Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Steganografické šifry

2019

Bc. Ondřej Kašpar

Vedoucí práce: doc. RNDr. Miroslav Kolařík, Ph.D.

Studijní obor: Informatika, prezenční forma

Bibliografické údaje

Autor:	Bc. Ondřej Kašpar
Název práce:	Steganografické šifry
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2019
Studijní obor:	Informatika, prezenční forma
Vedoucí práce:	doc. RNDr. Miroslav Kolařík, Ph.D.
Počet stran:	78
Přílohy:	1 DVD
Jazyk práce:	český

Bibliograhic info

Author:	Bc. Ondřej Kašpar
Title:	Steganography cyphers
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2019
Study field:	Computer Science, full-time form
Supervisor:	doc. RNDr. Miroslav Kolařík, Ph.D.
Page count:	78
Supplements:	1 DVD
Thesis language:	Czech

Anotace

Práce stručně zmiňuje historii steganografie, popisuje metody digitální steganografie (ukrývání zpráv do textu, obrázků a videonahrávek), vysvětluje základy steganoanalýzy a uvádí čtenáře do základů kryptografie. Následně obsahuje popis implementace vybraných steganografických šifer.

Synopsis

This thesis briefly mentions the history of steganography, describes the methods of digital steganography (hiding messages in texts, pictures and videos), explains the basics of steganoanalysis, and introduces the reader to the basics of cryptography. This is followed by the description of implementation of specific steganographic cyphers.

Klíčová slova: steganografické šifry; steganografie; steganoanalýza; kryptografie; tajná informace; utajená komunikace; metoda nejméně významného bitu; imitace dat

Keywords: steganography cyphers; steganography; steganalysis; cryptography; secret information; concealed comunication; least significant bit method; data mimicry

Děkuji panu docentovi Miroslavovi Kolaříkovi za vedení diplomové práce a nespočet cenných rad. Dále pak své rodině a přátelům za podporu.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

1 Úvod **9**

2 Historie steganografie **11**

3 Kryptografie **13**
3.1 Entropie . . . 13
3.2 Šifry založené na klíči . . . 15
3.2.1 Symetrické šifry . . . 15
3.2.1.1 DES . . . 16
3.2.1.2 AES . . . 16
3.2.2 Asymetrické šifry . . . 16
3.2.2.1 Základy modulární aritmetiky . . . 17
3.2.2.2 RSA . . . 18
3.3 Důkaz bez vyzrazení informace (Zero-knowledge proof) . . . 19

4 Steganografie **21**
4.1 Použití digitální steganografie . . . 21
4.1.1 Metody digitální steganografie . . . 21
4.1.2 Způsoby využití steganografie . . . 23
4.1.2.1 Vodoznak . . . 23
4.1.2.2 Utajená komunikace . . . 24
4.1.2.3 Rozšíření datových struktur . . . 25
4.2 Samoopravné kódy . . . 25
4.2.1 Samoopravné kódy ve steganografii . . . 25
4.2.2 Samoopravné kódy a způsob fungování . . . 26
4.3 Periodické kódy . . . 28
4.4 Rozdělení tajné informace . . . 29
4.4.1 Potřeba všech n částí . . . 29
4.4.2 Potřeba jen k částí z n částí . . . 30
4.4.3 Reálný model principu rozdělení informace . . . 31
4.4.4 Různé váhy podklíčů . . . 32
4.4.5 Rozdělení tajné informace v asymetrické kryptografii . . . 32
4.4.6 Steganografické souborové systémy . . . 32
4.5 Komprese dat . . . 33
4.5.1 Komprese ve vztahu ke steganografii . . . 34
4.5.2 Huffmanovo kódování . . . 34
4.5.3 GZSteg . . . 36
4.6 Imitace dat (Data mimicry) . . . 37
4.6.1 Ukrytí informace v imitovaných datech . . . 38
4.6.2 Problémy ukrývání tajné informace v imitovaných datech . 39
4.6.3 Imitace dat v obrázcích . . . 40
4.6.4 Imitace dat za pomoci formálních gramatik . . . 40

4.6.4.1 Kódování tajné informace . . . 41
4.6.4.2 Parsování imitovaných dat a získání tajné informace 42
4.6.4.3 „Zkreslení“ formální gramatiky . . . 42
4.6.4.4 Bezpečnost a efektivita . . . 43
4.6.5 Použití konceptu zpětného výpočtu a Turingova stroje . . 44
4.6.5.1 Zpětný výpočet . . . 44
4.6.5.2 Reverzibilní Turingův stroj . . . 45
4.7 Užití ruchu v digitálním souboru . . . 46
4.7.1 Metoda nejméně významného bitu (Least Significant Bit; LSB) . . . 47
4.7.2 Úskalí při užívání ruchu k ukrytí tajné informace . . . 48
4.7.3 Souborový formát GIF . . . 49
4.7.3.1 EzStego . . . 52
4.7.4 Rozdělení a rozprostření tajné informace . . . 53
4.7.5 Více kanálů v jednom souboru . . . 53
4.7.6 Souborový formát JPEG . . . 54
4.7.6.1 Využití formátu JPEG k efektivnější steganografii 54
4.7.6.2 Metoda JSteg . . . 55
4.7.6.3 Metoda F4 . . . 55
4.8 Klíče . . . 56
4.9 Změna pořadí . . . 57

5 Steganoanalýza **59**
5.1 Obtížnost odhalení steganografických šifer . . . 59
5.2 Typy útoků na steganografii . . . 59
5.3 Přístupy k odhalení steganografie . . . 61
5.3.1 Vizuální (poslechová) analýza . . . 61
5.3.2 Strukturální analýza . . . 62
5.3.3 Statistická analýza . . . 62

6 Programátorská dokumentace **64**
6.1 Microsoft Ribbon . . . 64
6.2 Metoda LSB Swap . . . 64
6.3 Metoda LSB Vigenère . . . 65
6.4 Metoda Comments cypher . . . 67

7 Uživatelská dokumentace **69**
7.1 Spuštění aplikace . . . 69
7.2 Práce s aplikací . . . 69
7.2.1 Metoda LSB Swap . . . 69
7.2.1.1 Šifrování . . . 69
7.2.1.2 Dešifrování . . . 71
7.2.2 Metoda LSB Vigenère . . . 72
7.2.2.1 Šifrování . . . 72
7.2.2.2 Dešifrování . . . 73

7.2.3 Metoda Comments cypher 73
7.2.3.1 Šifrování 73
7.2.3.2 Dešifrování 74

Závěr **75**

Conclusions **76**

A Obsah přiloženého DVD **77**

Reference **78**

Seznam obrázků

1 Ukázka vodoznaku 24
2 Jeden bit navíc 27
3 Dva bity navíc 28
4 Potřeba dvou přímek k získání tajemství 30
5 Huffmanův strom 39
6 Fredkinovo hradlo 44
7 Fredkinovo hradlo implementující AND 45
8 Metoda nejméně významného bitu 48
9 Obrázek s oblastí nekvalitního šumu (řeka) 50
10 Maska nejméně významných bitů (řeka) 51
11 Hlavní okno aplikace 70
12 Menu aplikace 70
13 Šifrování metody LSB Swap 72

Seznam tabulek

1 Část meta-tabulky formátu GIF 52

1 Úvod

Snaha utajené komunikace provází lidstvo už od pradávna. Věda, která se zabývá utajením přenášené informace při komunikaci se nazývá kryptografie. Tato věda určitým způsobem (záleží na druhu šifry) transformuje zprávu na její „nečitelnou“ formu. Provede se tak většinou za pomoci takzvaného klíče. Dále existuje také podobor kryptografie zvaný steganografie, jejíž princip je založen na utajení procesu komunikace samotné, kde v tomto případě je klíč vlastně samotný princip utajení.[1] Nevýhodou kryptografie oproti steganografii je snadné zpozorování toho, že zpráva skrývá nějakou informaci. Naproti tomu steganografie se snaží tento fakt ukrýt společně s nesenou informací. Jinými slovy se snaží skrýt to, že jde vůbec o nějakou zprávu. Šifra, jejíž bezpečnost je založena na neznalosti principu utajení, je označována termínem *security through obscurity* [2], a právě mnoho steganografických šifer má tento charakter. Naopak kryptografické šifry jsou většinou založeny na principu sdíleného klíče (*shared key*), díky kterému jsou šifru schopni rozšifrovat pouze majitelé klíče, a takové kryptosystémy jsou tedy většinou bezpečné, i když je znám princip jejich fungování.[2] V obou případech vyzrazení, či nalezení klíče, znamená možnost dešifrování a získání tajné informace.

