VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

DIGITÁLNÍ AUDIO STEGANOGRAFIE

DIGITAL AUDIO STEGANOGRAPHY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAN MORÁVEK**
AUTHOR

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

BRNO 2018

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Morávek Jan**
Obor: Informační technologie
Téma: **Digitální zvuková steganografie**
Digital Audio Steganography
Kategorie: Bezpečnost

Pokyny:

1. Vytvořte přehled metod z oblasti digitální steganografie pro skrývání informace ve zvukových datech (dále jen "zvuková steganografie"), jejich vlastností a shrňte současný stav v této oblasti.
2. Zvolte typ skrývané informace (textová, obrazová apod.) a její vlastnosti. Na základě existující či vlastní analýzy způsobů ukládání zvukových dat a typu skrývané informace zužte a zvolte vhodné způsoby pro ukládání dat a vhodné metody pro zvukovou steganografii.
3. Implementujte několik existujících metod zvukové steganografie, zvažte jejich modifikace, popř. návrh a implementace vlastních metod.
4. Demonstrujte a vyhodnoťte funkčnost a vlastnosti implementovaných metod z hlediska skrývání a odkrývání zvoleného typu informace.
5. Dosažené výsledky diskutujte, navrhněte možné návaznosti a rozšíření předloženého řešení.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Strnadel Josef, Ing., Ph.D.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

V současnosti se stále více informací předává a uchovává v elektronické podobě. Právě digitalizace umožňuje jednodušší přístup k informacím nejenom pro uživatele, ale i pro případné útočníky. Tento fakt vede k rozvoji oblasti informační bezpečnosti. Vyjma kryptografie do této oblasti spadá i steganografie. Tato práce se zabývá obecným popisem steganografie v dnešním světě a konkrétněji se zabývá digitální audio steganografií. Součástí práce je implementace několika vybraných metod a jejich vyhodnocení.

Abstract

Nowadays much more information is transfered and saved in electronic files. The digitalization allows easier access to information, not only for the user, but also for a possible attacker. This fact leads to progress in information security area. Besides cryptography, this also includes steganography. This thesis is focused on steganography in general and describes digital audio steganography in detail. Implementation and evaluation of several methods are also parts of this thesis.

Klíčová slova

steganografie, audio, bezpečnost, skrytá zpráva, komunikace, WAVE, MP3

Keywords

steganography, audio, security, secret message, communication, WAVE, MP3

Citace

MORÁVEK, Jan. *Digitální audio steganografie.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Josef Strnadel, Ph.D.

Digitální audio steganografie

Prohlášení

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

.....
Jan Morávek
15. května 2018

Poděkování

Velice rád bych poděkoval panu Ing. Josefu Strnadelovi, Ph.D., za ochotu a vřelý přístup při vedení práce. Dále bych rád poděkoval všem, kteří mi pomáhali při testování výsledků práce.

Obsah

Kapitola 1

Úvod

Bezpečnost informací má v dnešním světě velkou váhu. Současné trendy směřují k většímu používání digitálních technik pro komunikaci i pro uchovávání informací. Tento fakt můžeme pozorovat v řadě různých odvětví, příkladem nám může být směřování k digitalizaci státní správy, kde je uchováváno velké množství osobních dat, nebo rozšiřování sociálních sítí, kde probíhá komunikace obsahující velké množství osobních informací. Právě díky tomuto faktu dochází k rozvoji vědeckých disciplín zabývajících se bezpečností informací.

Jednou z vědeckých disciplín v této oblasti je i steganografie. Steganografie se primárně věnuje problematice skrytí citlivých informací pro zajištění bezpečné komunikace. Tato bakalářská práce se konkrétně věnuje oblasti digitální audio steganografie, tedy ukrývání citlivých informací do souborů s audio nahrávkami.

Cílem této práce je dokumentovat současný stav v oblasti digitální audio steganografie a vytvořit přehled využívaných metod pro ukrývání dat do zvukových souborů a jejich vlastností. Dále je obsahem práce implementace několika metod. Závěr práce obsahuje evaluaci implementovaných metod a návrh jejich rozšíření.

V kapitole 2 je obecně popsána problematika steganografie jako vědní disciplíny a rozdíl oproti ostatním vědním disciplínám z oblasti bezpečnosti. Je zde popsána historie vývoje této metody až do současné doby. V poslední části kapitoly je popsán princip fungování digitální steganografie a její dělení do kategorií.

V kapitole 3 jsou rozebrány využívané soubory pro uschování informací v digitální audio steganografii. Jsou zde popsány druhy využívaných zvukových nahrávek a také konkrétní využívané formáty zvukových souborů.

Kapitola 4 je věnována samotným metodám z oblasti digitální audio steganografie. Jsou zde popsány vlastnosti, podle kterých lze metody evaluovat. Následuje popis samotných metod, příklady jejich využití a případné zhodnocení jejich výhod a nevýhod.

V kapitole 5 je popsána implementace vytvořené aplikace. Fungování této aplikace, zvolený programovací jazyk, využité knihovny a funkce. Kapitola také podrobně popisuje implementaci vybraných metod.

V kapitole 6 je blíže popsán postup evaluace implementovaných metod. Také jsou zde uvedeny výsledky testování a jednotlivé metody jsou porovnány.

V kapitole 7 jsou popsány výsledky práce. Je zde zhodnocena funkčnost vytvořené aplikace a jsou navržena případná rozšíření.

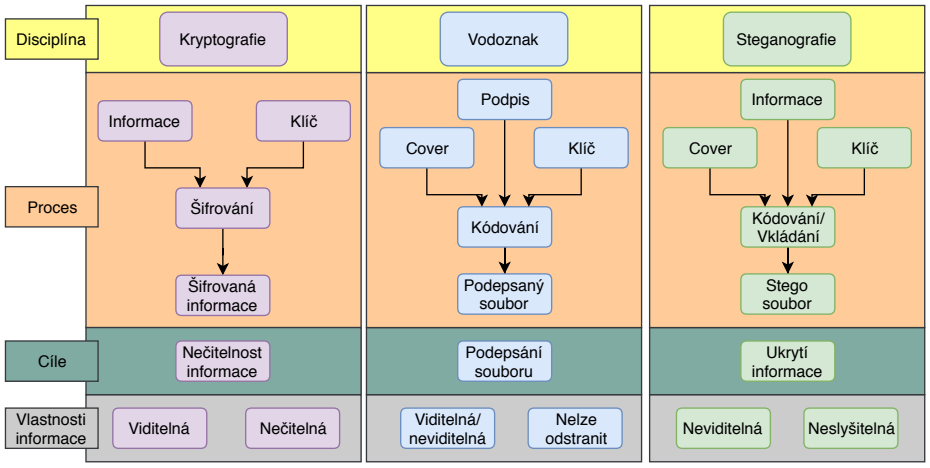
Kapitola 2

Vědní disciplína steganografie

V této kapitole jsou stručně vysvětleny základní principy steganografie. V sekci 2.1 je popsána podstata klasické steganografie a její rozdíl oproti kryptografii a vodoznaku. V sekci 2.2 je rozebrán historický vývoj této disciplíny do vzniku digitální steganografie, která se využívá v současné době. Poslední část 2.3 se zabývá právě samotnou digitální steganografií. Rozebírá principy digitální steganografie, její využití a do jakých kategorií ji lze dělit.

2.1 Obecný význam steganografie

Steganografie je vědní disciplína s dlouhou historií vývoje a je úzce spojena s kryptografií a vodoznakem. Všechny tři jmenované disciplíny se zabývají ochranou dat, nicméně každá má své jasné teoretické vymezení. Název steganografie vznikl složením řeckých slov „steganos“ – „ukrytý“ a „graphein“ – „psát“ a volně přeložen znamená „skryté psaní“ [18], což už samo naznačuje podstatu této disciplíny.

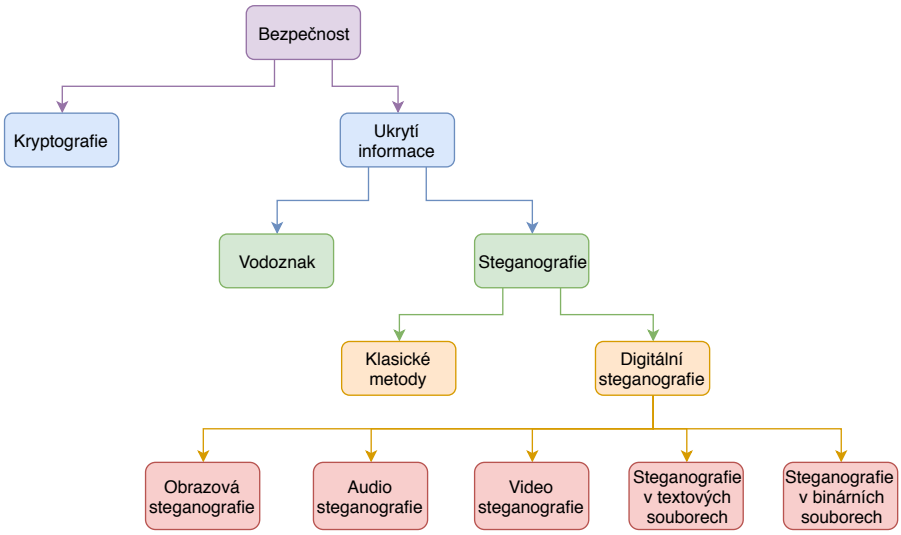


Obrázek 2.1: Ukázka rozdílů mezi jednotlivými disciplínami na různých úrovních.

Pro bližší pochopení steganografie bychom si měli nejdříve stručně přiblížit zbylé dvě disciplíny. Cílem kryptografie je změnit citlivé informace do takové podoby, ve které nejsou čitelné bez znalosti určitého klíče. Můžeme říci, že informace zůstává po zašifrování viditelná, avšak nečitelná. Oproti tomu u vodoznaku vkládáme krátkou informaci do určitého dokumentu a tím tento dokument můžeme podepsat [12]. Informace nemusí, ale může být

viditelná, nicméně by ji nemělo být možné odstranit bez zničení samotného dokumentu. Rozdíl mezi disciplínami je ukázán na obrázku 2.1.

Cílem steganografie je uschování většího množství citlivých informací na místo, kde nebudou očekávány. Tedy to, v čem se informace ukrývají, je všem na očích, ale samotné informace zůstávají skryty [7]. Nejčastěji jsou informace uschovávány pro následný bezpečný přenos těchto informací. Pokud dochází k přenášení informací mezi dvěma stranami, měl by se celý přenos jevit, pro třetí stranu, jako naprosto běžný. V případě, kdy třetí strana zjistí, že dochází k přenosu utajovaných informací, jsou citlivé informace automaticky prozrazeny a steganografická metoda selhala. Z tohoto důvodu se samotné steganografické metody ve většině případů nepoužívají a často jsou kombinovány s kryptografickými metodami. Informace je pomocí kryptografie zašifrována a poté je pomocí steganografie uschována. Pro správnou funkčnost steganografických metod se předpokládá, že princip použité metody je znám oběma stranám, které spolu komunikují.



Obrázek 2.2: Ukázka větvení disciplín zabývajících se bezpečností.

2.2 Historie a vývoj steganografie

První zmínky o využití steganografických metod pochází z dob starověkého Řecka. V 5. století byla využita metoda, při které byla otrokovi na oholenou hlavu vytetována zpráva. Když mu vlasy znovu vyrostly, tak zpráva nebyla na první pohled viditelná a otrok mohl být vyslán, aby zprávu doručil [8]. Jinou možností bylo napsat zprávu na papír, který byl zmačkán do kuličky a obalen ve vosku. Vosková kulička se dala následně spolknout bez poškození zprávy a ta mohla být nepozorovaně přepravena. Další metodou bylo vyrytí zprávy do dřevěné destičky, která byla následně zalita voskem a destička se zdála na první pohled neporušena.

