VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

DIGITÁLNÍ TEXTOVÁ STEGANOGRAFIE

DIGITAL TEXT STEGANOGRAPHY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PETR NODŽÁK**
AUTHOR

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

BRNO 2018

Abstrakt

Cílem této bakalářské práce bylo se seznámit s digitální textovou steganografií, s metodami sloužící pro ukrývání textu, naimplementovat vlastní nebo již existující metody a nakonec srovnat dosažené výsledky těchto metod.

Práce představuje čtyři metody (Zero distribution, sémantická, syntaktická a Format based), které ukrývají tajná data do textu, každá jiným způsobem.

Provedeným výzkumem bylo zjištěno, že nejlepší vlastnosti stego textu produkuje Sémantická metoda, hned za ní Format based s Zero distribution a nejhůře dopadla syntaktická metoda. Na základě zjištěných údajů je tedy možné vybrat metodu podle vlastních potřeb.

Abstract

The goal of this bachelor project was to learn about digital text steganography, about methods for hiding the text, to implement my own or already existing methods and finally to compare acomplished results of these methods.

This project introduces four methods (Zero distribution, semathics, syntax and Format based) which hide secret data into cover text, each of them by its own way.

Research has found that semanthics method produces the best properties of stego text followed by Format based and Zero distribution, syntax method ended up worst. Based on the data found we can choose which method we will use.

Klíčová slova

Steganografie, Formátovací metody, Lingvistika, Textové soubory

Keywords

Steganography, Format based methods, Linguistics, Text files

Citace

NODŽÁK, Petr. *Digitální textová steganografie*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Josef Strnadel, Ph.D.

Digitální textová steganografie

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně
pod vedením...
Další informace mi poskytli...
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Nodžák
7. května 2018

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce, který mi poskytl jak odbornou, tak i morální podporu při tvorbě, také svým rodičům, kteří se mnou měli trpělivost a pochopení.

Obsah

Kapitola 1

Úvod

Tato bakalářská práce se zabývá digitální textovou steganografií, je to metoda ukrývání dat do textového souboru. Steganografie vznikla již ve starověku, od té doby se postupně vyvíjela až do dnešní doby internetu, kde nachází své uplatnění ve formě tajných komunikací. Tento vývoj je v práci postupně zachycen, až se dostane k samotné digitální textové steganografii, kde jsou popsány moderní principy pro ukrývání dat v textu. Hlavním cílem této práce bylo naimplementovat, demonstrovat funkčnost a porovnat vlastnosti výsledků všech metod.

Práce se zaměřuje na digitální textovou steganografii, jako přenosové médium tedy slouží textový soubor, který je uložen na záznamovém zařízení. Pro zobrazení a editování takového souboru slouží textové editory nebo prohlížeče. Cílem textové steganografie je ukrýt data do textového souboru a v ideálním případě zařídit, aby změny po ukrytí nebyly lidským okem viditelné.

Tato práce pojednává o steganografii, jejím využití, kde steganografie vznikla a jak se s postupem času rozvíjela.

Tato práce se dále věnuje návrhu aplikace, která obsahuje čtyři metody pro ukrývání dat do textu. V neposlední řadě vyobrazuje vlastnosti výsledků jednotlivých metod, které byly zjištěny pomocí experimentování a jejich následné srovnání.

Práce je rozdělena do pěti kapitol, každá kapitola se věnuje jiné problematice. Druhá kapitola je věnována obecné steganografii, její historii, z čeho se skládá a jak funguje stegosystém nebo o vývoji steganografie. Třetí kapitola odkrývá hlavní principy ukládání dat do digitálních textů, zvolený typ ukrývané informace, detekci a útoky na steganografický text. Ve čtvrté kapitole je prezentován návrh aplikace a implementace metod, které odpovídají principům ze třetí kapitoly. Vlastnosti výstupních textů z naimplementovaných metod, jejich porovnání mezi sebou, návrhy na vylepšení a využití, jsou shrnuty v poslední páté kapitole.

Kapitola 2

Steganografie

V této kapitole se věnuji obecnému popisu steganografie a co to vlastně je, rozdílům s kryptografií a jejich možné kombinování. Stručně shrnu jaká média a typy souborů se v moderní steganografii využívají. Dále je přiblížena samotná historie steganografie, kde pravděpodobně vznikla, kdo ji poprvé nazval svým dnešním názvem a jak se uchytila v obou světových válkách. Podkapitola 2.1.2 znázorňuje známý *Prisoner's problem*, který je názornou ukázkou, jak samotná steganografie funguje. V podkapitole 2.1.3 je vysvětleno, co to takový stego-systém je, z čeho se skládá a jak funguje. Následuje sekce 2.2, kde se nachází výpis a přiblížení různých steganografických metod. Upozorňuji, že se nejedná o všechny metody, co k této problematice existují, jedná se jen o výběr těch metod, které mi přišly nejvíce zajímavé.

2.1 Steganografie

Steganografie je umění tajné komunikace [8]. Kryptografie a Steganografie jsou nejznámější metody, jak bezpečně přenášet data přes internet. Hlavním smyslem kryptografie [8] je převést celou zprávu do podoby, kterou jde přečíst jen pomocí speciálních znalostí. Z toho plyne, že každý si je vědom toho, že zde probíhá tajná komunikace. Naproti tomu steganografie ukrývá data do nevině vyhlížejícího objektu zvaného „cover". Tím se komunikace projevuje jako neškodná a zaručuje, že skrytou zprávu bude číst jen požadovaná osoba, což je smysl celé steganofrafie. Je možné, že i přes to je komunikace odhalena, proto se steganografie často kombinuje s kryptografií, kdy se tajná zpráva ještě vhodně zakóduje.

Moderní steganografické metody prioritně využívají elektronická média, než fyzické objekty a texty. Data která mají být skryta se ve steganografii nazývají skrytá data. K ukrytí skrytých dat se používá například obrázek, textový, zvukový nebo video soubor. Toto nosné médium se nazývá **cover**. K detekování je potřeba znalost skrytých dat nebo postupu, kterým data bylo ukryta.

Nejčastějším médiem pro steganografii je obrázek, jelikož je v něm možnost změnit velké množství bitů, aniž by to lidské oko dokázalo rozeznat, protože není uzpůsobené k rozeznání tak malých detailů, které se při ukrývání dat mění. Obecně nejvhodnější médium pro steganografii bude to, které má největší redundanci.

2.1.1 Historie

První použití steganografie zdokumentoval řecký historik Herodotus [8], který zachytil konflikt mezi Persií a Řeckem. Vyhoštěný řek, žijící v Persii, se rozhodl informavat svoji rodnou

zem před invazí Xerxa, pomocí dřevěné tabulky s voskem, kterou následně zalil další vrstvou vosku.

Ve své knize „Dějiny“ Herodotus zdokumentoval [8] další použití steganografie. Popisuje zde řeka jménem Histaiaeus, který chtěl povzbudit Aristagorase z Milétu k revoluci proti perskému králi, toho docílil tak, že oholil hlavu jednomu ze svých poslů, posléze na ni napsal zprávu a počkal až mu znovu narostou vlasy. Tato metoda byla mnohem bezpečnější, jelikož posel nenesl žádný fyzický předmět, jako tomu bylo u prvně popisované metody.

První zaznamenané použití slova **steganografie** šahá až do roku 1499, kde jej Johannes Trithemus (1462-1516) použil ve své knize „Steganographia, a treatise on cryptography and steganography disguised as a book on magic“. Název steganografie pochází ze spojení dvou řeckých slov [8], znamenající „skryté“ a „psaní“.

Během První Světové války se steganografie úspěšně uchytila [8]. Jedním z příkladů byla Turningova mřížka, která vylepšovala Cardonovu mřížku 2.2.10 tím, že po každém přečtení znaku se čtecí mřížka otočila o 90 stupňů.

Ve Druhé Světové válce, byla vynalezena technika [8] mikrotečky 2.2.1, která zmenšila fotografii textu, do velikosti menší než jeden milimetr. Mikrotečka byla přidána k nevinnému dopisu.

2.1.2 Prisoner's problem

Jedním z výborných příkladů k pochopení, co to vlastně steganografie je, Simmons popsal problém vězňů [8].

Alice a Bob jsou dvě fiktivní osoby, které byly zadrženy a umístěny do různých cel. Jejich úkolem je najít způsob, jak z vězení uniknout [8]. Jediná možnost jak spolu mohou komunikovat je přes dozorkyni Wendy. Wendy, jakožto schopná dozorkyně, je nenechá komunikovat zakódovaně a pokud nabude nějakého podezření, okamžitě je umístí na samotku. V tom případě musí Bob a Alice komunikovat takovým způsobem, aby nevzbudili žádné podezření, toho docílí za použití steganografie.

Příklad pokračuje tak, že nejlepší řešení je ukrýt zprávu do nevině vyhlížejícího textu [8] nebo obrázku. Bob by mohl nakreslit obrázek s modrou krávou a zelenou pastvinou, požádat Wendy aby obrázek předala Alici. Wendy, než obrázek předá, si ho samozřejmě pečlivě prohlédne a pomyslí si, že se jedné jen kus abstraktního umění. Netuší ale, že barvy na obrázku ukrývají skrytou zprávu.

Toto řešení zároveň odkrývá několik slabin steganografie. Wendy by mohla pozměnit obrázek, ať už záměrně nebo omylem, tím by došlo i ke změně nebo zničení skryté zprávy [8]. Pokud by Wendy záměrně změnila obrázek, jednalo by se o aktivní útok. Další forma útoku je škodlivý útok, pokud by Wendy podstrčila Alici vlastní zprávu a předstírala by, že to je od Boba.

Model problému vězňů lze uplatnit v mnoha případech, kde je pro komunikaci potřeba použít steganografii. Alice a Bob jsou dvě strany, které potřebují tajně komunikovat a Wendy je naslouchající. Ikdyž je tento model efektivním způsobem skryté komunikace, vždy je třeba brát zřetel na potencionální pasivní, aktivní nebo jiné škodlivé útoky.

2.1.3 Stego-systém

V této podkapitole si vysvětlíme, co to je textový stego-systém, pro ostatní stego-systémy platí vesměs to stejné.

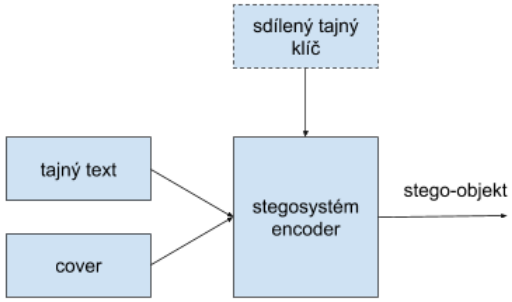
Stego-systém (steganografický systém) [5] v textové steganografii je systém, který je schopen ukrýt tajnou zprávu do textu (obecně coveru), ze kterého třetí strana nezíská

podezření, že nějaká skrytá komunikace probíhá. Text který je výstupem tohoto procesu se nazývá stegogram (stego text), dbá se na to, aby bylo zajištěno, že výsledný stegogram bude vypadat co nejvíc nejneviněji, jak jen to je možné, aby tajná zpráva vypadala nerizikově a tak by zajistila, že tajná komunikace bude opravdu tajná.

Stego-systém nejen, že zajišťuje zakódování zprávy, ale taky umožňuje přečíst skrytou zprávu [5] za pomocí dekodéru, když je doručena cílovému adresátovi. Stego-systém se tedy dělí na dva podsystémy, enkodér a dekodér.