Velmi hezkým příkladem steganografie je karetní hra Kent (Kemps). K dosažení výhry je nutné nasbírat čtyři karty stejného druhu (například čtyři krále) a následně dát svému spoluhráči tajné znamení (poškrábáním na tváři nebo specifickým držením karet), který poté zvolá slovo „Kent!“ a dvojice vítězí. Je nutné, aby ono tajné znamení nebylo odhaleno protihráči, kteří mohou své soupeře zastavit zvoláním „Stop Kent!“ a tím také vyhrát. Právě takové tajné znamení zde funguje jako steganografická šifra, pomocí níž spoluhráči tajně komunikují [13].

Diplomová práce se věnuje hlavně digitální steganografii, jejíž myšlenka je taková, že se do určitého objektu (obrázku, textové zprávy) zakóduje nějaká informace. Takový objekt je poté zaslán příjemci přes nezabezpečený kanál. Nezabezpečeným kanálem rozumíme to, že kdokoli může tento objekt zachytit. Nicméně jen ti, co ví, že daný objekt obsahuje skrytou informaci, mohou být schopni tuto informaci z objektu extrahovat, znají-li princip ukrytí informace. Samozřejmě tak jako kryptografické šifry, mohou ty steganografické využívat tajného klíče, který je nezbytný pro rozšifrování, i když je znám princip utajení. Takové šifry už jsou mnohdy právě kombinací se šiframi kryptografickými.

V první kapitole je stručně shrnuta historie a zajímavosti z oblasti steganografie. Druhá kapitola pojednává o vědě, která je velmi příbuzná steganografii – kryptografii. Kryptografie úzce souvisí s pojmem entropie, která je také součástí této kapitoly a dále jsou zde popsáni hlavní zástupci dvou nejvýznamnějších skupin kryptografie. Z kategorie symetrických šifer jsou to šifry DES a AES.

[1] Navíc silnější steganografické šifry mohou využívat klíč ve stejné smyslu jako je tomu u těch kryptografických.

[2] Slabé šifry mohou být prolomeny i bez znalosti klíče hrubou silou, lexikální analýzou (posuvná šifra, substituční šifra a další).

Z kategorie asymetrických je to šifra RSA. Následuje kapitola detailně popisující digitální steganografii, její metody a způsoby použití. Kapitola pokračuje tématy věnujícím se samoopravným kódům, kompresi dat, imitaci dat a užití šumu digitálního souboru, které úzce souvisí se steganografií. Dále práce pokračuje stručným úvodem do problematiky steganoanalýzi, která se zabývá metodami pro odhalení steganografické šifry, a tím i získání skryté informace. Na samotný závěr práce obsahuje programátorskou a uživatelskou dokumentaci k naprogramované aplikaci implementující steganografické šifry.

2 Historie steganografie

Historie steganografie sahá do minulosti mnohem dále, než vůbec přišla digitální steganografie, kterou se věnuje zbytek diplomové práce. Bylo použito mnoho zajímavých a velmi důmyslných technik, které se ale dají nazývat spíše jen jako triky bez jakéhokoli formálního základu. V drtivé většině byla steganografie používaná špióny v rámci různých válek. Kapitola čerpá z [1, 2, 10].

První zmínky o použití myšlenky steganografie pochází z pátého století před naším letopočtem, kdy řecký historik Herodotus popsal ve svém díle Histories událost ze stejné doby. Vůdce Miletusu Histiaeus, tehdejšího řeckého městského státu, oholil hlavu svému nejdůvěryhodnějšímu otrokovi, vytetoval mu na lebku zprávu, která měla spojence informovat o započetí revolty proti perským okupovatelům. Následně pak počkal, až otrokovi vlasy dorostou, a nechal ho poslat za svými spojenci se skrytou zprávou. Podobný trik provedl Demeratus, který informoval o tažení perských vojsk na Spartu, přičemž použil dřevěnou destičku, do které vyryl zprávu, kterou následně přelil voskem.

Pojem steganografie byl poprvé použit v díle německého opata Johannese Trithemiuse v 15. století v dílech *Polygraphia* a *Steganographia*. Tento opat byl považován za teoretika ohledně magie a čarodějnictví, a jeho díla byla církví zakázána. Jedna z knih byla poté učencům podezřelá z ukrývání tajné zprávy a tajemství bylo po nějaké době odhaleno.[3]

V historii také mnoho umělců používalo steganografii pro označení svých děl jako vlastních a nemožnosti zpochybnění tohoto faktu.[4] Například známý umělec Giovanni Boccaccio zakódoval tři sonety z jeho díla *Amarosa visione* do počátečních písmen jeho ostatních básní. Taková steganografie, která je prováděna nad textem, se nazývá lingvistická steganografie.

Dalším příkladem může být kombinace normálního fontu a kurzívy pro text, kde tajná zpráva je zakódována právě v použití variací písmen z daného fontu. Tato technika byla použita v šestnáctém století, kdy byla velká nekonzistence v typografii, a tak tato metoda nevzbuzovala tolik pozornosti. S textem související steganografická metoda je také technika, kdy řádky textu byly posouvány o určitou vzdálenost, která nebyla pouhým okem znatelná. Při bližší analýze se dal poznat rozdíl a tím i odhalit tajnou zprávu. Řádky se posouvaly o $\frac{1}{300}$ palce, což činí přibližně $0,1$ milimetru.

Jsou známy případy, kdy se pro utajení zprávy používal neviditelný inkoust složený z různých organických kapalin, například z citronové šťávy, který byl viditelný pouze po ohřátí, či pod UV zářením. Jako alternativy pro inkoust se dále používaly ještě cukerný roztok, mléko, ocet, zředěný med nebo dokonce i moč.

Mnoho dalších technik bylo použito za dob světových válek. Taková komuni-

[3] Jedna ze skrytých zpráv obsahovala text: „The quick brown fox jumps over the lazy dog.“ Tato věta je zajímavá tím, že obsahuje všechna písmena anglické abecedy právě v jednom výskytu.

[4] Tato technika se nazývá vodoznak, viz sekce 4.1.2.1.

kace vypadala například tak, že se skrytá zpráva kódovala pomocích miniaturních teček na papír, které se následně přetřely speciální látkou[5], která vytvořila lesklý povrch, a snížila tím pravděpodobnost zpozorování mikroteček. Papír se samozřejmě neposílal takhle „prázdný“, ale s nějakou klasicky napsanou zprávou neobsahující citlivé informace. Další důmyslné médium, které ukrývalo skryté zprávy, sloužilo jako fotosenzitivní sklo, které lokálně mění barvu po ozáření UV paprsky. Takové sklo se mohlo distribuovat jako například sklenička s abstraktním zdobením.

Kuriozitou se stal šachový zápas mezi Viktorem Korchnoiem a Anatoly Karpovem v roce 1978, kdy Karpovovi přinesl jeho asistent kelímek jogurtu. Jelikož bylo období studené války a šachové turnaje, jakož to i jiné sporty, byly velice kompetitivní a vyhrocené, soupeřův tým protestoval proti tomuto jednání. A právě hlavním důvodem bylo to, že mohl tým Karpovovi podávat tajné informace ve formě barvy jogurtu, nebo například na základě určitého času doručení jogurtu, kde taková tajná zpráva mohla obsahovat určitá doporučení, například jestli navrhnout remízu nebo remízu naopak odmítat. Organizace turnaje protestům vyhověla a přistoupila ke kompromisu, kdy Karpov mohl dostat jen určitý typ jogurtu (růžový) a v určitém časovém okamžiku.

Zdaleka ne poslední, a zároveň velmi netradiční metodou steganografie, bylo počínání amerického vojáka Jeremiaha Dentona, který byl zajatcem v severním Vietnamu. Ten v televizním přenosu poslal tajnou zprávu mrkáním v Morseově abecedě, která informovala o mučení amerických zajatců ve Vietnamu. [10]

[5]Používalo se kolodium, které se používá v lékařství.

3 Kryptografie

Dříve než budou podrobněji rozebrány pojmy související se steganografií, je nutné uvést základy kryptografie, které se steganografií úzce souvisí. V této kapitole bylo čerpáno z [1, 3 – 5, 11].

