
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ**
**DEPARTMENT OF COMPUTER SYSTEMS**


# DIGITÁLNÍ ZVUKOVÁ STEGANOGRAFIE

**DIGITAL AUDIO STEGANOGRAPHY**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** — **PETR KABELKA**
**AUTHOR**

**VEDOUCÍ PRÁCE** — **Ing. JOSEF STRNADEL, Ph.D.**
**SUPERVISOR**



**BRNO 2023**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

# Zadání bakalářské práce

144770

Ústav: Ústav počítačových systémů (UPSY)
Student: **Kabelka Petr**
Program: Informační technologie
Specializace: Informační technologie
Název: **Digitální zvuková steganografie**
Kategorie: Bezpečnost
Akademický rok: 2022/23

Zadání:

1. Vytvořte přehled metod z oblasti digitální steganografie, proveďte rešerši v podoblasti steganografie zabývající se skrýváním informace ve zvukových datech (stručně "zvuková steganografie", angl. Digital Audio Steganography, DAS).
2. Zvolte typ skrývané informace (textová, obrazová, zvuková apod.) a její vlastnosti. Na základě existující či vlastní analýzy způsobů ukládání zvukových dat a typu skrývané informace zvolte vhodné způsoby pro ukládání dat a vhodné metody pro DAS.
3. V souladu s bodem 2 připravte vhodnou sadu dat pro ověřování vlastností zvolených steganografických metod a navrhněte mechanismus vyhodnocování jejich vlastností na základě těchto dat.
4. Implementujte několik publikovaných metod DAS; diskutujte jejich možné modifikace, navrhněte a implementujte alespoň jednu vlastní metodu DAS.
5. Ověřte funkčnost implementovaných metod; vyhodnoťte jejich vlastnosti a porovnejte je jak navzájem, tak s daty z vybraných publikací.
6. Shrňte dosažené výsledky, diskutujte možné směry využití a rozvoje předloženého řešení.

Literatura:

- H. Dutta, R. K. Das, S. Nandi, S. R. M. Prasanna: An Overview of Digital Audio Steganography, IETE Technical Review, 37:6, pp. 632-650, 2020. DOI: 10.1080/02564602.2019.1699454.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Strnadel Josef, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1.11.2022
Termín pro odevzdání: 10.5.2023
Datum schválení: 31.10.2022

# Abstrakt

Obor steganografie se zabývá ukrýváním informací s cílem zakrýt jejich samotnou existenci. Cílem této práce je vytvořit přehled existujících metod digitální zvukové steganografie a implementovat vybrané metody. Tato práce představuje jednoduše rozšiřitelnou a použitelnou multiplatformní knihovnu pro účely zvukové steganografie v programovacím jazyce Python včetně programu, který ji využívá. Na závěr jsou implementované metody srovnány na základě kvality, kapacity a robustnosti mezi sebou a s publikovanými metodami. Jelikož je velmi těžké najít na internetu konkrétní implementace různých publikovaných metod, umožňuje vytvořená knihovna téměř komukoliv jednoduše využívat implementované steganografické metody v praxi a pokračovat ve výzkumu této oblasti bez nutnosti reimplementace všech metod.



# Abstract

The field of steganography deals with concealing information with the goal of hiding their very existence. The goal of this work is to create a summary of existing digital audio steganography methods and implement some of them. This work presents easily extensible and usable multiplatform library for audio steganography written in the Python programming language and a program that uses it. At the end of this work are the implemented methods compared against each other and against the published methods. Because it is hard to find concrete implementations of various published methods, the presented library enables almost anyone to easily use the implemented steganographic methods and to continue working on research in this field without the need to reimplement all of them.



# Klíčová slova

zvuková steganografie, ukrývání informací, bezpečnost, soukromí, vodoznak, WAV, Fourierova transformace, Python, NumPy, nejméně významný bit, ozvěnová steganografie, přímé rozprostřené spektrum, fázové kódování, paritní kódování, vkládání tónu, úseky ticha, vlnková transformace

# Keywords

audio steganography, information hiding, security, privacy, watermark, WAV, Fourier transform, Python, NumPy, least significant bit, echo steganography, direct sequence spread spectrum, phase coding, parity coding, tone insertion, silence interval, wavelet transform



# Citace

KABELKA, Petr. *Digitální zvuková steganografie*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Josef Strnadel, Ph.D.

# Digitální zvuková steganografie

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Josefa Strnadela Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

. . . . . .
Petr Kabelka
2. května 2023


## Poděkování

Rád bych poděkoval panu Ing. Josefu Strnadelovi Ph.D. za odborné vedení mé práce a za cenná doporučení při řešení. Také bych chtěl poděkovat mé rodině za podporu a trpělivost při mém studiu.

# Obsah

# Kapitola 1

# Úvod

Steganografie je obor zabývající se skrýváním informace takovým způsobem, aby nebylo možné zjistit její přítomnost. Důvod pro takovéto skrývání informace může být zajištění bezpečnosti před třetí stranou, pro kterou není informace určena. Steganografie se dnes dělí na tradiční a digitální. V digitální steganografii je velké množství typů dat, do kterých je možné vkládat utajované informace a digitální zvuk je jedním z nich.

V oblasti digitální zvukové steganografie je dostupné velké množství kvalitní literatury popisující teoretické fungování metod, kterými se jednotlivé publikace zabývají. Je však těžké najít konkrétní implementace popisovaných metod i přesto, že autoři je implementovali pro získání výsledků. Proto je hlavním cílem této práce vytvořit softwarovou knihovnu a program v programovacím jazyce Python, aby měl kdokoliv možnost pracovat na výzkumu v této vědecké disciplíně. Ne vždy stačí pouze algoritmický popis různých metod pro pochopení jejich fungování. Implementace v konkrétním programovacím jazyce nám přináší formalismus, který umožňuje snazší pochopení. Chtěl bych proto touto prací přispět v této oblasti tím, že bude přístupnější pro širší veřejnost.

V následujících kapitolách této práce je shrnut současný stav tohoto oboru, jsou vysvětleny cíle a technické prostředky použité k jejich dosažení a zhodnoceny dosažené výsledky. V kapitole 2 jsou do hloubky popsány steganografické metody vybrané pro implementaci a některé další používané. Kapitola 3 popisuje, proč byl vybrán programovací jazyk Python a použité knihovny. Je zde také popsána struktura balíčků pro Python a steganografické metody, které byly vybrány pro implementaci. Na konci kapitoly je popsán způsob vyhodnocení kvality metod a popis vlastní metody. Kapitola 4 je zaměřená na popis rozvržení kódu a implementaci steganografických metod. Kapitola 5 obsahuje výsledky testování a srovnání metod. Na závěr obsahuje kapitola 6 shrnutí výsledků práce a vyhlídky do budoucna.



[1] https://creativecommons.org/licenses/by/4.0

# Kapitola 2

# Digitální steganografie

Tato kapitola představí steganografii obecně a její klasifikace. Poté bude stručně popsán způsob, jakým se dnes reprezentuje digitální zvuk. Dále budou vysvětleny vlastnosti, na základě kterých budou metody hodnoceny a srovnávány včetně způsobu jejich ověření. Zbytek této kapitoly se bude věnovat existujícím steganografickým metodám včetně jejich možných modifikací.

## 2.1 Základní definice a klasifikace

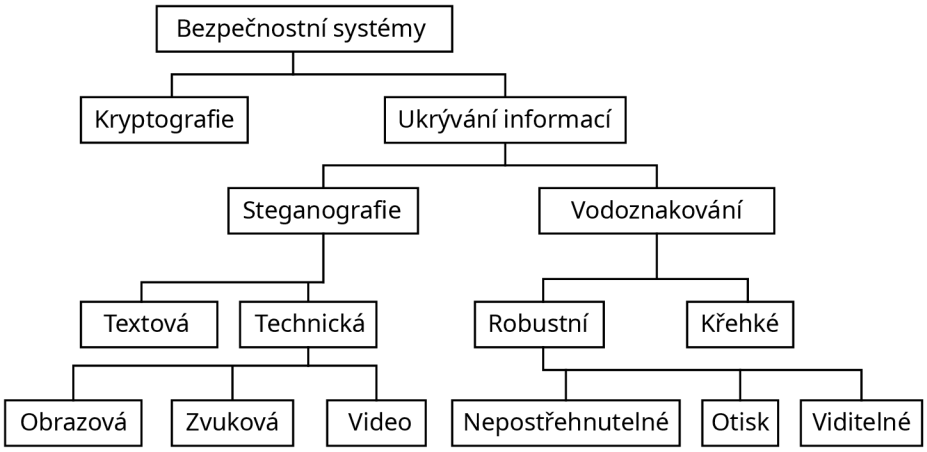
Steganografie je praktika skrývání informace s cílem zakrýt její samotnou existenci a ne jen znemožnit její čtení [1][2][10][11]. Původ této praktiky se vrací až do antiky – vyrytím písma do destičky, která se následně zalila voskem; ve středověké Evropě položením papíru nebo dřevěné šablony na nevinně vypadající text, čímž zvýrazní tajnou zprávu. Steganografie se tedy v různých podobách praktikovala po tisíciletí až do dnešní počítačové doby [2], kdy je považována za podobor zabezpečení dat [10] a více se přesouvá od tradiční k digitální steganografii.

V dnešním světě se stalo zabezpečení dat zájmem všech a steganografie jako proces ukrývání zprávy do jiného typu média slouží k ochraně před neautorizovanými či neoprávněnými příjemci [11]. Médium, do kterého je informace skrývána, se nazývá nosič a může být různých typů, jako například obrázek, audio, video, IP datagram [11], text a mnoho dalších. Hierarchickou klasifikaci je možné vidět na obrázku 2.1.

Pokud je nosičem počítačový soubor, pak se může nazývat *krycí soubor*. Názvy *nosič* a *krycí soubor* jsou však často zaměnitelné. Každý typ nosiče má svoji oblast výzkumu, která se jej týká. Některé steganografické metody používané u různých typů nosičů jsou konceptuálně stejné a v praxi se aplikují podobným způsobem. Tato práce se však bude zabývat pouze metodami použitelnými pro digitální zvuk a proto následující podkapitola vysvětlí způsob, jakým se zvuk reprezentuje v digitální formě.

## 2.2 Reprezentace a způsob uložení digitálního zvuku

V reálném světě se zvuk vyskytuje jako vibrace materiálu při frekvenci slyšitelné člověkem, která se pohybuje mezi 10 Hz a 20 000 Hz [30]. Zvuk je ve skutečném světě spojitý signál – má nekonečný počet hodnot – a je tedy definován všude. Základními parametry signálu jsou *frekvence*, *perioda* a *amplituda*. Amplituda udává maximální výchylku signálu. Frekvence, nebo také kmitočet, vyjadřuje počet period za jednotku času a udává se v Hertzích (Hz).




Obrázek 2.1: Klasifikace oboru informační bezpečnosti a steganografie. Přeloženo z [19].

Perioda je úzce svázaná s frekvencí a vyjadřuje délku jednoho opakování v signálu, udává se v sekundách [6]. Vztah mezi frekvencí a periodou lze vidět v rovnici 2.1, kde $T$ je perioda a $f$ je frekvence.

$$T = \frac{1}{f} \tag{2.1}$$

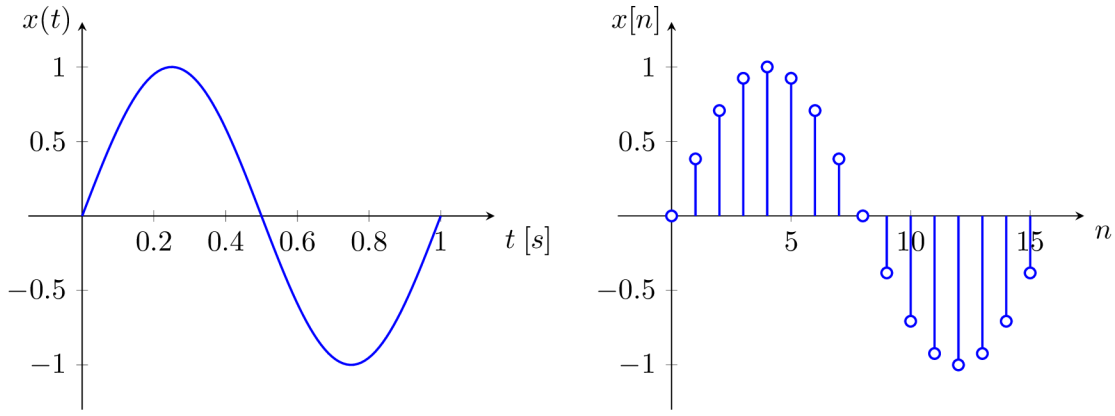
Aby bylo možné se zvukem pracovat digitálně, je nutné jej převést do diskrétní formy, která má konečný počet hodnot. Toho je možné docílit vzorkováním signálu na určité vzorkovací frekvenci a poté kvantováním [6]. Posledním krokem pro převod spojitého signálu na digitální je kódování. Tyto tři procesy tvoří systém, který se nazývá *pulzně-kódová modulace* (PCM) [22], který je neopominutelným základem pro práci s digitálním audiem.

### Vzorkování

Vzorkování probíhá tak, že se každá sekunda spojitého signálu rozdělí na tolik bodů, kolik udává vzorkovací frekvence. V každém bodě se zjistí amplituda ve stejném bodě spojitého signálu a ta se stane jeho hodnotou. Rozdíl mezi spojitým a navzorkovaným signálem je vidět na obrázku 2.2. Aby nedošlo ke ztrátě informace, je nutné dodržet Nyquist-Shannonův vzorkovací teorém, podle kterého musí být nejvyšší frekvence přítomná v signálu maximálně polovina vzorkovací frekvence [28]. Nejčastěji používané vzorkovací frekvence jsou 44 100 Hz a 48 000 Hz. Nejvyšší frekvence, které lze na těchto vzorkovacích frekvencích reprezentovat bez ztráty informace jsou příslušně 22 050 Hz a 24 000 Hz.

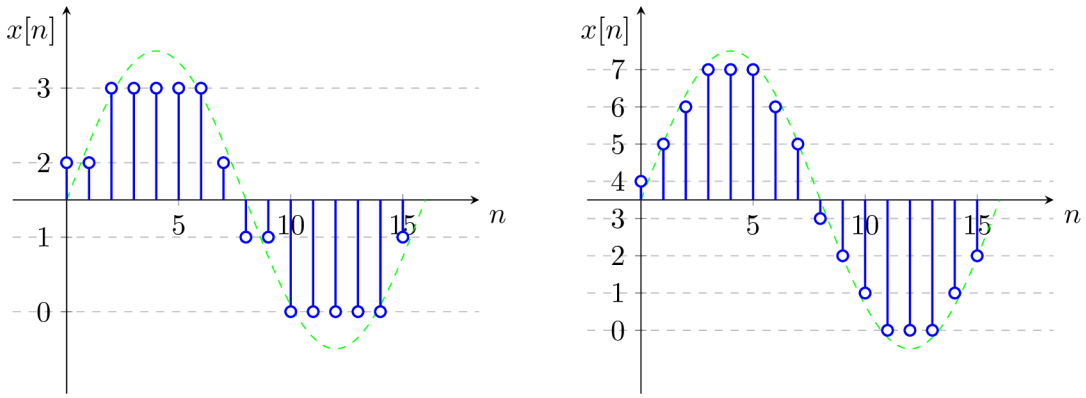
### Kvantování

Pro převedení spojitého signálu na digitální nestačí pouze vzorkování. Vzorkování diskretizuje časovou komponentu spojitého signálu na pevně daný počet vzorků za jednotku času, ale nijak neovlivňuje hodnoty, kterých jednotlivé vzorky mohou nabývat. Kvantování je proces diskretizace hodnot zaokrouhlováním k předem určeným hodnotám [22]. Hodnoty nebo také hladiny, ke kterým se zaokrouhluje, jsou dané počtem použitých bitů. Vyšší počet bitů vede ke zlepšení kvality, protože zaokrouhlené hodnoty jsou blíže jejich původním. Tento jev lze vidět na obrázku 2.3. Proces kvantování vždy zavede do výsledného signálu




Obrázek 2.2: Spojitý signál (vlevo) a navzorkovaný signál (vpravo) vzorkovaný na frekvenci 16 Hz

takzvaný *kvantizační šum* [22], protože převedení reálných čísel na digitální není exaktní. Tento šum je rozdílem původního signálu a kvantovaného signálu. Použití menšího počtu bitů způsobí, že kvantizační šum bude výraznější, ale výsledný signál bude mít menší velikost. Pro kvalitní audio se dnes nejčastěji používá alespoň 16 bitů.




Obrázek 2.3: Signál z obrázku 2.2 vzorkovaný na frekvenci 16 Hz a kvantovaný na 2 bitech (vlevo) a 4 bitech (vpravo)

Posledním krokem pro uložení a přenos digitálních signálů je kódování vzorků. Nejjednodušší způsob pro zakódování kvantovaných hodnot je převod na binární čísla. Jednotlivé jedničky a nuly pak mohou při přenosu reprezentovat přítomnost pulzu (1) a nepřítomnost pulzu (0). Tento systém, který obsahuje vzorkování, kvantování a kódování, se nazývá *pulzně-kódová modulace* (PCM) [22]. Pulzně-kódová modulace je důležitá obecně pro zpracování digitálních signálů, ale i pro účely této práce, protože ji využívá zvukový formát *Waveform Audio File Format*, který je popsán v následující podsekci.

### Formáty digitálního zvuku

Digitálních audio formátů je velké množství, každý se svými odlišnými vlastnostmi. Hlavní kategorie, na které se digitální zvukové formáty dělí, jsou *nekomprimované* a *komprimované*

– ty se dále dělí na formáty se ztrátovou a bezeztrátovou kompresí. Příklady některých známých formátů jsou v tabulce 2.1.

Tabulka 2.1: Tabulka známých zvukových formátů

| Nekomprimovaný | Komprimovaný | |
|---|---|---|
| | Bezeztrátový | Ztrátový |
| WAV | FLAC | MP3 |
| AIFF | ALAC | AAC |
| RAW | | OGG Vorbis |
| | | Opus |
| | | WebM |

Výhodou komprimovaných formátů je úspora datového prostoru. Naopak hlavní nevýhodou komprimovaných formátů je složitější proces kódování a dekódování. Se vzorky komprimovaných formátů není možné přímo pracovat, ale je potřeba je nejprve dekomprimovat a převést do formátu, který umožňuje pracovat se zvukovými daty přímo. Jedním z nekomprimovaných formátů je *Waveform Audio File Format* (dále jen WAV). Jedná se o proprietární formát pro digitální audio vytvořený firmami Microsoft a IBM. Je široce rozšířený a není k němu potřeba žádná licence. Tento formát je nástavbou generického multimediálního formátu *Resource Interchange File Format* (RIFF). Konkrétní typ pulzně-kódové modulace, který WAV formát nejčastěji používá je lineární pulzně-kódová modulace (LPCM) [12] – kvantovací úrovně mají mezi sebou rovnoměrné rozestupy [18]. Může ale používat i jiné typy kódování, jako *IEEE Float* nebo komprimované μ-Law a A-Law [12].

V roce 2020 provedli AlSabhany, Ali, Ridzuan a další [1] systematické srovnání 134 publikací zabývající se digitální zvukovou steganografií a zjistili, že 84% z nich používalo formát WAV jako nosič. Za účelem jednoduššího srovnávání metod bude tato práce pracovat pouze s formátem WAV.

## 2.3 Digitální zvuková steganografie

Poslední dobou se začaly objevovat nové směry založené na steganografických přístupech pro zajištění bezpečnosti a utajení dat, které by v kombinaci s konvenčními bezpečnostními technikami mohly vést k lepším výsledkům [10]. Jedna z konvenčních bezpečnostních technik kombinovaných se steganografií je kryptografie. Zatímco steganografie se zabývá ukrýváním informace, hlavním cílem kryptografie je převést zprávu do formy, která je pro neoprávněné osoby nečitelná [1]. I když se tedy někdo pokusí analyzovat komunikaci pro přítomnost utajené zprávy, není možné obsah rozeznat od náhodného šumu.

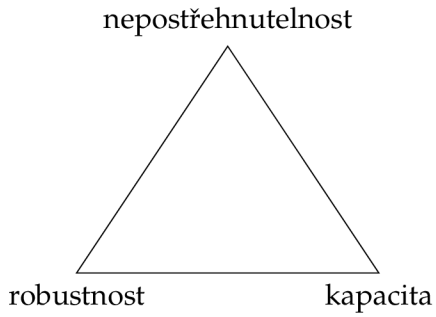
Digitální zvuková steganografie má široké spektrum využití. Mezi praktické příklady patří: monitorování reklam v rádiu, indexování video nahrávek, zachování soukromí jednotlivce, utajená komunikace mezi vojenskými rozvědkami nebo bezpečnostními složkami a použití všude tam, kde není možné použít šifrování [11].

