,0,0,0.2,0.2,0.4,0.4,0.4,0.6,0.6,0.6,0.8,0.8},{0.2,0.4,
      0.6,0,0.8,0,0.3,0.8,0,0.3,0.8,0.2,0.6}};d=13,
      "H",c={{0,0,0,0,0,0.2,0.4,0.6,0.8,0.8,0.8,0.8,0.8},{0,0.2,0.4,0.6,
      0.8,0.4,0.4,0.4,0,0.2,0.4,0.6,0.8}};d=13,
      "I",c={{0,0,0.2,0.2,0.4,0.4,0.4,0.4,0.4,0.6,0.6,0.8,0.8},{0,0.8,0,
      0.8,0,0.2,0.4,0.6,0.8,0,0.8,0,0.8}};d=13,
      "J",c={{0,0.2,0.4,0.6,0.8,0.8,0.8,0.8},{0.2,0,0,0,0.2,0.4,0.6,0.8}}
      ;d=8,
      "K",c={{0,0,0,0,0,0.2,0.4,0.4,0.6,0.6,0.8,0.8},{0,0.2,0.4,0.6,0.8,
      0.4,0.3,0.5,0.2,0.6,0,0.8}};d=12,
      "L",c={{0,0,0,0,0,0.2,0.4,0.6,0.8},{0,0.2,0.4,0.6,0.8,0,0,0,0}};d=9
      ,
      "M",c={{0,0,0,0,0,0.2,0.4,0.6,0.8,0.8,0.8,0.8,0.8},{0,0.2,0.4,0.6,
      0.8,0.6,0.4,0.6,0,0.2,0.4,0.6,0.8}};d=13,
      "N",c={{0,0,0,0,0,0.2,0.4,0.6,0.8,0.8,0.8,0.8,0.8},{0,0.2,0.4,0.6,
      0.8,0.6,0.4,0.2,0,0.2,0.4,0.6,0.8}};d=13,
      "O",c={{0,0,0,0.2,0.2,0.4,0.4,0.6,0.6,0.8,0.8,0.8},{0.2,0.4,0.6,0,
      0.8,0,0.8,0,0.8,0.2,0.4,0.6}};d=12,
      "P",c={{0,0,0,0,0,0.2,0.2,0.4,0.4,0.6,0.6,0.8},{0,0.2,0.4,0.6,0.8,
      0.4,0.8,0.4,0.8,0.4,0.8,0.6}};d=12,
      "Q",c={{0,0,0,0.2,0.2,0.4,0.4,0.6,0.6,0.8,0.8,0.8},{0.2,0.4,0.6,0,
      0.8,0,0.8,0.2,0.8,0,0.4,0.6}};d=12,
      "R",c={{0,0,0,0,0,0.2,0.2,0.4,0.4,0.4,0.6,0.6,0.6,0.8,0.8},{0,0.2,
      0.4,0.6,0.8,0.4,0.8,0.2,0.4,0.8,0.1,0.4,0.8,0,0.6}};d=15,
      "S",c={{0,0,0,0.2,0.2,0.2,0.4,0.4,0.4,0.6,0.6,0.6,0.8,0.8,0.8},{0,
      0.5,0.7,0,0.4,0.8,0,0.4,0.8,0,0.4,0.8,0.1,0.3,0.8}};d=15,
      "T",c={{0,0.2,0.4,0.4,0.4,0.4,0.4,0.6,0.8},{0.8,0.8,0,0.2,0.4,0.6,
      0.8,0.8,0.8}};d=9,
```

