Steganografie s tajným klíčem

Bc. Radek Fojtík

Diplomová práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Radek Fojtík**
Osobní číslo:	**A16222**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Steganografie s tajným klíčem**
Téma anglicky:	**Secret Key Steganography**

Zásady pro vypracování:

1. **Popište problematiku steganografie.**
2. **Poreferujte o možnostech aplikace steganografie, včetně druhů a užívaných metod.**
3. **Definujte a analyzujte možnosti steganografie řízené klíčem.**
4. **Navrhněte nový algoritmus steganografie řízené klíčem a vyberte vhodné médium.**
5. **Implentujte navržený algoritmus.**
6. **Analyzujte a vhodně reprezentujte získané výsledky.**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **KATZENBEISSER, Stefan. a Fabien A. P. PETITCOLAS. Information hiding techniques for steganography and digital watermarking: Stefan Katzenbeisser, Fabien A.P. Petitcolas, editors. Boston: Artech House, 2000. ISBN 1-58053-035-4.**
2. **WAYNER, Peter. Disappearing cryptography: information hiding : steganography & watermarking. 2nd ed. Boston: MK/Morgan Kaufmann Publishers, c2002. ISBN 1558607692.**
3. **WU, Min a Bede LIU. Multimedia Data Hiding. New York, NY: Springer New York, 2003. ISBN 9780387217543.**
4. **ANDERSON, Ross J. a Fabien A.P. PETICOLAS. On The Limits of Steganography. IEEE Journal of Selected Areas in Communications. 1998, (16), 474-481. ISSN 0733-8716.**
5. **SINGH, Simon. Kniha kódů a šifer: tajná komunikace od starého Egypta po kvantovou kryptografii. Praha: Dokořán, 2003. Aliter (Argo: Dokořán). ISBN 80-86569-18-7.**
6. **BENDER, W., D. GRUHL, N. MORIMOTO a A. LU. Techniques for data hiding. IBM Systems Journal [online]. 1996, 35(3.4), 313-336 [cit. 2017-11-23]. DOI: 10.1147/sj.353.0313. ISSN 0018-8670. Dostupné z:http://ieeexplore.ieee.org/document/5387237/**

Vedoucí diplomové práce: **Ing. Petr Žáček**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **3. prosince 2018**

Termín odevzdání diplomové práce: **15. května 2019**

Ve Zlíně dne 7. prosince 2018

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

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
garant oboru

Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně, dne

Radek Fojtík, v. r.
podpis diplomanta

ABSTRAKT

Tato diplomová práce se zabývá návrhem nového algoritmu pro aplikaci steganografie řízené klíčem a jeho implementací v programovacím jazyce C#. V teoretické části je popsána problematika steganografie a možnosti její aplikace, včetně druhů a užívaných metod. Dále je definována steganografie řízená klíčem. V praktické části je popsán návrh nového algoritmu a jeho implementace formou knihovny. Část práce se věnuje popisu uživatelského rozhraní vytvořené aplikace, prostřednictvím které je možné aplikovat steganografii navrženým algoritmem. V závěru práce jsou získány výsledky algoritmu, které jsou analyzovány a vyhodnoceny.

Klíčová slova:

Steganografie, Stego klíč, SPCI, Tajný klíč, Metody steganografie, Steganografie v obrázku

ABSTRACT

This master thesis deals with the design of the new algorithm for application of key-driven steganography and implementation of this algorithm in programming language C#. In theoretical part is described problematic of steganography and possibilities of its implementation, including the types and methods used. There is also defined steganography with secret key. In practical part is described design of the new algorithm and its implementation in the form of library. Next part is about graphical interface of the created application through which steganography can be applied by the new algorithm. At the end of the thesis are analysed results based on experiments.

Keywords:

Steganography, Stego key, SPCI, Secret key, Steganography methods, Image steganography

Děkuji Ing. Petru Žáčkovi za odborné vedení a vstřícný přístup, za poskytování cenných rad a připomínek. Také bych rád poděkoval své přítelkyni a své rodině za toleranci a podporu v období vypracovávání diplomové práce.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Tématem této diplomové práce je steganografie. Tato vědní disciplína se stejně jako kryptografie zabývá ochranou informací. S nástupem počítačů a z důvodu rychlého vývoje v oblasti digitalizace dostala steganografie jiný rozměr, protože se otevřely nové možnosti pro její aplikaci. Steganografie nám umožňuje svobodně komunikovat bez toho, aniž by o této komunikaci věděla třetí osoba. Může být nápomocná například v zemích, kde je velký stupeň cenzury, ale také lehce zneužitelná (například pro komunikaci teroristických uskupení). Na tento problém upozorňuje řada bezpečnostních expertů, a proto také vznikají algoritmy takzvané stegoanalýzy, které mají za cíl steganografii detekovat, získat informaci a poté ji zničit. To je jeden z hlavních problémů dnešní steganografie. Je potřeba vyvíjet nové, lepší a odolnější algoritmy pro aplikaci steganografie.

Představte si následující situace – jdete po ulici a vidíte reklamu na nový film, nebo hledáte na internetu novou ledničku a posloucháte při tom rádio. Máte jistotu, že tato reklama, obrázky ledniček na internetovém obchodu nebo hudba z rádia neobsahuje například plány nějaké vojenské základny nebo jiná tajná data? Nemáte. To je podstata steganografie. Schovat informace takovým způsobem, aby si jich nikdo nevšimnul. V dnešní době může být steganografie přítomna téměř všude.

Cílem teoretické části je vysvětlit problematiku steganografie a možnosti její aplikace včetně steganografie řízené klíčem, díky které je detekce a případná rekonstrukce tajných dat pro útočníka mnohem složitější. Steganografie stejně jako jiné vědy (například kryptografie) procházela postupným vývojem. V práci jsou uvedeny příklady použití steganografie před nástupem internetu a také moderní steganografie. Vysvětleny jsou také používané metody pro steganografii a princip steganografie řízené tajným klíčem.

Praktická část této diplomové práce se zabývá návrhem nového algoritmu steganografie řízené klíčem. Je navržen nový algoritmus, který je poté implementován formou knihovny. Dále je vytvořená knihovna, která implementuje nový algoritmus, použita formou reference v aplikaci, pomocí které je možné vkládat a rekonstruovat data novým algoritmem. Nakonec jsou analyzovány výsledky tohoto nového algoritmu.

I. TEORETICKÁ ČÁST

1 STEGANOGRAFIE

Steganografie chrání informace a používá se v různých formách přes 2500 let. Nachází využití například v armádě nebo diplomacii. Stručně řečeno je steganografie věda, která se zabývá ukrytím informace do objektu takovým způsobem, aby informace nebyla na první pohled zřejmá. Nositelem zprávy může být prakticky cokoliv. Objekt, který obsahuje skrytou zprávu je nazýván stego objektem. Objektem může být například svetr s vyšitou přerušovanou čárou, která ve skutečnosti představuje Morseovu abecedu [1].

1.1 Historie

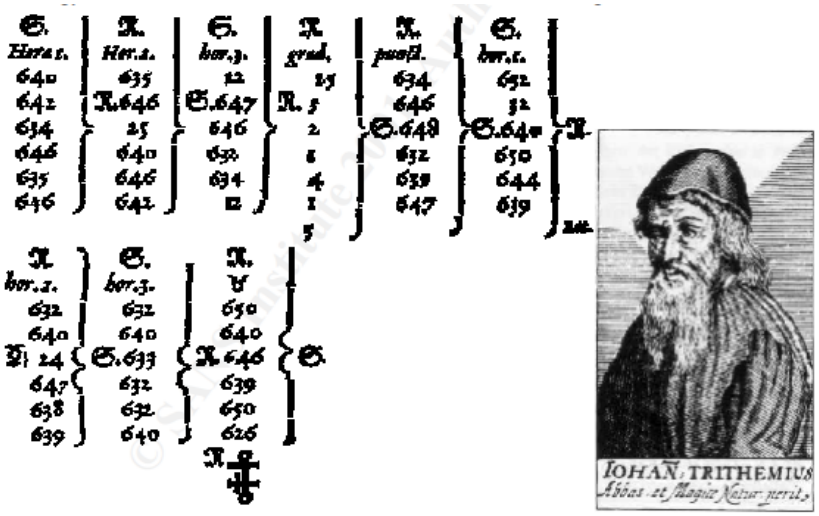
Steganografie pochází z řeckého slova stegonos, což znamená skryté nebo tajné a graphy (psát nebo kreslit).

Jedna z prvních technik steganografie byla vyvinuta v Řecku okolo roku 440 před naším letopočtem. Vládce Histaeus oholil svého otroka a na hlavu mu nechal vytetovat tajnou zprávu. Doručit zprávu se otrok vydal až poté, co mu znovu narostly vlasy. Zpráva tak byla před okolím dokonale skryta [2].

Od starověkého Říma je také známa technika psaní mezi řádky dokumentu neviditelným inkoustem vyrobeným z mléka nebo džusu, který se objeví až po nahřátí. Němci ve druhé světové válce využívali mikrotečky k ukrytí velkého množství dat. Tečky byly maskovány za znaky interpunkce. Dalším příkladem z historie je královna Irska Mary, která využila kombinaci steganografie a kryptografie k tomu, aby se dostala z vězení. Její dopisy byly šifrovány a ukryty na dně pivního sudu, který se bez povšimnutí dostal mimo území věznice [1].

Roku 1499 publikoval Johannes Trithemius (1462-1516) jednu z prvních knih o steganografii – Steganographia. Jednalo se o trilogii, kde třetí díl byl navenek o okultní astrologii. Kniha byla plná tabulek a čísel.

V roce 1996 výzkumní pracovníci Dr. Thomas Erns a Dr. Jim Reeds nezávisle na sobě nabili přesvědčení, že třetí díl trilogie obsahuje skrytý kód. Relativně rychle zjistili, jak skryté zprávy získat. Informace ale nebyly příliš významné. Pro příklad jedna ze skrytých zpráv - „Rychlá hnědá liška skáče přes lenivého psa.“ [2]



Obr. 1. Příklad tabulky z knihy Steganographia od Johannese Trithemiuse [2].

1.2 Definice steganografie

K tomu abychom dokázali porozumět steganografii, musíme nejdříve rozumět jejímu předchůdci – kryptografii. Kryptografie je věda, která se zabývá ochranou informací. Informace je transformována do nečitelné podoby (šifrovaný text) a k jejímu dešifrování je zapotřebí klíč. Kdežto hlavním cílem steganografie je skrýt samotnou přítomnost zprávy. Kryptografie a steganografie se často používá společně [1].

Steganografie je vědní disciplína, která zastřešuje všechny způsoby vkládání tajných zpráv do objektů takovým způsobem, že existence tajné zprávy zůstává skryta. Tajnou zprávou může být například text, obrázek, šifrovaný text nebo cokoliv jiného, co může být representováno sekvencí bitů (digitální steganografie) [3].

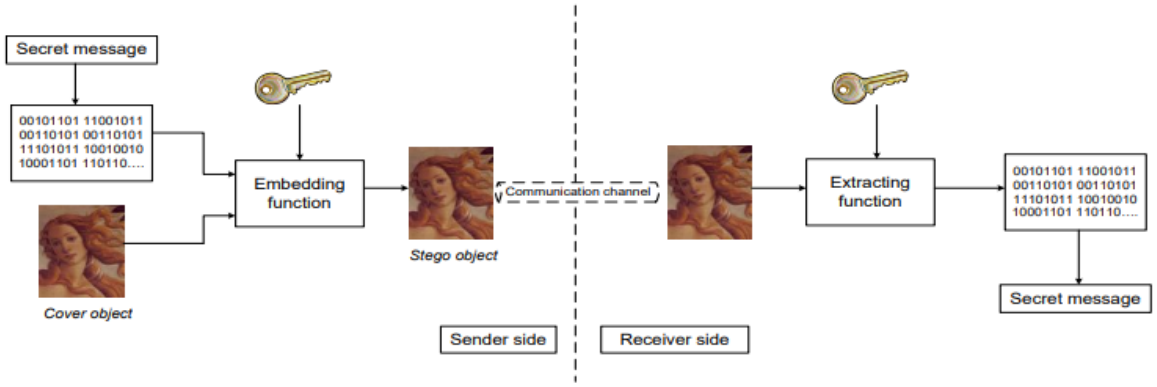
Tab. 1. Srovnání kryptografie a steganografie.

	Kryptografie	**Steganografie**
Cíl	Udržet obsah zprávy tajný.	Udržet v tajnosti samotnou existenci zprávy.
Použití	Zabezpečení informace proti potenciálním odposlechům.	Zabezpečení informace proti potenciálním odposlechům.
Bezpečnostní služby	• Důvěryhodnost • Integrita dat	• Důvěryhodnost

	• Identifikace a autentizace	• Identifikace a autentizace
Problémy	• Distribuce klíče • Vymáhání práva • Kryptoanalýza	• Stegoanalýza • Distribuce klíče (v případě steganografie s tajným klíčem)

V současné době je rozšířené ukrývání informací například do hudby nebo grafických souborů (obrázků, videí apod.). Existuje množství algoritmů a způsobů, pomocí kterých je informace do objektu zaintegrována [2].

Vkládání zprávy do objektu (cover objekt) bývá někdy doplněno o tajný klíč, který je nazýván stego klíčem. Bez znalosti klíče je pro neautorizovanou stranu velice obtížné skrytou zprávu detekovat a extrahovat.



Obr. 2. Model procesu steganografie [4].

Na obrázku č. 2 je ilustrován hlavní model pro steganografii, kde cover objektem je obrázek. Odesilatel nejprve transformuje tajnou zprávu a poté manipulací bitů obrázku vytvoří stego objekt. Následně je stego objekt pomocí přenosového kanálu (například internetu) doručen příjemci, který reverzním způsobem získá tajnou zprávu. Pokud je použit tajný klíč, tak jej příjemce i odesílatel musí znát [4].

Rozdíl mezi steganografií a kryptografií je evidentní. Existují ale další dvě technologie, které se steganografií souvisí a kde rozdíl není tak znatelný – watermarking a fingerprinting. Liší se především účelem, robustností a kapacitou.

Watermarking (digitální vodoznak) je metoda pro vkládání dat do digitálního multimediálního obsahu. Používá se k ověření důvěryhodnosti obsahu, nebo k rozpoznání identity vlastníka digitálního obsahu. Na druhou stranu fingerprinting vkládá odlišné unikátní značky do jednotlivých kopií objektu, které jsou pak dodávány zákazníkům. To umožňuje vlastníkům jednoznačně identifikovat zákazníka, který porušil licenční smlouvu [3].

Hlavním rozdílem mezi těmito technologiemi je fakt, že objektem komunikace pro watermarking a fingerprinting je profesní objekt, do kterého jsou vkládána data za účelem ochrany autorských práv. Informace o skrytých datech bývá často veřejná – někdy dokonce viditelná – kdežto u steganografie je maskování informace naprosto zásadní. Za úspěšný útok na steganografický systém lze považovat takový útok, který odhalí skryté informace uvnitř objektů. Zatímco u fingerprintingu nebo watermarkingu se útočníci snaží detekovat značku a tu pak odstranit. Rozdíly mezi těmito třemi technologiemi jsou shrnuty v tabulce č. 1 [4].

Tab. 2. Shrnutí rozdílů steganografie, watermarking a fingerprinting.

	Watermarking	**Fingerprinting**	**Steganografie**
Účel	Ochrana autorských práv	Ochrana autorských práv pomocí snadné identifikace strany, která porušila licenční podmínky.	Přenos tajné zprávy bez povšimnutí.
Neviditelnost	Žádoucí, ale ne rozhodující.	Žádoucí, ale ne rozhodující.	Zásadní. Vkládaná informace nesmí být viditelná.
Odolnost vůči nepřátelskému odstranění, zničení nebo padělání	Zásadní je, aby nikdo nebyl schopen odstranit informaci.	Zásadní je, aby nikdo nebyl schopen odstranit informaci.	Žádoucí, ale ne rozhodující.
Kapacita	Nedůležitá, protože informace je obecně malá.	Nedůležitá, protože informace je obecně malá.	Velmi důležitá, pokud je potřeba přenést velkou zprávu.

1.3 Tradiční využití steganografie

Obecně používají steganografii lidé, kteří si přejí komunikovat tajně a svobodně. Utajená komunikace je zvláště důležitá v cenzurovaných nebo monitorovaných prostředích. Steganografie se také používá k ochraně soukromé komunikace tam, kde použití kryptografie není povoleno nebo by mohlo vyvolat podezření [5].

Steganografie se také používá společně s ostatními zabezpečovacími mechanismy pro zajištění vícevrstvé bezpečnosti. V případě prolomení jedné vrstvy musí útočník prolomit ještě další.

Nenápadná komunikace je stěžejní například pro vojenské a zpravodajské agenty. I když je obsah zprávy šifrován, tak samotná detekce zprávy může na moderním bojišti rychle vést k útoku na odesilatele. Steganografie zajistí, aby byla zpráva skryta.

Steganografie se také používá pro uložení informací, které nechceme s nikým komunikovat. Citlivé informace, například bankovní údaje, mohou být vloženy do objektu (např. obrázku) a zůstat uloženy na osobním počítači [4].

1.4 Steganografie 20. století

Steganografie se netýká pouze digitálního světa počítačů. Existovala v různých formách a v různých oblastech již před vzestupem počítačů a internetu. Ve 20. století se hojně využívala ve špionáži během války. Níže je popsáno několik technik [6].

1.4.1 Null Cipher

Null Cipher je technika založen na tom, že tajná zpráva je ukryta v textu za původními znaky textu. Například v první světové válce odeslala německá ambasáda v USA telegraf do Berlína – „*President's Embargo Ruling Should Have Immediate Notice. Grave Situation Affecting International Law. Statement Foreshadows Ruin Of Many Neutrals. Yellow Journals Unifying National Excitement Immensely*“ [5, s. 130]. Pokud se zaměříme pouze na první písmena slov, tak dostaneme zprávu – „Pershing Sails from NY June I“.

1.4.2 Jargon Code

Jargon Code je technika, která spočívá v tom, že skupina jednotlivců vytvoří vlastní kód v určitém jazyce, kterému ostatní nerozumí. Například skupina špionů může navrhnout

vlastní Jargon Code a tajně komunikovat například pomocí fráze „Děti dnes přišly ze školy dříve.“, což ve skutečnosti může znamenat „Peníze už jsou na bankovním účtu.“.

1.4.3 Anamorphosis

Anamorphosis je řecké slovo, které znamená „Změna tvaru“. Jedná se o techniku, která vkládá informaci do obrázku. Informace je viditelná pouze v případě, že se na obrázek podíváme z jiného úhlu, nebo pomocí speciálního zařízení.

1.4.4 Neviditelný inkoust

Neviditelný inkoust je látka, která pro zviditelnění potřebuje například teplo, světlo nebo speciální přípravek. Používá se po staletí a vyrábí se například z mléka, citrónu nebo chemikálií typu chloridu a amoniaku.

1.4.5 Mikrotečky

Mikrotečky jsou ve skutečnosti velké fotografické objekty, u kterých je možné zásadně zredukovat jejich velikost tak, aby vypadaly jako normální tečky v interpunkci (přibližně jeden milimetr velké). Pomocí mikroskopu je pak zpráva extrahována. Mikrotečky byly využívány vojenskými jednotkami během druhé světové války [5].

1.5 Digitální steganografie

Digitální steganografie se zabývá ukrytím tajné zprávy v digitálních souborech (obrázek, text, video, zvuk atd.). Zásadní je, aby zpráva nebyla viditelná a také, aby se vlastnosti objektu po vložení tajné informace nijak zvlášť nezměnily (velikost, kvalita) [7].

Formálně lze model steganografie matematicky definovat jako (znázorněn na obrázku č. 3):

- Originální soubor, do kterého je tajná zpráva vložena je označen jako A
- Tajná zpráva je označena jako M
- Výsledný stego objekt je označen jako C (visuálně/zvukově identický originálnímu souboru A)
- Algoritmus kódování, který je použit pro vložení tajné zprávy M do souboru A je označen jako S(A,M)
- Algoritmus dekódování, který je použit pro obnovení zprávy M z objektu C je označen jako S(C) = M

Obr. 3. Matematický model steganografie.

1.5.1 Prvky digitální steganografie

Algoritmicky zastřešuje steganografie dva procesy. Proces vložení informace a proces extrahování informace (proces extrahování je v podstatě reversní proces vložení). Moderní steganografie se řídí pěti klíčovými prvky:

- Skrytá data – jde o informaci, která má být ukryta. Informace můžete představovat cokoliv, co lze konvertovat do binární podoby (od textu po spustitelné soubory).
- Zařízení pro zobrazení – typicky je to soubor, do kterého se vkládají skrytá data. Zařízením je pak jakýkoliv počítač, který dokáže přečíst soubor jako obrázek, text apod.
- Stego objekt – výsledný soubor, který je složen z originálního souboru a tajné zprávy.
- Typ souboru – označuje typ souboru, do kterého je tajná zpráva vložena. Například BMP, JPG, MP3, PDF atd.
- Kapacita – označuje množství dat, které lze použít pro skrytá data bez toho, aniž by došlo ke zkreslení originálního souboru [6].

1.6 Problémy steganografie

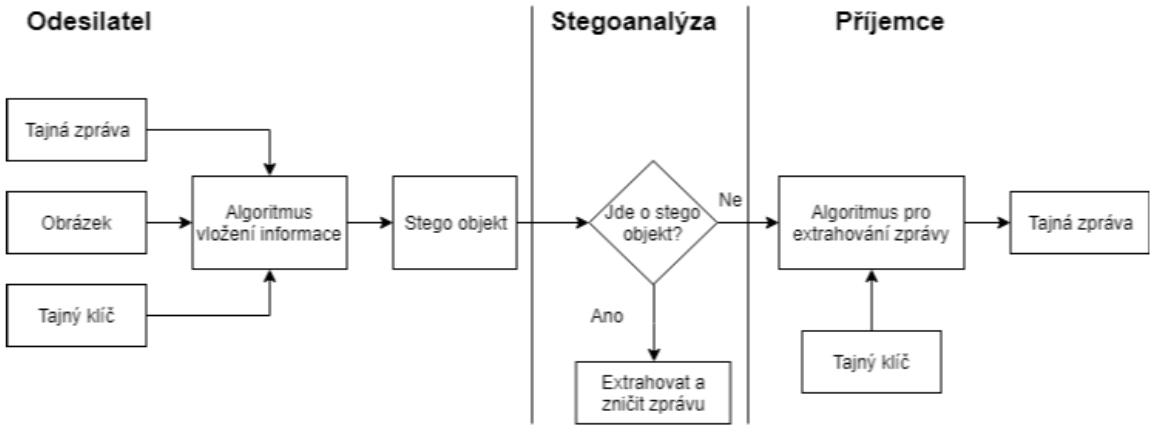
Největším problémem steganografie je rychlý vývoj v oblasti výzkumu stegoanalýzy. Stegoanalýza je technologie, která se snaží detekovat steganografii. V poslední době počítačoví

specialisté a bezpečnostní pracovníci upozorňují na to, že nezákonné použití steganografie může být bezpečnostní hrozbou pro celý svět. Steganografie může být zneužita například teroristy pro jejich komunikaci.