Dalším často používaným pojmem spojeným se steganografií je vodoznak. Vodoznaky se používají pro vložení informací o původu, vlastnictví, či autorských právech dat [10][11][30]. Hlavní vlastností vodoznaku je, že by nemělo být možné jej odstranit bez poškození nosného média. Proto u audia nesmí být součástí hlavičky souboru nebo být

přenášen zvlášť, ale být jeho součástí a musí být nedetekovatelný. Musí být také odolný proti transformaci signálu například při přenosu audia nebo převedení do formátu, který používá ztrátovou kompresi [30]. Velice známým příkladem konvenčních vodoznaků jsou vodoznaky na bankovkách nebo *žluté tečky* v tiskárnové steganografii – známé také jako *yellow dots* – které barevné tiskárny tisknou na každý list. Tyto tečky obsahují zakódované sériové číslo tiskárny a čas tisku [11]. Pokud je vodoznak použit pro ochranu intelektuálního vlastnictví, pak se ještě rozlišuje mezi vodoznakem a otiskem. Vodoznakem se značí všechny objekty stejně, ale otisk se může měnit například pro každého zákazníka pro jeho pozdější identifikaci. To může být užitečné, pokud poruší licenční podmínky například sdílením zakoupené hudby [30].

## 2.4 Vlastnosti a jejich ověřování

Aby bylo možné jednotlivé metody srovnávat mezi sebou, je nutné určit vlastnosti, podle kterých budou hodnoceny. Těch může být mnoho, ale různí autoři steganografických publikací se shodují, že nejvýznamnějšími vlastnostmi jsou *robustnost*, *kapacita* a *nepostřehnutelnost* [1][10][11]. Tyto tři vlastnosti je možné si představit jako trojúhelník s každou vlastností v jednom rohu, jak je vidět na obrázku 2.4. Důvodem je fakt, že není možné dosáhnout vysoké úrovně všech tří vlastností zároveň [11]. Zvýšení kvality jedné z vlastností vždy vede ke snížení kvality alespoň jedné zbývající [1][10]. V následujících podsekcích budou popsány jednotlivé vlastnosti podrobněji. Bude také několikrát použit pojem *stego soubor*, což je soubor, který vznikne použitím určité steganografické metody pro ukrytí zvolené informace do krycího souboru.




Obrázek 2.4: Trojúhelník se třemi nejvýznamnějšími vlastnostmi steganografických metod

### Robustnost

Robustnost indikuje odolnost metody proti úmyslným či neúmyslným modifikacím skrytých informací [1][11]. Míru robustnosti lze hodnotit na základě množství skryté informace, která zůstane nezměněná po různých úpravách stego souboru. Podle [10] mohou tyto modifikace být:

- zesílení amplitudy – může poškodit skryté informace,
- filtrování – skryté informace mohou být odstraněny filtrací určitého pásma,

- překvantování – snížením počtu bitů a zpětné překvantování na původní hodnotu zavede nejen kvantizační šum, ale může být poškozena nebo odstraněna skrytá informace,
- převzorkování – podobná situace jako u překvantování, snížení vzorkovací frekvence způsobí ztrátu informace,
- přidání šumu – šum může skrytá data zamaskovat nebo poškodit,
- překódování – skrytá informace může být poškozena převedením stego souboru do formátu, který používá ztrátovou kompresi.

Robustnost metody je obzvlášť důležitá pro tvorbu vodoznaku popsaného v podkapitole 2.3. Vodoznak musí být schopen odolat výše zmíněným modifikacím, aby bylo možné správně identifikovat jeho vlastníka [30].

### Kapacita

Kapacita udává množství informace, které je možné do nosiče zakódovat a úspěšně dekódovat [11][10]. Kapacitu metody je možné vyjádřit v bitech za sekundu ($bps$ nebo také $b/s$) nebo jako procentuální poměr velikosti skryté informace k velikosti nosiče [1][11]. Výhodou zvukových steganografických metod je větší velikost souborů, se kterými metody pracují, což umožňuje ukrytí většího množství informací. Zároveň je možné ukládat informace do nevyužitých polí v hlavičkách souborů [11].

### Nepostřehnutelnost

Tato vlastnost vyjadřuje podobnost stego souboru a krycího souboru a je spojená s detekovatelností přítomnosti skryté informace [1]. Úroveň nepostřehnutelnosti je možné měřit pomocí *perceptual evaluation of speech quality* (PESQ) testu. Hodnota 4,5 znamená, že naměřený stego soubor je totožný s krycím souborem. Hodnota 1 indikuje maximální degradaci kvality. Další způsob pro porovnání rozdílů dvou signálů je *segmentový poměr signálu k šumu* (SegSNR) [10]. V již zmíněné studii z roku 2020 [1] zjistili autoři, že nejpoužívanější metodou napříč analyzovanými publikacemi byl globální *poměr signálu k šumu* (SNR) a dále *špičkový poměr signálu k šumu* (PSNR).

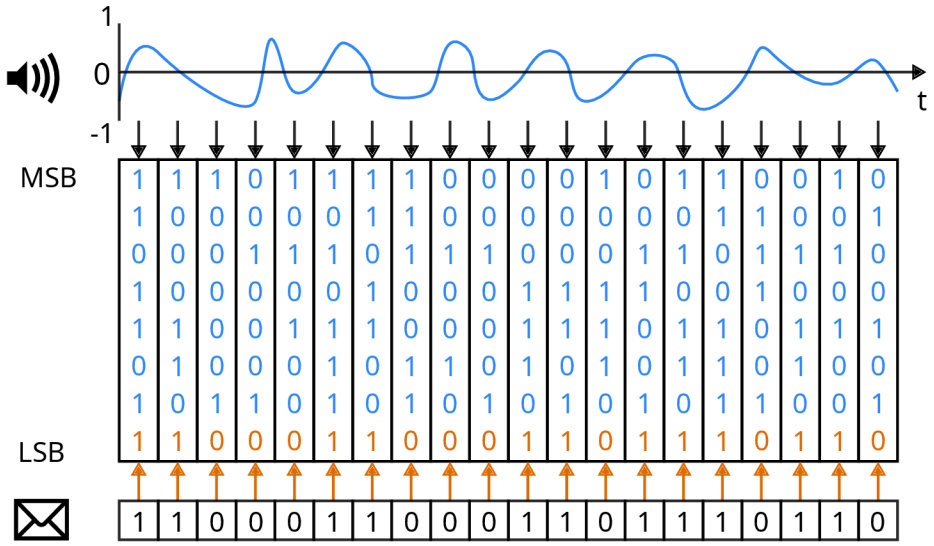
Zvukové steganografické metody mají tu nevýhodu, že lidské zvukové ústrojí je velmi citlivé, což ztěžuje ukrývání informací nepostřehnutelně. Přesto má zvukové ústrojí nedostatky, které je možné využít, jako například maskování tónu, který následuje po hlasitém impulzu. Dalším příkladem je maskování informace, která je zakódovaná ve frekvenci zvuku, který se nachází poblíž [11].

## 2.5 Existující řešení

V této podkapitole bude popsáno několik existujících metod digitální zvukové steganografie. Popis každé metody bude obsahovat její princip a silné a slabé stránky. Každý popis bude obsahovat konkrétní způsob jakým je možné zakódovat a dekódovat informace danou metodou. Na závěr popisu některých metod budou diskutovány možnosti modifikace metody pro zlepšení některé ze steganografických vlastností.

### Metoda nahrazení nejméně významného bitu

Metoda využívající nahrazení nejméně významného bitu (anglicky *least significant bit substitution*, dále jen LSB) je nejjednodušší metodou pro vkládání informací do nosiče. Principem této metody je nahrazení nejméně významného bitu v nosiči za bit ukrývané informace [11]. Princip kódování pomocí této metody je vizualizován na obrázku 2.5.




Obrázek 2.5: Ukládání informačních bitů do nejméně významného bitu krycího signálu. Upraveno z [10].

Tato metoda se vyznačuje velmi vysokou kapacitou [10] a jednoduchou implementací. Kapacita této metody je přímo úměrná počtu a bitové hloubce vzorků nosiče. Je také vhodná pro kombinaci s dalšími steganografickými technikami [10], jako je šifrování nebo komprese. Hlavní nevýhodou této metody je její nízká robustnost. Informace ukryté LSB metodou je velmi jednoduché poškodit triviálními úpravami stego souboru jako je například zesílení, filtrace, přidání šumu, ztrátová komprese [10], převzorkování nebo překvantování. Robustnost této metody a odolnost proti některým z těchto modifikací je možné zvýšit vkládáním do jiné bitové úrovně než je poslední [10].

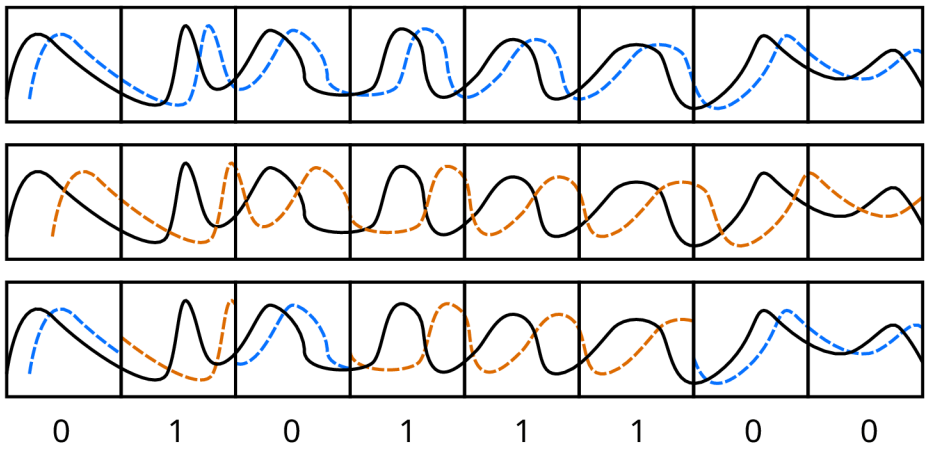
Nemodifikovaná LSB metoda není příliš bezpečná, neboť je možné jednoduše vyčíst hodnoty určité bitové úrovně a tím odhalit ukryté informace [10]. Pro lepší ukrytí informací je možné ukládat bity do náhodných vzorků a náhodných úrovní v nich [11]. Bezpečnost LSB metody je možné zvýšit šifrováním ukrývaných informací. Ukrývaná data je vhodné nejprve zkomprimovat například Lempel-Ziv-Welch (LZW) kompresí. Poté je možné je zašifrovat vybraným šifrovacím algoritmem. Vhodným algoritmem pro symetrické šifrování je *Advanced Encryption Standard* (AES, česky standard pokročilého šifrování). Pro asymetrické šifrování je vhodný algoritmus Rivest-Shamir-Adleman (RSA) [11].

Kapacitu LSB metody je možné zvýšit například ukládáním do více bitových úrovní za sebou v jednotlivých vzorcích. Tato modifikace však produkuje šum, který je lidské ucho schopno zachytit. Lepším způsobem, jak zvýšit kapacitu, je ukládání do 4 nejnižších bitových úrovní pomocí algoritmu *nahrazení nejmenší chybou* (anglicky minimum-error replacement) a rozprostření chyby do následujících 4 vzorků. Tato modifikace zvyšuje kapacitu LSB metody o 33%, zatímco poměr signálu k šumu je nižší než u nemodifikované metody se stejným počtem použitých bitů [7].

### Metoda skrývání pomocí ozvěny

Steganografická metoda ukrývání informací pomocí ozvěny (anglicky *echo hiding*) funguje na principu přidání ozvěny do krycího média [10]. Tato metoda má vyšší robustnost vůči filtrování, převzorkování, či ztrátové kompresi. Lidské sluchové ústrojí má široký dynamických rozsah výkonu a frekvence, ale přesto je možné využít nedostatků, jako je maskování tichých zvuků hlasitějšími [14]. Nevýhodou této metody je zkreslení způsobené přidáním ozvěny do krycího média, které je srovnatelné s rozdílem poslechu audia ve sluchátkách a přes reproduktory, kde se zvuk odráží od stěn v místnosti. Při vhodném nastavení parametrů je možné docílit výborné nepostřehnutelnosti [14]. Těmito parametry jsou indexy zpoždění, amplituda ozvěny a míra exponenciálního dozvuku. Tato metoda má dostatečnou míru robustnosti a je tedy vhodná pro zakódování vodoznaku do krycího média. Vhodné nastavení parametrů docílí velmi nízké pravděpodobnosti odhalení či odstranění vodoznaku neoprávněnou osobou [14]. Po přidání ozvěny si stego médium udrží stejné statistické a percepční vlastnosti [10].

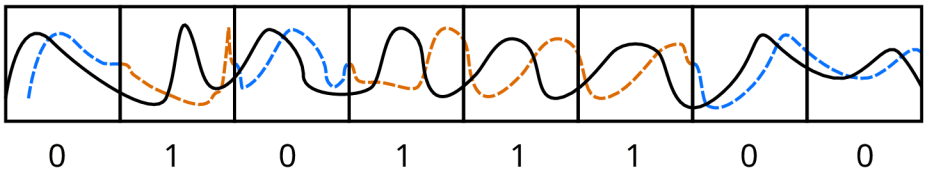
Pro implementaci této metody se krycí médium rozdělí na segmenty do kterých se bity ukrývané informace zakódují vytvořením ozvěny daného segmentu. Pro rozlišení hodnoty bitu se použije jiná míra zpoždění pro logickou 1 a logickou 0. Kódování probíhá konvolucí segmentu a konvolučním jádrem pro příslušný bit. Jednotlivé segmenty se následně spojí a přidají ke krycímu médiu [14]. V praxi je ale výhodnější provést konvoluci krycího média s oběma konvolučními jádry a použít bity ukrývané informace jako masku, která určí jaké segmenty ponechat a přidat ke krycímu médiu [14]. Pro snížení míry zkreslení na rozmezí segmentů je vhodné vyhladit změnu mezi hodnotou 0 a 1 [31]. Příklad ozvěn, které vniknou použitím masky bez vyhlazení přechodů mezi segmenty jsou na obrázku 2.6. Příklad ozvěn, které vniknou použitím masky s vyhlazením přechodů jsou na obrázku 2.7.




Obrázek 2.6: Ozvěny vzniklé konvolucí s konvolučními jádry pro jednu ozvěnu bez vyhlazení.

#### Ozvěna s jedním konvolučním jádrem na bit

Tuto metodu poprvé představil Daniel Gruhl a Walter Bender v roce 1996 [14]. Používá dvě konvoluční jádra dané rovnicí 2.2, kde $\delta$ je Kroneckerova delta funkce, $d_i$ je zpoždění pro bit logické 0 a logické 1 a $\alpha$ je amplituda ozvěny [11].




Obrázek 2.7: Ozvěny vzniklé konvolucí s konvolučními jádry pro jednu ozvěnu s vyhlazením přechodů mezi segmenty.

$$h_i[n] = \delta[n] + \alpha\delta[n - d_i], i \in 0, 1 \tag{2.2}$$

Zpožděné signály vniknou konvolucemi krycího média s konvolučními jádry dané rovnicí 2.3, kde $s[n]$ je krycí médium a $h_i[n]$ je konvoluční jádro z rovnice 2.2.

$$k_i[n] = s[n] * h_i[n] \tag{2.3}$$

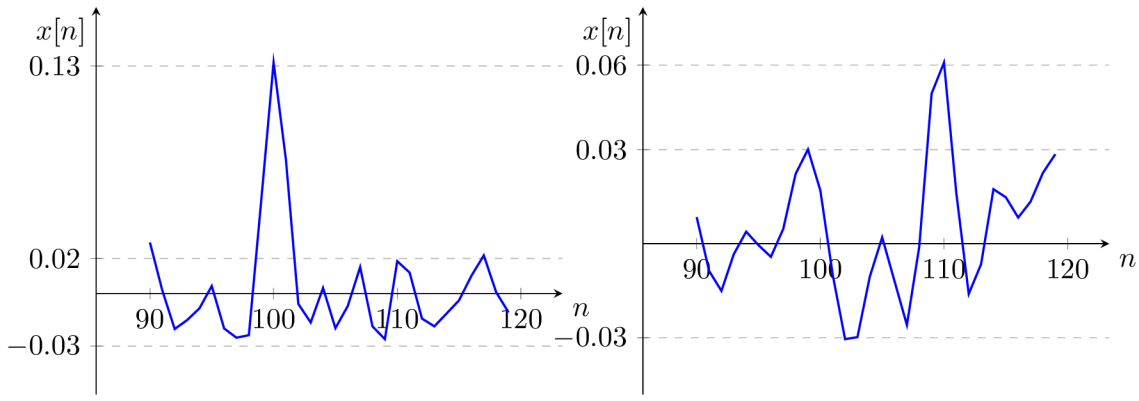
Příklad ozvěn, které vniknou konvolucemi krycího média s konvolučními jádry jsou na obrázku 2.6.

### Dekódování

K dekódování této metody se používá cepstrální analýza. Stego signál se rozdělí na stejný počet segmentů jako při kódování. Poté se vypočítá jejich cepstrum, které je definováno jako inverzní Fourierova transformace ($\mathcal{F}^-1$) logaritmu absolutní hodnoty Fourierovy transformace ($\mathcal{F}$) signálu [31]. Vzorec pro získání cepstra signálu $x(t)$ je definován rovnicí 2.4.

$$C = \mathcal{F}^-1(\log|\mathcal{F}(x(t))|) \tag{2.4}$$

Po výpočtu cepstra segmentu je možné zjistit hodnotu zakódovaného bitu kontrolou, která hodnota na pozici $d_0$ a $d_1$ je vyšší. Pokud je hodnota $C(d_0)$ vyšší než $C(d_1)$, pak je hodnota bitu daného segmentu logická 0. A pokud je hodnota $C(d_1)$ vyšší než $C(d_0)$, pak je hodnota bitu daného segmentu logická 1 [14]. Příklad cepstra segmentu včetně špiček na indexech zpoždění 100 a 110 lze vidět na obrázku 2.8.




Obrázek 2.8: Přiblížená cepstra segmentů. Levé kóduje logickou 0. Pravé kóduje logickou 1.

Nevýhodou při dekódování této metody je například falešná detekce, pokud se ve zvuku vyskytuje ozvěna, která nebyla vytvořena touto metodou. Dále může být těžké

detekovat informaci, pokud je amplituda ozvěny příliš malá a špičky na indexech zpoždění budou proto splývat s okolím. Zesílení amplitudy zlepší míru detekce, ale zvýší také náchylnost na neoprávněnou detekci [16].

#### Modifikace metody skrývání pomocí ozvěny

Pro zvýšení robustnosti a nepostřehnutelnosti použili autoři Oh, Seok, Hong a Youn [21] konvoluční jádro, které nejprve vytvoří ozvěnu se zápornou a poté s kladnou amplitudou (anglicky bipolar echo). Zjistili, že tato modifikace má vyšší míru robustnosti i po překódování do ztrátového formátu MP3. Dekódování této modifikace spočívá v detekování špiček v autokorelaci výkonového cepstra. Výkonové cepstrum je podobné normálnímu cepstru a je definováno rovnicí 2.5.

$$C_p = \mathcal{F}^{-}1(\log|\mathcal{F}(x(t))|^2) \tag{2.5}$$

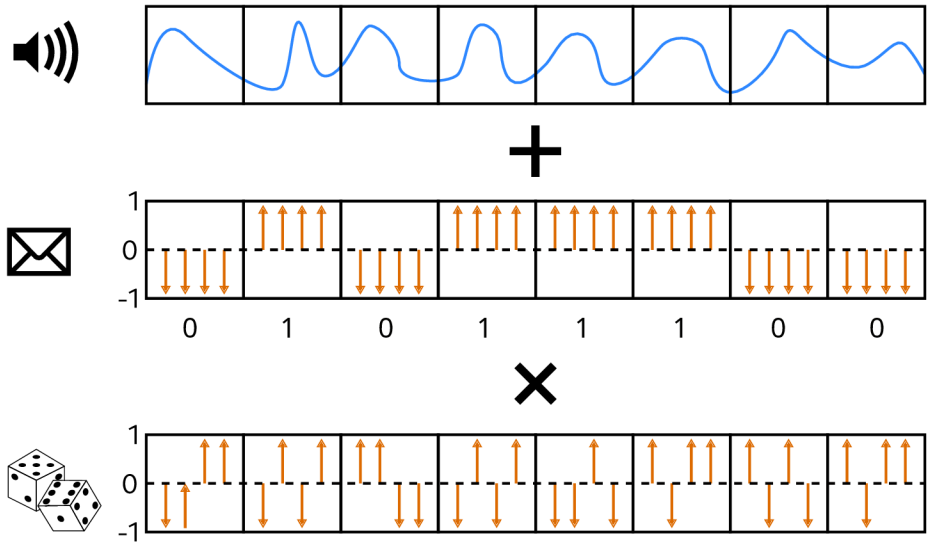
O dva roky později publikovali autoři Kim a Choi [16] způsob, jak znovu zvýšit robustnost pomocí konvolučního jádra se dvěma ozvěnami posunutými vpřed i vzad (anglicky backward-forward echo). Později publikovali autoři Wu a Chen [33] modifikaci ozvěnové metody kombinující již popsané modifikace. Ta používá konvoluční jádro pro vytvoření celkem čtyř ozvěn. Dvě jsou posunuté vzad a dvě vpřed přičemž obě strany mají ozvěnu s kladnou a zápornou amplitudou. Autoři této modifikace ji porovnávali s výše popsanými variantami této metody a ve všech testech měla vyšší poměr úspěšně dekódovaných bitů a zároveň také vyšší poměr signálu k šumu než ostatní varianty.

