Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Metody moderní steganografie

Diplomová práce

Autor: Bc. Pavel Švarc
Studijní obor: Aplikovaná Informatika

Vedoucí práce: Ing. Zuzana Němcová, Ph.D.

Hradec Králové duben 2020

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 29.4.2020 Pavel Švarc

Poděkování:

Děkuji své vedoucí bakalářské práce Ing. Zuzaně Němcové, Ph.D. za pomoc při psaní práce, cenné rady a trpělivost při konzultacích. Dále pak děkuji Univerzitě Hradec Králové za příležitost k vytvoření této práce.

Anotace

Předložená diplomová práce se zabývá steganografií a metodami pro její implementaci. Vzhledem k tomu, že kořeny steganografie sahají až do starověku, snaží se práce čtenáři poskytnou ucelené informace jak o historii, tak o možném využití této vědní disciplíny v moderní době.

První část práce se zabývá teoretickou rovinou steganografie. Popisuje tak právě její historii, základní pojmy, rozdělení, metody a jejich využitelnost v praxi. Hlavní myšlenkou této části práce je informovat čtenáře o možnostech steganografie a postavit teoretické základy pro následující praktickou část a zvýšit tak její srozumitelnost. Protože steganografie je velmi spjata se stegoanalýzou, popisuje tato část také toto odvětví včetně metod a postupů, které jsou ve stegoanalýze používány.

Ve druhé polovině práce je popsaná vlastní implementace metod pro obrázkovou steganografii, společně s chytrým rozhodováním o tom, která metoda bude pro aplikaci steganografie použita. Veškerá implementace je popsaná a vyobrazena v programovacím jazyce Java. V závěru práce jsou všechny implementované metody porovnány a vyhodnoceny pomocí několika kritérií.

Annotation

Title: Methods of modern steganography

Submitted diploma thesis, deals with steganography and with its methods of implementation. Due to the steganography is quite historical and its roots date back to antiquity the thesis tries to provide an overall view about history and about nowadays, current usage as well.

The first part of the thesis is consisting of a theoretical point of view. It describes its history, basic terms, partition, methods their use in practice. The main idea of this work and this part is to inform the reader about steganography and its possibilities and build knowledge basics for a better understanding of the following practical part. As steganography is closely connected to steganalysis it describes this field as well, together with steganalysis methods and its usage.

In the second half, the thesis describes implementation methods used for image steganography together with the implementation of smart method selection. Every implementation's snippet is shown and described in the Java programming language. At the end of this thesis all methods, used in the application, are compared and results of the comparison are described.

Obsah

Seznam obrázků

Seznam tabulek

Seznam ukázek kódu

1 Úvod

Steganografie – kryptografická vědní disciplína, jejíž počátky sahají až do starověkého Egypta. Od té doby byla a je využívána po celém světě, a to jak s dobrým úmyslem, tak s úmyslem špatným. Steganografie je tak spojena s celou řadou světových událostí, jako jsou například útoky 11. září, či obě světové války. Jak byla v minulosti využita a zneužita? Kde se v aktuálním době využívá? Jaké jsou její možnosti? A kde by mohla najít uplatnění v budoucnu? Právě na tyto otázky odpovídá předložená práce.

Ze strukturálního hlediska je práce rozdělená na dvě části. První část práce se věnuje teoretické rovině steganografie a zabývá se tak definicí steganografie, jejím rozdělením (dle různých hledisek), historií a teoreticky popisuje metody, které mohou být pro aplikaci steganografie použity, případně které byly použity v minulosti. Primární pozornost je věnována moderní digitální steganografii. Právě metody používané digitální steganografií, jsou pro tuto práce stěžejní, a tak jsou popsány důkladněji než metody nedigitální steganografie. V této části práce je také popsaná stegoanalýza, její možnosti, metody a aplikace.

Druhá část práce se věnuje praktické rovině steganografie. Cílem této části je popsat technické řešení aplikace vyvinuté v rámci této práce. Je tak popsáno, jakým způsobem byly naprogramovány jednotlivé metody, jaké externí knihovny jsou k implementování potřebné a jaká je časová náročnost jednotlivých metod. Dále pak popisuje, jak bylo v aplikaci implementováno chytré rozhodování a další funkcionality především pro zvýšení uživatelské přívětivosti aplikace.

Na závěr praktické části jsou porovnány výsledky všech implementovaných steganografických metod v aplikaci. Byly porovnávány obrázky před a po aplikování steganografie. Takto testovány byly všechny implementované metody, nad různými obrázky a různě dlouhými vkládanými informacemi.

2 Cíl práce

Primárním cílem práce bylo vytvořit počítačovou aplikaci, která implementuje obrázkovou steganografii. Podmínky kladené na tuto aplikaci byly primárně tři. Aplikace musí implementovat více metod, aby byla použitelná jak pro obrázky v komprimovaném formátu, tak pro obrázky v nekomprimovaném formátu. Mezi těmito metodami musí aplikace aktivně volit a přizpůsobovat výběr metody situaci. Zvolená metoda by tak měla uživateli za daných podmínek nejvíce vyhovovat. Třetí stěžejní kritérium klade důraz na jednoduchost aplikace. A to jak z pohledu uživatelské přívětivosti, tak z pohledu jejího následného rozšíření případně pochopení aplikační logiky.

Druhým cílem práce je porovnání použitých metod. Vzhledem k tomu, že aplikace pracuje pouze s obrázkovou steganografií, porovnávaly se obrázky před a po aplikovaní steganografie. Do různý obrázků tak byl vkládán text různých délek a následně byly obrázky testovány několika metodami, které zkoumaly jejich kvalitu.

Vedlejší cíl je obecné seznámení čtenáře se steganografií, jejími možnostmi a využitelnosti v praxi.

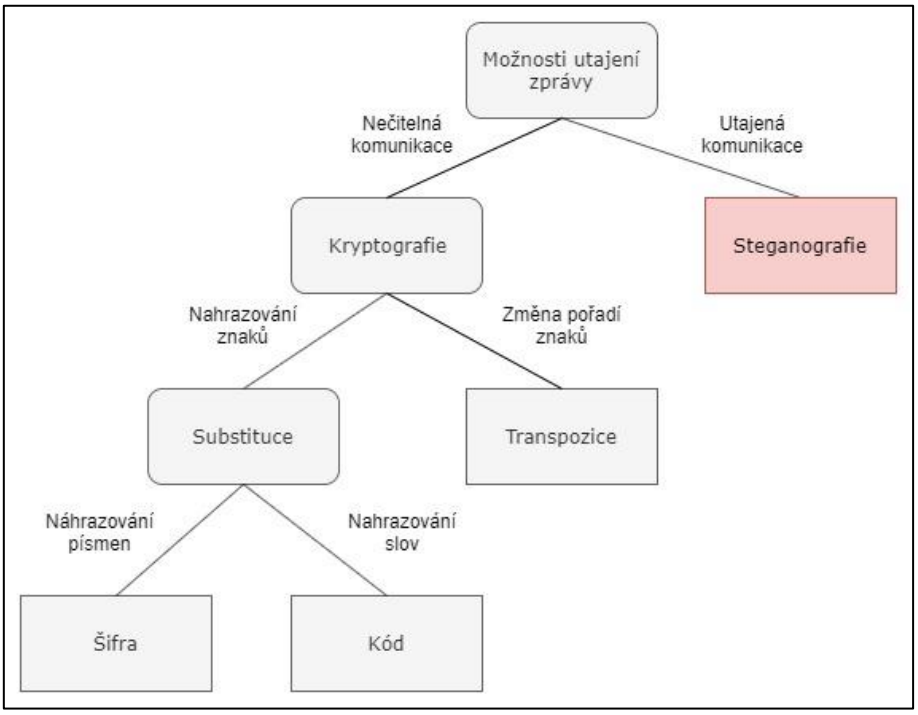
3 Metodika zpracování

První část této diplomové práce je zpracována formou rešerše, která pojednává o steganografii jako celku. Obsahuje tedy také informace o přidružených tématech, jako je například stegoanalýza. Veškeré informace jsou opřeny o vědecké články, odbornou literaturu a věrohodné internetové zdroje.

Druhá — praktická část se zabývá implementováním popsaných steganografických metod v rámci počítačové aplikace napsané v programovacím jazyce Java. Implementováno je také chytré rozhodování o tom, která metoda bude použita společně s dalšími funkcionalitami, které zvyšují uživatelskou přívětivost. Stěžejní části zdrojového kódu jsou v této práci znázorněny a popsány. Následuje porovnání výsledků metod na testovacích obrázcích.

4 Úvod od steganografie

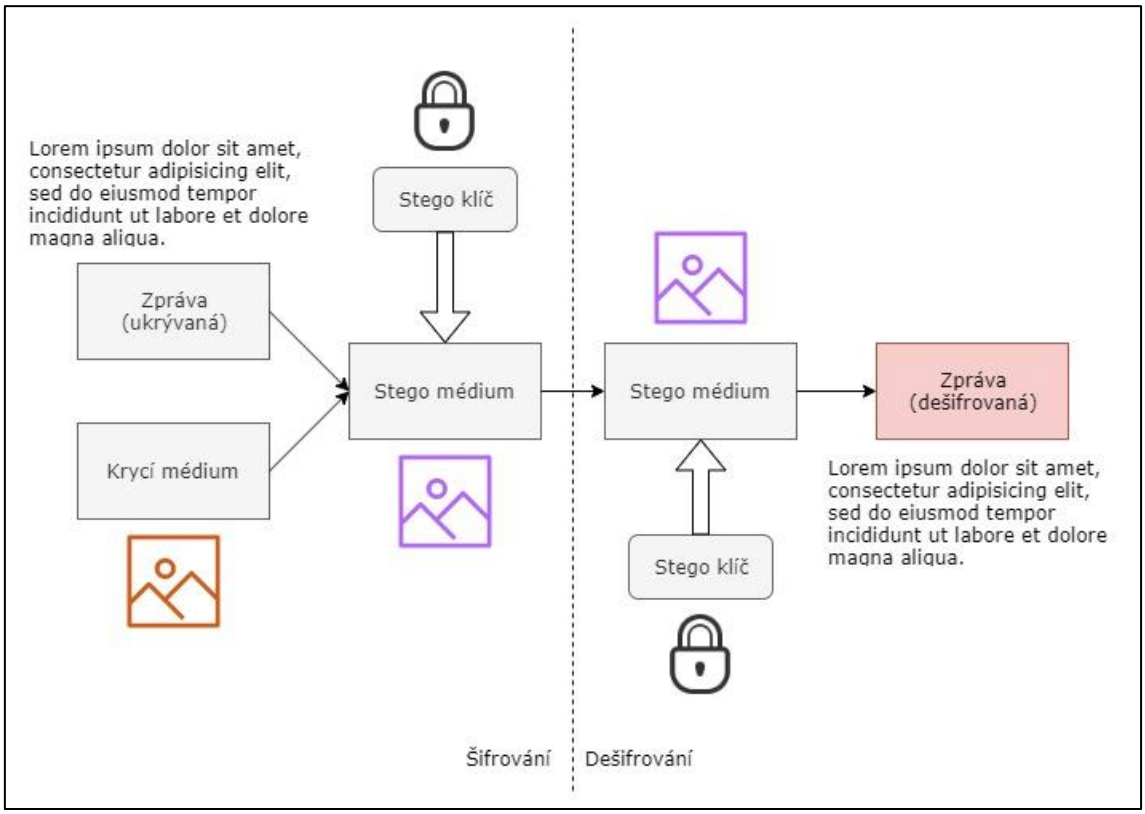
Steganografie je vědní disciplína, která se zabývá ukrýváním informací. Její počátky sahají až do Starověku, což z ní dělá jeden z nejstarších podoborů kryptografie (viz kapitola 5). Narozdíl od kryptografie, steganografie neklade důraz na zašifrování zprávy, ale na její utajení. To v praxi znamená, že v případě steganografie není vydáváno úsilí na to, aby byl text zprávy změněn do nerozluštitelné podoby, ale na to, aby celá zpráva byla utajena. Takto ukrytá zpráva může být následně poslána skrze otevřený kanál, a protože nevykazuje žádné známky tajné komunikace, není předpokládáno její odchycení. Aby takto odeslaná zpráva mohla být adresátem přečtena, je nutné, aby o ní adresát věděl a také aby věděl jakým způsobem má být zpráva přečtena. Na Obr. 1 lze vidět zařazení steganografie do oboru kryptografie. (AL-Ani a spol 2010)



Obr. 1 Rozdělení kryptografie
Zdroj: (AL-Ani a spol 2010)

K této problematice bylo definováno několik pojmů, které jsou uvedeny a vysvětleny níže. Doplněny jsou také o jejich anglický ekvivalent a graficky znázorněny na Obr. 2.

- Zpráva — nejčastěji textový řetězec citlivých dat, který má být ukrytý před veřejností. Zpráva může mít takové podobu obrázku, zvuku či matematických příkladů.
- Krycí médium (cover medium) — data, do kterých je ukryta zpráva. Plní funkci krytí, tedy na první pohled jsou viditelná pouze tato data, nikoli ukrytá zpráva.
- Stego médium (stego medium) — vznikne kombinací zprávy a krycího média. Stego médium je pak posíláno skrze veřejný kanál adresátovi, který z něho získá utajenou zprávu.
- Stego klíč (stego key) — slouží ke snížení rizika odhalení zprávy. Stego klíč se přidává ke zprávě a společně s ní se pak uloží do stego média. Při dešifrování zprávy je povinnost nejprve zadat Stego klíč a až následně je zpráva zobrazena. Kombinuje se tak steganografie a kryptografie.
- Datový obsah (payload) — definuje množství informací, které je možno uschovat do krycího média.



Obr. 2 Proces steganografie
Zdroj: (AL-Ani a spol 2010)

5 Historie steganografie

Podobně jako v jiných oblastech i v oblasti ukrývání a šifrování komunikace jsou významnými milníky První a Druhá světová válka. Z tohoto důvodu je tato kapitola rozdělená do tří podkapitol – Steganografie do konce První světové války, Steganografie do konce Druhé světové války a Steganografie po druhé světové válce. (Das a spol 2013)

Následující kapitoly a samotné rozřazení byly inspirovány publikací *Investigator's Guide to Steganography* od Gregory Kipper a následně doplněny dalšími odbornými publikacemi.

5.1 Steganografie do konce První světové války

5.1.1 Egyptské hieroglyfy

Přestože hieroglyfy jako takové nelze považovat za metodu steganografie, objevovaly se ve starověkém Egyptě případy, kdy tyto „znaky" byly modifikovány tak, aby ukryly jejich skutečný význam. Pouze adresát, který věděl o ukryté zprávě ji dokázal identifikovat a následně správně přečíst. Tento způsob popisuje publikace *Investigator's Guide to Steganography* od autora Gregory Kipper, která jej označuje za vůbec první použití steganografie na světě. (Kipper 2004)

5.1.2 Zpráva na kůži

Tato metoda spočívala v tom, že poslovi byla oholena hlava, na kterou byl následně napsán vzkaz. Poté co poslovy vlasy dorostly byl vyslán, aby zprávu doručil. V případě, že by byl odchycen, nesl také jinou, nedůležitou zprávu, pomocí které by ospravedlnil důvod své cesty. Příjemce pak již jen poslovi hlavu opět oholil a zprávu si mohl přečíst. (Judge 2001)

5.1.3 Pod voskem dřevěných destiček

Také v Řecku bylo zaznamenáno použití steganografie. Jednalo se o situaci, kdy se řek jménem Demaratus rozhodl varovat Spartu před útokem perských vojáků. Způsob, jakým ukryl zprávu, byl na tu dobu vcelku inovativní. V tu dobu se používaly dřevěné destičky pokryté vrstvou vosku jako první zápisníky. Psaní probíhalo

vrytím textu do vosku. Poté co text již nebyl důležitý/potřebný byl zahřátý a připravený k napsání nové zprávy.

Demaratus z těchto destiček vosk seškrábal a napsal zprávu o plánovaném útoku na samotné destičky. Následně je opět zalil voskem, takže destičky nevykazovaly žádné známky uryté zprávy. (Kipper 2004)

5.1.4 Neviditelný inkoust

Velice známý způsob steganografie je použití neviditelného inkoustu. Tato metoda byla objevena Římě a využívala vlastností rostliny Tithymalus. Esence z této rostliny se nanesla na pergamen, ale při psaní pomalu mizela, až zmizela úplně. Pro opětovné získání textu bylo nutné pergamen dostatečně zahřát, například nad plamenem svíčky. Zahřátí způsobilo zuhelnatění esence a zpráva byla zobrazena.

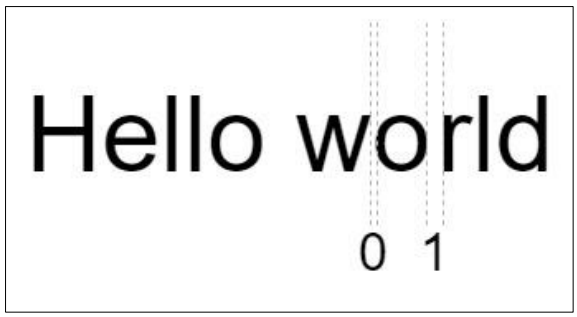
Toto byl pouze začátek této metody steganografie. V průběhu dalších let se používaly další, zdokonalené techniky, které již nevyužívaly tuto rostlinu, ale různé chemikálie. (McKay a McKay, Kate 2011)

5.2 Steganografie do konce Druhé světové války

5.2.1 Textové semagramy

Semagram je symbol, který má předem určený význam. Tento význam zná pouze odesílatel a příjemce. Z toho vyplývá, že při používání této metody se neukrývá text do krycího média, ale využívá se znalost a jedinečnost symbolu.

Textové semagramy využívají grafické modifikace textu, pomocí kterých lze zprávu ukrýt. Často staví na tom, že jsou nevýrazné, ale nezanedbatelné. Jedním takovým příkladem jsou miniaturní mezery. V textu nejsou mezery jen mezi slovy, ale i mezi jednotlivými písmeny. Velikost těchto mezer může být pro lidské oko zanedbatelná, téměř neviditelná, ale pro stroje může mít veliký vliv. Velikost těchto mezer lze totiž transformovat do binárního kódu a s jeho pomocí sestavit zprávu. Zjednodušeně – velká mezera = „1“ a malá mezera = „0“. Ilustrační příklad lze vidět na Obr. 3. (Kipper 2004)



Obr. 3 Miniaturní mezery
Zdroj: (Kipper 2004)

Výše zmíněná metoda vyšla ze starší metody, která využívala psací stroj. Výstup psacího stroje, který býval často speciálně upraven, mohl být modifikován umístěním odchylek od běžného výstupu. Do těchto odchylek mohla být následně ukryta zpráva. Takovými odchylkami mohly být jak mezery zmíněné výše, tak speciální úprava písmen. (Kipper 2004)

5.2.2 Mikrotečky

Metoda vyvinuta Němci a hojně používaná právě během Druhé světové války. Mikrotečky jsou textové řetězce anebo obrázky zmenšené tak moc, že pomocí lidského oka není viditelný žádný obsah. Běžně mívají velikost kolem jednoho milimetru a odpovídají tak například velikosti tečky nad písmenem „i“. Proces tvorby mikrotečky je následující.

Zpráva, která má být ukrytá se vyfotografuje. Tato fotografie se zmenší na velikost přibližně 25x25mm. Po použití inverzního mikroskopu je fotografie zmenšena na již zmíněný milimetr a je vytvořen její negativ. (Kipper 2004, Kumar 2010)

5.2.3 Null cipher

Metoda využívající jiný text jako krycí médium. Její princip je v ukládání písmen ukrývané zprávy do n-tého znaku slov či celého textu. Hodnota n je klíčová k rozluštění vlastní zprávy. (Nag 2019) Mějme například text:

„Fishing freshwater bends and saltwater coasts rewards anyone
feeling stressed. Resourceful anglers usually find masterful leapers
fun and admit swordfish rank and overwhelming any day.”

Vezme-li se pouze každé třetí písmeno každého slova, dostaneme:

„Send lawyers guns and money.“

Příklad byl převzat z publikace *Investigator's Guide to Steganography* od Gregory Kipper. (Kipper 2004)

5.3 Steganografie po Druhé světové válce

Veliký rozmach počítačů během a po Druhé světové válce otevřel nespočet dalších steganografických metod. Nové metody mohou využívat počítače, jejich velký výpočetní výkon a je tak možné urýt skutečně velké množství dat do relativně malého prostoru za velmi krátkou jednotku času. Je zde řeč především o digitální steganografii viz kapitola 6.2.