Z důvodu této hrozby se výzkumní pracovníci aktivně pokoušejí najít nedostatky v existujících steganografických systémech, pomocí kterých dokáží detekovat přítomnost skrytých dat a případně je extrahovat nebo zničit.

Stegoanaláza zahrnuje dvě hlavní techniky – visuální analýzu a statistickou analýzu. Visuální analýza se snaží odhalit skrytou informaci pouhým pohledem. Na druhou stranu statistická analýza se snaží odhalit steganografii pomocí drobných změn ve statistických charakteristikách způsobených přítomností tajné zprávy.

Stegoanalýza může být například součást firewallu poštovního serveru. V případě že je v emailu příloha, tak je zkontrolována pomocí stegoanalýzy a případně je odhalená skrytá zpráva zničena [7].



Obr. 4. Stegoanalýza.

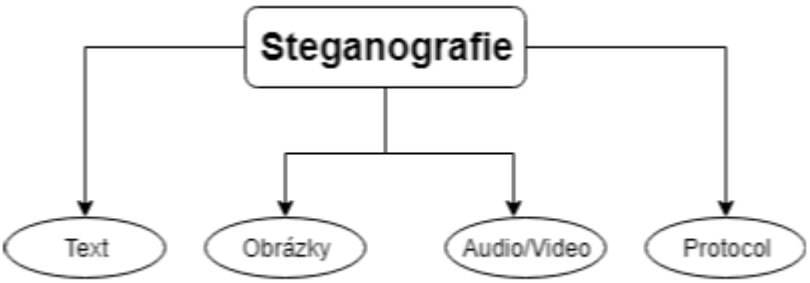
1.7 Kategorie steganografie

Existují dva hlavní přístupy, podle kterých můžeme steganografii kategorizovat. Podle typu souboru, do kterého je informace vložena a podle techniky, která je pro vkládání použita.

1.7.1 Rozdělení podle typu souboru

Typ souboru, do kterého je informace vkládána rozděluje steganografii. Různé formáty souborů mají různé vlastnosti, na základě kterých lze jiným způsobem informaci vložit. Například u obrázků se nejčastěji používá metoda nejméně důležitého bitu (Least Significant Bit

– dále jen LSB). Dnešní obrázky mohou mít přes 16 milionů barev, ale lidské oko dokáže rozpoznat asi 10 milionů. V každém pixelu je možné nahradit nejméně významný bit a ten použít pro zprávu, kterou chceme ukrýt [8].



Obr. 5. Steganografie podle typu souboru.

Na obrázku č. 4 jsou zobrazeny čtyři hlavní formáty souborů, které mohou být použity pro steganografii. Každý z těchto formátů používá různé techniky pro streganografii.

Steganografie v textu je historicky jedna z nejdůležitějších metod steganografie. Často užívaná byla technika Null Cipher, která je popsána v kapitole 1.4. V digitálním světě se steganografie v textu už příliš nepoužívá, protože textové soubory jsou příliš malé a poskytují tak malou možnou kapacitu pro skrytou zprávu [6].

Jedním z nejvhodnějším a nejpoužívanějším typem pro steganografii je obrázek. Dnešní digitální obrázky obsahují množství redundantních informací, které mohou být nahrazeny tajnou zprávou.

Steganografie ve zvukových souborech je založena na výzkumech, které se zabývaly vlastnostmi lidského ucha. Konkrétně zkoumaly množství dat, které lze ze zvukových souborů odstranit bez toho, aby si toho posluchač všiml. Tyto výzkumy využívají kompresní algoritmy, například MPEG, ale jsou také používány pro steganografii [7].

Video je kolekcí obrázků a zvuků. Techniky steganografie používané pro obrázky a zvuky lze taktéž aplikovat pro video. Výhodou videa je, že poskytuje velké množství dat pro zprávu. Nevýhodou je, že je výsledný stego objekt velký.

V digitálním světě se pro steganografii jako objekt používá také síťový paket, který je po síti přenášen pomocí daného protokolu. V samotných datech paketu je možné ukrýt zprávu.

1.7.2 Rozdělení podle metody

Steganografii lze rozdělit také podle metody, která je použita pro vložení dat. V podstatě existují tři hlavní způsoby, jak data ukrýt.

- **Injection** – tato metoda najde v souboru místo, které je ignorováno a umístí data zde. Velké množství souborů používá značku, která říká, že jde o konec souboru (end-of-file, dále jen EOF). Například v případě audia aplikace přehrává zvuk a pokud narazí na značku EOF, tak přehrávání ukončí, protože si myslí, že jde o konec souboru. Skrytá zpráva může být vložena za značku EOF. Nebude pak mít žádný vliv na zvuk souboru.
- **Substitution** – metoda substituce je založena na nahrazení určitých bitů v souboru (nejznámější metoda LSB). Například zvukový soubor – každý zvuk, který slyšíme se skládá z několika bytů. Pokud změníme nejméně významný bit každého bytu, tak výsledný zvuk bude mírně jiný. Nicméně tak, že rozdíl nebude slyšitelný.
- **Generation** – tato metoda je unikátní v tom, že nepoužívá objekt pro vložení tajné zprávy, ale objekt je vygenerován na základě tajné zprávy. Obrázek je vytvořen na základě bitové sekvence skryté zprávy – například 0 může představovat žlutou a 1 zelenou. Velkou výhodou této metody je fakt, že výsledný stego objekt nemůže být porovnán s originálním objektem [6].

2 APLIKACE DIGITÁLNÍ STEGANOGRAFIE

Aplikace steganografie je různorodá. Zahrnuje následující oblasti:

- **Tajná komunikace** – zašifrovaná zpráva může vzbudit nechtěné podezření, jestliže je přenášena určitým kanálem viditelně. Mimo to, použití kryptografie může být v mnoha situacích omezeno zákonem. Použitím steganografie se vyhýbáme zveřejnění samotné komunikace a předcházíme tomu, že zpráva bude odhalena. V mnoha případech je důležité, aby byl utajený nejen obsah zprávy ale i samotná komunikace.
- **Digitální vodoznak** – k ověření pravosti se do digitálních souborů vkládá tajná značka (autorská práva). Využívá se algoritmů pro steganografii, protože viditelná značka může dílo znehodnotit. Digitální vodoznak se používá k jednoznačné identifikaci vlastníka. Pomáhá také v boji proti pirátství, protože odhaluje nezákonně kopírované materiály.
- **Integrita dat** – použitím steganografie lze např. do obrázku vložit tajnou značku, která zaručuje, že data se po doručení nezměnila (například, že obrázek nebyl nijak poškozen kompresí apod.). Jedná se o formu steganografie, která je označována jako „Digitální podpis“. Kdykoliv je možno ověřit integritu dat [8].

2.1 Klasifikace technik steganografie

Existující algoritmy, které aplikují steganografii lze klasifikovat do těchto kategorii:

- **Spatial domain** – data představující bitovou sekvenci jsou ukryty v objektu nahrazením originálních bitů. Jsou vkládány přímo na předem známé místo. Nejznámější a nepoužívanější metodou je LSB. Výhodou je velká kapacita pro skrytou zprávu a malá šance na degradaci originálního objektu. Nevýhodou je, že skrytá data můžou být snadno zničena (například kompresí) [4].
- **Transform domain** – jedná se o více komplexní způsob, který využívá různé transformace a algoritmy pro ukrytí tajné zprávy. Tyto techniky umožňují vytvořit stego objekt, který bude odolný například kompresi (informace nebude poškozena). Většina těchto technik není závislá na formátu obrázku a je tak možno provádět převody mezi ztrátovými a bezztrátovými formáty. Transformační metody ukrývají data ve významných oblastech objektu, což zaručuje větší robustnost a odolnost vůči kompresi. Využívá se například DFT (diskrétní Furierova transformace), DCT (diskrétní kosínová transformace) a DWT (diskrétní vlnková transformace) [8].

- **Spread spectrum** – tajná data jsou rozložena v celé šířce frekvenčního pásma signálu. Tato technika se využívá například v rádiovém přenosu. Zpráva je ukryta pod úrovní frekvence šumu. Poměr signálu k šumu v každém frekvenčním pásmu je tak malý, že je velmi obtížné detekovat přítomnost dat. Je velmi složité skrytá data zcela odstranit bez toho, aniž by byl zničen původní signál. Jedná se o velmi robustní přístup používaný armádou.
- **Statistické metody** – tyto techniky využívají existenci „1-bit“ steganografických schémat, které vkládají informaci o velikosti jednoho bitu do digitálního objektu. Pokud je přenášena hodnota 1, tak se významně změní statistické charakteristiky obrázku. V opačném případě je obrázek nezměněn. Příjemce musí být schopen rozlišit mezi změněnými a nezměněnými obrázky nebo bloky. V případě, že chceme vložit zprávu o délce větší než 1 bit, tak musíme originální objekt rozdělit do tolika bloků, kolik bitů má tajná zpráva. Každý blok je pak změněn nebo nezměněn v závislosti na daném bitu tajné zprávy. K určení, zda blok obsahuje změněný bit se používá testovací funkce [4].
- **Distortion metody** – během procesu extrakce tajné zprávy je zapotřebí originální objekt (nikoli pouze stego objekt, tak jako tomu je u jiných metod). Odesilatel aplikuje sekvenci změn, která odpovídá tajné zprávě. Zpráva je zakódována do pixelů, které jsou vybrány pseudonáhodně. Pokud se stego objekt na daném pixelu liší od originálního objektu, tak bitem zprávy je 1 (v opačném případě 0). Tato technika je často používaná pro ukrytí tajné zprávy v textu. Například vkládání neviditelných znaků (konce řádků, mezery, tabulátory) do HTML souborů – webový prohlížeč tyto znaky ignoruje.
- **Metody generování** – výsledný stego objekt není vytvořen z originálního objektu a tajné zprávy, ale je vygenerován na základě daného algoritmu z tajné zprávy. Vzniká pouze za účelem přenosu tajné informace [9].

Tab. 3. Srovnání technik steganografie.

Technika	**Nepostřehnutelnost**	**Robustnost**	**Kapacita**
Spatial domain	Vysoká	Slabá	Vysoká
Transform domain	Vysoká	Vysoká	Slabá
Spread spectrum	Vysoká	Střední	Vysoká

Statistické metody	Střední	Slabá	Slabá
Disortion metody	Slabá	Slabá	Slabá
Metody generování	Střední	Slabá	Vysoká

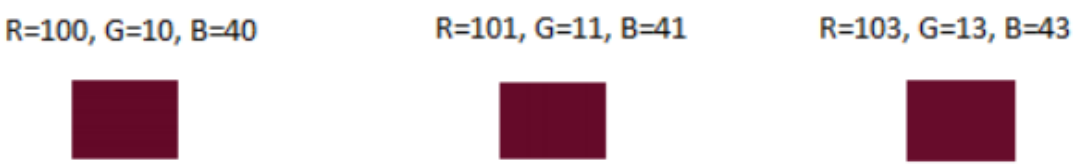
2.2 Seganografie v obrázku

Digitální obrázek je reprezentován jako pole čísel (bytů), které specifikují danou barvu a její intensitu. V závislosti na bitové hloubce obrázku jsou byty seskupeny do pixelů (v případě 16-bitové hloubky tvoří jeden pixel 16 bitů). 16-bitový obrázek (2^{16} = 65553 různých barev pro daný pixel) o rozlišení 800x600 pixelů obsahuje 800*600*16 = 7 680 000 bitů. Malá změna v jednotlivých bitech pixelu není lidským okem postřehnutelná. Obrázek je tak ideálním objektem pro steganografii. Jedná se o nejpopulárnější objekt pro steganografii. Níže je popis některých algoritmů, které vkládají tajnou zprávu do digitálního obrázku [6].

2.2.1 LSB

Metoda nejméně významného bitu je jednou z nejúčinnějších a nepopulárnějších metod steganografie. Lze ji aplikovat na soubory různých formátů (obrázek, video atd.). Nicméně nejčastější použití LSB je v obrázku. Je založena na tom, že člověk jen těžce okem detekuje jemnou změnu například v odstínu v jednotkách pixelů.

Obrázek je reprezentovaný maticí pixelů. V případě 24-bitového obrázku se každý pixel skládá ze tří bytů (RGB). LSB vychází z toho, že v případě změny nejméně významného bitu v každém bytu pixelu, není změna detekovatelná lidským okem. Na obrázku č. 6 lze vidět, jak malý je rozdíl v pixelech, které se liší pouze nejméně významným bitem [12].



Obr. 6. Srovnání pixelů.

Máme-li obrázek o velikosti 800 x 600 pixelů, tak pro tajnou zprávu můžeme využít celkově 1 440 000 bitů (180 000 B).

Příklad:

Mějme tří pixelový 24-bitový obrázek:

(00101101 00011100 11011101)

(10100111 11000101 00001101)

(11010010 10101101 01100011)

Tajnou zprávou, kterou chceme do tohoto obrázku vložit pomocí metody LSB je číslice 500, jenž je binárně reprezentovaná jako 11110100.

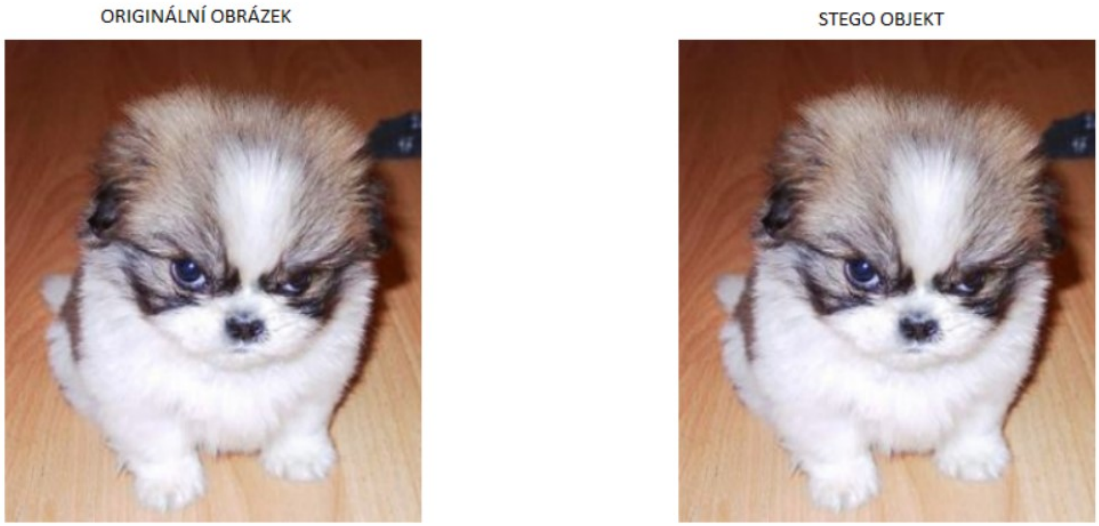
Výsledné pixely stego objektu:

(00101101 0001110**1** 11011101)

(10100111 1100010**0** 00001101)

(11010010 1010110**0** 01100011)

Přestože vkládáme informaci o délce 8 bitů, tak ve výsledném obrázku změníme pouze 3 bity (potrženy). Výsledná změna nebude lidským okem detekovatelná a je tak úspěšně ukrytá. Statisticky nevýznamně měníme pouze polovinu bytů.

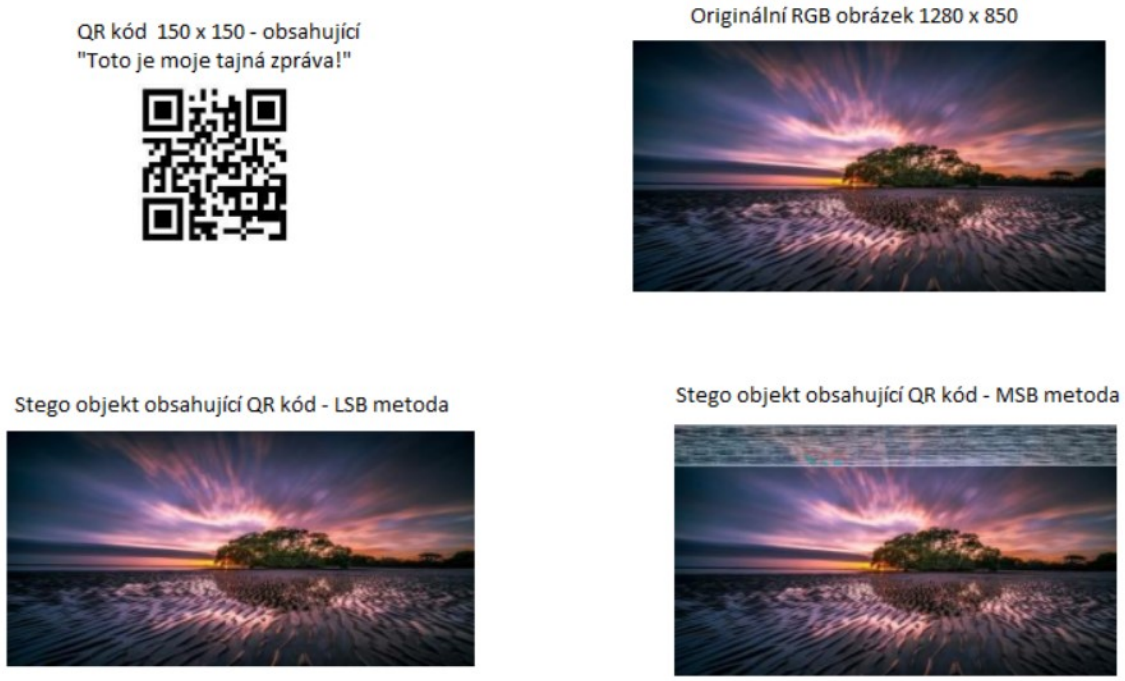


Obr. 7. Aplikace LSB.

Tajné data mohou být vloženy do jednoho bitu každé barevné komponenty, jež jsou reprezentovány byty. Z toho vyplývá, že originální objekt musí být nejméně 8krát větší než data tajné zprávy.

Na obrázku č. 8 je srovnání použití metody LSB a metody MSB (most significant bit). Metoda MSB se v praxi nepoužívá, důvod je viditelný na obrázku. Jestliže budeme měnit nejvíce významný bit, tak lze jednoduše poznat, že je obrázek znehodnocen a má v sobě něco, co do něj nepatří.

Technika nejméně významného bitu je nejjednodušší způsob, jak vložit tajnou zprávu. Je ale také velice náchylná na jakoukoli malou změnu stego objektu. V mnoha případech útočníkovi stačí ke zničení tajné informace ztrátová komprese [6].

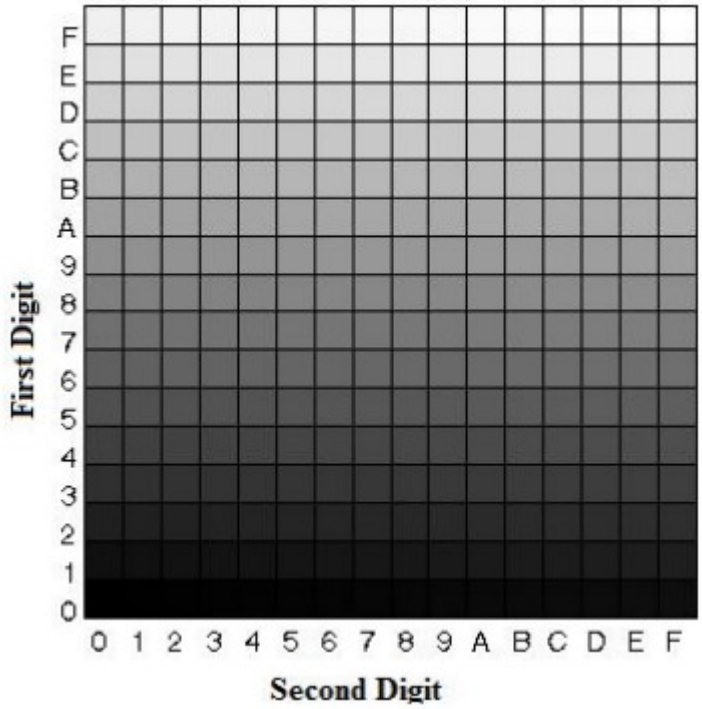


Obr. 8. Aplikace LSB metody a metody MSB.

2.2.2 Metoda generování

Technika generování se odlišuje od ostatních steganografických technik, protože nevyžaduje originální soubor, do kterého je zpráva vkládána. Jde o techniku, která vytváří stego objekt přímo z tajné zprávy a za jediným účelem – skrytí tajné zprávy. Velkou výhodou této metody je fakt, že nikdo nemůže porovnat stego objekt s originálním objektem.

Vygenerovat obrázek lze například pomocí barevné stupnice šedi. Jedná se o 256 variací šedé barvy, které vznikají kombinací černé a bílé barvy. Každých 8 bitů kombinovaných společně reprezentuje jinou barvu. Tajná zpráva (text) je převedena do bitové podoby a poté může být pomocí operace XOR a tajného klíče zakódována. Tajný klíč má většinou jinou velikost než zpráva, proto je potřeba klíč opakovat až do délky zprávy. Potom co provedeme operaci XOR nad klíčem a zprávou, je potřeba rozdělit výslednou bitovou sekvenci do skupin po 8 bitech. Z toho důvodu, že každá skupina 8 bitů (jeden byte) reprezentuje jednu barvu. Místo sekvence bitů, která reprezentuje tajnou zprávu zakódovanou pomocí tajného klíče, jsme získali sekvenci barev ze stupnice šedi [10].



Obr. 9. Stupnice šedi [10].

Příklad:

Tajná zpráva, kterou chceme skrýt je slovo „Thesis“. Vytvoříme si bitovou sekvenci pomocí ASCII tabulky – každý znak našeho textu (tajné zprávy) představuje 8 bitů.

Tab. 4. ASCII kód „Thesis“.