```
    "U",c={{0,0,0,0,0.2,0.4,0.6,0.8,0.8,0.8,0.8},{0.2,0.4,0.6,0.8,0,0,
    0,0.2,0.4,0.6,0.8}};d=11,
    "V",c={{0,0,0,0.2,0.4,0.6,0.8,0.8,0.8},{0.4,0.6,0.8,0.2,0,0.2,0.4,
    0.6,0.8}};d=9,
    "W",c={{0,0,0,0,0.2,0.4,0.6,0.8,0.8,0.8,0.8},{0.2,0.4,0.6,0.8,0,
    0.2,0,0.2,0.4,0.6,0.8}};d=11,
    "X",c={{0,0,0.2,0.2,0.4,0.6,0.6,0.8,0.8},{0,0.8,0.2,0.6,0.4,0.2,
    0.6,0,0.8}};d=9,
    "Y",c={{0,0.2,0.4,0.4,0.4,0.6,0.8},{0.8,0.6,0,0.2,0.4,0.6,0.8}};d=7
    ,
    "Z",c={{0,0,0.2,0.2,0.2,0.4,0.4,0.4,0.6,0.6,0.6,0.8,0.8},{0,0.8,0,
    0.2,0.8,0,0.4,0.8,0,0.6,0.8,0,0.8}};d=13,
    " ",c={{},{}};d=0
     _,c={{},{}};d=16
  ];

  If[d<16,
   For[j=1,j<17-d,j++,
    AppendTo[c[[1]],0];
    AppendTo[c[[2]],0];
    ]
   ];

  If[h==1,h=0,
   e=AppendTo[e,c[[1]]];
   e=AppendTo[e,c[[2]]];
   ];
  ];

 f=Flatten[e];
 f=Round[(f*10)+1

 For[i=1,i<Length[f]+1,i++,
  g=g+f[[i]]*10^(Length[f]-i);
  ];

 Print["Zasifrovany text: ",g]
]
```

PŘÍLOHA P III: ALGORITMUS PROGRAMU PRO FRAKTÁLNÍ DEŠIFROVÁNÍ

```
Needs["ProgrammingInMathematica`AffineMaps`"]
Needs["ProgrammingInMathematica`IFS`"]

desifrovani[]:=Module[{},
  text=Input["Zadejte zasifrovany text"];

  a={};c={{},{}};
  i=1;
  While[i==1,
   a=PrependTo[a,(Mod[text,10]-1)/10];
   text=Quotient[text,10];
   If[text<1,i=0]
   ];
  b=Length[a];
  b1=0;

  While[b>31,
   c[[1]]=AppendTo[c[[1]],Take[a,16]];
   c[[2]]=AppendTo[c[[2]],Take[a,{17,32}]];
   a=Drop[a,32];
   b=b-32;
   b1=b1+1;
   ];

  i=1;f={};g={};
  While[i<Dimensions[c][[2]]+1,
   d=c[[1]][[i]];
   e=c[[2]][[i]];
   For[j=16,j>0,j--,
    If[d[[j]]!=0 ,
      f=AppendTo[f,Take[d,j]];
      g=AppendTo[g,Take[e,j]];
      j=0,
      If[j==1,
        f=AppendTo[f,{0.9}];
        g=AppendTo[g,{0.9}];
        j=0;
        ];
      ];
    ];
   i++;
   ];

  f1={};l=0;m=0;
```

```
  For[i=1,i<b1+1,i++,
   h=f[[i]];
   k=g[[i]];
   For[j=1,j<Length[h]+1,j++,
    If[h[[j]]<0.9,
      AppendTo[f1,{0.195,0,0,0.195,h[[j]]+l,k[[j]]+m}],
      l=l-0.5;
      ];
    ];
   l=l+1.1;
   If[l>21,
    l=0;
    m=m-1.5;
    ];
   ];

  ctvr=Show[Graphics[{RGBColor[0,0,0],Polygon[
      {{0,0},{1,0},{1,1},{0,1}}]}],
      Axes→False,Frame→False,PlotRange→All];
  AM[{a_, b_, c_, d_, e_, f_}] :=map[{{a,-b,e},{c,d,f}}];
  f2=AM/@f1;
  ifs0=IFS[f2];

  Print["Desifrovany text:"];
  Show[Nest[ifs0,ctvr,1]]
]
```

PŘÍLOHA P IV: ALGORITMUS PROGRAMU PRO

NEUROFRAKTÁLNÍ ŠIFROVÁNÍ