5.3.1 Autorská práva

Steganografii lze také využívat k zamezování porušování autorských práv. Vzhledem k tomu, že naprostá většina filmů, hudby či jiných autorských souborů je uchovávána v digitální podobě, je poměrně snadné šířit tyto soubory bez patřičných autorských práv. K zamezení tohoto problému je nutné díla označit. Označení by však pro běžného uživatele mělo být neviditelné, jinak by mohlo dojít k jeho snadnému odstranění (například oříznutím či překrytím). Neviditelné označení souboru může nést například informace o autorovi, licenčních podmínkách, či unikátní identifikátor, pomocí kterého může být zpětně dohledán originál, ze kterého plagiát vznikl. I v tomto případě je důležité, aby informace byla nesmazatelná a z toho důvodu se označení vkládá do souboru vícekrát na různá místa. (Sae-Tang a spol 2019)

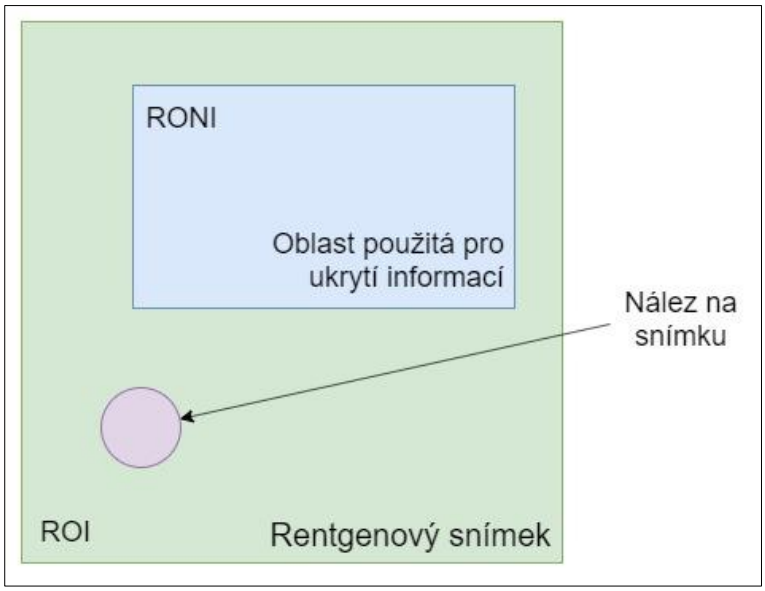
Takto chráněné mohou být také audio soubory. Do zvukové nahrávky jsou opět vložena autorská práva a dohledání originálu, ze kterého byla nahrávka okopírována je významně jednodušší.

5.3.2 Medicína

Využívá se také v medicíně, a to pro spárování snímků s pacientem. V minulosti se již vícekrát stalo, že byl například rentgenový snímek oddělen od pacientovi složky a následně přiřazen k nesprávnému pacientovi. To může mít vážné následky.

Jedním z digitálních způsobů, jak tomu zabránit je steganografie. Pomocí ní jsou vložena data o pacientovy diagnóze anebo navrhovaná léčba přímo do obrazových dat snímku, například rentgenového. Důvod, proč tyto informace nemohly být vkládány přímo nad či pod samotný rentgenový snímek je ten, že data by měla být upravitelná, a především by měl být zjistitelný autor uvedených informací. Mimo jiné v případě dlouhých informací by se stal rentgenový snímek také nepřehledný.

Publikace *Medical image region based watermarking for secured telemedicine* od autora Swarja K. popisuje metodu zvanou Region based. Tato metoda rozděluje snímek na dvě oblasti – Oblast zájmu (region-of-interest – ROI) a Oblast nezájmu (region-of-non-interest – RONI). ROI je taková oblast snímku, která obsahuje důležitá data pro vlastní rentgenový snímek jako je například zlomenina či jiný nález. Z tohoto důvodu nemůže být tato oblast nijak modifikována. Na Obr. 4 lze vidět grafické znázornění této metody. (Swaraja 2018)



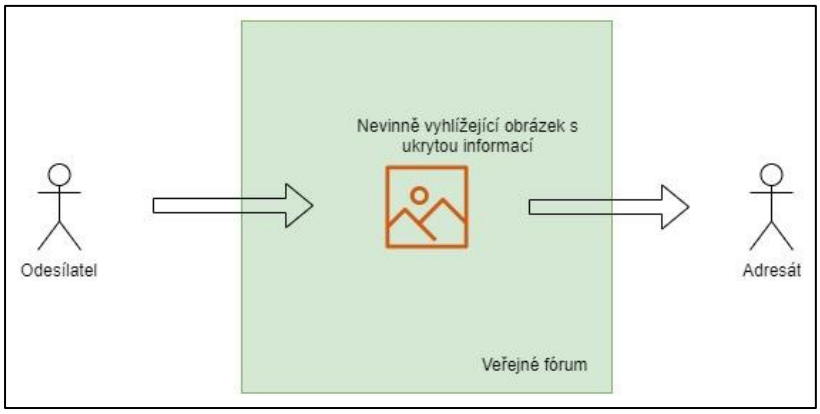
Obr. 4 Region based metoda
Zdroj: (Swaraja 2018)

5.3.3 Terorismus

Jsou zaznamenány případy, že i relativně staré a jednoduché techniky se stále využívají a mohou být efektivní. Po sérii teroristických útoků v září roku 2001 se bezpečnostní složky domnívaly, že veřejná prohlášení, která byla vydávána teroristickou skupinou, mohou obsahovat ukryté informace. Tyto informace mohly být ukryty různými způsoby v barvě oblečení, volených slovech gestech, symbolech, nebo například ve videu jako takovém. Aby se zamezilo šíření potenciálních zpráv dále, byla videa upravována, parafrázována a byla jim věnována jen minimální pozornost. Proto se také tato videa téměř nedostala do zpravodajských médií. (Conway 2003)

Toto podezření se potvrdilo, když byl zajat jeden z kompliců teroristické skupiny, která stála za útoky ze září roku 2001. Komplic prozradil, že skutečně byla využívána jednoduchá forma steganografie. Jeho role byla taková, že předstíral, že je studentem na vysoké škole a psal si se svojí přítelkyní z Německa. Komunikace obsahovala zástupná slova, která ukrývala hlubší význam. Například názvy fakult, které se v komunikaci objevily, ve skutečnosti označovaly cíle teroristických útoků. Časy docházení na fakulty pak časy teroristických útoků. (Conway 2003)

Autor Choudhary ve své publikaci uvádí jiný způsob, jak spolu mohou příslušníci teroristických organizací komunikovat. Tento způsob se využíval již za studené války a nazývá se *Dead Drop*. Metoda spočívá v uložení zprávy na předem smluvené místo, kde je ponechána a následně vyzvednuta adresátem. Odesílatel a příjemce se tak nemusí osobně setkat a ani vzájemně znát svou totožnost. Zatímco v době studené války měly často zprávy fyzickou podobu (například lísteček s textem) a byly umístěny na veřejné místo, ve 21. století se prostření i zprávy změnily. Zpráva je nahrána na volně dostupnou internetovou stránku, kde může být později vyzvednuta adresátem viz Obr. 5. Pro tuto formu komunikace mohou být často využívány chatovací místnosti anebo veřejná fóra. Samotná zpráva pak může být uložena v podobě obrázku, videa či souboru ve formátu pdf. Modifikovaný název této metody je pak *Internet Dead Drop.* (Choudhary 2012)



Obr. 5 Internet Dead Drop
Zdroj: (Choudhary 2012)

Publikace *Steganography in games: A general methodology and its application to the game of Go* zase popisuje vkládání informace do herního prostředí. Komunikace může probíhat na základě smluvených gest a symbolů, nebo přímo vepsáním informace na herní obrazovku. Například ve hře Go mohou být kameny poskládány tak, že tvoří smysluplný text. Tato forma je téměř nezjistitelná, a přesto že je složitá na organizaci. Autor se domnívá, že je hojně využívána. V praxi totiž neexistuje efektivní způsob, jak sledovat herní prostředí a jak interpretovat ukryté zprávy. (Hernandez-astro a spol 2006)

5.3.4 Bankovky

Ve snaze zabránit jejich padělání, případně ho značně zkomplikovat, se steganografie objevuje také na bankovkách. Pravděpodobně se jedná o nejznámější použití steganografie mezi širokou veřejností. Například bankovka s nominální hodnotou tisíc korun českých obsahuje následující ochranné prvky:

- vodoznak,
- okénkový proužek s mikrotextem,
- barevná vlákna,
- soutisková značka,
- skrytý obrazec,
- proměnlivá barva,
- iridiscentní pruh,
- mikrotext.

Ve své podstatě každý z těchto prvků lze označit za určitou formu steganografie.

Zvýšenou pozornost si zaslouží mikrotext. Ten se nachází na více místech bankovky a nese číselnou a textovou nominální hodnotu bankovky. Je vytisknut pomocí hlubotisku i pomocí tisku z plochy. Příklady mikrotextu jsou znázorněny na Obr. 6. (Shah a spol 2015, ČNB 2019)

Obr. 6 Mikrotext
Zdroj: (ČNB 2019)

6 Typy steganografie a jejich praktické využití

Tato kapitola rozděluje steganografické metody z technického pohledu do dvou hlavních podkategorií – nedigitální steganografie a digitální steganografie. Vzhledem k tomu, že některé metody v této kapitole byly již podrobněji probrány v kapitole číslo pět, je na začátku každého typu steganografie uveden seznam již zmíněných metod.

6.1 Nedigitální steganografie

Jak již název napovídá, jedná se o steganografické metody, které nevyužívají digitální technologie. V angličtině jsou často tyto metody označovány jako *low-tech* neboli primitivní metody. Přestože mohou evokovat pocit nízké bezpečnosti z pohledu ukrývání zprávy, jsou často velice účinné a sofistikované. Jejich hlavní nevýhodou je to, že se nedají přenášet digitální cestou, ale pouze fyzicky mezi adresátem a odesílatelem.

Z již zmíněných metod v kapitole číslo pět sem patří:

- egyptské hieroglyfy,

- zpráva na kůži,
- textové semagramy.

Další nedigitální metody jsou pak následující:

6.1.1 Mrkání vězně

Případ, který byl zaznamenán během Války ve Vietnamu pojednává o docela jiné formě steganografie. Americký velitel Jeremiah Denton byl zajat, a pro podporu propagandy byla zveřejňována videa s jeho osobou. Velitel si byl vědom, že nahrávky budou sledovány a ukryl do nich tedy zprávu pomocí mrkání a Morseovy abecedy. Předal tak zprávu, která ukrývala jedno slovo – TORTURE (mučení). Důvod tohoto poselství byl zřejmý – informovat o tom, jak se s vězni na území Vietnamu zachází. (Judge 2001)

6.1.2 Tap code

Tap code je další steganografické technika, která byla zaznamenána během války ve Vietnamu. Tato technika pomohla ke komunikaci mezi americkými vězni a byla založena na tabulce (nejčastěji 5x5, případně 5x7 při rozšíření o čísla). Tato tabulka byla vyplněna písmeny a každému písmenu tak byla přiřazena posloupnost poklepání viz Tab. 1.

	1	**2**	**3**	**4**	**5**
1	A	B	C/K	D	E
2	F	G	H	I	J
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z
6	0	1	2	3	4
7	5	6	7	8	9

Tab. 1 Tap code tabulka
Zdroj: (Judge 2001)

Komunikace následně probíhala tak, že každé písmeno bylo zadáno dvojící poklepání (čísel). První počet poklepání znázorňoval pohyb na ose X a druhý počet

poklepání znázorňoval pohyb po ose Y. Mezi prvním a druhým klepáním byl menší časový rozestup a mezi jednotlivými písmeny delší. Nevýhodou takového přenosu zprávy je nutnost znalosti tabulky. (Judge 2001)

6.2 Digitální steganografie

Tato kapitola je inspirována publikací *Investigator's Guide to Steganography* od autora Gregory Kipper.

Se začátkem rozmachu výpočetní technologie steganografie nezmizela, ale pomalu se začala transformovat právě do digitálního prostředí až se mu plně přizpůsobila. V dnešní době existuje mnoho metod a technik, které se nabízejí k ukrytí zprávy. Stejně tak existuje velké množství krycích médií, do kterých je zprávu možné ukrýt. Příklady krycích médií jsou následující:

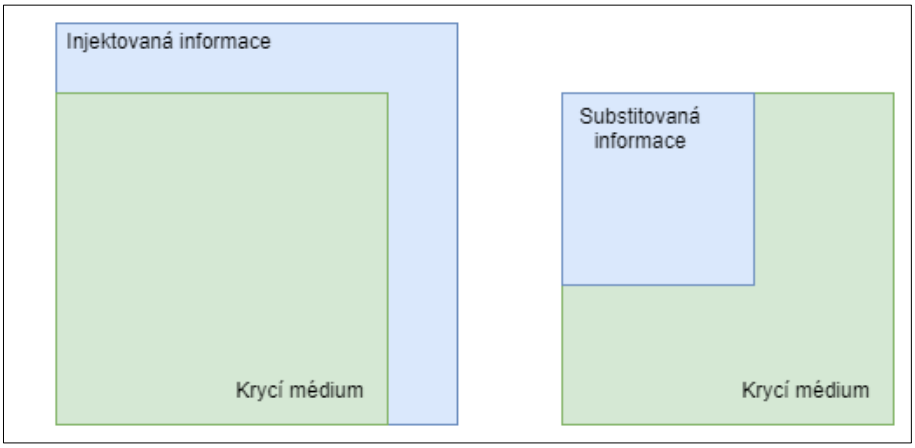
- textové soubory,
- obrázkové soubory,
- video soubory,
- audio soubory,
- síťová komunikace (pakety).

Digitální steganografie, obdobně jako nedigitální steganografie a kryptografie, staví na třech základních principech – injekci, substituci a generování souborů. Uváděné pilíře jsou volně převzaté z publikace *Exploring steganography: Seeing the unseen* od autorů F. Johnsona a S. Jajodia. (Johnson a Jajodia 1998)

Injekce žádným způsobem neovlivňuje původní data krycího média, ale pouze k nim přidává nová data, nesoucí požadovanou informaci. Tím, že se přidávají nová data k již stávajícím, se zvětšuje velikost stego média, a to může vést, především v případě velkých zpráv, ke snadnějšímu detekování stego média.

Naopak substituce data krycího média nahrazuje (modifikuje). Proces nahrazování je takový, že se vytvoří duplikát originální části dat (v případě obrázku například pixel). Tento duplikát je upraven – obohacen o část ukrývané informace a následně umístěn zpátky do originálního celku. Během procesu je snaha o to, aby modifikovaná část krycího média byla, pokud možno, co nejvíce podobná originální

části. A to jak z vizuálního hlediska, tak z hlediska digitálního. Tyto dva pilíře jsou znázorněny na Obr. 7.



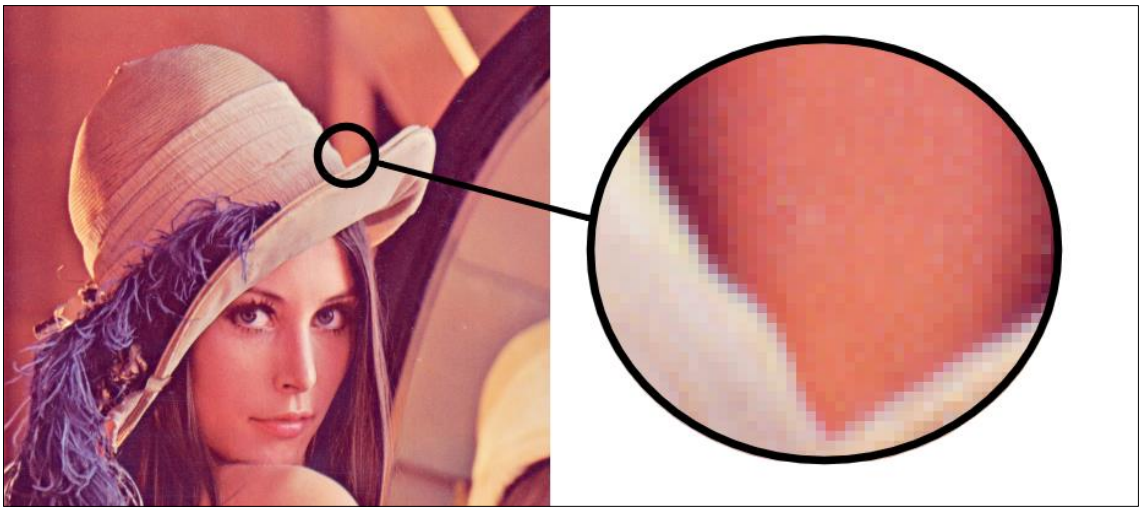
Obr. 7 Injekce a substituce
Zdroj: (Kipper 2004)

Poslední zmiňovaný princip je generování souborů. Jedná se o zcela odlišný způsob ukrývání informace pomocí digitální steganografie. Krycí médium je v tomto případě vygenerováno jen proto, aby ukrývalo zprávu a samo o sobě je zcela bezpředmětné a bezvýznamné. Jediné podmínky, které jsou tak kladeny, je jeho nenápadnost a dostatečný datový obsah. Příklad takového krycího média může být například spam, kde je například informace ukryta v textu anebo přiloženém obrázku.

6.2.1 Obrázkové soubory

Obrázkové soubory jsou v digitální steganografii rozhodně jedním z nejběžnějších typů cover media, a to z více důvodů. První důvod je ten, že jich je na internetu velké množství. Dle serveru Gizmodo.com je pouze na sociální síť Facebook nahráno více než 300 milionů obrázků každý den (Casey 2012). Přičteme-li k tomuto číslu také veškeré e-shopy, fóra, chatovací aplikace a tak dále. Pravděpodobně se lze dostat k číslu 1,8 miliardy obrázků denně, jak uvádí server TheAtlantic.com (Eveleth 2015). Za druhé nabízejí veliký payload, tedy i do relativně malého obrázku, lze vložit obsáhlou informaci. A v neposlední řadě je jednoduché jejich generování. K tomu postačí mobilní telefon s fotoaparátem, webová kamera v počítači, či jednoduchý software na vytváření výstřižků z obrazovky.

Každý digitální obrázek se skládá z několika pixelů, což jsou nejmenší adresovatelné jednotky obrázku. To z nich dělá základní kameny každého obrázku. Každý z těchto pixelů, má barvu, která je určená kombinací tří základních barev – červené, zelené a modré. Každý z těchto bavených kanálů je reprezentován číslem z rozsahu 0-255 (binárně pak 00000000–11111111). Při různé kombinaci intenzit jednotlivých barev lze získat jakoukoliv jinou barvu, která je v digitálním světě realizovatelná. Pixely mohou být detailněji spatřeny na Obr. 8. (Choudhary 2012)



Obr. 8 Pixely v obrázku
Zdroj: (USC 1973)

Do obrázkových souborů nemusí být vkládán pouze text. Častým případem je také vkládání obrázku do obrázku. Vkládání obrázku je realizováno stejným principem jako vkládání textu s tím rozdílem, že se z výsledných extrahovaných bitů neskládá text, ale obrázek. Toto může vyžadovat použití enkodéru a dekodéru. Vkládání obrázku je však mnohem náročnější na velikost dat a tudíž v tomto případě cover médium musí být mnohem větší než vkládaný obrázek. Větší množství vkládaných dát má za následek také větší výpočetní náročnost. (Kumar 2010, Rawat a Bhandari 2013)

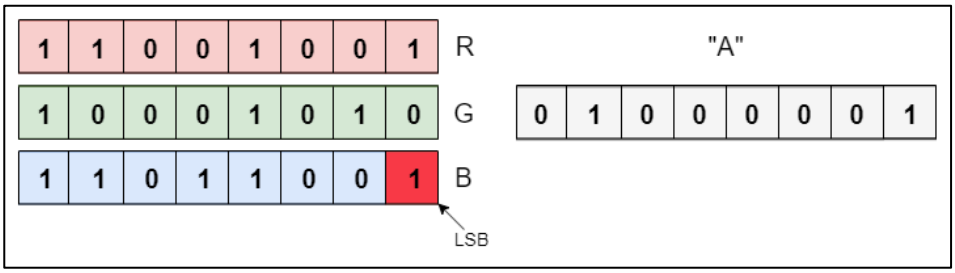
Existuje relativně velké množství metod, pomocí kterých lze do obrázku vložit tajnou informaci. Některé metody jsou použitelné pouze pro specifický formát, jiné jsou univerzální. Některé metody jsou robustní a jiné ne. Z toho vyplývá, že výběr správné metody je pro obrázkovou steganografii stěžejní a může tak velmi

ovlivnit kvalitu a vizuální věrohodnost obrázku, v němž je informace ukrytá. V následujících podkapitolách budou vybrané metody hlouběji popsány.