Kryptografie je vědou o utajování obsahu zpráv takovým způsobem, že zpráva je transformována na jiný „náhodný“ text (tzv. kryptogram), který je schopný převést zpět na obsah původní zprávy pouze zamýšlený příjemce. Proces transformace se nazývá šifrování, či dešifrování podle toho, o který směr transformace se jedná. Kromě příjemce by neměl být schopen zprávu nikdo rozluštit.[6] Z původní zprávy je určitým způsobem pomocí klíče vytvořen kryptogram, který je poslán příjemci přes nezabezpečený kanál. Příjemce pomocí svého klíče z šifry vytvoří zpět originální zprávu.

K šifrování se často používají klíče, ale to neplatí u všech šifer. Existují také tzv. omezené šifry, které, stejně jako mnoho steganografických šifer, zakládají svou bezpečnost na neznalosti systému šifry. Takové šifry se téměř nepoužívají, protože mají mnoho nevýhod.

- Když konkrétní omezenou šifru používá skupina lidí, není možno šifru zachovat při odchodu byť jediného člena – musí se vymyslet nová.
- Při vyzrazení fungování šifry veřejnosti ji není možné nadále používat, protože kdokoli je schopen zprávy dešifrovat.
- Každá skupina lidí si musí vymyslet svoji unikátní šifru.
- Skupina si nemůže být bez odborníka jista, že používají kvalitní šifru.

Všechny tyto problémy odpadají používáním šifer založených na klíči.

3.1 Entropie

U zašifrované zprávy pomocí kryptografické metody je snaha o to, aby výsledný kryptogram měl tu vlastnost, že výskyty jednotlivých znaků mezi sebou vůbec nesouvisí. Když by poté nějaký statistický algoritmus procházel kryptogram, který by takovou vlastnost neměl, tak by byl schopen na základě nějakých častěji vyskytujících se vzorů odhadovat nezašifrovanou zprávu. Právě v lidském jazyce se mnoho takových vzorů vyskytuje. Například v anglickém jazyce je nejčastěji se vyskytující trigram slovo *the*, nebo také po písmenu *q* se v drtivé většině případů vyskytuje písmeno *u*. Proto, v ideálním případě, by měl mít kryptogram výskyt každého znaku se stejnou pravděpodobností.

Čím hůře se dají předpovídat znaky v kryptogramu, tím vyšší míru neurčitosti daný kryptogram obsahuje. Taková míra neurčitosti se označuje jako entropie. Pojem entropie v informatice byl popsán a definován Cloudem Shannonem

[6] Pokud jde o tzv. symetrické šifrování, tak zprávu je schopný rozluštit zpětně i odesilatel (ten, kdo zprávu transformoval).

v 70. letech 19. století. Čím je kryptogram delší a čím obsahuje méně informace (je snadné předvídat znaky), tím je větší možnost jeho bezeztrátové komprese. Entropie se vypočítá pomocí vzorce:

$$E(y) = -\sum_{z \in y} p(z) \cdot log_B p(z),$$

kde

- *E(y)* – entropie kryptogramu y
- *p(z)* – pravděpodobnost výskytu znaku z v kryptogramu y
- B – základ pro logaritmus

Když je základ logaritmu roven roven dvěma, tak je jednotkou entropie bit. Tato hodnota logaritmu se bude nadále uvažovat.

Následuje příklad, kdy entropie textového řetězce je velmi nízká (mezery jsou pro jednoduchost při výpočtu entropie vynechány):

```
barbara a barbora u baru
```

V řetězci délky 20 se vyskytují znaky s následující četností:

znak	a	b	o	r	u
četnost	7	5	1	5	2

Například pro znak *a* bude část výpočtu vypadat:

$$\frac{7}{20} \cdot log_2(\frac{7}{20}) \cong -0,5301$$

Při dopočítání ostatních členů sumy je celková entropie:

$$E(\text{"}barbaraabarboraubaru\text{"}) \cong 2,0784.$$

U stejně dlouhého řetězce, který by se skládal ze vzájemně různých znaků, by entropie vyšla přibližně 4,322.

Vypočítaná hodnota entropie ještě není úplně přesná, protože je potřeba navíc uvažovat výskyty znaku za podmínky, že mu předchází určitý znak, dále výskyty znaku za podmínky, že mu předchází určitá uspořádaná dvojice znaků a tak dále. U potenciálně nekonečného proudu znaků je tak vlastně entropie nevypočítatelná za předpokladu, že neznáme přesné pravděpodobnosti výskytů znaků.

3.2 Šifry založené na klíči

Šifry založené na klíči se od omezených šifer liší tak, že znalost jejich fungování nenarušuje jejich bezpečnost (až na jednoduché šifry[7]). Jejich bezpečnost závisí na znalosti klíče, pomocí kterého je možné zprávu šifrovat a dešifrovat. Zde se dále šifry dělí na symetrické, kde odesilatel a příjemce sdílí stejný klíč a asymetrické, kde se pro šifrování používá šifrovací klíč a pro dešifrování dešifrovací klíč. Dále ještě existují šifry hybridní, které kombinují oba přístupy.

Šifra je formálně označována jako kryptosystém, který se skládá z množiny zpráv, které se pomocí šifrovací funkce a klíčů zobrazí na kryptogramy. Navíc existuje ještě dešifrovací funkce, která kryptogram zobrazí zpět na originální zprávu za použití stejného klíče. To znamená, že pro dvojici zobrazení musí platit následující vztah:

$$d(e(x, k_e), k_d) = x,$$

kde je:

- $e(x, k_e)$ ... šifrovací funkce
- $d(y, k_d)$ ... dešifrovací funkce
- k_e ... šifrovací klíč
- k_d ... dešifrovací klíč
- x ... zpráva.

3.2.1 Symetrické šifry

V tomto druhu šifer jsou šifrovací a dešifrovací funkce stejné, nebo mezi nimi existuje velmi jednoduchý vztah. Mezi tyto šifry patří například posuvná šifra, afinní šifra, substituční šifra, Vigenèrova šifra, proudové šifry, DES, AES. Symetrické šifrování je velmi rychlé, ale oproti asymetrickému šifrování má tu nevýhodu, že je nutno nějakým způsobem bezpečně šifrovací funkci distribuovat k příjemci či odesilateli.

Pro ukázku velmi podobného vztahu mezi šifrovací a dešifrovací funkcí jsou uvedeny funkce pro posuvnou šifru:

$$e(x, k) = x + k,$$

$$d(y, k) = y - k.$$

[7] Dokonce při dostatečně dlouhém klíči je i ta nejjednodušší šifra naprosto neprolomitelná.

3.2.1.1 DES

Pozornost si zaslouží šifrování DES, které je oproti posuvné šifře mnohem komplikovanější. Na rozdíl od ní využívá dva důležité kryptografické principy, a to konfúzi a difúzi. Posuvná šifra používá pouze konfúzi. Konfúze je proces, při kterém je snaha vstup nějakým způsobem pozměnit a to tak, aby nebyla zjevná spojitost mezi původní zprávou a použitým klíčem. Kdežto difúze má za úkol to, aby vstup co možná nejvíce ovlivňoval zašifrovanou zprávu ve smyslu, že sebemenší změna ve vstupním textu velmi ovlivní zašifrovanou zprávu, ideálně v průměru její polovinu.

Při procesu šifrování DES se pracuje s 64-bitovými bloky zprávy, které se šifrují pomocí 56-bitového klíče. Proces se skládá ze šestnácti iterací konfúze a difúze. Nejdříve se 64-bitová zpráva rozdělí na dvě poloviny, kdy nad levou polovinou se provede operace konfúze pomocí takzvaných S-boxů[8] a klíče. Následně se výsledek ještě zkombinuje s pravou stranou původní zprávy (difúze) a výstup je novou levou polovinou pro další iteraci. Nová pravá polovina je pouze kopie levé poloviny původní zprávy. Tento proces se provádí v šestnácti iteracích, které se nazývají rundy.

Z bezpečnostních důvodu byla délka klíče (56 bitů) shledána jako nedostatečná, a proto se pro větší bezpečnost používá princip 3DES. Ten funguje zřetězením tří DES šifer za sebou, kde pro každé šifrování je použit jiný klíč ($3 \cdot 56 = 168$ bitů). Při této metodě je možno použít jen dva klíče, kde první se používá při první a třetí fázi šifrování (dešifrování).