```
<<NeuralNetworks`

neurosifra[]:=Module[{},
  cislo=Input["Zadejte zasifrovane cislo"];

  kopie=cislo;
  a=IntegerLength[kopie];
  c={};
  While[a>31,
   c=PrependTo[c,Mod[kopie,10^32]];
   kopie=Quotient[kopie,10^32];
   a=a-32;
  ];

  pocet=25;
  k1=Table[i,{i,0,(2^pocet)-1,Quotient[(2^pocet)-1,26]}];
  k2=IntegerDigits[k1,2];
  For[i=1,i<Length[k2]+1,i++,
   If[Length[k2[[i]]]<pocet,PrependTo[k2[[i]],0];i--];
  ];

  pismenoAin=k2[[1]];pismenoBin=k2[[2]];pismenoCin=k2[[3]];
  pismenoDin=k2[[4]];PismenoEin=k2[[5]];pismenoFin=k2[[6]];
  pismenoGin=k2[[7]];pismenoHin=k2[[8]];pismenoIin=k2[[9]];
  pismenoJin=k2[[10]];pismenoKin=k2[[11]];pismenoLin=k2[[12]];
  pismenoMin=k2[[13]];pismenoNin=k2[[14]];pismenoOin=k2[[15]];
  pismenoPin=k2[[16]];pismenoQin=k2[[17]];pismenoRin=k2[[18]];
  pismenoSin=k2[[19]];pismenoTin=k2[[20]];pismenoUin=k2[[21]];
  pismenoVin=k2[[22]];pismenoWin=k2[[23]];pismenoXin=k2[[24]];
  pismenoYin=k2[[25]];pismenoZin=k2[[26]];mezerain=k2[[27]];


pismenoAout=ToExpression[Characters["11133557799911111353739371351111"]];
pismenoBout=ToExpression[Characters["11111333555777991357915915915937"]];
pismenoCout=ToExpression[Characters["11133557799111113571919193711111"]];
pismenoDout=ToExpression[Characters["11111335577999111357919191935711"]];
pismenoEout=ToExpression[Characters["11111333555779911357915915919191"]];
pismenoFout=ToExpression[Characters["11111335579111111357959599911111"]];
pismenoGout=ToExpression[Characters["11133555777991113571914914937111"]];
pismenoHout=ToExpression[Characters["11111357999991111357955513579111"]];
pismenoIout=ToExpression[Characters["11335555577991111919135791919111"]];
pismenoJout=ToExpression[Characters["13579999111111113111357911111111"]];
pismenoKout=ToExpression[Characters["11111355779911111357954637191111"]];
pismenoLout=ToExpression[Characters["11111357911111111357911111111111"]];
pismenoMout=ToExpression[Characters["11111357999991111357975713579111"]];
pismenoNout=ToExpression[Characters["11111357999991111357975313579111"]];
pismenoOout=ToExpression[Characters["11133557799911113571919193571111"]];
pismenoPout=ToExpression[Characters["11111335577911111357959595971111"]];
pismenoQout=ToExpression[Characters["11133557799911113571919391571111"]];
pismenoRout=ToExpression[Characters["11111335557779911357959359259171"]];
pismenoSout=ToExpression[Characters["11133355577799911681591591592491"]];
pismenoTout=ToExpression[Characters["13555557911111119913579991111111"]];
pismenoUout=ToExpression[Characters["11113579999111113579111357911111"]];
pismenoVout=ToExpression[Characters["11135799911111115793135791111111"]];
pismenoWout=ToExpression[Characters["11113579999111113579131357911111"]];
pismenoXout=ToExpression[Characters["11335779911111111937537191111111"]];
```

```
pismenoYout=ToExpression[Characters["13555791111111119713579111111111"]];
pismenoZout=ToExpression[Characters["113335557779911119139159179191111"]];
mezeraout=ToExpression[Characters["11111111111111111111111111111111"]];