O první rozsáhlejší zdokumentování se zasloužil opat Johannes Trithemius, když roku 1499 sepsal dílo s názvem „Steganographia“. V tomto díle jako první sepsal postupy kryptografických a steganografických metod, které sám vytvořil, a tím položil základy pro další rozvoj v této oblasti [8]. Poslední větší rozmach klasické steganografie, před příchodem digitální steganografie, proběhl během druhé světové války. V této době došlo ke zdokonalení metody s využitím neviditelného písma a vznikla chemická neviditelná písma, která se ob-

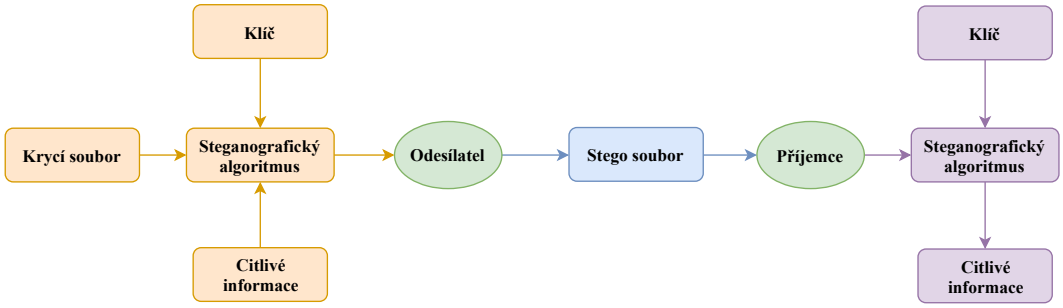
jevila jen při nanesení konkrétní chemikálie. Jinou známou metodou z tohoto období je metoda mikroteček [8].

2.3 Digitální steganografie

V dnešním světě se stále více informací uchovává či předává v digitální podobě. Tento fakt vedl i k rozvoji steganografických metod a vzniku digitální steganografie. Jak již bylo dříve zmíněno, tak steganografie využívá něco, do čeho může uschovat citlivé informace. V případě digitální steganografie se jedná o libovolný soubor. Právě podle typu tohoto souboru můžeme rozlišovat několik druhů digitální steganografie [7]. Mezi nejrozšířenější patří například:

- digitální audio steganografie,
- digitální obrazová steganografie,
- digitální textová steganografie,
- digitální video steganografie.

Soubor, do kterého jsou informace uschovány, nazýváme *krycí soubor* (cover file) [13]. Rozdělení metod podle typu krycího souboru je ukázáno na obrázku 2.2. Právě druh krycího souboru nám definuje, jaké metody můžeme využít pro uschování informací, protože například metodu, která pracuje s mezerami v textu, těžko aplikujeme na audio soubor. Druh krycího souboru je vhodné zvolit podle prostředí, ve kterém budeme soubory přenášet, tak, aby přenášené soubory budily co nejmenší podezření. V této práci se však zabýváme pouze audio soubory.



Obrázek 2.3: Schéma průběhu komunikace s využitím steganografické metody.

Podle zvolené metody máme jasný klíč, který nám určuje, jak jsou informace uschovány a jak je znovu získat. Pokud uschováme informace pomocí klíče do nějakého krycího souboru, tak nám vznikne soubor, který je odlišný od krycího souboru. Pokud by soubor byl stejný, tak by ukládané informace byly prázdné. Nově vzniklý soubor se nazývá *stego soubor* [13]. Ukázku probíhající komunikace s využitím steganografické metody můžeme vidět na obrázku 2.3.

Jak již bylo popsáno, tak druhy digitální steganografie se liší typem krycího souboru a využívanými metodami. Tedy digitální audio steganografie je definována zvukovými krycími soubory, které jsou popsány v kapitole 3 a využívanými metodami, které jsou popsány v kapitole 4.

Kapitola 3

Krycí soubory digitální audio steganografie

Tato kapitola pojednává o krycích souborech využívaných v digitální audio steganografii. V sekci 3.1 jsou naznačeny výhody a nevýhody využití zvukových souborů jako krycích souborů. Dále jsou v sekci 3.2 zmíněny typy zvukových souborů, které se nejčastěji využívají a důvody jejich využití. Hlavní částí této kapitoly je sekce 3.3, popisující zvukové formáty, na které jsem se během mé práce zaměřil a jsou podporovány implementovanou aplikací.

3.1 Výhody a nevýhody zvukových souborů

Pro efektivní využití digitální steganografie je nutné, aby přenos krycího souboru nevzbuzoval žádné podezření pro třetí stranu. Z tohoto důvodu přispěl k rozvoji digitální audio steganografie masový přenos zvukových souborů ve společnosti. Například přenos písničky mezi dvěma lidmi je v dnešní době naprosto běžný a nevzbuzuje žádné podezření. Díky tomu jsou zvukové soubory ideální volbou jako krycí soubory pro steganografické metody.

Nevýhodou využití audio souborů jako krycích souborů je úroveň fyziologického zpracování zvuku u člověka. Sluchové ústrojí je mnohem citlivější než zrakové ústrojí [12] a z tohoto důvodu je zvukový soubor horší volbou jako krycí soubor než obrazový soubor [12]. Nicméně i sluchové ústrojí má své omezení, a tak i zvukové soubory je možné použít jako krycí soubory, avšak rozsah změn v audio souborech musí být menší.

3.2 Typy zvukových souborů

Jako zvukový soubor můžeme označit nahrávku jakéhokoliv zvuku uloženého do souboru v audio formátu. Tedy je zřejmé, že zvukový soubor je široký pojem pokrývající mnoho různých druhů nahrávek. Steganografické metody by měly být v ideálním případě efektivní pro všechny druhy zvukových souborů, avšak pokud je vytvořena metoda zaměřená na jeden konkrétní druh, tak je možné dosáhnout větší efektivity této metody. V mé práci jsem se zaměřil na dva nejvíce využívané druhy zvukových souborů.

Prvním typem zvukového souboru jsou nahrávky hudby. Dále můžeme tuto oblast dělit na menší části a to například dle různých žánrů hudby, které v dnešní době existují, ale pro tuto práci jsem uvažoval hudební nahrávky jako jeden velký celek. Do této oblasti jsem zařadil jak akustické nahrávky, tak i elektronickou hudbu. Jsou zde zahrnuty nahrávky obsahující zpěv, ale také nahrávky čistě instrumentální.

Druhým typem zvukových souborů jsou nahrávky mluveného slova. Tato oblast je již menším celkem než hudba, avšak dá se také dále dělit. Do této oblasti jsem zahrnul nahrávky různých přednášek, rozhovorů, ale také monology, při kterých osoba například čte nějaký seznam. Dále sem patří, v dnešní době velice rozšířené, audio knihy. Zajímavým prvkem v této oblasti, který jsem využil v mé práci, je převod textu na zvukovou nahrávku pomocí speciálních programů.

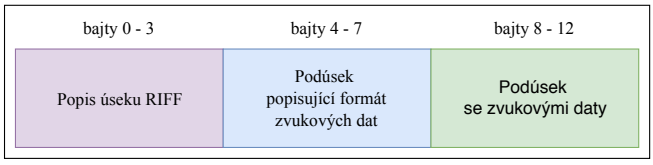
Mezi další typy zvukových souborů můžeme zařadit nahrávky zvuku v různém prostředí, jako je například zvuk přírody, ptačí zpěv nebo jen hluk ulice v rušném městě. Jako krycí soubor se dá použít i jakýkoliv druh šumu. Nicméně u většiny těchto typů zvuku je diskutabilní stránka bezpečnosti, jelikož není úplně běžné, aby si dvě strany posílaly nahrávku ptačího zpěvu, a tím může přenos vzbuzovat nežádanou pozornost. Právě nahrávky hudby a mluveného slova jsou nejvíce rozšířené a jsou běžně přenášeny, to je důvodem, proč jsem se ve své práci zaměřil na tyto typy.

3.3 Audio formáty

Hlavním formátem této práce je *Waveform Audio File Format*, který je základním formátem v oblasti zvukových souborů. Dále se tato práce okrajově zabývá formátem MP3, který jsem zvolil, protože je v dnešní době často využíván.

3.3.1 Waveform Audio File Format

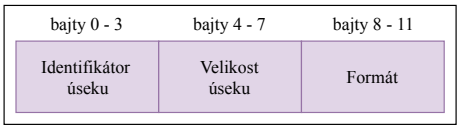
Waveform Audio File Format (dále jen WAVE) je formát používaný od roku 1991, kdy byl vytvořen firmami Microsoft a IBM jako hlavní formát pro ukládání zvuku. WAVE formát vychází ze souborového formátu *Resource Interchange File Format* (dále jen RIFF), ze kterého vychází i formát *Audio Video Interleave* (AVI) pro ukládání videa. Soubor uložený ve formátu WAVE se skládá z jednoho úseku formátu RIFF a dvou podúseků [5, 4]. Rozložení celého souboru je ukázáno na obrázku 3.1, rozložení jednotlivých úseků je ukázáno na obrázcích 3.2, 3.3 a 3.4.



Obrázek 3.1: Rozložení úseků v souboru formátu WAVE.

Struktura hlavičky celého úseku vycházejícího z formátu RIFF:

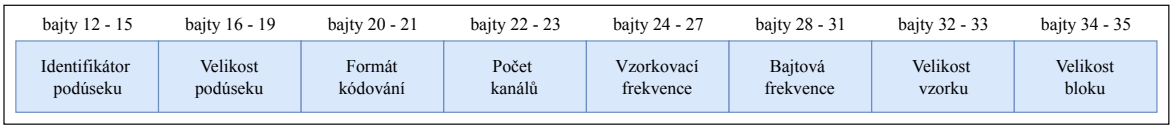
- **Identifikátor úseku** – vždy bude obsahovat název „RIFF“, každý bajt reprezentuje jedno písmeno a je uložen jako big endian.
- **Velikost úseku** – obsahuje velikost úseku v bajtech. Respektive obsahuje velikost souboru od následujícího bajtu. Tedy velikost souboru bez prvních osmi bajtů, kde je uložen název úseku a právě tato velikost. Je uložen jako little endian.
- **Formát** – v případě WAVE souborů bude vždy obsahovat název „WAVE“, každý bajt reprezentuje jedno písmeno a je uložen jako big endian.



Obrázek 3.2: Struktura hlavičky úseku RIFF.

Struktura podúseku obsahujícího formát zvukových dat:

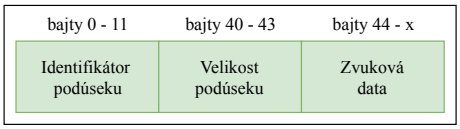
- **Identifikátor podúseku** – vždy bude obsahovat název „fmt“, každý bajt reprezentuje jedno písmeno a je uložen jako big endian.
- **Velikost podúseku** – obsahuje velikost tohoto podúseku od následujícího bajtu. Tedy velikost podúseku obsahující specifikaci formátu zvukových dat bez 8 bajtů. Pokud jsou data uložena ve formátu PCM, tak je tato délka vždy 16 bajtů.
- **Formát kódování** – obsahuje specifikaci formátu, podle kterého jsou uložena data. Pro formát *Pulse Code Modulation* (PCM) je přiřazena hodnota 1.
- **Počet kanálů** – obsahuje počet kanálu zvukového souboru.
- **Vzorkovací frekvence** – obsahuje vzorkovací frekvenci zvukového souboru. Respektive počet vzorků zvukových dat, které tvoří jednu sekundu zvuku.
- **Bajtová frekvence** – obsahuje hodnotu, která určuje počet bajtů za jednu sekundu zvuku. Tedy velikost vzorku násobená vzorkovací frekvencí.
- **Velikost vzorku** – obsahuje velikost jednoho vzorku v bajtech. Tedy velikost bloku násobená počtem kanálů.
- **Velikost bloku** – obsahuje velikost jednoho bloku v bajtech.



Obrázek 3.3: Struktura podúseku s formátem zvukových dat.

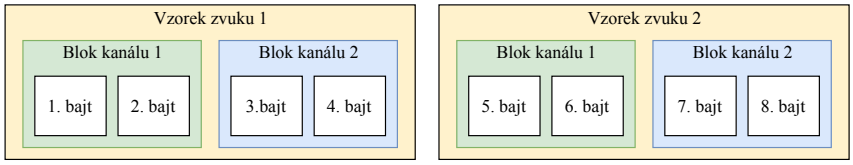
Struktura podúseku obsahující samotná zvuková data:

- **Identifikátor podúseku** – vždy bude obsahovat název „data“, každý bajt reprezentuje jedno písmeno a je uložen jako big endian.
- **Velikost podúseku** – obsahuje velikost podúseku od následujícího bajtu. Tedy jedná se o velikost samotných zvukových dat v bajtech.
- **Zvuková data** – následují samotná zvuková data. Způsob uložení dat ve formátu PCM je popsán v následujícím odstavci.



Obrázek 3.4: Struktura podúseku se zvukovými daty.

Ve formátu PCM jsou zvuková data rozložena do vzorků, které následují v souboru přímo za sebou. Každý vzorek je tvořen bloky pro každý kanál zvuku, tedy v jednokanálovém zvuku bude jeden vzorek obsahovat jeden blok. Velikost jednoho bloku kanálu je předem definována. Na obrázku 3.5 je uveden příklad rozložení vzorků pro soubor, který má délku bloku 16 bitů a obsahuje dva kanály. Počet vzorků, které tvoří jednu sekundu zvuku, je určen vzorkovací frekvencí.