3.2.1.2 AES

Za zmínku také stojí šifra AES, která ze šifry DES přímo vychází, ale na rozdíl od ní využívá polynomiální aritmetiky nad prvočíselnými tělesy. Tato šifra je implementována pomocí šifry Rijndael, která byla vytvořena Belgickými kryptology.

Délka klíče je na rozdíl od DES výrazně zvětšena, a to dokonce ještě volitelná, kde možnosti délky klíče jsou 128, 192 nebo 256 bitů. Tato šifra také šifruje iterativně v rundách a konkrétně pro 128-bitový klíč se provádí 10 rund, pro 192-bitový klíč 12 rund a nakonec pro klíč velikosti 256 bitů 14 rund.

Použití šifry AES je do dnešní doby stále velmi populární – používá se například v síťovém šifrovacím protokolu SSH, pro šifrovanou komunikaci bezdrátového připojení WiFi (WPA2) a také v oblíbené komunikační aplikaci Skype.

3.2.2 Asymetrické šifry

Na rozdíl od symetrických šifer používají asymetrické šifry pro šifrování a dešifrování různé funkce. Tento způsob šifrování je sice pomalejší než symetrické, avšak řeší problém distribuce klíče pomocí tzv. Diffie-Hellmanova principu. Největším zástupcem této třídy je bez pochyby šifra RSA.

[8] S-boxy nejsou nic jiného, než tabulka reprezentující substituci.

Princip komunikace funguje tak, že je vygenerována dvojice klíčů, kde se takzvaný veřejný klíč zveřejní a druhý, privátní, se uchová v tajnosti. Když někdo bude chtít poslat zprávu osobě vlastnící privátní klíč, svou zprávu zašifruje pomocí veřejného klíče, takto zašifrovanou zprávu zašle a příjemce si zašifrovanou zprávu dešifruje svým privátním klíčem.

3.2.2.1 Základy modulární aritmetiky

Důležitým pojmem pro pochopení principu fungování šifry RSA je poje kongruence podle modulu. O dvou celých číslech a a b se řekne, že jsou kongruentní podle modulu n, jestliže a a b dávají stejný zbytek po dělení číslem n značeno následovně:

$$a \equiv b \pmod{n}.$$

Například platí $12 \equiv 17 \pmod{5}$, protože $12 = 2 \cdot 5 + 2$ a $17 = 3 \cdot 5 + 2$.

Čísla, která jsou vzájemně kongruentní podle daného modulu, patří do tzv. zbytkové třídy podle modulu. Relace $\equiv$ je ekvivalence a její třídy jsou právě všechny možné zbytkové třídy podle daného modulu. Těchto tříd je právě tolik, kolik je hodnota modulu, a i tolik je samozřejmě různých zbytků po celočíselném dělení tímto modulem. Proto se pro každou z těchto tříd zvolí jeden zástupce, což budou právě všechny tyto různé zbytky po dělení modulem. Zbytková třída se označuje $[a]_n$, kde a je libovolný prvek z této třídy, ale nejčastěji je to právě jeden ze zbytků po dělení modulem. Množina všech zbytkových tříd se značí:

$$\mathbb{Z}_n = \{[0]_n, [1]_n, [2]_n, \ldots, [n-1]_n\},$$

a zkráceně (a jestli nedochází ke konfliktu značení s celými čísly) značení vypadá:

$$\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}.$$

Poslední vsuvkou v této kapitole je definice operací sčítání $(+)$ a násobení $(\cdot)$ na této množině zbytkových tříd. Tyto operace jsou definovány takto:

$$[a]_n + [b]_n = [a+b]_n,$$

$$[a]_n \cdot [b]_n = [a \cdot b]_n.$$

Kupříkladu platí:

$$(3+2) \pmod{3} = 2 \pmod{3},$$

nebo

$$(3 \cdot 2) \pmod{3} = 0 \pmod{3}.$$

3.2.2.2 RSA

Mezi asymetrické šifry patří například RSA, která je v praxi hojně používaná. Funguje tím způsobem, že se vygenerují vysoká prvočísla p a q (velikost je 1024 bitů a více). Tato čísla se musí uchovat v tajnosti. Dále se vypočítá číslo $p \cdot q = n$, které již není utajováno (je součást veřejného klíče). Číslo n slouží jako modul při šifrovací a dešifrovací funkci. Hlavní bezpečnostní prvek algoritmu spočívá v tom, že je velmi těžké rozložit (faktorizovat) číslo n na součin prvočísel p a q.

Hlavní a důležitou charakteristikou, která pro číslo n platí, je, že existuje číslo φ takové, že když libovolné číslo $x < n$ umocníme číslem φ a tuto operaci provedeme v modulu n, tak získáme opět číslo x.

$$x^{\varphi} \pmod{n} = x \tag{1}$$

Když se takové číslo nalezne, je možné ho rozložit na součin dvou čísel $a \cdot b = \varphi$.[9] Jedno z těchto čísel, například a, slouží jako veřejný exponent (druhá součást veřejného klíče) a druhé, b, slouží jako soukromý exponent (tajný klíč).

Poté šifrovací a dešifrovací funkce vypadají následovně:

$$e(x, a) = x^a \pmod{n} = y, \tag{2}$$

$$d(y, b) = y^b \pmod{n} = x. \tag{3}$$

Následuje otázka, jak rozložit φ na součin dvou čísel, jestli tohle vůbec lze a jak takové φ najít. Existuje takzvaná Eulerova funkce $\varphi(n)$, která bere jako jediný argument přirozené číslo a vrátí počet přirozených čísel, menší než n, která jsou s n nesoudělná. Triviálně pro prvočíslo s platí: $\varphi(s) = s - 1$. Opět hlavním prvkem bezpečnosti algoritmu, podobně jako to bylo s nesnadnou faktorizací n, je nesnadná faktorizace φ za předpokladu, že neznáme čísla p a q. Když jsou tato čísla známá, tak se $\varphi(n)$ vypočítá jako $\varphi(n) = (p - 1) \cdot (q - 1)$.

Poté se veřejný a soukromý exponent vypočítá tak, že jeden je zvolen náhodně, ale musí být nesoudělný s $\varphi(n)$. Druhý exponent se dopočítá jako inverze prvního exponentu, tentokrát ale v modulu $\varphi(n)$:

$$b = a^{-1} \pmod{\varphi(n)}. \tag{4}$$

Tato inverze se efektivně počítá pomocí takzvaného rozšířeného Euklidova algoritmu.

Ještě nutno poznamenat pár technických detailů ke konkrétní implementaci RSA šifrování. První z nich je, že se textová zpráva rozdělí do číselných bloků, kde každé číslo musí být menší nebo rovno modulu a šifrování se provádí právě nad těmito čísly.

Jelikož se v šifrovací a dešifrovací funkci vyskytují operace umocňování s vysokým exponentem, je nutné tuto operaci nějakým způsobem zefektivnit. To se provádí pomocí takzvaného rychlého umocňování, pomocí vhodných variací

[9] Číslo φ takto opravdu rozložit jde (není to prvočíslo) – bude popsáno dále.

operací druhé mocniny a násobením základem čísla. Detaily této metody jsou popsány v [4].

Poslední důležitá poznámka se týká zvolení vysokých náhodných prvočísel p a q, což není zrovna jednoduchý úkol. Provádí se tak pomocí různých algoritmů, nejčastěji takzvaných Monte Carlo. Tyto algoritmy nevrací zaručeně správný výsledek, ale pracují rychle a při jejich opakovaném spouštění a ověřování nad daným číslem, můžeme s určitou pravděpodobností určit, že jde o prvočíslo. Příkladem algoritmů tohoto typu testující prvočíselnost jsou Fermatův test nebo test Miller-Rabinův. Ovšem existují i algoritmy, které jsou schopny deterministicky v polynomickém čase otestovat číslo na prvočíselnost s absolutní jistotou. Hlavním zástupcem této třídy testů je AKS [8]. Nicméně v praxi se i přes to používají algoritmy Monte Carlo vzhledem k jejich vyšší rychlosti.[10]

Všechny zmíněné technické detaily včetně Euklidova algoritmu jsou detailněji popsány v [3, 4].