Enkodér

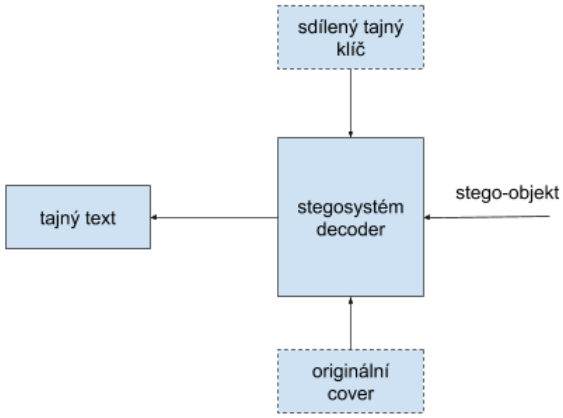
Srdcem steganografického systému je enkodér stego-systému [5]. Umožňuje vložit tajnou zprávu do texotvého cover média. V textové steganografii enkodér přečte tajnou zprávu a převede ji do binární podoby. Co se bude dít dál už záleží na zvolené steganografické metodě, která vygeneruje svůj cover text, nebo upraví již existující. Obrázek 2.1 znázorňuje grafickou reprezentaci objektů a procesů, které jsou typicky spjaté s enkodérem.



Obrázek 2.1: Základní ilustrace obecného enkodéru stego-systému s tajným sdíleným klíčem

Dekodér

Dekodér stego-systému umožňuje příjemci stegogramu obdržet alespoň odhad [5] tajné zprávy. Dobrý stego-systém zajišťuje, že tento odhad bude co nejvíc podobný originální tajné zprávě, jak jen to je možné. Obrázek 2.2 znázorňuje grafickou reprezentaci objektů a procesů, které jsou typicky spjaté s dekodérem stego-systému. Jak můžeme vidět, k dékódování tajné zprávy je potřeba sdílený tajný klíč a stegogram. Kdokoliv má k dispozici tyto dvě věci, tak se může dostat k tajné zprávě. Někdy je potřeba mít i originální cover (např. vhodné při metodách, které upravují formát textu).



Obrázek 2.2: Základní ilustrace obecného dekodéru stego-systému

2.2 Vývoj steganografie

2.2.1 Mikrotečky

Mikrotečka [8] je fotka o velikosti stránky, která byla zmenšena v průměru na 1 milimetr. Mikrotečky se staly populární během 2. světové války. Vytvoření mikrotečky spočívá v tom, že se vyfocená zpráva zmenší na velikost poštovní známky, poté se za pomocí reverzního mikroskopu zmenšila na velikost 1 milimetru. Nakonec se vyvolá negativ a obrázek se vyrazí do filmu. Většinou toho bylo docíleno pomocí injekční stříkačky, jejíž hrot byl upilovaný.

2.2.2 One-time pads

Tato metoda [8] používá náhodný klíč k zakódování zprávy pouze jednou. Nežáleží na tom, jak je moderní kryptoanalýza úspěšná nebo kolik má zdrojů, prolomení této metody by mělo být téměř nemožné. Klíč by se měl pokaždé měnit.

2.2.3 Semagramy

Semagramy [8] jsou znaky a symboly, které jsou schopny ukrýt zprávu. Můžou to být různé kresby, rozmístění předmětů na určitém místě a pořádí, které se pro nezainteresovanou osobu můžou jevit nevině, ale pro adresáta odkrývat skrytou zprávu.

2.2.4 Nullové šifry

Nullové šifry [8] ukrývají tajnou zprávu ve větší zprávě, která zdánlivě vyhlíží nevinně. Né vždy se podaří vygenerovat text, který by dával úplně smysl. K dekódování skryté zprávy je potřeba číst každé n-té písmeno ve zprávě.

> ***N**ews **E**ight **W**eather: **T**onight **i**ncreasing **s**now. **U**nexpected **p**recipitation **s**mothers **e**astern **t**owns. **B**e **e**xtremely **c**autious **a**nd **u**se **s**nowtires **e**specially **h**eading **e**ast. **T**he **h**ighways **a**re **k**nowingly **s**lippery. **H**ighway **e**vacuation **i**s **s**uspected. **P**olice **r**eport **e**mergency **s**ituations **i**n **d**owntown **e**nding **n**ear **T**uesday.*

Ve výše uvedeném příkladu [8], je zakódovaná zpráva v prvím písmeně každého slova. Výsledná věta zní „Newt is upset because he thinks he is President“.

2.2.5 Anamorfóza

Metoda [8] využívá jen obrázky, které vypadají deformovaně, dokud nejsou zobrazeny z určitého úhlu nebo za pomoci speciálních nástrojů.

Obrázek 2.3: Nalevo originální obrázek, napravo pohled ze specifického úhlu

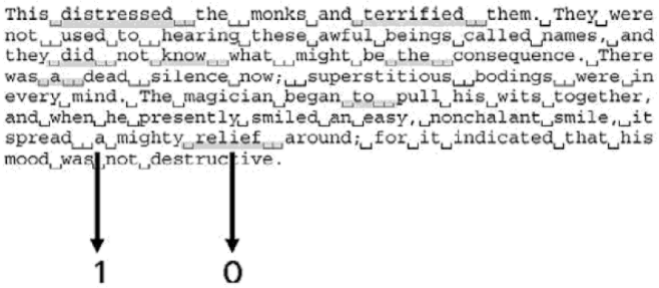
2.2.6 Akrotika

Akrotika [8] jsou většinou obsaženy v básních nebo v sérii řádků, kde po přečtení prvního znaku na každém řádku, dají dohromady jméno, motto nebo zprávu.

2.2.7 Psaní mezer a odsazení

Metoda [8] využívá mezer mezi slovy k zakódování tajné zprávy, čímž deformuje původní text. Takto přidané mezery vychýlí některé znaky z jejich normální pozice. Znaky které jsou vychýleny indikují tajnou zprávu.

Digitální způsob využívá redundantních mezer mezi řádky a slovy, na jejich základě indikuje 1 nebo 0 v binární soustavě. Mezery jsou přidávány a detekovány pomocí steganografické metody.



Obrázek 2.4: Mezery přidané do textu, pomocí steganografického nástroje

2.2.8 Neviditelný inkoust

Metoda [8] zakládá na tom, že skrytá zpráva je neviditelná, teda pokud se nepoužije speciální metoda k jejímu zobrazení. K tomu slouží například zahřátí, světlo nebo speciální chemická substance. Bezbarvá tekutina je aplikovaná na papír a vysušena, čímž se stane neviditelnou. V minulosti bylo využíváno mnoho takových tekutin, například mléko, ocet, citrónový džus, dokonce i moč. Neviditelné inkousty se často výráběly za pomocí složitějších procedur, aby je nešlo snadno odhalit. Příklady některých postupů, jak vytvořit neviditelný inkoust:

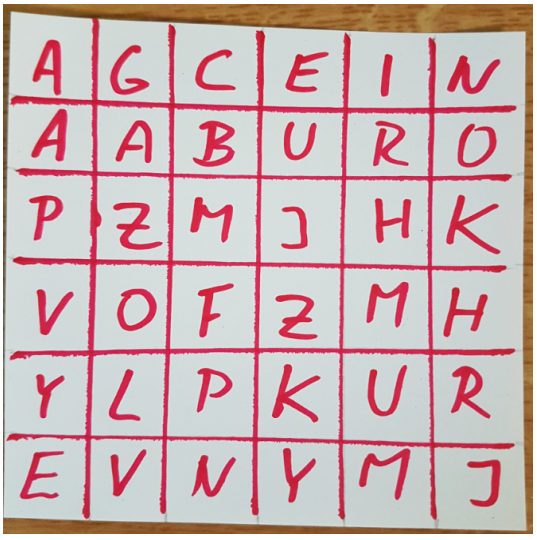
> *Vejce byla použita ke skrytí tajné zprávy. Zpráva byla zapsána inkoustem na skořápku čistého vejce, který projde přes skořápku. Když je vejce uvařeno na tvrdo, skořápka je opatrně oloupána a odhalí se skrytá zpráva.*
>
> *Oxid kobaltu rozpuštěný v chlorovodíku nebo kyselině dusičné vytvoří tekutinu, která je neviditelná dokud není vystavena ohni, ve kterém se rozsvítí do modra. Následným foukáním na list papíru by měla modrá zmizet.* [8]

2.2.9 Novinový kód

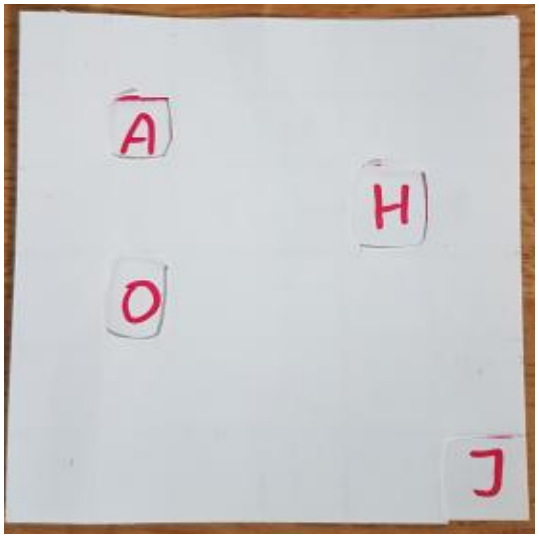
Novinový kód [8] spočívá v tom, že nad určitými písmeny v novinách byla vyražená díra, po postupném přečtení takových písmen, se dala získat tajná zpráva. Později byly díry nahrazeny neviditelným inkoustem. Metoda měla jednu velkou nevýhodu a tou byla rychlost v doručení novin.

2.2.10 Cardanova mřížka

Metoda [8] je pojmenovaná po svém vynálezci Girolamo Cardano. Každý příjemce má u sebe šablonu, kterou se překryje textová zpráva. V šabloně jsou díry, které po překrytí zobrazí tajnou zprávu.



Obrázek 2.5: Zpráva

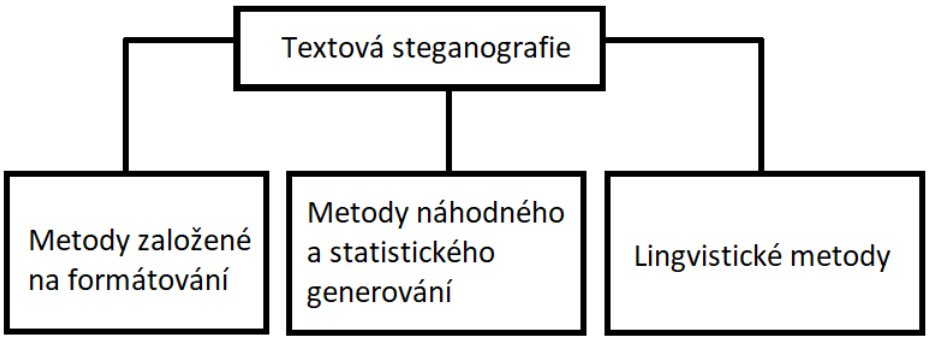


Obrázek 2.6: Tajná zpráva po přiložení správné šablony

Kapitola 3

Digitální textová steganografie

V této kapitole nastíním vybraný formát ukrývané informace 3.1, tři základní principy digitální textové steganografie. Metody založené na formátování 3.3, náhodném a statistickém generování 3.4 a lingvistické metody 3.5, ty jsou popsány a vysvětleny více do hloubky. Zaměřím se hlavně na lingvistické metody, jelikož se jeví nejzajímavěji. Rozdělení textové steganografie na obrázku 3.1.