Znak	**Binární podoba**	**Decimální podoba**
T	0101 0100	84
h	0110 1000	104

e	0110 0101	101
s	0111 0011	115
i	0110 1001	105
s	0111 0011	115

Převedením „Thesis“ do binární podoby získáváme bitovou sekvenci:

T h e s i s

01010100 01101000 01100101 01110011 01101001 01110011

Jako klíč použijeme slovo „Sara“, které se skládá pouze ze čtyř znaků. Klíč v binární podobě:

S a r a

01010011 01100001 01110011 11100001

Z důvodu toho, že náš klíč má menší velikost než zpráva, tak je potřeba klíč rozšířit – opakovat do velikosti zprávy. Náš klíč tedy bude mít podobu:

S a r a S a

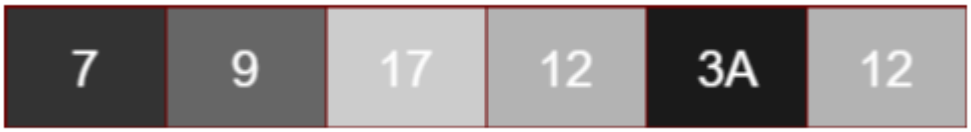
01010011 01100001 01110010 01100001 01010011 01100001

K zakódování zprávy provedeme operaci XOR nad bitovou sekvencí klíče a bitovou sekvencí zprávy.

Tab. 5. Operace XOR nad klíčem a zprávou.

XOR	**01010100**	**01101000**	**01100101**	**01110011**	**01101001**	**01110011**
	01010011	01100001	01110010	01100001	01010011	01100001
Binární podoba	00000111	00001001	00010111	00010010	00111010	00010010
Hexadecimální podoba	7	9	17	12	3A	12

Výsledek v hexadecimální soustavě reprezentuje 6 odlišných barev ze stupnice šedi.



Obr. 10. Výsledná sekvence barev.

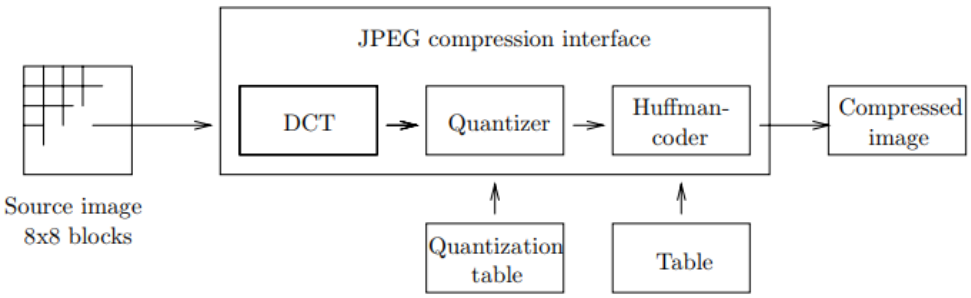
Proces extrakce zprávy z obrázku je jednoduchý. Získáme binární podobu a provedeme operaci XOR s bitovou sekvencí klíče, který musíme znát. Poté převedeme bity na alfanumerické znaky a získáme zprávu.

Hlavní výhodou této steganografické techniky je to, že nevyžaduje originální soubor, do kterého se tajná zpráva vloží. Další výhodou je malá velikost výsledného souboru. Je stejně velký jako tajná zpráva. Nevýhodou je slabá ochrana proti útoku (stegoanalýza) [10].

2.2.3 Metoda JSTEG

Pro implementaci steganografie se také využívá diskrétní kosínová transformace (DCT). Tato metoda spadá do kategorie transformačních technik. Hlavní výhodou je to, že vložená informace je do jisté míry odolná vůči například kompresi nebo jinému zásahu do stego objektu. Transformace může být aplikovaná na celý obrázek nebo pouze na určité bloky [11].

Dvou dimensionální DCT používá jedna z nejpopulárnějších ztrátových kompresí obrázků dnešní doby – JPEG. JPEG nejdříve konvertuje obrázek do barevného modelu YCbCr a poté rozdělí obrázek do bloků 8x8 pixelů. Následně je na tyto bloky aplikována DCT transformace. Z každého bloku je získán DCT koeficient, který je vydělen podle předdefinovaných hodnot a zaokrouhlen (ztrátová část). Koeficienty jsou poté zakódovány pomocí entropického kódování – například Huffmanovo kódování. Pří dekódování JPEG je po dekvantizování aplikována k rekonstrukci obrazu inversní DCT [12].



Obr. 11. Proces JPEG komprese [4].

Steganografie pomocí DCT vychází z faktu, že komprese JPEG je rozdělena do ztrátové a bezztrátové etapy. Steganografie obvykle skrývá informace v redundantních datech a ty jsou během ztrátové komprese odstraňovány. Steganografie vstupuje do procesu mezi kvantováním DCT (ztrátová část) a entropickým kódováním (bezztrátová část). JPEG steganografie vkládá informaci do nejméně významného bitu nenulového DCT koeficientu před aplikací Huffmanova kódování. Namísto vložení informace přímo do pixelu obrázku, jinými slovy do prostorové oblasti, je informace vložena do DCT koeficientu, což učiní informaci dokonale skrytou [11].

Algoritmus DCT steganografie aplikovaný na straně odesilatele:

Krok-1: Převod tajné zprávy do binární podoby

Krok-2: Výběr objektu, do kterého bude zpráva vložena

Krok-3: Rozdělit objekt na bloky o velikosti 8x8 pixelů

Krok-4: Pomocí DCT získat z každého bloku koeficient

Krok-5: Každý blok kompresovat pomocí kvantizační tabulky

Krok-6: Získat LSB každého DCT koeficientu a nahradit bity tajné zprávy

Krok-7: Vytvořit stego objekt

Algoritmus pro obnovení zprávy:

Krok-1: Získat stego objekt

Krok-2: Rozdělit stego objekt do bloků o velikosti 8x8 pixelů

Krok-3: Rozdělit objekt na bloky o velikosti 8x8 pixelů

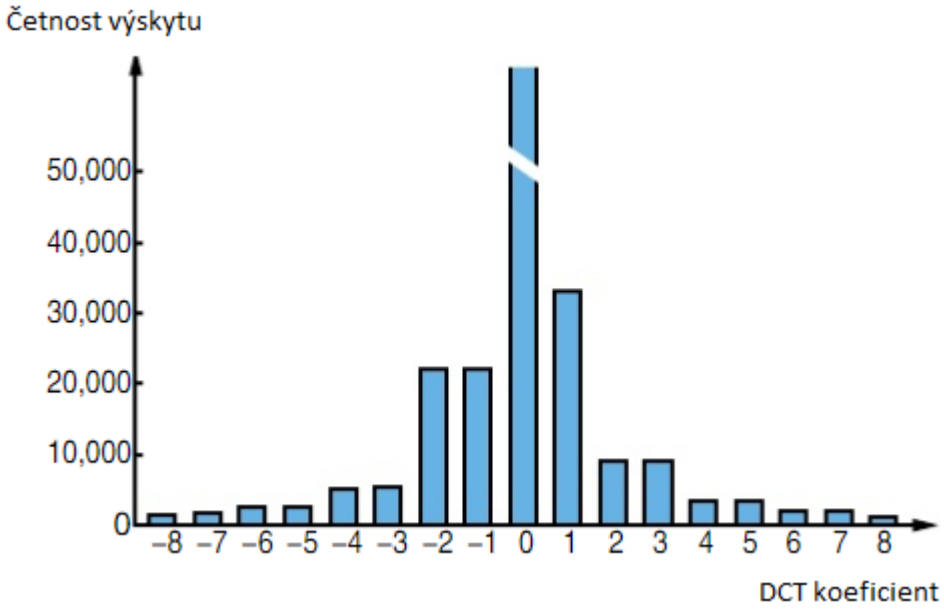
Krok-4: Pomocí DCT získat z každého bloku koeficient

Krok-5: Získat LSB každého DCT koeficientu a přidat do výsledné sekvence bitů

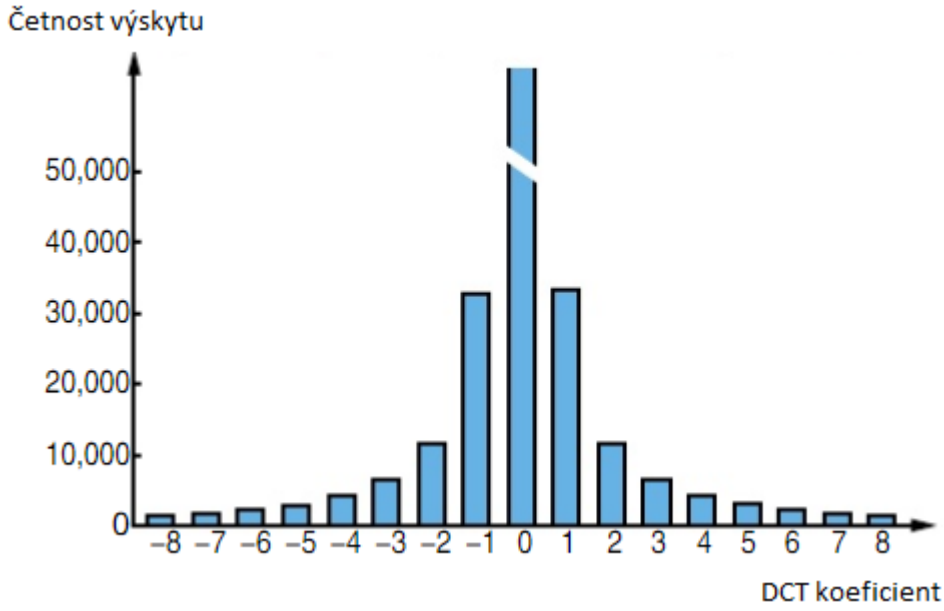
Krok-6: Sekvenci bitů rozdělit do skupin po 8 bitech a každý byte konvertovat na znak

Histogram koeficientů DCT přirozeného obrazu ukazuje určitý symetrický tvar s rozložením koeficientů kolem nuly pro všechny snímky JPEG. Znalost těchto charakteristik může být použita k určení, zda se v obrázku vyskytují nějaká tajná data. Protože JPEG steganografie

nahrazuje bity v DCT koeficitentech, tak se histogram odchyluje od normy a přítomnost vložených bitů může být statisticky zjištěna. Tento fakt je hlavní slabinou DCT steganografie. Histogram DCT koeficientů po aplikaci steganografie a před aplikací steganografie je znázorněn na obrázku č. 12 a obrázku č. 13 [13].



Obr. 12. Histogram DCT koeficientu po aplikaci steganografie [13].



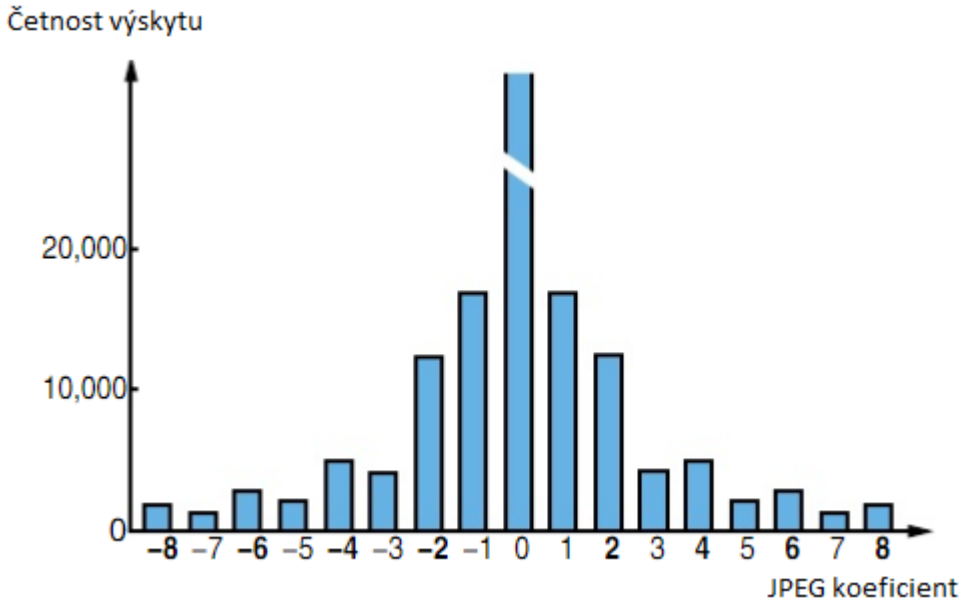
Obr. 13. Histogram DCT koeficientu bez aplikace steganografie [13].

K eliminaci rizika detekce skrytých dat v JPEG obrázku pomocí histogramu DTC koeficientů byly vyvinuty nové algoritmy. Například F3, F4 a F5 [12].

2.2.4 Metoda F3

Algoritmus F3 pracuje také s DCT koeficientem, ale nepřepisuje hodnotu jeho LSB v případě, že hodnota není shodná. Namísto přepisování LSB se dekrementuje absolutní hodnota koeficientu. S výjimkou nulového koeficientu. LSB nenulových koeficientů jsou shodné s bity skryté zprávy, ale bity koeficientů nejsou přepisovány, protože chi-kvadrát test jednoduše odhalí tyto změny.

V případě, že se dekrementuje absolutní hodnota -1 nebo 1, tak dojde k zanesení nulového koeficientu, který nenese tajnou zprávu. Je potřeba vkládat dotčený bit opakovaně. Z tohoto důvodu dochází k přebytku sudých koeficientů. Proces vkládání informace algoritmem F3 tedy produkuje více sudých než lichých koeficientů [13].



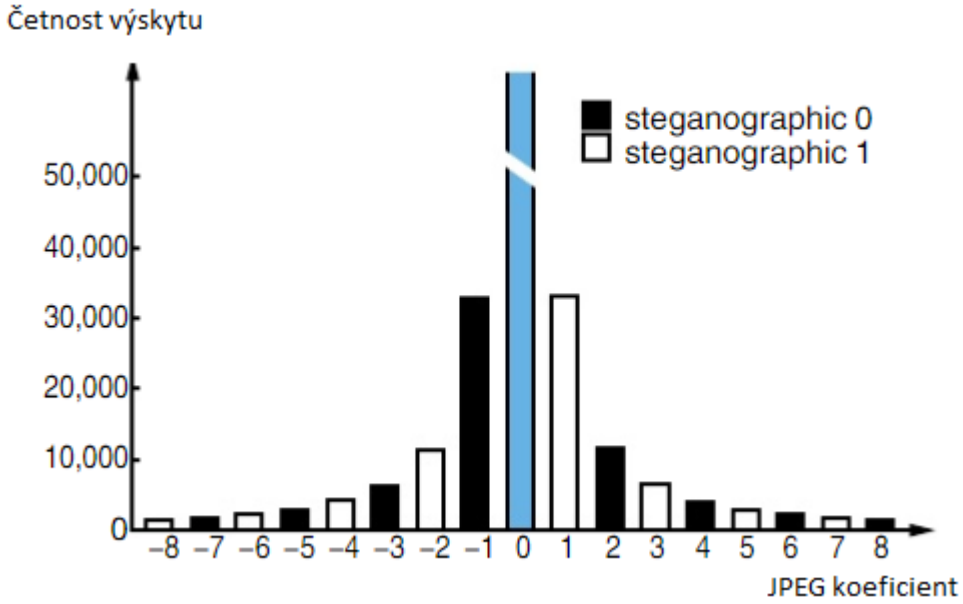
Obr. 14. Histogram JPEG koeficientu po aplikaci algoritmu F3 [13].

2.2.5 Metoda F4

Zásadní slabinou algoritmu F3 je fakt, že je stejně jako JSTEG statisticky detekovatelný z histogramu JPEG koeficientů. Je to zapříčiněno tím, že produkuje více nul než jedniček [13].

Algoritmus F4 eliminuje tuto slabinu tím, že mapuje záporný koeficient na inverzní steganografickou hodnotu:

- Sudý záporný koeficient reprezentuje steganografickou 1
- Lichý záporný koeficient reprezentuje steganografickou 0
- Sudý kladný koeficient reprezentuje steganografickou 0
- Lichý kladný koeficient reprezentuje steganografickou 1



Obr. 15. Histogram JPEG koeficientů s interpretací F4 steganografických hodnot [13].

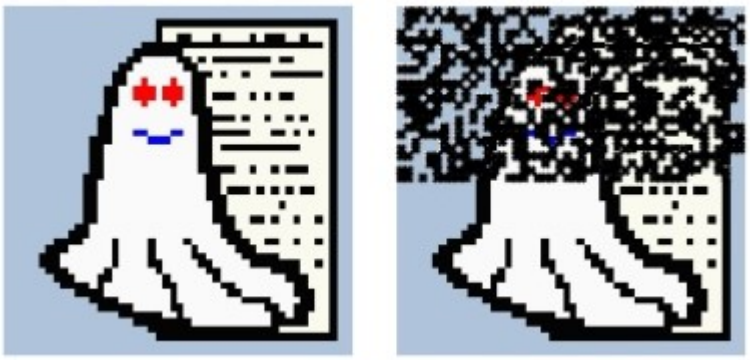
Zdrojový Java kód pro vkládací funkci algoritmu F4:

```
int nextBitToEmbed = embeddedData.readBit();
for (int i=0; i<coeff.length; i++) {
if(coeff[i] == 0) continue; // vynechat 0 koeficient
if (coeff[i] > 0) {
if ((coeff[i]&1) != nextBitToEmbed)  coeff[i]--; // dekremetovat koeficient
} else {
if ((coeff[i]&1) == nextBitToEmbed) coeff[i]++; // inkremtovat koeficient
}
if (coeff[i] != 0) { // successfully embeddedif
 if (embeddedData.available()==0) break;
```

nextBitToEmbed = embeddedData.readBit();}}

2.2.6 Metoda F5

Na rozdíl od streamových médií (například videokonference) poskytují obrazové soubory pouze omezenou kapacitu. V mnoha případech vkládaná data nepotřebují využít celou kapacitu souboru. Z toho důvodu zůstává část souboru nevyužita (znázorněno na obrázku č. 16). Pro prevenci detekce je důležité, aby vkládací funkce rozprostřela tajná data pravidelně po celém objektu. Toto zajišťuje algoritmus F5 [12].



Obr. 16. Skrytá data vkládána od začátku souboru.

Aby se předešlo problém s kontinuálním vkládáním, tak algoritmus F5 používá techniku nazývanou „Permutative Straddling“ založenou na permutaci (k rozptýlení dat po celém objektu). Mechanismus nejprve zamíchá všechny koeficienty použitím permutace. Poté algoritmus vkládá informace do permutované sekvence (počet koeficientů se nemění, mění se pouze jejich hodnoty) způsobem stejným jako F4. Permutace je závislá na klíči, který je vygenerovaný z hesla. F5 dodává steganograficky změněné koeficienty v původní sekvenci do Huffmanova kódéru. Se správným klíčem bude příjemce schopen opakovat permutaci [15].

Pro minimalizaci změn používá algoritmus F5 takzvanou matici kódování.

Překládejme, že chceme vložit 2 bity x1 a x2. Bity mohou být vloženy na místo a1, a2 nebo a3, tak aby bylo změněn pouze jeden bit.

$x1 = a1 \oplus a3, \qquad x2 = a2 \oplus a3 => nic\ se\ nemění$

$x1 \neq a1 \oplus a3, \qquad x2 = a2 \oplus a3 => změna\ a1$

$x1 = a1 \oplus a3, \qquad x2 \neq a2 \oplus a3 => změna\ a2$

$x1 \neq a1 \oplus a3, \qquad x2 \neq a2 \oplus a3 => změna\ a3$

Ve všech čtyřech případech není potřeba měnit více než 1 bit [13].

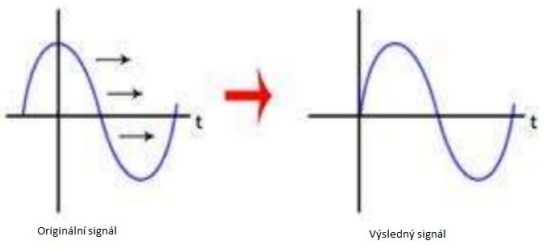
2.3 Audio steganografie

Audio steganografie se zabývá vkládáním tajných zpráv (dokumentů, obrázků atd.) do audio souborů typu WAV, MP3 aj. Audio steganografie je založena na tom, že lidský sluch nerozezná mírné variace zvukových frekvencí. Do audio souborů lze vložit tajnou zprávu bez toho, aniž by byla poškozena kvalita zvukového souboru nebo došlo ke změně velikosti souboru.

Pro steganografii ve zvukových souborech se také používá metoda LSB, kdy dojde k nahrazení nejméně významného bitu v každém vzorku zvukového souboru. Tuto malou změnu nedokáže lidský sluch identifikovat [17].

Níže je přehled nejpoužívanějších technik pro audio steganografii.

- **Skrytí dat v ozvěně** – v této technice jsou skrytá data vložena do audio souborů jako krátká akustická ozvěna. Ozvěna je v podstatě replikace původního zvuku, který je posluchači doručen s krátkým zpožděním. Jestliže je ozvěna slyšitelná, tak musí být její amplituda snížena tak, aby se stala pro lidský sluch nepostřehnutelnou. Bity tajné zprávy s hodnotou 0 jsou representovány jako ozvěna se zpožděním 1ms. Bity s hodnotou 1 jsou representovány jako ozvěna se zpožděním 2ms. Hlavní nevýhodou této techniky je nízká kapacita pro tajná data [6].
- **Kódování fáze** – tato technika nahrazuje fázi zvukového vzorku fází, která vyjadřuje tajná data. Algoritmicky je zvukový signál rozdělen do menších vzorků, jejichž velikost je stejná jako velikost tajných dat. Poté je aplikována DFT (diskrétní Furierova transformace) a vygeneruje se matice fází. Fáze v této matici jsou změněny v závislosti na datech tajné zprávy. Nakonec je audio signál obnoven výpočtem z původní matice a z matice, která obsahuje změněné fáze.



Obr. 17. Signál po aplikaci techniky kódování fáze.

- **Diskrétní vlnková transformace** (DWT) – tato technika ukrývá tajná data v nejméně významném bitu vypočteného koeficientu DWT [16].

2.4 Textová steganografie

Skrytí tajné zprávy v textu může být provedeno různými způsoby. Některé techniky mění originální text (například přidáním bílých znaků) a jiné jsou zase založeny na relaci originálních znaků se znaky tajné zprávy (například vytvoření slovníku, který mapuje slova tajné zprávy se slovy v originálním textu) [4].

Níže je přehled nejpoužívanějších technik pro skrytí dat v textu.