3.3 Důkaz bez vyzrazení informace (Zero-knowledge proof)

Jako poslední metoda spjata s kryptografií je uvedena metoda *Zero-knowledge proof*, která slouží k dokazování informace, jejíž obsah není nutno odhalovat.

Nejjednodušším příkladem takového důkazu bez vyzrazení informace je případ, kdy dokazující má dva identické předměty, které se liší pouze v barvě a má za úkol dokázat barvoslepému, že jsou tyto dva předměty různé. Proces dokazování vypadá tak, že si barvoslepý vezme oba předměty do rukou, schová je za záda a na základě svého uvážení si je tajně před dokazujícím prohodí nebo je nechá v původním pořadí. Následně předměty odhalí a za předpokladu, že dokazující barvy vidí, je schopný rozhodnout, jestli byly předměty za zády prohozeny nebo nikoli. Ovšem za předpokladu, že by dokazující barvy nebyl ve skutečnosti schopen rozeznat, tak má zde 50% pravděpodobnost, že uhodne. To se dále řeší opakovaným dokazováním tak dlouho, dokud barvoslepý uzná, že dokazující netipuje, ale informaci opravdu ví [11].

Důležitými vlastnostmi této metody jsou:

- Úplnost – za předpokladu, že dokazující zná tajemství, a je tím pádem schopen vždy správně odpovědět (např. ohledně prohození předmětů), tak ověřující bude vždy přesvědčen o tom, že dokazující zná tajemství
- Korektnost – za předpokladu, že dokazující tajemství nezná, přesvědčí ověřujícího jen s velmi malou pravděpodobností (o jak malou pravděpodobnost se jedná záleží na počtu iterací testu)
- Nevyzrazení tajemství (Zero knowledge) – při dokazování dokazující nevyzradí ověřujícímu vůbec nic až na správné tvrzení při testech. Kdyby byl

[10] Jistota sice není 100%, ale tyto testy velmi rychle dosáhnou velice pravděpodobného výsledku oproti délce řešení algoritmu AKS.

ověřující následně v pozici dokazujícího, tak by nebyl schopný tímto testem projít.

Podobný důkaz se může provést například s balíčkem karet, kde je přesně 26 karet černých a 26 karet červených. Za předpokladu, že dokazující chce dokázat, že náhodně vytažená karta je červená, stačí mu ukázat právě všech 26 černých karet, čímž dokazuje, že náhodně vytažená karta je opravdu červená bez toho, aniž by prozradil její číslo. Tento důkaz je již věrohodný bez nutného opakování.

Tato metoda dokazování se používá například v autentizačních systémech. Za použití metody *Zero-knowledge proof* může klient dokázat svoji identitu serveru bez toho, aniž by musel vyzrazovat jakékoli citlivé informace (například heslo).

4 Steganografie

Jak již bylo v úvodu řečeno, steganografie je věda, která se zabývá utajením komunikace mezi dvěma stranami, pro níž se využívala v historii široká škála důmyslných technik, a která se v dnešní době spíše omezuje na techniky modifikující digitální soubory (textové, obrázkové, zvukové, či video). V kapitole jsou použity informace z [1, 2, 5, 9].

4.1 Použití digitální steganografie

S příchodem počítačů, které používají digitální soubory, se také ihned začala rozšiřovat steganografie i do těchto sfér. Z hlediska toho, jak mohou být digitální soubory složité a v drtivé většině pro člověka nečitelné, poskytují steganografickým šifrám ideální útočiště. Některé steganografické algoritmy jsou velmi úzce spjaty s těmi kryptografickým. Také se dají oba přístupy zkombinovat do jedné šifry. Nejčastěji se kombinují tak, že se nejdřív tajná zpráva zašifruje na kryptogram a poté se provede steganografická část šifrování. Následné dešifrování takové složené šifry se provádí samozřejmé v opačném pořadí.

Digitální steganografie může sice napomoci kriminálníkům ukrývat svou komunikaci, ale na druhou stranu může být použita i k dobrým účelům, například:

- anonymní poradenství ohledně citlivých záležitostí (kupříkladu sebevražda)
- hledání nové práce bez obav že, aktuální zaměstnavatel něco zjistí, nebo byť jen nabude podezření, a případně za to daného zaměstnance „vyrazí“
- anonymní podávání informací ohledně bezpráví a nezákonného chování bez rizika zjištění
- anonymní vyjádření názoru (například politického), přičemž nedojde k opovržení druhou stranou
- ochrana osobních údajů při jejich přenosu
- utajená komunikace policejních složek ohledně tajného zásahu proti nelegální činnosti.

Konkrétně tajná komunikace mezi politiky by byla velmi užitečná, protože se často přou o tom, zda-li měl nebo neměl mít přístup ten daný politik k určitým informacím.

4.1.1 Metody digitální steganografie

Existuje mnoho různých způsobů, jak zakomponovat skrytou informaci do digitálního souboru:

- Užití šumu v digitálním souboru – každý obrázek (nahrávka) obsahuje hodnoty reprezentující, jakou má mít prvek (jedna z barevných složek pixelu, hlasitost v daném čase) intenzitu barvy nebo zvuku. Je možnost tyto hodnoty lehce pozměnit, čímž se sice sníží kvalita obrázku/zvuku, ale při přiměřené změně je rozdíl nezpozorovatelný pouhým okem/uchem. Tyto změny hodnot mohou nést právě ukrytou zprávu. Konkrétní implementací této techniky je například Metoda nejméně významného bitu (Least Significant Bit; LSB), která, jak už název napovídá, využívá k ukrytí informace právě nejméně významné bity, které se nejméně podílejí na výsledném obrázku/zvuku.

- Šíření informace ven (Spread the information out, často označovaná také termínem spread spectrum) – sofistikovanější technika, která šíří skrytou informaci ven skrze pixely nebo momenty ve zvukové nahrávce. Je obdobně těžké odhalit člověkem a také počítačem, který se dívá po statistickém profilu a snaží se tak zachytit nějaký vzor. Mnoho těchto technik se poprvé použilo při ukrývání informací v šumu radiokomunikace. Často je možné při použití této techniky nemít k dispozici úplně celou zprávu, nebo mít zprávu do určité míry poškozenou. Informaci je i tak možno znovu zrekonstruovat i z neúplných dat pomocí samoopravných kódů (error-correction codes).

- Využití statistického profilu dat – mnoho dat obsahuje v sobě určité vzory, kterých se dá využít. Například v anglickém jazyce se v drtivé většině případů po písmenu q vyskytuje písmeno u. Je tedy vhodné ukládat tajnou informaci využitím těchto vzorů, které jsou poté pro statistickou analýzu hůře odhalitelné.

- Nahrazení pseudo-náhodných dat – některá data mohou obsahovat část dat, která jsou vytvořena pseudo-náhodnými generátory. Například se s nimi můžeme setkat v počítačových hrách, kde vypadá mnohem reálněji, když jsou postavy vytvořeny z určité části náhodně (například tvar obličeje, umístění a tvar jizev, znamínek atd.). Tato pseudonáhodná data se dají přepsat a uložit v nich informace.

- Změna pořadí – u dat, která reprezentují seznam nějakých věcí (například seznam s nákupem) je mnohdy jedno, v jakém jsou pořadí. Ale právě toto pořadí samotné nese nějakou informaci, která může být libovolně zaměněna.

- Rozdělení informace – skrytá informace nemusí být obsažena pouze v jednom souboru, ale může být uložena napříč několika soubory, které se můžou po síti navíc směrovat různými cestami k příjemci. Také může být steganografický algoritmus vylepšen o to, že může stačit pouze libovolných k fragmentů z celkových n.

- Zamaskování zdroje odesílání – svým způsobem do steganografie může patřit i vlastnost, kdy odesilatel zprávy je anonymizován, což se může v některých případech také hodit.

Tyto techniky se dají samozřejmě dále kombinovat. Například nejdříve může být informace zakódována v pořadí seznamu, poté může být tento seznam skryt v ruchu souboru a nakonec může ještě dojít k anonymnímu vyslání zprávy.

4.1.2 Způsoby využití steganografie

Do teď bylo o steganografii pojednáváno tím způsobem, že může sloužit pouze k utajené komunikaci. To není tak úplně pravda, protože může sloužit i právě opačným způsobem a to ve formě vodoznaku (watermark). Dále, ale již spíše v minoritní míře, se dá použít na rozšíření datových struktur.