Obrázek 3.1: Rozdělení textové steganografie

3.1 Ukrývaná informace – ASCII

Textová informace je nejlepší kandidát k ukrytí do textového souboru. Každý text potřebuje znakovou sadu, kterou je kódovaný, nejlepší možný kompromis je ASCII.

ASCII (American Standard Code for Information Interchange) [1] je znaková sada, která je vhodná pro ukrývání informace do textu. ASCII původně kódovala znaky do sedmi bitů (128 možných znaků), to v hodně případech nestačilo, tak se rozšířila na osmi bitovou, ta už zvládla zakódovat 256 znaků. Kóduje jen ty nejdůležitější znaky (písmena, číslice a speciální

znaky). Oproti jiným znakovým sadám (Unicode, atd..), dokáže zakódovat anglický text s nejnižším počtem bitů. Sedmi bitová ASCII tabulka je vyobrazena na obrázku 3.2.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Obrázek 3.2: Sedmi bitová ASCII tabulka [1]

3.2 Textová steganografie

Textová steganografie využívá text jako médium, do kterého se ukrývají data. Do textové steganografie se řadí cokoliv, od změny formátu existujícího textu, změn slov v textu, generování náhodných posloupností znaků nebo využití bezkontextových gramatik ke generování čitelného textu. Tyto metody mají jednu společnou vlastnost a tou je, že skrytá data jsou vložena do textového média. To co tyto metody odlišuje je, pokud cover text už existuje nebo jestli byl vygenerován některým ze steganografických nástrojů, dále pokud výsledný stego text je výsledkem náhodných změn, statistického generování, nebo lingvistického generování a modifikace. [13]

3.3 Metody založené na formátování

Jedny z metod, pro ukrývání zpráv do textu jsou metody založené na formátování. Mění formát již existujícího textu k tomu, aby do něj ukryli požadovanou tajnou zprávu. Vkládání mezer, záměrné pravopisné chyby rozmístěné po textu nebo změny fontu jsou jedny z mnoha metod.

Všechny tyto metody, které zakládají na formátování, potřebují k dekódování tajné zprávy původní nezměněný soubor nebo znalost k jakému formátování došlo.

3.3.1 Open Space Methods

Je zde mnoho způsobů, jak využít open space metody [11] k zakódování informací. Tyto metody fungují, protože obyčejný čtenář by abnormality v textu, způsobené přidáním me-

zer na konec řádku nebo mezi slova, neměl mít šanci na první pohled rozeznat. Jeden z podporovaných formátů, s kterým tyto metody fungují je ASCII formát.

- **Inter-sentence space** metoda [11] kóduje logickou „0“ přidáním jedné mezery za tečku v Anglické próze. Přidáním dvou mezer zakóduje logickou „1“. Tato metoda funguje, ale vyžaduje velké množství cover textu a také existuje mnoho textových nástrojů, které automaticky opravují a odstraňují nadbytečné bílé znaky v textu.

- **Line-shifting** [11] kódování zahrnuje posouvání každého řádku vertikálně dolů (binární „0“) nebo nahoru (binární „1“) asi o tři milimetry. Podle toho jestli je řádek posunut výš nebo níž od originálu, se může vyčíst tajná zpráva v binární podobě.

- **End-of-line spaces** metoda [11] vkládá bílé znaky na konec každého řádku. Tato metoda funguje lépe než Inter-sentence scape metoda, jelikož je schopna ukrýt více informací ve stejně velkém cover textu. Čím víc mezer se přidá, tím víc se může ukrýt informací. Například dvě mezery zakódují jeden bit, čtyři mezery dva bity, osm mezer tři bity a tak dále.

- Zarovnání doprava textu může být také použito k zakódování tajné informace do textu. Kódování je docíleno počítáním a kontrolováním mezer mezi slovy v textových souborech. Jedna mezera mezi slovy znamená logickou „0“, dvě mezery znamenají logickou „1“. Tento postup se ukázal jako těžký k dekódování tajné informace. Proto se přišlo s podobnou technikou založenou na **Manchestrově kódování** [7]. Sekvence bitů „01“ je interpretována jako logická „1“ a „10“ je interpretována jako logická „0“. Zbytek („00“ a „11“) je brán jako nullový bit. [11]

- Specifické metody [11], u kterých zakódování tajné zprávy zahrnuje změny textových atributů, jako je typ, velikost nebo tloušťka fontu. Patří zde například **Baconova šifra**, je to vlastně pěti bitový binární kód, který kóduje jedno písmeno pomocí pěti znaků, tyto znaky značí dva podobné fonty (A a B). V tabulce 3.1 je znázorněno kódování Baconovi šifry. Z toho plyne, že cover text musí být pětkrát větší než urývaná zpráva.

  Ukázka zakódování zprávy „ahoj“:

  Kódování: AAAAA AABBB ABBAB ABAAA

  Použité fonty: pro úkázku byl použitý font celého dokumentu jako font A a tučný font jako B, v reálné situalci by si fonty A a B měly být co nejvíce podobné, nejlíp od sebe nerozeznatelné

  Steganografický text: JE OBECN**E** **ZN**AM**O**U **VEC**I ZE

A	AAAAA	**I+J**	ABAAA	**R**	BAAAA
B	AAAAB	**K**	ABAAB	**S**	BAAAB
C	AAABA	**L**	ABABA	**T**	BAABA
D	AAABB	**M**	ABABB	**U+V**	BAABB
E	AABAA	**N**	ABBAA	**W**	BABAA
F	AABAB	**O**	ABBAB	**X**	BABAB
G	AABBA	**P**	ABBBA	**Y**	BABBA
H	AABBB	**Q**	ABBBB	**Z**	BABBB

Tabulka 3.1: Kódovací tabulka Baconovi šifry

3.4 Náhodné a statistické generování

Tyto metody [6] zakládají na tom, že si generují vlastní cover text. Některé vlastnosti takto vygenerovanáho textu můžou vzbudit podezření, že text není plně legitimní. Naproti tomu se vyhýbají problému, že třetí strana by mohla znát cover text. Vygenerovaný text se snaží napodobit některé vlastnosti reálného textu, obvykle pomocí přiblížení k náhodné statistické distribuci písmen a slov, nalezené v reálném textu.

3.4.1 Sekvence znaků

Jedna metoda [6] ukrývá informaci v sekvenci znaků, která se jeví jako kompletně náhodná posloupnost znaků. Samozřejmě, že náhodně se musí jevit pro všechny, kterým zpráva není určena, adresát s klíčem dokáže zprávu přečíst. Samotná náhodná sekvence znaků už by mohla vzbuzovat lehké podezření.

Druhá metoda [6] při generování znaků bere statistické délky slov a frekvenčnost znaků, které jsou použity pro vytváření slov. Tyto slova se jeví, že mají stejné statistické vlastnosti jako reálná slova v daném jazyku, ačkoliv nemají žádnou lexikální hodnotu. Takto vygenerovaná slova, by měla projít počítačovými systémy na odhalování ukrytého textu, které se zaměřují na statistickou analýzu, na druhou stranu tyto texty budou podezřelé pro moderní počítačové systémy a samozřejmě lidské oko.

3.4.2 Sekvence slov

Pro zabránění detekce nelexikálních sekvencí [6], jsou použity slova z reálných slovníků, které mohou být využity k zakódování jednoho nebo více bitů. Takto vygenerovaná slova nemají žádnou sémantickou strukturu a mohou být detekovány jak lidmi, tak i počítači. Ikdyž program na odhalování nedokáže provést komplexní syntaktickou analýzu, může upozornit útočníka, že v textu by mohla probíhat skrytá komunikace, stačí mu provést klasifikaci slov a všímat si nepravděpodobných vzorů, jako je například mnoho sloves za sebou, žádné předložky a tak dále. Dokonce textové procesory, které jsou schopny ověřit gramatické vlastnosti, dokáží odhalit gramaticky špatné texty.

3.4.3 Mimika

Mimika napodobuje některý reálný text, z kterého bere znaky a na základě jejich frekvenčnosti v reálném textu z nich skládá slova a následně celé věty.

K vygenerování n-tého řádu steganografického textu pro daný zdrojový text se sestrojí seznam všech možných kombinací slov o délce n, které se vyskytují ve zdrojovém textu. Zároveň se musí sledovat počet výskytů jednotlivých slov. Náhodně se vybere jedno slovo, které bude jako první pro steganografický text. [13]

K vybrání dalšího písmene se veme posledních $n-1$ písmen ze steganografického textu, prohledá se statistická tabulka a najdou se všechny kombinace slov, které začínají $n-1$ písmeny. Poslední písmena těchto kombinací dávají hohromady set možných kombinací pro další písmeno, které bude přidáno do steganografického textu. K vybrání dalšího písmana je použit obrácený Huffmanův algoritmus. Takto se generuje každý další znak, dokud není vygenerováno dostatečné množství stego textu. Tento postup platí jen pro $n > 1$. Čím větší n je, tím se čitelnost a kvalita výsledného steganografického textu stává důvěryhodnější. Statistická frekvenčnost výskytu znaků bude díky tomuto postupu korektní pro daný jazyk. Ukázka prvního a čtvrtého řádu [13]:

First Order islhne[hry saeeooisnre uo ‘ w nala al coehhs pebl
e to agboean ce ed cshcenapch nt
sibPah ea m n [tmsteoia lahid egnndl y et r yf arleo awe
l eo rttnntnnhtohwiseoa a dri 6oc7teit2t lenefe clktoi
l mlte r ces. woeiL , misetemd2np eap haled&oolrcc yttr
tr,oh en mi elarlbeo tyNunt . syf es2 nrrpmdo,0 reet dadwn’dysg
te.ewn1ca-ht eitxrni ntoos xt eCc oh sao vhsoh0mhgr

Obrázek 3.3: První řád

Fourth Order captionary. Image and to compression lest
constance tree. Family for into be mode of bytes in
algorith a file of that cosition algorithm that word
even that a size summarge factal size are:

ite position scien Raps.

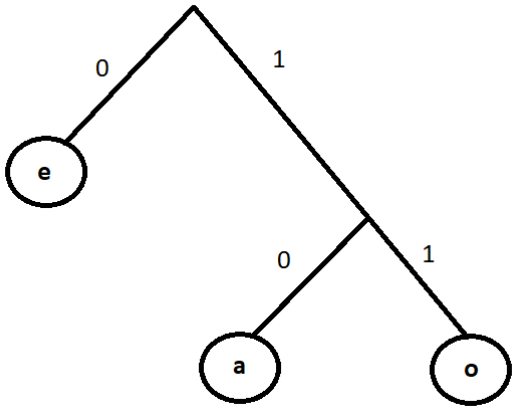
The is are up much length ence, the if the a
refsec-ent sec-ent of fits to the crans usuall
numberse compression

A good ways that in algoright. The brase two wants to
hidea of English Cash the are compres then matimes formatimes
from the data finding pairst. This only be ression o

Obrázek 3.4: Čtvrtý řád

K zakódování tajného textu slouží právě obrácený Huffmanův algoritmus [13], kde je vytvořena stromová struktura z písmen, které mohou následovat po $n-1$ znacích. Takto vytvořený strom značí statistickou frekventovanost písmen, kde znaky které se vyskytují častěji zakódují méně dat. Mějme slovo „The l“, po kterém ve zdrojovém textu [13] následuje písmeno „e“ šestnáctkrát z dvaceti případů a písmena „o“ a „a“, obě dvakrát z dvaceti případů. Pro takto danou situaci vznikne strom na obrázku 3.5.