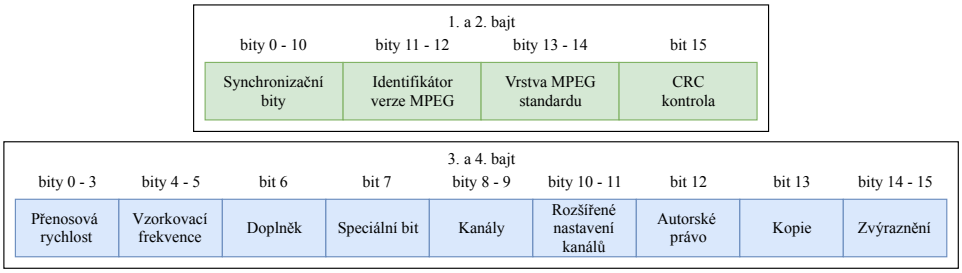


Obrázek 3.5: Ukázka rozložení vzorků v 16bitovém stereo souboru.

3.3.2 MP3

Zvukový formát MP3 vychází ze standardů *Moving Picture Experts Group* (dále jen MPEG). Standardy MPEG jsou vytvářeny stejnojmennou pracovní skupinou, která byla v roce 1988 založena organizacemi *International Organization for Standardization* (dále jen ISO) a *International Electrotechnical Commission* (dále jen IEC). Formát MP3 je také formálně nazýván jako MPEG-1 Audio Layer 3 nebo také MPEG-2 Audio Layer 3.

Podrobnější informace o struktuře formátu MP3 jsou popsány ve standardu *ISO/IEC 11172-3:1993* a *ISO/IEC 13818-3:1995*, které je možné zakoupit na oficiálních stránkách [3]. Pro tuto práci byly využity jako zdroj informací o struktuře souboru neplacené webové zdroje [2, 1] a experimentální testování s MP3 soubory.



Obrázek 3.6: Struktura hlavičky jednoho rámce.

Soubor ve formátu MP3 je tvořen třemi částmi. Na začátku souboru jsou uloženy tagy verze 2, následují rámce se zvukovými daty a po rámcích je soubor ukončen tagy verze 1. Každý rámec obsahuje hlavičku o velikosti 4 bajty, za kterou následují zvuková data. Jeden rámec reprezentuje 0,026 sekundy zvuku. Rámce jsou navzájem nezávislé části zvukového souboru a jejich hlavičky se mohou lišit. Rozložení jednotlivých částí hlavičky je ukázáno na obrázku 3.6. Struktura hlavičky jednoho rámce je následující:

- **Synchronizační bity** – všechny bity jsou nastaveny na hodnotu 1. Slouží jako ukazatel začátku rámce, avšak tato hodnota se může v souboru vyskytovat náhodně, proto je nutné zkontrolovat hodnoty zbytku hlavičky, například použitá vrstva a verze MPEG standardu by měla být u jednotlivých rámců stejná.
- **Identifikátor verze MPEG** – obsahuje identifikátor použité verze MPEG standardu. Ve většině případů se bude jednat o verzi 1, ale v některých případech se může jednat i o verzi 2. Význam jednotlivých bitových kódů je uveden v tabulce 3.1.

Kód	Význam kódu
00	MPEG verze 2.5
01	rezervované
10	MPEG verze 2
11	MPEG Verze 1

Tabulka 3.1: Význam sekvence bitů určujících verzi MPEG standardu.

- **Vrstva MPEG standardu** – určuje použitou vrstvu MPEG standardu. V MP3 formátu by zde měla být uložena hodnota určující 3. vrstvu, jelikož v této vrstvě je MP3 formát definován. Význam jednotlivých bitových kódů je uveden v tabulce 3.2.

Kód	Význam kódu
00	rezervované
01	3. vrstva
10	2. vrstva
11	1. vrstva

Tabulka 3.2: Význam sekvence bitů určujících vrstvu MPEG standardu.

- **CRC kontrola** – určuje, jestli soubor využívá kontrolu chyby *Cyclic Redundancy Check* (dále jen CRC). Ve většině souborů tato kontrola není využita.
- **Přenosová rychlost** – zde je uložena přenosová rychlost rámce. Hodnota je uložena v kilobitech za sekundu a význam jednotlivých bitových kódů je uveden v tabulce 3.3.

Kód	Význam kódu
0000	rezervované
0001	32 kbps
0010	40 kbps
0011	48 kbps
0100	56 kbps
0101	64 kbps
0110	80 kbps
0111	96 kbps

Kód	Význam kódu
1000	112 kbps
1001	128 kbps
1010	160 kbps
1011	192 kbps
1100	224 kbps
1101	256 kbps
1110	320 kbps
1111	rezervované

Tabulka 3.3: Význam sekvence bitů určujících přenosovou rychlost.

- **Vzorkovací frekvence** – obsahuje vzorkovací frekvenci v hertzech, tedy počet vzorků, na které je rozdělena 1 sekunda zvuku. Význam jednotlivých bitových kódů je uveden v tabulce 3.4.

Kód	Význam kódu
00	44 100 Hz
01	48 000 Hz
10	32 000 Hz
11	rezervované

Tabulka 3.4: Význam sekvence bitů určujících vzorkovací frekvenci.

- **Doplněk** – délka rámce nemusí vždy přesně pasovat do přenosové rychlosti, k tomu se využívá doplňující bajt, který se může přidat na konec rámce. Doplňující bajt může být přidán k několika rámcům, aby přesně platila přenosová rychlost. Pokud je nastaven na hodnotu 1, tak je na konec rámce přidán doplňující bajt, v opačném případě doplněn není.
- **Speciální bit** - může být využit pro různé účely v různých aplikacích.
- **Kanály** – obsahuje počet kanálů a jejich nastavení. Soubor může mít dva kanály, ale může s nimi pracovat jinak (Stereo a Joint Stereo). Význam jednotlivých kódů je uveden v tabulce 3.5.

Kód	Význam kódu
00	Stereo
01	Joint Stereo
10	Dual
11	Mono

Tabulka 3.5: Význam sekvence bitů určujících počet a nastavení kanálů.

- **Rozšířené nastavení kanálů** – tyto bity jsou nastaveny pouze, pokud je využito nastavení joint-stereo. Blíže specifikuje, jaký způsob joint-stereo je využit.
- **Autorské právo** – bit určující, jestli je soubor chráněn autorským právem. Pokud je chráněn, je nastaven bit na hodnotu 1.
- **Kopie** – tento bit určuje, jestli se jedná o originální soubor, nebo kopii originálního souboru. Pokud je nastaven na hodnotu 1, tak se jedná o originál.
- **Zvýraznění** – určuje, jestli soubor obsahuje zvýrazněné frekvence.

Struktura následujících zvukových dat je různá podle použité komprese. V rámci experimentování se soubory jsem význam jednotlivých bitů nezjistil. Pro práci s rámcem je nutné znát jeho délku. Pokud je l délka rámce, BR přenosová rychlost a SR vzorkovací frekvence, tak pro výpočet délky úseku platí následující vzorec:

$$l = int(144 * BR/SR)$$

Pokud je nastaven doplňující bit na hodnotu 1, tak je rámec delší o jeden bajt.

Kapitola 4

Metody digitální audio steganografie a jejich vlastnosti

Praktické využití metod v průběhu let ukázalo, v jaké oblasti mají jednotlivé metody své využití. Některé metody se v dnešní době považují za zastaralé a využívají se jen zřídka, nicméně každá metoda má své odůvodnitelné využití v určitých situacích. V této kapitole jsou popsány vlastnosti, podle kterých jsou jednotlivé metody hodnoceny, tímto popisem se zabývá sekce 4.1. Následně jsou v sekci 4.2 popsány jednotlivé metody, které jsou běžně využívány v oblasti digitální audio steganografie.

4.1 Vlastnosti metod

Pro evaluaci steganografických metod je nutné specifikovat jejich vlastnosti, podle kterých můžeme metody hodnotit. Každá metoda má odlišnou kvalitu konkrétní vlastnosti. Teoreticky tedy nelze určit nejlepší, respektive nejhorší metodu, jelikož každá metoda má své využití v konkrétním případě. V praxi je při každém využití steganografie nutné určit, která vlastnost je pro nás v tu chvíli nejvíce relevantní a podle toho zvolit konkrétní metodu.

4.1.1 Slyšitelnost změn

Pro bezpečnost steganografie je velice důležitý prvek nenápadnosti a neviditelnosti přenášené informace. Pokud je na první pohled zřejmé, že stego soubor se liší od krycího souboru, je bezpečnost informace významně ohrožena a steganografická metoda selhala. V případě digitální audio steganografie je možné tyto rozdíly slyšet. Evaluace souborů může probíhat podle rodiny standardů *Perceptual Evaluation of Speech Quality* (PESQ) určených pro automatické porovnávání kvality zvuku [6] nebo pomocí jiných postupů.

4.1.2 Rychlost vkládání

Další velice důležitou vlastností je rychlost kódování informací do krycího souboru. Tato vlastnost se dá měřit jako počet vložených bitů informace do krycího souboru za jednu sekundu. Problém této vlastnosti nastává v přesnosti měření. Ta je totiž ovlivněna několika dalšími jevy jako je způsob implementace nebo využitý programovací jazyk. Provedení metody by mělo být sice teoreticky stejné, například *Metoda nejméně významného bitu* bude kódovat poslední bit krycího souboru, ale samotná implementace toho, jak je kódování provedeno, se může značně lišit, a tím zkreslovat rychlost zápisu informace. Jak již

bylo zmíněno, další zkreslení může nastat podle zvoleného programovacího jazyka, jelikož samotný jazyk a jeho funkce jsou různě rychlé.

4.1.3 Kapacita

Při využití steganografických metod je pro nás také zásadní maximální velikost informace, kterou můžeme do krycího souboru uložit, jednoduššeji řečeno kapacita. Každá využívaná metoda má jinou kapacitu. Například při využití metody, která kóduje informaci na nejnižší bit krycího souboru, dosahujeme největší kapacity z využívaných metod [11]. Kapacita této metody se dá však ještě zvýšit využitím určitých postupů [10]. Kapacita je ale nejen ovlivněna použitou metodou, ale také použitým krycím souborem. Například při využití metody *Ukrývání v úsecích ticha* je kapacita ovlivněna počtem úseků ticha v krycím souboru.

4.1.4 Odolnost

Při volbě steganografické metody hraje také velkou roli její odolnost. Odolností chápeme schopnost stego souboru uchovat vloženou informaci i po provedení určitých změn tohoto souboru. Pro ověření této vlastnosti můžeme na soubor aplikovat několik postupů a pokusit se znovu získat informaci z pozměněného souboru. Mezi změny souboru můžeme řadit:

- přidání šumu do zvuku,
- filtrování vybraných částí spektra,
- kvantování zvuku na jiný počet bitů,
- ztrátová komprese.

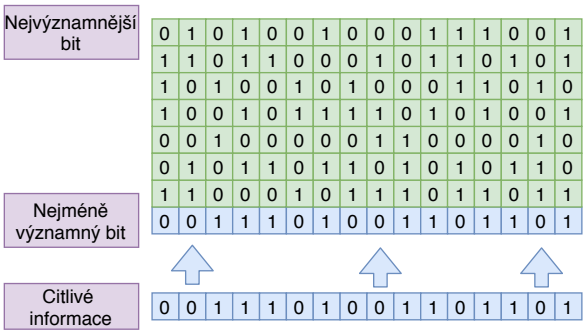
Odolnost je úzce spjata s kapacitou. Pokud máme metodu, jejíž kapacita je zásadně menší než kapacita jiné, tak při změně stejné části souboru dojde k menší ztrátě informací, jelikož jich tam je zkrátka uloženo méně. Pro zvýšení odolnosti metody můžeme informace například opakovat, to má ale za následek úměrné snížení kapacity krycího souboru [12].

4.2 Metody digitální audio steganografie

Metody digitální steganografie mohou být založeny na různých principech. Některé metody je možné po menší úpravě použít na různé druhy krycích souborů. Například metodu, která využívá kódování na nejméně významný bit, je možné využívat i v digitální obrazové steganografii.