### Metoda rozprostřeného spektra

Metoda rozprostřeného spektra byla nejprve vytvořena jako metoda pro zvýšení spolehlivosti přenosu dat a zajištění doručení informací. Ze stejných důvodů se rozprostření spektra používá i ve steganografii. Princip této metody spočívá v opakování informačních bitů $n$-krát podle čipové rychlosti (anglicky *chip rate*). Ta určuje, kolika bity bude kódován každý informační bit. [1]. Dále popisované variantě této metody se říká přímé rozprostřené spektrum (anglicky *direct sequence spread spectrum*).

Kódování začíná převedením ukrývaných dat z binárního formátu na čipy. Čip je impulz, který používá hodnotu *-1* místo binární hodnoty *0* a hodnotu *1* používá beze změny [17]. Poté je každý bit vynásoben pseudonáhodnou čipovou sekvencí $n$-krát, čímž dojde k rozprostření informačních bitů přes více přenesených bitů. Zvýšení počtu kopií bitů zvýší robustnost a zároveň sníží kapacitu této metody [1]. Výsledné stego médium vznikne přidáním vynásobených hodnot ke krycímu médiu [17]. Informace zakódovaná touto metodou je odolná proti šumu, protože obsahuje přebytečné kopie bitů. V případě poškození části stego média je možné z přebytečných bitů obnovit ukryté informace [10]. Popsaný princip kódování je možné vidět na obrázku 2.9, kde se ke krycímu signálu přidají rozprostřené bity informace vynásobené pseudonáhodnou sekvencí čipů.

Tato metoda je dostatečně robustní pro přidání více vodoznaků. Například pro přidání více autorů k hudební skladbě. Při detekci konkrétního vodoznaku jsou ostatní považovány jako šum. Dále je tato metoda dostatečně robustní vůči přidanému šumu a překódování do ztrátového formátu [5]. Aby byl náhodný šum přidaný touto metodou neslyšitelný, je nutné nastavit při kódování amplitudu rozprostřené sekvence zhruba na 0,5% dynamického rozsahu krycího média [4]. Pro kvalitní dekódování má tato metoda kapacitu 4 bity za sekundu [4].




Obrázek 2.9: Kódování rozprostřených bitů (uprostřed) pomocí metody přímého rozprostření spektra vynásobením s pseudonáhodnou sekvencí (dole) a přidáním ke krycímu signálu (nahoře).

Pro dekódování informačních bitů je nezbytné, aby příjemce měl stejnou pseudonáhodnou sekvenci čipů, která byla použita k zakódování. Pro dekódování se stego médium a pseudonáhodná sekvence rozdělí na segmenty a vypočítá se korelace mezi každým segmentem stego média a pseudonáhodné sekvence. Pro určení hodnoty bitu je možné použít rozhodovací pravidlo v rovnici 2.6, kde $b$ je hodnota bitu a $c$ je výsledek korelace [17].

$$b = \begin{cases} 0, & c < 0 \\ 1, & c > 0 \end{cases} \tag{2.6}$$

### Metoda fázového kódování

Metoda fázového kódování je založená na nahrazení fáze prvního segmentu audia referenční fází reprezentující ukrývaná data. Fáze následujících segmentů je upravena tak, aby byla zachována relativní fáze mezi segmenty. Metoda fázového kódování dosahuje nejlepších hodnot poměru signálu k šumu. Avšak příliš velká změna fáze frekvenční komponenty vede k rozptylu fáze a tím ke zhoršení kvality zvuku. Dokud jsou ale změny fáze dostatečně malé, což je subjektivní, pak je možné dosáhnout neslyšitelného kódování [4].

Kódování pomocí této metody začíná rozdělením krycího audia na segmenty s délkou počtu ukrývaných bitů. Poté je každý segment převeden do frekvenční domény pomocí Fourierovy transformace. Z transformovaného signálu je možné vytvořit matici fází – značeno $\phi_n(\omega_k)$ – a matici amplitud segmentů – značeno $A_n(\omega_k)$. Písmeno $n$ značí index segmentu a písmeno $k$ značí index vzorku uvnitř segmentu. Poté se vypočítá rozdíl fází segmentů napříč řádky [4]. Matematický zápis rozdílu fází je v rovnici 2.7.

$$\Delta\phi_n(\omega_k) = \phi_{n+1}(\omega_k) - \phi_n(\omega_k) \tag{2.7}$$

Poté se bity ukrývaných dat převedou do formátu $\frac{\pi}{2}$ reprezentující hodnotu 0 a $-\frac{\pi}{2}$ reprezentující hodnotu 1. Touto sekvencí se nahradí první segment v matici fází. Poté se postupně počítají nové fáze z původní fázové matice a matice rozdílů fází. Matematický

zápis je v rovnici 2.8, kde $\phi'$ značí novou matici fází a $N$ značí index posledního segmentu [4].

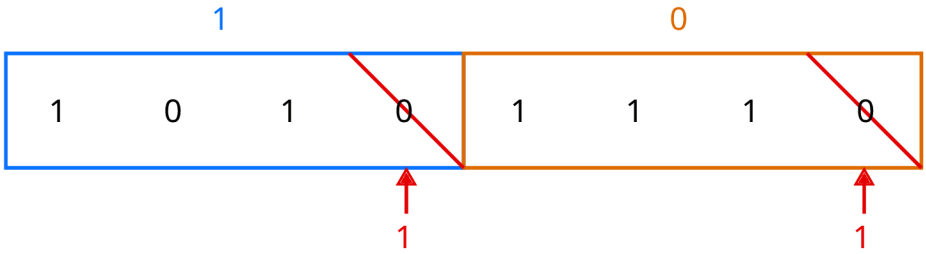
$$\begin{bmatrix} \phi'_1(\omega_k) & = & \phi'_0(\omega_k) & + & \Delta\phi_1(\omega_k) \\ \phi'_n(\omega_k) & = & \phi'_{n-1}(\omega_k) & + & \Delta\phi_n(\omega_k) \\ & \vdots & & & \\ \phi'_N(\omega_k) & = & \phi'_{N-1}(\omega_k) & + & \Delta\phi_N(\omega_k) \end{bmatrix} \tag{2.8}$$

Na závěr je možné sestavit stego signál z amplitud a nových fází pomocí inverzní Fourierovy transformace [4]. Z obou matic je potřeba zkombinovat hodnoty amplitud a fází v jednotlivých segmentech pro vytvoření komplexních čísel v exponenciálním tvaru. Na každý segment se poté aplikuje inverzní Fourierova transformace. Tyto segmenty se znovu spojí dohromady, čímž vznikne výsledné stego médium. Dekódování probíhá pouhým převedením fází z prvního segmentu stego média na bity stejným způsobem jako při kódování.

Kapacita této metody se pohybuje v rozsahu 8 bitů za sekundu až 32 bitů za sekundu podle množství šumu přítomného v krycím médiu. Více šumu vede k vyšší kapacitě [4]. Existují modifikace této metody, které využívají jiné fáze krycího signálu. Například vyšší frekvence jsou hůře detekovatelné lidským sluchovým ústrojím [9]. Autoři [9] vybírají pro kódování pouze frekvenční regiony s vysokou energií.

### Metoda paritního kódování

Metoda paritního kódování je založená na ukrývání do nejméně významného bitu skupiny vzorků na rozdíl od ukrývání do jednotlivých vzorků. Pro zakódování informace se krycí médium rozdělí na segmenty a vypočítá se paritní bit segmentu. Pokud je paritní bit stejný jako informační, pak se neprovádí nic. Pokud je paritní bit rozdílný, pak se změní nejméně významný bit libovolného vzorku v daném segmentu. Díky tomu je možné snížit detekovatelnost informace [3], čímž ale dojde ke snížení robustnosti. Způsob kódování pomocí této metody je vizualizován na obrázku 2.10.



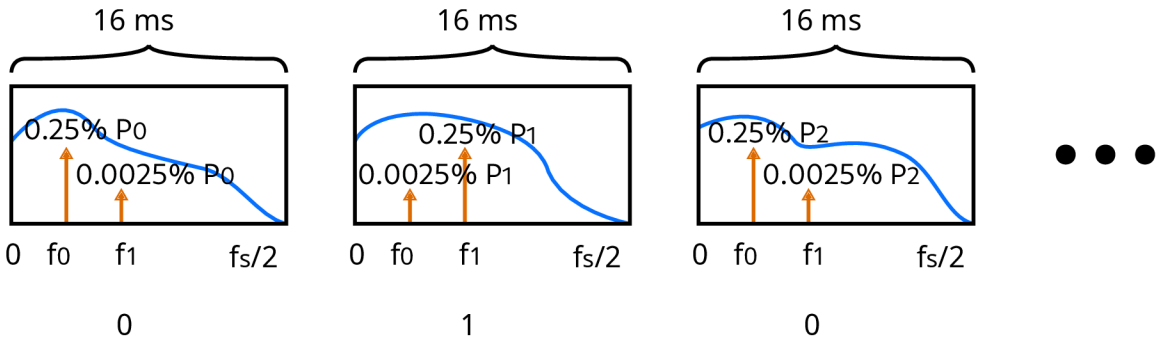

Obrázek 2.10: Metoda paritního kódování. Pro zakódování je potřeba docílit stejné parity jako je informační bit.

### Metody založené na nedostatcích lidského sluchového ústrojí

Tyto metody byly využívány již od počátků steganografie a využívají percepčních vlastností lidského sluchového ústrojí a psychoakustických vlastností řeči. Tyto metody využívají maskování v časové nebo frekvenční doméně. Maskování v časové doméně využívá nutnosti chvíli počkat po zaslechnutí hlasitého tónu. Pro ukrytí určitého tónu je možné vložit

jej do bezprostřední blízkosti hlasitého impulzu. Nápodobně, maskování ve frekvenční doméně probíhá vložením frekvence, která se nachází v blízkém okolí jiné frekvence s vyšší amplitudou [11].

Jedna z metod pracující ve frekvenční doméně je metoda vkládání tónu [11]. Ta spoléhá na neschopnost vnímání tónu v přítomnosti mnohem hlasitějších. Pro kódování se krycí médium rozdělí na segmenty a vypočítá se výkon každého z nich. Poté se vloží dva tóny na frekvencích $f_0$ a $f_1$. Výkony tónů jsou nastaveny v předem známém poměru k výkonu segmentu [10]. Princip kódování této metody lze vidět na obrázku 2.11, kde poměr výkonů tónu na frekvencích $f_0$ a $f_1$ kóduje bity `010`.




Obrázek 2.11: Metoda vkládání tónu. Specifické poměry tónů na frekvencích $f_0$ a $f_1$ kódují 0 nebo 1.

Dekódování probíhá porovnáním poměrů výkonu segmentu k výkonu tónu na dané frekvenci [10]. Dekódovací pravidlo lze vidět v rovnici 2.9, kde $b$ značí hodnotu bitu a výkon segmentu je označen jako $P_i$, přičemž $i$ značí index segmentu a $f_0$ a$f_1$ značí použité frekvence.

$$b = \begin{cases} 0, & \frac{P_i}{P_{f_0}} > \frac{P_i}{P_{f_1}} \\ 1, & \text{jinak} \end{cases} \tag{2.9}$$

Tato metoda má malou kapacitu a vložené tóny je jednoduché detekovat, proto je vhodné použít čtyři a více párů frekvencí, které se střídají na základě určitého klíče [10].

### Metoda ukrývání v úsecích ticha

Tato metoda využívá ke skrývání informací úseky ticha nacházející se v nahrávkách řeči [10]. Pro zakódování informací je nejprve potřeba najít úseky ticha v nahrávce. Ty se vyskytují jako posloupné sekvence vzorků s amplitudou nižší než určitá hodnota. Z nalezených úseků se ponechají pouze ty, jejichž délka je větší než minimální délka úseku [29]. Autoři [29] zjistili následující optimální hodnoty:

- maximální amplituda úseku: 15% maximálního rozsahu signálu,
- minimální délka úseku: 200 vzorků; kratší úseky se většinou vyskytují mezi částmi slov a tyto skoky mohou být detekovány,
- maximální počet bitů pro ukrytí do segmentu: 4 b.

Pokud $X$ je kódovaná hodnota, $N$ je počet bitů kódované hodnoty a $L$ je délka konkrétního segmentu, pak pro kódování do konkrétního segmentu se vypočítá jeho nová délka $D$, tak aby platil vztah v rovnici 2.10 [29].

$$X = D \bmod 2^N \tag{2.10}$$

Novou délku segmentu lze jednoduše vypočítat z rovnice 2.11.

$$D = L - [(L - X) \bmod 2^N] \tag{2.11}$$

Pokud by byla nová délka $D$ kratší než je minimální stanovená délka úseku, pak je potřeba použít jiný úsek. Pro dekódování informace stačí použít zmíněný vztah 2.10 [29].

Nevýhodou této metody je její citlivost na transformaci stego signálu. Změny délek úseků ticha povedou k dekódování špatných hodnot. Pro zvýšení robustnosti v tomto ohledu je možné snížit amplitudu úseků ticha a zvýšit amplitudu okolních úseků, aby nedošlo ke změně délky některého z úseků například kvůli kompresi [10].

### Metoda využívající vlnkové transformace

Nejlepší metodou pro ukrývání informací v transformační doméně je diskrétní vlnková transformace (anglicky *discrete wavelet transform*, dále jen DWT), protože prezentuje informace o frekvencích, které lze získat z Fourierovy transformace jako funkci v čase. Vlnková transformace poskytuje dobrý kompromis mezi časovou doménou signálu a frekvenční doménou, protože Fourierova transformace frekvencím nepřiřazuje čas. Další transformační funkcí je diskrétní kosinová transformace (anglicky *discrete cosine transform* – DCT). Ta není pro účely ukrývání informací příliš vhodná, protože zavádí do audia artefakty. Metody v transformační doméně jsou poměrně méně odolné proti šumu [11].

Pro zakódování informace do nosiče pomocí DWT se nosič rozdělí na segmenty po 512 vzorcích. Na každý segment je pětkrát aplikována dekompozice na nízkofrekvenční a vysokofrekvenční komponenty [8]. Každá následující dekompozice je aplikována na nízkofrekvenční složku z předchozí úrovně [25]. Výsledných 512 koeficientů je převedeno na binární hodnoty do jejichž nejméně významných bitů jsou uloženy bity ukrývané informace [8]. Experimentální výsledky v [8] ukázaly, že při modifikaci až 7 nejnižších bitů bylo téměř nemožné rozeznat rozdíl mezi originální a upravenou nahrávkou. A i při modifikaci více úrovní byly výsledky lepší než při použití konvenční metody nahrazení nejméně významného bitu popsané v podsekci 2.5 s dvakrát menším počtem upravených bitových úrovní. Tato metoda má velmi vysokou kapacitu a je schopna ukrýt až o 220,5 kb/s více dat než konvenční metoda nahrazení nejméně významného bitu [8].

Proces dekódování probíhá podobně jako kódování. Stego signál se rozdělí na segmenty, provede se dekompozice a koeficienty se převedou na binární hodnoty. Z nejnižších bitů je možné vyčíst bity ukryté informace [24].

# Kapitola 3

# Analýza a návrh

V této kapitole bude popsán řešený problém a konceptuální návrh řešení. Bude popsána struktura softwarové knihovny a vestavěného programu pro zvukovou steganografii a vysokoúrovňový popis struktury projektu, použitých technik a metod k tvorbě kvalitního a rozšiřitelného řešení. Budou také popsány použité softwarové nástroje a na závěr kapitoly bude popsán princip navrhované metody pro digitální zvukovou steganografii.

## 3.1 Rozbor řešeného problému

Hlavním problémem v tomto oboru je nedostatek kvalitně implementovaných řešení, přestože je množství literatury poměrně bohaté. Velké množství existujících řešení je zastaralých a často nepodporují volbu steganografické metody.

### Výhody a nevýhody stávajících řešení

V celém oboru steganografie je těžké najít kvalitní existující řešení. Většina existujících řešení má dva společné problémy: málo implementovaných metod a špatnou dokumentaci. Zde je seznam nalezených řešení a krátký popis každého z nich:

- *Steghide*[1] – Jedno z lepších nalezených řešení. Jedná se o svobodný software s GPL licencí, podporuje šifrování pomocí algoritmu *AES-128* a používá grafový algoritmus pro kódování. Program je ale velmi zastaralý a nepodporuje jiné metody kódování.
- *Hide4PGP*[2] – Další otevřený software s podporou šifrování pomocí *OpenPGP*[3]. Kódování rozprostírá informační bity přes celý soubor.
- *Xiao Steganography*[4] – Proprietární program s podporou pěti šifrovacích a čtyř hešovacích algoritmů, které jsou avšak velmi zastaralé a jejich používání se nedoporučuje.
- *audio-steganography-algorithms*[5] – Sada implementací metod pro program MATLAB. Obsahuje metodu rozprostřeného spektra (viz podsekce 2.5), ukrývání pomocí ozvěny

[1] https://steghide.sourceforge.net
[2] http://www.heinz-repp.onlinehome.de/Hide4PGP.htm
[3] https://www.openpgp.org
[4] https://download.cnet.com/Xiao-Steganography/3000-2092_4-10541494.html
[5] https://github.com/ktekeli/audio-steganography-algorithms

(viz podsekce 2.5), nejméně významného bitu (viz podsekce 2.5) a fázového kódování (viz podsekce 2.5). Bohužel jsou modifikace metod v nečitelném proprietárním formátu.

- *Steganography*[6] – Program v jazyce *Python*. Pro audio soubory ve formátu *WAV* podporuje pouze metodu nahrazení nejméně významného bitu.
- *Audio steganography - Phase encoding*[7] – Ukázkový program v jazyce Python pro kódování pomocí metody fázového kódování.
- *py-stego-phase*[8] – Další program v jazyce Python pro metodu fázového kódování.
- *InvisibleSecrets*[9] – Proprietární program s podporou šifrování.
- *StegoStick*[10] – Svobodný software s podporou šifrování, avšak zastaralého.
- *DeepSound*[11] – Proprietární program s podporou šifrování. Šifrovací algoritmus je současně nejvíce doporučovaný algoritmus *AES-256*.

## 3.2 Návrh řešení

Řešením problému je vytvoření softwarové knihovny a programu, která bude implementovat steganografické metody popsané v sekci 2.5. Aby bylo řešení jednoduše rozšiřitelné, bude použit objektově orientovaný návrh. Každá ze steganografických metod bude mít společné rozhraní, aby bylo použití a implementace nových metod co nejjednodušší. Dalším požadavkem na řešení je, aby bylo čitelné a jednoduše pochopitelné. Musí zde být však určitý kompromis mezi čitelností a optimalizací kódu. Řešení by mělo být relativně rychlé a proto by kódování mělo probíhat v řádech několika sekund až minut.

Při volbě tajné zprávy bude mít uživatel programu na výběr textový vstup nebo obsah souboru. Rozhraní jednotlivých metod bude pracovat s bity ukrývané informace, díky čemuž nebude záležet na typu kódovaných dat. Pro kódování je tedy postačí převést z bajtů na bity a při dekódování naopak. Většina steganografických metod má několik parametrů které mají vliv na robustnost, kapacitu a nepostřehnutelnost metody. Tyto parametry budou uživateli zpřístupněny jako dodatečné argumenty při spouštění programu nebo při kódování a dekódování v knihovní části této práce.

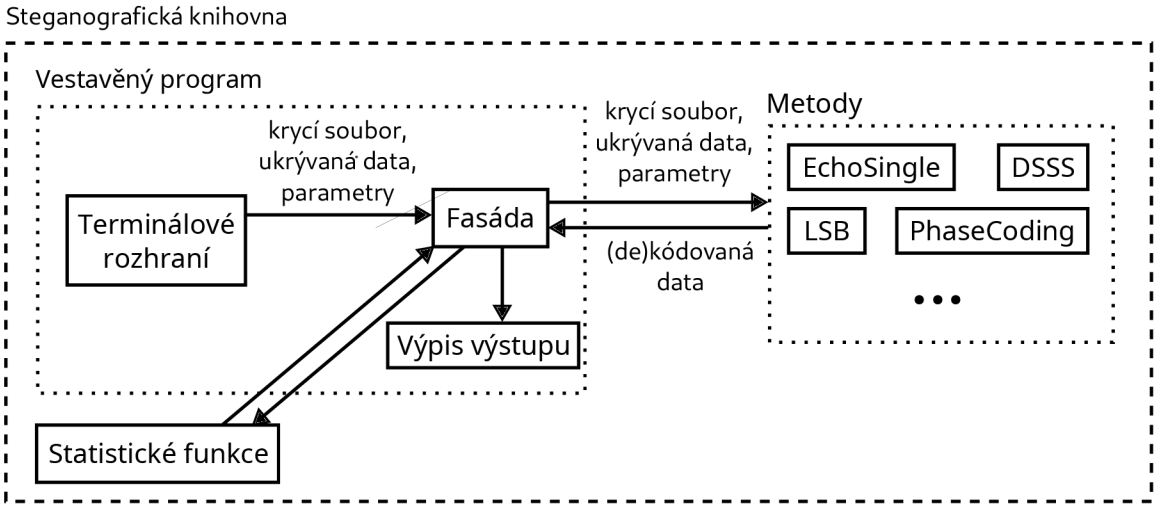
Z metod popsaných v sekci 2.5 jsem plánoval implementovat všechny, avšak některé metody se ukázaly být velmi složité na implementaci. Implementována nebude metoda vlnkové transformace a také metoda paritního kódování, protože se jedná prakticky o modifikaci metody nahrazení nejméně významného bitu. U metod vybraných k implementaci budou také implementovány některé popsané modifikace metod.