Následující písmeno je vybráno podle toho, který bit potřebujeme zakódovat, pokud je potřeba zakódovat nulu, vybere se písmeno „e“. Pokud je potřeba zakódovat jedničku, vybere se „a“ nebo „o“, záleží na tom, co bude následovat po jedničce. Pro „a“ to bude „10“ a pro „o“ „11“. Z čeho plyne, že znaky, které se vyskytují méně často ve zdrojovém textu, dokáží ukrýt více tajných informací.



Obrázek 3.5: Huffmanův strom

3.5 Lingvistická steganografie

Novodobá výpočetní technika je schopna analyzovat lingvistické struktury, které jsou více a více komplexnější. Musí zlepšovat lingvistické vlastnosti a zároveň lexikální a ortografické rozmístění. Lingvistická steganografie pro vygenerovaný a modifikovaný text bere v úvahu lingvistické vlastnosti textu a používá lingvistické struktury jako objekt ke skytí tajných informací. [13]

Mimika je schopná oklamat jen statistickou analýzu textu, ale ani zdaleka není schopna vygenerovat steganografický text, který by prošel některou analýzou zaměřenou na gramatiku. Tato sekce popisuje jak vytvořit steganografický text, který bude gramaticky správný a pro takové analýzy téměř nezjistitelný. [13]

3.5.1 Bezkontextové gramatiky

Notace vyvinuta Noamem Chomským [13] k vysvětlení toho, jak jazyky fungují. Jejich struktura je více méně matematický zápis věty. Na těchto gramatikách staví většina programovacích jazyků svoji syntaxi. Bezkontextové gramatiky se skládají ze tří částí:

- **Terminály** - jsou symboly, které se vyskytují v produkčních pravidlech a nemohou být změněny použitím pravidel dané gramatiky. Rekurzivním opakováním pravidel na neterminály obvykle vznikne finální věta, která obsahuje jen terminály
- **Neterminály** - jsou ty symboly, které mohou a nemusí být nahrazeny neterminály nebo terminály, v následujících příkladech budou označovány **tučně**
- **Produkce** - každá gramatika je definovaná produkčními pravidly, které specifikují jaké symboly(neterminály i terminály) nahradí jiný symbol(neterminály)

Bezkontextové gramatiky [13] jsou definovány tak, že každé produkční pravidlo může mít na levé straně jen jeden neterminál. To znamená, že né všechny jazyky mohou být vygenerovány těmito gramatikami, ty které mohou se nazývají Bezkontextové jazyky. Příklad jednoduché bezkontextové gramatiky v tabulce 3.2

Samotný steganografický text vzniká ze syntaxe takto definovaného jazyka. Například při zakódování nuly se vybere levá strana z výběru. Naopak pro jedničku se vybere pravá

Start	->	**podmět přísudek**
podmět	->	Petr || Radek
přísudek	->	si hraje **kde** || běhá **kde**
kde	->	na hřišti **tmp** || okolo domu **tmp**
tmp	->	**kdy** s rodiči || **kdy** s kamarády
kdy	->	večer || ráno

Tabulka 3.2: Produkční pravidla

strana. Při zvětšení výběru možností na jedno produkční pravidlo, se zvedá počet bitů zakódovaných na jeden terminál. V tabulce 3.2 mají všechny produkční pravidla svoji množinu možností M a v ní pouze dva členy, z toho plyne, že je možné zakódovat jen jeden bit. Při zvýšení počtu členů v množině na čtyři to budou dva bity, osm je rovno třem bitům. Vycházím z matematického vzorečku $members = 2^{b_i}$, kde b_i je počet bitů a $members$ je počet členů v množině možností M_i.

krok	průběžný text	bit k ukrytí	výběr produkce
1	**Start**	-	**Start** -> **podmět přísudek**
2	**podmět přísudek**	0	**podmět** -> Petr
3	Petr **přísudek**	1	**přísudek** -> běhá **kde**
4	Petr běhá **kde**	1	**kde** -> okolo domu **tmp**
5	Petr běhá okolo domu **tmp**	0	**tmp** -> **kdy** s rodiči
6	Petr běhá okolo domu **kdy** s rodiči	0	**kdy** -> večer
7	Petr běhá okolo domu večer s rodiči	-	-

Tabulka 3.3: Použití pravidel k vygenerování steganografického textu

Tabulka 3.3 vyobrazuje příklad zakódování požadované bitové sekvence. Ve větě „Petr běhá okolo domu večer s rodiči“ je skrytá posloupnost bitů „01100“

Komplexnější gramatiky zvládnou vygenerovat čitelnější výsledný text. Při návrhu takové gramatiky se musí dbát na to, aby všechny slova a fráze po sestavení vět k sobě seděly. To mnohdy vyžaduje pořádnou dávku kreativity a štěstí.

Parsování

Získání dat z vět mění bezkontextové gramatiky v reálný nástroj k tajnému šíření informací [13]. Opačný proces generování vět pomocí bezkontextových gramatik se nazývá parsování. To nám umožňuje z již vygenerovaných vět získat sekvenci bitů, které reprezentují tajnou zprávu. Programovací jazyky, jedním z nich je C, jsou postaveny na bezkontextových gramatikách. Programovací jazyk parsuje tento kód k interpretaci jeho intrukcí.

Většina programovacích jazyků je navrhnuta k jednoduchému parsování, tak aby proces parsování mohl probíhat co nejrychleji. Téměř vždy se dají parsovat věty z jakékoliv bezkontextové gramatiky a tím získat sekvence produkcí. Pokud je gramatika navržena správně, pak je kdokoliv schopný vyparsovat skrytá data. [13]

Jsou dvě pravidla [13], která musí být dodržena, aby se data daly korektně vyparsovat. Za prvé se musíme ujistit, že gramatika je jednoznačná, za druhé gramatika musí být v Greibachově normální formě (GNF). Pokud dvě stejné věty mohou být vyprodukovány za

pomocí dvou rozdílných produkčních pravidel, pak je gramatika nejednoznačná, na základě toho lze tvrdit, že gramatika není uzpůsobena k skrývání dat, protože zde není možnost ke korektnímu získání informací z takto vygenerovaného textu.

Start	->	**podmět přísudek** || **kdo co**
podmět	->	Petr || Radek
přísudek	->	si hraje || běhá
kdo	->	Petr || Radek
co	->	si hraje || běhá

Tabulka 3.4: Nejednoznačná gramatika

Při použití nejednoznačné gramatiky z tabulky 3.4 může věta „Petr si hraje“ vzniknout dvěma rozdílnými cestami. Zakódováním této věty by mohly vzniknout dvě různé sekvence bitů „000“ a „100“. Proto je zapotřebí, aby gramatika byla jednoznačná.

Pokud je bezkontextová gramatika v Greibachově Normální Formě, znamená to, že má neterminály na konci produkcí [13]. V tabulce 3.5 lze vidět, které produkce toto pravidlo splňují a které ne.

Produkce			je v GNF ?
Start	->	**podmět přísudek**	ANO
podmět	->	Petr || Radek	ANO
přísudek	->	si hraje **kde** || běhá **kde**	ANO
tmp	->	**kdy** s rodiči || **kdy** s kamarády	NE
tmp	->	**kdy sKým**	ANO

Tabulka 3.5: Nejednoznačná gramatika

Pokud některá produkce z gramatiky toto pravidlo nesplňuje, potom se musí převést do GNF [13]. Tento převod je velmi jednoduchý, stačí přidávat produkce, dokud celá gramatika není v GNF. V našem případě přetvoříme produkční pravidlo pro **tmp** tím, že terminály „s rodiči“ a „s kamarády“ nahradíme neterminálem **sKým**. Tím vznikne nové produkční pravidlo, pro neterminál **sKým**.

Start	->	**podmět přísudek**
podmět	->	Petr || Radek
přísudek	->	si hraje **kde** || běhá **kde**
kde	->	na hřišti **tmp** || okolo domu **tmp**
tmp	->	**kdy sKým**
kdy	->	večer || ráno
sKým	->	s rodiči || s kamarády

Tabulka 3.6: Produkční pravidla

Gramatika v tabulce 3.6 generuje úplně stejné věty, jak tomu bylo u gramatiky z tabulky 3.2. Jediný rozdíl je v tom, v jakém pořádí jsou produkční pravidla prováděna. V nově upraveném neterminálu **tmp** se neprovádí žádné rozhodnutí, takže nebude uložen žádný bit. Věta „Petr běhá okolo domu večer s rodiči“ nyní vznikne z bitové sekvence „01100“

Proces parsování z bezkontextové gramatiky, která je v Greibachově Normální Formě, je už jednoduchý. Funguje na podobném principu, který byl znázorněn při zakódování sekvence bitů „011000“ v tabulce 3.3. Mějme větu z předchozích příkladů „Petr běhá okolo domu večer s rodiči“, parsování takové věty je znázorněno v tabulce 3.7.

krok	parsovaná věta	výběr z produkce	získaný bit
1	Petr běhá okolo domu večer s rodiči	**podmět** -> Petr || Radek	0
2	Petr běhá okolo domu večer s rodiči	**přísudek** -> si hraje **kde** || běhá **kde**	1
3	Petr běhá okolo domu večer s rodiči	**kde** -> na hřišti **tmp** || okolo domu **tmp**	1
4	Petr běhá okolo domu večer s rodiči	**tmp** -> **kdy sKým**	-
5	Petr běhá okolo domu večer s rodiči	**kdy** -> večer || ráno	0
6	Petr běhá okolo domu večer s rodiči	**sKým** -> s rodiči || s kamarády	0

Tabulka 3.7: Parsování bitové sekvence z textu

K získání celé zakódované zprávy „01100“ je potřeba projít všech šest kroků, ve kterých se porovnávají jednotlivé produkční pravidla s parsovanou větou.

3.5.2 Sémantická metoda

V této metodě se pro ukrývání informace v textu používají záměny slov stejného nebo podobného významu (synonyma). Jednou z velkých výhod této metody, oproti metodám založených na formátování, je ochrana proti přepsání nebo použití různých formátovacích programů. Po uložení skryté informace může text dost výrazně změnit význam, oproti původnímu textu. [9]

Mohammadova metoda

Mohammad Shirali-Shahreza navrhl sémantickou metodu [9], která je schopna ukrýt data v Anglicky psaných textech. Metoda zakládá na tom, že psaní některých britských a amerických slov je jiné, ale podobné (UK - dialogue, US - dialog). Nahrazení takovýchto slov za jejich anglické nebo americké protějšky umožňuje skrýt bitové sekvence do textu. Vzniklý text je pak smíchanina anglických a amerických slov, kterých si určitě člověk znalý anglického jazyka všimne, ale pro použití v zemích, kde angličtina není primární jazyk a její znalost není na nějak extra vysoké úrovni, je tato metoda dobře použitelná a výsledný stego text nevzbudí tolik pozornosti.

Tato metoda musí mít předem připravenou tabulku slov [9] jak pro ukrývající, tak odkrývající část, které mají stejný význam a jinak se píší v Americe a Británii. Příklad

Americká slova	Britská slova
Color	Colour
Center	Centre
Criticize	Criticise
Liter	Litre
Theater	Theatre
Dialog	Dialogue
Flavor	Flavour
Check	Cheque
Labor	Labour
Neighbor	Neighbour

Tabulka 3.8: jinak psaná slova se stejným významem

takových slov znázorňuje tabulka 3.8. Ukrývací (enkódující) část této metody hledá v textu slova, která se shodují se slovy v tabulce. Když nalezne takovou shodu, tak vloží americké slovo do textu místo slova, které mělo shodu, to znamená že byl uložen bit 0. Britské slovo se vkládá, pokud je potřeba uložit bit 1. Takto bude uložena celá tajná zpráva v binární podobě. K získání tajné zprávy z takto přijatého stego textu slouží odkrývající (dekódující) část. Prochází se stego text a pokud narazí na slovo, které odpovídá nějakému slovu z tabulky, tak si uloží bit 0 pro americké slovo a bit 1 pro britské slovo, takto se zrekonstruuje bitová sekvence tajné zprávy. Nakonec se převede do pro člověka čitelné podoby.