4.2.1 Kódování nejméně významného bitu

Jedná se o nejzákladnější metodu využívanou i v obrazové steganografii. Jak je z názvu patrné, metoda je založena na kódování informace na nejméně významný bit krycího souboru. V některých případech metoda nevyužívá pouze jeden nejméně významný bit, ale případně dva bity nebo i více bitů. To zajišťuje větší kapacitu, ale zároveň výrazně snižuje kvalitu souboru, tedy rozdíl mezi stego souborem a krycím souborem je slyšitelný.



Obrázek 4.1: Příklad vložení informace na nejnižší bit do souboru se vzorkem o délce 8b.

Největší výhodou této metody je poměrně velká kapacita, u vzorkovací frekvence 48 kHz je kapacita krycího souboru 48 kbps. Při využití dvou nejnižších bitů je kapacita dvojnásobná. Nevýhodou této metody je malá odolnost. Při ztrátové kompresi, přidání hluku, filtrování a dalších manipulacích je většinou informace nevratně zničena. V současnosti již došlo ke vzniku různých postupů, které částečně odstraňují zmíněné nevýhody, například [9] a [10].

4.2.2 Metoda paritního bitu

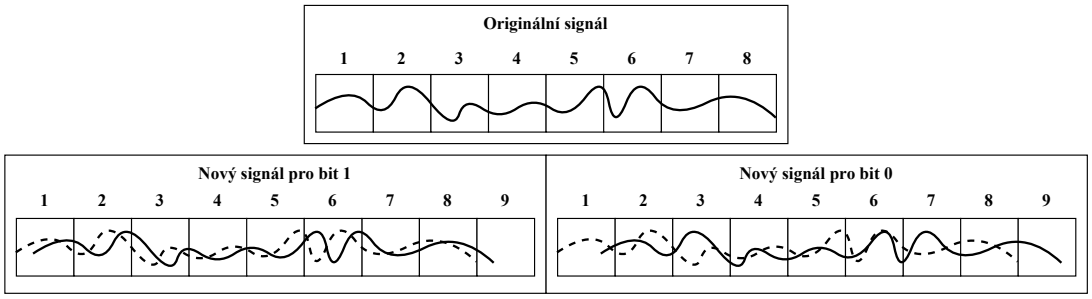
Metoda využívá kódování paritního bitu. Pro využití této metody je zvuk z krycího souboru rozdělen na námi zvolené úseky (dále regiony) o stejné délce. Pro každý tento region je spočítán paritní bit a je porovnán s bitem vkládané informace [13]. Pokud se bity shodují, tak se nic nemění, pokud je ale paritní bit opačný než vkládaný bit, tak je změněn jeden z bitů v daném regionu. Nejlépe je změněn jeden z nejnižších bitů jednoho vzorku daného regionu.

Tato metoda se dá považovat za rozšíření metody *Kódování nejméně významného bitu*, protože je do jisté míry odolnější, avšak využívá podobného principu, a to kódování nejnižšího bitu vzorku. Další výhodou oproti předchozí metodě je větší bezpečnost, jelikož existuje mnoho různých možností, jak zvolit regiony [13]. Pro přečtení zprávy je nutné znát délku úseků, na které je zvuk rozdělen. Bez znalosti této délky není útočník schopný určit paritní bity a následně přečíst ukryté informace.

4.2.3 Ukrývání ozvěny

Další ze starších metod je metoda využívající ozvěny. Do zvuku krycího souboru je přidána ozvěna. Originální signál je posunut o hodnotu α, tím vznikne nový signál reprezentující kódování bitu 1. Pro kódování bitu 0 je vytvořen také nový signál, který je posunut o hodnotu β [16]. Pro dekódování zprávy je nutné znát hodnoty obou posunů. Pro možnost vložení více bitů je zvuk rozdělen na úseky. V každém úseku následně přičteme k originálnímu signálu nově vytvořený signál s nízkou amplitudou. Který nově vytvořený signál přičteme, je určeno podle kódovaného bitu. Na obrázku 4.2 jsou ukázány příklady originálního a nově vytvořených signálů.

Tuto metodu nelze aplikovat na všechny krycí soubory. Konkrétně jsou problematické zvukové nahrávky, které již obsahují nějakou ozvěnu nebo obsahují delší úseky ticha, například u některých nahrávek mluveného slova. Celkově se jedná o poměrně zastaralou metodu s nízkou bezpečností, která již v dnešní době není nijak zásadně rozšířena [12].



Obrázek 4.2: Příklad originálního a nově vytvořených signálů.

4.2.4 Ukrývání v úsecích ticha

Předchozí metoda *Ukrývání ozvěny* není vhodná pro mluvené slovo, pro tyto případy můžeme využít následující metodu. Metoda využívá delších úseků ticha v krycím souboru. Pro korektní využití je nutné určit nejmenší použitelnou délku úseku ticha. Tato délka je různá na základě využitého krycího souboru. Nicméně se dá říci, že pokud využíváme nahrávku mluveného slova, měla by být minimální délka použitého úseku větší než je délka nejdelšího úseku ticha mezi slovy. Tím zajistíme, že metoda nebude zasahovat do úseků ticha mezi slovy, jelikož by došlo k slyšitelnému narušení plynulosti věty.

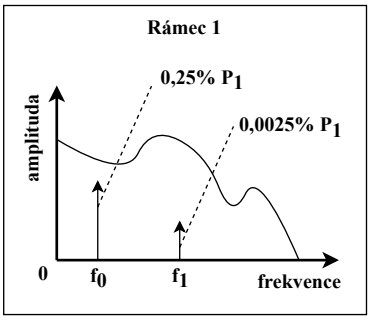
Následuje spočítání počtu vzorků, které tvoří určitý úsek ticha a počet bitů potřebný pro kódování jednoho znaku zprávy (dále jen jako PBK). Následně je spočítána konstanta x tak, aby splňovala dále zmíněné podmínky. Při odečtení konstanty od počtu vzorků nám musí vyjít číslo, které když vydělíme 2^{PBK}, tak zbytek po dělení je roven požadované hodnotě kódovaného znaku zprávy v decimální hodnotě [17]. Následné získání hodnoty ze stego souboru probíhá podle následujícího vzorce:

$$znak = mod(novaDelkaIntervalu, 2^{PBK})$$

Jako příklad si můžeme vzít písmeno „A“, toto písmeno je v ASCII tabulce reprezentováno decimální hodnotou 65. Jeden znak z ASCII tabulky je reprezentován 7 bity, tedy platí $PBK = 7$. Pokud nalezneme úsek ticha o délce 242 vzorků, tak úsek zkrátíme na délku 193 vzorků a poté platí $mod(193, 128) = 65$. Pokud by měla být výsledná délka úseku menší než minimální délka hledaných úseků, tak tento úsek nebude při dekódování nalezen, proto je nutné kódovanou hodnotu znovu zakódovat do dalšího úseku.

4.2.5 Metoda přidaného tónu

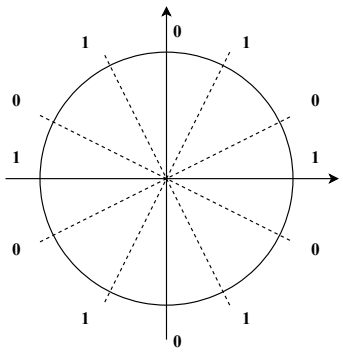
Tato metoda využívá přidání slabých tónů do původního signálu [12]. Pro kódování zvolíme dvě frekvence f_0 a f_1, u kterých budeme měnit amplitudu [15]. Amplituda bude rozložena mezi frekvence podle námi zvoleného procentuálního zastoupení, například 0,25 % a 0,0025 %. Originální signál rozdělíme do rámců a pro každý rámec spočítáme celkový výkon p_i. Novou amplitudu frekvencí spočítáme pomocí celkového výkonu rámce. Tím zajistíme, že nová amplituda bude podobná zbytku rámce a nebude tím vznikat výrazná změna v signálu. Nově spočítaná amplituda signálu je rozdělena mezi frekvence f_0 a f_1 podle námi zvoleného procentuálního zastoupení. Pokud chceme zakódovat hodnotu 0, tak musí po změně signálu platit $\frac{p_1}{p_{f_0}} > \frac{p_1}{p_{f_1}}$ a naopak pro kódování hodnoty 1. Příklad kódování je viditelný na obrázku 4.3. Výhodou této metody je vysoká odolnost [15].



Obrázek 4.3: Příklad kódování hodnoty 0.

4.2.6 Metoda kódování fáze

Další metoda využívá změn ve fázi vybraných frekvencí. Tato metoda funguje na principu změn fáze vybraných frekvencí podle předem definovaných hodnot, které musí být známé pro obě strany, tedy pro stranu, která zprávu kóduje, i pro stranu, která zprávu dekóduje. Pro kódování je zjištěna původní fáze, tato fáze je nahrazena nejbližší definovanou hodnotou, podle kódovaného bitu [14]. Příklad hodnot je ukázán na obrázku 4.4. Výhodou této metody je vysoká odolnost vůči narušení signálu [12].



Obrázek 4.4: Rozložení hodnot pro změnu fáze.

4.2.7 Shrnutí dalších využívaných metod

V práci [12] jsou zmíněny další metody v této oblasti. Nicméně jejich dokumentace mi přišla nedostatečná pro následný přesný popis jejich fungování. Z tohoto důvodu jsem je v této práci podrobněji nerozebíral. Steganografické metody však mohou fungovat na různých dalších principech, které nejsou v této práci zmiňovány.

Jednou ze zajímavých oblastí steganografických metod je kódování do příznaků zvukového souboru, jako je například jméno autora. Některé zvukové formáty také obsahují rezervované bity. Tyto bity můžeme využít pro kódování informace.

Kapitola 5

Implementace

Tato kapitola blíže popisuje implementaci vybraných metod. Popisuje vybraný programovací jazyk pro implementaci aplikace a použité knihovny tohoto jazyka. Implementované metody, s drobnými úpravami, vycházejí z metod popsaných v podkapitole 4.2, výjimkou je metoda 5.4.3, která byla mnou navrhnuta. Popis každé metody obsahuje pseudokód, který názorně ukazuje fungování metody.

5.1 Použitý jazyk a knihovny

Pro implementaci aplikace byl použit programovací jazyk *Python*. Programovací jazyk nebyl v zadání práce specifikován a ze zadání nevycházely žádné speciální požadavky, které by volbu programovacího jazyka ovlivňovaly. Při výběru programovacího jazyka jsem se rozhodoval mezi jazykem Python a C++. Programovací jazyk Python byl vybrán na základě dřívější zkušenosti s tímto jazykem. Ještě jednou neopomenutelnou možností bylo programovací prostředí MATLAB. Při vytváření mé práce jsem se rozhodl, že chci, aby výsledkem byla jednoduše použitelná aplikace, a proto MATLAB není vhodnou volbou. Aplikace byla implementována ve verzi 3.6.4 jazyka Python a tato verze je také jedinou otestovanou verzí, tedy funkčnost v jiných verzích není zaručena. Python 3 není zpětně kompatibilní s Python 2.

Aplikace využívá pouze standardní knihovny jazyka Python. Pro MP3 soubory nebyla nalezena vhodná knihovna, která by dostatečně usnadňovala práci s tímto formátem. Pro soubory formátu WAVE byla původně použita standardní knihovna WAVE. Tato knihovna byla v pokročilé části implementace nahrazena mnou implementovanými metodami pro práci s formátem WAVE. Důvodů pro tuto změnu bylo několik. Jedním z důvodů bylo zrychlení aplikace, dalším důvodem bylo omezené množství funkcí této knihovny, tedy stejně by bylo nutné implementovat některé funkce samostatně. Posledním důvodem byl výskyt nespecifikovaných případů, při kterých knihovna nebyla funkční. Důvody pro nefunkčnost knihovny v těchto případech, respektive konkrétní případy, ve kterých knihovna nebyla funkční, jsem nezjistil. Dokumentace této knihovny také nebyla příliš detailní.

5.2 Struktura aplikace

Aplikace je rozdělena do několika modulů, rozdělení je ukázáno na obrázku 5.1. Význam modulů je následující:

- **main.py** – obsahuje zpracování vstupních argumentů aplikace a následné volání požadované funkce v jiných modulech. Dále tento modul zpracovává cestu ke vstupním souborům pro další použití ve funkcích. Pokud je aplikace spuštěna pro kódování zprávy do zvukového souboru, tak je v tomto modulu zpráva. Zpráva je zpracována tím způsobem, že je každý znak převeden do binární ASCII hodnoty a tyto binární hodnoty jsou za sebe uloženy do textového pole.

- **wave_lbe.py** – obsahuje funkce pro zpracování metody, která využívá kódování nejméně významného bitu do souboru ve formátu WAVE, bližší implementace metody je popsána v sekci 5.4.1.