4.1.2.1 Vodoznak

Velmi důležitý pojem úzce související se steganografií je vodoznak (watermark). Ten slouží tak trošku k jiným účelům, než bylo doposud ohledně steganografie nastíněno. Digitální soubor je opatřen vodoznakem za účelem toho, že obsahuje různé informace například:

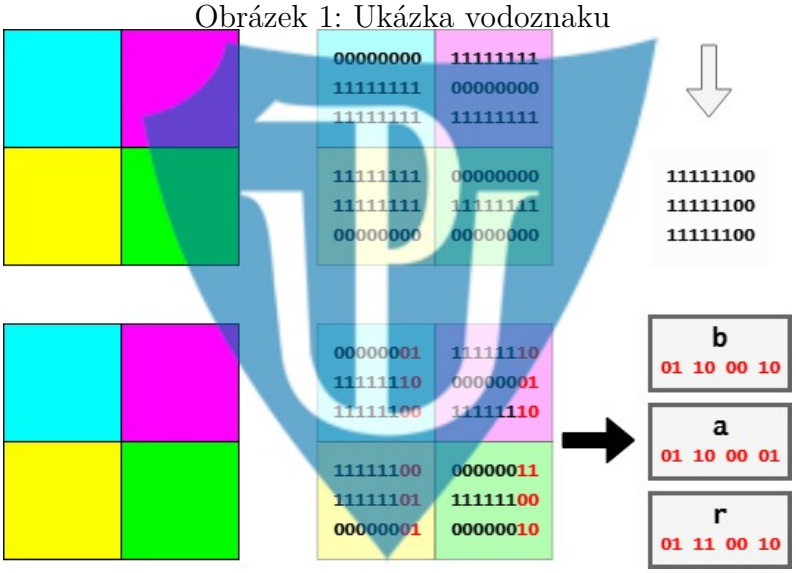
- kdo je autorem souboru
- kdo jej smí používat
- za jakým účelem lze používat
- jakým způsobem jej lze používat
- v jaké míře jej lze používat.

Video například nebude možno přehrát při neautentifikaci spotřebitele.

Hlavní vlastností vodoznaku je to, že ho není jednoduché v souboru identifikovat za účelem jeho potenciálního odstranění nebo změnění. Velmi často také vodoznak bývá na digitálním obrázku viditelný, aby se zabránilo jeho šíření a neautorizovanému používání. V hudební souboru může takový vodoznak vypadat tak, že opět jsou ve formě malého šumu ukryty tyto informace.

Problém, který se konkrétně vodoznaku týká, je ten, že vědci chtěli, aby nové poznatky ohledně steganografie byly veřejně dostupné. Toho se obávaly podnikatelské subjekty z důvodu toho, že by to mohlo být hrozbou pro jejich byznys. Argumentem bylo možnost napadení jejich systému – když se ví, na jakém principu funguje vodoznak, tak se může snadno „rozbít“. Proto je potřeba nějaká ochrana navíc. Přesně naopak je to u kryptografie – tam se všichni shodují, že znalost algoritmu podporuje celkovou bezpečnost. V historii veřejnost velmi napomohla odhalení mnoha nedokonalostí v kryptografických algoritmech a pomohla v jejich zdokonalení.

Obrázek 1: Ukázka vodoznaku



Kvalitní vodoznak by měl být schopen v souboru přetrvat i při určité míře změny souboru například při jeho otočení, zmenšení, či kompresi. Když už by měl být vodoznak v souboru nerozeznatelný, tak až v takovém momentě, kdy samotný soubor je nerozeznatelný od originálního, který prošel velkým zkreslením. Takový vodoznak je označován jako silný.

Na druhé straně existuje také slabý vodoznak, který naopak funguje tím způsobem, že když je se souborem sebemíň manipulováno ve smyslu jeho změny, tak vodoznak zmizí. Tím je tato změna jednoduše odhalitelná na první pohled. Toho může být využito například jako obrana před jiným pokusem o steganografii.

4.1.2.2 Utajená komunikace

Na rozdíl od vodoznaku, který se snaží být zpozorovatelný a co možná nejvíce odolný vůči útokům, které se ho snaží zničit, se dá steganografie použít naopak k ukrytí tajné zprávy, aby nebyla detekována nikým jiným než příjemcem.

Takové nejzajímavější použití by mohlo být například v politické sféře, kde je nutnost utajené komunikace mezi spojenci. Na druhou stranu nepřátelé vynakládají co největší úsilí o odhalení této komunikace a získání co nejvíce informací z ní. Pomocí steganografie je možnost výměnu informací mezi komunikujícími stranami držet v tajnosti. Takže ne jen, že útočník neví, o čem komunikují (o což se stará kryptografie), ale vůbec neví, že vlastně komunikují.

Nejjednodušší myšlenka, jakým způsobem ukrýt informaci do digitálního souboru (do obrázku nebo zvukové nahrávky), je modifikovat nejméně významné bity v bytech, které reprezentují pixely v obrázku nebo zvukové segmenty ve zvukové nahrávce. Dále se používají například tzv. mimické funkce, které pro nějaký soubor vytvoří podobný soubor. Tento soubor je podobný jak ve smyslu obsahovém (například text, který se člověku zdá stejný, nebo velmi podobný), tak ve smyslu statistických vlastností (například četnost výskytu jednotlivých

znaků v textovém souboru). Podrobněji budou tyto techniky popsány dále.

Konkrétně pro textové zprávy v editorech podporující formátování textu může steganografická šifra vypadat tak, že text a pozadí má stejnou barvu. V takových editorech je dále možnost formátovat i neviditelné znaky, jako například mezery nebo tabulátory. Ty také tedy nesou skrytou informaci na první pohled nepozorovatelnou. Kromě textových editorů může skrytou informaci obsahovat i HTML kód, který může mít na různých místech neviditelné znaky (třeba na konci řádků). Dále některé Unicode znaky mohou v některých systémech vypadat stejně jako znaky z ASCII, a právě tyto znaky mohou nést extra informaci, kterou objeví jen ten, kdo dokáže znaky Unicode a standardní ASCII rozeznat.

Nemusí jít jen o ukrývání tajné zprávy v souborech samotných. Skrytá zpráva může být zakódována třeba v délkách zpoždění posílání paketů v internetové síti.

4.1.2.3 Rozšíření datových struktur

Již mnohem méně časté, ale taktéž možné, použití může být u datových struktur, u kterých se dříve neplánovalo, či nevědělo, že by mohly obsahovat nějaké další vlastnosti. Ty je možno přidat právě pomocí steganografie. Původní software, pracující s datovou strukturou, ani vůbec nemusí zaregistrovat změnu. Například uložení informace o obrázku přímo v něm samotném. Informace se šíří s obrázkem a software pro práci s ním bude stále funkční.

Konkrétním příkladem by mohlo být například přikládání komentářů a poznámek do rentgenových snímků, což by mohlo ušetřit mnoho peněz na zakoupení nového vybavení.

4.2 Samoopravné kódy

K informačním technologiím neodmyslitelně patří takzvané samoopravné kódy. Ty jsou běžně potřeba pro to, když se zanese chyba do dat. Taková chyba může vzniknout například při posílání dat sítí, kdy přenosové médium není naprosto spolehlivé a může, ač velmi nepravděpodobně, převrátit hodnotu bitu. Například nějaký bit, který měl před vysíláním skrze síť hodnotu 1 má po přijetí hodnotu 0. Je nutno proto mít nějaký mechanismus, který tyto chyby odhalí, nebo je dokonce i opraví.

4.2.1 Samoopravné kódy ve steganografii

Takové samoopravné kódy se ale dají využít i tak trošku jiným způsobem, a až překvapivě podobným se steganografií. Například se do nějaké zvukové nahrávky úmyslně zanese nějaká drobná chyba, která ale bude opravitelná pouze specifickým softwarem. Zatímco takový software chybu opraví a nahrávku přehraje, jiný software zhavaruje na chybě. Tím vzniká omezení na přehrávání takové nahrávky, což je velmi podobné tomu, co má za úkol vodoznak.

Nebo také může být daná chyba opravitelná právě na jakémkoli zařízení. Poté taková chyba může nést skrytou informaci pro tajnou komunikaci.