Problém této metody je nízká kapacita ukrytých dat k velkému množství coverových textových dat. Takže není vůbec vhodná k ukrývání velkého množství dat, na druhou stranu se může hodit, když potřebujeme ukrýt malé množství dat. Metoda je docela nová, takže pravděpodobnost prolomení je nízká. [9]

3.6 Detekce a útoky

Existuje řada útoků [8], které mohou výsledný stego text poškodit, pozměnit nebo za pomocí testů a detektorů se může útočník pokusit přečíst ukryté data. Řada metod se těmto útokům snaží zamezit, již při návrhu samotné metody. Podle toho, jak je metoda schopná se těmto detekcím a útokům bránit je určována kvalita a použitelnost metody.

3.6.1 Detekce

- **Statistické testy** [8] - pomocí statistických testů zjištěním, že statistické vlastnosti samotného stego textu se vymykají z norem, se může odhalit, že cover text byl pozměněn.
- **Detekce mezer a neviditelných znaků** [8] - tato detekce je zaměřena výhradně na metody, které využívají bílých znaků pro ukrytí tajných dat. Bílé znaky mohou být vloženy mezi samostatné slova, věty, nebo jednotlivé paragrafy. Možností, kde vložit bílý znak navíc je mnoho. Pokud se stego text jeví tak, že má až moc bílých znaků, je vhodný kandidát na prověření. Samotné prověření probíhá v moderních textových procesorech, které jsou schopny tyto paterny odhalit.

3.6.2 Druhy útoků

- **stego-only attack** - tento útok se používá, když vlastníme jen stego text a chceme detekovat nebo rovnou vytáhnout skryté data.
- **known-cover attack** - k útoku dochází pokud vlastníme jak stego text, tak cover text, takže může dojít k porovnání stego a cover textů.
- **known-message attack** - pokud známe tajnou zprávu a stego text, můžeme najít metodu, která byla použita pro ukrytí tajné zprávy.
- **chosen-stego attack** - používá se, když známe stego text i metodu nebo stegonagrafický nástroj, kterým bylo médium vygenerováno.
- **chosen-message attack** - při tomto útoku se vygeneruje stego text za pomocí konkrétní metody, hledáním podobností s jinými stego texty umožní jejich odhalení.

Kapitola 4

Návrh a implementace metod

V této kapitole popisuji motivaci 4.1, proč jsem se rozhodl vytvořit grafické uživatelské rozhraní (GUI), moje požadavky na aplikaci 4.1.1, proč jsem zvolil takový programovací jazyk 4.1.2 a takový framework 4.1.3 pro vývoj GUI.

Následuje podrobnější popis implementovaných metod (4.2.1, 4.2.2, 4.2.3, 4.2.4), ke každé metodě je vyobrazen algoritmus pro enkódování a dekódovaní v pseudokódu (u Format based metody není dekódovací algoritmus, zdůvodněno v samotné sekci). Tyto algoritmy reflektují celkový postup, pro enkódování/dekódování v metodě, takže se nejedná přímo o algoritmus, který by byl aplikovatelný na jednu specifickou funkci.

Metody byly navrženy tak, aby na vstupech byly schopné přijmout jakýkoli text, ovšem jen se znaky kódovanými v ASCII. Proto nemohu zaručit funkčnost metod s jinými znakovými sadami.

Všemi metodami (kromě syntaktické) jsem se inspiroval z různých vědeckých článků, metoda Zero distribution 4.2.1 [12], sémantická metoda 4.2.2 [9] a Format based metoda 4.2.4 [10]. Syntaktickou metodu 4.2.3 jsem vytvořil na základě znalostí z lingvistické steganografie 3.5 a bezkontextových gramatik 3.5.1. Zdrojové kódy lze najít na CD ve složce source/. Metody jsou schopny přijmout na vstupu jen cover text ve formátu .txt.

4.1 Návrh aplikace

Pro demonstrování a vyzkoušení různých metod, jsem navrhl aplikaci s grafickým uživatelským rozhraním (dále jen GUI), která obsahuje čtyři metody. Každá tato metoda zakládá na různém principu enkódování a dekódování, to proto aby šly rozdíly a vlastnosti jednotlivých metod při enkódování/dekódování vidět a hlavně u výsledného stego textu, který je vygenerován/upraven pomocí těchto metod. Pro GUI jsem se rozhodl z důvodu přehlednosti a jednoduchosti při používání aplikace. Bez něj by si uživatel musel složitě do konzole přidávat cesty k pomocným souborům a nastavovat ostatní parametry. To podle mého názoru není vhodná cesta jak uspokojit uživatele aplikace.

Na CD ve složce app/ je přiložen soubor projekt.exe, kdterý je samostatně spustitelný. K jeho spuštění je potřeba mít všechny knihovny a pomocné soubory, které jsou již ve složce obsaženy. Je nutné zachování adresářové struktury.

4.1.1 Požadavky na aplikaci

Vzhledem k tomu, že v zadání bakalářské práce nebyly nijak zvlášť specifikované požadavky na implementaci, specifikoval jsem si je sám, podle vlastního uvážení.

Požadavky na metody

- schopnost uložit textové data (vytvoření stego textu)
- schopnost extrahovat textové data ze stego textu
- možnost vybrat si kterou metodu použijeme
- u některých metod možnost vybrat svoje vlastní slovníky/upravit gramatiku podle svého

Požadavky na GUI

- přehlednost aplikace
- intuitivnost ovládání aplikace
- informační hlášení o chybách a výsledcích

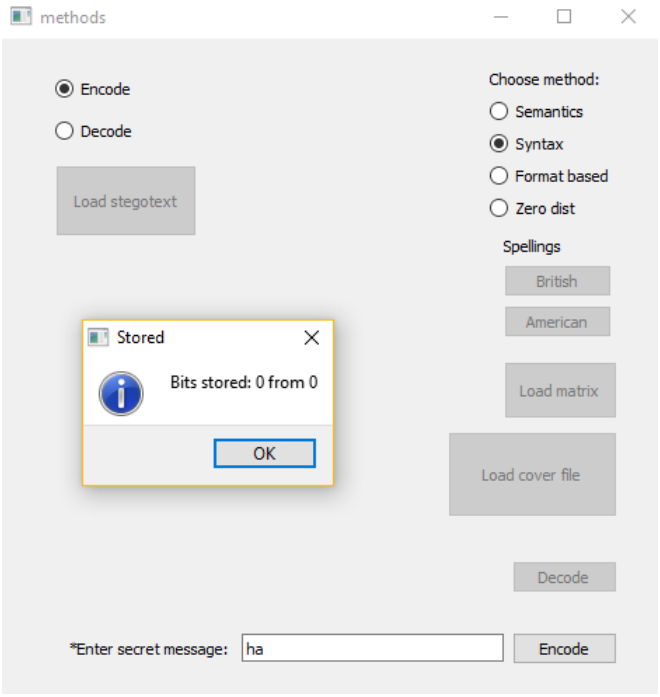
4.1.2 Implementační jazyk

Jako programovací jazyk jsem zvolil C++. Jeden z hlavních faktorů při výběru bylo více pozitivních zkušeností s tímto jazykem. Tento jazyk nabízí širokou škálu knihoven a jelikož se v něm dají provádět i objektově orientované návrhy (také nemusí), byl pro mě jasnou volbou. Ve vybraných metodách se hlavně pracuje s textem, různé měnění, rozšiřování, umazávání, vyhledávání v textu. Tyhle operace skvěle zvládá za použití vestavěných knihoven, proto jsem dal přednost před Cčkem, kde se většina těchto operací musí implementovat zvlášť. Programy napsané v C++ jsou také mnohem rychlejší, než programy napsané v jazyce, který potřebuje pro běh programu interpret. Rychlost programu pro mě byla taky cílovým faktorem, tudíž jsem dal přednost C++ před Javou, která je jazyk interpretovaný. Jedna z nevýhod C++ oproti Javě je ta, že se pro každý operační systém musí překládat zvlášť, u Javy stačí, aby měl operační systém k dispozici interpret Javy(virtuální stroj Javy).

4.1.3 Návrh grafického rozhraní

Pro návrh GUI jsem zvolil multiplatformní aplikační framework Qt. Návrh GUI mi usnadnilo vývojové prostředí Qt Creator, kde je možné navrhnout celý vzhled aplikace během pár minut, funkčnost různých tlačítek a přepínačů doimplementovat pomocí pár kliknutí. Qt je plně kompatibilní s projekty vytvořenými v C++, takže nebyl problém nejdřív navrhnout funkční backend čistě v C++ a až potom vytvářet vzhled aplikace(frontend) pomocí Qt Creatoru.

Samotný zdrojový soubor k funkčnosti tlačítek a XML soubor k vygenerování grafického rozhraní je k nalezení na CD v souborech source/methods.cpp a methods.ui. Na obrázku 4.1.3 je zobrazen vzhled prvotní verze aplikace.



Obrázek 4.1: Prvotní vzhled aplikace

4.2 Vybrané metody

Metody jsem vybíral podle různorodého stylu enkódování a dekódování. Vlastnosti stego textu a kapacita ukrytí tajných dat se u každé metody liší. U některých metod jde na první pohled poznat, že se jedná o podezřelý text, u dalších by bylo na odhalení tajné komunikace, potřeba zkoumat stego text více do hloubky.

4.2.1 Zero distribution

Zdrojové soubory k metodě lze nalézl v souborech na CD source/zero_dist/zero_dist.cpp a zero_dist.h. Pokud to jde, tak dojde k zredukování tajné zprávy pomocí akronymů, které lze nalézt v globálních polích *acronyms[]* a *meanings[]*. Implementace je provedena ve funkci *ZDreduceMSG()*. S využitím této funkčnosti na zkrácení textu lze dosáhnout efektivnějšího enkódování tajné zprávy. Čím více akronymů pro různé jazyky bude k dispozici, tím efektivnější tato funkce bude.

Algoritmus 1: Enkódování zprávy pomocí metody Zero distribution

```
Input: tajná zpráva TZ, cover text CT
Output: pole pozic M
převedení TZ do binární podoby ZB;
zjištění počtu bitů PZB binárního řetězce ZB;
S je pozice bitu v ZB;
převedení CT do binární podoby CTB;
zjištění počtu bitů PCTB binárního řetězce CTB;
C je pozice bitu v CTB;
S = 0;
C = 0;
while C < PCTB do
    if S < PZB then
        if bit z ZB na pozici S = bit z CTB na pozici C then
            ulož C na pozici S v M;
            S = S + 1;
        end if
        C = C + 1;
    end if
    else
        konec
    end if
end while
```

Algoritmus 1 naznačuje, jak probíhá ukrytí zprávy do cover textu. Celý cover text se musí nejprve převést do binární podoby, poté lze porovnávat jednotlivé bity cover textu s bity tajné zprávy, pokud se rovnají, uloží se pozice bitu v cover textu do **matice**, která se odesílá s cover textem, aby bylo možné zprávu zpětně dekódovat. Toto porovnání je implementované ve funkci *ZDmatchingBits()*. S touto metodou se původně používá i šifrování, pozice bitů v matici se vynásobí různými konstantamia a různě přehází. Pro zvýšení opatření proti odhalení tajné zprávy určitě není od věci, tuto funkčnost naimplementovat, ale pro demonstrativní účely nám postačí samotná metoda bez šifrování. Výstupní matici lze potom najít v souboru na CD app/zero_dist/output.data.