6.2.1.1 LSB metoda

LSB metoda je označována jako nejjednodušší, a to především díky jednoduché implementaci a pochopitelnosti. Upravuje nejméně významný bit pixelu obrázku. Tento bit je po úpravě shodný s jedním bitem ukrývané zprávy. V případě, že je nejméně významný bit obrázku již shody s bitem, která je potřeba ukrýt, není vykonána žádná změna a algoritmu pokračuje na další LSB dalšího pixelu. Tento prvek „náhodnosti“ může mít ve finále vliv na výpočetní výkon prováděného vkládání. Teoretická šance, že bude mít bit správnou hodnotu je 50 %.

Po vložení zprávy je rozdíl v obrázku je běžným okem nepostřehnutelný. Na Obr. 9 lze spatřit rozdělení barevných kanálů, jejich rozsah v binární podobě, nejméně významný bit a ASCI binární hodnotu pro znak *A*.



Obr. 9 LSB Metoda
Zdroj: (Reddy 2011)

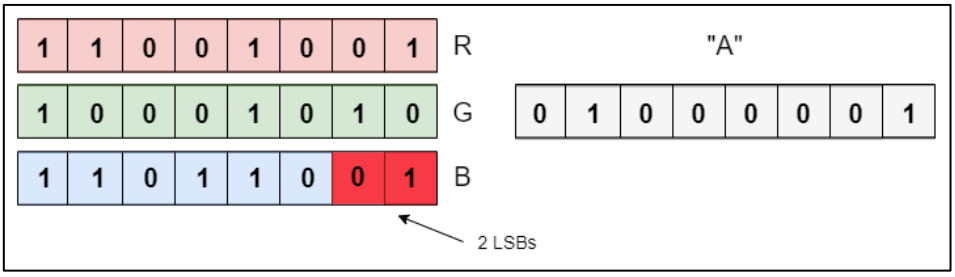
Přestože je tato metoda výkonná z hlediska payloadu (do obrázku se vejde dlouhá zpráva), má velice nízkou robustnost. Stačí jen minimálně obrázek upravit a celá zpráva je nenávratně ztracena. V Obr. 10 bylo uschováno 150 znaků pomocí metody LSB.

Obr. 10 Porovnání cover média a stego média
Zdroj: (USC 1973)

6.2.1.2 LSB metoda pro 2 bity

Jedná se o lehkou modifikaci LSB metody. Rozdíl je však v tom, že není editovaný pouze nejméně významný bit (LSB), ale jsou modifikované dva poslední bity. Na Obr. 1 lze vidět rozdíl oproti klasické LSB metodě. Při této modifikaci, lze do stejně velkého obrázku uložit až dvojnásobně dlouhou informaci. Vzhledem k tomu, že se upravují dva nejméně významné bity, dochází zde již ke znatelnému zhoršení kvality obrázků. Zejména pak u obrázků citlivých na modrou složku.

Obdobným způsobem lze modifikovat ještě více bitů (znázorněno na Obr. 11). Takové vkládání zprávy je ale téměř nepoužitelné, protože cover médium je již znatelně poškozeno, a i při pouhém pohledu je patrné, že s obrázkem není něco v pořádku. Tím by ztrácela takováto steganografie smysl, protože by obrázek již nepůsobil důvěryhodně.

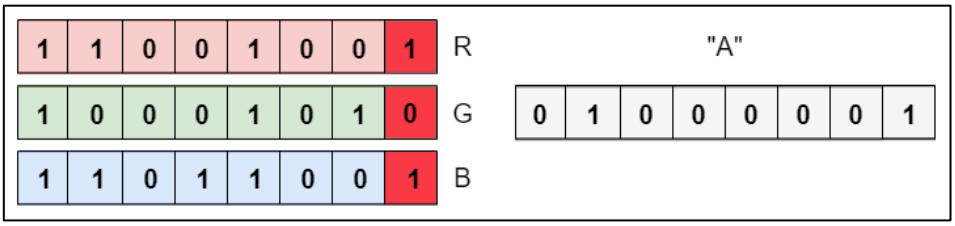


Obr. 11 LSB metoda - 2 bity
Zdroj: (Reddy 2011)

6.2.1.3 LSB metoda pro 3 bity

Dalším zajímavým rozšířením této metody může být vkládání bitů do nejméně významných bitů každé barevné složky viz Obr. 12. Takové vkládání se vyplatí použít především ve dvou případech. První z nich je, že obrázek není dostatečně

veliký pro uložení celé zprávy. Jinými slovy má nedostatečný payload a je potřeba klasickou LSB metodu upravit. Druhý případ je již trošku specifičtější. Ukládání do různých barevných složek může mít za důsledek logické oddělení zprávy. Bude-li se uvažovat tak, že každá složka obsahuje informace o něčem jiném, lze jednoduchým selektivním způsobem zobrazit pouze tu informaci, která je relevantní, případně uživateli dostupná. (Swain a Lenka 2012, Zhao a Kunii 2013)



Obr. 12 LSB metoda 3 bity
Zdroj: (Swain a Lenka 2012, Zhao a Kunii 2013)

Pomocí této metody by se tak čistě teoreticky dalo rozlišit, jaká část zprávy má být zobrazena aktuálnímu uživateli. Lze si představit aplikaci tohoto řešení, kde široká skupina uživatelů smí zobrazovat základní obsah zprávy, ale pouze užší skupina (například vedení) smí zobrazovat veškerý obsah. Široká skupina by v tomto případě měla přístup pouze k modré barevné složce. Skupina vedení pak ke všem barevným složkám. Podobným způsobem by se dalo také omezovat, kdo smí zapisovat a kdo pouze číst.

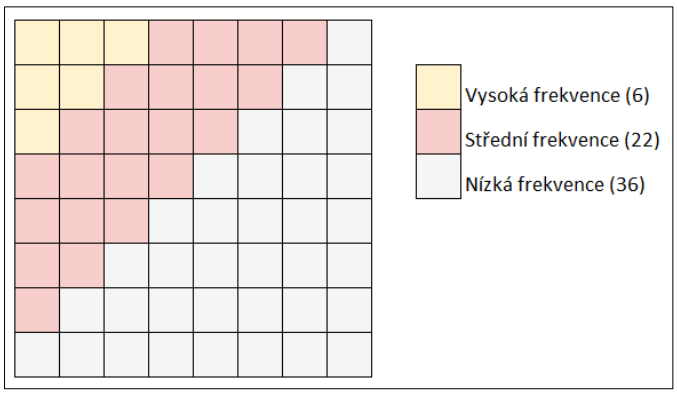
6.2.1.4 Transform domain — DCT metoda

Metoda DCT využívá k ukrývání zprávy prostor transformací, který vzniká při ztrátové kompresi během .jpeg kódování. Obecný a zjednodušený proces vytváření .jpeg souboru je následující:

1. Obrázek je rozdělen na bloky pixelů 8x8
2. Z každého bloku je získána R, G, B složka (rozsah hodnot je <0; 255>)
3. Přepočítání složek na DCT matici (hodnoty jsou nyní desetinná čísla)
4. Kvantizování hodnot pomocí kvantizační matice
5. Zaokrouhlení hodnot na celé čísla
6. **Úprava LSB prvku v kvantizované matici**
7. Získání dat pomocí Zig-Zag metody
8. Komprimace dat pomocí RLE (Run-length encoding)
9. Komprimace dat pomocí Huffmanova kódování
10. Složení .jpeg obrázku

Zpráva je vkládána v bodě číslo 6 z předchozího seznamu. Jestliže se tedy do obrázku žádná zpráva neukládá a je pouze převáděn do .jpeg formátu, je tento bod vynechán. Samotné vkládání zprávy pak již funguje na podobném principu jako vkládání zprávy pomocí LSB metody.

DCT matice reprezentuje barevné koeficienty vzhledem k jejich frekvenci v bloku. Tyto koeficienty se dají rozdělit do tří skupin – koeficienty s nízkou, střední a vysokou frekvencí. Toto rozdělení je znázorněno v Obr. 13. Aby mohl být upraven nejméně významný bit jedné hodnoty je nejprve nutné koeficienty převést na celé čísla. Toho lze docílit použitím kvantizační matice, kterou se DCT matice vynásobí. V kvantizované matici se určí zpravidla jedna hodnota, která bude upravena. Výběr této hodnoty je pro věrohodnost obrázku klíčový. Doporučované je použít koeficient se střední frekvencí, protože koeficienty s nízkou frekvencí jsou velice náchylné, a i malá změna může způsobit relativně veliký, lidským okem rozpoznatelný, rozdíl v obrázku. Tím by se samozřejmě snížila věrohodnost obrázku. Koeficienty s vysokou frekvencí jsou zase velmi náchylné na změnu. Stačila by sebemenší změna v obrázku a tyto koeficienty mohou mít jiné hodnoty. To by mohlo způsobit ztrátu informace a vedlo by tak k mnohem menší robustnosti této metody. (Banik a Bandyopadhyay 2015)



Obr. 13 DCT matice
Zdroj: (Banik a Bandyopadhyay 2015)

Jak již bylo zmíněno, tato metoda dosahuje mnohem větší robustnosti (vzhledem k použití koeficientů ze střední frekvence), než metoda LSB. Na druhou stranu má ale také mnohem menší payload, protože v bloku 8x8 pixelů je ukrytý pouze jeden bit zprávy.

6.2.1.5 Transform domain — DWT metoda

Obdoba metody DCT s tím rozdílem, že není použita Diskrétní Kosinová Transformace, ale je použitá Diskrétní Vlnová Transformace (Discrete Wavelet Trasformation — DWT).

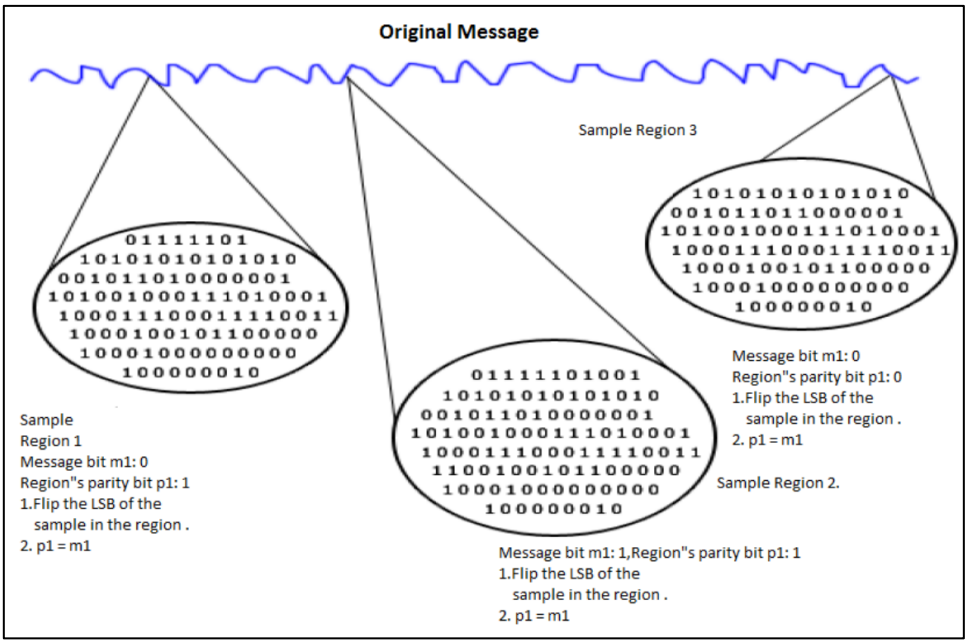
6.2.2 Audio soubory

Podobně jako u jakékoliv jiného způsobu steganografie je i v tomto případě nutné volit kompromis mezi velikostí payloadu, robustností a nenápadností.

Následující metody jsou převzaty z publikace *Audio Steganography by Different Methods* od autorů Swati M., Manishe S. a Dr. Anubhutiho K. (Malviya a spol 2012)

První a nejjednodušší způsob, jak vložit informaci do audio souboru je opět pomocí LSB techniky. V tomto případě se edituje nejméně významný bit zvukového souboru, a to obdobným způsobem jako při použití obrázkové steganografie. Jen s tím rozdílem, že se editují bity zvukového souboru, a ne obrázkového souboru. Jestliže je nahrávka méně kvalitní lze editovat více než jen jeden bit. Šum, který je úpravou generován se v nekvalitní nahrávce ztratí a nebude tolik nápadný. Obecně se však doporučuje needitovat více jak 3 bity. Nevýhody této metody jsou zřejmé. První je, že informace není ukryta ve zvuku jako takovém, ale v souboru, který nahrávku reprezentuje. Dále pak robustnost metody – například využití analogového přenosu právu úplně zničí. (Bannet 2004, Malviya a spol 2012)

Dalším z velmi používaných metod v audio steganografii je takzvaná kontrola parity (Parity checking). Tato metoda využívá paritní bit, což je bit, který se ukrývá vždy za určitou částí zvukového signálu. Jeho účel je takový, že kontroluje úplnost, kvalitu a konzistentnost té části audio signálu za kterou je umístěn. Samotné ukládání informace je pak realizováno pomocí těchto paritních bitů. Jestliže se paritní bit neshoduje s bitem ukrývané zprávy je změněn LSB příslušné části signálu, čímž se změní také paritní bit a vyhovuje tak ukládané zprávě. Tyto změny není možné v celkové zvukové nahrávce poznat. V opačném případě, kdy bit vyhovuje bitu ukládané zprávy se nevykonává žádná akce. (Malviya a spol 2012)



Obr. 14 Paritní kódování
Zdroj: (Malviya a spol 2012)

Dalším velice jednoduchým způsobem je ukrytí informace do šumu nahrávky. Zde se již pro přenos informace nevyužívá soubor jako takový (respektive jeho binární podoba), ale skutečný zvuk. Informace, která má být ukryta se nejprve namluví na nějaké nahrávací zařízení. Tato nahrávka je následně transformována do neslyšitelné frekvence (může mít podobu šumu) a přidána do originální nahrávky – v tomto případě stego média. Příjemci pak již jen zbývá od sebe dvě nahrávky oddělit a šum transformovat. Výsledkem by měla být originální nahrávka. (Malviya a spol 2012)

6.2.3 Video soubory

Obecně řečeno, video soubor je velké množství obrázků jdoucích za sebou, doplněných o zvukovou nahrávku. Lze tedy uvažovat, že video steganografie je určité rozšíření obrázkové a audio steganografie — mohou se využívat metody obou typů. Vzhledem k velkému počtu obrázků, které jsou navíc doplněny o zvukovou nahrávku, která může také uchovávat informaci, dosahuje video steganografie mnohonásobně většího payloadu než předešlé typy steganografie. V době internetu, je však nevýhodou vysoká paměťová náročnost videí. (Sadek a spol 2014)

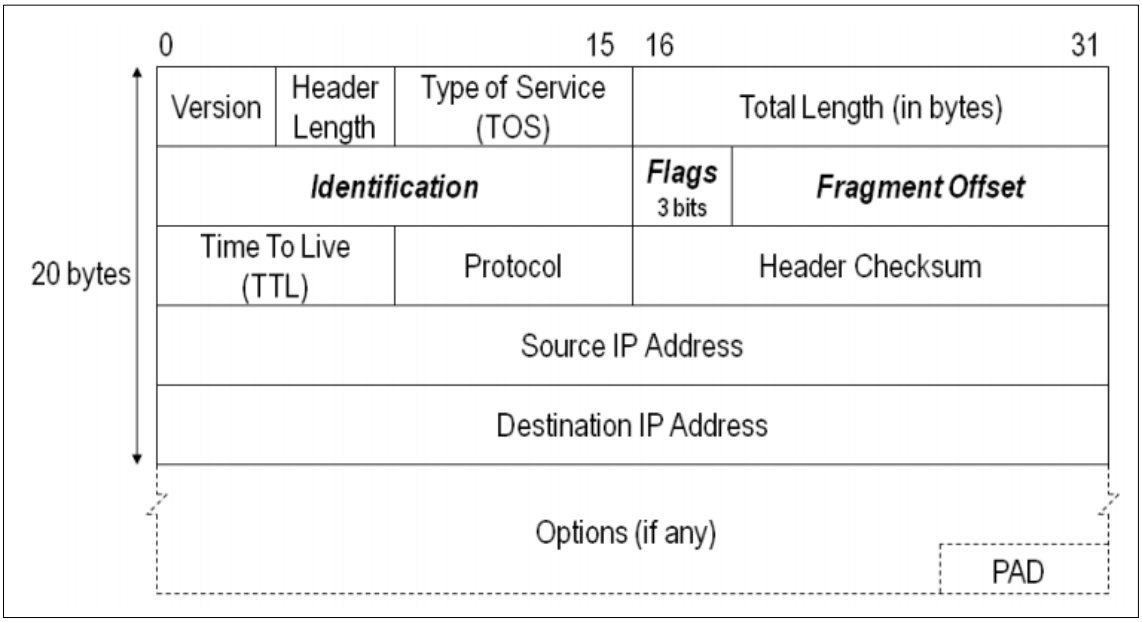
Publikace *Secure Data Transmission Using Video Steganography* od autorů R. Balajiho a G. Naveena popisuje rozšířenou aplikaci video steganografie. Autoři využili již známé steganografické metody a staví právě na tom, že se video skládá z velkého množství obrázků. Nicméně informace není vložena do všech obrázků, ale pouze do náhodně vybraných. Další obrázky videa jsou také upraveny, ale jsou do nich vloženy znaky, které nijak nesouvisí s ukládanou zprávou. Stěžejní pro tuto metodu je tedy určit které obrázky mají být korektně upraveny, má do nich být vložena část ukrývané informace, a které mají sloužit pouze jako krytí. Pro tuto identifikaci autoři používají takzvaný Index Frame (Indexový obrázek). Který definuje obrázky, ve kterých se ukrývá informace a ve kterých ne. Tento Index Frame je definován pomocí předem určené funkce, která pracuje s parametry daného videa, jako je například jeho délka, počet obrázků a tak podobně. Tím pádem je Index Frame pro každé video jiný a také je možné jej jednoznačně určit při získávání informace z videa. Získávání informace pak probíhá takovým způsobem, že se nejdříve vypočítají parametry videa, pomocí nich a předem definované funkce se určí Index Frame a následně se z definovaných obrázků vyčte samotná informace. (Balaji a Naveen 2011)

6.2.4 Síťová komunikace

Tato kapitola byla inspirována publikací *A Study on Network Steganography Methods* od autorů Sekhara A., Kumara M. a Rahimana A. (Sekhar a spol 2015)

Paket – základní prvek síťové komunikace. Realizuje komunikaci a obsahuje data, které se posílají skrze počítačovou síť adresátovi. Tato data anebo samotná struktura paketu může být odesílatelem upravena, tak aby obsahovala část skryté zprávy. V takovém případě se jedná o síťovou steganografii. Existuje vícero metod, které se používají. Primární rozdíl mezi nimi je potom v tom, jaká část paketu se upravuje. Obecným způsobem, je vložit část informace do hlavičky paketu. Hlavička paketu (znázorněná na Obr. 15) obsahuje informace, které jsou volitelné anebo nesouvisí přímo se síťovou komunikací. Lze tak ukrýt zprávu. aniž by se narušil přenos originální informace. Důležité je dohlédnout na to, aby tajná informace nebyla vložena do těch částí hlavičky, které se v průběhu komunikace mění.

V takovém případě by mohlo dojít ke ztrátě informace. (Sekhar a spol 2015, Mazurczyk a spol 2016)



Obr. 15 IPv4 hlavička paketu
Zdroj: (Mazurczyk a spol 2016)

Následuje seznam některých metod, využívaných pro síťovou steganografii. Metody byly zvoleny pro svou zajímavost a odlišné řešení.

RSTEG (Retransmission Steganography) — metoda, která najde uplatnění ve všech protokolech, které využívají opakovaného přenosu. Její princip spočívá v tom, že se nepotvrdí příjem úspěšně přijatého paketu, a tak je vyvolán jeho opětovný přenos. Namísto přenášených dat, znovu zaslaný paket přenáší steganografickou informaci. Vzhledem k velkému počtu opakovaných přenosů paketu, je velice těžké takovouto formu internetové steganografie zachytit.

SCTP (Stream Control Transmission Protocol) — metoda, která může být použita jak v TCP protokolu, tak v UDP protokolu. Využívá toho, že datový přenos je rozdělen do takzvaných chunků a ty jsou rozděleny do jednotlivých proudů. Bity ukrývané informace jsou následně vloženy do proudů. Každý proud může obsahovat například 2 bity ukrývané zprávy. Po následném a správném složení proudů a chunků lze dostat ukrývanou informaci.

PadSteg (Padding Steganography) — metoda využívaná v LAN sítích. Informace je ukrývaná v takzvaných padding bitech Ethernetových framů. Význam

těchto bitů je takový, že je-li payload framu menší než minimální payload (42 oktetů), jsou přidány tyto bity, aby prázdné místo vyplnily. Tato metoda tyto bity zamění za bity ukrývané informace.