- **wave_parity.py** – obsahuje funkce pro zpracování metody, která využívá kódování paritního bitu do souboru ve formátu WAVE, bližší implementace metody je popsána v sekci 5.4.2.

- **wave_zero.py** – obsahuje funkce pro zpracování metody, která využívá kódování nulových úseků do souboru ve formátu WAVE, bližší implementace metody je popsána v sekci 5.4.3.

- **wave_silence.py** – obsahuje funkce pro zpracování metody, která využívá kódování úseků ticha do souboru ve formátu WAVE, bližší implementace metody je popsána v sekci 5.4.4.

- **wave_info.py** – obsahuje funkce pro zpracování informací ze souborů ve formátu WAVE.

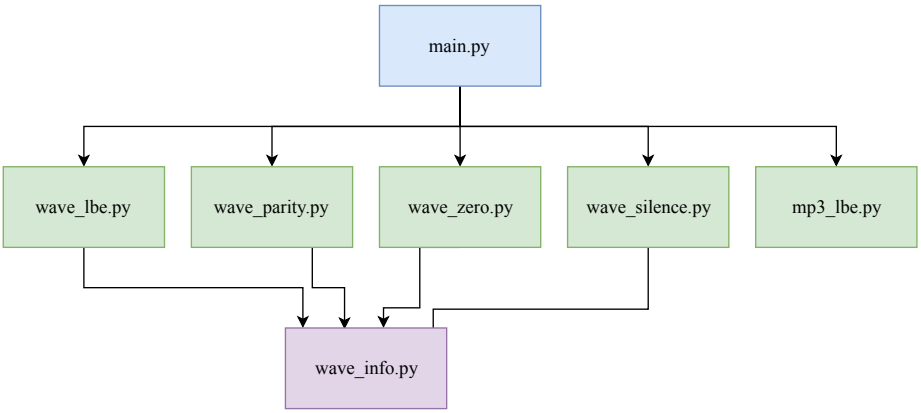
- **mp3_lbe.py** – obsahuje funkce pro zpracování metody, která využívá kódování posledního bitu do souboru ve formátu MP3. Tento modul zároveň obsahuje funkce pro zpracování souborů ve formátu MP3.

Moduly obsahující implementace jednotlivých metod vždy obsahují 3 základní funkce a případně další pomocné funkce. Tyto funkce jsou následující:

- **info** – funkce sloužící pro zjištění informací o souboru, například zjišťuje velikost zvukových dat, počet kanálů, vzorkovací frekvenci a další vlastnosti. Funkce zároveň počítá maximální počet znaků, které je možné vybranou metodou do souboru zakódovat.

- **embed** – tato funkce slouží pro zakódování zprávy do souboru s využitím vybrané metody.

- **disembed** – tato funkce slouží pro dekódování uložené zprávy ze souboru pomocí vybrané metody.



Obrázek 5.1: Ukázka rozdělení souborů aplikace.

5.3 Vstupy a výstupy aplikace

Vstupy aplikace jsou různé podle požadované akce, respektive pro kódování je nutné zadat jeden argument navíc oproti ostatním akcím. Argumenty aplikace jsou zpracovány pomocí modulu *argparse*. Požadované argumenty jsou následující:

- **Formát** – tento argument je povinný a specifikuje v jakém formátu je soubor, s kterým bude aplikace pracovat.
- **Metoda** – tento argument je povinný a určuje metodu, která je použita. Tedy určuje, z kterého modulu je zavolána příslušná funkce.
- **Akce** – tento argument je povinný a specifikuje požadovanou akci. Tyto akce se shodují se třemi základními funkcemi, které jsou popsané v kapitole 5.2.
- **Vstupní soubor** – tento argument je povinný a obsahuje cestu ke vstupnímu souboru. U akce kódování obsahuje cestu ke krycímu souboru a u akce dekódování obsahuje cestu k souboru, z kterého chceme zprávu dekódovat.
- **Zpráva** – tento argument je požadován, pokud je zvolena akce kódování, u ostatních akcí tento argument nelze zadat. Obsahuje cestu k textovému souboru se zprávou, kterou chceme kódovat.

Výstupy všech akcí jsou vypsány na standardní výstup. Při kódování zprávy do souboru je zároveň vytvořen nový upravený soubor, tedy aplikace nepřepisuje vstupní soubor. Nově vytvořený soubor je uložen do složky *output_files*, která je předem vytvořena ve stejné složce, jako se nachází aplikace. Výstupní soubor má stejný název jako vstupní soubor.

5.4 Implementace metod pro formát WAVE

Tato podkapitola obsahuje popis implementace tří základních funkcí, které byly popsány v podkapitole 5.2. Funkce z modulu pro zjišťování informací o souborech formátu WAVE vychází z popisu formátu v kapitole 3.3.1, jelikož tyto funkce nejsou nijak komplexní, tak zde jejich implementace popsána není.

5.4.1 Metoda nejnižšího bitu

Tato metoda vychází z popisu metody 4.2.1. Metoda byla upravena tak, aby bylo maskováno větší množství bitů v jednom bajtu. Množství maskovaných bitů vychází z velikosti jednoho bloku zvukových dat. Pro použití této metody je nutné zjistit počet kanálů a velikost jednoho bloku, tedy zjistit strukturu jednoho vzorku, tato struktura je blíže popsána v sekci 3.3.1. V každém vzorku je následně změněn příslušný počet posledních bitů u příslušných bajtů. Respektive u každého bloku ve vzorku je určen poslední bajt tohoto bloku a v tomto bajtu je změněno několik posledních bitů.

Počet bitů, které jsou změněny, byl určen na základě testování. Ve výsledné implementaci je tento počet určen jako čtvrtina celkového počtu bitů v jednom bloku. Tedy pro 16bitový soubor ve formátu wave jsou změněny poslední 4 bity bloku. Touto změnou bylo docíleno dvojnásobné kapacity metody při zachování neslyšitelnosti změn.

Algoritmus 1: Algoritmus kódování zprávy na nejméně významné bity.

Input: BA je obsah krycího souboru v bajtovém poli
PB je počet měněných bitů
SB je šířka bloku
MSG je pole se zprávou
Output: upravené bajtové pole BA

```
1  J = 0 // index pole s kódovanou zprávou
2  I = 0 // index pole BA
3  K = 0 // index bitu v upravovaném bajtu, indexováno od posledního bitu
4  while J < délka zprávy do
5      for K v rozsahu 0 až PB do
6          v prvku BA[I] změň bit na indexu K podle hodnoty prvku MSG[J]
7          J = J + 1
8      end for
9      I = I + SB
10 end while
```

5.4.2 Metoda paritního bitu

Implementace této metody vychází z popisu metody 4.2.2. Pro použití této metody je nutné definovat velikost regionu, pro který se zjišťuje paritní bit. Tento region je v implementaci určen jako velikost jednoho vzorku zvuku násobená definovanou konstantou. Z výsledků testování byla velikost regionu ponechána na velikosti jednoho vzorku, respektive konstanta, kterou se vzorek násobí, má hodnotu 1.

Pro změnu paritního bitu na příslušnou hodnotu podle zprávy je nutné změnit jeden bit tohoto regionu. Při kódování je nutné zachovat neslyšitelnost změny, tedy měněný bit by neměl zásadně ovlivňovat kvalitu zvuku. Pro tuto změnu je využit poslední bit posledního bajtu v regionu. Z popisu struktury WAVE formátu v kapitole 3.3.1 je zřejmé, že poslední bit posledního bajtu v regionu bude vždy jedním z nejméně významných bitů, bez ohledu na

velikost bloku a počet kanálů zvukového souboru. Pro sudou paritu je kódována hodnota 0 a pro lichou paritu hodnota 1.

Algoritmus 2: Algoritmus kódování zprávy pomocí paritního bitu.

Input: BA je obsah krycího souboru v bajtovém poli
PR je parita regionu
SR je šířka regionu
MSG je pole se zprávou
Output: upravené bajtové pole BA

```
1  C = 0 // pomocná proměnná
2  J = 0 // index pole s kódovanou zprávou
3  for projdi všechny prvky BA do
4  |  vezmi z BA další bajt a přičti jeho paritu do PR C = C + 1 if C == SR then
5  |  |  if MSG[J] == 0 a hodnota SR je lichá then
6  |  |  |  změň poslední bit posledního bajtu regionu
7  |  |  else if MSG[J] == 1 a hodnota SR je sudá then
8  |  |  |  změň poslední bit posledního bajtu regionu
9  |  |  PR = C = 1 J = J + 1
10 end for
```

5.4.3 Metoda nulových úseků

Tato metoda byla mnou navržena a implementována. Tato metoda byla navržena primárně pro ukrývání do nahrávek, které vznikly převodem textu do mluveného slova. V těchto souborech se nachází dlouhé úseky bajtů s nulovou hodnotou. Na základě tohoto faktu jsem metodu pojmenoval. Původně měla tato metoda pracovat na základě vyhledávání úseků bajtů se stejnou hodnotou, nicméně byla následně rozšířena a vyhledává úseky bajtů, které mají hodnoty v nějakém definovaném rozpětí. Tato modifikace vznikla z toho důvodu, že ostatní typy zvukových nahrávek většinou neobsahují mnoho úseků, ve kterých by měly všechny bajty nulovou nebo obecně stejnou hodnotu.

Po modifikaci metoda vyhledává úseky stejným způsobem, jako jsou vyhledávány úseky ticha u metody 4.2.4. Rozdílem je délka vyhledávaných úseků, v této metodě je minimální délka úseku nastavena na hodnotu 4. Hodnota, udávající maximální rozdíl mezi maximálním a minimální prvkem v úseku, se výrazně neliší. Dalším výrazným rozdílem je, že v této metodě jsou do souboru kódovány pouze hodnoty 0 a 1, tedy zpráva je kódována pouze v binární podobě.

Délka nalezeného úseku je následně upravena podle kódované hodnoty. Sudá délka bloku reprezentuje kódovanou hodnotu 0 a lichá délka úseku hodnotu 1. Pokud by výsledný úsek byl kratší než minimální délka vyhledávaných úseků, tak musí být kódovaná hodnota zakódována znovu do dalšího úseku, jelikož tento úsek nebude při dekódování nalezen. Al-

goritmus této metody 3 je modifikovaný algoritmus 4 používaný při ukrývání do úseků ticha.

Algoritmus 3: Algoritmus kódování zprávy do nulových úseků.

Input: obsah krycího souboru v bajtovém poli BA
L je minimální délka úseku
D je maximální rozdíl prvků v úseku
Output: upravené bajtové pole BA

```
1  MAX = MIN = 0 // hodnoty maximálního a minimálního prvku v úseku
2  DU = 0 // délka aktuálního úseku
3  for projdi všechny prvky BA do
4      X = hodnota následujícího prvku BA
5      if X > MAX a X − MIN > D then
6          if DU >= L then
7              if je nutné měnit délku úseku then
8                  if DU == L then
9                      změň délku úseku tak, aby byl úsek kratší než L
10                 else
11                     podle kódované hodnoty změň délku úseku
12                 end if
13             začni počítat nový úsek, MAX = MIN = X, DU = 1
14         else
15             MAX = X a najdi nové MIN, aby platilo MAX − MIN < D, spočítej
                nové DU
16             pokud neexistuje, začni počítat nový úsek, MAX = MIN = X, DU = 1
17         end if
18     else if X < MIN a MAX − X > D then
19         if DU >= L then
20             if je nutné měnit délku úseku then
21                 if DU == L then
22                     změň délku úseku tak, aby byl úsek kratší než L
23                 else
24                     podle kódované hodnoty změň délku úseku
25                 end if
26             začni počítat nový úsek, MAX = MIN = X, DU = 1
27         else
28             MIN = X a najdi nové MAX, aby platilo MAX − MIN < D, spočítej
                nové DU
29             pokud neexistuje, začni počítat nový úsek, MAX = MIN = X, DU = 1
30         end if
31     else
32         pokud je X nové MAX nebo MIN, ulož ho
33         DU = DU + 1
34     end if
35 end for
```

5.4.4 Metoda úseků ticha

Tato metoda vychází z popisu metody 4.2.4. Úseky ticha jsou hledány v souboru jako sekvence vzorků, které mají stejnou nebo blízkou hodnotu. Tuto hodnotu si předem definujeme, z výsledků testování 4.2.4 jsem v aplikaci nastavil hodnotu 15. Jeden znak je v binární ASCII hodnotě reprezentován 7 bity, tedy z popisu metody vychází, že hledáme úseky ticha dlouhé minimálně 128 bajtů, a v úseku musí platit, že rozdíl mezi maximální a minimální hodnotou v úseku musí být menší 15.