Algoritmus 2: Dekódování zprávy pomocí metody Zero distribution

Input: pole pozic M, cover text CT
Output: dekódovaná zpráva TZ

převedení CT do binární podoby CTB;
zjištění počtu bitů $PCTB$ binárního řetězce CTB;
C je pozice bitu v CTB;
F je řetězec dekódovaných bitů z CTB;
X je index pro M;
$X = 0$;
$C = 0$;
while $C < PCTB$ **do**
    if *pozice* C = *pozice uložená v* M *na indexu* X **then**
        do F se uloží bit z pozice C v CTB;
        $X = X + 1$;
    end if
    $C = C + 1$;
end while
převod na ascii znaky po 8mi bitech z F *a uložení do* TZ;

Pro dekódování je potřeba mít jak cover text, tak matici pozic bitů, podle kterých vytáhneme jednotlivé bity z cover textu, tím se složí samotné písmena. Jádro algoritmu 2, jak se bity dostávají z cover textu pomocí matice, reflektuje funkce *ZDencrypt()*. Dekódovanou zprávu lze najít v souboru na CD app/zero_dist/output.txt.

4.2.2 Sémantická

Zdrojové soubory k metodě lze nalézl v souborech na CD source/semantics.cpp a semantics.h. Jako vhodného reprezentanta sémantických metod jsem zvolil Mohammadovu [9] metodu. Hlavní princip je v tom, že jsou k dispozici dva slovníky britský a americký se slovy, které mají stejný význam ale jinak se píší, na základě toho, jestli potřebujeme uložit jedničku nebo nulu, se potom zvolí slovo z těchto slovníků a nahradí se v cover textu. Tyto slovníky jsou volně dostupné z webu [2], předem jsem připravil jak britský, tak i americký ve složce na CD app/semantics/ nazvané uk_spelling.txt a us_spelling.txt. Uživatel si při enkódování i dekódování musí tyto soubory nahrát, to proto, že jsem chtěl aby měl možnost si nahrát i svoje vlastní připravené slovníky a tím jsou otevřeny dveře i pro jazyky, kde je podobné nářečí, například by šlo vytvořit slovníky českých a hanáckých slov, to by se pak dal použít i český cover text. Při použití předpřipravených slovníků je kritické, aby byl cover text anglický, jinak by nemělo význam používat tyto slovníky.

Algoritmus 3: Enkódování zprávy pomocí Sémantické metody

Input: britské slova UK, americké slova US, cover text CT, tajná zpráva TZ
Output: změněný CT

převedení TZ do binární podoby ZB;
C je pozice bitu v ZB;
P je pozice v seznamech UK a US;
$C = 0$;
while *konec* CT **do**
  if $S ==$ *slovo na pozici* P *v* UK **then**
    if C. *bit z* $ZB == 0$ **then**
      nahradí se S z CT za slovo na pozici P v US;
    end if
    $C = C + 1$;
  end if
  if $S ==$ *slovo na pozici* P *v* US **then**
    if C. *bit z* $ZB == 1$ **then**
      nahradí se S z CT za slovo na pozici P v UK;
    end if
    $C = C + 1$;
  end if
end while

Algoritmus 3 pro enkódování zprávy zobrazuje jak tajná zpráva přechází do binární podoby (funkce *SEasciiConversion()* a *SEasciiToBinary()*) a nakonec se pomocí ní zaměňují slova (né vždy to jde, záleží na vhodném cover textu) v cover textu a vzniká stego text. Samotné vyhledávání a nahrazování slov probíhá ve funkci *SEfindAtransform()*, která vytvoří výsledný stego text, funkce *SEcheckCS()* zajišťuje že porovnávání slov z cover textu se slovy ze slovníků bude case insensitive (nezáleží na velikosti písmen).

Algoritmus 4: Dekódování zprávy pomocí Sémantické metody

Input: stego text ST, britské slova UK, americké slova US
Output: dekódovaná zpráva TZ

S je slovo z ST;
F je rětězec dekódovaných bitů z ST;
while konec ST **do**
  načte se S;
  if S v UK **then**
    $F = F +$ "1";
  end if
  if S v US **then**
    $F = F +$ "0";
  end if
end while
převod na ascii znaky po 8mi bitech z F *a uložení do* TZ;

Algoritmus 4 znázorňuje dekódování textu s využitím předpřipravených slovníků. Funkce *SEdecodeStego()* dekóduje vstupní stego text a tajná zpráva se objeví v souboru na CD app/semantics/secret.txt. Pokud narazí na slovo, které je v jednom ze slovníků, dekóduje bit 1 (když se shodovalo se slovem z britského slovníku) a bit 0 (když se shodovalo se

slovem z amerického slovníku). Z toho plynou jasné nedostatky, jeden z nich je, když bude mít stego text více takovýchto slov, které nebyly enkódovány, pak k tajné zprávě přibude změť náhodných redundantních znaků. Nebo naopak, pokud bude cover text slabý a enkodér nedokázal uložit celý byte, tak se pak těžko bude moci tajná zpráva číst.

4.2.3 Syntaktická

Zdrojové soubory k metodě lze nalézl v souborech na CD source/syntax/syntax.cpp a syntax.h. Tuto metodu jsem vybral jako vhodného a nejlepšího reprezentanta Lingvistické steganografie. Samotný stego text vzniká z navržené gramatiky. Pro jednoduchost a demonstraci metody jsem navrhl jednoduchou gramatiku, která obsahuje čtyři neterminály (**podmět**, **přísudek**, **s kým** a **kde**). Samotná věta se pak skládá z neterminálů: **podmět přísudek s kým kde** (+ tečka na konci). Skládání věty může kdykoli skončit (pokud došly bity k uložení), takže se může stát, že věta bude obsahovat jen podmět, to je ale detail, který by se dal v budoucnu řešit pomocí přidání pár ifů, ty by za cenu redundance přidaly náhodně další slova. Gramatika je uložena v souboru na CD app/syntax/grammar.txt, jde do ní přidávat nové terminály, vždy aby byl výsledný počet terminálů pro jeden neterminál mocninou dvou (2, 4, 8, 16, atd..). Gramatika se parsuje a načítá ve funkci *SYreadGrammar()*, kde dochází k naplnění vektorů *subject*, *prejudice*, *where* a *with*. Čím víc jeden neterminál bude mít terminálů, tím víc je schopen právě jeden terminál uložit bitů. Tvoření vět lze nalézt ve funkci *SYmakeSentences()*, která vrátí výsledný stego text.

Algoritmus 5 znázorňuje jak probíhá zpracování vstupní gramatiky a následné skládání vět. Výstupem této metody je textový soubor, který lze po enkódování nalézt ve složce na CD source/syntax/ s názvem stegotext.txt.

Algoritmus 5: Enkódování zprávy pomocí Syntaktické metody

Input: tajná zpráva TZ, gramatika G
Output: stego text ST

převedení TZ do binární podoby ZB;
zjistí se počet bitů D v ZB;
terminál T z G;
výsledek V převodu bitů do desítkové soustavy;
C je pozice bitu v ZB;
G obsahuje pole terminálů SU, PR, WH, WI;
DSU, DPR, DWH, DWI jsou počty bitů potřebné k zakódování všech T v SU, PR, WH, WI;

```
while C < D do
    zjistí se VDSU bitů od pozice C;
    vybere se T na V. pozici z SU;
    ST = ST + T + "";
    C = C + DSU;
    if C < D then
        zjistí se VDPR bitů od pozice C;
        vybere se T na V. pozici z PR;
        ST = ST + T + "";
        C = C + DPR;
    end if
    if C < D then
        zjistí se VDWH bitů od pozice C;
        vybere se T na V. pozici z WH;
        ST = ST + T + "";
        C = C + WH;
    end if
    if C < D then
        zjistí se VDWI bitů od pozice C;
        vybere se T na V. pozici z WI;
        ST = ST + T + "";
        C = C + WI;
    end if
    ST = ST + ".";
end while
```

Nedílnou součástí implementované metody je i dekódování přijatého stego textu, který byl vygenerován pomocí známé gramatiky. O to se stará funkce *SYdecode()*. Bez ní by nebylo možné tajnou zprávu ze stego textu dekódovat. Možné by to teda bylo, ale bylo by potřeba velké množství vzorků stego textu vygenerováno z této gramatiky, které by následně byly zanalyzovány, až potom by šla sestavit původní gramatika. S takovým případem ale nepočítám, tudíž je na vstupu při dekódování potřeba gramatika ze které byl stego text

vygenerován. Parsování vstupního stego textu je znázorněno v algoritmu 6. Dekódovanou zprávu lze nalézt v souboru na CD app/syntax/output.txt.

Algoritmus 6: Dekódování zprávy pomocí Syntaktické metody

Input: stego text ST, gramatika G
Output: dekódovaná zpráva TZ

S je slovo z ST;
F je rětězec dekódovaných bitů z ST;
G obsahuje pole terminálů SU, PR, WH, WI;
binární reprezentace B indexu pole z $SU|PR|WH|WI$;
while konec ST **do**
  načte se S;
  if S v SU **then**
    $F = F + B$;
  end if
  if S v PR **then**
    $F = F + B$;
  end if
  if S v WH **then**
    $F = F + B$;
  end if
  if S v WI **then**
    $F = F + B$;
  end if
end while
převod na ascii znaky po 8mi bitech z F *a uložení do* TZ;

Syntaktická metoda má obrovský potenciál, budoucnost této metody vidím v tom, že by se přidaly další gramatiky a věty by se skládaly různorodě, podle náhodného systému. Tudíž by nedocházelo k viditelně opakovanému textu, jak je tomu teď. V mém případě je jen jedna velice jednoduchá až primitivní gramatika, což se projevuje na výstupním stego textu. Na první pohled i laikovi je jasné, že s textem není něco v pořádku.

4.2.4 Format based

Tato metoda ukrývá tajnou informaci pomocí formátování textu. Metoda je vyloženě závislá na tom, že potřebuje bohatý textový formát(.doc, .docx, atd..), aby se dalo vhodně manipulovat s textem. Vstupní formát, pro jednoduchost a demonstraci samotné metody, jsem zvolil obyčejný .txt, v budoucnu by se dalo přejít na vstupní formát .doc, který je zároveň i výstupní, nedocházelo by tedy k žádným podezřením v tomto ohledu. V mnou implementovaném algoritmu dochází k ukládání po dvou bitech, jelikož jsou u této metody k dispozici čtyři rozdílné funkce na formátování textu. Implementování dekódování tajné zprávy pomocí této metody by zabralo mnoho času, tudíž jsem se rozhodl dekódování vynechat. Zdrojové kódy k této metodě je možné najít na CD v souboru source/format_based/format_based.cpp a format_based.h. Ve funkci *FBmakeStegoText()* je implementováno rozhodování pro každý řádek z cover textu, zda se bude zmenšovat/zvětšovat velikost fontu, přidávat mezera za každou mezeru nebo speciální znak. Samotné zmenšování/zvětšování fontu/přidávání mezer je implementováno ve funkcích *FBexpand()*, *FBshrink()*, *FBaddSpace()*, *FBaddSpecial()*. Každá z těchto funkcí zdeformuje vstupní cover text, čímž vznikne zdeformovaný cover text

a ten je výstupem celé metody. Ke změně formátu textu využívám vkládání XML příkazů, kterým rozumí a zpracovává výsledný .doc formát, možno vidět ve výše zmíněných funkcích. Zdrojové kódy dále obsahují pomocné funkce, pro práci s textovými řetězci, převody z/do ASCII a binární formy.