TranSteg (Trancoding Steganography) — metoda pro VoIP protokol. Funguje na principu komprimace dat, čímž se zmenší jejich velikost a je tak vytvořen prostor pro ukrytou zprávu. Přitom je udržena vysoká kvalita hlasu. Co se týká detekovatelnosti, je tato metoda lepší než kterákoliv jiná metoda využívající VoIP protokol. Vzhledem k tomuto protokolu, najde využití v real-time systémech.

6.2.5 Souborové systémy

Tato práce do této kategorie zařazuje dva typy steganografie. První z nich by se dal také označit jako disková steganografie. Jde o to, že pevný disk v počítači je rozdělen na takzvané clustery. Cluster je nejmenší adresovatelná jednotka pevného disku. Když se zapisuje soubor na pevný disk, nejprve se alokuje potřebné množství clusterů, vzhledem k velikosti souboru, do kterých se následně zapisuje soubor. V drtivé většině případů se však nevyužije kompletní velikost posledního clusteru, a právě do tohoto clusteru lze vložit bity ukryté informace. Tomuto volnému místu se také říká slack-space. (Kipper 2004)

Druhý typ je realizovaný speciálním souborovým systémem. Takový souborový systém je navržen tak, že ukládaná data jsou uschována do dat, které jsou náhodná. Oproti klasickému zašifrovanému souborovému systému, tato data nejsou běžně viditelná a případný útočník se tak o nich nedozví. Existují dvě metody, které toto ukládání řeší. Metoda číslo 1 nejprve inicializuje skupinu náhodných souboru a následně do těchto souborů data ukládá. Velikost těchto krycích souborů musí být dostatečně velká, ale nezabírá celý disk. Metoda číslo 2 alokuje celý disk náhodnými daty a informace jsou ukrývané do nich. (Anderson a spol 1998)

7 Stegoanalýza a její možnosti

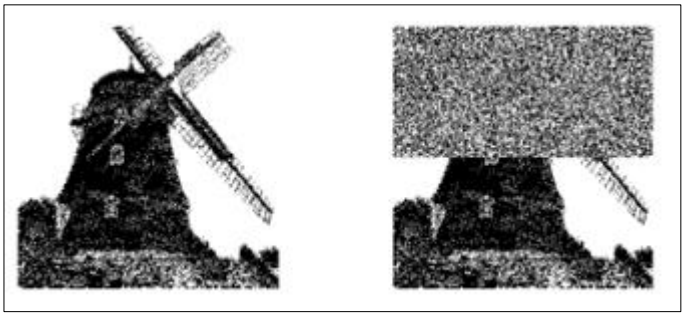
Stegoanalýza je reverzní operace ke steganografii. Jedná se o proces zjišťování, jestli obrázek obsahuje ukrytou informaci – tedy jestli na něj byla aplikovaná nějaká forma steganografie. Sám o sobě je tento proces složitější. Do aplikace stegoanalýzy vstupuje řada neznámých proměnných, jako například určení, jaký obrázek může

potencionálně informaci ukrývat anebo jaká metoda byla využita. Zabývá se tedy extrakcí ukryté zprávy ze stego-média.

Časté využívání steganografických algoritmů (metod) však vede k tomu, že vznikají i metody na její odhalování – stegoanalytické metody. Vyvinuto bylo vícero metod, které se ale dají zařadit do dvou hlavních skupin – vizuální stegoanalýza a statistická stegoanalýza. Nejlepších výsledků v extrakci zpráv dosahují statistické analýzy. Oproti tomu vizuální analýza je snadněji proveditelná. (Provos a Honeyman 2003)

7.1 Vizuální stegoanalýza

Tato metoda se snaží pomocí filtrace obrazu vizuálně odstranit takové informace, které nesouvisí s utajenou zprávou. Takovéto odkrývání zpráv/informací může vytvořit atypické chování (anomálie). Tyto anomálie, lze následně vyhodnotit pouhým pohledem. Pozorovatel dokáže rozpoznat, jestli zbylé informace jsou součást původního nosiče, nebo jestli se jedná o skrytou zprávu.



Obr. 16 Stegoanalýza – filtrace
Zdroj: (Westfeld a Pfitzmann 2000)

Na Obr. 16, lze vidět, jak filtrace pomohla zjistit, že s obrázkem není něco v pořádku. Na první pohled a pouhým lidským okem. (Kipper 2004, Westfeld a Pfitzmann 2000)

7.2 Statistická stegoanalýza

Statistické testování zjišťuje, zda je v souboru skrytá informace, pomocí statistických metod. Test probíhá tak, že se pozoruje, jaká je odchylka určité vlastnosti od statistické normy. V případě, že je odchylka větší než určitá mez, je pravděpodobné, že je v souboru ukrytá informace. Některé tyto testy jsou zcela

nezávislé na použitém typu souboru. Jediným ukazatelem je pak entropie[1]. (Kessler 2004)

V publikaci *Hide and seek: an introduction to Steganography* autoři zmiňují že statistická analýza je výpočetně i časově náročná. A to především z toho důvodu, že nové steganografické algoritmy jsou odolné proti statistickým metodám prvního řádu (statistické metody prvního řádu jsou například průměry, nebo rozptyly). Zašifrování ukrývané zprávy také komplikuje statistické útoky, protože taková data vykazují mnohem větší míru náhodnosti, a to je v tomto případě nežádoucí. (Provos a Honeyman 2003)

7.2.1 Raw Quick Pair analýza

Tato metoda odhaluje zprávy ukryté pomocí metod LSB. Celá tato detekce pracuje na principu srovnávání dvojic barev, které jsou si vzájemně blízké. V obrázku se nejprve určí počet unikátních barev a následně se z této množiny určí počet párů barev, které si jsou nejvíce podobné – tzv. blízké barvy. Vyjádří se rovnice:

$$R = \frac{P}{\binom{U}{2}}$$

kde U je počet jedinečných barev a P je počet párů blízkých barev. Dále jsou vybrány zcela náhodné pixely a do jejich nejméně významných bitů jsou vloženy bity testovacího textu[2]. Když je obrázek takto upravený opět se zjistí počet unikátních barev a počet párů blízkých barev. Nakonec se provede znovu výpočet poměru R podle předchozí rovnice a výsledky se porovnají. Obrázek s ukrytou informací bude mít vždy nižší poměr než obrázek se zprávou neukrytou. (Fridrich a Goljan 2002)

7.2.2 RS analýza

RS analýza je metoda, pomocí které lze nejen zjistit, jestli je v souboru ukrytá informace, ale také její délku, jestliže je ukrytá pomocí techniky LSB. Primárně se

---

[1] Entropii lze chápat jako hodnotu, která udává míru neuspořádanosti, respektive uspořádanosti. Pro snadnější pochopení, lze tuto hodnotu popsat také jako rozložení bodů v prostoru. Jsou-li body „ostře rozložené“ například jako výsledek prahování, mají hodnotu entropie velice nízkou. Nejvyšší entropie dosahují hodnoty, které mají normální rozdělení.

[2] Testovací text by měl mít optimální délku, neměl by tedy být ani příliš krátký ani příliš dlouhý. Optimální délka se nejlépe určí testováním na podobném obrázku.

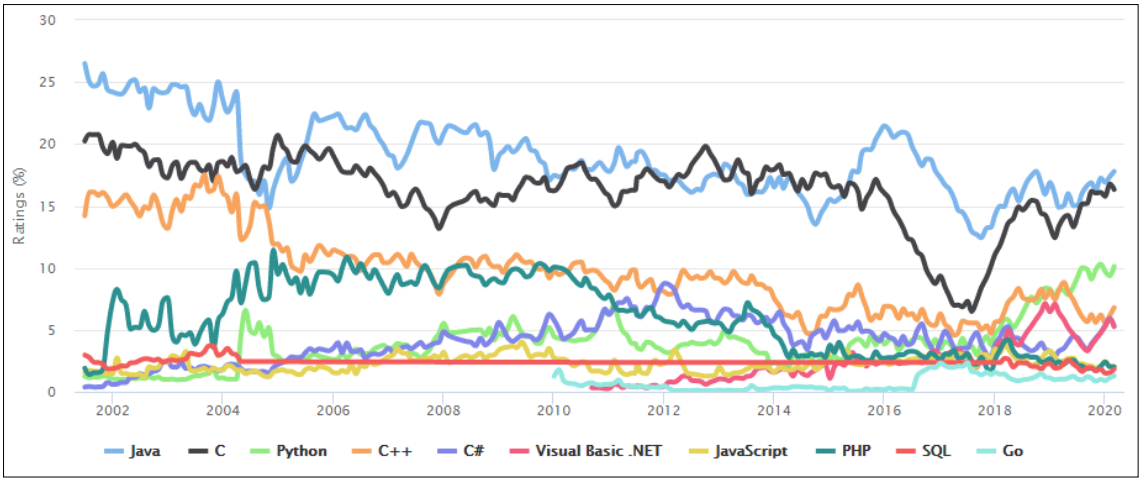
používá na detekci osmi bitových černobílých obrázků anebo dvaceti čtyř bitových nekomprimovaných, barevných obrázků. Důležité je, že je určena pouze pro nekomprimované formáty. Princip spočívá v tom, že mezi sousedními pixely vzniká nerovnováha. Tato nerovnováha následně způsobuje šum v obrázku, který je již měřitelný pomocí definované funkce. Je-li následně hodnota šumu vysoká, lze předpokládat, že se v obrázku nachází skrytá zpráva. (Fridrich a spol 2003)

7.2.3 Chí-square analýza

Tato analýza odhaluje zašifrované zprávy pomocí techniky LSB. Přepsáním tohoto bitu dojde ke změně barvy (pro lidské oko nepostřehnutelná změna). Následně se porovnává statisticky předpokládaná četnost barev a aktuální četností a na základě tohoto porovnání lze určit, jestli se v obrázku ukrývá skrytá zpráva. (Westfeld a Pfitzmann 2000)

8 Programovací jazyk Java

Java je objektově orientovaný programovací jazyk, jehož hlavní výhodou je jeho přenositelnost. Tento jazyk lze provozovat na desktopech, mobilních zařízeních anebo například jednočipových počítačích. Firma TIOBE se zabývá měřením popularity programovacích jazyků vzhledem k vyhledávání v internetových vyhledávačích. Vytvořila tak takzvaný TIOBE index. Programovací jazyk Java dosahuje v průměru nejvyšších hodnot, dalo by se tak říct, že dle tohoto indexu je Java nejpopulárnější programovací jazyk, jak lze vidět na Obr. 17. Základní vlastnosti tohoto programovacího jazyku jsou následující. (TIOBE 2020)



Obr. 17 TIOBE index – časová osa
Zdroj: (TIOBE 2020)

Typovaný jazyk — všechny deklarované proměnné musí mít předem definovaný datový typ. Typování přispívá k přehlednosti kódu a zrychluje debuggování[3].

Objektově orientovaný jazyk — v Javě jsou všechny datové typy objektové. Výjimku tvoří pouze osm primitivních datových typů. Tyto primitivní datové typy slouží jako základní stavební prvky objektů (Eckle 2006). Primitivní datové typy jsou uvedeny v Tab. 2.

Primitive type	Size	Minimum	Maximum	Wrapper type
boolean	—	—	—	**Boolean**
char	16 bits	Unicode 0	Unicode 2^{16}- 1	**Character**
byte	8 bits	-128	+127	**Byte**
short	16 bits	-2^{15}	$+2^{15}-1$	**Short**
int	32 bits	-2^{31}	$+2^{31}-1$	**Integer**
long	64 bits	-2^{63}	$+2^{63}-1$	**Long**
float	32 bits	IEEE754	IEEE754	**Float**
double	64 bits	IEEE754	IEEE754	**Double**
void	—	—	—	**Void**

Tab. 2 Primitivní datové typy
Zdroj: (Eckle 2006)

[3] Debuggování neboli ladění je proces, kdy vývojář hledá chybu v kódu.

Interpretovaný jazyk — využívá tedy program zvaný interpret, který zdrojový kód vykonává. V Javě se tento interpret nazývá Virtuální Stroj Javy (JVM). Opakem interpretovaných jazyků jsou jazyky kompilované. U nich se zdrojový kód překládá překladačem do strojového kódu.

Architektonicky nezávislí jazyk — aplikace vytvořená v programovacím jazyce Java běží na jakémkoliv stroji s jakoukoliv architekturou. Jedinou podmínkou je, aby stroj disponoval Virtuálním Strojem Javy, který interpretuje zdrojový kód.

Robustní jazyk — vzhledem k tomu, že je jazyk silně typovaný, lze předpokládat, že software, který produkuje bude mnohem robustnější (méně náchylný na chyby).

Víceúlohový jazyk — jazyk Java podporuje vyvíjení více vláknových aplikací.

Následuje ukázka Hello World aplikace (viz Kód 1) v jazyce Java. Tato aplikace do konzole vypíše zprávu „*Hello, World*“.

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Kód 1 Ukázka aplikace Hello World
Zdroj: (Vlastní zpracování)

8.1 Historie

Kapitola věnující se historii jazyka Java je inspirována webem *javatpoint.com.* (Javatpoint 2018)

Programovací jazyk Java byl původně vyvinutý pro takzvanou interaktivní televizi. Myšlenka byla taková, že by televize mohla „komunikovat“ s uživatelem pomocí set-top boxu. Software v tomto set-top boxu by byl naprogramován právě Javou. Bohužel však tato myšlenka příliš nadčasová a nerealizovatelná vzhledem k technologii, kterou používala kabelová televize. Za touto myšlenkou stál GreenTeam. Hlavními osobnostmi tohoto týmu byli James Gosling, Mike Sheridan, a Patrick Naughton. Tento tým vývojářů Javu také naprogramoval.

První název tohoto projektu však nebyl Java. Jazyk byl poprvé uveden pod názvem *GREENTALK by James Gosling* a to v roce 1994. Tento název se nepoužíval

dlouho. Následně byl jazyk přejmenován na *OAK*. Oak znamená v překladu z angličtinu dub. Tento název byl údajně zvolen proto, že dub je symbol síly. V roce 1995 společnost *OAK Technoligies* si stěžovala, že název je příliš podobný tomu jejich a jazyk *OAK* byl nucen se přejmenovat. Vznik tak název Java. Java je také název ostrovu v Indonésii, kde byla produkována první káva. Aktuálně je Java dceřiná společnost společnosti *Oracle Corporation*. (Javatpoint 2018)

8.2 Verze

V seznamu níže jsou uvedeny jednotlivé verze Javy včetně roku vydání. Verze jsou také doplněny o vybrané funkcionality, které přináší.

- **JDK 1.0** (23. leden, 1996)
  - První stabilní verze Javy
- **JDK 1.1** (19. únor, 1997)
  - Vnitřní třídy
  - JDBC
  - Renovace AWT modelu
- **J2SE 1.2** (8. prosinec, 1998)
  - Integrováno grafické rozhraní SWING
- **J2SE 1.3** (8. květen, 2000)
  - JavaSound
  - JPDA přidáno (Java Platform Debugger Architecture)
- **J2SE 1.4** (6. únor, 2002)
  - Vylepšené knihovny
  - Přidány PERL regulární výrazy
  - Podpora IPv6
  - Kryptografické a bezpečnostní rozšíření přidáno
- **J2SE 5.0** (30. září, 2004)
  - SWING skinny look
  - Metadata a anotace
  - Statické importy
- **JAVA SE 6** (11. prosinec, 2006)

    - Podpora JDBC 4.0
    - Vylepšená podpora webových služeb
- **JAVA SE 7** (28. červenec, 2011)
    - JVM podpora pro dynamické jazyky
    - Vylepšené zachycování výjimek
- **JAVA SE 8** (18. březen, 2014)
    - Podpora lambda výrazů
    - Vkládání Java Scriptového kódu do aplikací
    - Podpora anotací
- **JAVA SE 9** (21. září, 2017)
    - Přidáno API pro práci s měnami
    - Podpora automatického škálování
- **JAVA SE 10** (20. březen, 2018)
    - Přidán interface pro garbage collector
    - Lepší alokace datového typu Heap
- **JAVA SE 11** (25. září, 2018)
    - Odstranění modulů JavaFX, Java EE a CORBA3
    - Podpora Unicode 10.0
- **JAVA SE 12** (19. březen, 2019)
    - Výchozí CDS archivy
- **JAVA SE 13** (17. září, 2019)
    - Přidán datový typ Text Block

Informace o verzích byly převzaty z webových stránek *javatpoint.com* a *openjdk.java.net.* (Javatpoint 2018, OpenJDK 2020)

9 Vývojové prostředí IntelliJ IDEA

Jako vývojové prostředí pro vývoj aplikace, která je vytvořena v rámci této práce, bylo zvoleno prostředí s názvem IntelliJ IDEA od společnosti JetBrains. Toto IDE má, mimo autorovi osobní preference, mnoho výhod a pokročilých funkcionalit, které se při vývoji jakékoliv aplikace v programovacím jazyce Java velmi hodí a usnadňují tak

samotný vývoj. Prostředí pak přichází na trh ve dvou variantách – Community a Ultimate. Community verze je šířena pod Open-source licencí, ale je ochuzena o nějaké, především pro firmy a webové aplikace kritické, funkce. Na Tab. 3 lze vidět rozdíly mezi těmito verzemi (k 5.4.2020). Cena Ultimate verze se pohybuje od 7 3754 kč do 12 290 kč za rok pro společnosti a od 2 274 kč do 3 790 kč za rok pro jednotlivce. (JetBrains 2020)

	Ultimate	**Community**
Licence	Komerční	Open-source
Java, Kotlin, Groovy, Scala	✓	✓
Android	✓	✓
Maven, Gradle, sbt	✓	✓
Git, SVN, Mercurial	✓	✓
Debugger	✓	✓
Profiling tools	✓	×
Spring, Java EE, Micronaut, Quarkus, Helidon, and more	✓	×
Swagger, Open API Specifications	✓	×
JavaScript, TypeScript	✓	×
Database Tools, SQL	✓	×
Detecting Duplicates	✓	×

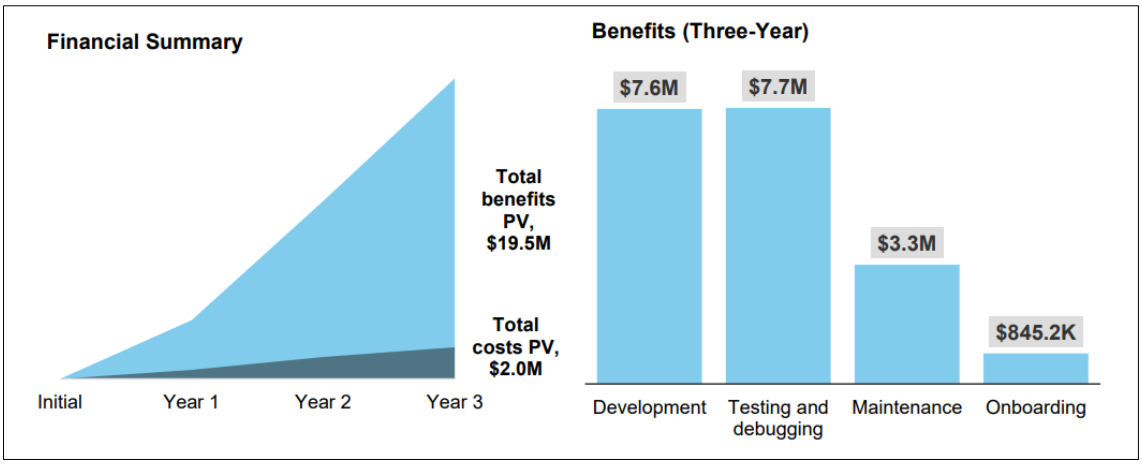
Tab. 3 IntelliJ IDEA porovnání verzí
Zdroj: (JetBrains 2020)

Mezi hlavní výhody, mimo velice intuitivní uživatelské rozhraní, patří výborný systém napovídání. Vzhledem k tomu, že je celý zdrojový kód indexován je prostředí schopno rychle napovídat relevantní části kódu. Napovídání je však závislé na aktuální třídě či metodě a je tak skutečně použitelné v daný moment. Nejsou napovídány hodnoty, které jsou za syntaktického hlediska správné, ale z logického špatné. Napovídání není použitelné pouze pro programovací jazyk Java, ale pro všechny jazyky, které mohou být použity v tomto IDE. Jedná se například o jazyky JavaScrit, SQL či HTML. (JetBrains 2020)

Další výhodou je, že veliké množství funkcionalit, jako například verzování, je součástí instalačního balíku prostředí. Není tak nutné tyto funkcionality separátně instalovat pomocí pluginů. (JetBrains 2020)

Tyto výhody společně pak s dalšími (integrovaný HTTP klient, vícejazyčná podpora, detekování duplikátů, navrhování oprav, definování vlastních zkratek atd.) a velkou podporou frameworků (Kotlin, Groovy, Spring, Jave EE, Grails, React Native, Android, React, Node.js, Angular atd.) dělají z tohoto IDE velice mocný nástroj. (JetBrains 2020)

V roce 2018 konzultační společnost Forrester zveřejnila případovou studii, ve které řeší, návratnost investic právě do vývojového prostředí IntelliJ IDEA v rámci organizací. Hlavní výstupy této studie byly, že firmě se v průměru za 6 měsíců vrátí veškeré investice vložené do tohoto IDE. Za 3 roky je pak vratnosti investic až 850 % viz Obr. 18. Dále pak z ní například vzešlo, že vývojáři díky tomuto IDE ušetří v průměru 270 hodin za rok. (Forrester 2018)



Obr. 18 IntelliJ IDEA finanční návratnost
Zdroj: (Forrester 2018)

10 Úvod do praktické části

Problém nastíněný v kapitole 6.2.1 je tedy v tom, jakou metodu pro aplikaci obrázkové steganografie zvolit. Tento problém by se dal označit jako stěžejní. Je žádoucí, aby metoda umožňovala uložit co nejdelší zprávu (měl největší payload) a zároveň aby dosahovala požadované robustnosti. Pro dosáhnutí nejlepších výsledků, by rozhodování mělo probíhat podle proměnlivých parametrů jako je například délka ukládané zprávy, typ cover media, nebo typ zdroje cover media (URL adresa či soubor).