Algoritmus 4: Algoritmus kódování zprávy do úseků ticha.

Input: obsah krycího souboru v bajtovém poli BA
L je minimální délka úseku
D je maximální rozdíl prvků v úseku
Output: upravené bajtové pole BA

```
1  MAX = MIN = 0 // hodnoty maximálního a minimálního prvku v úseku
2  DU = 0 // délka aktuálního úseku
3  NDU = 0 // nová délka úseku, která je určena podle kódované hodnoty
4  for projdi všechny prvky BA do
5      X = hodnota následujícího prvku BA
6      if X > MAX a X − MIN > D then
7          if DU >= L then
8              podle kódovaného znaku spočítej NDU
9              změň prvek v úseku tak, aby měl úsek délku NDU
10             pokud bylo NDU < L, tak neposouvej ukazatel v poli se zprávou
11             začni počítat nový úsek, MAX = MIN = X, DU = 1
12         else
13             MAX = X a najdi nové MIN, aby platilo MAX − MIN < D, spočítej
                nové DU
14             pokud neexistuje, začni počítat nový úsek, MAX = MIN = X, DU = 1
15         end if
16     else if X < MIN a MAX − X > D then
17         if DU >= L then
18             podle kódovaného znaku spočítej NDU
19             změň prvek v úseku tak, aby měl úsek délku NDU
20             pokud bylo NDU < L, tak neposouvej ukazatel v poli se zprávou
21             začni počítat nový úsek, MAX = MIN = X, DU = 1
22         else
23             MIN = X a najdi nové MAX, aby platilo MAX − MIN < D, spočítej
                nové DU
24             pokud neexistuje, začni počítat nový úsek, MAX = MIN = X, DU = 1
25         end if
26     else
27         pokud je X nové MAX nebo MIN ulož ho
28         DU = DU + 1
29     end if
30 end for
```

Jelikož jeden soubor může být tvořen několika miliony bajtů, jedná se o poměrně výpočetně náročnou operaci. Z tohoto důvodu jsem experimentoval s různými algoritmy, které

by byly schopné tuto operaci efektivně provést. Ze tří testovaných algoritmů jsem vybral výsledný algoritmus 4, který byl oproti ostatním až desetkrát rychlejší, jeho rychlost kódování je popsána u testování 6.3.4.

Výsledný algoritmus má však i svá úskalí. V současné podobě prochází postupně všechny bajty se zvukovými daty, avšak z popisu formátu WAVE 3.3.1 je zřejmé, že vzorek je tvořen jedním bajtem pouze u 8bitových WAVE souborů. Tedy skutečné úseky ticha tento algoritmus najde jen právě u 8bitových WAVE souborů a nebo u souborů, kde mají všechny bajty vzorku podobnou hodnotu, respektive splňují výše popsanou podmínku. Tyto úseky ticha jsou například u nahrávek, které vznikly převodem textu na mluvené slovo. Metodu je možné v budoucnu rozšířit, aby podporovala všechny druhy WAVE souborů, avšak u použitého algoritmu to velice zvyšuje komplexnost a jeho výpočetní náročnost.

5.5 Implementace metod pro formát MP3

Tato podkapitola se zabývá implementací jediné metody pro formát MP3, se kterou bylo v rámci této práce experimentováno. Během experimentování s touto metodou bylo zřejmé, že nebude reálně využitelná, jelikož změny v souboru zvuk naprosto zničily. Z tohoto důvodu nebyla ani implementována funkce pro dekódování zrávy.

5.5.1 Kódování nejméně významného bitu

Princip této metody vychází z popisu metody 4.2.1. Metoda byla původně implementována tak, že měnila poslední bit každého bajtu zvukových dat. Z důvodu naprostého zničení zvuku byla následně metoda modifikována tak, aby měnila poslední bit jen některých bajtů. Z důvodu nevyužitelnosti metody, nepovažuji za nutné se blíže zabývat její implementací.

Kapitola 6

Testování

Tato kapitola se zabývá testováním implementovaných metod. Nachází se zde specifikace testovaných souborů 6.1, na jak velkém množství a na jakých typech zvukových souborů testování probíhalo. Další sekce 6.2 popisuje, jak probíhalo samotné testování, respektive jakým způsobem byly testovány jednotlivé vlastnosti implementovaných metod. V poslední sekci 6.3 jsou popsány výsledky testování implementovaných metod.

6.1 Testované soubory

Pro testování implementovaných metod byl vytvořen balíček souborů ve formátu WAVE. Tento balíček je složen z 50 různých souborů, z toho 25 souborů jsou hudební nahrávky a 25 souborů jsou nahrávky mluveného slova. Hudební nahrávky jsou složeny z částí písniček různých stylů, některé nahrávky obsahují zpěv. Nahrávky mluveného slova obsahují záznamy přednášky, nahrávky čtení textu a nahrávky vytvořené převodem textu na mluvené slovo. Text byl převeden na mluvené slovo pomocí programu *Balabolka* s různým nastavením. Všechny nahrávky mluveného slova jsou v anglickém jazyce.

Délka nahrávek se ve většině případů pohybuje mezi 10 sekundami a 2 minutami. Většinou se jedná o 16bitové soubory se vzorkovací frekvencí 44,1 kHz, jelikož tyto soubory jsou nejčastěji využívány, ale v menším zastoupení jsou zde i soubory s jinou délkou bloku a různou vzorkovací frekvencí. Soubory byly staženy z různých webových stránek, které poskytují volně šířitelné soubory ve formátu WAVE.

6.2 Postup testování

U každé metody jsou testovány vlastnosti této metody. Popis jednotlivých vlastností, které jsou testovány, je v sekci 4.1. Do testovaných souborů byla všemi metodami zakódována zpráva o délce 10 milionů znaků. Pro testování byla každá metoda upravena tak, aby kódování nebylo přerušeno, pokud je zpráva delší, než je kapacita souboru. Díky této úpravě a velké délce zprávy bylo využito maximální kapacity souborů.

Při experimentování s implementovanými metodami jsem určil dvě různá nastavení pro každou metodu. Tato nastavení byla určena většinou tak, aby byla zachována největší možná kapacita metody, ale zároveň byly změny co nejméně slyšitelné. Do každého testovaného souboru byla následně pomocí každé metody dvakrát zakódována zpráva, jednou pro každé zvolené nastavení. Jedním z cílů testování bylo určit, které nastavení bude vhodné v implementovaných metodách zachovat pro běžné využívání aplikace.

6.2.1 Slyšitelnost změn

Při hodnocení slyšitelnosti změn v souboru je možné využít různých softwarů pro evaluaci kvality zvukového souboru, nicméně hlavní je, aby změny nebyly slyšitelné při běžném poslechu. Pro testování slyšitelnosti změn v souborech jsem oslovil skupinu 10 lidí. Protože citlivost sluchového ústrojí je u každého člověka trochu jiná, tak byla skupina věkově i genderově rozmanitá. Původním cílem bylo mít skupinu hodnotitelů větší, nicméně časová náročnost testování byla poměrně vysoká, tedy najít více ochotných lidí bylo těžké. Každý člověk hodnotil slyšitelnost změn po poslechu nahrávek. Pro každý soubor u každé metody vznikla pro testování trojice souborů, která byla složena z původního nezměněného souboru a dvou souborů se zakódovanou zprávou s různým nastavením.

Každý hodnotitel měl za úkol poslechnout si vždy trojici souborů a následně jim udělit hodnocení od 1 do 3, kde hodnota 1 je nejlepší. Každému hodnotiteli se před hodnocením pustila trojice ukázkových souborů. V této trojici byly změny dostatečně slyšitelné, díky tomu měli hodnotitelé lepší představu, jak se změny v souboru projeví. Pokud hodnotitelům připadaly nějaké nahrávky shodné, tak jim bylo uděleno stejné hodnocení. Ideálním výsledkem pro metodu je možnost, kdy nelze změny rozeznat, tedy původní soubor i změněný soubor dostaly hodnocení 1. Výsledné hodnocení slyšitelnosti změn je uvedeno v tabulkách 6.1, 6.4, 6.7 a 6.10.

6.2.2 Rychlost kódování

Pro testování této vlasnosti je do implementace aplikace přidán výpis doby běhu programu a množství zakódovaných znaků. Při kódování do souborů byly tyto hodnoty zaznamenány a následně vyhodnoceny. Výsledné hodnoty jsou spočítány jako počet zakódovaných znaků za jednu sekundu doby běhu programu. Rychlost kódování metody je silně ovlivněna konkrétní implementací celé aplikace. Například u kódované zprávy, která měla délku jeden milion znaků, byla doba zpracování zprávy ze vstupního souboru do využívaného řetězce nezanedbatelná. Doba zpracování zprávy je v měřené době běhu programu zahrnuta, jelikož je součástí implementované aplikace. Rychlosti kódování jednotlivých nastavení metod jsou uvedeny v tabulkách 6.2, 6.5, 6.8 a 6.11.

6.2.3 Kapacita

Pro testování kapacity souboru, byl do implementace přidán výpis počtu zakódovaných hodnot z řetězce zprávy. Při kódování do souborů byl tento výpis zaznamenán a následně vyhodnocen u každé metody. Výsledné hodnoty jsou počítány jako počet znaků zakódovaných na jeden bajt velikosti souboru. Velikost souborů může být větší, než je reálný počet zvukových dat, jelikož soubor může obsahovat přidané informace, jako je například jméno autora. To může do jisté míry ovlivňovat výpočet kapacity, nicméně u WAVE souborů se většinou jedná o několik stovek bajtů. Nejmenší testovaný soubor měl velikost okolo 600 000 bajtů, takže tyto přidané informace považuji za zanedbatelné. Kapacity jednotlivých nastavení metod jsou uvedeny v tabulkách 6.3, 6.6, 6.9 a 6.12.

6.2.4 Odolnost

Pro testování odolnosti byly na soubory aplikovány různé druhy změn, a následně byly soubory znovu dekódovány. Pokud zpráva nebyla vůbec čitelná, tak byl soubor manuálně

zkontrolován. Tedy byly postupně vypsány všechny bajty původního a změněného souboru, a následně porovnány.

Jako použité změny bylo využito změny vzorkovací frekvence na jinou frekvenci, než krycí soubor využíval, a následně byla vzorkovací frekvence změněna zpět. Stejný postup byl využit v případě délky bloku, tedy 16bitové soubory byly převedeny na 8bitové soubory a zpět. Dále bylo využito zesílení zvuku o několik decibelů. Pro tyto změny byla využita aplikace Audacity. Dále bylo záměrem zjistit odolnost metod vůči MP3 kompresi. Bohužel v oblasti jednotlivých MP3 kompresí jsem nenalezl dostatek dokumentů, které by popisovaly výslednou strukturu souboru. Z tohoto důvodu nebylo možné potvrdit, jak velká část informací byla při kompresi ztracena.

Pro všechny metody jsou výsledky testování odolnosti stejné, z tohoto důvodu jsem je dále nezmiňoval v popisu testování jednotlivých metod. Při změně vzorkovací frekvence a při změně velikosti bloku nedošlo ke ztrátě žádné informace. V případě zesílení zvuku došlo ke změněně všech kódovaných bajtů, tedy celá zpráva byla ztracena.

6.3 Výsledky testování metod

U každé implementované metody je popsán výsledek testování pro každou testovanou vlastnost. Také je u každé metody popsáno, jaká konkrétní nastavení byla pro testování zvolena a jaké byly očekávané výsledky těchto nastavení.

6.3.1 Metoda nejnižšího bitu

Při testování této metody se různá nastavení lišila počtem bitů, které byly v jednom bloku měněny. První nastavení zohledňovalo velikost bloku. Za každý bajt, který tvořil jeden blok, byly kódovány dva bity v posledním bajtu, tedy pro 16bitový soubor byly kódovány 4 bity v druhém bajtu. Druhé nastavení neuvažovalo velikost bloku a pouze kódovalo poslední bit v každém bloku. Očekával jsem, že při prvním nastavení budou změny v některých případech mírně slyšitelné. Kapacita metody by měla být v prvním případě samozřejmě vyšší.