Algoritmus 7: Enkódování zprávy pomocí Format based metody

Input: tajná zpráva TZ, cover text CT
Output: transformovaný CT

převedení TZ do binární podoby ZB;
zjištění počtu bitů D binárního řetězce ZB;
C je pomocné počítadlo;
$C = 0$;
počet dvojic $B = D/2$;
P je pozice bitu v ZB;
N je číslo řádku CT;
$N = 0$;
Q je celkový počet řádků CT;
if $D\%2! = 0$ **then**
  $ZB =$ "0" $+ ZB$;
  aktualizuje se D;
end if
while $C < B \&\& N < Q$ **do**
  načte se dvojce bitů DB na pozici P a $P + 1$;
  načte se N. řádek z CT;
  if $DB ==$ "00" **then**
    zmenšení fontu N. řádku;
  end if
  if $DB ==$ "11" **then**
    zvětšení fontu N. řádku;
  end if
  if $DB ==$ "01" **then**
    na N. řádku se za každou mezeru přidá další mezera;
  end if
  if $DB ==$ "10" **then**
    na N. řádku se za každý speciální znak přidá mezera;
  end if
end while

Algoritmus 7 znázorňuje jak probíhá rozhodování a co vše je potřeba provést před samotným procházením a ukrýváním tajné zprávy do cover textu. Tato metoda je schopna uložit celý byte na čtyři řádky, to mě vede k myšlence, že je vhodná pro ukládání skryté informace do básní a podobných textů, kde se vyskytuje mnoho řádků. Na druhou stranu je absolutně nepoužitelná, pokud cover text nemá žádný nebo dostatečný počet řádků. Při převodu tajné zprávy do binární podoby, pro zvýšení bezpečnosti, kdyby došlo k odhalení zprávy, by se dalo zakomponovat i kódování, například převod z osmi bitů na deset. Výstupní stego text lze najít v souboru na CD app/format_based/output/stegotext.doc.

Kapitola 5

Experimenty

V této kapitole se budu věnovat dosaženým výsledkům naimplementovaných steganografických metod. Podívám se na tři hlavní vlastnosti, které se dají u výstupního stego textu těchto metod sledovat, těmi jsou postřehnutelnost, robustnost a kapacita ukrytých dat. V sekci 5.1 lze vidět tabulky s výstupními hodnotami, které jsem získal za použití různých cover textů a různých tajných zpráv, které se liší především ve velikosti souboru. Pro cover texty jsem využil výplňového textu Lorem ipsum [4] a textu z eknihy [3]. Samotné texty, které ukrývám, jsem získal z Lorem ipsum. V sekci 5.2 potom porovnám a shrnu výsledky všech metod dohromady. Nakonec navrhnu vylepšení a možné využití metod 5.3.

5.1 Vlastnosti naimplementovaných metod

Pro robustnost a kapacitu ukrytých dat jsem navrhl sadu testů, které vypovídají o těchto dvou vlastnostech u stego textu.

Kapacitu testuji pomocí různých cover a secret textů, kde sleduji počet bitů, které metoda ukryla a velikost výsledného stego textu. Vstupní cover texty lze najít ve složce na CD source/covers/, zvolil jsem různé velikosti textu, většina je vygenerovaná pomocí nástroje a jeden je text reálný (napsaný člověkem). V tabulce 5.1 je přehled použitých cover textů. Ve složce na CD source/secrets/ lze najít sadu textů, které jsem zvolil pro ukrývání do cover textů, názvy a velikosti lze vidět v tabulce 5.2. Velikosti použitých souborů dále neuvádím. Syntaktická metoda se jako jediná z ostatních naimplementovaných metodách liší v tom, že nepotřebuje cover text a výsledný stego text si generuje sama.

název	cover1.txt	cover2.txt	cover3.txt	cover4.txt	cover5.txt
velikost [B]	1	5	318	969	188194

Tabulka 5.1: Názvy a velikosti cover textů

název	secret1.txt	secret2.txt	secret3.txt	secret4.txt
velikost [B]	1	5	776	32849

Tabulka 5.2: Názvy a velikosti textů k ukrytí

Robustnost testuji pomocí různých operací, které nějakým způsobem mění vstupní stego text, přidáním/odebráním/změněním znaků. Při dekódování sleduji co se děje s výstupními soubory v kterých očekávám dekódovanou tajnou zprávu.

Postřehnutelnost by se těžko testovala strojově, tudíž jsem zvolil vlastní zrak a sleduji jak vypadá výstupní text. Zaměřuji se na velikost, počet souborů a vzhled textu (formát, syntaxi, sémantiku).

5.1.1 Zero distribution

Zero distribution metoda má jeden zásadní problém a tím je, že potřebuje pro dekódování k původnímu cover textu další pomocný soubor, vygenerovaný na straně enkodéru. Problém je v tom, jak tento pomocný soubor dostat k příjemci, kterému byl určen, tím se zabývat ale nebudu, to už je kapitola sama o sobě(jak spolehlivě a nepozorovaně přenášet soubory).

	cover1.txt	**cover2.txt**	**cover3.txt**	**cover4.txt**	**cover5.txt**
secret1.txt					
Ukryto bitů [%]	100	100	100	100	100
Velikost matice [B]	24	24	24	24	25
secret2.txt					
Ukryto bitů [%]	30	100	100	100	100
Velikost matice [B]	39	150	152	152	153
secret3.txt					
Ukryto bitů [%]	0,2	0,75	20,8	63,56	100
Velikost matice [B]	39	170	6988	22474	36411
secret4.txt					
Ukryto bitů [%]	0,005	0,02	0,49	1,5	100
Velikost matice [B]	39	170	6988	22474	1983457

Tabulka 5.3: Kapacita ukrytých dat Zero distribution metody

V tabulce 5.3 lze vidět velikost pomocného souboru *Velikost matice*, výsledná velikost stego textu je tedy součet *pomocneho_souboru* + *cover_text* (např. 1983457 + 188194 = 2171651 bytů pro cover5.txt a secret4.txt). Když pomineme nadměrnou velikost stego textu, tak se s metodou dají ukrývat celkem snadno i velké objemy dat.

Postřehnutelnost: U této metody musíme vzít v potaz, že s cover textem musí putovat i pomocný soubor, který obsahuje pozice bitů v cover textu k dekódování. V tomto případě je rizikovost odhalení tajné komunikace vysoká. Samotný cover text nijak podezřelý není, to se ale nedá říct o matici, která obsahuje indexy bitů.

Robustnost: Jelikož dekódovací matice obsahuje indexy, určující který bit z cover textu vybrat pro složení tajné zprávy, nastává problém, když se cover text nějakým způsobem posune, tím dojde k naprostému zničení části zprávy (od posunutí až do konce). Změna znaku například změní jeden znak v tajné zprávě, pokud se tedy nezmění mnoho znaků, je zpráva s dávkou fantazie stále čitelná.

V tabulce 5.4 je přehled operací provedených nad cover textem a výstupů (ve formě tajné zprávy) po dekódování.

Operace	Výstup	Chování
odřádkování v půlce cover textu	část textu, zbytek změť znaků	dekóduje se vše
přidání mezery v půlce cover textu	část textu, zbytek změť znaků	dekóduje se vše
smazání mezery v půlce cover textu	část textu, zbytek změť znaků	dekóduje se vše
změna znaku	celý text, pár znaků změněných	dekóduje se vše
přidání znaků na konec cover textu	celý text	dekóduje se vše
operace na začátku cover textu	změněný celý text	dekóduje se vše

Tabulka 5.4: Operace a výstupy Zero distribution metody při dekódování

5.1.2 Sémantická

V tabulce 5.5 lze vidět úspěšnost ukrytí dat pomocí této metody. V prvních čtyřech cover textech nebyla schopná ukrýt ani bit, jelikož se jednalo o texty Lorem Ipsum, tam se těžko bude vyskytovat nějaké anglické slovo. V posledním cover textu cover5.txt, který byl napsaný v angličtině už byla schopná zakódovat až 289 bitů pro tento určitý text o velikosti 188 194 bytů.

	cover1.txt	**cover2.txt**	**cover3.txt**	**cover4.txt**	**cover5.txt**
secret1.txt					
Ukryto bitů [%]	0	0	0	0	100
Velikost stegotextu [B]	1	5	318	969	188324
secret2.txt					
Ukryto bitů [%]	0	0	0	0	100
Velikost stegotextu [B]	1	5	318	969	188333
secret3.txt					
Ukryto bitů [%]	0	0	0	0	4,65
Velikost stegotextu [B]	1	5	318	969	188408
secret4.txt					
Ukryto bitů [%]	0	0	0	0	0,11
Velikost stegotextu [B]	1	5	318	969	188408

Tabulka 5.5: Kapacita ukrytých dat semantické metody

Postřehnutelnost: podezřívat stego text z této metody, by bylo možné jen pokud by autor vstupního textu byl známý a rodilý mluvčí v jazyku, v kterém je text napsaný (angličtina) a samozřejmě čtenář by musel taky perfektně znát rozdíly mezi psaním britských a amerických slov. Velikost výstupního stego textu, není nijak rozdílná od vstupního cover textu. Z toho vyvozuji závěr, že postřehnutelnost u této metody je téměř nulová.

Robustnost: pokud se v cover textu nachází více slov, které by mohly být zakódovány, projeví se to potom při dekódování stego textu, ve výstupním souboru se za tajnou zprávou

objeví změť znaků. S tím se ale musí počítat, jelikož algoritmus předem nezná, jak dlouhý má být zakódovaná zpráva, proto je potřeba projít text celý a sestavit znaky i z bitů, které neprošly zakódováním. Pokud se při některé operaci znázorněné v tabulce 5.6 netrefí do slova, které slouží pro uložení bitů, nedojde k žádným změnám, které by měly vliv na dekódování tajné zprávy.

Operace	**Výstup**	**Chování**
odřádkování v půlce stego textu	celý text	dekóduje se vše
přidání mezery v půlce stego textu	celý text	dekóduje se vše
smazání mezery v půlce stego textu	celý text	dekóduje se vše
změna znaku	celý text	dekóduje se vše
přidání znaků na konec stego textu	celý text	dekóduje se vše
jakákoli operace na začátku stego textu	celý text	dekóduje se vše

Tabulka 5.6: Operace a výstupy sémantické metody při dekódování

5.1.3 Syntaktická

Výsledky z tabulky 5.7 byly pořízeny s využitím jedné gramatiky. Použitou gramatiku lze nalézt na CD syntax/grammar.txt. Metoda hravě zvládne ukrýt i velké množství dat. Po zprůměrování výsledků z tabulky 5.7 docházím k závěru, že na 1 byte ukrytých dat, se v průměru vytvoří 29.045 bytů stego textu. To je způsobeno slabou gramatikou, stačilo by zvětšit počet terminálů v jednotlivých neterminálech a výrazně by se poměr velikosti ukrývaného textu na stego text zmenšil.

	Velikost stegotextu [B]	Ukryto bitů [%]
secret1.txt	26	100
secret2.txt	135	100
secret3.txt	29317	100
secret4.txt	832737	100

Tabulka 5.7: Kapacita ukrytých dat syntaktické metody

Postřehnutelnost: Ze čtenářského hlediska je výsledný text téměř nepřijatelný, věty mají stejnou délku, více než podobný tvar a hlavně se můžou opakovat, to považuji za zásadní nedostatek a postřehnutelnost je proto vysoká.

Robustnost: Syntaktická metoda je velmi náchylná na jakékoliv změny ve stego textu, stačí přidat nebo odebrat znak a od toho místa je zbytek zprávy nedekódovatelný, protože už neodpovídá gramatice.