### Blokové schéma a komponenty systému

Struktura řešení se bude skládat z několika modulů a balíčků. Pro zjednodušení, modul je jeden soubor a balíček je kolekce modulů. Jádrem knihovny bude balíček s moduly im-

[6] https://github.com/ragibson/Steganography
[7] https://github.com/dan-oak/secret_in_wav
[8] https://github.com/Galarius/py-stego-phase
[9] https://www.east-tec.com/invisiblesecrets
[10] https://sourceforge.net/projects/stegostick
[11] https://jpinsoft.net/DeepSound/Overview.aspx

plementovaných metod pro digitální zvukovou steganografii. Druhý hlavní balíček bude obsahovat moduly týkající se vestavěného programu s terminálovým rozhraním. Nejdůležitějším modulem pro programovou část bude modul s fasádou, která zjednoduší volání jednotlivých metod při použití z terminálu. Bude také poskytovat výpočty různých statistických funkcí na základě kterých bude možné vyhodnotit pro konkrétní krycí soubor úroveň jednotlivých vlastností steganografických metod popsaných v sekci 2.4. Blokové schéma popsané struktury je možné vidět na obrázku 3.1.




Obrázek 3.1: Blokové schéma knihovny pro digitální zvukovou steganografii

### Realizační prostředky a metody

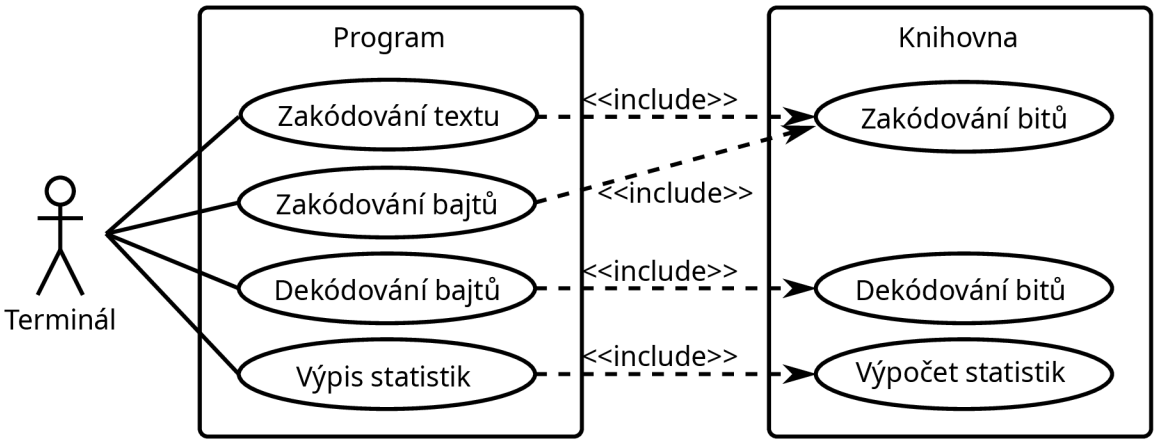
K implementaci navrženého řešení jsem zvolil programovací jazyk *Python* minimální verze *3.8*. Tento jazyk jsem vybral, protože se jedná o vysokoúrovňový jazyk s dobře čitelnou syntaxí. Je také dynamicky typovaný a umožňuje kombinovat objektově orientovaný a funkcionální styl programování, díky čemuž je možné psát elegantní kód. Další kvalitou je dobré rozhraní s jazykem *C*, díky kterému mohou být knihovny velmi rychlé. Jednou takovou knihovnou je knihovna *NumPy*[12], která poskytuje optimalizované funkce pracující s vícedimenzionálním polem jménem *NDArray*. Na této datové struktuře a funkcích staví celá řada dalších numerických knihoven, jako například knihovna *SciPy*[13], která pomocí NumPy implementuje velké množství funkcí pro účely zpracování signálů. Další knihovnou, která využívá NumPy je knihovna *pandas*[14], která slouží pro zpracování dat. Tato knihovna bude použita pro zpracování dat vygenerovaných při vyhodnocování vlastností metod.

Kód bude strukturován podobně jako jiné knihovny a balíčky pro jazyk Python. Obecně obsahují Python knihovny primárně: zdrojový kód, konfigurační soubory pro vytvoření balíčku k distribuci, softwarovou licenci a soubor `README.md`, který obsahuje případné prerekvizity a příklady použití či spuštění. Jako softwarovou licenci pro tento projekt jsem zvolil licenci *Apache-2.0*, protože je velmi otevřená – dovoluje i komerční použití a distribuci bez zdrojového kódu – a je také často používaná Python knihovnami.

[12] https://numpy.org
[13] https://scipy.org
[14] https://pandas.pydata.org

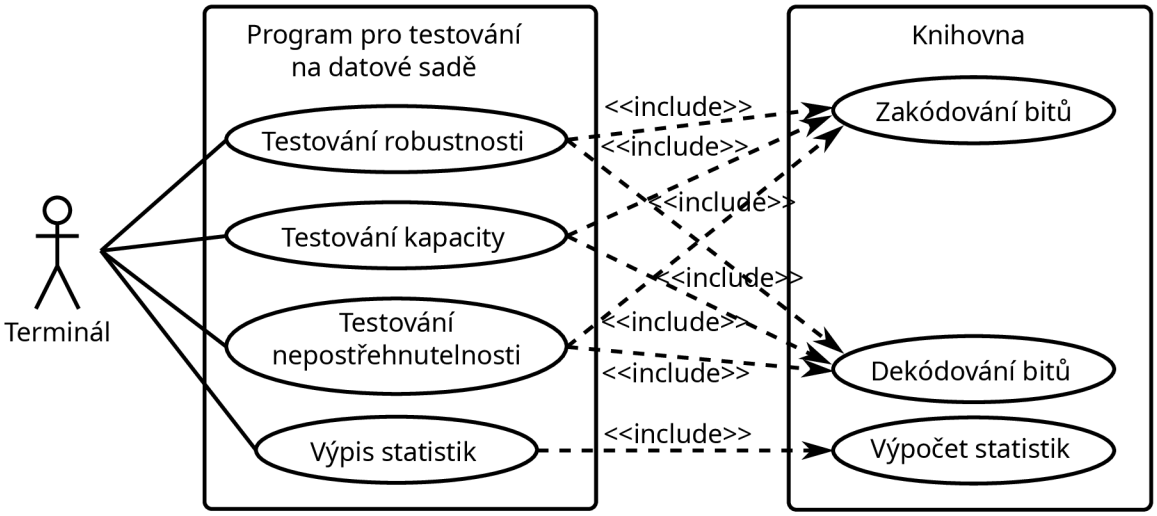
### Případy užití

Program bude umožňovat uživateli zvolit steganografickou metodu a pomocí ní zakódovat a dekódovat tajnou zprávu ve formě textu nebo souboru do krycího audio souboru. Interakci mezi použitím hlavního programu a knihovny je možné vidět na obrázku 3.2.




Obrázek 3.2: Diagram případů užití primárního vestavěného programu a knihovny.

Druhý vestavěný program bude umožňovat testování jednotlivých steganografických vlastností. Diagram případů užití pro druhý program je na obrázku 3.3.




Obrázek 3.3: Diagram případů užití druhého vestavěného programu pro testování steganografických vlastností na datové sadě.

### Ověřování vlastností

Pro ověření vlastností steganografických metod bude součástí knihovny druhý vestavěný program, který bude systematicky testovat jednotlivé vlastnosti každé metody. Testování bude probíhat na často používaných datových sadách pro digitální zvukovou steganografii. Podle [1] jsou to tyto datové sady: *GTZAN* [32], *Noizeus* [15] a *UrbanSound8K* [27]. K těmto datovým sadám jsem ještě přidal vlastní sadu souborů, která bude s ostatními více popsána

níže. Podle [1] patří mezi používané sady také datový korpus *TIMIT* [13], avšak ten zde nebude testován.

Hodnoty jednotlivých vlastností se vyčíslují pomocí statistických funkcí, které se často objevovaly v průzkumu [1]. Tyto funkce jsou:

- poměr signálu k šumu měřen v decibelech (anglicky *signal-to-noise ratio*, dále jen SNR),
- špičkový poměr signálu k šumu měřen v decibelech (anglicky *peak signal-to-noise ratio*, dále jen PSNR),
- míra bitové chybovosti (anglicky *bit error rate*, dále jen BER),
- střední kvadratická chyba (anglicky *mean squared error*, dále jen MSE),
- směrodatná odchylka (anglicky *root-mean-square deviation*, dále jen RMSD).

Rovnice těchto funkcí jsou v příloze B. Při testování se nejprve vyhodnotí maximální kapacita metody bez modifikace stego signálu s pomocí funkce BER. Poté bude testována robustnost následujícími metodami:

- přidání šumu,
- filtrace spektra,
- převzorkování,
- překvantování,
- překódování do ztrátového formátu.

Nepostřehnutelnost je sice subjektivní vlastnost, ale je možné částečně vyhodnotit její míru automaticky pomocí úrovně SNR a PSNR.

### Popis použitých datových sad

Datová sada *GTZAN* vznikla mezi lety 2000-2001 a obsahuje 1000 hudebních nahrávek po 30 sekundách. Žánr a zdroj jednotlivých nahrávek je různorodý. Datová sada obsahuje celkem 10 hudeních žánrů. Nahrávky obsahují jeden kanál a jsou vzorkovány na frekvenci 22050 Hz a kvantovány na 16 bitech [32].

Datová sada *NOIZEUS* vnikla v roce 2007 za účelem lepšího srovnání algoritmů pro zkvalitnění hlasových dat. Sada obsahuje 30 nahrávek s jedním PCM kanálem vzorkovaným na frekvenci 8 kHz a kvantovaným na 16 bitech. Obsahem nahrávek jsou všechny fonémy americké angličtiny. Součástí sady je také osm modifikovaných verzí nahrávek s přidaným ruchem různého druhu. Každá modifikace má ještě čtyři úrovně hlasitosti ruchu určeného mírou poměru signálu k šumu s hodnotami: 0 dB, 5 dB, 10 dB a 15 dB [15].

Datová sada *UrbanSound8K* vznikl v roce 2014 za účelem vytvoření volně dostupného korpusu nahrávek z městského prostředí. Nahrávky jsou ve formátu WAV, ale nemají jednotnou vzorkovací frekvenci ani počet bitů na vzorek. Celkem obsahuje sada 8732 krátkých nahrávek rozdělených do deseti kategorií městských zvuků [27].

K výše popsaným datovým sadám jsem vytvořil vlastní sadu nahrávek. Jedná se o 25 segmentů nahrávky řeči z videa *A Narrated Tour of the Moon*[15] od NASA. Délka nahrávek se pohybuje od ~4 sekund do ~20 sekund. Nahrávky obsahují jeden kanál PCM zvuku vzorkovaného na frekvenci 22050 Hz a kvantovaného na 16 bitech. Pro shrnutí je v tabulce 3.1 přehled parametrů jednotlivých datových sad.

Tabulka 3.1: Přehled parametrů datových sad

| Název | Vzorkovací frekvence | Bitová hloubka | Počet nahrávek |
|---|---|---|---|
| GTZAN | 22050 Hz | 16 b | 1000 |
| NOIZEUS | 8000 Hz | 16 b | 960 |
| UrbanSound8K | – | – | 8732 |
| A Narrated Tour of the Moon | 22050 Hz | 16 b | 25 |

## 3.3 Popis vlastní metody digitální zvukové steganografie

Princip mnou navrhované metody je podobný metodě popsané v [26], kde autoři kódují informaci do modulu (absolutní hodnoty) koeficientů Fourierovy transformace obrázku. Rozdíl je ale v tom, že navrhovaná metoda upravuje reálnou část komplexního koeficientu Fourierovy transformace, čímž upravuje modul i fázi. Druhá část této metody je kombinace s metodou přímého rozprostřeného spektra (viz podsekce 2.5). Aby bylo kódování robustnější, informační bity jsou nejprve rozprostřeny po celé délce krycího signálu a poté jsou přičteny ke koeficientům Fourierovy transformace krycího signálu. Princip kódování je možné vidět v algoritmu 1.

**Algoritmus 1:** Navrhovaná metoda kombinující modifikace Fourierovy transformace a metodu rozprostřeného spektra.

**Vstup:** krycí signál (cover), bity tajné informace (secret), klíč pseudonáhodné sekvence (key)
**Výstup:** stego signál (stego)

1: mixer ← rozprostři_bity(secret, délka(cover))
2: pn ← vygeneruj_pseudonáhodnou_sekvenci_čipů(key)
3: stego ← IDFT(DFT(cover) + mixer × pn)
4: **return** *stego*

Dekódování této metody probíhá podobně jako u metody rozprostřeného spektra. Po získání pseudonáhodné sekvence čipů se vypočítá Fourierova transformace stego signálu a spolu s pseudonáhodnou sekvencí se rozdělí na segmenty. Každý segment je korelován s příslušným segmentem pseudonáhodné sekvence. Tento popis je zapsán v algoritmu 2.

[15] https://svs.gsfc.nasa.gov/cgi-bin/details.cgi?aid=10929

**Algoritmus 2:** Dekódování navrhované metody.

**Vstup:** stego signál (stego), klíč pseudonáhodné sekvence (key), počet bitů tajné informace ($L$)

**Výstup:** bity tajné informace (secret)

```
1: pn ← vygeneruj_pseudonáhodnou_sekvenci_čipů(key)
2: pn ← rozděl_signál_na_segmenty(pn, L)
3: segmenty ← rozděl_signál_na_segmenty(DFT(stego), L)
4: for n = 1 do L do
5:     korelace ← sečti(segmenty[n] × pn[n])
6:     if korelace > 0 then
7:         secret[n] ← 1
8:     else
9:         secret[n] ← 0
10:    end if
11: end for
12: return secret
```

# Kapitola 4

# Realizace systému

V této kapitole bude popsána konkrétní struktura modulů a tříd tak, jak jsou v řešení implementovány. Bude popsáno, jak je dosaženo polymorfizmu pomocí společného rozhraní metod. A poté budou popsány oba vestavěné programy a implementace jednotlivých metod v knihovně.

## 4.1 Struktura modulů knihovny a programu

Řešení je rozděleno do několika modulů a balíčků, které plní specifickou činnost. V kořenovém balíčku se nachází následující balíčky:

- `cli` – obsahuje moduly týkající se vestavěného programu pro individuální spouštění,
- `methods` – obsahuje moduly jednotlivých steganografických metod a také balíček s jejich automatickými testy,
- `tests` – obsahuje automatické testy pro moduly v kořenovém balíčku.

V kořenovém balíčku se nacházejí moduly společné pro ostatní balíčky. Tyto moduly jsou:

- `audio_utils` – obsahuje pomocné funkce pro práci se signály; nejdůležitější funkcí je `split_to_n_segments`, která rozdělí pole na segmenty s požadovanou délkou,
- `stat_utils` – obsahuje implementované statistické funkce popsané v příloze B,
- `exceptions` – obsahuje vlastní výjimky použité v knihovně a programu,
- `tests_common` – obsahuje automatizované testy společné pro všechny metody.

### Implementace vestavěného programu

Všechny moduly pro hlavní vestavěný program se nachází v balíčku `cli`. Program je možné spustit příkazem `python -m audio_steganography`, čímž se automaticky spustí soubor `__main__.py` v kořenovém balíčku. Tento soubor jsem převzal a upravil z projektu `yt-dlp`[1]. Jeho jediným úkolem je upravit proměnné prostředí, aby bylo možné importovat a spustit funkci `main`, která se nachází v souboru `cli/__init__.py`.

[1] https://github.com/yt-dlp/yt-dlp

Po spuštění funkce `main` se vyhodnotí argumenty předané programu. Parser argumentů má několik společných argumentů, avšak všechny argumenty specifické pro jednotlivé metody jsou generovány automaticky voláním funkcí `get_encode_args` a `get_decode_args` pro každou steganografickou metodu.

### Argumenty pro spuštění

Jelikož jsou argumenty generovány ze společného rozhraní metod, je i rozhraní argumentů velmi předvídatelné. Kostra volání je následující:

```
python3 -m audio_steganography <metoda> <encode/decode> ...
```

Prvním argumentem je název požadované metody a poté požadovaný režim – `encode` pro kódování a `decode` pro dekódování. Poté je nutné zadat všechny povinné přepínače a případně nastavit vhodně i volitelné přepínače. Povinný společný přepínač je pouze `-s/--source` – při kódování se jedná o krycí soubor do kterého budou zakódována data a při dekódování se jedná o stego soubor ze kterého budou data dekódována. Při kódování je ještě potřeba použít jeden z následujících přepínačů:

- `-t/--text` – umožňuje zadání kódovaného textu jako argument programu,
- `-f/--file` – kódovaná data budou načtena jako surové bajty ze souboru.

Většina metod má parametry, které ovlivňují jejich kvalitu a rychlost kódování, robustnost, kapacitu či nepostřehnutelnost. Nejlepší způsob jak zjistit o jaké parametry se u metod jedná je zadáním přepínače `-h` nebo `--help` po vyplnění základní kostry volání.

Pro výstup je možné použít argument `-o/--output`, čímž je možno určit název výstupního souboru po kódování/dekódování. Pokud je specifikován název `-` v režimu dekódování, budou dekódovaná data vytištěna na standardní výstup. Výstup je uvozen uvnitř uvozovek s písmenem `b` na začátku. Jedná se pouze o výstupní formát bajtového pole v jazyce Python. Pokud jsou ve výstupu znaky mimo znakovou sadu *ASCII*, pak bude hodnota bajtu obsahovat prefix `\x` a některé netisknutelné znaky budou obsahovat jejich příslušnou „escape sekvenci“. Příklad výstupu slov „Dobrý“ a „den“, každý na novém řádku je:

```
b'Dobr\xc3\xbd\nden'
```

Výstup obsahuje také asociované pole hodnot ve formátu *JSON* s dodatečnými parametry potřebnými pro dekódování u některých metod. Poslední společné volitelné argumenty jsou:

- `-y/--overwrite` – potlačí chybovou hlášku, pokud výstupní soubor existuje a bude přepsán bez dotázání,
- `--stats` – přidá do dodatečného výstupu programu výstup statistických funkcí.

Po vyhodnocení zadaných parametrů se spustí fasáda s vybranou metodou a parametry.

### Implementace fasády

Třída `MethodFacade` se nachází v modulu `cli/method_facade` a jejím účelem je abstrakce některých činností při použití vestavěného programu. Krycí signál je načten ve funkci `main` z *WAV* souboru do *NumPy* `NDArray` pole pomocí funkce `scipy.io.wavfile.read`. V případě kódování je dalším krokem přípravy načtení ukrývaných dat do pole typu `uint8`, které bude obsahovat informační bity. Tento převod je proveden načtením kódovaných dat jako pole bajtů a poté převedením na jednotlivé bity. Ze zadaných argumentů se vytvoří fasáda pro kódování či dekódování na základě zvoleného módu.

Po přípravě dat je možné použít funkce `encode` a `decode`, které předají jejich parametry stejnojmenným funkcím instance vybrané metody.

## 4.2 Implementace metod

Každá implementovaná metoda dědí ze společné abstraktní třídy `MethodBase` z modulu `methods/method_base`. Ve společném rozhraní jsou pro vestavěný program funkce `get_encode_args` a `get_decode_args`, které vrací pole argumentů pro kódování a dekódování, které jsou specifické pro danou metodu.

Dále jsou zděděné metody nuceny implementovat funkce `encode` a `decode`. Obě funkce vrací proměnné stejných typů – prvním je NDArray pole obsahující buď vzorky zakódovaného signálu nebo dekódované bity a druhou proměnnou je slovník s dodatečným výstupem. Bohužel není možné dosáhnout polymorfizmu perfektně, protože různé steganografické metody požadují různé parametry pro kódování a dekódování. Naštěstí Jazyk Python umožňuje při volání funkcí expandovat do argumentů slovník (asociované pole hodnot) s přiřazenými parametry. Pokud je v hlavičce funkce použito `**kwargs`, pak Python ignoruje přebytečné argumenty. Při volání funkcí `encode` a `decode` je díky tomu možné přednastavit argumenty pro metody předem do slovníku a poté pouze vybrat použitou metodu. Tato funkcionalita je využita, jak ve třídě `MethodFacade`, tak i při volání fasády ve funkci `main`. Zjednodušený příklad kódu z funkce `main` je ve výpisu 4.1.

```
options = {
    MethodEnum.lsb: {
        (a:='depth'): get_attr(args, a),
        (a:='only_needed'): get_attr(args, a),
        'l': get_attr(args, 'len'),
    },
    ...
}
steganography = MethodFacade(...)
method = MethodEnum[args.method]
output, additional_output = steganography.encode(**options.get(method, {}))
```

Výpis 4.1: Zobecněné použití fasády a steganografických metod.

Poslední částí společného rozhraní jsou proměnné `_source_data` a `_secret_data`. Proměnná `_source_data` obsahuje vzorky krycího signálu a v proměnné `_secret_data` jsou uloženy bity ukrývané informace.