Z výsledků hodnocení slyšitelnosti změn vychází, že většina hodnotitelů považovala v drtivé většině případů soubory za shodné. Případy, kdy byla nějaká nahrávka hodnocena jako rozdílná, byly různé. Tyto případy se nacházely u různých souborů s nahrávkou i u různých verzí nahrávek, v některých případech byla i originální nahrávka považována za horší než ta upravená. Jelikož jsou tyto případy velice ojedinělé a není mezi nimi žádná zřejmá korelace, tak je považuji za zanedbatelné pro hodnocení různých nastavení. Testováním se tedy zjistilo, že mezi prvním a druhým nastavením není rozdíl ve slyšitelnosti změn.

Druh souboru	originál	1. nastavení	2. nastavení
Průměrné hodnocení	1,006	1,022	1,018

Tabulka 6.1: Výsledky hodnocení slyšitelnosti změn u kódování nejméně významného bitu.

Rychlost kódování se u této metody nijak nelišila podle typů využitých krycích souborů, je nejvíce závislá na množství kódovaných bitů. Testováním rychlosti kódování u této metody jsem došel k zajímavým výsledkům, jelikož první nastavení metody má větší rychlost kódování než druhé nastavení. To je způsobeno pravděpodobně tím, že v době běhu programu je zahrnuta i doba zpracování zprávy. Tedy u každého kódování byla zpracována

celá zpráva o délce jednoho milionu znaků, ale ve výsledku byla zakódována pouze část této zprávy, protože soubor neměl dostatečnou kapacitu. První nastavení má větší kapacitu, tedy doba zpracování zprávy se při měření projevila méně než u druhého nastavení, kde byla kapacita výrazně menší.

Druh souboru	1. nastavení	2. nastavení
Rychlost kódování (počet znaků za sekundu)	90563,265	53089,373

Tabulka 6.2: Výsledky rychlosti kódování u kódování nejméně významného bitu.

Z výsledků testování je zřejmé, že kapacita této metody není nijak závislá na typu krycího souboru. Kapacita této metody by měla být při prvním nastavení konstantní, jelikož je vždy kódována čtvrtina zvukových dat. Při druhém nastavení by se měla kapacita lišit podle velikosti bloku. Oba tyto předpoklady byly potvrzeny a z testování kapacity jasně vyplynulo, že první metoda má větší kapacitu.

Druh souboru	1. nastavení	2. nastavení
Kapacita (počet znaků na bajt)	0,2856955	0,0677041

Tabulka 6.3: Výsledky kapacity metody u kódování nejméně významného bitu.

Z výsledků testování vyšlo, že první nastavení má ve všech ohledech lepší nebo shodné vlastnosti. V případě slyšitelnosti nebyl žádný rozdíl ve zvoleném nastavení. V případě kapacity a rychlosti vkládání mělo první nastavení lepší výsledky. Z těchto důvodů bylo zvoleno jako výsledné nastavení této metody v implementaci.

6.3.2 Metoda paritního bitu

Různá nastavení se u této metody lišila ve velikosti regionu, ve kterém byla počítána parita. Pro testování jsem jako první nastavení zvolil velikost regionu shodnou s velikostí jednoho bloku zvukových dat. Při druhém nastavení měl region velikost 4 bloků. Očekával jsem, že v obou případech nebudou změny slyšitelné, nicméně zvolit velikost regionu menší než jeden blok mi přišlo nelogické a v případě 8bitových souborů to není možné.

Výsledky hodnocení slyšitelnosti změn jsou skoro shodné jako výsledky u metody kódující nejméně významný bit, které můžeme vidět v tabulce 6.1. Případy, kdy byla nahrávka hodnocena jako rozdílná, byly znovu velice ojedinělé a není mezi nimi žádná zřejmá korelace. Proto považuji tyto případy za zanedbatelné pro hodnocení nastavení.

Druh souboru	originál	1. nastavení	2. nastavení
Průměrné hodnocení	1	1,022	1,018

Tabulka 6.4: Výsledky hodnocení slyšitelnosti změn při využití paritního bitu.

Stejně jako u předchozí metody bylo testováním zjištěno, že rychlost kódování u této metody není nijak závislá na typu krycího souboru. Rychlost kódování je při obou nastaveních závislá na počtu kódovaných bitů. Zajímavým výsledkem testování je znovu vyšší rychlost prvního nastavení, jelikož má toto nastavení vyšší kapacitu. Tento výsledek pova-

žuji za stejnou odchylku jako při předchozí metodě, tedy zpracování zprávy ovlivnilo dobu běhu programu.

Druh souboru	1. nastavení	2. nastavení
Rychlost kódování (počet znaků za sekundu)	3791,807	1127,138

Tabulka 6.5: Výsledky rychlosti kódování při využití paritního bitu.

Testování kapacity této metody potvrdilo, že kapacita není závislá na typu krycího souboru. Z testování je zřejmé, že obě nastavení jsou závislá na velikosti bloku krycího souboru, tento fakt je zřejmý i z implementace metody. Dle očekávání testování potvrdilo, že první nastavení má výrazně větší kapacitu. Konkrétně je kapacita u první metody přibližně čtyřikrát větší, což bylo předpokládáno, jelikož region je u prvního nastavení také čtyřikrát větší.

Druh souboru	1. nastavení	2. nastavení
Kapacita (počet znaků na bajt)	0,0074401	0,0018599

Tabulka 6.6: Výsledky kapacity metody při využití paritního bitu.

Z testování této meto metody jasně vychází, že první nastavení má lepší vlastnosti. Slyšitelnost změn byla v obou případech srovnatelná. Kapacita i rychlost kódování byly výrazně lepší u prvního nastavení. Z těchto důvodů bylo první nastavení zvoleno jako výchozí nastavení v implementaci.

6.3.3 Metoda nulových úseků

V této metodě se nastavení lišila největším možným rozdílem mezi maximální a minimální hodnotou v úseku. Obě nastavení hledala úseky o minimální délce 4 prvky. První nastavení využívalo jako maximální rozdíl hodnotu 10, druhé nastavení využívalo hodnotu 3. Při prvním nastavení jsem očekával větší kapacitu metody, avšak větší slyšitelnost změn.

Výsledky slyšitelnosti změn u této metody dokazují, že mezi jednotlivými nastaveními je výrazný rozdíl. První nastavení ve většině případů způsobovalo slyšitelné změny, u druhého nastavení jsou změny slyšitelné jen u některých nahrávek. Slyšitelnost změn u této metody se také liší podle krycích souborů. Krycí soubory s hudebními nahrávkami mají často lepší hodnocení než nahrávky mluveného slova. V nahrávkách mluveného slova jsou časté úseky ticha. V těchto úsecích je změna výraznější.

Druh souboru	originál	1. nastavení	2. nastavení
Průměrné hodnocení	1	2,656	1,784

Tabulka 6.7: Výsledky hodnocení slyšitelnosti změn při využití nulových úseků.

Výsledky rychlosti kódování této metody jsou výrazně rozdílné u použitých nastavení. První nastavení dosahuje větší rychlosti kódování, i když doba běhu programu se u různých nastavení zásadně neliší. Větší rychlost kódování u prvního nastavení je způsobena větším množstvím zakódovaných dat. Při kódování do hudebních nahrávek s využitím druhého

nastavení je v některých případech množství zakódovaných informací v řádu jednotek znaků. To může způsobovat výrazně nižší rychlost kódování.

Druh souboru	1. nastavení	2. nastavení
Rychlost kódování (počet znaků za sekundu)	123,109	34,503

Tabulka 6.8: Výsledky rychlosti kódování při využití nulových úseků.

Kapacita této metody je při prvním nastavení výrazně vyšší než při druhém nastavení. Kapacita je rozdílná podle použitého krycího souboru. Při kódování do krycích souborů s hudbou je kapacita ve většině případů několikanásobně nižší než při kódování do nahrávek mluveného slova.

Druh souboru	1. nastavení	2. nastavení
Kapacita (počet znaků na bajt)	0,0004651	0,0001309

Tabulka 6.9: Výsledky kapacity metody při využití nulových úseků.

Z výsledků testování je viditelné, že první nastavení dosahuje výrazně vyšší rychlosti kódování a kapacity. Nicméně změny jsou na tolik slyšitelné, že jsem se rozhodl jako výsledné nastavení využít druhé nastavení. Zároveň výsledky ukazují, že změny jsou výraznější u hudebních nahrávek, ale v případě nahrávek mluveného slova dosahujeme výrazně větší kapacity a rychlosti kódování. Využití této metody je tedy závislé na konkrétních situacích, kdy tuto metodu využíváme a jaké vlastnosti jsou pro nás primární.

6.3.4 Metoda úseků ticha

Různá natavení této metody se liší v minimální délce hledaných úseků ticha. Při prvním nastavení byly hledány úseky o minimální délce 200 bajtů, při druhém nastavení o minimální délce 150 bajtů. Nejvýraznější změnu mezi těmito nastaveními jsem očekával v kapacitě a rychlosti kódování informace. Při využití této metody byly pouze u 12 souborů zakódovány více jak dva znaky. Proto tedy měření vlastností probíhalo pouze na těchto 12 souborech.

Slyšitelnost změn je u obou nastavení podobná. Jiné hodnocení bylo uděleno u různých nahrávek, jelikož mezi nimi není zřejmá žádná korelace, tak tyto rozdíly považuji za zanedbatelné. Do většiny hudebních souborů se nepodařilo zakódovat žádné informace. Z tohoto důvodu není možné určit rozdíl ve slyšitelnosti změn u nahrávek hudby a mluveného slova.

Druh souboru	originál	1. nastavení	2. nastavení
Průměrné hodnocení	1	1,842	1,902

Tabulka 6.10: Výsledky hodnocení slyšitelnosti změn při ukrývání v úsecích ticha.

Rychlost vkládání této metody je závislá na mnoha faktorech. Doba běhu programu je ovlivněna například minimální délkou hledaného úseku, také významnou roli hraje konkrétní posloupnost bajtů. Respektive algoritmus často vyhledává minimální, nebo maximální hodnotu v nějakém úseku. Právě pozice této hodnoty může algoritmus 4 výrazně zpomalit. Z výsledků je zřejmé, že druhé nastavení má větší rychlost kódování. To je způ-

sobeno nejenom kratší dobou běhu programu, ale také větším množstvím zakódovaných hodnot.

Druh souboru	1. nastavení	2. nastavení
Rychlost kódování (počet znaků za sekundu)	5,888	8,601

Tabulka 6.11: Výsledky rychlosti kódování při ukrývání v úsecích ticha.

Z výsledků testování je zřejmé, že kapacita druhého nastavení je mírně vyšší. Kapacita této metody je závislá na použitém typu krycího souboru. U hudebních nahrávek je menší množství úseků ticha než u nahrávek mluveného slova. Další faktorem, proč bylo zakódováno pouze 12 souborů, je konkrétní implementace algoritmu. Jak bylo popsáno v sekci 5.4.4, která obsahuje implementaci této metody, tak v současné době je metoda efektivní pouze na 8 bitové soubory nebo soubory, které obsahují úseky ticha tvořené podobnými hodnotami. Z testovaných souborů byla metoda nejefektivnější na soubory vytvořené převodem textu na mluvené slovo.

Druh souboru	1. nastavení	2. nastavení
Kapacita (počet znaků na bajt)	0,0000141	0,00002

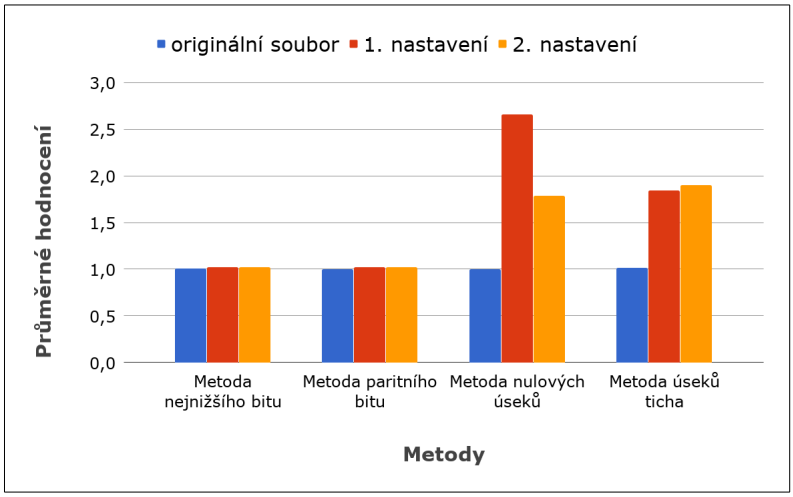
Tabulka 6.12: Výsledky kapacity metody při ukrývání v úsecích ticha.