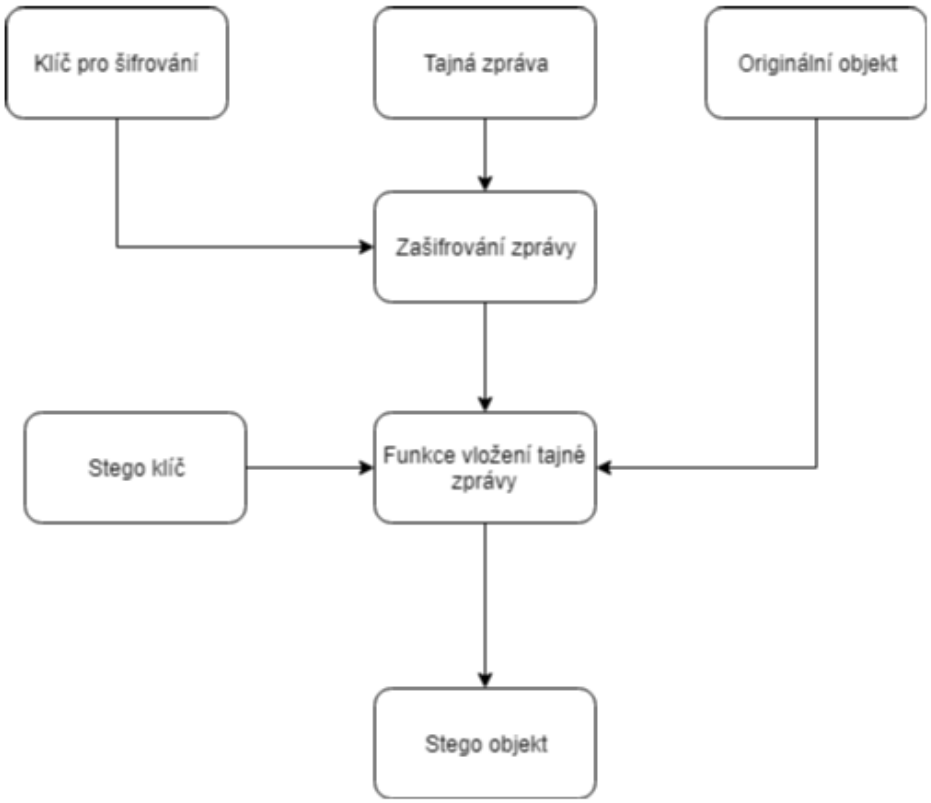
- **Selekce** – tato technika je založena na výběru určitého znaku slova originálního textu, který představuje znak tajné zprávy. Například první znak každého slova originálního textu představuje následující znak tajné zprávy. Nevýhodou této techniky je fakt, že i pro skrytí malé zprávy je zapotřebí velké množství textu.
- **Změna znaků** – tato technika je založena na změně znaků originálního textu. Používá se například u HTML dokumentů, protože HTML tagy nejsou case-sensitive – například tag <a title=”Text”> a <a TITle=”Text”> je stejný a nemá vliv na výsledné renderování dokumentu.
- **Bílé znaky** – tato technika je založena na vložení nových bílých znaků do originálního textu. Tajná data jsou převedena do binární podoby a následně vložena do originálního textu jako bílé znaky. Například každý bit s hodnotou 1 může představovat jednu mezeru navíc. V případě bitu s hodnotou 0 není mezera přidána. Pokud chceme vložit tajnou zprávu 1010 do textu „Děti šly dnes do školy“, tak výsledný stego text bude vypadat „Děti šly dnes do školy“.
- **Metoda zkratek** – tato metoda používá lexikální slovník, který obsahuje slova spolu s jejich zkratkami. Zkratky jsou označeny 0 nebo 1. Je-li slovo originálního textu v tomto slovníku, tak se nahradí danou zkratkou jejíž hodnota odpovídá tajným datům [17].

3 STEGANOGRAFIE ŘÍZENÁ KLÍČEM

Steganografie řízená klíčem je taková steganografie, která vyžaduje pro komunikaci předchozí výměnu tajného klíče (stego key). Algoritmus založený na steganografii s tajným klíčem vkládá tajnou zprávu do daného objektu na základě stego klíče. Pouze ty strany, které znají tajný klíč jsou schopné reversním procesem získat tajnou zprávu. V případě, že je steganografie detekována, tak je zapotřebí k extrakci zprávy získat stego klíč [16].

3.1 Proces

Steganografie řízená klíčem bývá často kombinována s kryptografií. Tajná zpráva je nejdříve zašifrována daným šifrovacím algoritmem a poté je pomocí tajného klíče vložena do určitého objektu. Stego klíč určuje místo, kde bude zpráva vložena. Což poskytuje vícevrstvé zabezpečení a jistotu, že se tajná zpráva v originální podobě nedostane ke straně, které nebyla původně určena. V tomto případě musí mít příjemce stego klíč pro extrakci tajné zprávy a klíč pro dešifrování zprávy. Důležitá je také znalost algoritmu, kterým je zpráva do objektu integrována [18].



Obr. 18. Proces steganografie řízené klíčem.

Na obrázku č. 17 je znázorněn proces steganografie řízené klíčem s využitím kryptografie. K získání tajné zprávy příjemce postupuje reversním procesem. Je potřeba použít stego klíč pro úspěšné extrahování zprávy a následné zprávu pomocí klíče dešifrovat. V praxi se pro šifrování zprávy používá asymetrická kryptografie, která nevyžaduje výměnu klíče mezi odesílatelem a příjemcem. Veřejným klíčem je zpráva zašifrována a pouze privátní klíč ji dokáže dešifrovat [14].

3.2 LSB řízená klíčem

Jak již bylo řečeno, LSB je jedním z nejznámějších a nepopulárnějších algoritmů steganografie. Změna nejméně významného bitu je využívána v několika dalších algoritmech steganografie – například JSTEG, F5 apod.

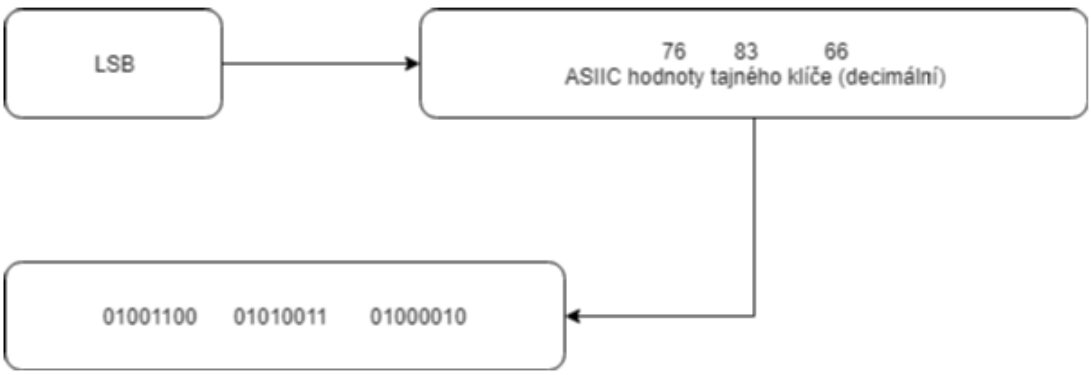
Tuto metodu lze úspěšně využít také ve steganografii řízené klíčem.

Příklad:

U obrázku s hloubkou 24 bitů se každý pixel skládá ze 3 bytů (24 bitů). Za předpokladu, že bychom chtěli skrýt následujících 9 bitů 101101101, tak pro to potřebujeme první 3 pixely obrázku. V každém pixelu dojde ke změně nejméně významného bitu každého bytu. Takto funguje nejjednodušší aplikace LSB metody.

U metody LSB, která je řízená klíčem rozhoduje o originálním bitu, který bude nahrazen stego klíč.

Mějme stego klíč „LSB“, který je v binární podobě reprezentován sekvencí bitů 010011000101001101000010.

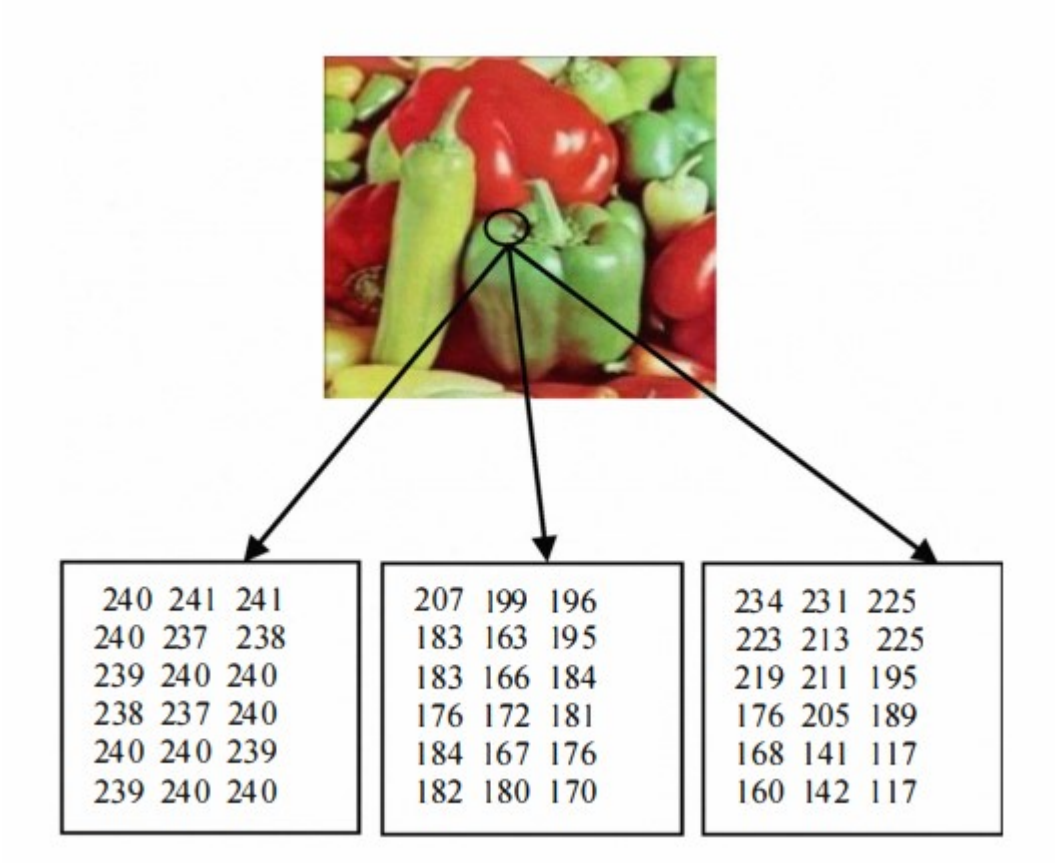


Obr. 19. Binární reprezentace stego klíče „LSB“.

Tajná zpráva, kterou budeme skrývat, je reprezentována sekvencí bitů:

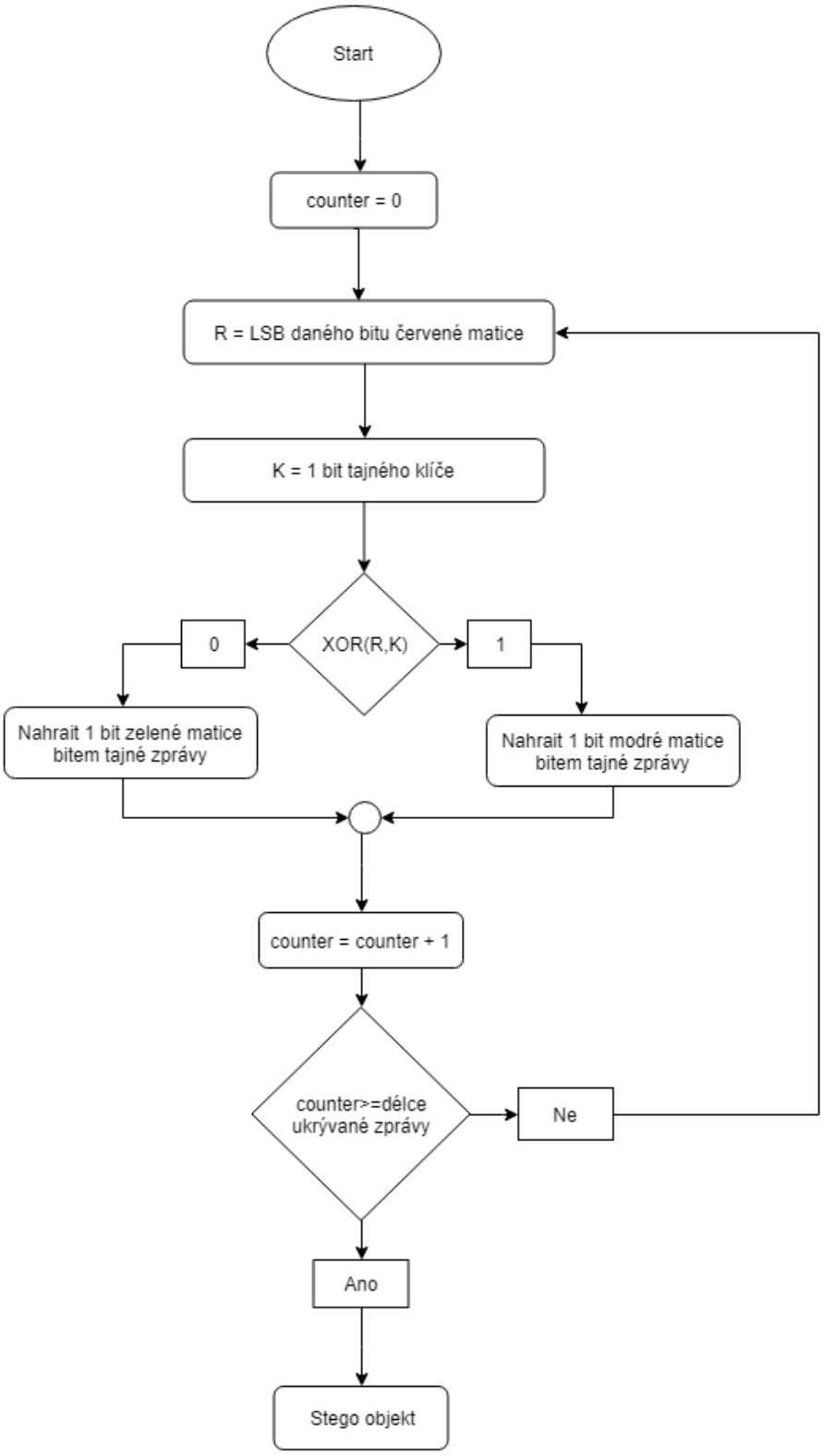
01110011 01101011 01101111 01101100 01100001

Objektem, do kterého bude naše zpráva pomocí stego klíče vložena je 24-bitový obrázek. Každý pixel tohoto obrázku obsahuje 3 byty, které představují barvu – červenou, zelenou a modrou. Obrázek si rozdělíme do tří matic. První z matic obsahuje byty červené barvy. Druhá obsahuje byty zelené barvy. Třetí obsahuje byty modré barvy. Získali jsme 3 matice, které jsou naplněny byty (znázorněno na obrázku č. 18) [15].



Obr. 20. RGB matice reprezentující originální objekt.

Červená matice společně s tajným klíčem rozhodne, zda se změní LSB na daném indexu matice modré nebo zelené. Je provedena operace XOR nad každým bitem tajného klíče a LSB bytu na daném indexu červené matice. V případě, že XOR vrátí hodnotu 1, tak se změní LSB toho bytu, které je na daném indexu v zelené matici (na hodnotu bitu tajné zprávy). V opačném případě je změněn LSB v modré matici. Stego klíč řídí umístnění tajné zprávy v objektu [15].



Obr. 21. Proces vkládání informace pomocí LSB s tajným klíčem.

3.3 DCT řízená klíčem

Steganografie řízená klíčem je také využívána pro kompresní ztrátové formáty jako JPEG. Následující metoda je založena na vygenerování pseudonáhodné sekvence koeficientů DCT, u kterých bude změněn LSB. Sekvence je vygenerována na základě klíče.

Po získání DCT koeficientů (před aplikací entropického kódování) je potřeba vygenerovat na základě klíče sekvenci těch koeficientů, u kterých dojde ke změně LSB. Z každého DCT koeficientu je získán předposlední bit a proveden XOR s bitem tajného klíče. V případě, že je výsledná hodnota 1, tak se daný DCT koeficient přidá do výsledné sekvence. Jestliže XOR vrátí hodnotu 0, tak tento koeficient nebude nositelem tajné zprávy. Bity tajného klíče určují, které koeficienty budou nositeli tajné zprávy [18].

Kroky algoritmu:

Krok-1: Získání DCT koeficientů obrázku

Krok-2: Získání sekvence koeficientů, které budou měněny – na základě XOR operace předposledního bitu koeficientu a daného bitu klíče

Krok-3: Nahrazení LSB koeficientů ve výsledné sekvenci

Krok-4: Vytvoření stego objektu

Pro obnovení tajné zprávy ze stego objektu je zapotřebí získat na základě klíče sekvenci těch koeficientů, jejichž LSB obsahuje tajnou zprávu [16].

3.4 Shrnutí

V podstatě veškeré známé metody čisté steganografie (steganografie, která nepoužívá tajný klíč) lze použít také pro steganografii řízenou klíčem. Jde pouze o vývoj algoritmu, který na základě klíče rozhoduje, které bity se budou měnit nebo vkládat – viz příklad DCT a LSB metody řízené klíčem.

Steganografie řízená klíčem poskytuje vícevrstvé zabezpečení tajných dat. V kombinaci s kryptografií je velice obtížné získat tajná data ze stego objektu bez znalosti daných klíčů. V této kapitole byl vysvětlen princip a proces steganografie řízené klíčem s příkladem dvou implementací [18].

II. PRAKTICKÁ ČÁST

4 NÁVRH ALGORITMU

Cílem této diplomové práce je návrh nového algoritmu pro implementaci steganografie řízené klíčem a následná implementace tohoto nového algoritmu.

Navrhovaný algoritmus patří do kategorie substitučních metod steganografie a je tak založen na nahrazení určitých bitů v objektu. Pracovní název tohoto algoritmu je „Seven parts of cover image“ (sedm částí krycího objektu, dále jen SPCI).

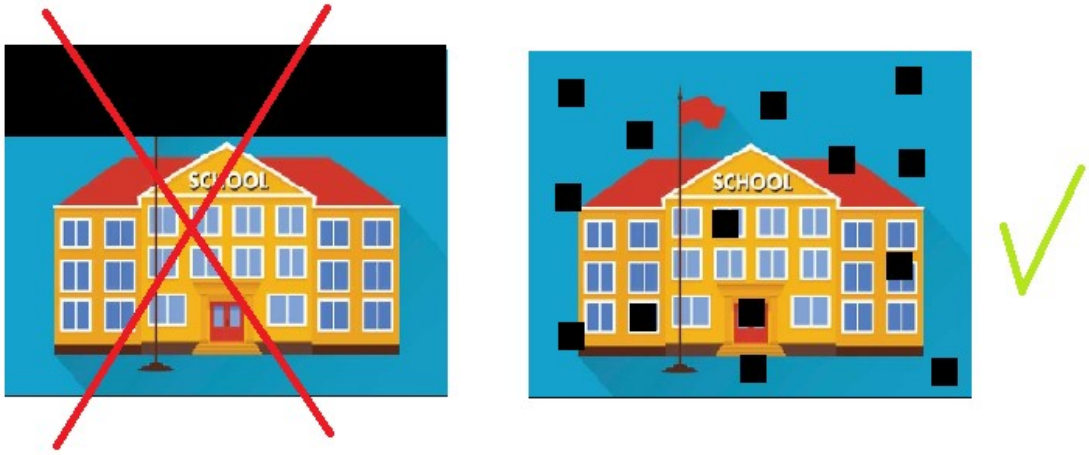
Algoritmus SPCI implementuje steganografii řízenou klíčem dle následující formule:

- (originální objekt) + (tajný klíč) + (tajná zpráva) = (výsledný stego objekt)

Kritéria zohledněná při návrhu:

- Rozprostřít vkládaná data po celém objektu
- Řídit nahrazení originálních bitů tajným klíčem
- Řídit rozprostření vkládaných bitů tajným klíčem

Důležitým prvkem je tajný klíč, podle kterého jsou určovány bity originálního objektu, jež budou nahrazeny tajnou zprávou. Bez znalosti klíče není možné rekonstruovat původní zprávu ze stego objektu. SPCI metoda řeší rozprostření zprávy po celém objektu tak, aby například tajná zpráva o velikosti 10 bitů nebyla zaintegrována ve stego objektu v prvních 10 bytech (viz obrázek č. 22).



Obr. 22. Rovnoměrné rozprostření vkládaných dat po celém objektu.