Toto by pomohla vyřešit aplikace, která bude využívat vícero metod, a která se bude sama aktivně rozhodovat, jaká metoda bude použitá k uložení dané informace.

Celý proces rozhodování, jakou metodu použít by byl přitom odstíněný od uživatele, což by velice zjednodušilo používání aplikace a zabránilo tak nesprávnému nastavení aplikace. Uvažujeme-li, že aplikaci budou používat také uživatelé méně technicky znalí, jeden z možných praktických scénářů by mohl být následující:

1. spočítat délku ukrývané zprávy,
2. spočítat payload cover image pro každou metodu,
3. dle payload určit nejvíce robustní metodu, která je může zprávu uschovat a pomocí které bude zpráva zakódována (obvykle platí, že čím robustnější metoda tím kratší zpráva může být uschována).

11 Konkurenční aplikace

V současné době dostupné steganografické aplikace toto neumožňují. Často buď implementují pouze jednu metodu, která není vždy, pro danou situaci, vhodná, nebo jsou předimenzované, což může způsobit jejich nepřehlednost a odradit uživatele. Následuje seznam, kde jsou jednotlivé aplikace doplněny také o stručný popis.

Hide In Picture — tato aplikace od autora Daviho Figueireda je velice minimalistická. Navzdory tomu obsahuje také kódování zprávy pomocí algoritmů Blowfish anebo Rijindael. Zprávu však lze vložit pouze do obrázků typu .gif a .bmp. Vzhledem k tomu, že to jsou typy velice náročné z pohledu velikosti souboru, nejsou vhodné pro použití v internetovém prostředí, a tudíž tato aplikace není vhodná. (Figueired 2013)

OpenPuff — aplikace vytvořená autorem Cosimem Olibonimim, známá také jako první steganografický nástroj. Aplikace obsahuje vysoké množství podporovaných souborů včetně zvukových a pdf souborů. Nicméně neobsahuje vkládání obrázků skrze URL adresu a vzhledem k několika-krokovému průvodci by se dala zhodnotit jako složitá a celkově neintuitivní. (Olibonimi 2019)

QuickStego Software — aplikace, která není šířena pomocí freewarové licence a je tady nutná její koupě. Mimo steganografii zvládá také zamykání souborů, šifrování emailů či správu hesel. Větší počet funkcí přispívá k celkové nepřehlednosti aplikace. (QuickCrypto 2017)

Invisible Secrets — jako v případě QuickStego Software se jedná o komplexní aplikace, která je také placená, a která obsahuje celou škálu bezpečnostních služeb. Disponuje také velice podobnými nevýhodami. (east-tec 2013)

SilentEye — aplikace je zajímavá zejména proto, že je multiplatformní. Mezi její nevýhody patří manuální volba mezi kódováním a dekódováním a nemožnost vložení obrázku skrze URL. (Chorein 2010)

Žádná z uvedených aplikací tedy nenabízí všechny prvky, které jsou stěžejní pro jednoduché, a přitom efektivní použití obrázkové steganografie pro méně znalé uživatele v době internetu. Tyto stěžejní prvky pak jsou následující:

- automatický výběr metody steganografie,
- načtení obrázku z URL adresy,
- vygenerování náhodného cover media (není-li dostupný),
- jednoduché a intuitivní uživatelské grafické rozhraní,
- freeware licence.

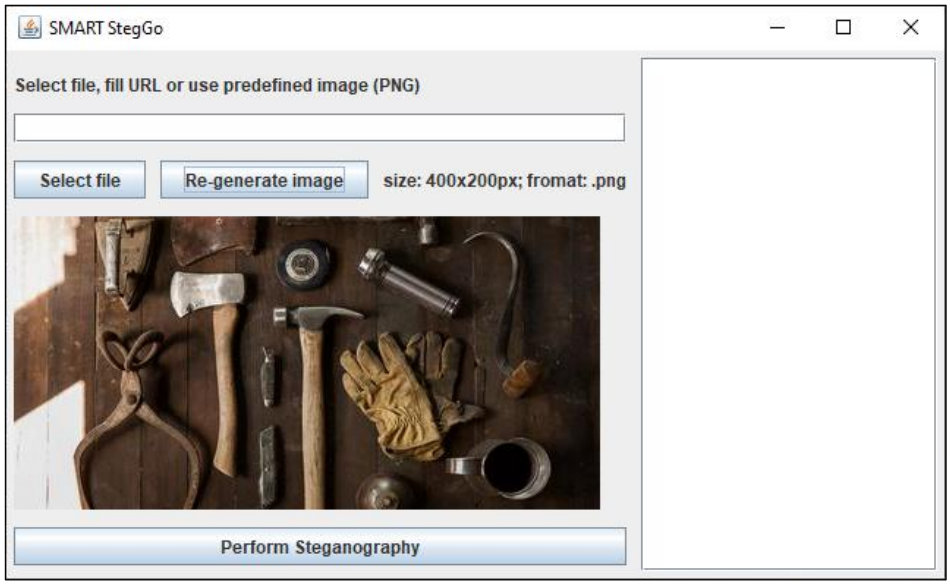
Žádná aplikace, která by se specializovala pouze na obrázkovou steganografii, také nevyhodnocuje použitou metodu na základě vstupních parametrů, jako je například délka vkládané zprávy, či formát obrázku.

12 Navržená aplikace

Navržená aplikace v této práci se specializuje pouze na obrázkovou steganografii. Není tak přehlcená možnostmi, a může tak být minimalistická a jednoduchá z pohledu uživatelského grafického rozhraní a uživatelské přívětivosti. Aplikace implementuje metody LSB (Least Significant Bit), LSB2 (upravená metoda LSB pro ukládání delší informace, využívá dva bity každého Bytu) a DCT (Cosine Discrete Transformation) a volí mezi nimi tu nejvhodnější vzhledem ke vstupům.

Aplikace dokáže sama rozeznat, jestli má z daného obrázku zprávu extrahovat, či ji do obrázku uložit. Tato funkcionalita výrazně ušetří čas uživateli a zjednoduší ovládání aplikace. V praxi to funguje tak, že je-li do aplikace nahrán obrázek, který již má v sobě uloženou zprávu, je tato zpráva automaticky zobrazena. Uživatel tak může snadno do této zprávy něco připsat, smazat ji anebo jen přečíst.

Dalšími prvky, kterými navržená aplikace disponuje jsou tzv placeholder a URL vstup. Placeholder reprezentuje náhodně vygenerovaný obrázek, který může být použitý v případě, že uživatel nemá dostupný žádný jiný. Tato funkcionalita opět šetří uživateli čas. Jestliže se uživateli placeholder nelíbí, může pomocí tlačítka *Re-generate image* (viz Obr. 19) vygenerovat jiný obrázek. URL vstup umožňuje vložit do aplikace obrázek pomocí URL adresy, což opět šetří čas uživatele a přispívá ke zlepšení ovládání aplikace. Uživatel není nucen obrázek zdlouhavě stahovat. Aplikace sama rozezná, jestli zda je zdroj obrázku lokální anebo skrze URL adresu a přizpůsobí tomu své chování.



Obr. 19 Grafické rozhraní aplikace
Zdroj: (Vlastní zpracování)

13 Implementace metody LSB

13.1 Bitové operace

Tato kapitola je inspirovaná publikací *Thinking in Java* od autora Bruce Eckleho. (Eckle 2006)

Bitové operace dovolují vývojáři manipulovat s jednotlivými bity – tedy s nejzákladnější informací v počítačové terminologii. Tyto operace byly představeny s programovacím jazykem C. Jazyk C je nízko úrovňový jazyk, takže se uvažuje častá a přímá manipulace s hardwarem. V takovém případě jsou tyto operace využitelné. Vzhledem k tomu že Java byla původně vyvinutá jako programovací jazyk, který

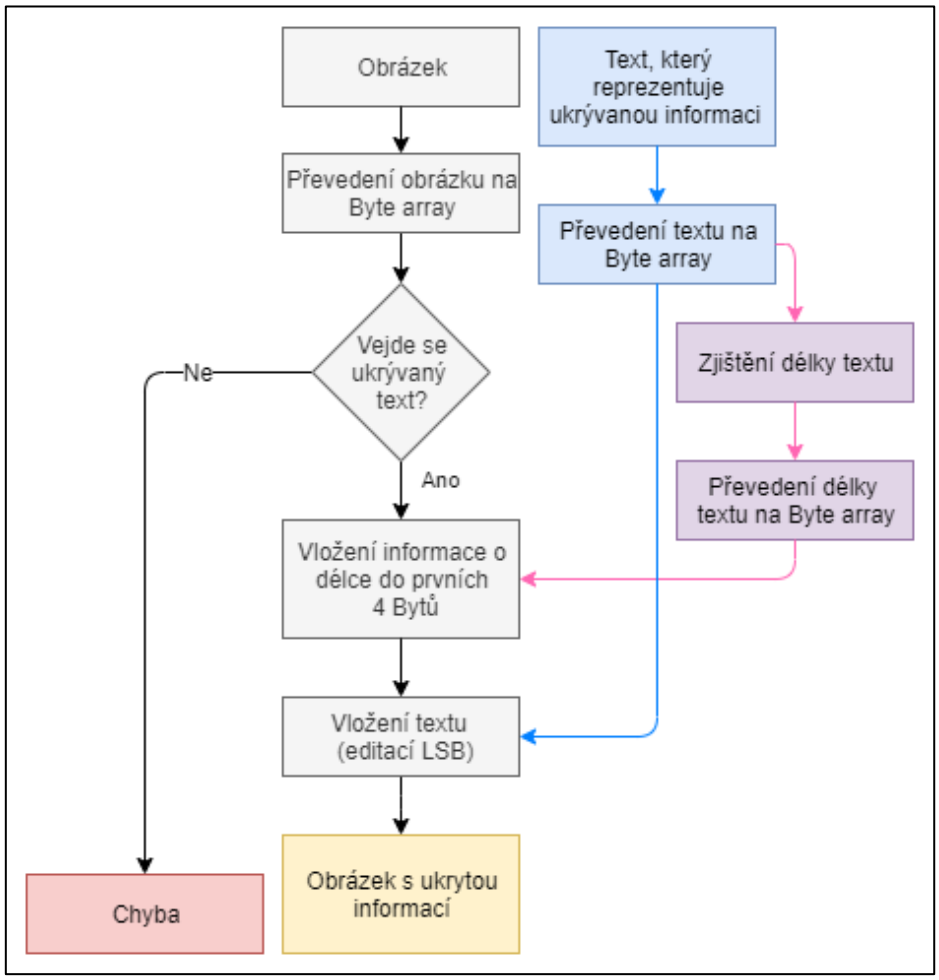
bude v televizních set-top boxech (podrobněji viz kapitola 8.1), dávaly tyto operace smysl, a tak byly do Javy zařazeny také. Následuje seznam těchto operací včetně popisu. (Eckle 2006)

- **AND** (&) — výstup této operace je jedna pouze v případě, že hodnoty na obou vstupech se rovnají jedné. Jinak je výstup nula.
- **OR** (|) — výstup této operace je jedna pouze v případě, že alespoň jedna hodnota se rovná jedné. Výstup je roven nule jen v případě že oba vstupy se rovnají nule.
- **XOR** (^) — výstup této operace je jedna pouze v tom případě že na vstupech je pouze jedna jednička. V případě dvou nul, či dvou jedniček je výstup nula.
- **NOT** (~) — je unární operátor. Má pouze jeden vstup a jeho výstup je negace tohoto vstupu.
- **Aritmetický levý posun** (<<) — tato operace posouvá jednotlivé bity. Má dva vstupy — řetězec bitů (číslo) a číslo reprezentující počet posunů. Aritmetický posuv nemá jasně definováno, jaká hodnota je do krajních pozic vkládána. Posun o jednu pozici vlevo ale reprezentuje násobení dvěma vstupního čísla.
- **Aritmetický pravý posun** (>>) — na rozdíl od aritmetického posunu vlevo, posun vpravo reprezentuje dělení dvěma.
- **Logický levý posun** (<<<) — na rozdíl od přechozích posunů, vždy na krajní hodnoty doplňuje nuly. Levý posun pak doplňuje nuly na nejméně významné pozice.
- **Logický pravý posun** (>>>) — operace přidávající nuly na nejvíce významné pozice.

13.2 Vlastní implementace

Implementace LSB metody je z důvodu minimálního využití výpočetního výkonu v aplikaci realizována pomocí bitových operací AND (&), OR (|) a bitových posunů (>>, <<). Tyto operace jsou detailněji popsány v kapitole 13.1. Celý proces kódování informace se pak skládá z převedení obrázku a ukrývané zprávy na pole Bytů. Dále,

pomocí následující logiky je upraven LSB Bytu obrázku dle bitu v ukrývané zprávě. Kompletní postup vkládání zprávy je znázorněn na Obr. 20.



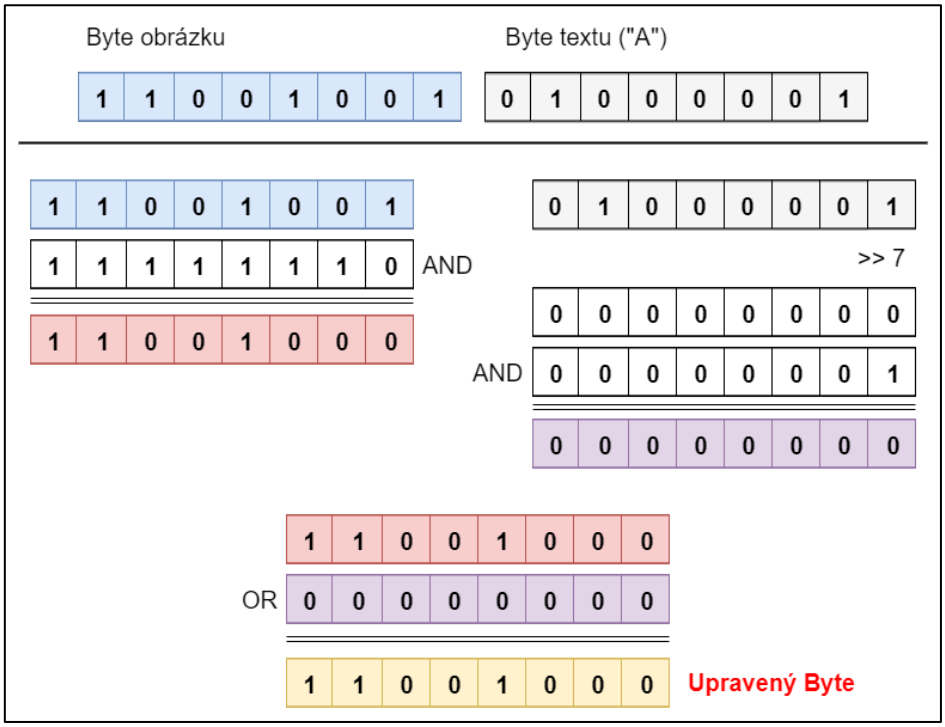
Obr. 20 LSB postup
Zdroj: (Reddy 2011)

Přidání informace do obrázku, tedy editování jednotlivých bitů, má na starosti metoda `addTextToImage()` znázorněná na Kód 3. Tato metoda nejprve zkontroluje, jestli se text do vybraného obrázku (stego média) vejde a až poté vykoná vlastní vkládání. To je realizováno tak, že jsou postupně procházeny Byty ukládané informace (ta byla již předtím převedena na Bytové pole viz Kód 2).

```
public byte[] getBytesFromText(String text) {
    return text.getBytes();
}
```

Kód 2 Byte pole z textu
Zdroj: (Vlastní zpracování)

Každý tento Byte je dále procházen na úrovni bitů a každý tento jeden bit je pomocí bitových operací vložen do jednoho Bytu obrázku, konkrétně do jeho nejméně významného bitu. Jak již bylo zmíněno, toto je realizováno pomocí bitových operací AND, OR a bitového posunu. Pro snadnější porozumění je bitová úprava znázorněna na Obr. 21. Obrázek reprezentuje jeden krok vnitřního cyklu, tedy cyklu, který iteruje skrz jednotlivé bity vkládané zprávy (viz Kód 3). V této metodě je vzhledem k ukládání také délky zprávy využitý až Byte, který je v pořadí 32. Předchozí Byty ukládají délku ukrývané zprávy.



Obr. 21 Bitové operace LSB
Zdroj: (Vlastní zpracování)

```
public BufferedImage addTextToImage(BufferedImage image,
        String message) {

    initialShift = 32;
    byte[] imageByteArray =
        this.utilsImage.getBytesFromImage(image);
    byte[] textByteArray =
        utilsText.getBytesFromText(message);

    if (imageByteArray.length + (initialShift / 8)
        < textByteArray.length) {

        throw new IllegalArgumentException("Image is too small");
    }
    for (byte b : textByteArray) {

        for (int j = 7; j >= 0; j--) {

            initialShift++;
            imageByteArray[initialShift] =
                (byte) ((imageByteArray[initialShift] & 0xFE)
                | ((int) b >>> j) & 1);
        }
    }
    return image;
}
```

Kód 3 LSB vkládání informace
Zdroj: (Wilson 2007)

Na konci tohoto kódu je bit upraven přímo v obrázku. Obrázek je následně uložen a může být dále poslán adresátovi.

Obdobným způsobem je následně zpráva z obrázku vyzvednuta. Aby ale mohla být zpráva dekódována v pořádku, je nutné také uložit její velikost. Jedině tak bude možné získávání zprávy z obrázku včas zastavit a nezobrazovat uživateli také znaky, které již do samotné zprávy nepatří (takové chování může nastat, když není využitý celý payload). Toto je realizováno tak, že se nejprve zjistí délka zprávy, následně se tato hodnota převede do pole Bytů a je vložena před samotnou zprávu. Při dekódování se tak nejprve přečte prvních 32 Bytů obrázku, které drží délku uložené zprávy a až poté se přečte zbytek zprávy. Samotná délka zprávy je uložena ve 4 Bytech. Uložení 4 Bytů však znamená uložit 32 bitů, tady využít 32 Bytů stego média. Díky této informaci může být čtení včas zastaveno. Získávání délky obrázku je znázorněno na Kód 4 metodou `extractTextLength()`, která v parametru

přijímá Bytové pole obrázku, prochází prvních 32 Bytů obrázku (stego média) a získává z nich délku uložené informace.

```
public int extractTextLength(byte[] image) {
    int textLength = 0;
        for (int i = 0; i < 32; i++) {
        textLength = (textLength << 1) | (image[i] & 1);
    }
    return textLength;
}
```

Kód 4 Získávání délky vložené informace
Zdroj: (Vlastní zpracování)

Samotná extrakce informace ze stego média je pak uskutečněna metodou `extractTextFromImage()`. Poté co je inicializovaná délka ukryté informace, tato metoda prochází každý Byte stego média. Nejméně významný bit z tohoto Bytu postupně ukládá do výsledku. Ten je v této metodě prozatím reprezentován také jako Bytové pole. Procházení Bytů stego média a ukládání výsledku je realizováno pomocí dvou vnořených cyklů. První iteruje přes Byty výsledku a druhý to Bytu vkládá bity. Metodu lze vidět na Kód 5.

```
public byte[] extractTextFromImage(BufferedImage image) {

    initialShift = 32;
    byte[] imageByteArray = this.utilsImage
        .getBytesFromImage(image);

    int textLength = extractTextLength(imageByteArray);
    byte[] result = new byte[textLength];

    for (int i = 0; i < result.length; i++) {
        for (int j = 0; j <= 7; j++) {
            initialShift++;
            result[i] = (byte) ((result[i] << 1)
            | (imageByteArray[initialShift] & 1));
        }
    }
    return result;
}
```

Kód 5 Získávání informace LSB
Zdroj: (Wilson 2007)

13.3 Vylepšení metody

Metodu LSB lze několik způsoby vylepšit. Tato vylepšení mohou buď zvýšit míru zabezpečení zprávy, robustnost anebo nenápadnost samotného stego média (obrázku).