### Metoda nahrazení nejméně významného bitu

Tato podsekce popisuje implementaci metody popsané v podsekci 2.5. Kódování pomocí této metody je velmi jednoduché, pokud jsou vzorky krycího signálu celočíselného datového typu. Pokud jsou vzorky datového typu *float*, pak je potřeba nejprve převést vzorky do celočíselného datového typu se stejnou bitovou velikostí. V knihovně NumPy je možné použít funkci `tobytes`, která převede pole se vzorky na pole sekvence bajtů tak, jak jsou uloženy v paměti. Bajty je možné znovu načíst do NDArray pole funkcí `frombuffer`, které specifikuje požadovaný celočíselný typ. Úryvek kódu 4.2 převede pole vzorků s plovoucí desetinnou čárkou na pole celočíselných typů. Slovník `dtype_conv` slouží jako tabulka pro převod plovoucích typů na celočíselné.

```
source = self._source_data
if source.dtype in dtype_conv:
    source = source.astype(source.dtype.newbyteorder('<')).tobytes()
    source = np.frombuffer(source,
        dtype=dtype_conv[self._source_data.dtype])
```

Výpis 4.2: Převedení datového typu s plovoucí desetinnou čárkou na celočíselný datový typ.

Implementace podporuje kódování do více bitových úrovní a zakódování pouze potřebných bitů. Při kódování do více úrovní se musí bity v poli `_secret_data` rozdělit na podpole o velikosti zvolené bitové hloubky. Podpole jsou poté převedeny na bajty. Ve vzorcích krycího signálu jsou vynulovány poslední bity až po zvolenou bitovou úroveň logickou funkcí `and` a zakódována ukrývaná data logickou funkcí `or`.

Při kódování pouze potřebných bitů bude upraveno pouze tolik vzorků, kolik je potřeba. Pokud je tato volba vypnutá, vyplní se zbytek vzorků náhodným šumem, aby ukrytá data splynula. Dekódování probíhá opačně, bajty jsou načteny funkcí `and` a převedeny na bity.

### Metoda skrývání pomocí ozvěny

Tato podsekce popisuje implementaci metody a modifikací popsaných v podsekci 2.5. Všechny varianty této metody používají stejné parametry a kontroly, proto třídy těchto metod dědí z abstraktní bázové třídy `EchoBase` v modulu `cli/echo_base`. Při kódování se po zavolání funkce `encode` přesměruje volání na funkci `encode_wrapper`, do které se k argumentům přidá konkrétní třída vybraná ke kódování. Ve funkci `encode_wrapper` se provedou kontroly argumentů a také výběr způsobu volitelného hledání optimálních zpoždění ozvěn.

Kódování pomocí této metody je těžko předvídatelné a výsledky silně závisí na zvolených hodnotách zpoždění. Proto jsem pro hledání optimálních hodnot implementoval dva způsoby automatického hledání. Prvním způsobem je hledání hrubou silou, které provede 300 iterací v okolí `d0` a `d1`. Tento způsob hledání většinou najde optimální hodnoty. Avšak doba běhu může trvat velmi dlouho, na základě délky krycího signálu. Efektivnějším způsobem je druhá varianta hledání a to pomocí optimalizační metody *Basin-hopping*[2] z knihovny *SciPy*. Jedná se o optimalizační funkci pro nalezení globálního minima dané funkce. V případě steganografických metod se minimalizovanou funkcí myslí funkce míry bitové chybovosti (dále jen BER, viz kapitola B). Použitím této optimalizační funkce je hle-

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.basinhopping.html

dání správných hodnot mnohem rychlejší než při hledání hrubou silou, ale není zaručeno, že výsledné hodnoty dosáhnou zakódování bez chyb.

Samotné kódování každé metody je implementováno v její příslušné třídě ve funkci `_encode`. Pokud není vybrána metoda hledání hodnot zpoždění, budou použity hodnoty zadané uživatelem bez ohledu na hodnotu BER. To umožňuje případně implementovat jiné způsoby hledání optimálních hodnot. Implementace obsahuje velký rozdíl oproti teoretickému popisu z důvodu optimalizace kódování, a to nahrazením vytvoření ozvěny konvolucí za zkopírování pole a doplnění příslušného počtu nul. Jednotlivé ozvěny jsou pak přiřazeny jako výřezy zkopírovaného pole. Ukázka kódu, který se v podobné formě nachází ve všech variantách ozvěnové metody je vidět ve výpisu 4.3.

```
# doplneni nul ke krycimu signalu
source_pad = np.pad(self._source_data, (d1 + 5, 0)) * alpha/2

# leva strana: d0 nul
echo_0_neg = source_pad[d1+5-d0:] * -1
# leva strana: d0 + 5 nul
echo_0_pos = source_pad[d1-d0:] * decay_rate

# leva strana: d1 nul
echo_1_neg = source_pad[5:] * -1
# leva strana: d1 + 5 nul
echo_1_pos = source_pad * decay_rate
```

Výpis 4.3: Vytvoření pouze jedné kopie krycího signálu a přiřazení jednotlivých ozvěn jako výřezy.

Pro výběr vzorků požadovaných segmentů ozvěn je použita funkce `spread_bits`, která rozprostírá bity přes zadanou délku. Rozprostřené bity ukrývané informace jsou použity jako maska, kterou se vynásobí vytvořené ozvěny. Funkci `spread_bits` lze vidět ve výpisu 4.4.

```
def spread_bits(secret_data: np.ndarray, signal_length: int) -> np.ndarray:
    mixer, rest = split_to_n_segments(np.ones(signal_length),
        len(secret_data))
    mixer = mixer * secret_data[:, None]
    return np.append(mixer, rest)
```

Výpis 4.4: Funkce pro rozprostření bitů na zadanou délku.

Dekódování každé varianty této metody se u každé mírně liší a je implementováno ve funkci `decode` příslušné třídy. Metody *single echo* a *bipolar backward-forward echo* používají pouze cepstrum stego signálu. Metody *bipolar echo* a *backward-forward echo* používají autokorelaci cepstra.

### Metoda rozprostřeného spektra

Tato podsekce popisuje implementaci metody popsané v podsekci 2.5. Pro vytvoření sekvence čipů je použita funkce `spread_bits`, která rozprostírá bity přes zadanou délku (viz výpis 4.4). Čipy jsou poté vynásobeny pseudonáhodnou sekvencí čipů vygenerované ze

zadaného klíče. Klíč je převeden na 32-bitové číslo hešovací funkcí *SHA256* a je použit jako počátek pseudonáhodného generátoru (anglicky *seed*). Implementace používá pro generování pseudonáhodné sekvence generátor *MT19937*[3]. Tento generátor není dostatečně bezpečný pro kryptografické účely, ale je použit v implementaci protože podporuje zadání „seedu“. Pro bezpečnější kódování by měla být v budoucnu podporována možnost zadání pseudonáhodné sekvence externě funkci `encode`. Sekvence čipů je poté jednoduše modulována s krycím signálem a rozprostřenými bity v proměnné `mixer` (viz výpis 4.5).

```
encoded = source + mixer * alpha * pn_sequence
```

Výpis 4.5: Modulace krycího signálu, rozprostřených bitů a pseudonáhodné sekvence čipů.

Pro dekódování se znovu vygeneruje pseudonáhodná sekvence čipů ze zadaného klíče. Sekvence čipů i stego signál se rozdělí na segmenty se zadanou délkou. Poté je provedena korelace mezi segmenty stego signálu a sekvence čipů. Korelace je implementována pouze jako součet vynásobených vzorků (viz výpis 4.6).

```
decoded = np.zeros(l, dtype=np.uint8)
for i in range(l):
    correlation = np.sum(source_segments[i] * pn_sequence_segments[i])

    if correlation > 0:
        decoded[i] = 1
    else:
        decoded[i] = 0
```

Výpis 4.6: Korelace segmentů a rozhodovací pravidlo.

### Metoda rozprostřeného spektra ve Fourierově transformaci (vlastní metoda)

Algoritmický popis této metody se nachází v podsekci 3.3. Kódování této metody je téměř identické jako standardní metoda rozprostřeného spektra popsaná výše. Rozdíl je v použití Fourierovy transformace pro modulaci sekvence čipů. Po modulaci je signál převeden zpět do časové domény inverzní Fourierovou transformací (viz výpis 4.7).

```
fft = np.fft.fft(source)
encoded = fft + mixer * np.abs(fft).max() * alpha * pn_sequence
encoded = np.fft.ifft(encoded).real
```

Výpis 4.7: Modulace Fourierovy transformace krycího signálu, rozprostřených bitů a pseudonáhodné sekvence čipů.

### Metoda fázového kódování

Tato podsekce popisuje implementaci metody popsané v podsekci 2.5. Kroky implementace se téměř shodují s teoretickým popisem této metody. Rozdíl je ve volbě fází v prvním segmentu. Autoři [4] nahradili fáze nultého segmentu novými fázemi, které reprezentují

[3] https://numpy.org/doc/stable/reference/random/legacy.html

ukrývanou informaci. Při implementaci jsem zjistil, že tato varianta velmi negativně ovlivňuje fáze na začátku a tím i výsledný stego signál. Rozhodl jsem se proto implementovat modifikaci této metody, která naopak nahrazuje fáze nejvyšších frekvencí v segmentu. Algoritmický popis této metody jsem našel v odpovědi Dana Oaka na fóru *StackExchange* [20].

Pro zakódování se zvolí velikost segmentů několikrát větší než je délka ukrývaných bitů, aby byly nové fáze uloženy ve vysokých frekvencích. Délka segmentů je vypočítána podle vzorce 4.1, kde $X$ je počet bitů ukrývané informace, $L$ je délka segmentů a $\lceil\ \rceil$ je funkce pro zaokrouhlení nahoru.

$$L = 2 \times 2^{\lceil \log_2 (2 \times X) \rceil} \tag{4.1}$$

Nové fáze se zapíší do horních frekvencí, které se nachází v polovině pole s fázemi segmentu. Délka segmentu musí být mnohem větší než je ukrývaná informace, protože jsou fáze segmentu symetrické. Pro zachování této symetrie jsou se nové fáze zapíší i do druhé půlky pole fází, ale opačně. Matematický zápis nahrazení fází se nachází v rovnici 4.2, kde $i \in \{1, \ldots, X\}$ a $\phi_d$ je pole nových fází.

$$\begin{aligned} \phi_0[L/2 - X + i] &= \phi_d[i] \\ \phi_0[L/2 + 1 + i] &= -\phi_d[X + 1 - i] \end{aligned} \tag{4.2}$$

Kód pro zápis nových fází ekvivalentní s rovnicemi 4.2 se nachází ve výpisu 4.8.

```
angles[0, segment_len//2 - secret_len : segment_len//2] = secret_angles
angles[0, segment_len//2 + 1 : segment_len//2 + 1 + secret_len] =
    -np.flip(secret_angles)
```

Výpis 4.8: Zápis nových fází do pole s fázemi segmentu.

### Metoda ukrývání v úsecích ticha

Tato podsekce popisuje implementaci metody popsané v podsekci 2.5. K nalezení úseků ticha je použita funkce `consecutive_values`, která v zadaném poli najde počáteční indexy a délky úseků opakujících se hodnot. Tato funkce byla převzata a upravena z odpovědi na fóru *StackOverflow* [23]. Knihovna NumPy umožňuje filtrovat zadané pole určitými podmínkami. Výsledkem je pole booleovských hodnot `true` a `false` jako výsledek zadané podmínky pro každý prvek vstupního pole. Pro nalezení úseků ticha se použije jako podmínka filtru zda je absolutní hodnota vzorku menší než 15% maximálního rozsahu signálu. V booleovských hodnotách jsou poté nalezeny souvislé sekvence hodnot `true`. Kód funkce `consecutive_values` a její použití pro nalezení úseků ticha je vidět ve výpisu 4.9.

```
def consecutive_values(input: Union[List, np.ndarray]):
    input = np.asanyarray(input)
    if input.size == 0: return np.array([]), np.array([])
    d = np.ones(input.shape, dtype=bool); d[1:] = np.diff(input)
    return np.nonzero(d)[0], np.diff(np.append(np.nonzero(d)[0],
        input.size))
starts, lens = consecutive_values(np.abs(source) <= 0.15 *
    np.abs(source).max())
```

Výpis 4.9: Nalezení indexů a délek úseků ticha.

Implementační rozdíl v této metodě je výchozí minimální délka úseku ticha, pro kterou jsem zvolil hodnotu 400 vzorků. Tuto hodnotu jsem zvolil experimentálním průzkumem. Tato hodnota však není pevně daná v kódu a je možné ji změnit při kódování a dekódování. Dekódování probíhá ekvivalentně stejně jako v teoretickém popisu této metody. Po nalezení délek úseků ticha jsou získány decimální hodnoty půlbajtů. Ty však nestačí pouze převést na bity, protože je půlka z nich nevyužitá. Proto je ještě potřeba extrahovat z každého bajtu pouze první 4 bity a 4 bity vynechat. Zápis extrakce 4 bitů z každého bajtu využívající vlastnosti NumPy pole NDArray lze vidět ve výpisu 4.10.

```
decoded = np.unpackbits(
    decoded, axis=-1,
    bitorder='little').reshape(-1, 4)[::2, :].reshape(-1)[:1]
```

Výpis 4.10: Extrakce prvních 4 bitů z každého bajtu.

### Metoda vkládání tónu

Tato podsekce popisuje implementaci metody popsané v podsekci 2.5. Ke kódování i dekódování je potřeba vypočítat výkon segmentů signálu. Publikace popisující tuto metodu nespecifikují rovnici pro výpočet výkonu segmentu a proto jsem zvolil jeden z možných výpočtů, který lze vidět v rovnici 4.3, kde $P$ je výkon segmentu a $N$ je počet vzorků segmentu.

$$P = \frac{1}{N} \sum_{n=0}^{n=N-1} |x[n]|^2 \tag{4.3}$$

Poté jsou vygenerovány tóny na zvolených frekvencích funkcí $\sin$. Z obou tónů jsou také vypočítány výkony, aby bylo možné nastavit jejich amplitudy tak, aby měly požadovaný poměr výkonu. Výpočet amplitud tónů k docílení požadovaných výkonů je vidět v rovnici 4.4, kde $A$ je požadovaná amplituda tónu, $M$ je požadovaný procentuální poměr výkonu, $P_s$ je výkon segmentu a $P_t$ je výkon tónu.

$$A = \sqrt{M \times \frac{P_s}{P_t}} \tag{4.4}$$

V implementaci jsou amplitudy předpočítány pro všechny kombinace bitů. Výpočet bude proveden efektivněji, protože knihovna NumPy použije na pozadí optimalizované vektorové instrukce. Výpočet tak bude mnohem rychlejší než při použití cyklu v jazyce Python. Předpočítané amplitudy jsou poté vybrány na základě kódovaného bitu a násobí se s příslušným tónem. Ukázka kódu pro výpočet amplitud a modulaci tónů se segmenty je vidět ve výpisu 4.11.

K dekódování jsem použil místo poměrů výkonů poměry korelačního koeficientu segmentu a jednotlivých tónů. Kód pro výpočet korelace a určení hodnoty bitu lze vidět ve výpisu 4.12.

```
f0_amplitudes = np.empty((2, len(segments)))
f0_amplitudes[0] = np.sqrt(0.0025 * segment_powers / tone_f0_power)
f0_amplitudes[1] = np.sqrt(0.000025 * segment_powers / tone_f0_power)

f1_amplitudes = np.empty((2, len(segments)))
f1_amplitudes[0] = np.sqrt(0.000025 * segment_powers / tone_f1_power)
f1_amplitudes[1] = np.sqrt(0.0025 * segment_powers / tone_f1_power)

for i, bit in enumerate(self._secret_data):
    segments[i] += (tone_f0 * f0_amplitudes[bit][i] +
                    tone_f1 * f1_amplitudes[bit][i])
```

Výpis 4.11: Výpočet amplitud a přidání tónů k segmentům.

```
for i in range(min(l, len(segments))):
    decoded[i] = (
        np.sum(segments[i] * tone_f1) / segment_len >
        np.sum(segments[i] * tone_f0) / segment_len)
```

Výpis 4.12: Výpočet korelace mezi segmenty a tóny.

## 4.3 Implementace programu pro ověření vlastností metod

Ověřování vlastností je rozděleno do dvou částí. První částí je program pro vyhodnocení vlastností metod a výpočet statistických funkcí popsaných v sekci 3.2. Druhou částí je program pro zpracování dat vygenerovaných předchozím programem.

### Argumenty pro spuštění programu pro vygenerování dat

První program pro generování se nachází v balíčku `cli.evaluation`. Kostra volání programu je následující:

```
python3 -m audio_steganography.cli.evaluation -d <cesta k datovým sadám> \
    -o <cesta výstupní složky> <testované metody>
```

Přepínač `-d/--datasets` specifikuje kořenovou složku obsahující použité datové sady. Očekávaná struktura je datových sad je následující:

```
<kořen datových sad>
|   <datová sada>
|   |   <kategorie>
|   |   |   <soubory>
|   |   |   ...
|   |   ...
|   ...
```

Obsahem výstupní složky, specifikované přepínačem `-o/--output`, budou soubory v textovém formátu s jednotlivými sloupci oddělenými středníkem. Tento formát měl původně

hodnoty oddělené čárkou (anglicky *comma-separated values*, CSV), ale protože se ve výstupu nachází čárky, tak jsem zvolil středník pro oddělení hodnot. Struktura výstupní složky je totožná s kořenovou složkou pro datové sady. Názvy souborů pouze obsahují navíc příponu `.csv`.

Pro výpis průběžného stavu vyhodnocování je možné použít přepínač `-l/--log`, který bude na standardní chybový výstup vypisovat informace o právě vyhodnocovaném souboru, metodě, parametrů a modifikací včetně dodatečných informací o prováděném kódování a dekódování.

Pro zrychlení vyhodnocování spouští program vyhodnocení souborů paralelně. Počet paralelních procesů je možné ovládat přepínačem `-p/--processes`. Výchozí hodnotou je maximální počet použitelných jader procesoru.

Přepínač `-e/--extended` přidá k vyhodnocovaným metodám dodatečné parametry pro hlubší testování, avšak vyhodnocování bude trvat podstatně déle.

Při implementaci a testování se objevily u testování překódování do formátu MP3 problémy, při kterých docházelo k náhlému zastavení programu. Nepovedlo se mi určit zdroj tohoto problému, protože se objevoval pouze u některých souborů z datových sad. Proto jsem přidal přepínač `--no-mp3` pro vypnutí překódování do formátů MP3.

Na závěr jsou po přepínačích zadány metody, které budou vyhodnocovány. Názvy metod je možné vypsat v nápovědě přepínačem `-h/--help`. Pro vyhodnocení všech metod je možné zadat argument `ALL` nebo nezadat žádnou metodu.

### Způsob vyhodnocení

Program nejprve vyhodnotí zadané argumenty, ze kterých zjistí primárně kořenovou složku obsahující datové sady. Při načítání cest souborů jsou ignorovány všechny soubory a složky začínající tečkou. Všechny soubory které, budou použity pro vyhodnocení se přidají spolu se zvolenými metodami jako argumenty pro funkci `evaluate_method`, která provádí kódování, modifikace a dekódování. Třída `Pool` z knihovny `multiprocessing` zajišťuje vytváření a spouštění paralelních procesů s funkcí `evaluate_method`.

Funkce `evaluate_method` obsahuje několik vrstev vnořených cyklů, které přidávají vyhodnocované aspekty. První cyklus testuje různé délky zpráv. Pokud je zpráva příliš dlouhá na konkrétní metodu a její nastavení, pak jsou ponechány hodnoty prázdné až na dobu kódování a dekódování, které jsou nastaveny na nekonečno. V dalším cyklu se každá zpráva kóduje do souboru zvolenými metodami. Každá metoda má také různé parametry, které ovlivňují míru nepostřehnutelnosti, robustnosti a kapacity. Proto musí být tyto parametry otestovány navzájem s ostatními pro nalezení optimálních hodnot. Po zakódování zprávy jsou aplikovány modifikace a vyhodnocení vlastností statistickými funkcemi popsanými v sekci 3.2. Výsledky pro vyhodnocený soubor jsou z cyklů postupně sesbírány do instance třídy `DataFrame` z knihovny *pandas* a zapsány ve formátu *CSV* do výstupní složky.

### Argumenty pro spuštění programu pro zpracování výsledků

Program pro zpracování výsledků se nachází v balíčku `cli.evaluation.process`. Kostra volání programu je následující:

```
python3 -m audio_steganography.cli.evaluation.process \
    -d <cesta k výsledkům> -o <cesta výstupní složky>
```

Přepínač `-d/--data` specifikuje kořenovou složku obsahující výsledky vyhodnocení metod na datových sadách. Očekávaná struktura je stejná jako u programu pro generování výsledků (viz podsekce 4.3).

Obsahem výstupní složky, specifikované přepínačem `-o/--output`, jsou soubory ve formátu CSV s hodnotami oddělenými středníkem a při použití přepínače `--graphs` také grafy z tabulek ve formátu PDF.

### Zpracování výsledků

Program načte na základě daných argumentů tabulky s výsledky ze všech souborů do jedné instance třídy `DataFrame` z knihovny *pandas*. Tato tabulka je předána funkci `process_data`, která data zpracuje, vyfiltruje a vytvoří tabulky a grafy s výslednými hodnotami zjišťovaných parametrů steganografických metod. Konkrétní parametry a hodnoty budou popsány v následující kapitole.