x={};y={};koef={};
  For[i=1,i<Length[c]+1,i++,
  {

{If[c[[i]]==11133557799911111353739371351111,AppendTo[koef,pismenoAin];
      If[MemberQ[x,pismenoAin]==False,AppendTo[x,pismenoAin];
      AppendTo[y,pismenoAout]]];
If[c[[i]]==111113335557779913579159159 15937,AppendTo[koef,pismenoBin];
      If[MemberQ[x,pismenoBin]==False,AppendTo[x,pismenoBin];
      AppendTo[y,pismenoBout]]];
If[c[[i]]==11133557799111113571919193711111,AppendTo[koef,pismenoCin];
      If[MemberQ[x,pismenoCin]==False,AppendTo[x,pismenoCin];
      AppendTo[y,pismenoCout]]];
If[c[[i]]==1111133557799911135791919193571 1,AppendTo[koef,pismenoDin];
      If[MemberQ[x,pismenoDin]==False,AppendTo[x,pismenoDin];
      AppendTo[y,pismenoDout]]];
If[c[[i]]==111113335557799113579159159191 91,AppendTo[koef,pismenoEin];
      If[MemberQ[x,pismenoEin]==False,AppendTo[x,pismenoEin];
      AppendTo[y,pismenoEout]]];
If[c[[i]]==11113335579111111357959599911111,AppendTo[koef,pismenoFin];
      If[MemberQ[x,pismenoFin]==False,AppendTo[x,pismenoFin];
      AppendTo[y,pismenoFout]]];
If[c[[i]]==11133555779911135719149149 37111,AppendTo[koef,pismenoGin];
      If[MemberQ[x,pismenoGin]==False,AppendTo[x,pismenoGin];
      AppendTo[y,pismenoGout]]];
If[c[[i]]==11111357999991111357955135791 11,AppendTo[koef,pismenoHin];
      If[MemberQ[x,pismenoHin]==False,AppendTo[x,pismenoHin];
      AppendTo[y,pismenoHout]]];
If[c[[i]]==113355555779911119191357919191 11,AppendTo[koef,pismenoIin];
      If[MemberQ[x,pismenoIin]==False,AppendTo[x,pismenoIin];
      AppendTo[y,pismenoIout]]];
If[c[[i]]==13579999111111113111357911111111,AppendTo[koef,pismenoJin];
      If[MemberQ[x,pismenoJin]==False,AppendTo[x,pismenoJin];
      AppendTo[y,pismenoJout]]];
If[c[[i]]==11111355779911111357954637191111,AppendTo[koef,pismenoKin];
      If[MemberQ[x,pismenoKin]==False,AppendTo[x,pismenoKin];
      AppendTo[y,pismenoKout]]];
If[c[[i]]==11111357911111111357911111111111,AppendTo[koef,pismenoLin];
      If[MemberQ[x,pismenoLin]==False,AppendTo[x,pismenoLin];
      AppendTo[y,pismenoLout]]];
If[c[[i]]==11111357999991111357975713579111,AppendTo[koef,pismenoMin];
      If[MemberQ[x,pismenoMin]==False,AppendTo[x,pismenoMin];
      AppendTo[y,pismenoMout]]];
If[c[[i]]==1111135799999111135797531357911 1,AppendTo[koef,pismenoNin];
      If[MemberQ[x,pismenoNin]==False,AppendTo[x,pismenoNin];
      AppendTo[y,pismenoNout]]];
If[c[[i]]==11133557799911113571919193571111,AppendTo[koef,pismenoOin];
      If[MemberQ[x,pismenoOin]==False,AppendTo[x,pismenoOin];
      AppendTo[y,pismenoOout]]];
If[c[[i]]==11111335577911111357959595971111,AppendTo[koef,pismenoPin];
      If[MemberQ[x,pismenoPin]==False,AppendTo[x,pismenoPin];
      AppendTo[y,pismenoPout]]];
If[c[[i]]==11133557799911113571919391571111,AppendTo[koef,pismenoQin];
      If[MemberQ[x,pismenoQin]==False,AppendTo[x,pismenoQin];
      AppendTo[y,pismenoQout]]];
```