13.3.1 Nerovnoměrné rozprostření upravených bitů

Uvažuje-li se standartní implementace metody LSB, pak jsou Byty v obrázku editovány postupně, od začátku. Toto repetitivní chování může vést ke snadnější detekci steganografie. Bylo by tedy vhodné, aby se Byty upravovaly náhodně, napříč celým obrázkem. Toho lze dosáhnout relativně jednoduchým způsobem, s pomocí generátoru pseudonáhodných čísel.

Takový generátor generuje náhodná čísla vzhledem ke vstupu. Právě tato vygenerovaná čísla mohou reprezentovat jednotlivé Byty v obrázku (případně pixely). Důležité je tedy zakódovat do obrázku také vstupní číslo. Po přečtení vstupu se již modifikace Bytů bude řídit podle výsledku implementovaného generátoru pseudonáhodných čísel. Obdobným způsobem lze zprávu také z obrázku získat. Nejprve se přečte vstupní hodnota generátoru a ten následně určí, které Byty byly v obrázku pozměněny. Z těchto Bytů se získá uschovaná informace. (Marwa a spol 2016)

13.3.2 Šifrování

Šifrování ukrývané zprávy může značně zvýšit bezpečnost samotného přenosu. Samo o sobě šifrování ale nepřispívá k nenápadnosti a robustnosti stego média, ale pouze přidává extra vrstvu zabezpečení. Bude-li zpráva ze stego média nějakým způsobem získána neoprávněnou osobou, je stále zašifrována a k jejímu dešifrování je potřeba heslo. Aby mělo šifrování smysl, je nutné předávat heslo jiným komunikačním kanálem. (Gupta,Ankur Goyal a Bhushan 2012)

13.4 Využité knihovny

- File — knihovna, která podporuje úkony se soubory a s cestami, kde se dané soubory nachází.

- ImageIO — obsahuje metody pro práci s načítáním a ukládáním obrázkových souborů.
- BufferedImage — tato třída rozšiřuje třídu Image. BufferedImage, reprezentuje barevný a rastrový model obrázkových dat.
- IOException — knihovna, která pomáhá signalizovat výjimky související se vstupními, nebo výstupními operacemi.

Popisy tříd jsou převzaty z oficiální dokumentace. (Oracle 2020)

14 Implementace metody LSB2

Tato metoda vychází z klasické metody LSB zmíněné v kapitole 13. Rozdíl je, že nevyužívá pouze jeden bit každého Bytu, ale využívá bity dva. To má za výsledek zhruba dvojnásobné zvětšení payloadu. Může se však stát, že na některých obrázcích (stego médiích) bude tato úprava viditelná a barvy nebudou příliš realistické.

```
public BufferedImage addTextToImage(BufferedImage image,
    String message) {

    initialShift = 16;
    byte[] imageByteArray =
        this.utilsImage.getBytesFromImage(image);
    byte[] textByteArray =
        utilsText.getBytesFromText(message);

    if (imageByteArray.length + (initialShift / 8) <
        textByteArray.length) {

        throw new IllegalArgumentException("Image is too small for
        the text");
    }

    for (byte b : textByteArray) {
        for (int j = 6; j >= 0; j=j-2) {
            initialShift++;
            imageByteArray[initialShift] =
                    (byte)((imageByteArray[initialShift] & 0xFC)
                    | ((int) b >>> j) & 3);
        }
    }
    return image;
}
```

Kód 6 LSB2 vkládání informace
Zdroj: (Vlastní zpracování)

Samotná implementace je provedena velice obdobným způsobem jako metoda LSB. Hlavní rozdíly jsou v bitových operacích. Ty musely být upraveny tak, že není modifikovaný pouze nejméně významný bit, ale také druhý nejméně významný bit. Upravenou metodu `addTextToImage()`, pro vkládání informace lze viděn na Kód 6.

Vzhledem k tomu, že jsou využívané dva bity v jednom Bytu, zabere uložení délky zprávy pouze 16 Bytů ve stego médiu. Přičemž velikost informace o délce ukrývaného textu zůstane zakódována ve čtyřech Bytech. Z toho důvodu je zde počáteční posunutí rovno šestnácti (u metody LSB to je 32).

Extrakce informace je pak také velice podobná extrakci v metodě LSB. Opět jsou jen upraveny bitové operace. Tyto úpravy jsou viditelné na Kód 7.

```
public byte[] extractTextFromImage(BufferedImage image) {
    initialShift = 16;
    byte[] imageByteArray =
        this.utilsImage.getBytesFromImage(image);

    int textLength = extractTextLength(imageByteArray);
    byte[] result = new byte[textLength];

    for (int i = 0; i < result.length; i++) {
        for (int j = 0; j <= 6; j=j+2) {
            initialShift++;
            result[i] = (byte) ((result[i] << 2) |
            (imageByteArray[initialShift] & 3));
        }
    }
    return result;
}
```

Kód 7 Získání informace LSB2
Zdroj: (Vlastní zpracování)

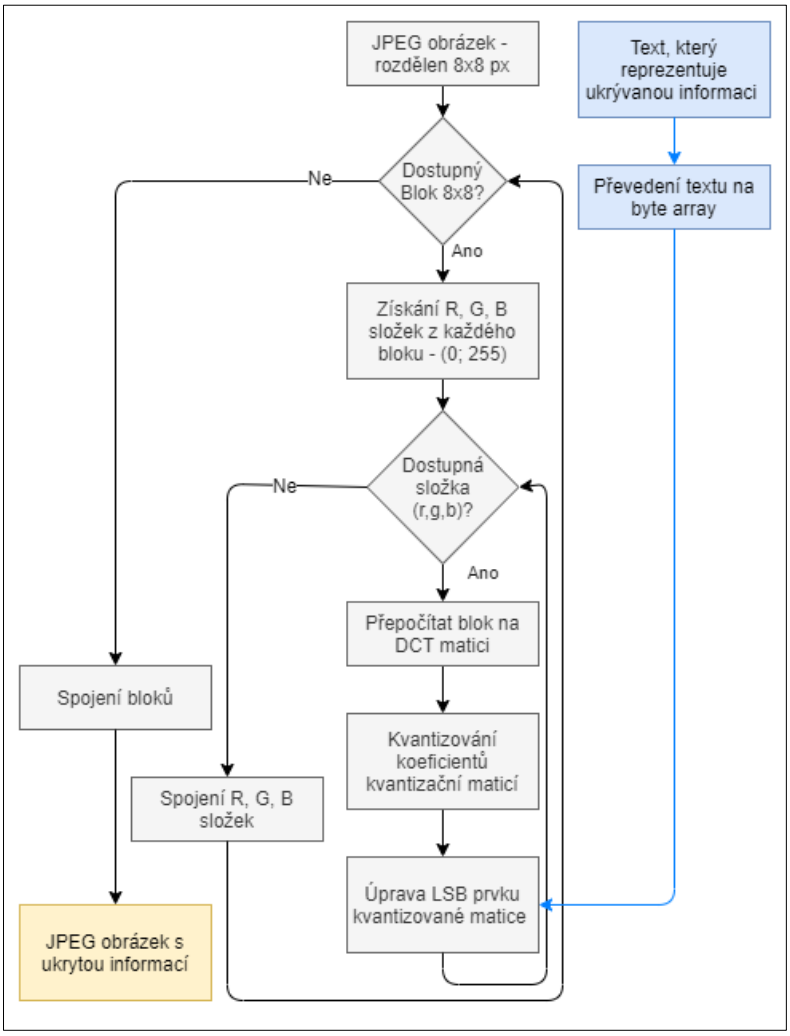
14.1 Využité knihovny

Knihovny využité pro implementaci této metody se plně shodují s knihovnami, které byly použité v implementaci metody LSB (kapitola 13.4).

15 Implementace metody DCT

Implementování DCT metody je složitější. Jelikož k uchování informace (zprávy) se využívá prostor transformací při ztrátové .jpeg kompresi, je tedy nutné uložit zprávu do obrázku v průběhu .jpeg kódování. V tomto projektu je použit enkodér

naprogramovaný Jamesem R. v roce 1998. Tento enkodér, byl upravený tak, že kvantizované koeficienty matice dále upravuje podle bitů v ukryté zprávě. Tím vzniká ukládání utajené zprávy. Vzhledem k tomu, že v tomto projektu je prioritní nenápadné uložení zprávy a nikoli redukce velikosti souboru, je nastavena maximální kvalita obrázku (100 %). Aby bylo možné zprávu dešifrovat, musel být implementován také dekodér, který opačným způsobem z prostoru transformací zprávu získá. V projektu byl použit .jpeg dekodér od autora Seana Breslina z roku 1997. Oba tyto kódy mohou být volně použity a upraveny na základě licenčních podmínek. Grafické znázornění průběhu DCT metody lze vidět na Obr. 22.



Obr. 22 DCT postup
Zdroj: (Wahab a spol 2019)

Samotný procese úpravy koeficientů kvantizované matice je znázorněný v na následujících ukázkách kódů (Kód 8, Kód 9) . Tyto úryvky kódů byly použity v

metodě `WriteCompressedData()` v Jpeg enkodéru. Tato metoda byla rozšířena o parametr přijímající ukrývanou zprávu.

Nejprve jsou inicializovány následující proměnné, které pomáhají uskutečnit změnu bitů.

```
int messageBytes, tmpByte, tmpBit;

messageBytes = message.length;
tmpByte = 0;
tmpBit = 7;

byte currentByte = messageBytes > 0 ? message[0] : (byte) 0;
```

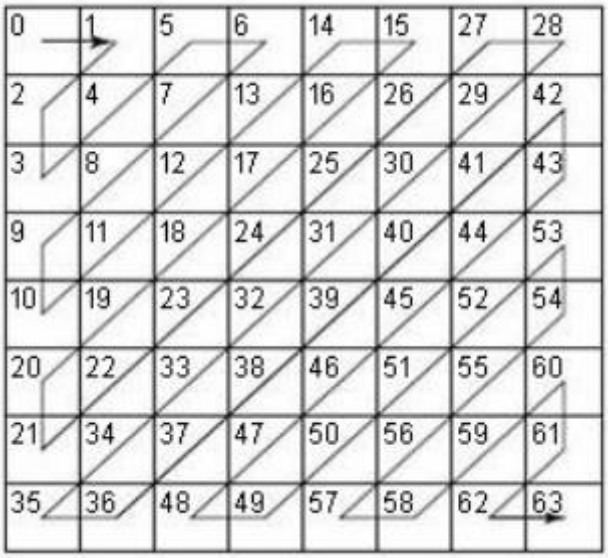
Kód 8 DCT – inicializace proměnných
Zdroj: (Vlastní zpracování)

Po této inicializaci je v několika cyklech realizovaná komprese bloků pixelů 8x8. Poté, co je získána kvantizovaná matice koeficientů, je prvek na pozici 23 upraven tak, aby uschoval jeden bit ukládané zprávy. Samotné vložení je uskutečněno několika podmínkami a znázorněno na Kód 1Kód 9.

```
if (tmpByte < messageBytes) {
    int bit = (currentByte >> tmpBit) & 1;
    tmpBit--;
    if (tmpBit == -1) {
        tmpBit = 7;
        tmpByte++;
        if (tmpByte < messageBytes) {
            currentByte = message[tmpByte];
        }
    }
    dctArray3[23] = (dctArray3[23] & 0xfffffffe) | bit;
}
```

Kód 9 Upravení kvantizované matice
Zdroj: (Vlastní zpracování)

Prvkem na pozici 23 je prvek odpovídají hodnotě [2][4] v kvantizované matici, při získávání dat pomocí Zig Zag vzoru. Pozice prvku je znázorněná na Obr. 23. (Mastriani a Gambini 2010)



Obr. 23 Zig Zag vzor
Zdroj: (Mastriani a Gambini 2010)

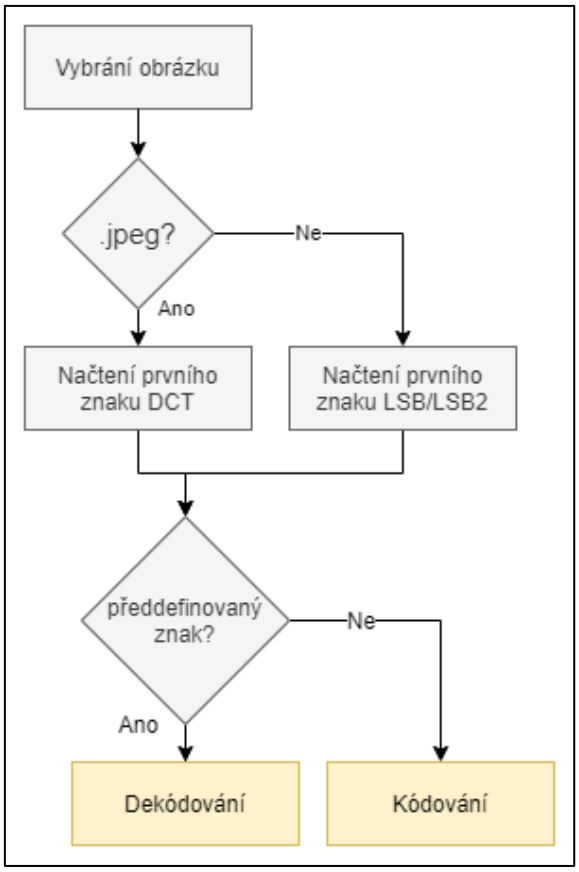
15.1 Využité knihovny

- JpegEncoder — JPEG enkodér, který zajišťuje převod obrázku do .jpeg formátu. Pro potřeby této steganografie, byl upraven.
- JpegDecoder — JPEG dekodér, která zajišťuje zobrazení .jpeg formátu. Pro potřeby této steganografie, byl upraven.
- InputStream — Nadřazená třída všem třídám, které pracují se streamy (proudy) Bytů.

16 Implementace adaptivního rozhodování

Samotná logika rozhodování, jakou metodu pro ukrytí zprávy použít, je pak popsána diagramy na Obr. 24 a Obr. 25.

V prvním rozhodovacím uzlu se vzhledem k formátu obrázku zvolí typ metody, pomocí které se přečte první znak případné uložené zprávy. Jestliže tento znak se rovná znaku předdefinovanému, je rozhodnuto, že se obrázek bude dekódovat (zpráva již byla zakódována). Provede se tak dekódování a uživateli je zobrazena ukrytá zpráva. V opačném případě aplikace čeká na zadání textu, který bude do obrázku vložen.



Obr. 24 První rozhodovací uzel
Zdroj: (Vlastní zpracování)

Na Kód 10 je znázorněna implementace Obr. 24. První metoda `decodeMethod()` určuje, pomocí jaké metody bude vyjmut první znak uložené zprávy. Toto rozhodování je určeno podle typu souboru.

```
public int decodeMethod(String path) {
    //LSB or LSB2
    int method = 0;
    if (utilsGeneral.isImageLoadedFromURL(path)) {
        if (utilsImage.getImageTypeURL(path)==1) {
            //DCT
            method = 1;
        }
    } else {
        if (isFileJpg(path)) {
            //DCT
            method = 1;
        }
    }
    return method
}
```

Kód 10 Metoda pro vyjmutí prvního znaku
Zdroj: (Vlastní zpracování)

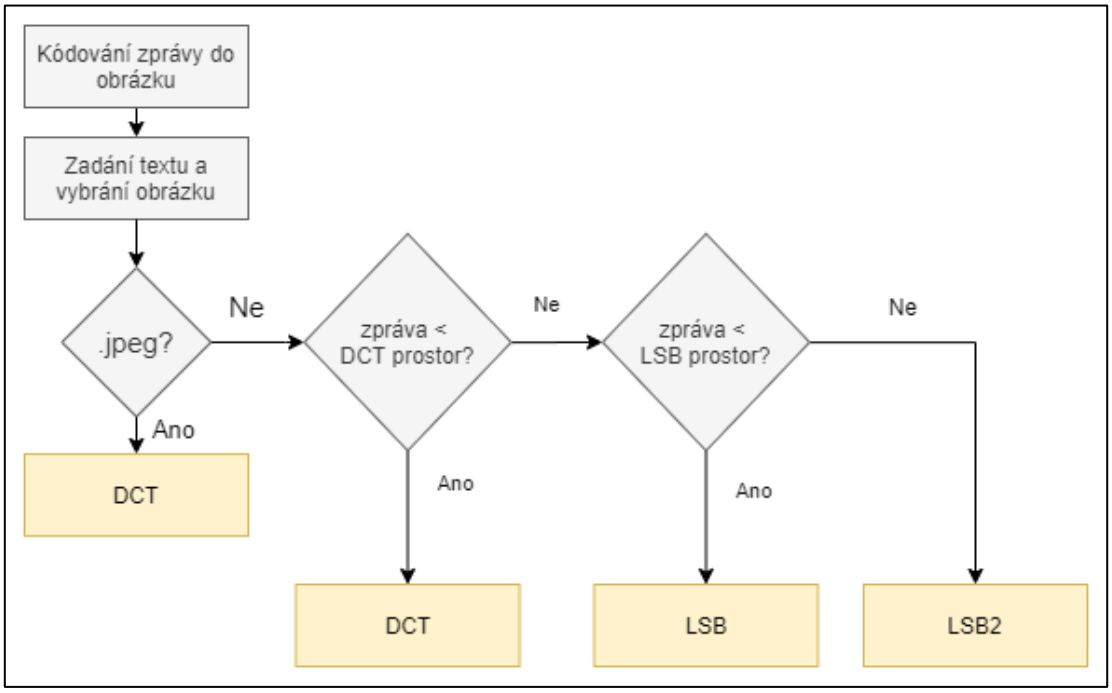
Následuje metoda `decideCodeOrDecode()` (Kód 11), která se pokusí získat první znak z uryté zprávy. Podle tohoto znaku aplikace pozná, nejen jestli je v obrázku zpráva uložena, ale také pomocí jaké metody byla zpráva uložena. Toto je důležité především pro rozpoznávání mezi metodou LSB a LSB2.

```
public String decideCodeOrDecode(String pathString) throws IOException
{
    int decodeMethod = decodeMethod(pathString);
    String firstChar = "";

    if (utilsGeneral.isImageLoadedFromURL(pathString) ||
pathString.isEmpty()) {
        return "code";
    }
    if (decodeMethod == 0) {
        firstChar = cryptoLSB.decodeFirstChar(pathString);
        if (!firstChar.equals(startChar)) {
            firstChar = cryptoLSB2.decodeFirstChar(pathString);
            if (firstChar.equals(startChar)) {
                lsb2 = true;
            }
        }
    } else {
        File file = new File(String.valueOf(pathString));
        FileInputStream fileInputStream = new FileInputStream(file);
        int[] coefficients =
cryptoDCT.extractCoefficients(fileInputStream, (int) file.length());
        int[] bitArray =
cryptoDCT.extractLSBFromCoefficientsMessageFirstChar(coefficients);
        firstChar =
cryptoDCT.decodeDCT(cryptoDCT.getByteArrayDCT(bitArray));
    }

    if (firstChar.equals(startChar) ) {
        return "decode";
    } else {
        return "code";
    }
}
```

Kód 11 Rozhodování kódování/dekódování
Zdroj: (Vlastní zpracování)

V druhém rozhodovacím uzlu se zjistí, jestli je obrázek ve formátu .jpeg. Jestliže ano, je automaticky použita metoda DCT. Důvod proč zde nemůže být použita metoda LSB je ten, že následující uložení souboru zpět do .jpeg formátu by vzhledem ke ztrátové kompresy zcela zničilo vloženou zprávu. Jediným řešením by v této situaci bylo uložit obrázek například ve formátu .png, tedy pomocí bezztrátové komprese. Tyto formáty jsou ale velice náročné na velikost a v případě použití

velikého .jpeg souboru by mohl dosahovat nadměrné velikosti. Jestliže je obrázek v jiném formátu, zvolení metody je určeno podle toho, jak velká je zpráva. Metoda LSB tak bude použita pouze v tom případě, že payload pro daný obrázek při použití metody DCT není dostatečný. Důvodem, proč je preferovaná metoda DCT je její robustnost. Vzhledem k publikaci zveřejněné v roce 2010 od autorů Dr. Ekta Walia, Payals Jaina a Navdeepa Navdeepa. Tato publikace nesoucí název *An Analysis of LSB & DCT based Steganography* porovnává výsledky dosažené při použití obou metod dle takzvaného PSNR (Peak signal-to-noise ratio) poměru. Při tomto porovnávání vychází metoda DCT mnohem lépe než metoda LSB. (Walia a spol 2010)



Obr. 25 Druhý rozhodovací uzel
Zdroj: (Vlastní zpracování)

Toto rozhodování bylo implementováno metodou `codeMethod()`. Tato metoda je zobrazena na následujícím úryvku kódu (Kód 12). Nejprve jsou vypočítány dostupné payloady pro jednotlivé metody. Tyto payloady jsou následně porovnány s délkou ukládané zprávy. Na základě této logiky je vybrána vhodná metoda.

```
public int codeMethod(BufferedImage sourceImage, String message) {
    int charCapacityLSB, charCapacityDCT, charCapacityLSB2;

    charCapacityLSB = (sourceImage.getHeight() *
sourceImage.getWidth()) / 8 - 5;
    charCapacityLSB2 = (sourceImage.getHeight() *
sourceImage.getWidth()) / 4 -5;

    double height = sourceImage.getHeight();
    int intHeight = (int) Math.round(height / 8);
    double width = sourceImage.getWidth();
    int intWidth = (int) Math.round(width / 8);
    charCapacityDCT = ((intHeight * intWidth) * 3) / 8 - 2;

    if (message.length() > charCapacityLSB2) {
        return 100;
    } else if (message.length() > charCapacityLSB) {
        //use LSB2
        return 3;
    } else if (message.length() > charCapacityDCT) {
        //use LSB
        return 0;
    } else {
        //use DCT
        return 1;
    }
}
```

Kód 12 Výběr kódovací metody
Zdroj: (Vlastní zpracování)

Třetím plánovaným rozhodovacím uzlem bylo rozhodování o výběru použité metody na základě dostupného výpočetního výkonu. Nicméně po otestování obou metod se ukázalo, že rozdíl mezi metodami není významný, a tak tento rozhodovací uzel implementován nebyl. Toto může být spatřeno v kapitole 19 na Tab. 5 a Tab. 6.