Problém může nastat tehdy, když stejnou myšlenku dostane více lidí, kteří do stejného souboru zakódují svou zprávu pomocí záměrných chyb. V tomto případě může nastat, že část již uložené zprávy se poškodí. Nicméně to může být zachráněno opět použitím samoopravného kódu ve zprávě, kterou ukrýváme. Samozřejmě tohle může fungovat jen omezeně – zpráva může být poškozena jen do určité míry.

4.2.2 Samoopravné kódy a způsob fungování

Na první pohled samoopravné kódy vypadají velmi zázračně, ale přináší jednu negativní věc – redundanci. Například, aby bylo možno odhalit a opravit jednu chybu v blocích po 4 bitech, je nutno do tohoto bloku začlenit další 3 bity. Při použití 6 bitů a přidání pouze jediného bitu pro mechanismus samoopravného kódu, je možné pouze odhalení maximálně jedné chyby.

Samoopravné kódy využívají takzvanou Hammingovu vzdálenost. Vzdálenost mezi dvěma stejně dlouhými bitovými bloky je potom taková, kolik bitů na odpovídajících pozicích je různých. Kupříkladu bloky 1111 a 1101 mají Hammingovu vzdálenost 1, protože se liší jediným bitem na třetí pozici. Jiným příkladem jsou bloky 1111 a 1000, kde je vzdálenost tři.

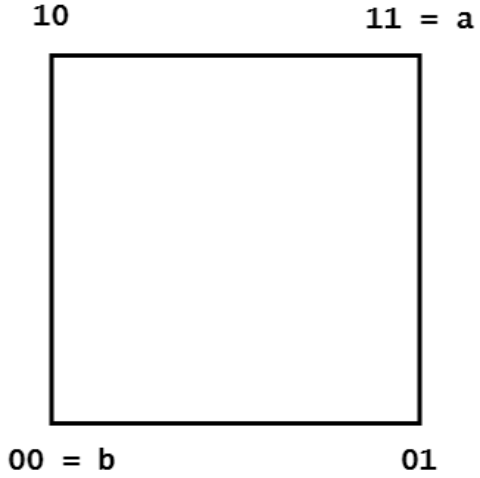
Samoopravný kód může vypadat tak, že každému znaku, který používáme v tajné zprávě, je přiřazena konkrétní bitová kombinace dané délky. Avšak nikoli všechny bitové kombinace této délky jsou využity – existují zde bitové kombinace, které nepatří k žádnému znaku. Tyto bloky jsou redundantní, ale slouží právě k odhalení chyb, nebo dokonce k jejich opravě. Když se v průběhu procesu narazí na takový bitový blok, je jasné, že muselo dojít k chybě.

Když je tento kód navržen tak, že změnou libovolného bitu libovolné bitové kombinace, která je přiřazena ke znaku, nezískáme bitovou kombinaci, která je přiřazena jinému znaku, je tento kód schopný detekovat jednu chybu. Jinak a zkráceně řečeno – kód je schopen detekovat jednu chybu právě tehdy, když Hammingova vzdálenost všech různých dvojic bitových kombinací, odpovídající znakům, je alespoň 2. Analogicky při detekci dvou chyb je nutno, aby mezi libovolnou dvojicí byla Hammingova vzdálenost alespoň 3.

Dále zbývá otázka, za jakých podmínek je algoritmus samoopravného kódu schopen opravovat chyby. Pro jednoduchost budeme uvažovat, že bitové kombinace, které odpovídají znakům, budou nazývány znakové a ostatní kombinace budou neznakové. Aby algoritmus byl schopný opravit jednu chybu, musí nejprve platit to, že všechny znakové kombinace musí být od sebe vzdáleny alespoň o Hammingovu vzdálenost 3. Z toho vyplývá, že pro libovolnou neznakovou kombinaci, která je od nějaké znakové ve vzdálenosti 1, platí, že musela vzniknout jednou chybou právě z této znakové kombinace. To znamená, že pokaždé, když se taková neznaková kombinace objeví, je detekována a opravena chyba.

Pokud má kód opravovat až dvě chyby, tak je předchozí podmínka zesílena. Tentokrát vzdálenost mezi znakovými kombinacemi musí být minimálně 5 a všechny neznakové kombinace v maximální vzdálenosti 2 od nějaké znakové kombinace jsou vytvářeny opět vznikem chyb z této znakové kombinace. Analo-

Obrázek 2: Jeden bit navíc



gickým navyšováním vzdáleností mezi znakovými kombinacemi se dají definovat kódy opravující více chyb. Obecně tedy platí, že pokud je vzdálenost mezi znakovými kombinacemi n, tak je v tomto kódu možno opravovat $\lfloor \frac{n-1}{2} \rfloor$ chyb. Kódům, které jsou schopny opravit právě jednu chybu, se říká Hammingovy kódy.

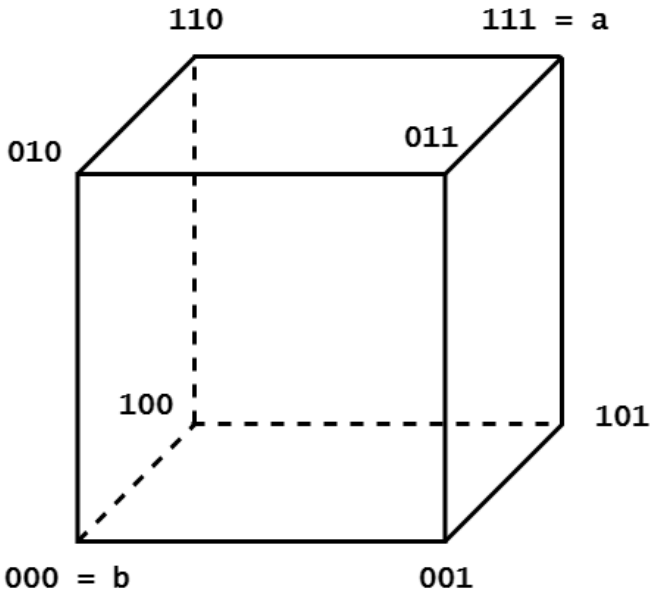
Pro jednoduchost uvažujme symboly, například a a b. Na jejich zakódování nám stačí 1 bit. Do symbolů zakomponujeme samoopravný kód nejdříve tak, že je rozšíříme o 1 bit a to tak, že a budeme kódovat jako 11 a b jako 00. Poté schéma Hammingových vzdáleností je zobrazeno na obrázku 2. Je patrné, že zbytek kombinací dvojic bitů, které jsou „volné“ (nekódují se jako nějaký symbol) mají Hammingovu vzdálenost od obou symbolů stejnou a neexistuje od nich žádná nejmenší vzdálenost k nějakému symbolu. Tedy v tomto případě není možno opravit žádnou chybu (pokud chyba nastane). Jelikož, když se například dvojice bitů 11 změní na 10, tak se neví, jestli byl změněn bit druhý (byla to tedy 11 a symbol a), a nebo jestli se chybou změnil první bit (byla to 00 a symbol b). Ale co se týče detekce chyby, tak právě při dvojici 01 nebo 10 je detekována chyba, jelikož neodpovídají žádnému symbolu.

Když se přidá ještě jeden bit navíc, tak už je možná i oprava jedné chyby. Ilustrací aktuální situace je nyní obrázek 3, kde symbol a kódujeme trojicí 111 a symbol b trojicí 000. Nastane-li změna jediného bitu v symbolu b, například na hodnotu 010, jde vidět, že tato hodnota má Hammingovu vzdálenost od a 2 a od b 1. To znamená, že za předpokladu, že nastala jediná chyba, je jednoznačné, že původní hodnota byl symbol b – je možné tuto chybu opravit.

Při jiných poměrech bitů pro samotné symboly a pro extra bity pro detekci a opravu chyb může nastat i případ, kdy je do určitého počtu chyb možnost opravy chyb a při ještě jedné chybě navíc je možno už jen chyby detekovat, ale již bez opravy.

Jak již bylo řečeno v sekci 4.2.1, je možné použít úmyslných chyb v samoopravném kódu jako techniku steganografie. To ale přináší úskalí v tom, že se již ztrácí možnost opravy chyb, když ještě navíc dojde k neúmyslné změně. V před-

Obrázek 3: Dva bity navíc



chozím příkladu se symbol a kódoval více různými způsoby (111, 011, 101 a 110). Když by se na mnoha místech v kódu blok 111 úmyslně zaměnil za 011 a poté by došlo k chybě na druhém místě, tedy by vznikl blok 001, který již odpovídá symbolu b.