Proces algoritmu SPCI (znázorněn na obrázku č. 24):

Krok-1: Získání bytů tajných dat a bytů, které popisují objekt tajných dat (formát)

Krok-2: Z těchto bytů vytvořit sekvenci bitů S, která reprezentuje tajná data

Krok-3: Získání tajného klíče a převedení do binární podoby

Krok-4: Načtení originálního objektu do sekvence bytů S1

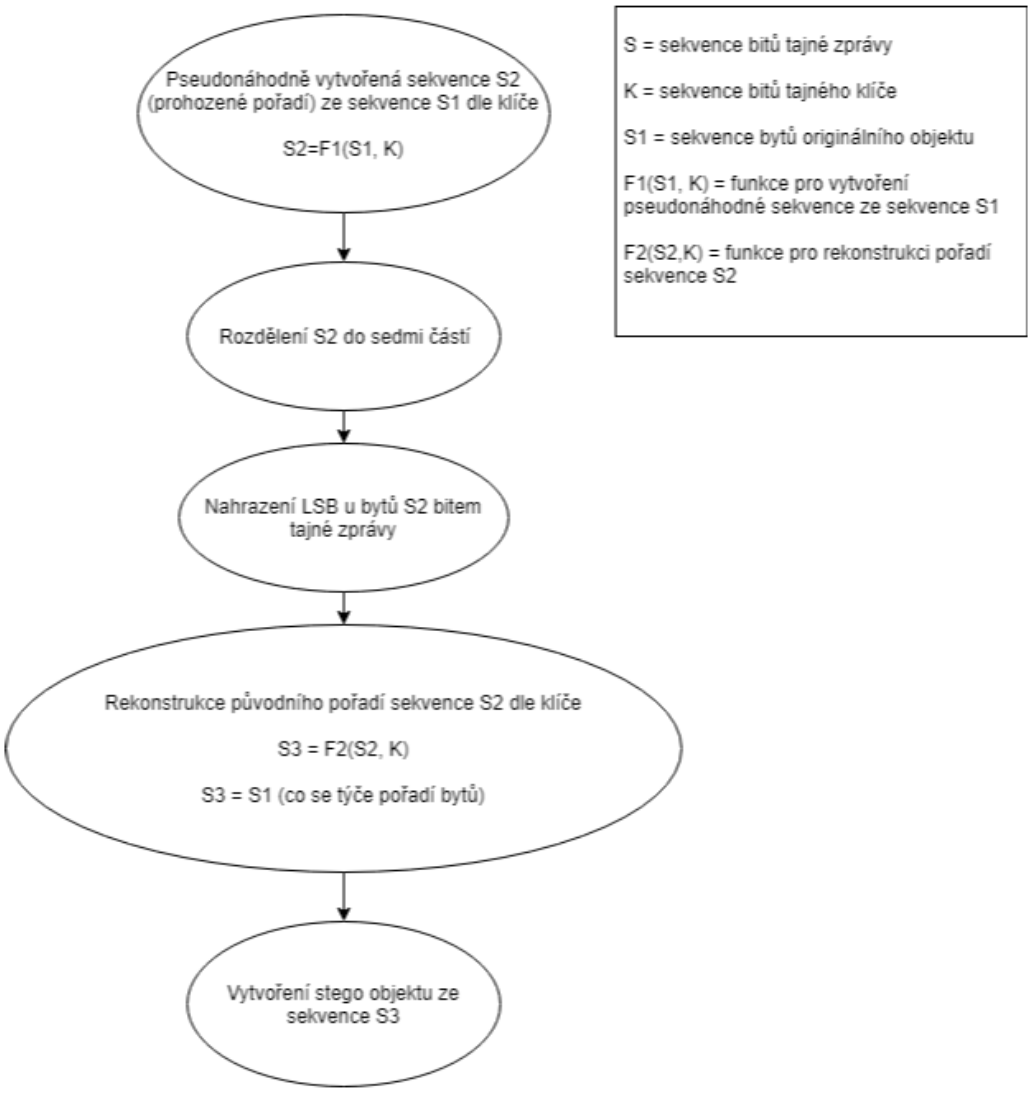
Krok-5: Dle klíče pseudonáhodně zamíchat sekvenci S1 (vysvětleno v kapitole 4.1) a vytvořit novou sekvenci bytů S2

Krok-6: Rozdělení pole S2 do sedmi částí

Krok-7: Dle klíče určit byty ze sekvence S2, u kterých bude nahrazen LSB bitem z tajné zprávy (vysvětleno v kapitole 4.2)

Krok-8: Dle klíče rekonstruovat sekvenci S2 do původní sekvence S1

Krok-9: Vytvoření stego objektu



Obr. 24. Proces algoritmu SPCI.

Pro zajištění vícevrstvé bezpečnosti je možné tajnou zprávu před samotným vkládáním zašifrovat a případně ještě kompresovat.

Pro úspěšnou rekonstrukci tajné zprávy ze stego objektu je zapotřebí výměna tajného stego klíče. Případně také klíč pro dešifrování zprávy. Proces extrahování zprávy ze stego objektu je v podstatě stejný jako při vkládání – s tím rozdílem, že klíčem určené bity nejsou nahrazovány, ale pouze přečteny.

4.1 Vstupní parametry

Vstupními parametry algoritmu SPCI pro vložení tajných dat jsou:

- Originální objekt
- Tajná data
- Informace o formátu a velikosti tajných dat (například .docx o velikosti 18 kB)
- Tajný klíč

Výsledná sekvence bytů, která se bude vkládat je vytvořena z bytů tajných dat a z bytů, které popisují tajná data (vytvořeno ze vstupního parametru). Díky tomu je z hlediska kapacity potřeba počítat s tím, že k samotným tajným datům ještě přibyde informace o jejich formátu a velikosti. Dojde k nepatrnému snížení kapacity pro tajná data. Na druhou stranu je při rekonstrukci tajných dat zjištěn i formát a může tak být jednoduše vytvořen původní objekt tajných dat. Například v případě plain textu je potřeba rekonstruovaná data konvertovat do stringu (neukládat výsledné byty do souboru), v případě WAW souboru je potřeba rekonstruovaná data zapsat na disk s příponou .WAW atd.

Vstupními parametry pro extrakci tajných dat ze stego objektu:

- Stego objekt
- Stego klíč

Algoritmus vrací výsledné byty tajných dat a informace o tajných datech.

4.2 Pseudonáhodná změna pořadí

K zajištění rozprostření vkládaných dat po celém objektu algoritmus SPCI zamíchá pořadím originálních bytů objektu a vytvoří novou sekvenci bytů, se kterou dále pracuje. Po vložení tajných dat do nově vytvořené sekvence je rekonstruováno původní pořadí.

Míchání pořadím je pseudonáhodné a je řízeno tajným klíčem – stejný klíč je použit i pro určení bitů k nahrazení. Jestliže známe klíč, tak jsme schopni sekvencí dle klíče zamíchat a poté dle klíče vrátit sekvenci do původního pořadí.

Pro sekvenci S1 o délce N je potřeba vygenerovat pole čísel P1 o velikosti N. Každé číslo v poli P1 je získáno pseudonáhodně dle klíče a nachází se v intervalu (0, N-1). Poté iterujeme všemi prvky v poli P1 a dle jeho hodnot vytváříme novou sekvenci S2.

Příklad:

Sekvenci S1 představuje pole písmen ze slova „škola“. Tajný klíčem je číslo 1024.

S1 = [š,k,o,l,a]

Dle klíče a pomocí třídy Random z .NET 4.7.2 frameworku je vygenerována pseudonáhodná sekvence čísel P1 o velikosti n = S1.Lenght (podrobněji pospáno v implementaci algoritmu).

P1 = [3,1,1,0,2]

Provedeme N iterací – viz tabulka. Výchozí hodnota i=N-1 (4). Každá iterace snižuje hodnotu i o jedničku. Iterujeme do doby, kdy i>=0. V tomto případě se jedná o 5 iterací. Výpočet je proveden v tabulce č 6 a v tabulce č.7.

Tab. 6 Změna pořadí sekvence S1 dle klíče P1.

Iterace	**š**	**k**	**o**	**l**	**a**	**Hodnota**	**Popis**
1. iterace				a	l	i = 4	x = P1[N – 1 – i] = P1[0] = 3 Prohodíme prvek S1[i] z S1[x]
2.iterace		a		k		i = 3	x = P1[N – 1 – i] = P1[1] = 1 Prohodíme prvek S1[i] z S1[x]
3. iterace		o	a			i = 2	x = P1[N – 1 – i] = P1[2] = 1 Prohodíme prvek S1[i] z S1[x]
4. iterace	o	š				i = 1	x = P1[N – 1 – i] = P1[3] = 0 Prohodíme prvek S1[i] z S1[x]
5. iterace	a		o			i = 0	x = P1[N – 1 – i] = P1[4] = 2 Prohodíme prvek S1[i] z S1[x]

Výsledek S2	**A**	**Š**	**O**	**K**	**L**		

Získali jsme novou sekvenci S2 s proházeným pořadím. Pro rekonstrukci původní sekvence S1 ze sekvence S2 postupujeme totožně. Pouze výchozí hodnota pro i = 0 a iterujeme do doby kdy i < N.

Tab. 7. Rekonstrukce původní sekvence S1 ze sekvence S2 na základě klíče P1.

Iterace	a	š	o	k	l	**Hodnota**	**Popis**
1. iterace	o		a			i = 0	x = P1[N – 1 – i] = P1[3] = 2 Prohodíme prvek S1[i] z S1[x]
2. iterace	š	o				i = 1	x = P1[N – 1 – i] = P1[3] = 0 Prohodíme prvek S1[i] z S1[x]
3.iterace		a	o			i = 2	x = P1[N – 1 – i] = P1[2] = 1 Prohodíme prvek S1[i] z S1[x]
4. iterace		k		a		i = 3	x = P1[N – 1 – i] = P1[1] = 1 Prohodíme prvek S1[i] z S1[x]
5. iterace				l	a	i = 4	x = P1[N – 1 – i] = P1[0] = 3 Prohodíme prvek S1[i] z S1[x]
Výsledek S1	**Š**	**K**	**O**	**L**	**A**		

4.3 Určení bitů k substituci

Po načtení originálních bytů a následném promícháním pořadí dle klíče je potřeba určit byty, do kterých bude vložena tajná zpráva.

Algoritmus SPCI rozdělí promíchanou sekvenci S2 o velikosti N na sedm částí C0 až C6 (pořadí bytů je zachováno) o velikosti N1 = N/7. V případě, že N1 není celým číslem, tak se N1 zaokrouhlí dolů a poslední část C6 bude mít velikost N2 = N – (N1 * 6).

Příklad:

S2 = [10,250,115,3,5,8,7,4,1,4,55,886,22,554,77,88,22,11,44,55,66,88]

N = 22

N1 = 22 / 7 = 3.1 => zaokrouhlení dolů = 3

N2 = N – (N1 * 6) = 22 – (6*3) = 4

C0 = [10,250,115]

C1 = [3,5,8]

C2 = [7,4,1]

C3 = [4,55,886]

C4 = [22,554,77]

C5 = [88,22,11]

C6 = [44,55,66,88]

Proces vkládání je rozdělen do 3 fází v závislosti na velikosti vkládané zprávy a originálního objektu – přičemž ve většině případů fáze 2 a 3 probíhat nebude.

Poté co jsou inicializovány pole C0 až C6, tak je možné vkládat tajná data. Data jsou vkládána na základě tajného klíče, který je postupně opakován do doby, dokud není zpráva kompletně vložená.

I. Fáze

Algoritmus iteruje všemi byty v poli C0. Z každého bytu jsou vytaženy všechny bity mimo LSB – tzn. celkově 7 bitů. Načte se příslušných 7 bitů tajného klíče – v případě první iterace je to první až sedmý bit klíče (druhá iterace 8 až 14 bit klíče atd.). Provede se operace XOR nad bity z C0 a bity tajného klíče. Zjistíme indexy, kde hodnota XOR je 1 a dle těchto indexů nahradíme LSB v daných polích C0 až C6.

Tajný klíč (sekvence bitů) K = [0,1,1,0,1,0,1,0]

Tajná zpráva (sekvence bitů) M = [0,1,1,0,1,0,0,1]

(Tajná zpráva se skládá z informací o skrývaném objektu a samotných dat objektu)

První iterace:

Krok-1: První byte z C0 = 10. V binární podobě 00001010 => načte 7 bitů (všechny mimo LSB) => x = 0000101

Krok-2: Získá 7 bitů klíče => k = 0110101

Krok-3: Provede XOR(x,k) = 01110000

Krok-4: Získá indexy, kde XOR(x,k) má hodnotu 1 => index 1 a 2.

Krok-5: Změní LSB u prvního bytu C1 bitem tajné zprávy

Krok-6: Změní LSB u prvního bytu C2 bitem tajné zprávy

(v první iteraci úspěšně vloženy 2 bity zprávy)

Druhá iterace:

Krok-1: Druhý byte z C0 = 250. V binární podobě 11111010=> načte 7 bitů (všechny mimo LSB) => x = 1111101

Krok-2: Získá 7 bitů klíče => k = 0011010

Krok-3: Provede XOR(x,k) = 1100111

Krok-4: Získá indexy, kde XOR(x,k) má hodnotu 1 => index 0 , 1 , 4 , 5 , 6 .

Krok-5: Změní LSB u druhého bytu C0 , C1, C4, C5, C6 bitem tajné zprávy

(v druhé iteraci úspěšně vloženo 5 bitů zprávy)

V momentě, kdy jsou vložena všechna data, tak je z polí C0 až C6 vytvořena sekvence S2 (v zachovaném pořadí) a ze sekvence S2 obnoveno původní pořadí sekvence S1 (dle klíče). Následně je vytvořen stego objekt.

V případě 24-bitové hloubky obrázku o velikosti 800x600 pixelů jsme v první fázi SPCI algoritmu schopni vložit data o velikosti m = 719 999 bitů (cca 90 kB):

$$n1 = (\frac{800*600*3}{7}) \doteq 205\ 714$$

n1…velikost pole C0

$$m = \frac{n1*7}{2} = \frac{205\ 714*7}{2} = 719\ 999 \text{ bitů}$$

V případě, že ve fázi jedna SPCI iteruje až k poslednímu prvku C0 a ani po této iteraci nebude mít vložena všechna data, tak pokračuje fází 2.

II. Fáze

Druhá fáze algoritmu je v podstatě totožná s fází první. S tím rozdílem, že nejsou nahrazovány byty v polích C0 až C6 na základě indexů XOR(x, k) s hodnou 1, ale s hodnotou **0**. Klíč je resetován tak, aby hodnota klíče v daných iteracích fáze 2 korespondovala s hodnotami v iteracích ve fázi 1. Iteruje se opět od začátku, postupně všemi byty v poli C0.

V případě 24-bitové hloubky obrázku o velikosti 800x600 pixelů jsme v rámci první a druhé fáze SPCI algoritmu schopni vložit data o velikosti m = 1 439 998 bitů (cca 180 kB):

$n1 = (\frac{800*600*3}{7}) \doteq 205\ 714$

n1…velikost pole C0

$m = n1 * 7 = \ 205\ 714 * 7 = 1\ 439\ 998$ bitů

V případě, že ve fázi dva bude SPCI iterovat až k poslednímu prvku C0 a ani po této iteraci nebude mít vložena všechna data, tak pokračuje fází 3.

III. Fáze

Algoritmus SPCI ve třetí fázi pracuje pouze s byty s C6, které se nachází na pozicích P, kde P > N1 - 1. Čili s těmi byty, které nadbývají v poli C6. Vždy se bude jednat o maximálně 6 bytů.

Krok-1: x = načtení 7 bitů z prvního bytu C6 na pozici N1

Krok-2: k = načtení prvních N2-N1 bitů klíče

Krok-3: XOR(k,x) a získání indexů s hodnou 1

Krok-5: nahrazení LSB dle indexu v bytech C6 na pozicích N1 až N2

Krok-6: v případě, že nedošlo k vložení celé zprávy v kroku 5, tak se získají indexy z XOR(k , x) s hodnotou 0

Krok-7: nahrazení LSB dle indexu v bytech C6 na pozicích N1 až N2

Krok-8: vytvoření stego objektu

Algoritmus SPCI je schopen využít všechny byty originálního objektu.

V případě 24-bitové hloubky obrázku o velikosti 800x600 pixelů jsme v rámci první, druhé a třetí fáze SPCI algoritmu schopni vložit data o velikosti m = 1 440 000 bitů (cca 180 kB):

$m = 800 * 600 * 3 = 1\ 440\ 000$ bitů

4.4 Omezení

Navržený algoritmus SPCI má následující omezení:

- Tajná data mohou mít maximálně tolik bitů, kolik je v originálním objektu bytů (v každém bytu je nahrazen LSB). Je potřeba počítat s tím, že k tajným datům bude přidána informace o jejich formátu a velikosti.
- Počet bytů originálního objektu musí být >= 7 (z důvodu rozdělení na 7 částí)
- Objekt, do kterého jsou tajná data vkládána nesmí používat ztrátovou kompresi

4.5 Použití

Algoritmus SPCI lze použít pro vložení tajné zprávy do objektů, jež nepoužívají ztrátovou kompresi. Kompresní algoritmy odstraňují nevýznamné bity, a právě do těchto bitů algoritmus zapisuje tajná data.

Nicméně algoritmus lze úspěšně použít například také v JPEG steganografických metodách, které modifikují bity v koeficientech vypočtených na základě diskrétní kosínové transformace. Jako výchozí sekvence pro algoritmus SPCI lze použít sekvence těchto vypočtených koeficientů.

Podporované formáty:

- Digitální obrázky – BPM, RAW, TIFF, PNG
- Digitální audio – WAW, AIFF

5 IMPLEMENTACE ALGORITMU

Implementace algoritmu SPCI je realizována v .NET Frameworku 4.7.2 v podobě knihovny – programovací jazyk C#. Implementace je univerzální, tak aby knihovna nebyla omezena na konkrétní formát (například pouze podpora vložení textového řetězce do BMP obrázku).

Pro použití knihovny je potřeba pouze přidat referenci do stávajícího projektu nebo aplikace.

5.1 Popis knihovny

Knihovna Spci.Steganography.dll se skládá z několika tříd, jejich metod a výčtových typů.

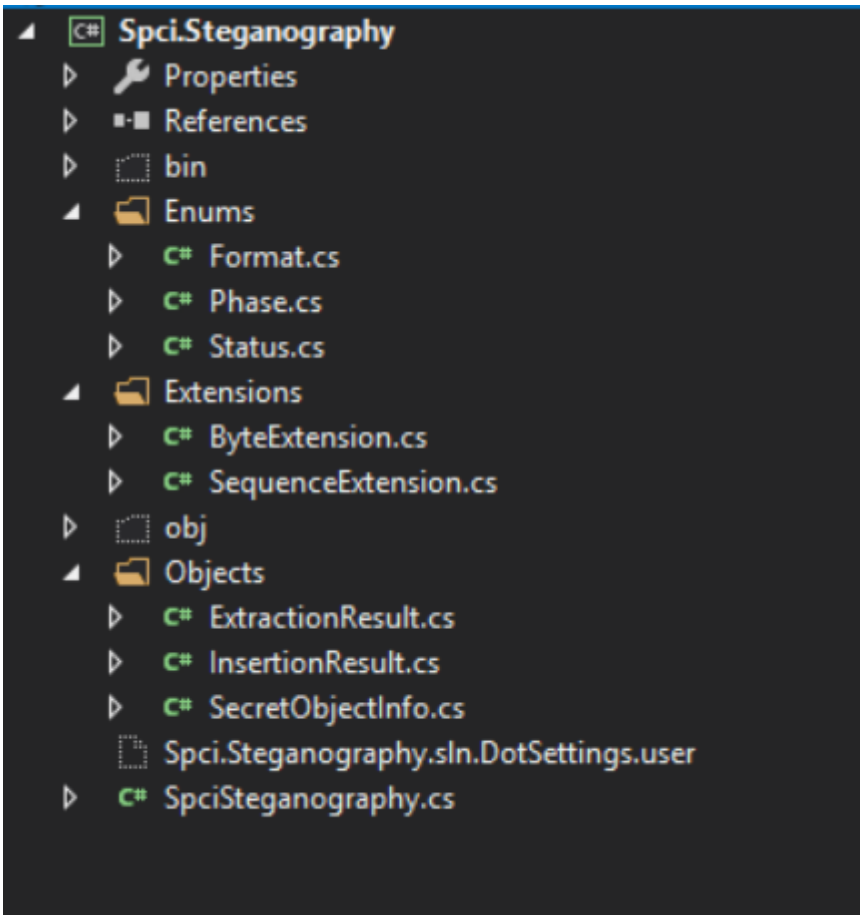
Tab. 8. Popis tříd knihovny Spci.Steganography.dll.

Třída	*Popis*
SpciSteganography	Implementuje logiku algoritmu a poskytuje tak metody pro vložení a extrakci dat. Nejedná se o statickou třídu. Je potřeba vytvořit instanci a nad ní zavolat danou metodu.
SecretObjectInfo	Třída popisuje tajná data – délku a formát. Používá se pro vložení dat a také je to jeden z objektů, který se vrací z metody, jež rekonstruuje tajná data.
InsertionResult	Poskytuje výsledky metody, která vkládá tajná data (výsledné pole bytů, počet změněných bitů, status kód a případně textovou zprávu). Jedná se o návratovou hodnotu této metody.
ExtractionResult	Poskytuje výsledky metody, která extrahuje tajná data ze stego objektu (výsledné byty, informace o tajných datech, status kód apod.). Jedná se návratovou hodnotu této metody.
ByteExtension	Statická třída, ve které jsou definované metody pro typ byte[] (extension methods), pro práci s jednotlivými bity v bytu.
SequenceExtension	Statická třída, ve které jsou definované metody pro typ byte[] (extension methods), pro účely pseudonáhodného míchání sekvence bytů.

Tab. 9. Popis výčtových typů knihovny Spci.Steganography.dll.

Výčtový typ	*Popis*
Format	Typ pro určení typu tajných dat. Používá jej třída SecretObjectInfo. Hodnoty – PlainText=1, File=2
Phase	Typ pro určení fáze, ve které se algoritmus nachází. Hodnoty – First =1, Second=2, ThirdA=3, ThirdB=4.
Status	Typ pro určení statusu metody vkládání nebo extrakce. Hodnoty – Successful=1, SecretDataIsTooLong = 2, WrongStegoKey = 3

Struktura projektu této knihovny je zobrazena na obrázku č. 25.



Obr. 25. Struktura projektu knihovny Spci.Steganography.dll.

5.2 Konstruktor

Hlavní třídou knihovny Spci.Steganography.dll je SpciSteganography. Pomocí této třídy je uživatel schopný s využitím algoritmu SPCI vložit tajná data nebo tajná data ze stego objektu rekonstruovat.

Pro vytvoření nové instance třídy SpciSteganograhy je potřeba využít definovaných konstruktorů. Konstruktory jsou dva:

- Konstruktor pro vytvoření instance pro operaci vložení tajných dat. Přijímá 3 parametry:
  - Pole bytů originálního objektu
  - Pole bytů stego klíče
  - Pole bytů tajných dat

```
/// <summary>
/// Initialize new instance for insertion of secret data
/// </summary>
/// <param name="embeddedData"></param>
/// <param name="objectData"></param>
/// <param name="stegoKey"></param>
public SpciSteganography(byte[] objectData, byte[] stegoKey,
byte[] embeddedData)
{
    _embeddedData = embeddedData;
    _objectData = objectData;
    _stegoKey = stegoKey;
}
```

- Konstruktor pro vytvoření instance pro operaci extrakce tajných dat ze stego objektu. Přijímá 2 parametry:
  - Pole bytů stego objektu
  - Pole bytů stego klíče

```
/// <summary>
/// Initialize new instance for extraction of secret data
/// </summary>
/// <param name="objectData"></param>
/// <param name="stegoKey"></param>
public SpciSteganography(byte[] objectData, byte[] stegoKey)
{
    _objectData = objectData;
    _stegoKey = stegoKey;
}
```

Po vytvoření instance dojde k automatické inicializaci hodnot (privátních), které algoritmus potřebuje k vložení tajných dat nebo k extrakci:

- Výpočet HASH kódu stego klíče
- Výpočet velikosti skupiny (7 skupin)

- Bitová velikost tajné zprávy

```
private readonly byte[] _objectData;
private readonly byte[] _stegoKey;
private byte[] _embeddedData;

private int StegoKeyHashCode => Encoding.UTF8.GetString(_stegoKey).GetHashCode();
private double GroupLength => Math.Floor((double)_objectData.Length / 7);
private double SecretDataBitLength => _embeddedData?.Length * 8 ?? 0;
```

Nad takto vytvořenou instancí je možné zavolat metodu, která provede vložení tajných dat nebo jejich extrakci.