# Kapitola 5

# Zhodnocení vlastností realizace

V této kapitole budou popsány výsledky vyhodnocení implementovaných metod na zvolených datových sadách popsaných v části 3.2. Vyhodnocování velkého množství kombinací parametrů metod způsobilo výrazné zpomalení programu a velikost datových sad tomu také nepřispěla. Proto byl z datové sady *UrbanSound8K* použit pouze oddíl s názvem *fold1*, který obsahuje 873 nahrávek. Avšak pouze soubor `88466-7-0-0.wav` byl vynechán, protože jsou vzorky tohoto souboru kódovány pomocí adaptivní diferenciální pulzně-kódové modulace, která není podporována funkcí `scipy.io.wavfile.read`, která se stará o načítání audio souborů.

## 5.1 Testované parametry

Testované délky zpráv jsou 32 bitů, 128 bitů, 512 bitů a v rozšířeném režimu i 256 bitů. Tyto délky byly zvoleny pro otestování kapacity a kvality metod. Z modifikací ze sekce 3.2 byly použity všechny vybrané s několika variantami. Varianty modifikací jsou následující:

- přidání šumu má tři varianty poměru signálu k šumu: 10 dB, 20 dB a v rozšířeném režimu i 15 dB,
- filtrace spektra je rozdělena na filtraci horních a spodních frekvencí od poloviny maximální frekvence signálu,
- převzorkování je testováno pouze snížením počtu vzorků na polovinu a zpět,
- překvantování je testováno obdobně snížením počtu bitů na vzorek na polovinu a zpět,
- pro překódování do ztrátového formátu byl zvolen formát MP3 s přenosovou rychlostí 96 kbit/s.

Robustnost a nepostřehnutelnost je měřena pro každou kombinaci metod a modifikací.

## 5.2 Výsledky metod

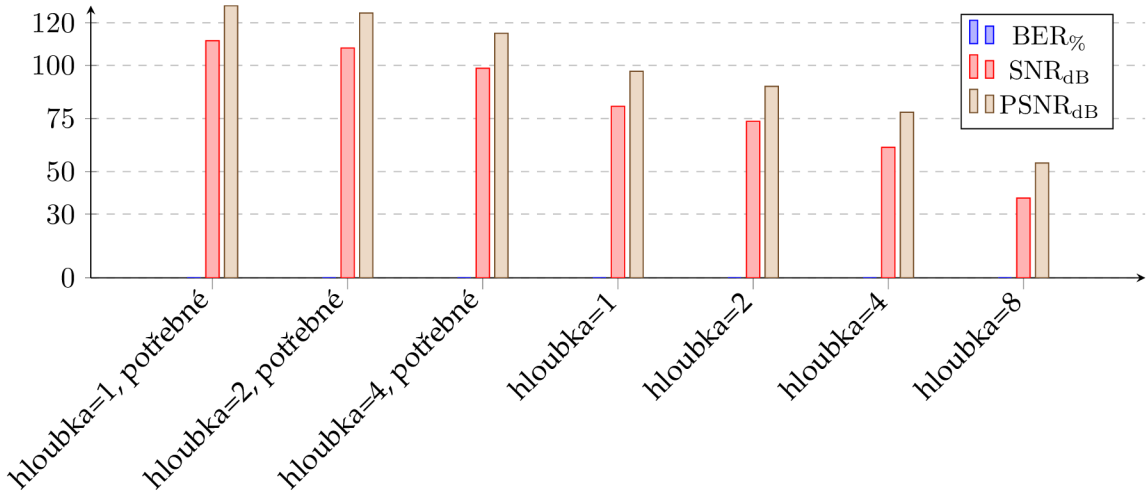
V prvním grafu u jednotlivých metod jsou průměrné hodnoty bitové chybovosti (BER, viz příloha B), poměru signálu k šumu (SNR, viz příloha B) a špičkového poměr signálu k šumu (PSNR, viz příloha B) pro jednotlivé testované parametry. Z výsledků prvního grafu každé

metody byly vybrány nejlepší parametry metod, které byly použity pro vytvoření grafů s výsledky průměrných hodnot BER, SNR a PSNR.

### Metoda nahrazení nejméně významného bitu

Tato metoda dosáhla bez aplikovaných modifikací nejlepších hodnot BER, SNR a PSNR. V obrázku 5.1 je také vidět jasná korelace mezi kódování pouze potřebných bitů a nepostřehnutelností tajné zprávy. Avšak tato statistika může být mírně zavádějící, protože nepoužití tohoto parametru přidává náhodný šum přes celou délku nahrávky, který se při kódování delší zprávy může lidskému posluchači jevit méně podezřele, pokud je šum slyšet všude na rozdíl od šumu pouze v části se zprávou.

Pokud nebylo se stego signálem manipulováno, pak má tato metoda perfektní rekonstrukci zakódované zprávy. Kapacitu této metody je možné vypočítat vynásobením počtu vzorků krycího signálu a použité bitové hloubky.

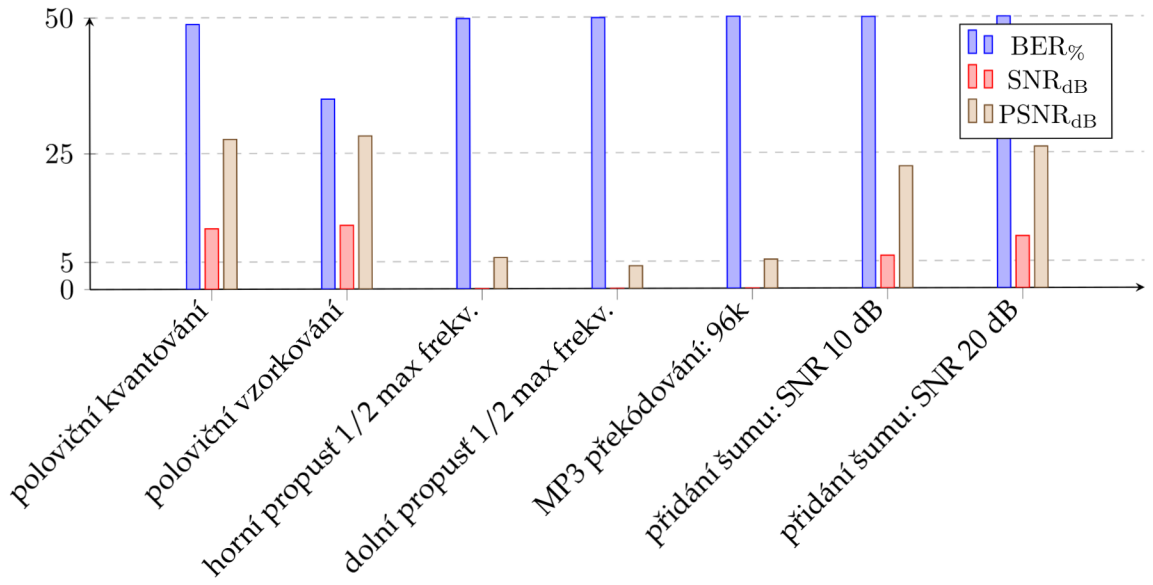



Obrázek 5.1: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ bez modifikací pro metodu nahrazení nejméně významného bitu.

Jak je vidět z grafu 5.2, má tato metoda velmi nízkou robustnost, ale překvapivě není z testovaných metod nejhorší. Jedinou modifikací, přes kterou byla tato metoda schopna dekódovat informace, bylo převzorkování na polovinu vzorků, čemuž také prakticky odpovídá hodnota BER.

### Metoda skrývání pomocí ozvěny

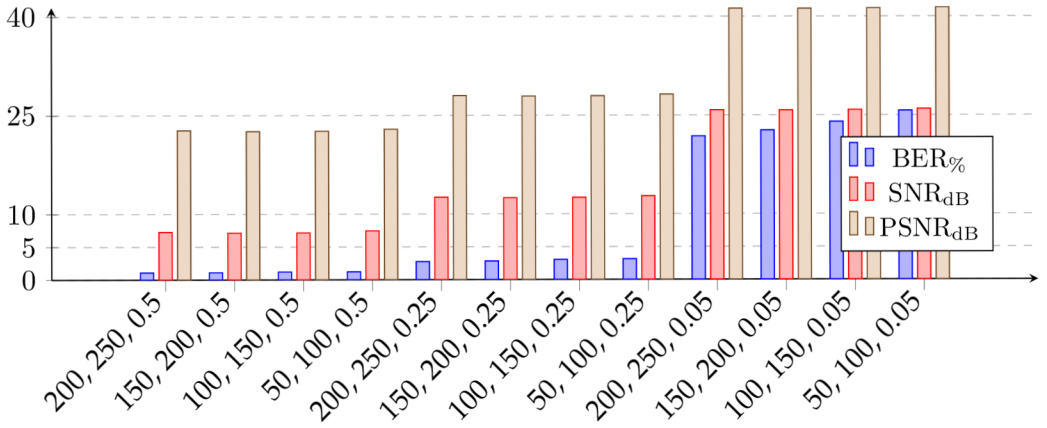
Implementovaná ozvěnová metoda má čtyři varianty, které se snaží vylepšit každou předchozí. Tyto metody dosahují nejvyšší robustnosti ze všech testovaných metod. Avšak jejich nepostřehnutelnost není příliš vysoká. Pro zmenšení grafů budou v prvním grafu bez modifikací vynechány názvy parametrů, které jsou v pořadí: *d0*, *d1*, *alpha*.




Obrázek 5.2: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ s modifikacemi metody nahrazení nejméně významného bitu.

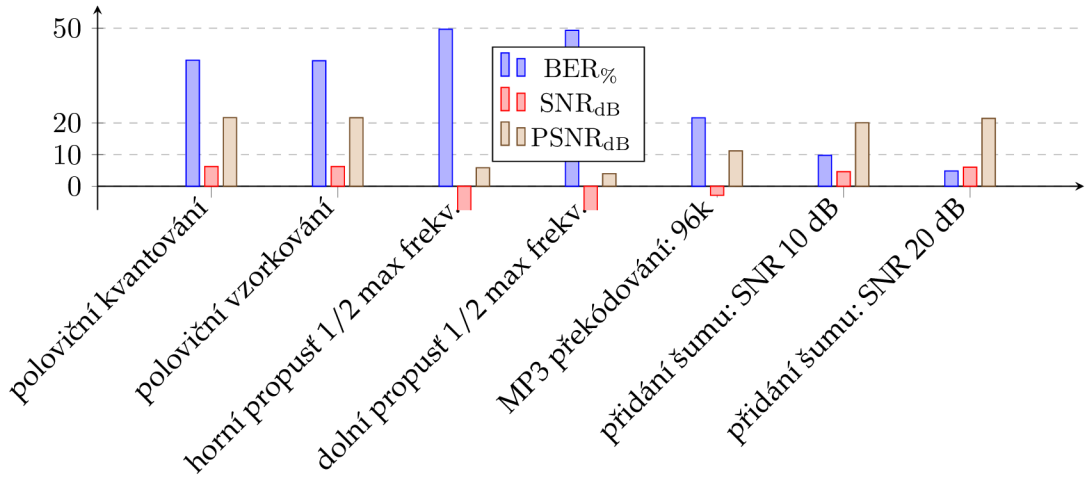
### Varianta s jednou ozvěnou na bit

Tato varianta vznikla jako první z ozvěnových metod, ale přesto dosahuje poměrně dobré robustnosti. Na obrázku 5.3 je vidět, že hodnoty této metody nejsou příliš ovlivněny zvolenými indexy ozvěn. Naopak jsou závislé na zvolené amplitudě přidaných ozvěn.




Obrázek 5.3: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ bez modifikací pro metodu ukrývání pomocí ozvěny s jednou ozvěnou na bit s parametry *d0*, *d1*, *alpha* na vodorovné ose.

Na obrázku 5.4 je vidět, že hodnoty BER po modifikacích přidání šumu a překódování do formátu MP3 jsou v přijatelných mezích (do 20%).

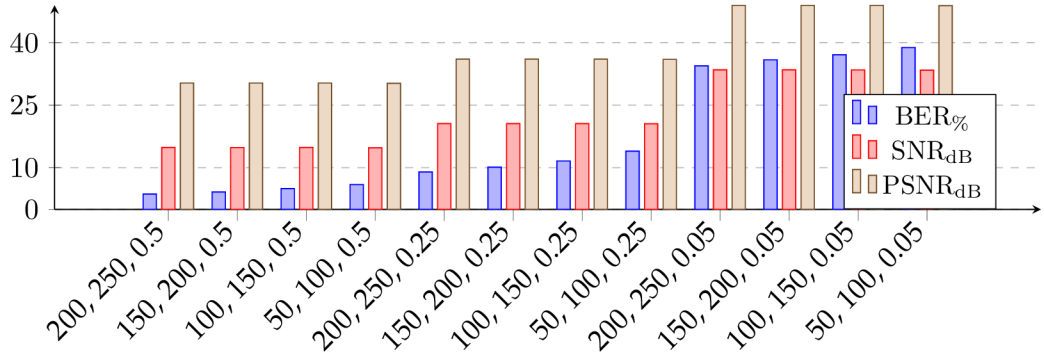



Obrázek 5.4: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ s modifikacemi metody ukrývání pomocí ozvěny s jednou ozvěnou na bit.

Ručním testováním a vhodným výběrem parametrů byla tato metoda schopna dosáhnout kapacity ~15 bitů za sekundu při vzorkovací frekvenci 44,1 kHz a hodnoty 27,6 dB SNR a 37 dB PSNR. Parametry byly zvoleny tak, aby byl výsledný stego signál lidským uchem nerozeznatelný od krycího signálu, použitím hodnoty 0,1 pro parametr `alpha`. Jako hraniční hodnota BER bylo zvoleno 15% stejně jako u autorů [14], kteří dosáhli průměrné kapacity 16 bitů za sekundu. Také došli k závěru, že parametr dozvuku (anglicky *decay rate*) byl nejdůležitějším parametrem pro dekódování. V automatickém testování byly původně testovány i různé hodnoty dozvuku a rozdíl oproti zvolené výchozí hodnotě 0,85 byl méně než 1% BER.

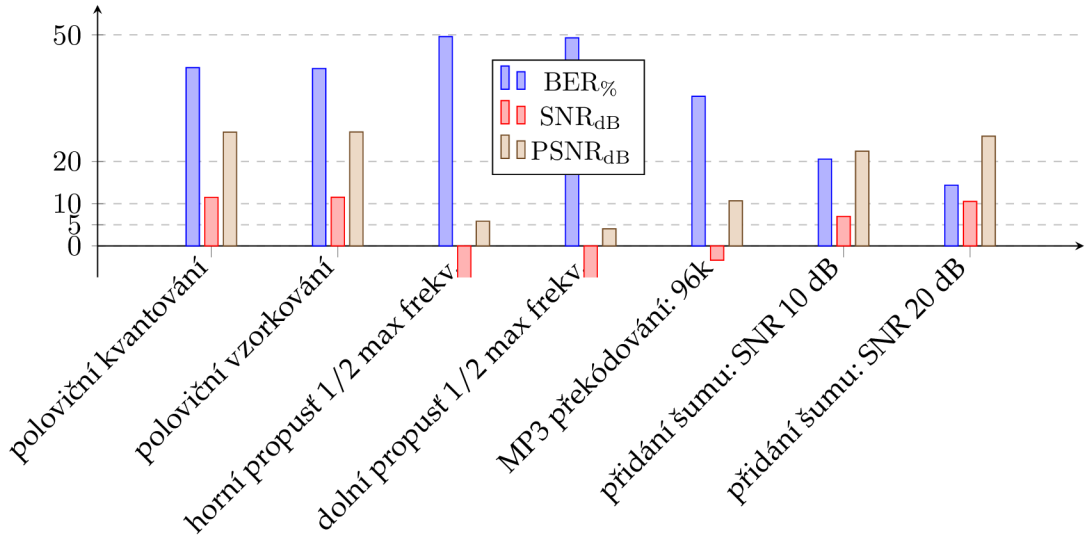
### Varianta ozvěn s kladnou a zápornou (bipolární) amplitudou

U této varianty začínají být vidět rozdíly při volbě jiných indexů ozvěn. Obrázek 5.5 naznačuje, že použití nižších hodnot indexů vede k nižším hodnotám BER a vyšším hodnotám SNR a PSNR. V porovnání s první variantou má tato metoda sice vyšší BER, ale má také podstatně vyšší SNR a PSNR a tím pádem by měla být méně detekovatelná.




Obrázek 5.5: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ bez modifikací pro metodu ukrývání pomocí ozvěny se dvěma bipolárními ozvěnami na bit s parametry *d0*, *d1* a *alpha* na vodorovné ose.

Tato varianta má u některých modifikací lepší hodnoty SNR a PSNR, ale nižší hodnoty BER než první varianta, jak lze vidět na obrázku 5.6.

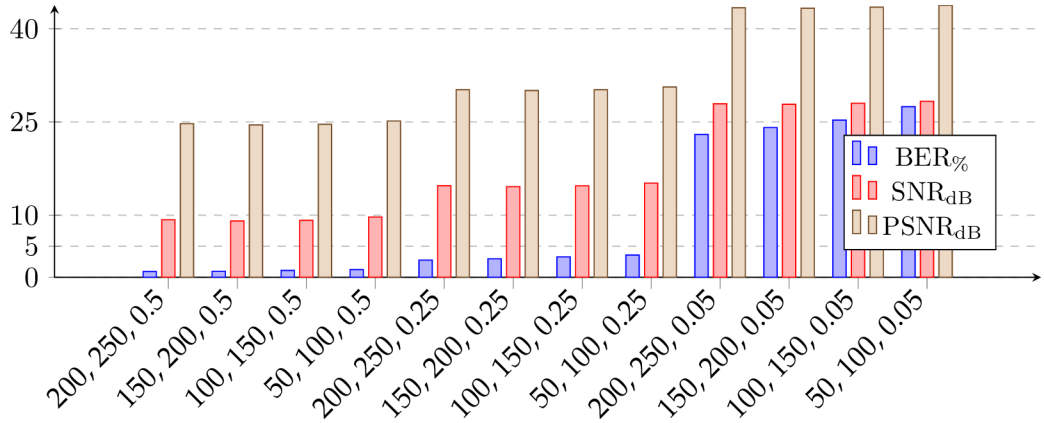



Obrázek 5.6: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ s modifikacemi metody ukrývání pomocí ozvěny se dvěma bipolárními ozvěnami na bit.

Autoři této metody [21] nezmínili kapacitu této varianty, ale z ručního testování podobnému jako v předchozí variantě byla kapacita také ~15 bitů za sekundu při vzorkovací frekvenci 44,1 kHz a použitím hodnoty 0,2 pro parametr `alpha`. Výhodou této varianty je vyšší hodnota SNR a PSNR při stejné hodnotě BER. Vhodnou volbou indexů ozvěn byla dosažena hodnota 30 dB SNR a 46,6 dB PSNR.

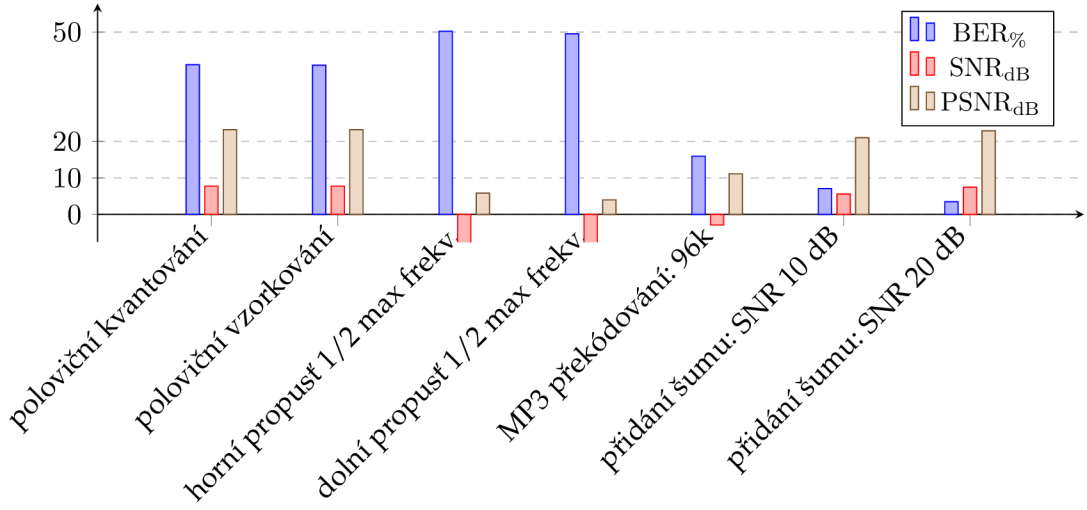
### Varianta s ozvěnou vpřed a vzad

Výsledky této varianty jsou velmi podobné první variantě, ale s mírně lepšími hodnotami, jak je vidět na obrázku 5.7.




Obrázek 5.7: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ bez modifikací pro metodu ukrývání pomocí ozvěny s ozvěnami vpřed a vzad s parametry *d0*, *d1* a *alpha* na vodorovné ose.

Výsledky této varianty s modifikacemi jsou také velmi podobné první variantě, také s mírně lepšími hodnotami, jak je vidět na obrázku 5.8.