17 Implementace URL vstupu

Pro podporu uživatelské přívětivosti bylo implementováno také vkládání informace do obrázku, který byl nahrán pouze pomocí URL adresy. Odpadne tak nutnost pokaždé obrázek stahovat na lokální úložiště a celkově se tak zrychlí proces vkládání informace. Poté co je do takového obrázku informace nahrána, je stego médium uloženo na lokální úložiště.

Rozhodování jestli má být obrázek načtený z lokálního úložiště anebo URL adresy má na starosti metoda `isImageloadeFromURL()` (Kód 13). Tato metoda

jednoduše sleduje, jaká cesta k obrázku je zadaná a jestliže je cesta ve formátu URL adresy vrátí metoda hodnotu true (pravda).

```
public boolean isImageLoadedFromURL(String pathString) {
    if (pathString.contains("https://") ||
                pathString.contains("http://") ||
            pathString.contains("www.") ||
            pathString.contains(".com") ||
            pathString.contains(".cz")) {
        return true;
    } else {
        return false;
    }
}
```

Kód 13 Rozhodování zda je vstup URL adresa
Zdroj: (Vlastní zpracování)

Jestliže má být obrázek načtený pomocí URL adresy, pak proces vložení zprávy je následující. Nejprve je ověřena URL adresa, je-li dostupná, obrázek se uloží na lokální úložiště a modifikuje se (vloží se zpráva). Nahrání obrázku z URL adresy má na starosti metoda `readImageURL()` (Kód 14 Načtení obrázku z URL).

```
public BufferedImage readImageURL(String urlPath) {
    BufferedImage bufferedImage;
    try {
        URL url = new URL(urlPath);
        bufferedImage = ImageIO.read(url);
        return bufferedImage;
    } catch (IOException e) {
        e.printStackTrace();
        return null;
    }
}
```

Kód 14 Načtení obrázku z URL
Zdroj: (Vlastní zpracování)

18 Implementace placeholderu

Pod pojmem placeholder se v této práci (aplikaci) rozumí náhodně vygenerovaný obrázek, který může sloužit jako stego médium v případě, že uživatel nedisponuje žádným použitelným obrázkem anebo potřebuje steganografii aplikovat rychle. Pro tuto funkcionalitu je využitá služba *Lorem Picsum* od autorů Davida Marbyho a Nijika Yonskaiiho, která generuje náhodně obrázky po zadání url adresy

(Yonskai a Marby 2020). Samotná implementace je znázorněna na Kód 15 Načtení placeholderu.

```
private String path = "https://picsum.photos/400/200";

placeholderBufferedImage = utilsImage.readImageURL(path);
ImageIcon placeHolder = new ImageIcon(placeholderBufferedImage);
iconLabel.setIcon(placeHolder);
```

Kód 15 Načtení placeholderu
Zdroj: (Vlastní zpracování)

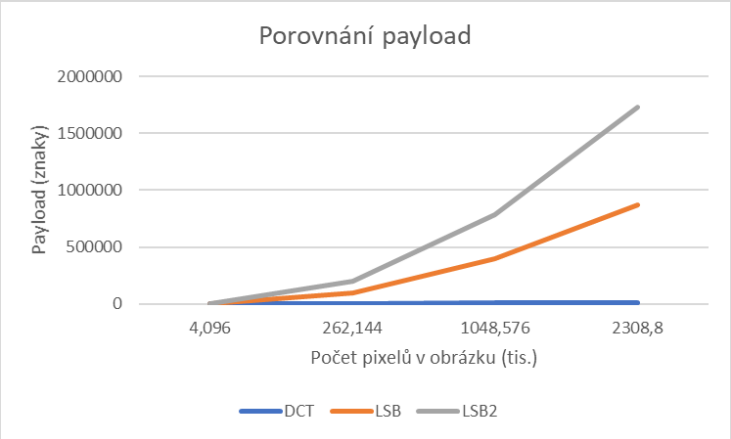
19 Porovnání výsledků metod

Prvním testovacím parametrem aplikace je payload. V Tab. 4 je uvedena délka textu, který může být vložen pomocí metod LSB a DCT do obrázku dané velikosti. Velikost obrázku je udávaná v pixelech a payload zprávy reprezentuje počet znaků.

Formát	Velikost	Payload DCT	Payload LSB	Payload LSB2
.png	64x64	22	1 521	3 057
.png	512x512	1 534	98 289	196 593
.png	1024x1024	6 142	393 201	786 417
.png	1924x1200	13 498	866 985	1 731 585

Tab. 4 Payload obrázků
Zdroj: (Vlastní zpracování)

Následující graf (Obr. 26) nabízí ucelený pohled na rozdíl payloadu mezi metodou LSB, LSB2 a metodou DCT. Je viditelný nepoměr payloadu mezi metodami LSB a metodou DCT.



Obr. 26 Porovnání payload
Zdroj: (Vlastní zpracování)

Na Tab. 5 a Tab. 6. je zobrazena časová náročnost ukládání a dekódování zpráv při použití jednotlivých steganografických metod.

Id	Formát	Velikost obrázku	Délka zprávy	DCT (ms)	LSB (ms)	LSB2 (ms)
1	.png	64x64	5	14	12	12
1	.png	64x64	20	14	12	12
1	.png	64x64	500	---	12	12
1	.png	64x64	1 000	---	12	13
1	.png	64x64	3 000	---	---	13
2	.png	512x512	500	81	54	54
2	.png	512x512	1 500	93	64	63
2	.png	512x512	65 500	---	71	85
2	.png	512x512	195 000	---	---	80
3	.png	1024x1024	2 000	173	180	176
3	.png	1024x1024	6 000	202	191	176
3	.png	1024x1024	260 000	---	230	331
3	.png	1024x1024	780 000	---	---	330
4	.png	1924x1200	5 000	357	379	376
4	.png	1924x1200	13 400	384	380	406
4	.png	1924x1200	575 000	---	424	444
4	.png	1924x1200	1 700 000	---	---	530

Tab. 5 Časová náročnost ukládání zprávy
Zdroj: (Vlastní zpracování)

Id	Formát	Velikost obrázku	Délka zprávy	DCT (ms)	LSB (ms)	LSB2 (ms)
1	.png	64x64	5	74	< 0	< 0
1	.png	64x64	20	67	< 0	< 0
1	.png	64x64	500	---	< 0	< 0
1	.png	64x64	1 000	---	< 0	< 0
1	.png	64x64	3 000	---	---	1
2	.png	512x512	500	114	12	11
2	.png	512x512	1 500	125	12	12
2	.png	512x512	65 500	---	16	17
2	.png	512x512	195 000	---	---	17
3	.png	1024x1024	2 000	121	41	32
3	.png	1024x1024	6 000	161	35	33
3	.png	1024x1024	260 000	---	41	41
3	.png	1024x1024	780 000	---	---	61
4	.png	1924x1200	5 000	199	86	83
4	.png	1924x1200	13 400	299	89	92
4	.png	1924x1200	575 000	---	114	89

4	.png	1924x1200	1 700 000	---	---	162

Tab. 6 Časová náročnost dekódování zprávy
Zdroj: (Vlastní zpracování)

Následuje porovnání obrázků před vložením skryté zprávy a po vložení zprávy. V popisku obrázku je vždy uvedeno id z předchozích tabulek 2 a 3. Lze tak zjistit jejich původní velikost a délku uložené zprávy.

19.1 Vizuální porovnání

Na následujících obrázkách (Obr. 27, Obr. 28, Obr. 29) je možné vizuálně porovnat obrázky před a po aplikování steganografie. Každé porovnání je také doplněno o použitou metodu steganografie a počet vložených znaků. Indexy obrázků korespondují s Tab. 5 a Tab. 6.

Obrázek s indexem 4, použitá metoda DCT, uloženo 13 400 znaků (obrázek byl pro lepší zobrazení zmenšen):

Obr. 27 Vizuální porovnání I
Zdroj: (WallHaven 2020)

Obrázek s indexem 2, použitá metoda LSB, uloženo 32 000 znaků (obrázek byl pro lepší zobrazení zmenšen):

Obr. 28 Vizuální porovnání II
Zdroj: (USC 1973)

Obrázek s indexem 1, použitá metoda LSB2, uloženo 1 010 znaků (obrázek byl pro lepší zobrazení 2x zvětšen):

Obr. 29 Vizuální porovnání III
Zdroj: (Yonskai a Marby 2020)

19.2 Metoda PSNR

Peak signal-to-noise ratio (špičkový poměr signálu k šumu) je název poměru, který porovnává obrázky vzhledem k maximální hodnotou signálu a šumu. Jeho výpočet je uskutečněn pomocí následujícího vzorce:

$PSNR = 10 * log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 * log_{10}(\frac{MAX_I}{\sqrt{MSE}})$ kde:

$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}|I(i,j) - K(i,j)|^2$

$MAX_I = maximální\ hodnota\ pixelu\ obrázku$

Budeme-li takto uvažovat pak pro dva naprosto shodné obrázky by hodnota MSE měla být rovna 0. Hodnota PSNR poměru, pak bude dosahovat nekonečna. Je tedy zřejmé, že čím vyšší hodnota PSNR, tím větší kvality obrázky dosahují (tím spíše jsou identické). Komprimované obrázky při porovnání se svojí nekomprimovanou variantou běžně dosahují hodnot 30–50 dB (Barni 2018). Pro představu přijatelná hodnota pro ztrátu kvality během bezdrátové komunikace se pohybuje mezi 20-25 dB (Thomos a spol 2006). Zjednodušeně lze tedy říct, že tento poměr udává, jak moc jsou obrázky odlišné.

Následuje porovnání obrázků před a po aplikování steganografie. Pro tuto potřebu byla použita aplikace, která se jmenuje PSNR, je šířena jako freeware vydavatelem GIPSA-lab a naprogramována byla Pascalem Bertolinim (Bertolino 2020). Pro zvýšení věrohodnosti byly vždy porovnávány stejných formátů. Případná komprese by totiž mohla také ovlivnit PSNR poměr.

Id	Formát	Velikost obrázku	Délka zprávy	DCT (ms)	LSB (ms)	LSB2 (ms)	PSNR
1	.jpeg	64x64	20	✓	×	×	60,47
1	.png	64x64	1 000	×	✓	×	51,61
1	.png	64x64	3 000	×	×	✓	45,99
2	.jpeg	512x512	1 500	✓	×	×	56,76
2	.png	512x512	65 500	×	✓	×	56,82
2	.png	512x512	195 000	×	×	✓	48,39
3	.jpeg	1024x1024	6 000	✓	×	×	58,01
3	.png	1024x1024	260 000	×	✓	×	56,51
3	.png	1024x1024	700 000	×	×	✓	44,84
4	.jpeg	1924x1200	13 400	✓	×	×	56,85
4	.png	1924x1200	575 000	×	✓	×	56,64
4	.png	1924x1200	1 700 000	×	×	✓	48,51

Tab. 7 PSNR porovnání obrázků
Zdroj: (Vlastní zpracování)

Na hodnotách v Tab. 7 si lze všimnou jasných výsledků. Nejmenší hodnotu PSNR poměru mají obrázky, do kterých byla vložena informace pomocí LSB2 metody. Nejlepších výsledků (největší PSNM poměr) mají obrázky, modifikované metodou DCT. Dále by bylo vhodné také zdůraznit, že čím je obrázek menší, tím jsou rozdíly, vzhledem k PSNR poměru větší. Naopak u velkých obrázků (1924x1200px) se rozdíly ztrácí.

19.3 Metoda SSIM

Structural similarity (strukturální podobnost) je název indexu určujícího kvalitu digitálního obrazu. Rozsah tohoto indexu je -1–1, kde 1 znamená, že jsou dva obrázky totožné. Tato metoda byla vytvořena Zhou Wangem a jeho týmem a je považována za metodu, která reflektuje vnímaní kvality lidského zrakového systému (Zhou Wang a spol 2004). Místo používání tradičních sčítacích metod (jako například využívá MSE) se SSIM zaměřuje na kombinací tří faktorů. Tyto faktory jsou: ztráta korelace, zkreslení jasu a zkreslení kontrastu. (Horé a Ziou 2010)

Na Tab. 8 lze vidět hodnoty SSIM indexu po vložení informace vzhledem k originálnímu obrázku. Pro vzájemné porovnání obrázků byl použit software Imatest od společnosti Imatest (Imatest 2020). Pro zvýšení věrohodnosti byly vždy porovnávány stejné formáty.

Id	Formát	Velikost obrázku	Délka zprávy	DCT (ms)	LSB (ms)	LSB2 (ms)	SSIM
1	.jpeg	64x64	20	✓	×	×	0,9998
1	.png	64x64	1 000	×	✓	×	0,998
1	.png	64x64	3 000	×	×	✓	0,9939
2	.jpeg	512x512	1 500	✓	×	×	0,9981
2	.png	512x512	65 500	×	✓	×	0,9975
2	.png	512x512	195 000	×	×	✓	0,9887
3	.jpeg	1024x1024	6 000	✓	×	×	0,9989
3	.png	1024x1024	260 000	×	✓	×	0,9955
3	.png	1024x1024	700 000	×	×	✓	0,9875
4	.jpeg	1924x1200	13 400	✓	×	×	0,9985
4	.png	1924x1200	575 000	×	✓	×	0.9964
4	.png	1924x1200	1 700 000	×	×	✓	0.9898

Tab. 8 SSIM porovnání obrázků
Zdroj: (vlastní zpracování)

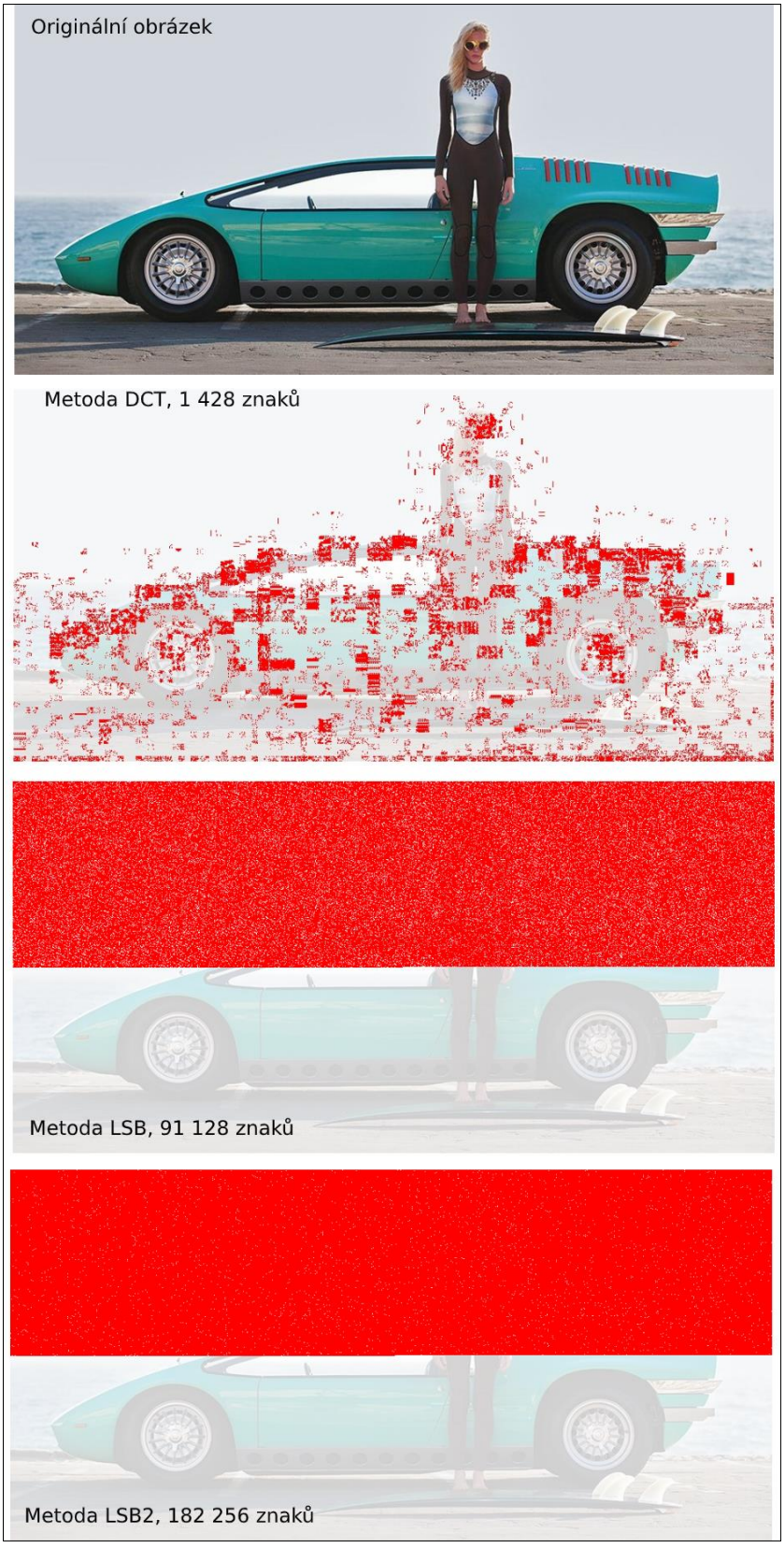
Lze si všimnout, že SSIM hodnoty jsou poměrně vysoké. To značí o tom, že z vizuálního hlediska si jsou obrázky velice podobné. To je v případě steganografie stěžejní.

19.4 Softwarové porovnání pixelů

Toto porovnávání využívá software, který porovnává hodnoty jednotlivých pixelů dvou obrázků. Celý proces tohoto porovnávání funguje tak, že se vezme pixel na

pozici [X, Y] jednoho obrázku a je porovnán s pixelem na pozici [X, Y] druhého obrázku. Jestliže se pixely liší, je tento pixel zvýrazněn. Vzhledem k tomu, že porovnávání vykonává program, který pixely převádí na celá čísla, je určení nepřesností naprosto přesné. Nevýhodou takového porovnávání je, že se musí porovnávat na pixel stejně veliké obrázky.

Pro následující porovnání (Obr. 30) byl využit online software *Online Image Diff* dostupný pod doménou www.online-image-comparison.com (WaKi Software 2020). Do stejného obrázku byla vložena zpráva o délce 50 % payloadu a následně porovnána s originálem.