5.3 Metoda pro vložení tajných dat

Metoda, která vkládá data je instanční a má tuto signaturu:

- `public InsertionResult EmbedSecretData(SecretObjectInfo secretObjectInfo)`

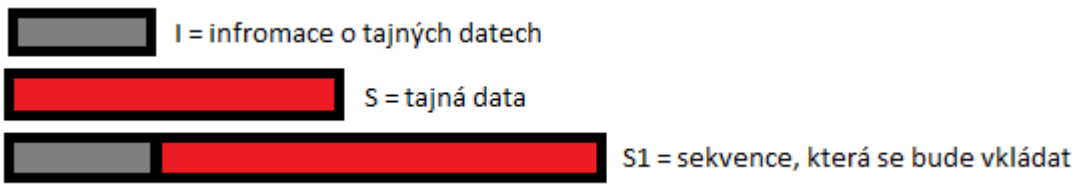
Jako parametr přijímá objekt SecretObjectInfo, který popisuje tajná data. SecretObjectInfo má tyto vlastnosti:

- FileExtension – string hodnota přípony souboru
- Length – délka tajný dat
- Format – hodnota z výčtového typu Format (PlainText nebo File)

Návratovým typem je objekt InsertionResult, který má tyto vlastnosti:

- ChangedBits – počet bitů, které bylo potřeba změnit
- StatusCode – hodnota z výčtového typu Status (Successful, SecretDataIsTooLong, WrongStegoKey)
- ResultBytes – výsledné pole bytů stego objektu
- Message – doplňující informace (string)

Ze všeho nejdříve je vytvořena nová sekvence bytů, které se budou vkládat. Tato sekvence se skládá z bytů tajných dat a z bytů z objektu SecretObjectInfo. Přičemž informace o tajných datech jsou v první – zobrazeno na obrázku č. 26.



Obr. 26. Výsledná sekvence pro vkládání.

Toto zajišťuje metoda SetSecretObjectInfo. Dále je kontrola, zda nejsou data na daný objekt příliš velká – v případě že ano, tak se vrací příslušný InsertionResult.

Následuje důležitý krok, který promíchá vstupní byty podle vypočteného HASHe stego klíče. Poté jsou inicializovány vstupní proměnné. Udržují se v paměti aktuální pozice bytů a bitů klíče a zprávy. Podle těchto hodnot je čtena zpráva a klíč.

Postupné zapisování bitů do stego objektu je realizováno cyklem while, který skončí v momentě, kdy jsou všechny bity vloženy. Podle iterace se vezme příslušný byte z promíchané sekvence a z něj 7 bitů. Pro každý bit je přidělen bit klíče a proveden XOR. Na základě výsledku operace XOR a fáze algoritmu je ne/přidán index do listu, který udržuje indexy, na kterých se budou nacházet tajná data. Potom co jsou získány indexy, tak se iteruje daným listem a dle indexů jsou zapisována tajná data.

Na konci cyklu while je kontrola, zda iterace není větší než velikost skupiny. V případě, že ano, tak se přepne fáze algoritmu a nastaví indexy klíče na výchozí hodnoty. Poté co jsou vloženy všechna tajná data, tak se cyklus while ukončí a vrací se příslušný InsertionResult.

Pro přístup k jednotlivým bitům bytu jsou naprogramované extension byte[] metody na třídě ByteExtension (kapitola 5.5).

Zdrojový kód:

```
/// <summary>
/// Embeds secret data by stego-key to the given bytes
/// </summary>
/// <param name="secretObjectInfo"></param>
/// <returns></returns>
public InsertionResult EmbedSecretData(SecretObjectInfo secretObjectInfo)
{
  var insertionResult = new InsertionResult();

  SetSecretObjectInfo(secretObjectInfo);

  if (SecretDataBitLength > _objectData.Length)
  {
     insertionResult.StatusCode = Status.SecretDataIsTooLong;
     insertionResult.Message = "Secret data is too long (maximum possible bytes: " +
                               _objectData.Length + "). Your secret data have " +
                               SecretDataBitLength / 8 + " bytes.";
     return insertionResult;
  }

  var byteSequence = _objectData.GetNewOrder(StegoKeyHashCode);
  var successfullyEmbedded = 0;
  var iteration = 0;
  var phase = Phase.First;
  var actualByteIndexKey = 0;
  var actualByteIndexMessage = 0;
  var actualBitIndexMessage = 7;
```

```
  var actualBitIndexKey = 7;
  var indexes = new List<int>();
 insertionResult.ChangedBits = 0;

 while (true)
 {
   indexes.Clear();
   var originalByte = byteSequence[iteration];
   var index = 0;
   for (var i = 7; i > 0; i--)
   {
      var xor = originalByte.IsBitSet(i) != GetBitFromKey(ref actualByteIndexKey,
                                                          ref actualBitIndexKey);
      AddIndexByXor(xor, index, phase, indexes);
      index += 1;
   }

   foreach (var i in indexes)
   {
     if (successfullyEmbedded >= SecretDataBitLength)
     {
       break;
     }
     var idx = GetByteIndex(phase, iteration, i);
     if (idx > byteSequence.Length - 1)
     {
       break;
     }

     var secretBit = GetSecretBit(ref actualByteIndexMessage,
                                  ref actualBitIndexMessage);

     SetSecretBit(secretBit, byteSequence, idx, insertionResult);
     successfullyEmbedded += 1;
    }

    if (successfullyEmbedded >= SecretDataBitLength)
    {
      break;
    }

    iteration += 1;
    if (iteration > GroupLength - 1 || phase == Phase.ThirdA)
    {
      actualBitIndexKey = 7;
      actualByteIndexKey = 0;
      phase = SetPhase(ref iteration, phase);
    }
  }

   insertionResult.StatusCode = Status.Successful;
   insertionResult.ResultBytes = byteSequence.GetOriginalOrder(StegoKeyHashCode);

   return insertionResult;
}
```

5.4 Metoda pro získání tajných dat

Metoda, která rekonstruuje tajná data ze stego objektu je instanční a má tuto signaturu:

- `public ExtractionResult GetSecretData()`

Návratovým typem je objekt ExtractionResult, který má tady vlastnosti:

- ObjectInfo – objekt typu SecretObjectInfo, nese informace o tajných datech (formát, délka)
- ExtractedBytes – výsledná sekvence bytů tajných dat
- StatusCode – hodnota výčtového typu Status (Successful, SecretDataIsTooLong, WrongStegoKey)
- SecretDataBitLenght – délka tajné zprávy v bitech

Metoda pro získání tajných dat funguje obdobně jako metoda pro zápis tajných dat (provádí se operace XOR, zjišťují indexy, přepínání fáze atd.). Jediný rozdíl je, že bity nejsou nahrazovány tajnou zprávou, ale jsou extrahovány do nové sekvence, která ve výsledku reprezentuje tajná data. Nejdříve se opět vytvoří pomocí stego klíče nová sekvence, ze které jsou bity vyčítány. Poté jsou inicializovány proměnné výchozími hodnotami (fáze, iterace, aktuální indexy klíče, aktuální indexy zprávy apod.). Metoda nejdříve hledá informace o tajných datech. Ty jsou ukryty v prvních bytech tajných dat. Poté co jsou tyto informace nalezeny, se doplní data do ExtractionResult, přenastaví se hodnota délky zprávy a vytvoří se pole o velikosti délky zprávy ve vlastnosti ExtractedBytes objektu ExtractionResult. Aktuální indexy zprávy se přenastaví na výchozí hodnoty.

V případě, že nejsou nalezeny byty, které specifikují tajná data (formát, délka), tak algoritmus vrací ExtractionResult s výsledekem špatného stego klíče. Cyklus while končí v momentě, kdy jsou vyčteny všechny bity ze stego objektu.

Zdrojový kód:

```
/// <summary>
/// Extracts secret data by stego key from the given bytes
/// </summary>
/// <returns></returns>
public ExtractionResult GetSecretData()
{
  var extractionResult = new ExtractionResult();
  var byteSequence = _objectData.GetNewOrder(StegoKeyHashCode);
  var successfullyExtracted = 0;
  extractionResult.SecretDataBitLength = _objectData.Length;
  extractionResult.ExtractedBytes =
                              new byte[extractionResult.SecretDataBitLength / 8];
```

```
  var iteration = 0;
  var phase = Phase.First;
  var actualByteIndexKey = 0;
  var actualByteIndexMessage = 0;
  var actualBitIndexMessage = 7;
  var actualBitIndexKey = 7;
  var indexes = new List<int>();

  while (true)
  {
    indexes.Clear();
    var originalByte = byteSequence[iteration];
    var index = 0;
    for (var i = 7; i > 0; i--)
    {
       var xor = originalByte.IsBitSet(i) != GetBitFromKey(ref actualByteIndexKey,
                                                           ref actualBitIndexKey);
       AddIndexByXor(xor, index, phase, indexes);
       index += 1;
    }
    foreach (var i in indexes)
    {
      if (successfullyExtracted >= extractionResult.SecretDataBitLength)
      {
         break;
      }
      var idx = GetByteIndex(phase, iteration, i);
      if (idx > byteSequence.Length - 1)
      {
          break;
      }
      var lsbValue = byteSequence[(int) idx].IsBitSet(0);

      WriteExtractedBit(ref actualByteIndexMessage, ref actualBitIndexMessage,
                        ref successfullyExtracted, extractionResult, lsbValue);

     }
     if (successfullyExtracted >= extractionResult.SecretDataBitLength)
     {
        break;
     }
     iteration += 1;
     if (iteration > GroupLength - 1 || phase == Phase.ThirdA)
     {
        actualBitIndexKey = 7;
        actualByteIndexKey = 0;
        phase = SetPhase(ref iteration, phase);
     }
  }

  if (extractionResult.ObjectInfo == null)
  {
     extractionResult.ExtractedBytes = new byte[0];
     extractionResult.StatusCode = Status.WrongStegoKey;
     extractionResult.SecretDataBitLength = 0;
  }
  else
  {
     extractionResult.StatusCode = Status.Successful;
  }
  return extractionResult;
}
```

5.5 Bitové operace

Pro práci s jednotlivými bity v určitém bytu jsou v knihovně Spci.Steganography.dll naprogramovány extension metody pro typ byte[]. Nachází se na statické třídě ByteExtension.

- IsBitSet(int position) – vrací true, jestliže má bit na dané pozici hodnotu 1
- SetBit(int position) – nastavuje bit na dané pozici na hodnotu 1
- UnsetBit(int position) – nastavuje bit na dané pozici na hodnotu 0

Knihovna přistupuje k bitům jako k boolean hodnotám – bit s hodnotou 1 je reprezentován jako true, bit s hodnotou 0 je reprezentován jako false.

Pro manipulaci s bity je použito bitových operátorů:

Tab. 10. Bitové operátory.

Symbol	**Operátor**
&	AND
|	OR
^	XOR
<<	Posun vlevo
>>	Posun vpravo
~	NOT

Díky těmto bitovým operátorům je algoritmus schopný zjistit hodnotu bitu nebo ji přenastavit.

```
/// <summary>
/// Evaluates whether is bit set in given position
/// </summary>
/// <param name="_byte"></param>
/// <param name="position"></param>
/// <returns></returns>
public static bool IsBitSet(this byte _byte, int position)
{
    return (_byte & (1 << position)) != 0;
}
```

```
/// <summary>
/// Sets bit in given position
/// </summary>
/// <param name="_byte"></param>
/// <param name="position"></param>
/// <returns></returns>
public static byte SetBit(this byte _byte, int position)
{
    return (byte)(_byte | (1 << position));
}

/// <summary>
/// Unsets bit in given position
/// </summary>
/// <param name="_byte"></param>
/// <param name="position"></param>
/// <returns></returns>
public static byte UnsetBit(this byte _byte, int position)
{
    return (byte)(_byte & ~(1 << position));
}
```

5.6 Pseudonáhodná změna pořadí

Postup, kterým je pseudonáhodně promícháno pořadí v sekvenci bytů, je podrobně pospán v kapitole 4.2. Rozveden ovšem nebyl způsob, kterým je na základě klíče získána sekvence indexů, které řídí změnu pořadí v požadované sekvenci.

Z daného tajného klíče je potřeba získat HASH, který je typu integer. HASH je pro stejný klíč vždy stejný. Potom je vytvořena instance objektu Random a v konstruktoru je předána HASH hodnota klíče.

var random = new Random(“skola”.GetHashCode());

Nad takto vytvořenou instancí můžeme volat metodu .Next(maxValue), které se předává v parametru maximální hodnota. Metoda vrátí pseudonáhodné číslo v intervalu (0,maxValue). Pseudonáhodnost zajišťuje vytvořená instance na základě čísla.

Pokud nad touto instancí zavoláme 10krát random.Next(10), tak dostaneme tyto hodnoty:

1,5,8,7,4,5,6,2,3,5

Jestliže si někdo jiný, na jiném počítači, v jiném čase vytvoří instanci Random ze stejného HASH kódu a zavolá 10krát .Next(10), tak dostane stejné hodnoty jako my:

1,5,8,7,4,5,6,2,3,5

Tímto způsobem je v algoritmu SPCI řešena pseudonáhodnost. Na zakládě HASH kódu klíče je vytvořena instance Random a z ní jsou pak získány pseudonáhodné indexy, které řídí změnu pořadí.

Metody jsou definované na statické třídě SequenceExtension:

- GetPseudoRandomIndexes – vrací pole indexů, podle kterých je měněno pořadí
- GetNewOrder – vrací novou promíchanou sekvenci na základě klíče
- GetOriginalOrder – vrací sekvenci, ve které je rekonstruováno pořadí do původní podoby na základě klíče

```
/// <summary>
/// Gets pseudo-random array of integers based on key
/// </summary>
/// <param name="size"></param>
/// <param name="key"></param>
/// <returns></returns>
public static int[] GetPseudoRandomIndexes(int size, int key)
{
    var integers = new int[size];
    var random = new Random(key);
    for (var i = size - 1; i >= 0; i--)
    {
         var n = random.Next(i + 1);
         integers[size - 1 - i] = n;
     }
     return integers;
 }

 /// <summary>
 /// Returns shuffled new sequence
 /// </summary>
 /// <param name="sequence"></param>
 /// <param name="key"></param>
 /// <returns></returns>
 public static byte[] GetNewOrder(this byte[] sequence, int key)
 {
   var size = sequence.Length;
   var bytes = sequence.ToArray();
   var randomArray = GetPseudoRandomIndexes(size, key);
   for (var i = size - 1; i >= 0; i--)
   {
       var j = randomArray[size - 1 - i];
       var tmp = bytes[i];
       bytes[i] = bytes[j];
       bytes[j] = tmp;
   }
   return bytes;
  }
```

```
/// <summary>
/// Returns new sequence with original order
/// </summary>
/// <param name="sequence"></param>
/// <param name="key"></param>
/// <returns></returns>
public static byte[] GetOriginalOrder(this byte[] sequence, int key)
{
   var size = sequence.Length;
   var bytes = sequence.ToArray();
   var randomArray = GetPseudoRandomIndexes(size, key);
   for (var i = 0; i < size; i++)
   {
       var j = randomArray[size - i - 1];
       var tmp = bytes[i];
       bytes[i] = bytes[j];
       bytes[j] = tmp;
   }
   return bytes;
 }
```

5.7 Použití

Pro použití knihovny je potřeba přidat referenci. Poté už jen vytvořit instanci třídy SpciSteganography a zavolat danou metodu.

Příklad vložení tajných dat:

```
var key = Encoding.ASCII.GetBytes("skola"); // ziskani bytu klice

var message = Encoding.ASCII.GetBytes("diplomova prace");//ziskani bytu tajne zpravy

var coverObject = "Test SPCI algoritmu. Test SPCI algoritmu. " +
                  "Test SPCI algoritmu. Test SPCI algoritmu. " +
                  "Test SPCI algoritmu. Test SPCI algoritmu. "+
                  "Test SPCI algoritmu. Test SPCI algoritmu." ;

var coverObjectBinary = Encoding.ASCII.GetBytes(coverObject); // ziskani bytu origi-
nalniho objektu

//vytvoreni nove instance SpciSteganography
var spciSteganography= new SpciSteganography( coverObjectBinary, key, message);

// vytvoreni SecretObjectInfo, ktery nese informace o tajnych datech
var objectInfo = new SecretObjectInfo()
{
   Lenght = message.Length,
   Type = Type.PlainText
};

// vlozeni tajnych dat
var result = spciSteganography.EmbedSecretData(objectInfo);
```

Příklad rekonstrukce tajných dat (proměnné z předchozího příkladu):

```
// vytvoreni nove instance
var spciSteganographyExtraction = new SpciSteganography(result.ResultBytes,key);

var extractionResult = spciSteganographyExtraction.GetSecretData();
```

6 APLIKACE

S použitím knihovny Spci.Steganography.dll byla vytvořena aplikace SpciApp, která poskytuje uživatelské rozhraní pro práci s algoritmem steganografie řízené klíčem – SPCI.

Aplikace umožňuje vložit tajná data do obrázku takového formátu, jež nepoužívá ztrátovou kompresi a má bitovou hloubku 24 bitů, nebo 32 bitů (3 byty v jednom pixelu, nebo 4 byty v jednom pixelu).

Tab. 11. Podporované formáty obrázku pro vložení tajných dat prostřednictví aplikace SpciApp.

Formát	**Komprese**
PNG	Používá kompresi, ale ta je vždy bezztrátová.
BMP	Nepoužívá kompresi.
TIFF (TIF)	Používá ztrátovou i bezztrátovou kompresi v závislosti na zvolené variantě. SpciApp podporuje pouze bezztrátovou variantu.
RAW	Bez komprese. RAW není přímo souborový formát, ale spíše jde o klasifikaci. Každý výrobce implementuje jiný formát RAW souborů (například Sony . RAW, Nikon . NEF).

Tajná data, která se budou do obrázku vkládat mohou mít podobu:

- Plain Text (prostý text) – text, který není obsahem žádného souboru
- Soubor – jakýkoliv digitální soubor, zde není žádné omezení (například dokument, spustitelný soubor, obrázek, video apod.). Soubor pouze musí být převoditelný do binární podoby, což v digitálním světe je každý soubor.

SpciApp poskytuje náhledy na originální obrázek a vytvořený stego objekt. Poté co jsou data vložena je výsledný stego objekt zobrazen a také jsou k dispozici statistická data z procesu vkládání:

- Celkový čas
- Celkový počet vkládaných bitů
- Celkový počet bitů, které bylo potřeba v originálním objektu změnit

Prostřednictvím aplikace lze také tajná data ze stego objektu získat. Je potřeba daný stego objekt a klíč (heslo). Po úspěšné extrakci tajných dat jsou zobrazeny informace o tajných datech (formát, počet bytů). V případě tajných dat v podobě plain textu, nebo obrázku jsou data přímo v aplikaci zobrazena. Jestliže jsou tajná data v jiném formátu, tak aplikace zobrazí cestu k souboru na disku.

6.1 Technická specifikace

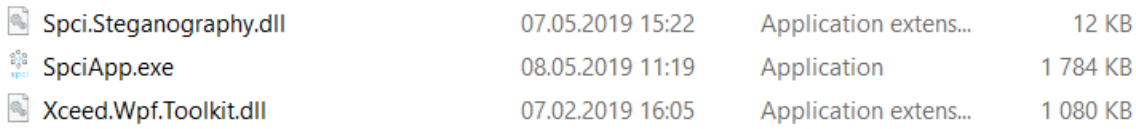
Aplikace SpciApp je vytvořena jako WPF aplikace (Windows Presentation Foundation) v prostředí .NET Framework. Konkrétně nad Framework .NET 4.7.2. Použitý programovací jazyk je C#. Aplikace je tudíž spustitelná pouze na operačním systému Windows. Uživatelské rozhraní je napsáno v jazyce XAML (Extensible Application Markup Language), což je značkovací jazyk využívaný k popisu grafického rozhraní (založeno na XML).

Tab. 12. Technická specifikace aplikace SpciApp.

Typ	WPF aplikace
Framework	.NET Framework 4.7.2
Programovací jazyk logiky	C#
Popis grafického rozhraní	XAML
Pro operační systém	Windows
Kompatibilita	Windows 7 a vyšší (podpora 64 bitové i 32 bitové verze)
Hardware požadavky	2 GB operační paměti

Aplikace nevyžaduje instalaci. Ke spuštění je potřeba těchto souborů:

- SpciApp.exe – spustitelný soubor
- Spci.Steganography.dll – knihovna s implementaci algoritmu SPCI
- Xceed.Wpf.Toolkit.dll – toolkit pro aplikace WPF



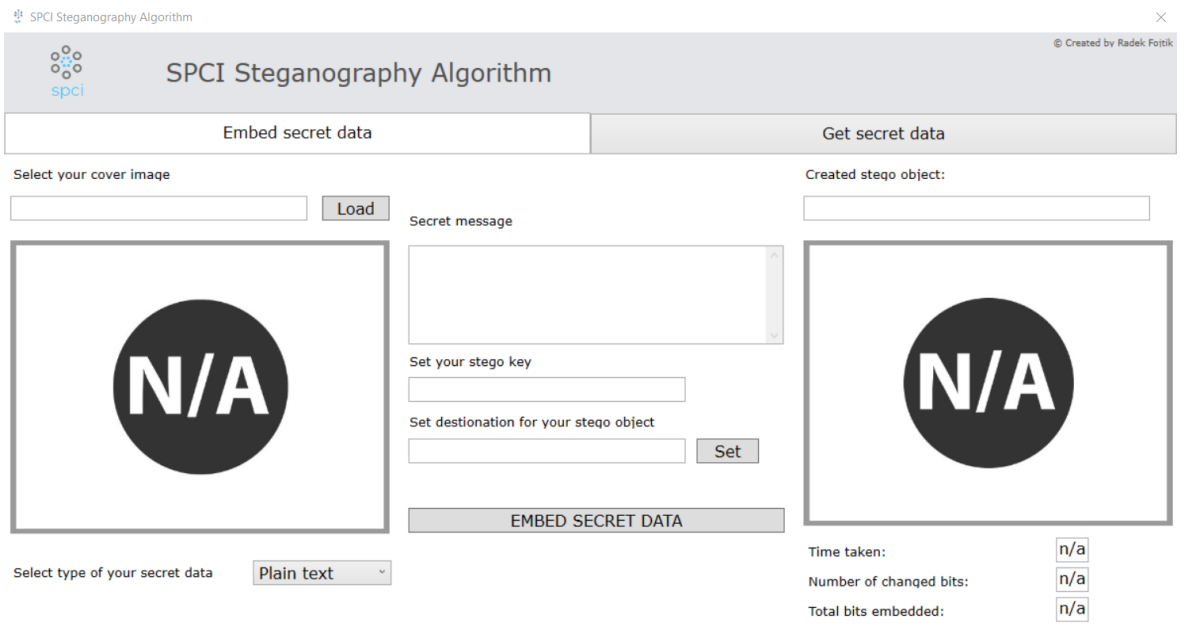
Obr. 27. Soubory aplikace SpciApp.