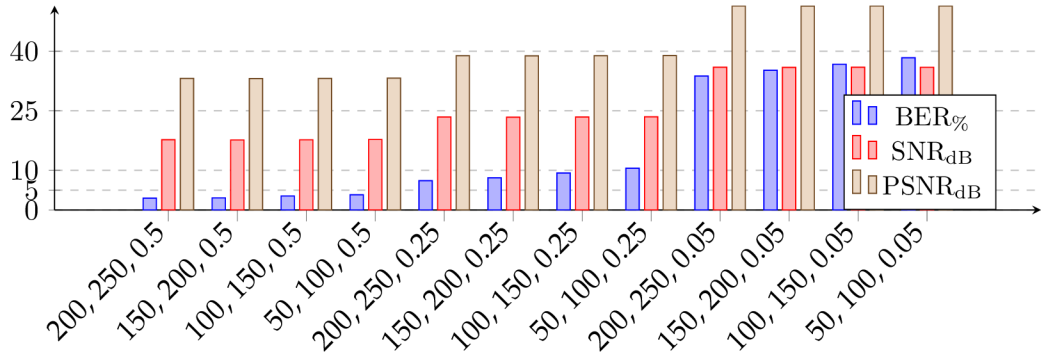



Obrázek 5.8: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ s modifikacemi metody ukrývání pomocí ozvěny s ozvěnami vpřed a vzad.

Ručním testováním jako v první a druhé variantě byla kapacita této varianty také vyhodnocena na ~15 bitů za sekundu při vzorkovací frekvenci 44,1 kHz a hodnotou 0,1 parametru `alpha`. Hodnota SNR byla 24,3 dB a 40,9 dB pro PSNR, což je sice horší výsledek než u předchozích dvou variant, ale na druhou stranu má tato varianta nejvyšší robustnost ze všech čtyř implementovaných variant jak vyplývá z obrázku 5.8. Autoři této metody [16] vyhodnotili kapacitu na 10 bitů za sekundu při vzorkovací frekvenci 44,1 kHz.

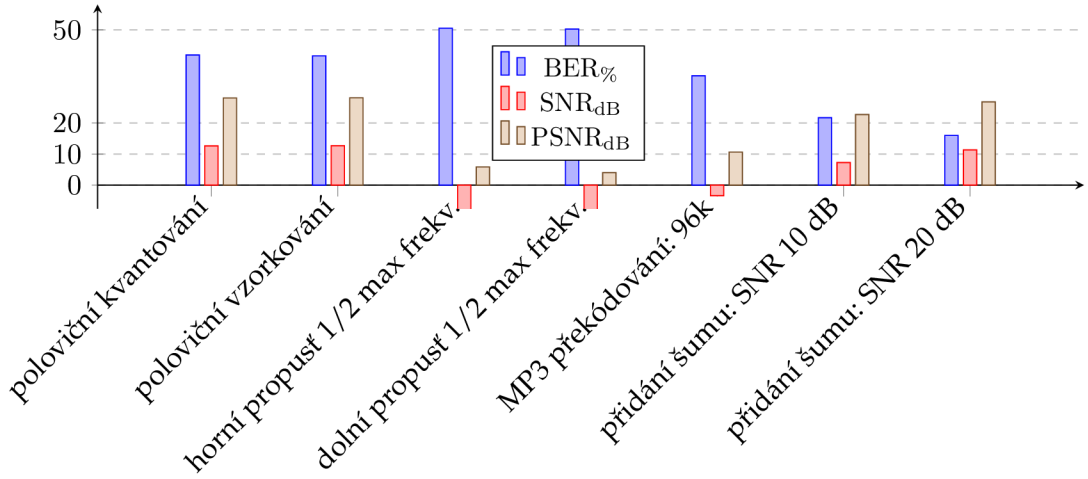
### Varianta s ozvěnami s kladnou a zápornou (bipolární) amplitudou vpřed a vzad

Tato varianta má ze všech čtyř nejvyšší hodnoty SNR a PSNR. Hodnoty BER má tato varianta podobné jako ozvěnová metoda s bipolárními ozvěnami, jak lze vidět na obrázku 5.9.




Obrázek 5.9: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ bez modifikací pro metodu ukrývání pomocí ozvěny s bipolárními ozvěnami před a vzad s parametry *d0*, *d1* a *alpha* na vodorovné ose.

I s modifikacemi má tato varianta téměř totožné výsledky jako ozvěnová metoda s bipolárními ozvěnami, jak lze vidět na obrázku 5.10).

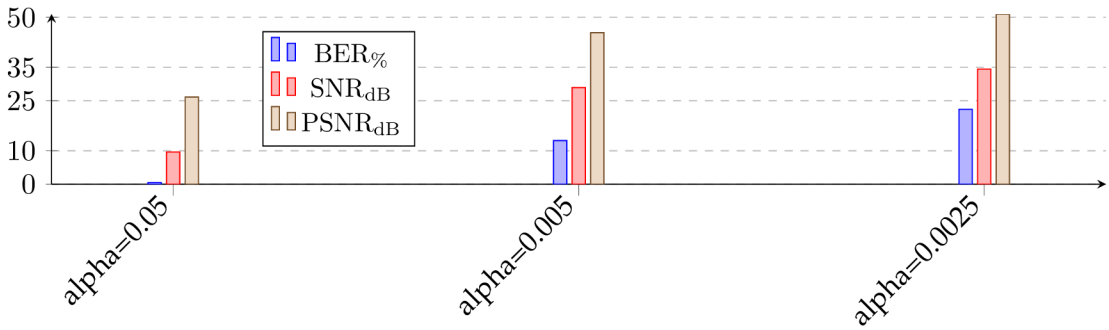



Obrázek 5.10: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ s modifikacemi metody ukrývání pomocí ozvěny s bipolárními ozvěnami před a vzad.

Ručním testováním jako v ostatních variantách byla kapacita této varianty také vyhodnocena na ~15 bitů za sekundu při vzorkovací frekvenci 44,1 kHz a hodnotami 31,9 dB SNR a 48,5 dB PSNR, což jsou nejvyšší hodnoty ze všech čtyř variant. Pro parametr `alpha` byla použita hodnota hodnota 0,2.

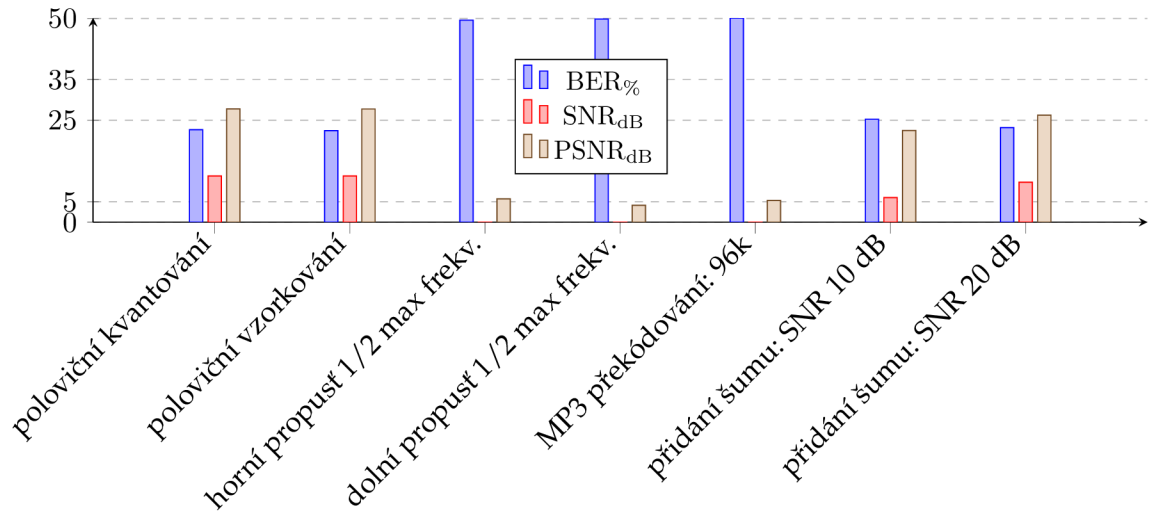
### Metoda rozprostřeného spektra

Tato metoda má velmi podobné výsledky jako ozvěnová metoda s bipolárními ozvěnami (viz část 5.2). Kódování s parametrem `alpha=0.05` odpovídá 5% maximální amplitudy. V testování se nachází pouze jako referenční hodnota pro BER blízko 0%. V praxi je tato hodnota nepoužitelná, protože je generovaný šum příliš hlasitý. Hodnoty `alpha=0.005` a `alpha=0.0025` už jsou blíže reálně použitelným hodnotám, ale stále nejsou nepostřehnutelné. Proto se hodí spíše do krycího signálu, který už sám o sobě obsahuje určité množství šumu, které zprávu zamaskuje. Nevýhodou nižších hodnot `alpha` je snížená kapacita metody a tudíž vyšší chybovost pro delší zprávy jak lze vidět z obrázku 5.11.




Obrázek 5.11: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ bez modifikací pro metodu ukrývání pomocí metody rozprostřeného spektra.

Jak lze vidět z obrázku 5.12, má tato metoda částečně odolnost proti převzorkování, překvantování a přidání šumu. Naopak má nulovou odolnost proti modifikacím, které upravují frekvenční spektrum signálu.

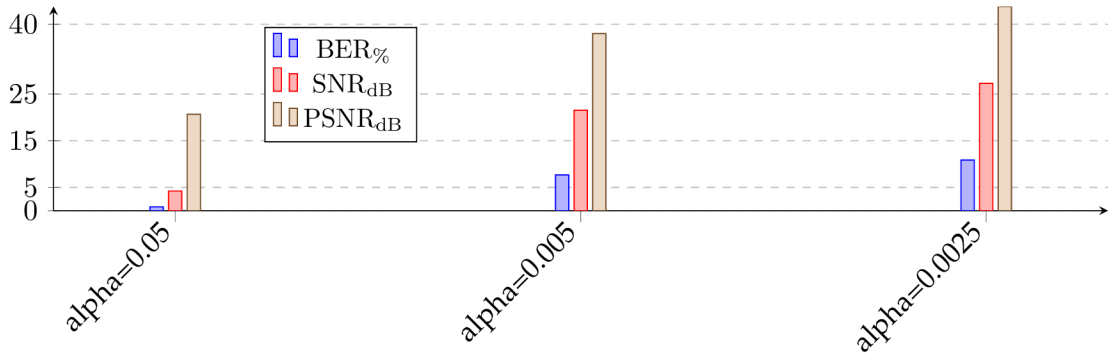



Obrázek 5.12: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ s modifikacemi metody ukrývání pomocí metody rozprostřeného spektra.

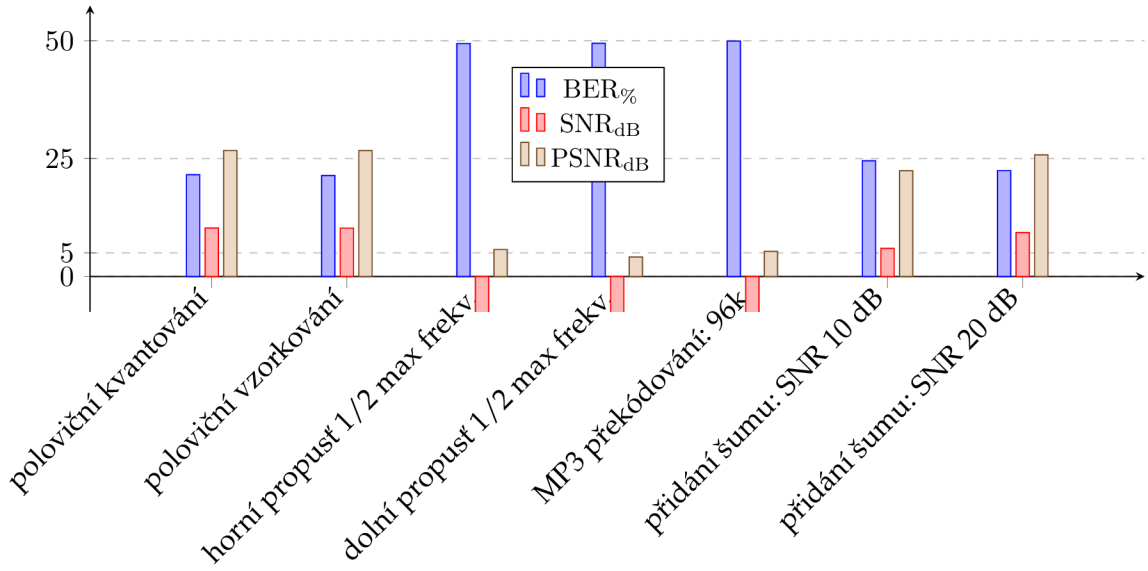
Při ručním testování byl použita hodnota 0,001 pro parametr `alpha`, při které byla vyhodnocena kapacita této metody na 4-8 bitů podle volby parametru `password`, který měl na hodnoty BER velmi vysoký vliv. Autoři [4] vyhodnotili kapacitu této metody na 4 bity za sekundu.

### Metoda rozprostřeného spektra ve Fourierově transformaci (vlastní metoda)

Rozprostírání bitů ve Fourierově transformaci krycího signálu má nižší hodnoty, pokud není stego signál modifikován, jak lze vidět z obrázku 5.13. Avšak má tato metoda mírně vyšší odolnost vůči aplikovaným modifikacím, jak lze vidět z obrázku 5.14. Kapacita této metody byla vyhodnocena také na 4-8 bitů jako u neupravené metody.




Obrázek 5.13: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ bez modifikací pro metodu ukrývání pomocí rozprostřeného spektra ve Fourierově transformaci.




Obrázek 5.14: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ s modifikacemi metody ukrývání pomocí rozprostřeného spektra ve Fourierově transformaci.

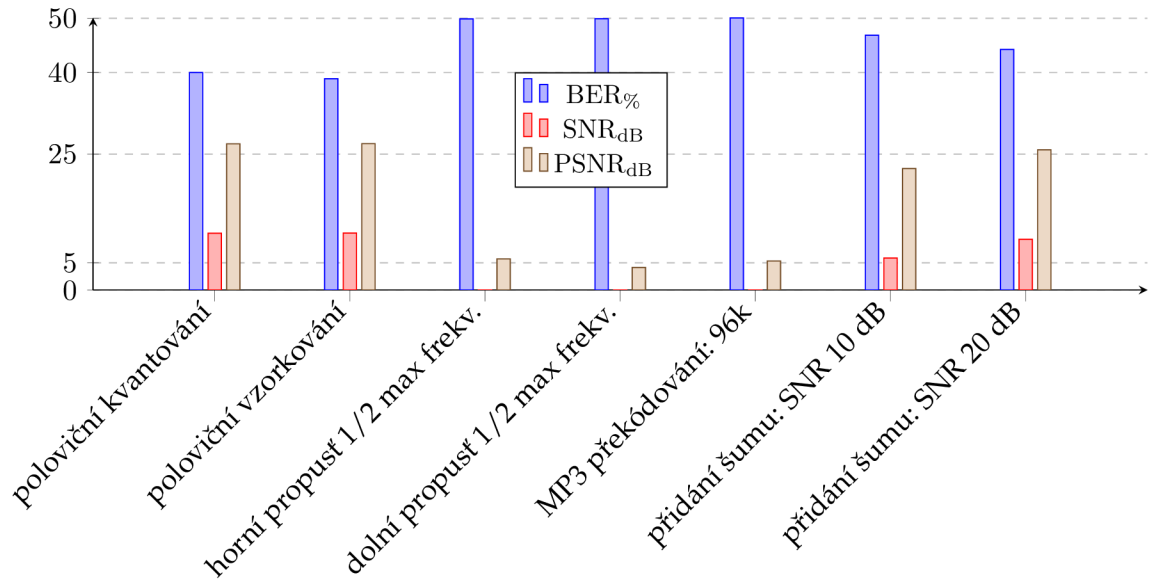
### Metoda fázového kódování

První graf byl nahrazen za tabulku 5.1, jelikož tato metoda nemá žádné parametry. Tato metoda ukrývá informace ve fázích vysokých frekvencí a má výbornou míru dekódovatelnosti bez modifikací.

Tabulka 5.1: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ bez modifikací pro metodu ukrývání pomocí metody fázového kódování.

| ber percent | snr db | psnr db |
|---|---|---|
| 0.47 | 15.61 | 32.07 |

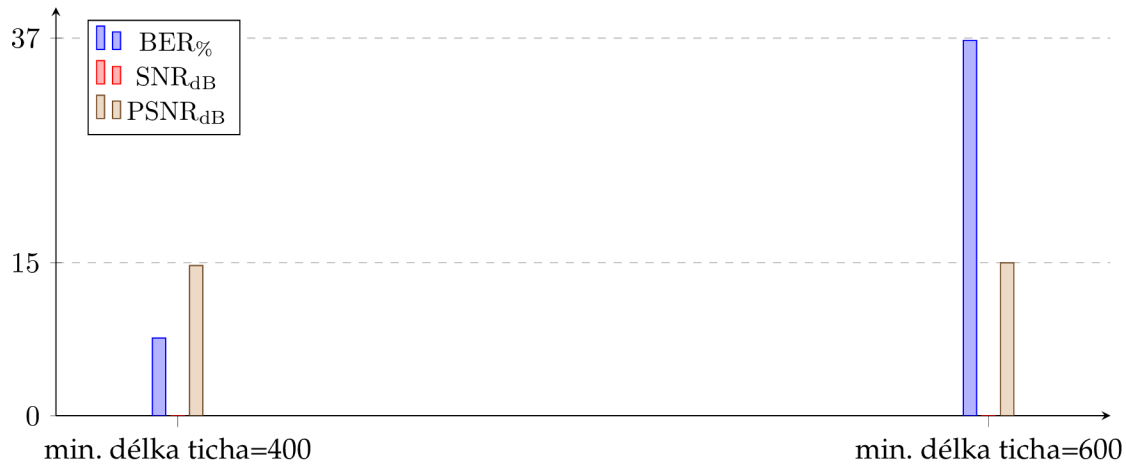
Tato metoda má velmi nízkou robustnost vůči všem aplikovaným modifikacím, jak lze vidět z obrázku 5.15. Podle autorů [4] má tato metoda kapacitu 4-16 bitů za sekundu na základě množství přítomného šumu v krycím signálu. Při ručním testování bylo zjištěno, že výsledný stego signál obsahuje praskání napříč celou délkou signálu. Toto praskání bylo možné zamaskovat použitím krycího signálu s velmi silným šumem. Maximální kapacita v signálu s šumem dosáhla až 9756 bitů za sekundu.



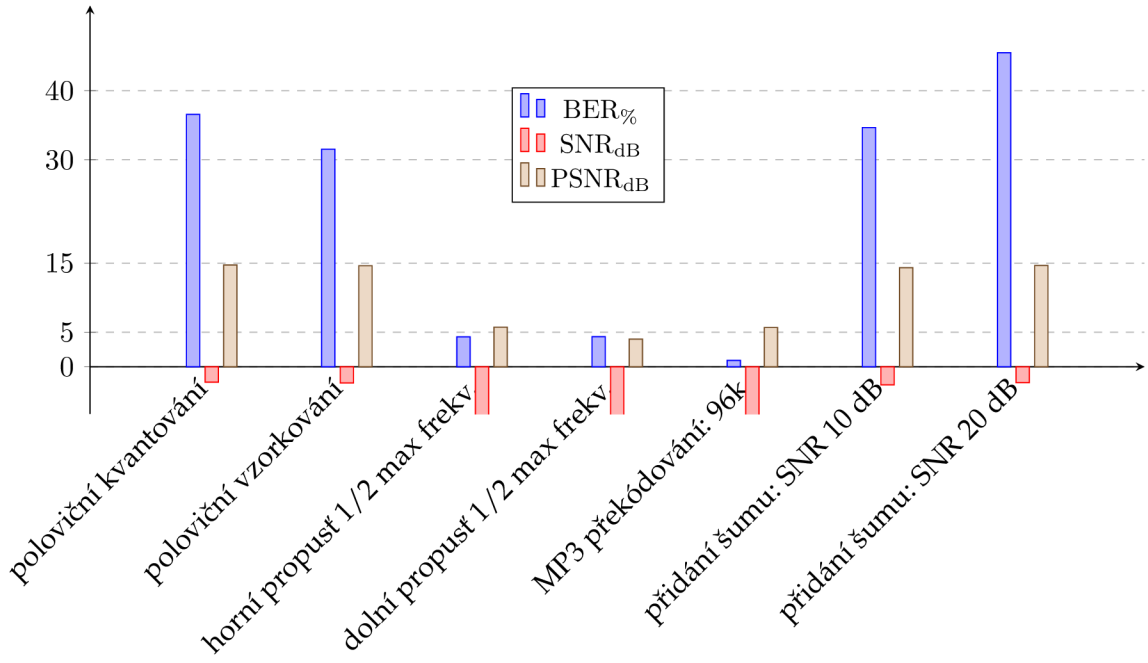

Obrázek 5.15: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ s modifikacemi metody fázového kódování.

### Metoda ukrývání v úsecích ticha

Hodnota SNR této metody v obrázku 5.16 může být zavádějící. Tuto nízkou hodnotu způsobuje posun vzorků zkrácením intervalů ticha. Na obrázku 5.17 lze vidět, že má tato metoda výbornou robustnost proti modifikacím spektra. Avšak ostatní modifikace mají silný vliv na detekci začátku a konce intervalu ticha. Pouhá změna jednoho vzorku nad hraniční hodnotu v některém z intervalů ovlivní dekódování. Kapacitu této metody není možné jednoznačně určit, protože se váže na množství úseků ticha v krycím signálu. Avšak v nerušených nahrávkách běžné řeči se kapacita pohybuje okolo 25 bitů na sekundu.