Operace	Výstup	Chování
odřádkování v půlce stego textu	část textu, zbytek ztracen	dekodování se zastaví
přidání mezery v půlce stego textu	část textu, zbytek ztracen	dekodování se zastaví
smazání mezery v půlce stego textu	část textu, zbytek ztracen	dekodování se zastaví
změna znaku	část textu, zbytek ztracen	dekodování se zastaví
přidání znaků na konec stego textu	celý text	nic navíc se nedekóduje
operace na začátku stego textu	nic se nedekóduje	dekodování se zastaví

Tabulka 5.8: Operace a výstupy syntaktické metody při dekódování

5.1.4 Format based

Maximální počet bitů, které je tato metoda schopna uložit lze předem od oka zjistit, jelikož kóduje dva bity na jeden řádek. Například z tabulky 5.9 lze zjistit počet řádku, pomocí maxima bitů, které metoda byla schopná zakódovat, pro secret3.txt a cover4.txt, což je 56 z toho $56/2 = 28$ řádků. Pro úspěšné ukrytí celé zprávy, bych volil cover text s dostatečným počtem řádků. Velikost výsledného stego textu se výrazně liší od velikosti vstupního cover textu, to je způsobeno převodem .txt formátu do .doc, nastavováním atributů pro každý řádek, z toho potom velikost výstupního textu naroste.

	cover1.txt	**cover2.txt**	**cover3.txt**	**cover4.txt**	**cover5.txt**
secret1.txt					
Ukryto bitů [%]	25	25	100	100	100
Velikost stegotextu [B]	87	91	589	1280	194983
secret2.txt					
Ukryto bitů [%]	5,55	5,55	50	100	100
Velikost stegotextu [B]	87	91	776	1751	195499
secret3.txt					
Ukryto bitů [%]	0,04	0,04	0,33	1,03	100
Velikost stegotextu [B]	87	91	630	1902	286969
secret4.txt					
Ukryto bitů [%]	0,0009	0,0009	0,008	0,02	2,8
Velikost stegotextu [B]	87	91	776	2028	306366

Tabulka 5.9: kapacita ukrytých dat Format based metody

Postřehnutelnost: U této metody se text nejeví úplně nevině, když pomineme změnu formátu, někomu může přijít nápadné, že pro každý řádek je jiné formátování. Z tohoto důvodu by někdo mohl stego text prohlásit za podezřelý, nebo dokonce dekódovat tajnou zprávu. Postřehnutelnost je proto velmi vysoká.

Robustnost: U této metody nemám možnost testovat robustnost, proto se pokusím nastínit jak by mělo fungovat dekódování se změněným stego textem. Přidáním znaku navíc, by nemělo mít na dekódovanou zprávu vliv, problém by byl, kdyby si někdo hrál s

formátováním, automatickým odstraněním nadbytečných mezer nebo přidáváním odstavců, to by už výrazně poničilo dekódovanou zprávu.

5.2 Porovnání výsledků

Každá z těchto metod má své nedostatky a výhody, ty v této sekci shrnu.

Nedostatky u Zero distribution metody bych rozhodně viděl v potřebě mít dva soubory pro dekódování stego textu, zároveň jeho výslednou velikost, naopak je za tuto cenu schopná ukrýt velké množství dat a zachovat přitom čitelnost, s robustností na tom taky není zase tak špatně.

Sémantická metoda je až moc závislá na vstupním cover textu, pokud ho uživatel špatně zvolí, může zapomenout na zakódování byť jednoho bitu. Na druhou stranu, když už data ukryje, text při tom zůstane čitelný a k dekódování nejsou potřeba žádné další pomocné soubory (když nepočítám briský a americký slovník). Metoda je zároveň hodně robustní. Při Enkódování i dekódování lze vidět, že tato metoda je viditelně nejpomalejší, jelikož pro každé slovo musí projít oba slovníky (může zabrat i několik sekund).

Syntaktická metoda je z těchto metod na tom kapacitně nejlépe, vygeneruje tolik stego textu, kolik je potřeba, takže se nemusíme bát, že by nějaký bit nebyl ukryt. Avšak je to na úrok postřehnutelnosti i robustnosti, kde se ani zdaleka nechytá.

Hlavní nevýhodou Format based metody je změna formátu textu, nárust velikosti stego textu oproti coveru a taky závislost na počtu řádků v cover textu.

Metoda	Kapacita	Postřehnutelnost	Robustnost
Zero dist.	2	3	2
Sémantická	4	1	1
Syntaktická	1	4	4
Format based	3	2	3

Tabulka 5.10: ohodnocení metod

V tabulce 5.10 jsou zhodnoceny vlastnosti metod, oznámkované od 1 do 4 (1 = nejlepší). Po zprůměrování známek z tabulky vychází, že nejlépe dopadla sémantická metoda. Pro výběr a ohodnocení metody si musí každý sám určit váhu pro vlastnosti, kteteré od stego textu vyžaduje.

5.3 Vylepšení a využití

U sémantické metody je možnost rozšířit slovníky podobných slov i přes to by se kapacita o moc nezlepšila. K viditelné změně na efektivnosti by se slovníky musely rozšířit až o několikanásobek současného počtu slov. Vzhledem k téměř nulové postřehnutelnosti, silné robustnosti a slabé kapacitě, je tuto metodu dobré používat pro ukrývání malého počtu dat.

Podobný případ je to u syntaktické metody, kde ke zlepšení postřehnutelnosti je potřeba navrhnout mnohonásobně větší gramatiku a né jednu, hned několik a náhodně z těchto gramatik vybírat, aby nedocházelo k tak časté repetitivnosti. U robustnosti je to už implementační problém, kde je potřeba vylepšit dekódovací algoritmus, slova která nepatří do gramatiky vynechávat a pokusit se získat alespoň nějaké data ze slov následujících,

které už třeba nemusí být poškozené. Metoda je vhodná pokud chceme uložit velké množství dat, přitom nám nevadí, že výsledný stego text vypadá podezřele a je vysoce náchylný na jakékoliv změny.

U Format based metody by se dalo přidat několik dalších funkcí, které by formátovaly text a tím dosáhnout větší efektivnosti při skrývání tajné zprávy. Pro zlepšení postřehnutelnosti bude určitě vhodné do budoucna doimplementovat podporu, kde na vstupu algoritmus příjme formát .doc.

Pro zlepšení postřehnutelnosti u Zero distribution metody by se matice pro dekódování měla posílat zvlášť a nejlíp zašifrovaná. Z hlediska použitelnosti je na tom velice podobně jak Format based metoda, takže když nám stačí od stego textu všechny vlastnosti průměrné, vyplatí se zvolit jednu z těchto dvou metod.

Kapitola 6

Závěr

Cílem této bakalářské práce bylo seznámit se s problematikou ukrývání dat do digitálního textového souboru a naimplementovat/navrhnout/modifikovat několik existujících/vlastních metod, demonstrovat funkčnost vybraných metod a porovnat vlastnosti stego textů z hlediska skrývání a odkrývání dat. Tento cíl byl splněn.

Při řešení práce jsem se seznámil s dávnou historií steganografie až po novodobé principy ukrývání dat do digitálních textových souborů. V dalším průběhu práce jsem se při návrhu aplikace seznámil s multiplatformním frameworkem Qt, který byl použit pro vývoj grafického uživatelského rozhraní. Aplikace je tedy intuitivní a rychle pochopitelná, umožňuje ukrýt data do textu pomocí čtyř různých metod, převzaté metody prošly mírnou modifikací (vynechání šifrování atd). Algoritmy a popsaná funkčnost metod je uvedena ve čtvrté kapitole.

Nad naimplementovanými metodami byla provededa sada experimentů, které otestovali tři základní vlastnosti stego textu (kapacita, robustnost, postřehnutelnost). Shrnutí těchto vlastností je pro každou metodu uvedeno v páté kapitole.

Byla navržena aplikace pro windows, která je schopná enkódovat a dekódovat tajnou zprávu z textu. Na základě velikosti byla textová informace vybrána jako vhodný typ informace pro skrývání do textových souborů. Jelikož se nepočítá s tím, že aplikace bude pracovat s jinými, než anglickými texty, byla zvolena ASCII znaková sada. Zdrojové kódy naimplementovaných metod a aplikace jsou přiloženy na CD.

Literatura

[1] ASCII Table and Description. https://www.asciitable.com/, [cit. 2018-04-30].

[2] Comprehensive list of American and British spelling differences. http://www.tysto.com/uk-us-spelling-list.html, [cit. 2018-04-30].

[3] FM8 Operation Manual. https://www.native-instruments.com/fileadmin/ni_media/producer/fm8/downloads/FM8-book_EN_ebook.pdf, [cit. 2018-04-30].

[4] Lorem Ipsum. https://www.lipsum.com/, [cit. 2018-04-30].

[5] Ahmed Saeed, M. K. I., Md Baharul Islam: A Novel Approach for Image Steganography Using Dynamic Substitution and Secret Key. ročník 2, 9 2013: s. 118–126.

[6] Bennett, K.: *Linguistic steganography: survey, analysis, and robustness concerns for hiding information in text.* Technická zpráva, Purdue University, Center for Education and Research in Information Assurance and Security, 2004.

[7] Fairhurst, G.: Manchester Encoding. http://www.erg.abdn.ac.uk/users/gorry/course/phy-pages/man.html, [cit. 2018-04-30].

[8] Kipper, G.: *Investigator's Guide to Steganography.* Auerbach Publications, 2004, ISBN 0-8493-2433-5.

[9] M. H. Shirali-Shahreza and M. Shirali-Shahreza: *A New Synonym Text Steganography.* In *Intelligent Information Hiding and Multimedia Signal Processing, 2008. IIHMSP '08 International Conference on [online]*, IEEE, 2008, ISBN 978-0-7695-3278-3, s. 1524–1526, doi:10.1109/IIH-MSP.2008.6, [cit. 2018-04-29].

[10] Roy, S.; Manasmita, M.: A novel approach to format based text steganography. In *Proceedings of the 2011 International Conference on Communication, Computing and Security*, 01 2011, s. 511–516.

[11] Thampi, S. M.: Information Hiding Techniques: A Tutorial Review [online]. 2008, [cit. 2018-04-29].

[12] Virendra Kumar Shivani, S. B., Saumya Yadav: A Novel Approach of Bulk Data Hiding using Text Steganography. In *Procedia Computer Science [online]*, ročník 57, Elsevier B.V, 2015, ISSN 1977-0509, s. 1401–1410, doi:10.1016/j.procs.2015.07.457, [cit. 2018-04-29].

[13] Wayner, P.: *Disappearing Cryptography: Information Hiding: Steganography and Watermarking*. 12 Academic Press, 2009, ISBN 978-0-12-374479-1.

Příloha A

Obsah CD

Veškeré zdrojové soubory jsou uloženy na přiloženém CD. Tento text slouží jako přesný popis adresářové struktury na CD:

- **app/** Samostatně spustitelnou aplikaci
- **latex/** Zdrojové soubory a obrázky potřebné pro vytvoření pdf této práce
- **pdf** Tento pdf soubor
- **source/** Zdrojové soubory aplikace

Příloha B

Manuál

Tento text slouží jako návod pro překlad aplikace

- Stáhnout Qt manager z `https://www.qt.io/download-qt-installer?hsCtaTracking=9f6a2170-a938-42df-a8e2-a9f0b1d6cdce%7C6cb0de4f-9bb5-4778-ab02-bfb62735f3e5`
- Nainstalovat verzi Qt 5.5.1
- Otevřít projekt source/projekt.pro
- Překládat s konzolí Desktop Qt 5.5.1 MinGW 32bit