6.2 Popis uživatelského rozhraní

Aplikace SpciApp je rozdělena do dvou částí. První část aplikace slouží pro vložení tajných dat a vytvoření stego objektu. Druhá část slouží pro extrakci tajných dat ze stego objektu. Rozdělení je realizováno taby.

6.2.1 Vložení tajných dat

Grafické rozhraní pro vložení tajný dat je zobrazeno na obrázku č. 28.



Obr. 28. Uživatelské rozhraní aplikace SpciApp pro vložení tajných dat.

Metoda, která vkládá tajná data je spuštěna ve svém vlastním vlákně (ne v hlavním vlákně aplikace). Během procesu vkládání je zobrazen loader.

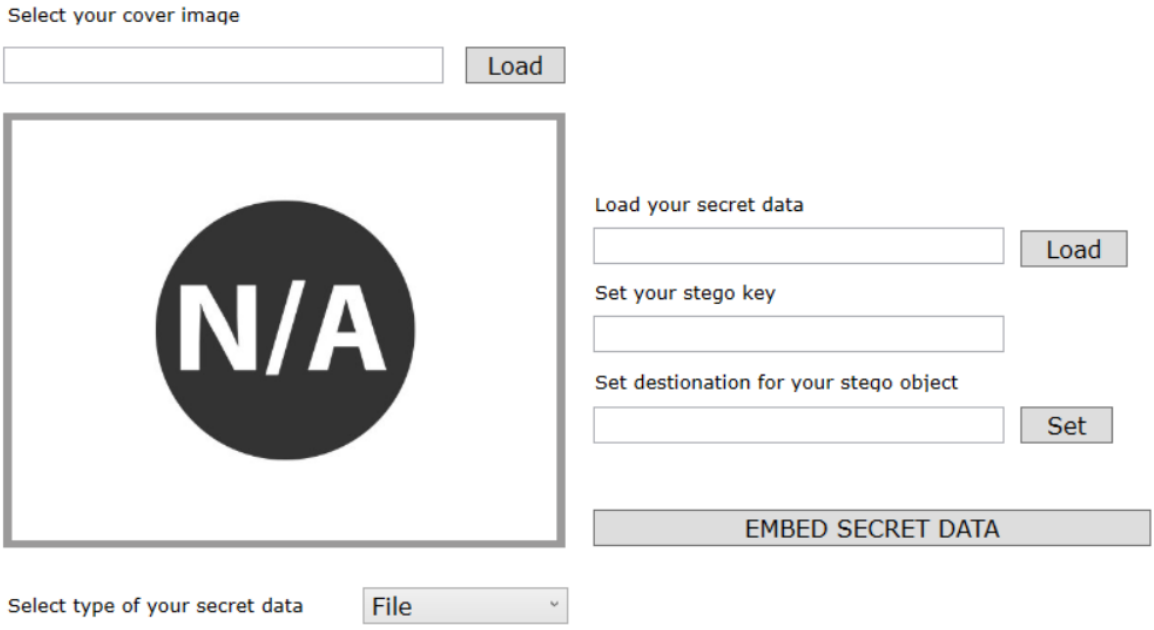
Pro úspěšné vložení tajných dat je potřeba:

- Vybrat obrázek, do kterého budou data vkládána
- Zvolit typ tajných dat (prostý text, nebo soubor)
- Zadat tajná data – vložit text nebo případně cestu k souboru
- Zadat stego klíč (heslo)
- Zadat cestu, kde se má výsledný stego objekt uložit

Po zadání těchto údajů stačí kliknout na tlačítko „EMBED SECRET DATA“ a poté dojde k vytvoření stego objektu. Jestliže nejsou vyplněny všechny tyto údaje, tak k vytvoření stego objektu nedojde (vytvořena validace – obrázek č. 28).

Obr. 29. Validace před vložením tajných dat.

V případě, že tajná data představují soubor, tak je potřeba zadat jeho umístnění na disku. Místo textového pole pro tajnou zprávu je k dispozici tlačítko pro výběr souboru z disku.



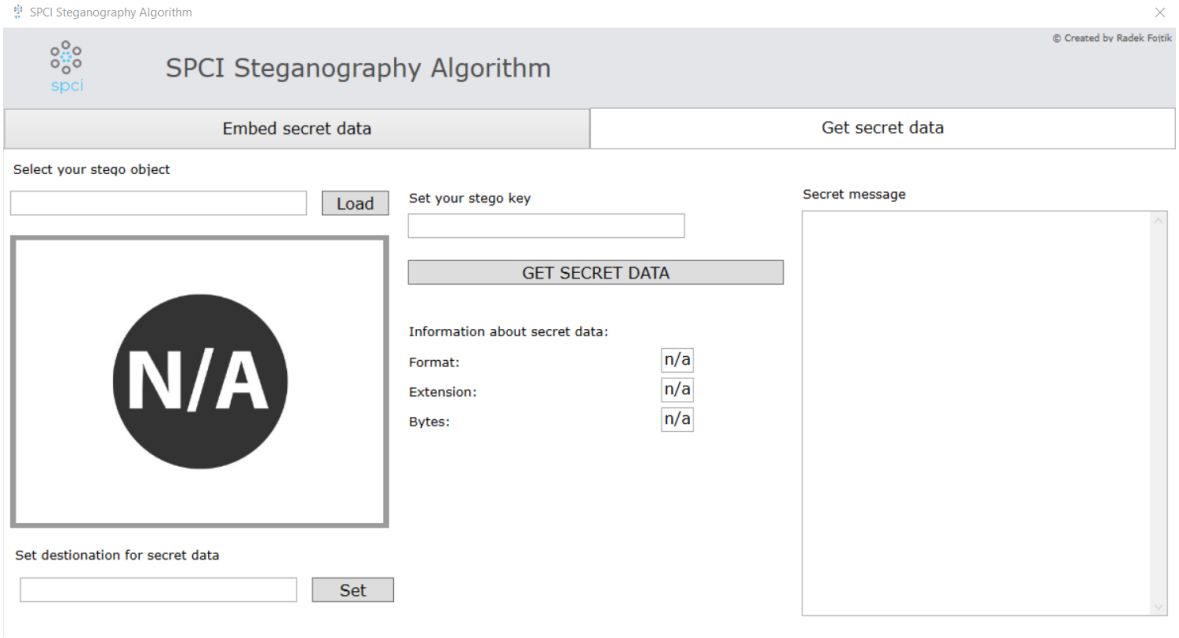
Obr. 30. Skrytá data jako soubor.

Po vytvoření stego objektu je v pravé části aplikace zobrazen jeho náhled spolu se statistickými údaji z procesu vkládání a jeho umístněním na disku.

Název výsledného stego objektu je vygenerován jako časová značka – například 636928558061087066.BMP.

6.2.2 Získání tajných dat

Grafické rozhraní pro získání tajných dat ze stego objektu je zobrazeno na obrázku č. 31.



Obr. 31. Grafické rozhraní pro získání tajných dat ze stego objektu.

Metoda, která získává tajná data je také spuštěna ve svém vlastním vlákně (ne v hlavním vlákně aplikace). Během procesu extrahování je zobrazen loader.

Pro úspěšné získání tajných dat je potřeba zadat:

- Stego objekt – vybrat ze souborů na disku
- Cestu, kde budou uložena získaná skrytá data – v případě prostého textu data ukládána nejsou (nicméně tuto informaci před samotným získáním těchto dat aplikace nemá)
- Stego klíč (heslo)

Bez těchto údajů není možné spustit metodu pro získání tajných dat (vytvořena validace). Po zadání všech povinných údajů stačí kliknout na tlačítko „GET SECRET DATA“ a poté dojde k obnovení tajných dat ze stego objektu.

Výsledná tajná data jsou zobrazena v pravé části aplikace. Pod tlačítkem „GET SECRET DATA“ jsou uvedeny informace o tajných datech. V případě prostého textu se tajná data nachází v textovém poli „Secret message“. V případě obrázku je textové pole „Secret message“ nahrazeno náhledem na tajná data v podobě obrázku (spolu s cestou k souboru na disku). V ostatních případech je uvedena pouze cesta k tajným datům na disku.

6.3 Použití

Použití aplikace SpciApp je velmi jednoduché.

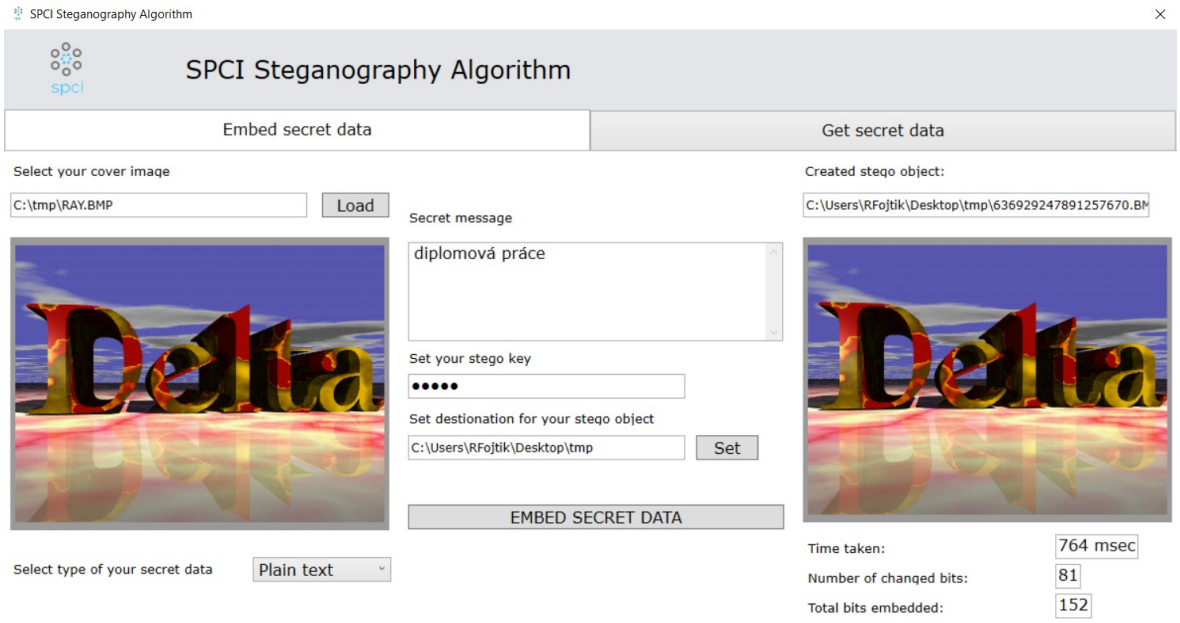
Příklad:

Originální objekt – obrázek RAY.BMP o velikosti 800x600 pixelů – 1 440 000 bytů

Tajná data – prostý text „diplomová práce“

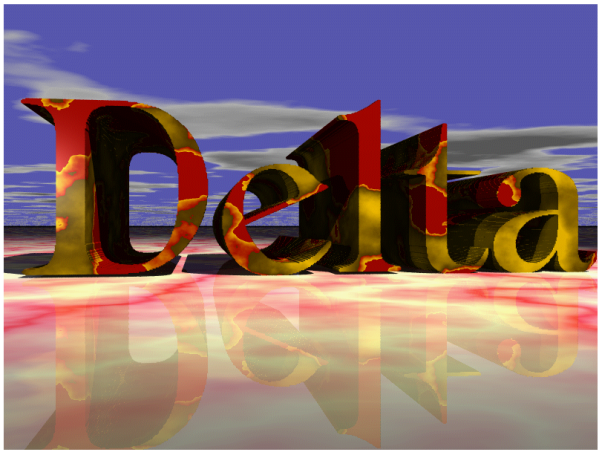
Stego klíč – „škola“

Zadáme povinné údaje a klikneme na „EMBED SECRET DATA“.



Obr. 32. Použití SpciApp pro vložení tajných dat.

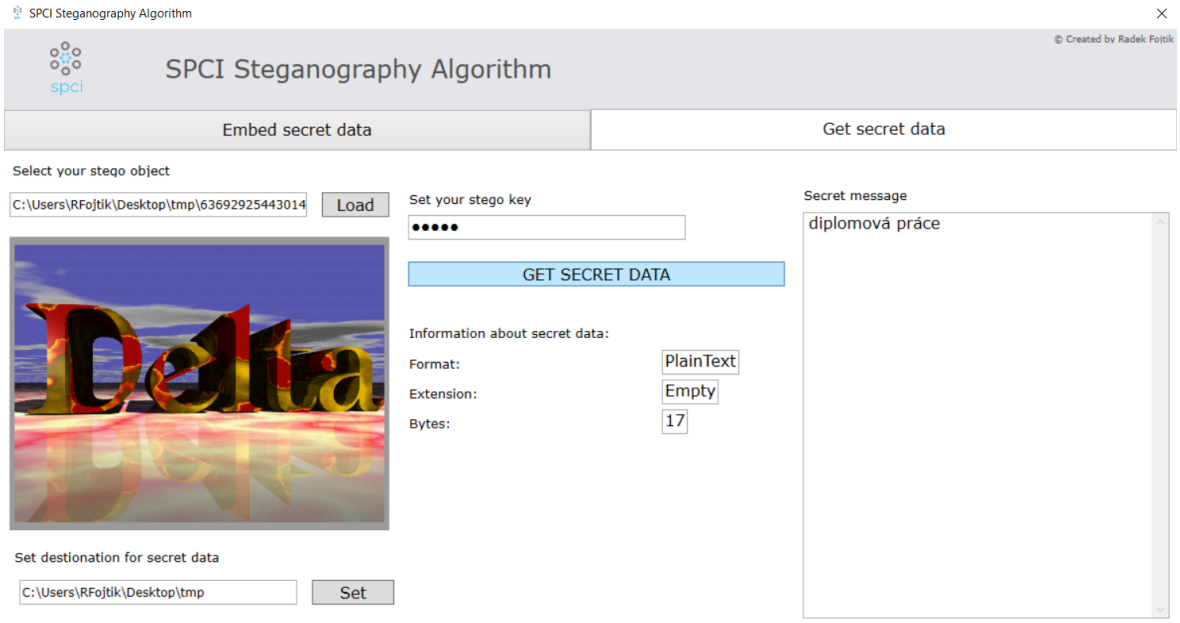
Získáme výsledný stego objekt. Níže je srovnání originálního obrázku a stego obrázku.



Obr. 33. Originální obrázek.

Obr. 34. Stego obrázek.

Vidíme, že rozdíl mezi obrázkem č. 34 a obrázkem č. 33 není lidským okem viditelný. Obrázky jsou stejně velké a liší se pouze v 81 bitech (LSB).



Obr. 35. Použití SpciApp pro získání tajných dat.

S využitím tajného klíče jsme dokázali získat tajná data ze stego obrázku.

7 ANALÝZA VÝSLEDKŮ

Za účelem analýzy výsledků algoritmu jsou vytvořeny 3 stego objekty. Každý z těchto objektů nese informaci jiného formátu a vychází z jiného originálního obrázku.

Tab. 13. Originální obrázky.

Název	Velikost	Formát
Land	1 255 KB	BMP
Mountain	1 248 KB	BMP
Venezia	3 126 KB	BMP

Obr. 36. Originální obrázky pro tajná data.

7.1 Vytvoření stego objektů

Nejdříve je potřeba specifikovat tajná data a určit, do kterého objektu (obrázku) se budou vkládat.

Tab. 14. Specifikace tajných dat a určení obrázku, do kterého se budou vkládat.

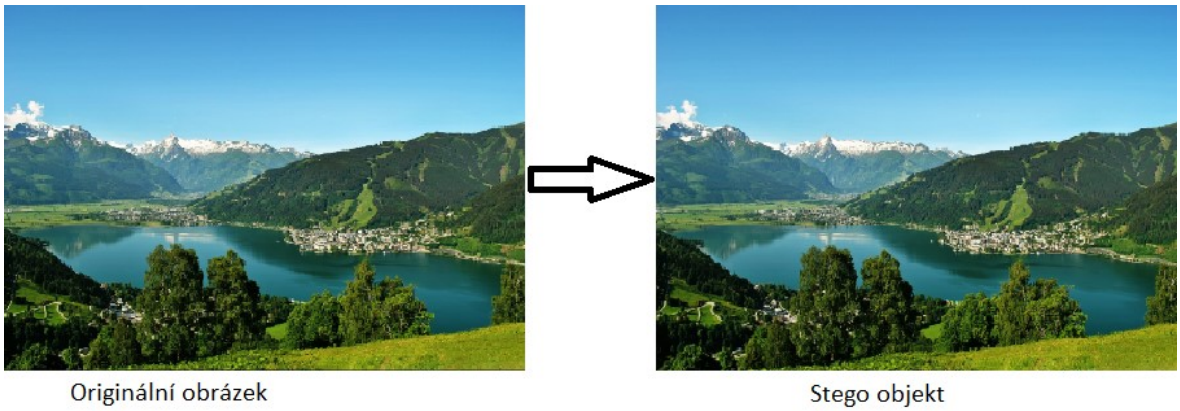
Název	Formát	Velikost	Vkládáno do obrázku
-	Plain text	302 B	Land
Rec.m4a	Zvuková nahrávka	76 980 B	Venezia
Tyger.jpg	Obrázek jpg	72 680 B	Mountain

Obr. 37. Tyger.jpg – tajná data.

První vytvořený stego objekt je Montain.BMP, který nese tajná data v podobě obrázku tygra.

Obr. 38. Vytvořený stego objekt Mountain.BMP.

Druhý vytvořený stego objekt je Land.BMP, který nese tajná data v podobě prostého textu.



Obr. 39. Vytvořený stego objekt Land.BMP.

Poslední vytvořený stego objekt je Venezia.BMP, který nese tajná data v podobě zvukové nahrávky.

Obr. 40. Vytvořený stego objekt Venezia.BMP.

7.2 Počet změněných bitů a časová náročnost

Algoritmus SPCI mění dané bity jenom v případě, že je to potřeba. Pokud je například vkládán bit s hodnotou 0 a je mu přiděleno místo v originálním bytu, kde se nachází hodnota 0, tak se v daném bytu nic nemění (statisticky lze říct, že se bude měnit pouze 50 % bitů z celkového počtu vkládaných). Hodnoty jsou shrnuty v tabulce č. 15 (použitý klíč „skola1234“).

Tab. 15. Statistické hodnoty procesu vkládání tajných dat.

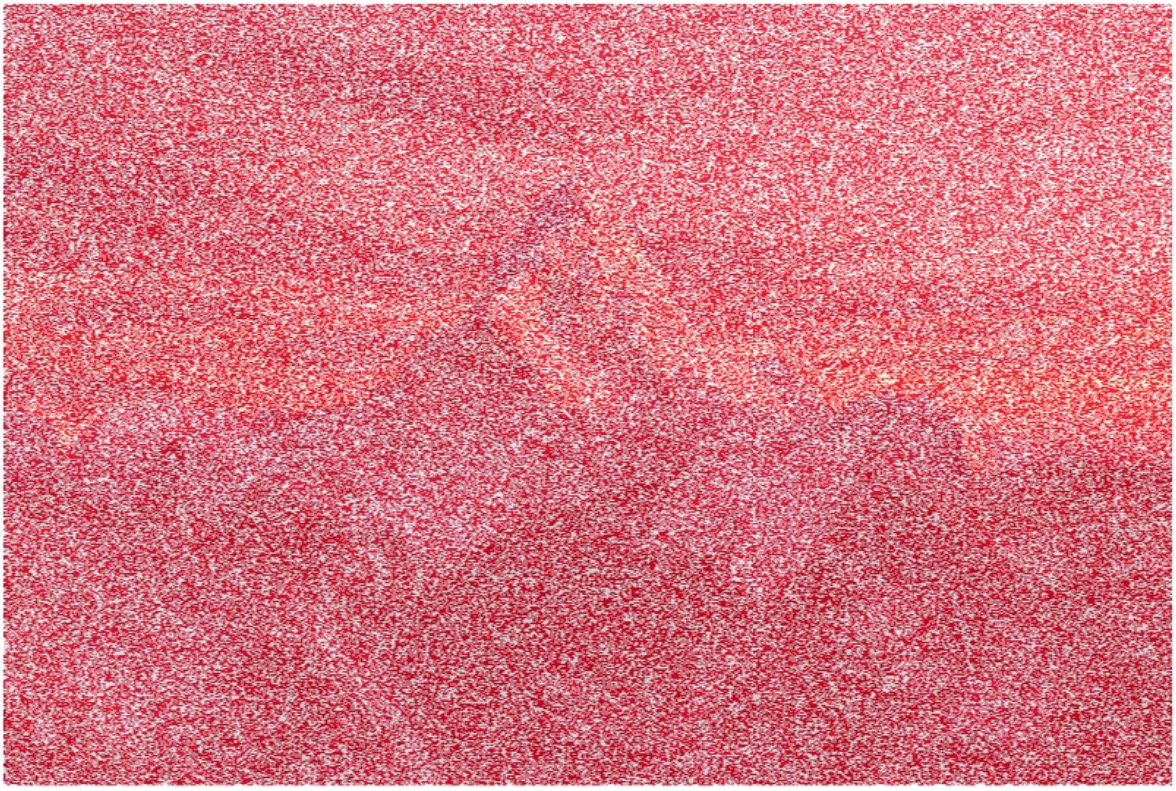
Stego objekt	**Celkem vkládaných bitů**	**Celkem změněných bitů**	**Procento změněných bitů**	**Časová náročnost**
Mountain.BMP	581 520 bit	290 671 bit	49,98 %	743 milisekund
Venezia.BMP	615 920 bit	307 498 bit	49,92 %	1650 milisekund
Land.BMP	2 664 bit	1 316 bit	49,39 %	628 milisekund

7.3 Rozložení vkládaných dat

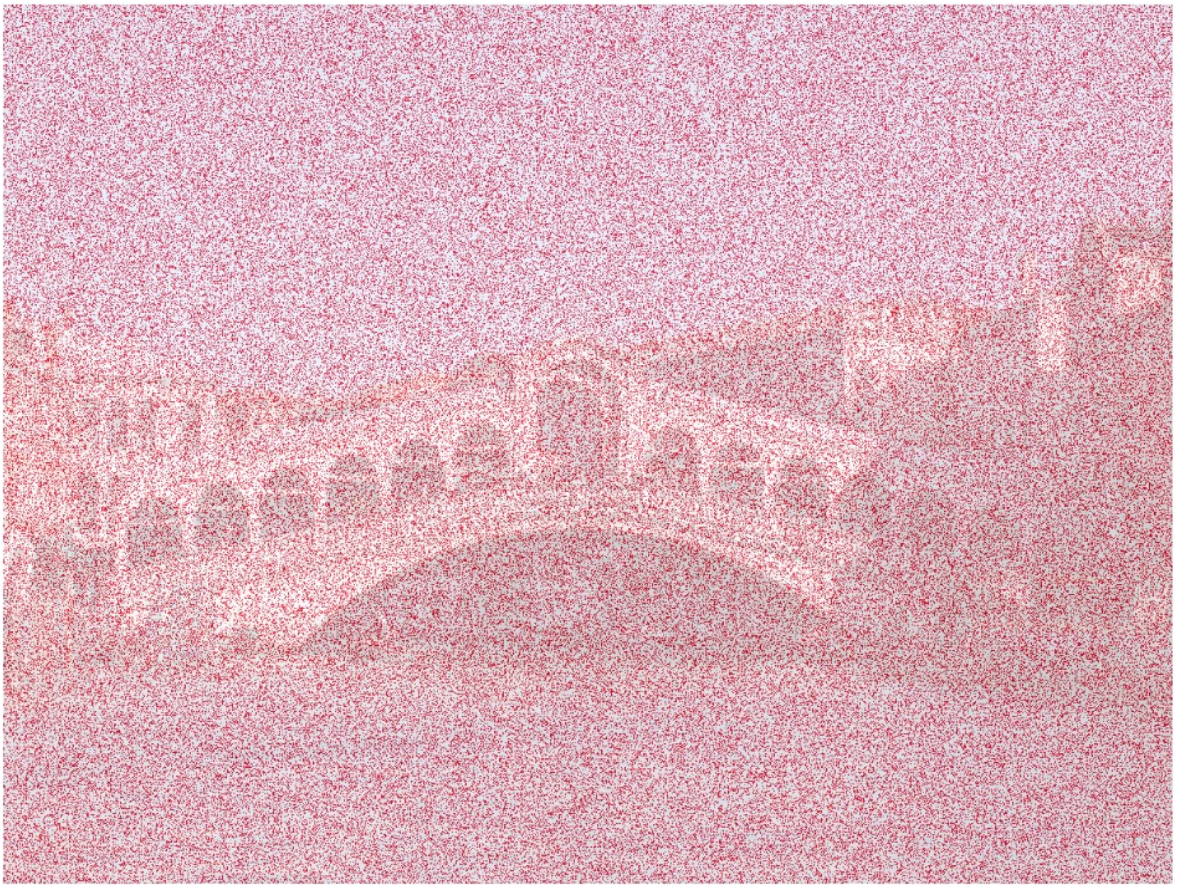
Jedno z kritérií zohledněných při návrhu algoritmu SPCI bylo na rozložení vkládaných (tajných) dat. Správně by měla být informace rozložena rovnoměrně po celém originálním objektu. Na základě srovnání mezi vytvořeným stego obrázkem a originálním obrázkem je na obrázcích č. 41 až 43 znázorněno, v jakých částech obrázku se skrytá data nachází.