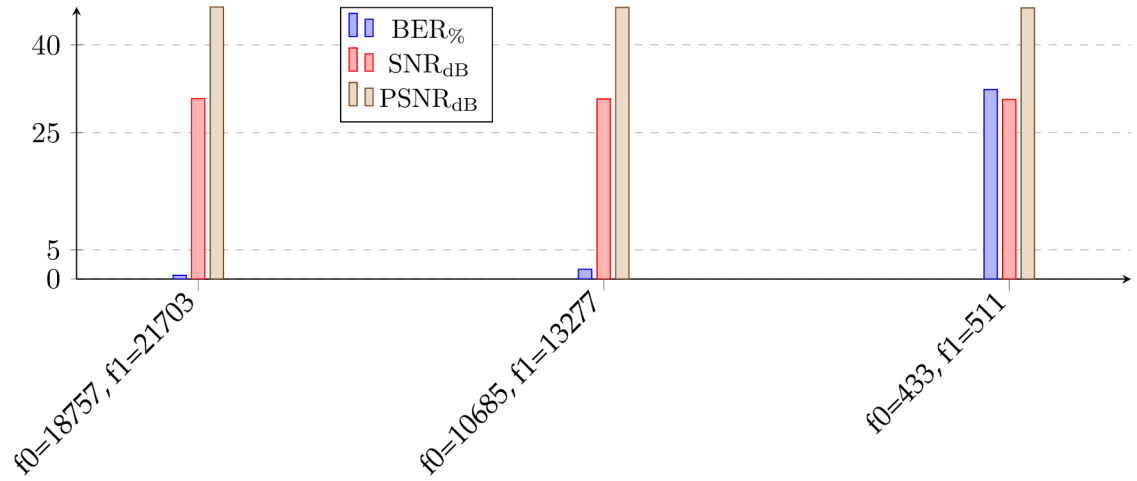

Obrázek 5.16: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ bez modifikací pro metodu ukrývání pomocí metody ukrývání v úsecích ticha.



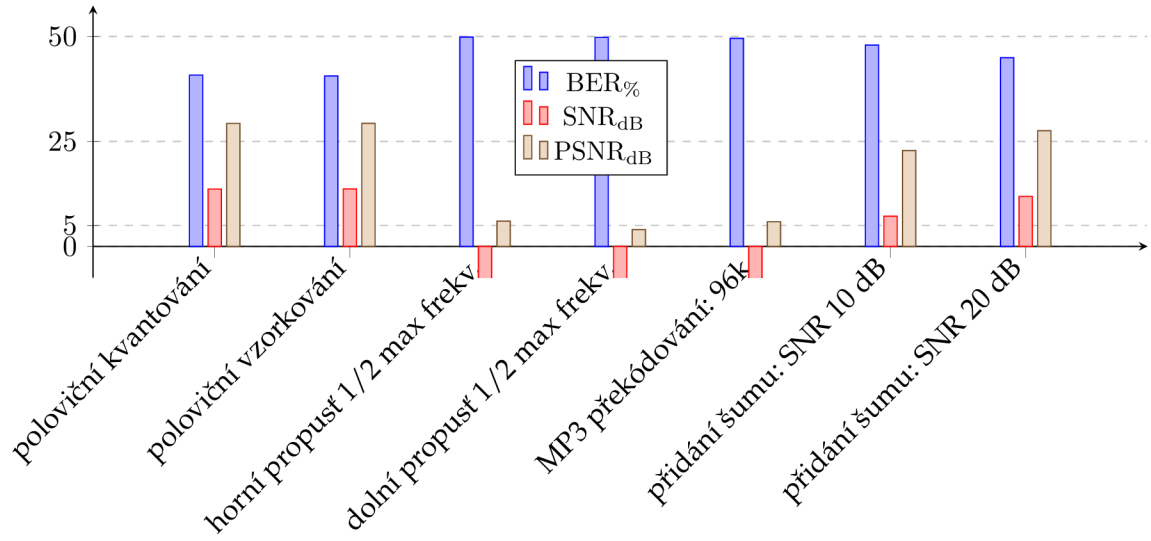

Obrázek 5.17: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ s modifikacemi metody ukrývání pomocí metody ukrývání v úsecích ticha.

### Metoda vkládání tónu

Tato metoda dosáhla výborných hodnot BER, SNR a PSNR bez modifikací, při správné volbě frekvencí vložených tónů, jak lze vidět na obrázku 5.18. Problém této metody však nastává při aplikaci modifikací. Tato metoda má extrémně nízkou robustnost vůči jakýmkoliv modifikacím, jak lze vidět na obrázku 5.19. Tento výsledek je stejně špatný jako u metody nahrazení nejméně významného bitu a metody rozprostřeného spektra ve Fourierově transformaci. Ručním testováním byla kapacita této metody vyhodnocena na 62 bitů za sekundu.




Obrázek 5.18: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ bez modifikací pro metodu ukrývání pomocí metody vkládání tónu.




Obrázek 5.19: Průměrné hodnoty $\mathrm{BER}_{\%}$, $\mathrm{SNR}_{\mathrm{dB}}$ a $\mathrm{PSNR}_{\mathrm{dB}}$ s modifikacemi metody ukrývání pomocí metody vkládání tónu.

## 5.3 Shrnutí výsledků

Pro lepší přehled výsledných hodnot nemodifikovaných stego signálů primárně z ručního testování, obsahuje tabulka 5.2 výsledné hodnoty SNR, PSNR a kapacitu metod pro krycí signál vzorkovaný na frekvenci 44,1 kHz.

Tabulka 5.2: Výsledky testovaných steganografických metod.

| Název metody | SNR | PSNR | Kapacita |
|---|---|---|---|
| Nahrazení nejméně významného bitu | 90–111 dB | 108–128 dB | 44 100 b/s |
| Ozvěny: jedna ozvěna na bit | 27,6 dB | 37 dB | 16 b/s |
| Ozvěny: ozvěny s bipolární amplitudou | 30 dB | 46,6 dB | 16 b/s |
| Ozvěny: ozvěny vpřed a vzad | 24,3 dB | 40,9 dB | 16 b/s |
| Ozvěny: s bipolární amplitudou vpřed a vzad | 31,9 dB | 48,5 dB | 16 b/s |
| Přímé rozprostřené spektrum (DSSS) | 33–43 dB | 52–59 db | 4–8 b/s |
| DSSS ve Fourierově transformaci | 36–51 dB | 54–68 dB | 4–8 b/s |
| Fázové kódování | 9–37 dB | 28–53 dB | max. 9756 b/s |
| Ukrývání v úsecích ticha | -2,8 dB | 16 dB | 25 b/s |
| Vkládání tónu | 24 dB | 42 dB | 62 b/s |

Z výsledků je patrné, že některé metody jsou k určitým účelům vhodnější než jiné. Pokud nebudou na stego signál aplikovány modifikace, pak je jasným vítězem metoda nahrazení nejméně významného bitu. Pokud je cílem uložit do audia vodoznak, pak jsou ozvěnové metody vhodnou robustní volbou. A nejlepší metodou proti modifikaci filtrace spektra byla při testování metoda ukrývání v úsecích ticha.

# Kapitola 6

# Závěr

Cílem této práce bylo vytvořit přehled metod z oblasti digitální zvukové steganografie a implementovat několik z nich. Z osmi metod popsaných v kapitole 2 jich bylo implementováno šest a jedna vlastní metoda navíc. U metod nahrazení nejméně významného bitu, metody ukrývání pomocí ozvěny a metody rozprostřeného spektra byly také implementovány modifikované varianty těchto metod.

Výsledným produktem této práce je softwarová knihovna pro programovací jazyk Python s jednoduše rozšiřitelným rozhraním metod. Knihovna také obsahuje vestavěný terminálový program pro kódování a dekódování libovolných dat do krycích audio souborů ve formátu WAV pomocí implementovaných metod. Dále obsahuje knihovna program pro vyhodnocení robustnosti a nepostřehnutelnosti na zadaných datových sadách a také program pro zpracování těchto dat. Výsledky většiny metod se shodují s výsledky autorů daných metod.

Jelikož bylo při implementaci přihlíženo spíše na množství metod a rozšiřitelnost, nebyly implementovány pokročilé techniky pro zlepšení robustnosti, nepostřehnutelnosti a kapacity. Mezi tyto techniky patří šifrování, paritní a samoopravné kódy, Huffmanovo kódování a jiné kompresní algoritmy. Dalším vylepšením do budoucna by mohla být možnost tyto techniky kombinovat stylem vrstvení.

Posledním požadavkem této práce je, aby byla co nejvíce otevřená a dostupná pro kohokoliv se zájmem o steganografii. Proto je text této práce dostupný pod licencí CC-BY-4.0[1] na adrese `https://github.com/pkabelka/audio-steganography-thesis`. Programová část této práce je dostupná pod licencí Apache-2.0 na adrese `https://github.com/pkabelka/audio-steganography`.

[1] `https://creativecommons.org/licenses/by/4.0`

# Literatura


[1] AlSabhany, A. A., Ali, A. H., Ridzuan, F., Azni, A. a Mokhtar, M. R. Digital audio steganography: Systematic review, classification, and analysis of the current state of the art. *Computer Science Review*. Listopad 2020, sv. 38, s. 100316, [cit. 2022-10-09]. DOI: https://doi.org/10.1016/j.cosrev.2020.100316. ISSN 1574-0137.

[2] Anderson, R. a Petitcolas, F. On the limits of steganography. *IEEE Journal on Selected Areas in Communications*. Květen 1998, sv. 16, č. 4, s. 474–481, [cit. 2022-10-23]. DOI: 10.1109/49.668971. ISSN 1558-0008.

[3] Bandyopadhyay, S., Bhattacharyya, D., Ganguly, D., Mukherjee, S. a Das, P. A tutorial review on steganography. In: *First International Conference on Contemporary Computing (IC3-2008)*. India, Noida: Macmillan Advanced Research Series, Srpen 2008, s. 105–114 [cit. 2023-02-08]. ISBN 023-063-619-5.

[4] Bender, W., Gruhl, D., Morimoto, N. a Lu, A. Techniques for data hiding. *IBM Systems Journal*. 1996, sv. 35, 3.4, s. 313–336, [cit. 2023-01-12]. DOI: 10.1147/sj.353.0313. ISSN 0018-8670.

[5] Boney, L., Tewfik, A. a Hamdy, K. Digital watermarks for audio signals. In: *Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems*. červen 1996, s. 473–480 [cit. 2022-12-27]. DOI: 10.1109/MMCS.1996.535015.

[6] Černocký, J. *Úvod do signálů a systémů*. Brno: Fakulta informačních technologií VUT v Brně, prosinec 2021 [cit. 2022-10-30]. 123 s.

[7] Cvejic, N. a Seppanen, T. Increasing the capacity of LSB-based audio steganography. In: *2002 IEEE Workshop on Multimedia Signal Processing.* Prosinec 2002, s. 336–338 [cit. 2022-09-30]. DOI: 10.1109/MMSP.2002.1203314.

[8] Cvejic, N. a Seppanen, T. A wavelet domain LSB insertion algorithm for high capacity audio steganography. In: *Proceedings of 2002 IEEE 10th Digital Signal Processing Workshop, 2002 and the 2nd Signal Processing Education Workshop.* říjen 2002, s. 53–55 [cit. 2023-02-15]. DOI: 10.1109/DSPWS.2002.1231075.

[9] Djebbar, F. a Ayad. Audio steganography by phase modification. In: Falk, R. a Westphall, C., ed. *SECURWAVE 2014 – 8th International Conference on Emerging Security Information*. Lisbon, Portugal: IARIA, Listopad 2014, s. 31–35 [cit. 2023-03-07]. ISBN 978-1-61208-376-6.

[10] Djebbar, F., Ayad, B., Meraim, K. A. a Hamam, H. Comparative study of digital audio steganography techniques. *EURASIP Journal on Audio, Speech, and Music Processing*

[online]. Říjen 2012, sv. 2012, č. 25, [cit. 2022-09-25]. DOI: 10.1186/1687-4722-2012-25. ISSN 1687-4722. Dostupné z: https://doi.org/10.1186/1687-4722-2012-25.

[11] Dutta, H., Das, R. K., Nandi, S. a Prasanna, S. R. M. An Overview of Digital Audio Steganography. *IETE Technical Review* [online]. Taylor & Francis. Prosinec 2020, sv. 37, č. 6, s. 632–650, [cit. 2022-09-25]. DOI: 10.1080/02564602.2019.1699454. Dostupné z: https://doi.org/10.1080/02564602.2019.1699454.

[12] Fleischman, E. *WAVE and AVI Codec Registries* [Internet Requests for Comments]. RFC 2361. RFC Editor, červen 1998 [cit. 2023-02-21]. Dostupné z: https://www.rfc-editor.org/info/rfc2361.

[13] Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., Pallett, D. S. et al. *TIMIT Acoustic-Phonetic Continuous Speech Corpus* [online]. Philadelphia: Linguistic Data Consortium, 1993 [cit. 2023-01-25]. DOI: 10.35111/17gk-bn40. Dostupné z: https://doi.org/10.35111/17gk-bn40.

[14] Gruhl, D., Lu, A. a Bender, W. Echo hiding. In: Anderson, R., ed. *Information Hiding*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, s. 295–315 [cit. 2022-10-09]. ISBN 978-3-540-49589-5.

[15] Hu, Y. a Loizou, P. Subjective Comparison of Speech Enhancement Algorithms. In: *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*. Květen 2006, sv. 1, s. I–I [cit. 2023-01-25]. DOI: 10.1109/ICASSP.2006.1659980. ISSN 2379-190X.

[16] Kim, H. J. a Choi, Y. H. A novel echo-hiding scheme with backward and forward kernels. *IEEE Transactions on Circuits and Systems for Video Technology*. Srpen 2003, sv. 13, č. 8, s. 885–889, [cit. 2022-12-23]. DOI: 10.1109/TCSVT.2003.815950. ISSN 1558-2205.

[17] Kuznetsov, A., Onikiychuk, A., Peshkova, O., Gancarczyk, T., Warwas, K. et al. Direct Spread Spectrum Technology for Data Hiding in Audio. *Sensors (Basel)*. Basel, Switzerland: Multidisciplinary Digital Publishing Institute. Duben 2022, sv. 22, č. 9, [cit. 2022-12-23]. DOI: 10.3390/s22093115. ISSN 1424-8220.

[18] Linear Pulse Code Modulated Audio (LPCM). *Sustainability of Digital Formats* [online]. 2022-04-26 [cit. 2022-11-01]. Dostupné z: https://www.loc.gov/preservation/digital/formats/fdd/fdd000011.shtml.

[19] Majeed, M. A., Sulaiman, R., Shukur, Z. a Hasan, M. K. A Review on Text Steganography Techniques. *Mathematics* [online]. Listopad 2021, sv. 9, č. 21, [cit. 2023-04-22]. DOI: 10.3390/math9212829. ISSN 2227-7390. Dostupné z: https://www.mdpi.com/2227-7390/9/21/2829.

[20] Oak, D. *Audio steganography using phase encoding technique* [online], 29. dubna 2016. 2018-05-25 [cit. 2023-03-07]. Dostupné z: https://dsp.stackexchange.com/a/30464.

[21] Oh, H. O., Seok, J. W., Hong, J. W. a Youn, D. H. New echo embedding technique for robust and imperceptible audio watermarking. In: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*. Květen 2001, sv. 3, s. 1341–1344 [cit. 2022-12-27]. DOI: 10.1109/ICASSP.2001.941176. ISSN 1520-6149.

[22] Oliver, B., Pierce, J. a Shannon, C. The Philosophy of PCM. *Proceedings of the IRE*. Listopad 1948, sv. 36, č. 11, s. 1324–1331, [cit. 2022-10-30]. DOI: 10.1109/JRPROC.1948.231941. ISSN 2162-6634.

[23] Paul. *NumPy grouping using itertools.groupby performance* [online], 10. ledna 2011. 2021-01-14 [cit. 2023-03-08]. Dostupné z: https://stackoverflow.com/a/4652265.

[24] Pooyan, M. a Delforouzi, A. LSB-based Audio Steganography Method Based on Lifting Wavelet Transform. In: *2007 IEEE International Symposium on Signal Processing and Information Technology*. Prosinec 2007, s. 600–603 [cit. 2023-02-17]. DOI: 10.1109/ISSPIT.2007.4458198. ISSN 2162-7843.

[25] Prabakaran., G. a Bhavani., R. A modified secure digital image steganography based on Discrete Wavelet Transform. In: *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*. Březen 2012, s. 1096–1100 [cit. 2023-02-17]. DOI: 10.1109/ICCEET.2012.6203811.

[26] Ramkumar, M., Akansu, A. a Alatan, A. A robust data hiding scheme for images using DFT. In: *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*. říjen 1999, sv. 2, s. 211–215 [cit. 2023-02-28]. DOI: 10.1109/ICIP.1999.822886.

[27] Salamon, J., Jacoby, C. a Bello, J. P. A Dataset and Taxonomy for Urban Sound Research. In: *22nd ACM International Conference on Multimedia (ACM-MM'14)*. Orlando, FL, USA: Association for Computing Machinery, Listopad 2014, s. 1041–1044 [cit. 2023-01-25]. DOI: 10.1145/2647868.2655045. ISBN 9781450330633.

[28] Shannon, C. Communication in the Presence of Noise. *Proceedings of the IRE*. Leden 1949, sv. 37, č. 1, s. 10–21, [cit. 2022-10-30]. DOI: 10.1109/JRPROC.1949.232969. ISSN 2162-6634.

[29] Shirali Shahreza, S. a Shirali Shahreza, M. Steganography in Silence Intervals of Speech. In: *2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. Srpen 2008, s. 605–607 [cit. 2023-02-13]. DOI: 10.1109/IIH-MSP.2008.5.

[30] Swanson, M. D., Zhu, B., Tewfik, A. H. a Boney, L. Robust audio watermarking using perceptual masking. *Signal Processing*. Květen 1998, sv. 66, č. 3, s. 337–355, [cit. 2022-09-30]. DOI: 10.1016/S0165-1684(98)00014-0. ISSN 0165-1684.

[31] Tekeli, K. a Asliyan, R. A COMPARISON OF ECHO HIDING METHODS. In: *The Eurasia Proceedings of Science Technology Engineering and Mathematics* [online]. ISRES Publishing, Listopad 2017, sv. 1, č. 53, s. 397–403 [cit. 2022-10-09]. ISSN 2602-3199. Dostupné z: http://www.epstem.net/en/pub/issue/31865/365048.

[32] Tzanetakis, G., Essl, G. a Cook, P. *Automatic Musical Genre Classification Of Audio Signals* [online]. The International Society for Music Information Retrieval, 2001 [cit. 2023-01-25]. Dostupné z: http://ismir2001.ismir.net/pdf/tzanetakis.pdf.

[33] Wu, W. chih a Chen, O.-C. Analysis-by-Synthesis Echo Hiding Scheme Using Mirrored Kernels. In: *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*. Květen 2006, sv. 2, s. II–II [cit. 2022-12-28]. DOI: 10.1109/ICASSP.2006.1660345. ISSN 2379-190X.

# Příloha A

# Obsah přiloženého média

```
text/                                   text této práce
|   ...
program/
|   audio-steganography.sh              skript pro jednodušší spuštění
|   requirements.txt                    seznam potřebných Python knihoven
|   README.md                           návod na použití v angličtině
|   README-CZ.md                        návod na použití v češtině
|   audio_steganography/
|   |   cli/
|   |   |   __init__.py                 obsahuje funkci main() programu
|   |   |   evaluation/
|   |   |   |   process.py              skript pro zpracování výsledků
|   |   |   |   __main__.py             spouští skript pro vyhodnocení
|   |   |   |   ...
|   |   |   ...
|   |   methods/                        implementované metody
|   |   |   echo_bipolar.py
|   |   |   echo_bf.py
|   |   |   echo_bipolar_bf.py
|   |   |   echo_single.py
|   |   |   dsss.py
|   |   |   dsss_dft.py
|   |   |   lsb.py
|   |   |   tone_insertion.py
|   |   |   phase_coding.py
|   |   |   silence_interval.py
|   |   |   ...
|   |   __main__.py
|   |   __init__.py                     spouští vestavěný program
|   |   ...
|   ...
data                                    vlastní datová sada
|   A Narrated Tour of the Moon
|   |   22050 Hz
|   |   |   ...
```

# Příloha B

# Statistické funkce

Tyto funkce slouží k vyhodnocování vlastností steganografických metod. V rovnicích představuje $x$ krycí signál, $y$ představuje výsledný stego signál po zakódování informace a $n$ je počet prvků v $x$ a $y$.

$$\text{MSE} = \frac{\sum (x-y)^2}{n} \tag{B.1}$$

$$\text{RMSD} = \sqrt{\text{MSE}(x,y)} \tag{B.2}$$

$$\text{SNR} = 10 \times \log_{10} \frac{\sum x^2}{\sum (x-y)^2}, \qquad [\text{dB}] \tag{B.3}$$

$$\text{PSNR} = 10 \times \log_{10} \frac{n}{\sum (x-y)^2}, \qquad [\text{dB}] \tag{B.4}$$

$$\text{BER} = \frac{\text{počet nesprávně dekódovaných bitů}}{\text{počet zakódovaných bitů}} \times 100, \qquad [\%] \tag{B.5}$$