Obr. 30 Porovnání pixelů
Zdroj: (WaKi Software 2020, DriveTribe 2016)

20 Možná rozšíření aplikace

Prvním a nejspíše nejužitečnějším rozšířením, by bylo přidání více metod pro aplikaci obrázkové steganografie, případně další modifikace metod stávajících. S každou přidanou metodou je však nutné také brát v potaz přidání dalšího uzlu rozhodovacího stromu. S tím souvisí také definování, kdy má být metoda použita. Toto rozhodování může být na základě payloadu, dostupného výpočetního výkonu anebo jiného kritéria. Další metoda by mohla být například DWT (Discrete Wavelet Transform).

Šifrování vkládaných dat by zcela určitě také zajistilo větší bezpečnost přenosu. Samotné šifrování by proběhlo ještě před samotným vložením dat do obrázku. Data by se vložila až po zašifrování v zašifrované, nečitelné, formě. Jako šifrovací algoritmus by mohl být použit například algoritmus BlowFish. Vzhledem k tomu, že tato funkcionalita není pro potřeby této práce stěžejní, nebylo šifrování implementováno.

Vzhledem k tomu, že ovládání aplikace je navrženo tak, aby bylo co nejjednodušší a práci s ní tak zvládl i méně technicky znalý uživatel, bylo by také dobré, aby v budoucnosti byla sjednocena a upravena vizuální stránka aplikace. Atraktivní vizuál by mohl pomoci budoucím uživatelům si aplikaci oblíbit a zpříjemnit její používání.

V neposlední řadě by také bylo žádoucí, aby aplikace byla dostupná jako webová služba případně mobilní aplikace. Implementování webové služby by zvýšilo dostupnost aplikace. Předpoklad je takový, že když nebude nutné aplikaci stahovat a až poté spouštět, ale bude stačit vstoupit na URL adresu, zvýší se počet aktivních uživatelů. V případě mobilní aplikace by se mohl využívat fotoaparát mobilního telefonu a zpráva by se mohla vkládat přímo do pořízené fotografie. To by zcela určitě také kladně přispělo k uživatelské přívětivosti a zrychlilo celý proces posílání zamaskované zprávy. Jak webová, tak mobilní aplikace by mohla implementovat stejnou logiku pro určení steganografické metody, jako aplikace vyvinuta v rámci této práce.

21 Závěry a doporučení

V této práci se podařilo navrhnout a naprogramovat aplikaci pro obrázkovou steganografii. Tato aplikace implementována v programovacím jazyce Java chytře a aktivně, na základě zadaných parametrů určuje, jaká implementovaná steganografická metoda bude pro uschování informace do obrázku použita. Aplikace obsahuje metodu LSB, LSB2 (modifikovaný ekvivalent metody LSB) a metodu DCT. Zatím co metody LSB vkládají zprávu do nekomprimovaných obrázkových souborů, metoda DCT vkládá informace do .jpeg souborů. Jako rozhodovací parametry jsou použity formát obrázkového souboru a velikost ukládané zprávy. Vzhledem k tomu, že aplikace klade důraz na jednoduché uživatelské rozhraní a sama zjišťuje, jaká metoda je nejvhodnější, najde uplatnění i u méně znalých uživatelů.

Každá metoda byla také podrobena několika testům, ve kterých se sledovalo, jak moc se liší obrázek před a po vložení tajné informace. Při těchto testech se také uvažovala maximální velikost vkládané informace. Právě maximální velikost vkládané informace je napříč metodami velice rozdílná. Například pro obrázek s rozlišením 1924x1200px (full HD rozlišení) může být pomocí metody DCT vloženo 13 498 znaků, zatím co pomocí metody LSB2 to je 1 731 585 znaků. Klade-li se důraz pouze na to, aby byl obrázek co nejvíce nenápadný a co nejvíce podobný originálnímu obrázku, vychází metoda DCT ve všech testech nejlépe.

Potencionálních využití v praxi je celá škála, například v medicíně. Pomocí steganografické aplikace by se mohla jednoduše spárovat data o pacientovi, navrhované léčbě či o provedených vyšetřeních s fotografií pacienta anebo fotografií nálezu. Zamezilo by se tak ztrátám dat. Více o tomto využití je napsáno v kapitole 5.3.2. Dalšími obory, ve kterých by tato práce našla uplatnění, jsou například zpravodajství, stavebnictví případně bezpečnostní složky. Z tohoto důvodu je potenciál steganografie i v dnešní době velký a tato práce najde uplatnění v různých odvětvích.

22 Seznam použité literatury

[1] AL-ANI, Zaidoon Kh, A A ZAIDAN, B B ZAIDAN a Hamdan O ALANAZI, 2010. Overview: Main Fundamentals for Steganography. 2(3), 8.

[2] ANDERSON, Ross, Roger NEEDHAM a Adi SHAMIR, 1998. The Steganographic File System. In: David AUCSMITH, ed. *Information Hiding* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, s. 73–82 [vid. 2020-03-17]. ISBN 978-3-540-65386-8. Dostupné z: doi:10.1007/3-540-49380-8_6

[3] BALAJI, R. a G. NAVEEN, 2011. Secure data transmission using video Steganography. In: *2011 IEEE International Conference on Electro/Information Technology (EIT 2011)*: *2011 IEEE INTERNATIONAL CONFERENCE ON ELECTRO/INFORMATION TECHNOLOGY* [online]. Mankato, MN, USA: IEEE, s. 1–5 [vid. 2020-03-04]. ISBN 978-1-61284-465-7. Dostupné z: doi:10.1109/EIT.2011.5978601

[4] BANIK, Barnali Gupta a Samir Kumar BANDYOPADHYAY, 2015. Implementation of image steganography algorithm using scrambled image and quantization coefficient modification in DCT. In: *2015 IEEE International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*: *2015 IEEE International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)* [online]. Kolkata, India: IEEE, s. 400–405 [vid. 2020-02-29]. ISBN 978-1-4673-6735-6. Dostupné z: doi:10.1109/ICRCICN.2015.7434272

[5] BANNET, Krista, 2004. Linguistic steganography: survey, analysis, and robustness concerns for hiding information in text. Dostupné z https://www.semanticscholar.org/paper/LINGUISTIC-STEGANOGRAPHY%3A-SURVEY%2C-ANALYSIS%2C-AND-FOR-Bennett/9ea284ed75ef281b3f53dae5951f5f00d86475dc

[6] BARNI, Mauro, 2018. *Document and Image Compression*. B.m.: CRC Press. ISBN 978-1-4200-1883-7.

[7] BERTOLINO, Pascal, 2020. *Pascal BERTOLINO software* [online] [vid. 2020-04-08]. Dostupné z: http://www.gipsa-lab.grenoble-inp.fr/~pascal.bertolino/software.html

[8] CASEY, Chan, 2012. What Facebook Deals with Everyday: 2.7 Billion Likes, 300 Million Photos Uploaded and 500 Terabytes of Data. *Gizmodo* [online] [vid. 2020-02-15]. Dostupné z: https://gizmodo.com/what-facebook-deals-with-everyday-2-7-billion-likes-3-5937143

[9] CHOREIN, Anselme, 2010. *SilentEye - Steganography is yours* [online] [vid. 2020-01-15]. Dostupné z: https://silenteye.v1kings.io/

[10] CONWAY, Maura, 2003. Code wars: Steganography, signals intelligence, and terrorism. Knowledge, Technology & Policy [online]. 16(2), 45–62. ISSN 0897-1986, 1874-6314. Dostupné z: doi:10.1007/s12130-003-1026-4

[11] ČNB, 2019. Ochranné prvky 1000 Kč - Česká národní banka [online] [vid. 2019-11-20]. Dostupné z: https://www.cnb.cz/cs/bankovky-a-mince/bankovky/ochranne-prvky-1000-kc/

[12] DAS, Debasis, Ujwal A. LANJEWAR a Sanjiv SHARMA, 2013. The Art of Cryptology: From Ancient Number System to Strange Number System.

[13] DRIVETRIBE, 2016. Bizzarrini Manta (ItalDesign). *DriveTribe* [online] [vid. 2020-04-10]. Dostupné z: https://drivetribe.com/p/bizzarrini-manta-italdesign-NIaw2AcPTWyllD4rPW7Bcg

[14] EAST-TEC, 2013. *Hide Files, Encrypt File Encryption Software | InvisibleSecrets* [online] [vid. 2020-01-15]. Dostupné z: https://www.east-tec.com/invisiblesecrets/

[15] ECKEL, Bruce. *Thinking in Java*. 4th ed. Upper Saddle River, NJ: Prentice Hall, c2006. ISBN 978-0131872486.

[16] EVELETH, Rose, 2015. How Many Photographs of You Are Out There In the World? *The Atlantic* [online] [vid. 2020-02-15]. Dostupné z: https://www.theatlantic.com/technology/archive/2015/11/how-many-photographs-of-you-are-out-there-in-the-world/413389/

[17] FIGUEIRED, Davi, 2013. Hide In Picture. *SourceForge* [online] [vid. 2020-01-15]. Dostupné z: https://sourceforge.net/projects/hide-in-picture/

[18] FRIDRICH, Jessica a Miroslav GOLJAN, 2002. Practical steganalysis of digital images: state of the art. In: Edward J. DELP III a Ping W. WONG, ed. *Electronic Imaging 2002* [online]. s. 1–13 [vid. 2020-03-08]. Dostupné z: doi:10.1117/12.465263

[19] FRIDRICH, Jessica, Miroslav GOLJAN, Dorin HOGEA a David SOUKAL, 2003. Quantitative steganalysis of digital images: estimating the secret message length. *Multimedia Systems* [online]. 9(3), 288–302. ISSN 0942-4962, 1432-1882. Dostupné z: doi:10.1007/s00530-003-0100-9

[20] GUPTA,ANKUR GOYAL, Shailender a Bharat BHUSHAN, 2012. Information Hiding Using Least Significant Bit Steganography and Cryptography. *International Journal of Modern Education and Computer Science* [online]. 4(6), 27–34. ISSN 20750161, 2075017X. Dostupné z: doi:10.5815/ijmecs.2012.06.04

[21] HERNANDEZ-CASTRO, Julio C., Ignacio BLASCO-LOPEZ, Juan M. ESTEVEZ-TAPIADOR a Arturo RIBAGORDA-GARNACHO, 2006. Steganography in games: A general methodology and its application to the game of Go. Computers & Security [online]. 25(1), 64–71. ISSN 01674048. Dostupné z: doi:10.1016/j.cose.2005.12.001

[22] HORÉ, Alain a Djemel ZIOU, 2010. Image quality metrics: PSNR vs. SSIM. In: [online]. s. 2366–2369. Dostupné z: doi:10.1109/ICPR.2010.579

[23] IMATEST, 2020. *imatest | Image Quality Testing Software & Test Charts* [online] [vid. 2020-04-09]. Dostupné z: https://www.imatest.com/

[24] JAVATPOINT, 2018. History of Java - Javatpoint. *www.javatpoint.com* [online] [vid. 2020-04-01]. Dostupné z: https://www.javatpoint.com/history-of-java

[25] JETBRAINS, 2020. IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains. *JetBrains* [online] [vid. 2020-04-05]. Dostupné z: https://www.jetbrains.com/idea/

[26] CHOUDHARY, Kaustubh, 2012. Image Steganography and Global Terrorism. IOSR Journal of Computer Engineering [online]. 1(2), 34–48. ISSN 22788727, 22780661. Dostupné z: doi:10.9790/0661-0123448

[27] JOHNSON, Neil F. a Sushil JAJODIA, 1998. Exploring steganography: Seeing the unseen. Computer [online]. 31(2), 26–34. ISSN 0018-9162. Dostupné z: doi:10.1109/MC.1998.4655281

[28] JUDGE, J C, 2001. Steganography: Past, Present, Future [online]. UCRL-ID-151879, 15006450. [vid. 2019-11-10]. Dostupné z: doi:10.2172/15006450

[29] KESSLER, Gary C., 2004. *An Overview of Steganography for the Computer Forensics Examiner*.

[30] KIPPER, Gregory, 2004. INVESTIGATOR'S GUIDE TO STEGANOGRAPHY. 2004. vyd. B.m.: AUERBACH PUBLICATIONS. ISBN 0-8493-2433-5.

[31] KUMAR, Arvind, 2010. Steganography- A Data Hiding Technique.

[32] M., Marwa, Abdelmgeid A. a Fatma A., 2016. An Improved Image Steganography Method Based on LSB Technique with Random Pixel Selection. *International Journal of Advanced ComputerScience and Applications* [online]. 7(3) [vid. 2020-03-22]. ISSN 21565570, 2158107X. Dostupné z: doi:10.14569/IJACSA.2016.070350

[33] MASTRIANI, Mario a Juliana GAMBINI, 2010. Fast Cosine Transform to increase speed-up and efficiency of Karhunen-Loeve Transform for lossy image compression. 37, 1296–1306.

[34] MAZURCZYK, Wojciech, Steffen WENDZEL, Ignacio Azagra VILLARES a Krzysztof SZCZYPIORSKI, 2016. On importance of steganographic cost for network steganography. *Security and Communication Networks* [online]. 9(8), 781–790. ISSN 1939-0122. Dostupné z: doi:10.1002/sec.1085

[35] MALVIYA, Swati, Manish SAXENA a Dr Anubhuti KHARE, 2012. *Audio Steganography by Different Methods*.

[36] MCKAY, Bret a MCKAY, KATE, 2011. History of Invisible Ink. *The Art of Manliness* [online]. [vid. 2020-03-17]. Dostupné z: https://www.artofmanliness.com/articles/man-knowledge-the-history-of-invisible-ink/

[37] NAG, Akash, 2019. Low-Tech Steganography for Covert Operations. International Journal of Mathematical Sciences and Computing [online]. 5(1), 18–30. ISSN 23109025, 23109033. Dostupné z: doi:10.5815/ijmsc.2019.01.02

[38] OLIBONIMI, Cosim, 2019. *OpenPuff - Steganography & Watermarking* [online] [vid. 2020-01-15]. Dostupné z: https://embeddedsw.net/OpenPuff_Steganography_Home.html

[39] OPENJDK, 2020. *JEP 0: JEP Index* [online] [vid. 2020-04-01]. Dostupné z: https://openjdk.java.net/jeps/0

[40] ORACLE, 2020. *Overview (Java Platform SE 7)* [online] [vid. 2020-04-07]. Dostupné z: https://docs.oracle.com/javase/7/docs/api/index.html

[41] PROVOS, N. a P. HONEYMAN, 2003. Hide and seek: an introduction to steganography. *IEEE Security & Privacy* [online]. 1(3), 32–44. ISSN 1540-7993, 1558-4046. Dostupné z: doi:10.1109/MSECP.2003.1203220

[42] QUICKCRYPTO, 2017. *Encryption Software & Privacy Software – QuickCrypto* [online] [vid. 2020-01-15]. Dostupné z: http://quickcrypto.com/

[43] RAWAT, Deepesh a Vijaya BHANDARI, 2013. *A Steganography Technique for Hiding Image in an Image using LSB Method for 24 Bit Color Image*.

[44] REDDY, V Lokeswara, 2011. Implementation of LSB Steganography and its Evaluation for Various File Formats. 02(05), 5.

[45] SADEK, Mennatallah M., Amal S. KHALIFA a Mostafa G. M. MOSTAFA, 2014. Video steganography: a comprehensive review. *Multimedia Tools and Applications* [online]. 74(17), 7063–7094. ISSN 1380-7501, 1573-7721. Dostupné z: doi:10.1007/s11042-014-1952-z

[46] SAE-TANG, Wannida, Masaaki FUJIYOSHI a Hitoshi KIYA, 2019. A New Copyright- and Privacy-Protected Image Trading System Using a Novel Steganography- Based Visual Encryption Scheme. 13.

[47] SEKHAR, Amritha, Manoj KUMAR a Abdul RAHIMAN, 2015. A Study on Network Steganography Methods [online]. ISSN 2278-1021. Dostupné z: doi:10.17148/IJARCCE

[48] SWAIN, Gandharba a Saroj Kumar LENKA, 2012. A Better RGB Channel Based Image Steganography Technique. In: P. Venkata KRISHNA, M. Rajasekhara BABU a Ezendu ARIWA, ed. *Global Trends in Information Systems and Software Applications* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, s. 470–478 [vid. 2020-02-25]. ISBN 978-3-642-29215-6. Dostupné z: doi:10.1007/978-3-642-29216-3_51

[49] SWARAJA K, 2018. Medical image region based watermarking for secured telemedicine. Multimedia Tools and Applications [online]. 77(21), 28249–28280. ISSN 1380-7501, 1573-7721. Dostupné z: doi:10.1007/s11042-018-6020-7
THOMOS, N., N.V. BOULGOURIS a M.G. STRINTZIS, 2006. Optimized transmission of JPEG2000 streams over wireless channels. *IEEE Transactions on Image Processing* [online]. 15(1), 54–67. ISSN 1941-0042. Dostupné z: doi:10.1109/TIP.2005.860338

[50] TIOBE, 2020. *index | TIOBE - The Software Quality Company* [online] [vid. 2020-03-11]. Dostupné z: https://www.tiobe.com/tiobe-index/

[51] USC, 1973. *SIPI Image Database - Misc* [online] [vid. 2020-04-10]. Dostupné z: http://sipi.usc.edu/database/database.php?volume=misc&image=12#top

[52] VORA, Komal, 2015. Paper on Currency Recognition Systém.

[53] WAHAB, Osama F. Abdel, Aziza I. HUSSEIN, Hesham F. A. HAMED, Hamdy M. KELASH, Ashraf A. M. KHALAF a Hanafy M. ALI, 2019. Hiding data in images using steganography techniques with compression algorithms. In: [online]. Dostupné z: doi:10.12928/telkomnika.v17i3.12230

[54] WALIA, Dr Ekta, Payal JAIN a Navdeep NAVDEEP, 2010. An Analysis of LSB & DCT based Steganography. Global Journal of Computer Science and Technology [online]. [vid. 2020-01-14]. ISSN 0975-4172. Dostupné z: https://computerresearch.org/index.php/computer/article/view/912

[55] WALLHAVEN, 2020. *Awesome Wallpapers - wallhaven.cc* [online] [vid. 2020-04-10]. Dostupné z: https://wallhaven.cc/

[56] WESTFELD, Andreas a Andreas PFITZMANN, 2000. Attacks on Steganographic Systems - Breaking the Steganographic Utilities EzStego. In: *Jsteg, Steganos, and S-Tools - and Some Lessons Learned," Lecture Notes in Computer Science*. B.m.: Springer-Verlag, s. 61–75.

[57] WILSON, William, 2007. *Steganography - Java Tutorials | Dream.In.Code* [online] [vid. 2020-04-10]. Dostupné z: https://www.dreamincode.net/forums/topic/27950-steganography/

[58] YONSKAI a MARBY, 2020. Lorem Picsum. *Lorem Picsum* [online] [vid. 2020-04-09]. Dostupné z: https://picsum.photos

[59] ZHAO, Qiangfu a Tosiyasu L. KUNII, 2013. Steganography based on image morphing. In: *2013 International Joint Conference on Awareness Science and Technology & Ubi-Media Computing (iCAST-UMEDIA)*: *2013 International Joint Conference on Awareness Science and Technology & Ubi-Media Computing (iCAST 2013 & UMEDIA 2013)* [online]. Aizuwakamatsu, Japan: IEEE, s. 157–163 [vid. 2019-10-23]. ISBN 978-1-4799-2364-9. Dostupné z: doi:10.1109/ICAwST.2013.6765426

[60] ZHOU WANG, A.C. BOVIK, H.R. SHEIKH a E.P. SIMONCELLI, 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* [online]. 13(4), 600–612. ISSN 1941-0042. Dostupné z: doi:10.1109/TIP.2003.819861

[61] WAKI SOFTWARE, 2020. *Online Image Diff - Website for easy online image comparison* [online] [vid. 2020-04-09]. Dostupné z: https://online-image-comparison.com/

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor:	**Bc. Pavel Švarc**
Studium:	I1800770
Studijní program:	N1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název diplomové práce:	**Metody moderní steganografie**
Název diplomové práce AJ:	Methods of modern steganography

Cíl, metody, literatura, předpoklady:

Cílem práce bude porovnat jednotlivé steganografické metody pro ukládání zpráv do obrázků a prakticky je implementovat. Práce bude vypracována dle metodických pokynů FIM UHK.

ČANDÍK, Marek. Bezpečnosť informačných systémov, steganografia a digitálna vodotlač. Praha: M. Čandík, 2005, 117 s. ISBN 80-239-5962-X. Schneier, Bruce. Applied cryptography. 2nd ed. New York, 1996. ISBN 0-471-11709-9. A. J. Menezes. Handbook of applied cryptography. 2001. ISBN 0-8493-8523-7.

Garantující pracoviště:	Katedra informačních technologií, Fakulta informatiky a managementu
Vedoucí práce:	Ing. Zuzana Němcová, Ph.D.
Datum zadání závěrečné práce:	21.10.2014