Na základě testování jsem jako výsledné nastavení v implementaci metody použil druhé nastavení. Důvodem byla vyšší kapacita a rychlost kódování metody při zachování podobné slyšitelnosti změn. Výsledky testování dokazují, že v současné podobě má metoda silně omezené využití a pro další možnosti využití metody je nutné změnit nebo rozšířit implementovaný algoritmus.

6.4 Porovnání implemenovaných metod

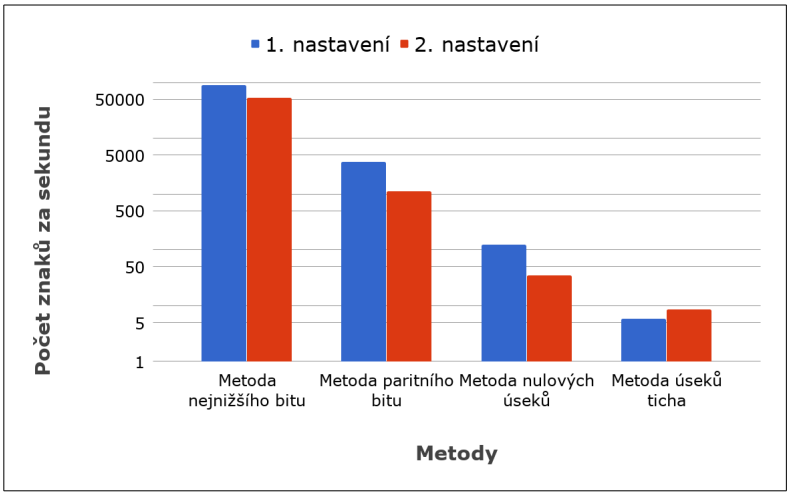
V kapitole 6.3 byly popsány výsledky jednotlivých metod, tyto výsledky můžeme použít pro porovnání implementovaných metod navzájem.

Výsledky hodnocení slyšitelnosti změn jednotlivých metod jasně ukazují, že *Metoda nejnižšího* bitu a *Metoda paritního bitu* mají změny nejméně slyšitelné. U obou metod pro obě nastavení jsou změny naprosto neslyšitelné. Při využití správného nastavení u *Metody nulových úseků* má tato metoda podobné výsledky jako *Metoda úseků ticha*.



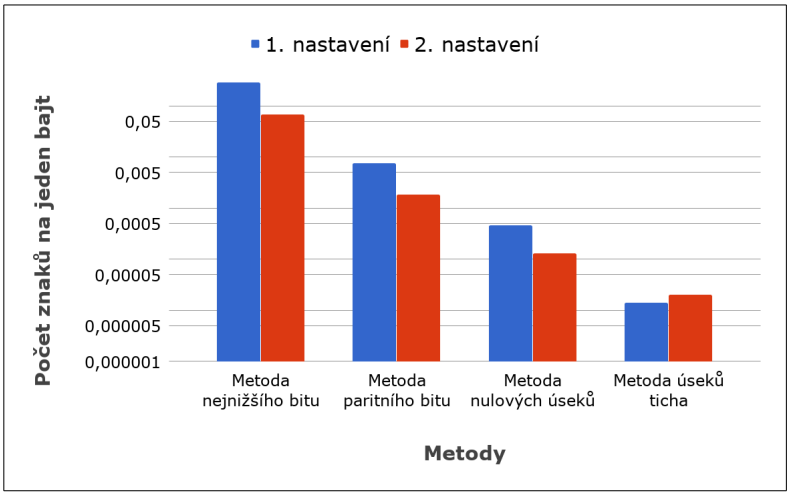
Graf 6.1: Porovnání hodnocení slyšitelnosti změn u všech nastavení.

U výsledků rychlosti kódování dosahujeme významných rozdílů mezi jednotlivými metodami. *Metoda nejnižšího bitu* dosahuje až 20 000krát vyšší rychlosti než *Metoda úseků ticha*. Z tohoto důvodu byla v grafu 6.2 využita logaritmická osa Y.



Graf 6.2: Porovnání rychlosti kódování u všech nastavení.

Výsledky testování kapacity metod ukazují, že žádná metoda nedosáhne kapacity jednoho znaku na jeden bajt. Tento stav je i silně nepravděpodobný, jelikož pravděpodobně žádná metoda nedokáže zakódovat 7 bitů do každého bajtu bez výrazně slyšitelné změny. I nejefektivnější metoda se správným nastavením dosáhla kapacity 0,285 znaku na jeden bajt souboru. Rozdíly v kapacitě jednotlivých metod jsou značné a z tohoto důvodu byla v grafu 6.3 využita logaritmická osa Y.



Graf 6.3: Porovnání kapacity u všech nastavení.

Z celkových výsledků testování jasně vyplývá, že nejefektivnější metodou je *Metoda nejnižšího bitu*, nicméně se jedná o velice známou a jednoduchou metodu, proto pravděpodobnost odhalení případným stegoanalytickým softwarem bude vysoká. Z tohoto důvodu je bezpečnost metody diskutabilní. Dále můžeme říct, že jedinou metodou, která se dokáže zmíněné metodě vyrovnat, je *Metoda paritního bitu*. Tato metoda dosahuje o něco horších výsledků než metoda předchozí, nicméně je stále velice efektivní. Bezpečnost této metody je ze stejných důvodů jako u předchozí metody také diskutabilní.

Metodu nulových úseků podle výsledků považuji za využitelnou, nicméně je nutné zvolit správné nastavení a správný krycí soubor, jelikož změny v krycím souboru byly často výrazně slyšitelné. Metoda úseků ticha sice ve výsledném hodnocení dopadla velice špatně, jelikož ji bylo možné efektivně aplikovat pouze na některé typy krycích souborů, avšak tyto nevýhody přisuzuji implementovanému algoritmu. Pokud by byl tento algoritmus rozšířen, aby byl použitelný na soubory s různými velikostmi bloku zvukových dat, tak by dle mého názoru dosahoval mnohem lepších výsledků.

Kapitola 7

Závěr

Cílem této práce bylo popsat současný stav oblasti digitální audio steganografie. V této oblasti bylo popsáno několik běžně používaných metod. U těchto metod byl uveden nejenom popis jejich fungování, ale také výhody a nevýhody, respektive vhodný způsob použití těchto metod. Celkově považuji oblast digitální audio steganografie za výrazně méně zdokumentovanou, než například oblast digitální obrazové steganografie.

Dále jsem se v práci věnoval implementaci několika vybraných metod. Pro tuto část práce bylo nutné detailně nastudovat fungování jednotlivých zvukových formátů. Původně jsem se zaměřil na formát MP3, protože je v dnešní době hodně rozšířený, ale tento formát jsem následně opustil. Důvodem byla dokumentace tohoto formátu, která nebyla veřejně přístupná, a to výrazně ztěžovalo další práci s tímto formátem. Následně jsem se přesunul k formátu WAVE, pro který jsem metody implementoval. Pro formát MP3 jsem experimentoval s jedinou metodou, nicméně tato metoda je považována za nefunkční, jelikož výstupní soubory jsou silně poškozeny.

Pro testování implementovaných metod jsem vybral čtyři vlastnosti, na které jsem se zaměřil. Vzorek testovaných souborů jsem sestavil z různých typů nahrávek s různým nastavením. Množství souborů jsem omezil tak, aby hodnocení u slyšitelnosti změn nezabralo hodnotitelům více než tři hodiny. Při testování jsem nebral v potaz bezpečnost jednotlivých metod, jelikož pro testování bezpečnosti by bylo nutné, mít přístupný software pro steganoanalýzu.

Implementované metody považuji v současném stavu za využitelné, nicméně pro případné větší použití doporučuji tyto metody rozšířit. Z důvodu větší bezpečnosti bych aplikaci rozšířil o funkci šifrování, aby byly informace před zakódováním zašifrovány. U *Metody paritního bitu* a *Metody nejnižšího bitu* je možné změnit výběr bajtů, do kterých je zpráva kódována. Výsledkem by mohlo být zvýšení bezpečnosti na úkor kapacity, nicméně kapacita těchto metod je tak vysoká, že pravděpodobně nebude nikdy plně využita. Pro *Metodu úseků ticha* by bylo vhodné rozšířit implementovaný algoritmus pro zpracování jiných než 8bitových souborů. V současném stavu aplikace dokáže zpracovávat pouze text, pro případné zpracování jiných informací je nutné aplikaci rozšířit.

Literatura

[1] *MP3 File Structure* [online]. [cit. 2018-05-10].
URL http://www.multiweb.cz/twoinches/mp3inside.htm

[2] *MPEG Audio Layer I/II/III frame header* [online]. [cit. 2018-05-10].
URL http://mpgedit.org/mpgedit/mpeg_format/MP3Format.html

[3] Standards.*International Organization for Standardization* [online]. [cit. 2018-05-10].
URL https://www.iso.org/standards.html

[4] *WAVE PCM soundfile format* [online]. [cit. 2018-05-10].
URL http://soundfile.sapp.org/doc/WaveFormat/

[5] *Multimedia Programming Interface and Data Specifications 1.0* [online]. Aug 1991, [cit. 2018-05-10].
URL http:
//www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/Docs/riffmci.pdf

[6] *Perceptual evaluation of speech quality (PESQ)* [online]. Feb 2001, [cit. 2018-05-10].
URL http://handle.itu.int/11.1002/1000/5374

[7] Artz, D.: *Digital steganography: hiding data within data. IEEE Internet Computing*, May 2001, ISSN 1089-7801, doi:10.1109/4236.935180.

[8] C Judge, J.: *Steganography: Past, Present, Future*. Sep 2001.

[9] Cvejic, N.; Seppänen, T.: *Increasing robustness of LSB audio steganography using a novel embedding method*. Apr 2004, doi:10.1109/ITCC.2004.1286709.

[10] Cvejic, N.; Seppänen, T.: *Reduced distortion bit-modification for LSB audio steganography*. Aug 2004, doi:10.1109/ICOSP.2004.1442244.

[11] Djebbar, F.; Ayad, B.; Hamam, H.; aj.: *A view on latest audio steganography techniques*. April 2011, doi:10.1109/INNOVATIONS.2011.5893859.

[12] Djebbar, F.; Ayad, B.; Meraim, K. A.; aj.: Comparative study of digital audio steganography techniques. *EURASIP Journal on Audio, Speech, and Music Processing*, Oct 2012, ISSN 1687-4722, doi:10.1186/1687-4722-2012-25.
URL https://doi.org/10.1186/1687-4722-2012-25

[13] Dutta, P.; Bhattacharyya, D.; Kim, T.-H.: *Data Hiding in Audio Signal: A Review*. Jan 2009.

[14] Gang, L.; Akansu, A. N.; Ramkumar, M.: *MP3 resistant oblivious steganography*. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, 2001, ISSN 1520-6149, doi:10.1109/ICASSP.2001.941182.

[15] Gopalan, K.; Wenndt, S.: *Audio steganography for covert data transmission by imperceptible tone insertion*. 05 2018.

[16] Gruhl, D.; Lu, A.; Bender, W.: Echo hiding. In *Information Hiding*, editace R. Anderson, Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, ISBN 978-3-540-49589-5.

[17] Shirali-Shahreza, S.; Shirali-Shahreza, M.: *Steganography in Silence Intervals of Speech*. Aug 2008, doi:10.1109/IIH-MSP.2008.5.

[18] Žilka, R.: *Steganografie a stegoanalýza*. Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno, 2008.
URL http://is.muni.cz/th/73058/fi_m/

Příloha A

Obsah přiloženého paměťového média

Přiložené paměťové medium obsahuje následující položky:

- `\app` – složka obsahující zdrojové kódy aplikace.
- `\app\output_files` – složka pro ukládání výstupních souborů aplikace.
- `\test_samples` – složka s několika zvukovými soubory a textovým souborem se zprávou pro testování aplikace.
- `\project` – složka se zdrojovými kódy pro vytvoření pdf souboru dokumentace.
- `readme.txt` – soubor s informacemi o spuštění aplikace.
- `projekt.pdf` – výsledný text práce.

Příloha B

Odkaz na testovací formulář a soubory

Archiv s testovanými zvukovými soubory byl umístěn na osobní uložiště, důvodem byla velikost archivu. Archiv je možné stáhnout z následujícího odkazu: https://drive.google.com/open?id=1FTaM54za5aZNLqA2f0cZ6c_hUwfgLniE.

Pro každého hodnotitele byla vytvořena nová tabulka, do které doplňoval své hodnocení. Vzorová tabulka je přístupná z následujícího odkazu: https://drive.google.com/open?id=1FfaBgJ6tTIOIsAQ_Arj5DcLbX-N7G4VNLPCeh4ng1EE.