Červené tečky představují pixely, ve kterých se tajná data nachází.

Obr. 41. Tajná data ve stego objektu Land.BMP.



Obr. 42. Tajná data ve stego objektu Mountain.BMP.



Obr. 43. Tajná data ve stego objektu Venezia.BMP.

Jak můžeme na obrázcích vidět, tak rozložení tajných dat je rovnoměrné. Nejvíce pixelů je změněných ve stego objektu Mountain.BMP a to z toho důvodu, že je pro skrytá data využito 45,57 % možné kapacity. U objektu Venezia.BMP je kapacita využita na 19,26 %. Zpráva v objektu Land.BMP využívá pouze 0,21 % možné kapacity.

7.4 PSNR a MSE

Termín peak-signal-to-noise-ratio (PSNR) je hodnota, která vyjadřuje poměr mezi maximální možnou hodnotou (výkonem) signálu a výkonem rušivého šumu, který ovlivňuje kvalitu signálu. Protože většina signálů má velmi široký dynamický rozsah (poměr mezi největšími a nejmenšími možnými hodnotami) je tato hodnota obvykle vyjádřena v logaritmickém měřítku decibelů [19].

Hodnocení kvality obrázku (zlepšení nebo zhoršení) může být na úrovni lidských smyslů subjektivní. Každý jednotlivec na základě svého subjektivního názoru může mít rozdílné hodnocení kvality obrázku. Z tohoto důvodu jsou pro hodnocení zlepšení nebo zhoršení

kvality obrázku zavedeny hodnoty jako PSNR nebo MSE (mean squared error, střední kvadratická chyba).

Pomocí stejné sady testovaných obrázků lze systematicky porovnávat různé algoritmy, které ovlivňují kvalitu obrázku a snadno lze určit, který algoritmus produkuje obrázek lepší kvality. Určující hodnotou je právě PSNR.

Typické hodnoty pro PSNR u algoritmů ztrátové komprese obrázku jsou v rozpětí mezi 30 až 50 dB. Čím je hodnota větší, tím je obrázek kvalitnější (čím větší PSNR tím lepší).

Matematický vzorec pro výpočet PSNR u obrázku je následující:

$$PSNR = 20 \log_{10} \frac{MAX_f}{\sqrt{MSE}}$$

Kde MSE je:

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

f…představuje matici (2D pole) dat v originálním obrázku

g…představuje matici (2D pole) dat v obrázku kde je ovlivněna kvalita

m...počet řádků pixelu v daném obrázku a „i“ představuje aktuální index řádku

n…počet sloupců v obrázku a „j“ představuje aktuální index sloupce

Max$_f$…maximální hodnota signálu, která existuje v originálním obrázku

Vypočtené hodnoty, které určují kvalitu obrázku na základě srovnání s originálním obrázkem se nachází v tabulce č. 16.

Tab. 16. Výpočet PSNR a MSE.

Stego objekt	**PSNR**	**MSE**	**Počet změněných pixelů**	**Procento změněných pixelů**
Land.BMP	78,02 dB	0,001034	1316	0,31 %
Mountain.BMP	54,53 dB	0,23	229 461	54 %
Venezia.BMP	58,30 dB	0,097	278 547	27 %

Střední kvadratická chyba představuje průměr čtverců „chyb“ mezi původním obrázkem a stego obrázkem (hodnota, o kterou se liší hodnoty původního obrázku od degradovaného obrázku).

Na základě těchto vypočtených hodnot (PSNR, MSE) a mého subjektivního hodnocení kvality stego objektů můžu říct, že zhoršení kvality není lidským okem pozorovatelné.

V případě převedení původních obrázků Land.BMP, Venezia.BMP a Mountain.BMP do formátu jpg (ztrátová komprese) jsou hodnoty PSNR výrazně nižší. Shrnuto v tabulce č. 17 (standardně se používá zachování 80 % kvality). Z mého pohledu není rozdíl mezi obrázky znatelný.

Tab. 17. Hodnoty originálních obrázků při konvertování do formátu jpg při zachování kvality 80%.

Obrázek	**PSNR**	**MSE**	**Velikost**
Land.JPG	32,04 dB	40,97	89 634 B
Venezia.JPG	36,34 dB	15,21	139 898 B
Mountain.JPG	32,68 dB	35,33	94 553 B

7.5 Využití maximální kapacity

Pro tuto analýzu použijeme obrázek Land.BMP.

Tab. 18. Vlastnosti obrázku Land.BMP.

Popis	**Hodnota**
Název	Land.BMP
Počet pixelů	427 728
Celkem bytů	1 283 184
Kapacita pro tajná data v bytech	160 398

Tajná data budou v podobě docx souboru.

Tab. 19. Vlastnosti tajných dat.

Popis	**Hodnota**
Název	Test_spci.docx
Formát	Dokument word
Počet bytů	159 074

Po vložení souboru Test_spci.docx do obrázku Land.BMP bude využito 99,18 % možné kapacity pro tajná data.

Obr. 44. Originální obrázek Land.BMP.

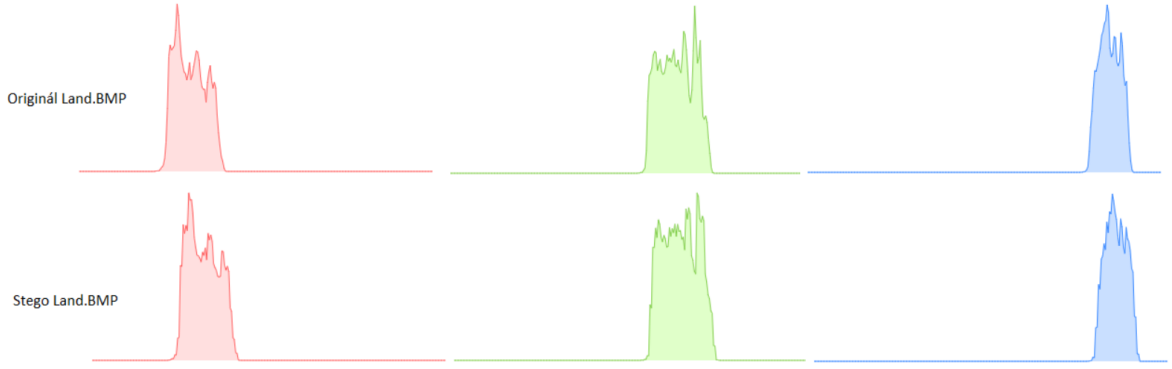
Obr. 45. Stego objekt obsahující soubor Test_spci.docx.

Obr. 46. Rozložení tajných dat ve stego objektu Land.BMP.

Tab. 20. Statistiky stego objektu.

	Hodnota
Počet vložených bitů	1 272 688 bit (159 086 B)
Počet změněných bitů	636 172 bit
Celkový čas	1 406 milisekund
PSNR	51.17 dB
MSE	0,49
Počet změněných pixelů	372 935

I při využití 99 % kapacity pro tajná data nedojde k takovému zkreslení, které by způsobilo postřehnutelnost steganografie lidským zrakem. Na obrázku č. 47 je zobrazen histogram jednotlivých barevných komponent (RGB) pro originální obrázek Land.BMP a stego obrázek Land.BMP. Lze vidět drobné rozdíly.



Obr. 47. Histogram originálního obrázku Land.BMP a stego obrázku Land.BMP

7.6 Komprese

Aplikace SpciApp umožňuje vkládat data do obrázků takového formátu, který nepoužívá kompresi, nebo používá kompresi bezztrátovou (například PNG, nebo TIFF). Povoleny nejsou formáty, které používají ztrátovou kompresi – například jpg apod.

V případě vkládání dat do obrázku, který nepoužívá kompresi vůbec (například BMP), je dosaženo trochu jiného výsledku než v případě formátu s bezztrátovou kompresí. V obou případech nejsou vkládány žádné bity navíc, dochází pouze ke změně stávajících bitů. Rozdíl

je způsoben podstatou samotné komprese, která se používá pro zmenšení velikosti souboru. Z důvodu změny některých bitů v obrázku jsou jiné vstupní hodnoty pro algoritmus komprese. Ta může být vypočtena jinak a výsledná velikost stego objektu může být jiná než velikost originálního obrázku. Například u formátu BMP je výsledná velikost stego objektu stejná jako velikost originálního obrázku.

Příklad:

Tab. 21. Specifikace originálního obrázku.

Popis	**Hodnota**
Název	Dp.PNG
Formát	PNG
Velikost	64 995 B
Počet pixelů	1149x820 = 942 180
Bitová hloubka	24 bitů

Logicky vzhledem k velikosti 64 995 bytů si můžeme myslet, že skrytá data musí mít menší velikost než samotný originální objekt. Jelikož se ale jedná o formát PNG, který používá bezztrátovou kompresi, tak toto neplatí. I přestože je velikost originálního obrázku jen **64 995 bytů**, tak jsme schopni do tohoto obrázku vložit tajná data o velikosti **353 317 bytů.**

Obrázek obsahuje 942 180 pixelů a každý pixel se skládá ze 3 bytů. Do každého bytu jsme schopni vložit jeden bit zprávy.

(942 180 * 3) / 8 = 353 317

Paradoxně jsme tam schopni vložit originální obrázek, který představuje tajná data do stejného originálního obrázku a budeme k tomu potřebovat pouze necelých 20 % jeho celkové kapacity pro skrytá data. Vlastnosti tohoto stego objektu jsou shrnuty v tabulce č. 22.

Výsledek je ale takový, že výsledná velikost stego objektu je větší než velikost originálního obrázku.

Tab. 22. Vlastnosti vytvořeného stego objektu Dp.PNG.

Popis	Hodnota
Název	Dp.PNG
Velikost	386 272 bytů
Celkově vkládaných bitů	520 040
Celkově změněných bitů	259 370
PSNR	58,50 dB
MSE	0,09
Počet změněných pixelů	236 462
Časová náročnost	1 423 milisekund

Na výsledném stego objektu, co se týče kvality není znát přítomnost tajných dat. Nicméně došlo ke znatelnému nárustu velikosti oproti originálnímu obrázku. Výsledná komprese je jiná, než v případě originálního obrázku.

7.7 Shrnutí

Výsledky experimentů ukazují, že navržený algoritmus SPCI poskytuje efektivní způsob pro integraci skrytých informací do originálního objektu bez významného zkreslení vůči originálnímu objektu. Pro neautorizované uživatele je velmi složité identifikovat změny ve výsledném stego objektu. I v případě detekce změn není možné bez znalosti tajného klíče tajná data ze stego objektu rekonstruovat.

V případě vkládání tajných dat do obrázku, který nepoužívá kompresi je výsledná velikost stego objektu rovna originálnímu objektu. Uživatel není schopen na základě prostého porovnání velikosti stego objektu a originálního objektu detekovat tajná data. U obrázků, jež používají kompresi se velikost stego objektu a originálního objektu může lišit.

Algoritmus vkládá tajná data na základě zadaného stego klíče a také zajišťuje rovnoměrné rozprostření vložených informací po celém objektu (kapitola 7.3). Jiný klíč znamená jiné umístění tajných dat v originálním objektu.

Výsledné hodnoty PSNR, které určují míru zkreslení stego objektu vůči originálnímu objektu jsou velmi dobré. I v případě využití 99 % možné kapacity pro tajná data je výsledné PSNR u stego objektu Land.BMP 51.17 dB – v případě konverze originálního obrázku Land.BMP do formátu JPG je hodnota PSNR 32,04 dB. V případě Land.JPG a ani v případě stego objektu Land.BMP nejsem schopen na základě visuální stránky poznat rozdíl vůči originálním obrázku Land.BMP.

Veškeré soubory, které byly použity při analýze výsledků jsou obsahem přílohy této diplomové práce. Stego klíč použitý pro vložení tajných dat je vždy stejný „skola1234“.

ZÁVĚR

Cílem teoretické části bylo popsat problematiku steganografie. Po přečtení práce je čtenář schopen definovat steganografii a metody pro její aplikaci. Jednotlivé metody steganografie se liší v závislosti na objektu, do kterého jsou informace vkládány. Samostatnou kapitolou v teoretické části je steganografie řízená klíčem. Je vysvětlen její princip s příkladem použití. V teoretické části jsou také popsány kategorie steganografie.

Cílem praktické části bylo navrhnout nový algoritmus steganografie řízené klíčem. Navržený algoritmus patří do kategorie substitučních metod steganografie a je tak založen na nahrazení určitých bitů v originálním objektu. Při návrhu algoritmu bylo zohledněno několik kritérií – určení nahrazovaných bitů na základě klíče, řídit rozložení dat na základě klíče a zajistit rovnoměrné rozložení tajných dat. Na základě analýzy výsledných stego objektů bylo prokázáno splnění těchto kritérií. Nový algoritmus pro steganografii řízenou klíčem představuje efektivní způsob pro vložení tajných informací a zajišťuje bezpečnost tajných dat prostřednictvím tajného klíče, bez kterého je velice složité steganografii detekovat, nebo tajnou zprávu rekonstruovat.

Výstupem praktické části této diplomové práce je knihovna, která implementuje nově navržený algoritmus a aplikace, která využívá vytvořenou knihovnu. Implementace v knihovně je univerzální. Není omezena na konkrétní formát tajných dat nebo objektu, do kterého budou data vkládána. Je tak možné ji použít pro ukrytí informací do jakýchkoli binárních dat. Vytvořená aplikace umožňuje vložit tajná data do obrázků, které nepoužívají ztrátovou kompresi a mají bitovou hloubku 24 nebo 32 bitů na pixel.

Tato diplomová práce přináší nový algoritmus steganografie řízené klíčem včetně jeho implementace a aplikace. Pro aplikaci steganografie je možné použít již vytvořenou aplikaci, nebo pouze knihovnu, ke které bude vytvořeno vlastní uživatelské rozhraní (knihovna může být použita také například v mobilní aplikaci).

SEZNAM POUŽITÉ LITERATURY

[1] SIPER, Alan, Roger FARLEY a Craig LOMBARDO. *The Rise of Steganography* [online]. New York, 2005 [cit. 2019-05-12]. Dostupné z: http://csis.pace.edu/~ctappert/srd2005/d1.pdf. Research. Pace University.

[2] JUDGE, James. *Steganography: Past, Present, Future* [online]. US, 2008 [cit. 2019-05-12]. Dostupné z: https://www.sans.org/reading-room/whitepapers/stenganography/paper/552. Research. SANS Technology Institute.

[3] COLE, Eric a Fabien A. P. PETITCOLAS. Hiding in plain sight: steganography and the art of covert communication. 2nd ed. New York: Wiley Pub., c2003. ISBN 04-714-4449-9.

[4] MORKEL, Tayana. *Image steganography applications for secure communication* [online]. Pretoria, 2012 [cit. 2019-05-12]. Dostupné z: https://repository.up.ac.za/bitstream/handle/2263/29906/dissertation.pdf;sequence=1. Dissertation. University of Pretoria.

[5] BASSIL, Youssef. Steganography & the Art of Deception: A Comprehensive Survey. LACSC [online]. 2013, (3), 12 [cit. 2019-04-23]. Dostupné z: http://www.lacsc.org/lacsc/papers/Paper32.pdf

[6] KATZENBEISSER, Stefan a Fabien A. P. PETITCOLAS. Information hiding techniques for steganography and digital watermarking. ISBN 15-805-3035-4.

[7] ANDERSON, Ross J. a Fabien A.P. PETICOLAS. On The Limits of Steganography. IEEE Journal of Selected Areas in Communications. 1998, (16), 474-481. ISSN 0733-8716.

[8] WU, Min a Bede LIU. Multimedia Data Hiding. New York, NY: Springer New York, 2003. ISBN 9780387217543.

[9] WAYNER, Peter a Fabien A. P. PETITCOLAS. Disappearing cryptography: Stefan Katzenbeisser, Fabien A.P. Petitcolas, editors. 2nd ed. Boston: MK/Morgan Kaufmann Publishers, c2002. ISBN 15-586-0769-2.

[10] KADRY, Seifedine a Sara NASR. New Generating Technique for Image Steganography [online]. Lecture Notes on Software Engineering, 2013(2) [cit. 2019-04-23]. Dostupné z: http://www.lnse.org/papers/43-IE2024.pdf

[11] JOHNSON, Neil F. a Stefan C. KATZENBEISSER. A survey of steganographic techniques. PDFS [online], 36 [cit. 2019-04-25]. Dostupné z: https://pdfs.semanticscholar.org/00cc/e201016ba60bd89177a655b0f549b5454574.pdf

[12] WALIA, Dr. Ekta a Pazal JAIN. An Analysis of LSB & DCT based Steganography. PDFS [online]. 2010(10), 36 [cit. 2019-04-25]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.178.7157&rep=rep1&type=pdf

[13] WESTFELD, Andreas. F5—A Steganographic Algorithm [online]. Dresden, 2001 [cit. 2019-04-25]. Dostupné z: https://www.academia.edu/3747199/F5_Steganography. Research. Technische Universit at Dresden.

[14] ABDULLAH, Sadoon Hussein. Steganography Methods and some application [online], Mosul, 2009 [cit. 2019-04-25]. Dostupné z: https://pdfs.semanticscholar.org/086b/ab82891f301826773149012b7c3adf50ad60.pdf. Research. University of Mosul.

[15] KARIM, S. M. Masud, Md. Saifur RAHMAN a Md. Ismail HOSSAIN. A New Approach for LSB Based Image Steganography using Secret Key [online], Khula, 2011 [cit. 2019-04-25]. Dostupné z: https://www.researchgate.net/publication/261421805_A_new_approach_for_LSB_based_image_steganography_using_secret_key. Research. Khulna University.

[16] WAYNER, Peter. Disappearing cryptography: information hiding : steganography & watermarking. 3rd ed. Boston: Morgan Kaufmann Publishers, c2009. ISBN 978-0-12-374479-1.

[17] COLE, Eric. Hiding in plain sight: steganography and the art of covert communication. New York: Wiley Pub., c2003. ISBN 0-471-44449-9.

[18] SHARP, Toby. (2001). An Implementation of Key-Based Digital Signal Steganography. Lecture Notes in Computer Science. 2137. 13-26. 10.1007/3-540-45496-9_2

[19] Peak Signal-to-Noise Ratio as an Image Quality Metric [online], [cit. 2019-04-25]. Dostupné z: http://www.ni.com/cs-cz/innovations/white-papers/11/peak-signal-to-noise-ratio-as-an-image-quality-metric.html.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

DCT	Discrete cosine transform
DFT	Discrete Fourier transform
DWT	Discrete wavelet transform
EOF	End of file
HTML	Hypertext Markup Language
LSB	Least significant bit
MSB	Most significant bit
MSE	Mean square error
PSNR	Peak signal to noise ratio
WAV	Waveform audio file format
WPF	Windows presentation foundation
XAML	Extensible application markup language
XOR	Exclusive OR

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH CD

Struktura obsahu přiloženého CD:

- Adresář **Text diplomové práce** – obsahuje text diplomové práce ve formátu PDF
- Adresář **Zdrojové kódy** – obsahuje zdrojové kódy aplikace SpciApp a knihovny Spci.Steganography.dll
- Adresář **Analýza** – obsahuje všechny soubory použité pro analýzu výsledků algoritmu SPCI
- Adresář **SpciApp** – obsahuje aplikaci SpciApp (spustitelný soubor)