```
If[c[[i]]==11111335557779911357959359259171,AppendTo[koef,pismenoRin];
      If[MemberQ[x,pismenoRin]==False,AppendTo[x,pismenoRin];
      AppendTo[y,pismenoRout]]];
If[c[[i]]==11133355577799911681591591592491,AppendTo[koef,pismenoSin];
      If[MemberQ[x,pismenoSin]==False,AppendTo[x,pismenoSin];
      AppendTo[y,pismenoSout]]];
If[c[[i]]==13555557911111119913579991111111,AppendTo[koef,pismenoTin];
      If[MemberQ[x,pismenoTin]==False,AppendTo[x,pismenoTin];
      AppendTo[y,pismenoTout]]];
If[c[[i]]==11113579999111113579111357911111,AppendTo[koef,pismenoUin];
      If[MemberQ[x,pismenoUin]==False,AppendTo[x,pismenoUin];
      AppendTo[y,pismenoUout]]];
If[c[[i]]==11135799911111115793135791111111,AppendTo[koef,pismenoVin];
      If[MemberQ[x,pismenoVin]==False,AppendTo[x,pismenoVin];
      AppendTo[y,pismenoVout]]];
If[c[[i]]==11113579999111113579131357911111,AppendTo[koef,pismenoWin];
      If[MemberQ[x,pismenoWin]==False,AppendTo[x,pismenoWin];
      AppendTo[y,pismenoWout]]];
If[c[[i]]==11335779911111111937537191111111,AppendTo[koef,pismenoXin];
      If[MemberQ[x,pismenoXin]==False,AppendTo[x,pismenoXin];
      AppendTo[y,pismenoXout]]];},
{If[c[[i]]==13555791111111119713579111111111,AppendTo[koef,pismenoYin];
      If[MemberQ[x,pismenoYin]==False,AppendTo[x,pismenoYin];
      AppendTo[y,pismenoYout]]];
If[c[[i]]==11333555779911119139159179191111,AppendTo[koef,pismenoZin];
      If[MemberQ[x,pismenoZin]==False,AppendTo[x,pismenoZin];
      AppendTo[y,pismenoZout]]];
If[c[[i]]==11111111111111111111111111111111,AppendTo[koef,mezerain];
      If[MemberQ[x,mezerain]==False,AppendTo[x,mezerain];
      AppendTo[y,mezeraout]]];}
   }
  ];

  delka=Length[x];a=1;
  While[a==1,
   For[i=1,i<delka+1,i++,
    x=AppendTo[x,x[[i]]];
    y=AppendTo[y,y[[i]]];
    ];
   delka=Length[x];
   If[delka>50,a=0];
  ];

  iteraci=10;

  skryta={Floor[Sqrt[Dimensions[x][[2]]*Dimensions[y][[2]]]]};
      fdfrwrd2=InitializeFeedForwardNet[x,y,Ceiling[skryta],
      Neuron→SaturatedLinear,OutputNonlinearity→None];
  {fdfrwrd1skryta,fitrecord}=NeuralFit[fdfrwrd2,x,y,iteraci];
  vahy=fdfrwrd1skryta[[1]];

  koef=Flatten[koef];
  koef=PrependTo[koef,1];
  Print["Koeficient k odeslani: ",FromDigits[koef]];
  Print["Matice vah k odeslani:"];
  vahy


  ]
```

PŘÍLOHA P V: ALGORITMUS PROGRAMU PRO NEUROFRAKTÁLNÍ DEŠIFROVÁNÍ

```
<<NeuralNetworks`

neurodesifra[]:=Module[{},
  vahy=Input["Zadejte vahy"];
  koef=Input["Zadejte cislo znacici koeficienty"];

  fdfrwrd1skryta=FeedForwardNet[vahy,{AccumulatedIterations→0,
      Neuron→SaturatedLinear,FixedParameters→None,
      OutputNonlinearity→None,NumberOfInputs→25}];


  a=Drop[IntegerDigits[koef],1];
  b=Partition[a,25];
  c={};


  For[i=1,i<Length[b]+1,i++,
   f=fdfrwrd1skryta[b[[i]]];
   d=StringJoin[ToString[IntegerPart[f]]];
   e=ToExpression[d];
   c=AppendTo[c,e];
  ];

  g=Flatten[c];
  If[g[[1]]==0,g=Drop[g,1];g=PrependTo[g,1]];

  Print["Desifrovane koeficienty: ",FromDigits[g]]

 ]
```