4.3 Periodické kódy

Samoopravné kódy vynikaly proto, že v určitém poměru bitů pro informaci a paritních bitů byly schopny opravit chyby v kódu. Ale aby bylo možno opravit nebo detekovat více chyb v odpovídajícím bloku je nutno použít velké množství paritních bitů. Proto existují ještě takzvané periodické kódy, které sice nevynikají v opravách chyb, ale zato velmi efektivně detekují více chyb na jednom místě (tzv. burst errors). Takové chyby mohou nastávat docela běžně například na poškrábaném CD.

Periodické kódy, nebo také konvoluční kódy, nefungují jako samoopravné kódy pouze po určitých blocích. Paritní bity se vyskytují periodicky v kódu za bloky bitů pro informaci určité délky. Hodnota paritních bitů je závislá rovnou na více bitech v různých blocích. To znamená že změna několika bitů v jednom bloku se promítne změnou rovnou na několika paritních bitech. Proto je jednoduché detekovat „burst“ změny.

Takový jeden blok i s paritním bitem p na konci může vypadat následovně:

$$b_{(i,1)}, b_{(i,2)}, b_{(i,3)}, b_{(i,4)}, p_{(i,1)},$$

kde hodnota paritního bitu p může být vypočítána například:

$$p_{(i,1)} = b_{(i,1)} + b_{(i-1,2)} + b_{(i-2,3)} + b_{(i-3,4)} \bmod 2,$$

přičemž $i-x$ jsou předchozí bloky. Z toho plyne, že takový paritní bit je kombinací hodnot aktuálního a dalších třech předešlých bloků.

Problém paradoxně je, když nastane pouze jediná chyba. Zde není možno rozeznat, jestli je chyba v nějakém bitu nesoucí informaci nebo v paritním bitu, a tím pádem znovu také v některém z bitů v několika různých blocích, kterým odpovídá tento paritní bit. Změnou například třech bitů nesoucí informaci, které jsou vedle sebe, se změní hodnota právě tří paritních bitů, podle jejichž kombinace je snadné nalézt chybu. Při extrémních chybách, kdy je jeden celý blok informace chybný je možné dokonce chybu opravit. Tudíž v tomto smyslu pracuje právě naopak než samoopravný kód.

4.4 Rozdělení tajné informace

Vcelku jednoduchým a na druhou stranu velmi efektivním přístupem pro utajenou komunikaci je rozdělení tajné informace (secret-sharing) do více zpráv, a ty poté směrovat různými cestami k příjemci. Poté je nutné mít všechny zprávy s „fragmenty" informace, aby bylo možné zrekonstruovat zpět tajnou informaci. Nebo je také možné, za použití techniky samoopravného kódu, mít pouze $n - k$ zpráv, kde k je počet chyb, které je schopen samoopravný kód opravit.

4.4.1 Potřeba všech n částí

Jako první bude rozebrán jednodušší případ a to ten, kdy je potřeba všech n zpráv pro zrekonstruování informace. Když máme nějaký klíč, kterým je nějaké přirozené číslo, můžeme tento klíč „rozdělit"[11] tak, že složením n podklíčů získáme tajný klíč. Takové složení může reprezentovat například binární operace sčítání. V případě kdy pro tajný klíč X platí, že $X = X_1 + X_2 + \cdots + X_n$ potřebujeme všech n podklíčů, abychom zjistili klíč.

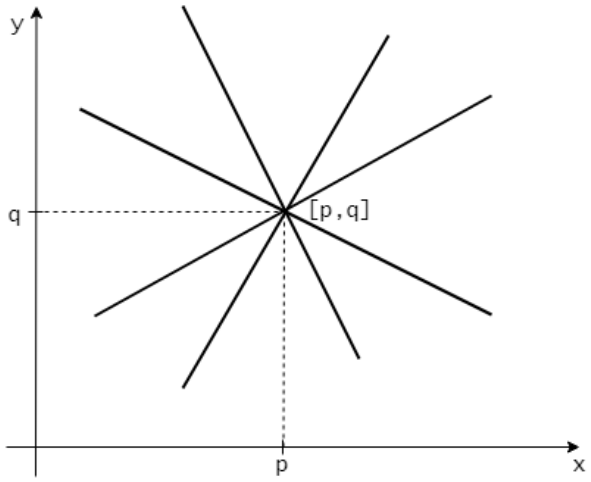
V praxi je lepší místo obyčejných čísel použít bitové bloky délky m jako podklíčů a pro složení bitovou operaci XOR (exkluzivní disjunkce, $\oplus$). Při „rozdělení" klíče na 4 části ($n = 4$) a použití pěti bitů pro každý podklíč ($m = 5$) může konkrétní příklad vypadat následovně:

$$\begin{aligned} X_1 &= 01101 \\ X_2 &= 10110 \\ X_3 &= 11010 \\ X_4 &= 01010 \\ X = X_1 \oplus X_2 \oplus X_3 \oplus X_4 &= 01011 \end{aligned}$$

Vytváření podklíčů v tomto případě je velice jednoduché, protože $n - 1$ podklíčů se může vygenerovat náhodně a poslední n-tý klíč se dá jednoduše vypočítat takto:

[11] Slovo „rozdělit" je v uvozovkách, protože klíč fyzicky nerozdělujeme. Viz dále.

Obrázek 4: Potřeba dvou přímek k získání tajemství



$$X_n = X \oplus X_1 \oplus X_2 \oplus \cdots \oplus X_{n-1}$$

Z pohledu entropie z rovnice popisující vztah mezi klíčem a podklíči $X = X_1 \oplus X_2 \oplus \cdots \oplus X_n$ vyplývá, že když každý podklíč X_i obsahuje m bitů a klíč je rozdělen na n podklíčů, tak celková entropie je $n \cdot m$ bitů. Za předpokladu, že byť jediný podklíč chybí, je zde stále m bitů míry neurčitosti, což odpovídá 2^m různým variacím, jak může vypadat poslední podklíč.

4.4.2 Potřeba jen *k* částí z *n* částí

Takovým nejjednodušším způsobem, jakým informaci schovat do více částí, a pak na její zrekonstruování potřebovat pouze část z nich, je využít geometrii. V tomto případě bude tajný klíč opět číslo, které uvažujeme na ose x v dvojrozměrném grafu. Poté náhodně zvolíme libovolné číslo na ose y a vznikne bod $[p, q]$. Tímto bodem povedeme n přímek, které budou reprezentovat tajné podklíče, jako v minulém případě.[12] Tento případ je zobrazený na obrázku 4. Je zjevné, že pomocí libovolné dvojice přímek jsme schopni získat daný bod $[p, q]$ a tím i tajné číslo p.

K tomu, aby nestačily pouze dvě přímky k odhalení tajemství, je nutno použít vícerozměrné prostory. Pro přímky v rovině platí, že jsou v $\mathbb{R}^2$ a v takovém prostoru stačí pouze 2 přímky ke zjištění bodu $[p, q]$. Ve zobecněném případě tedy platí, že v n-rozměrném prostoru $\mathbb{R}^n$ je nutno minimálně n $(n-1)$-podprostorů dimenze $\mathbb{R}^{n-1}$, které jednoznačně určují bod o n souřadnicích, jehož jedna složka je tajná informace. Samozřejmě $(n-1)$-podprostory nesmí být po dvou rovnoběžné, jinak by jednoznačně bod neurčily. Vícerozměrné prostory jsou už velmi složité na představu, ale u $\mathbb{R}^3$ je to ještě docela jednoduché. V něm průnik tří rovin jednoznačně určuje bod $[p, q, r]$.

[12] Přímky musí být samozřejmě funkce – přímka nesmí být kolmá k ose x, protože pak by byla touto přímkou jednoznačně daná hodnota klíče p.

Tohle řešení je také efektivnější oproti předchozímu řešení v podkapitole 4.4.1. Například v $\mathbb{R}^2$ znalost jediné přímky nepřidá více informace (nesníží neurčitost) ohledně toho, kde se souřadnice x nachází. Takže znalost této jediné přímky je k ničemu, pokud nezískáme alespoň dalších $k-1$ přímek. Takové způsoby utajování a rozdělování informací se nazývají jako perfektní (perfect secret-sharing systems). Hodnota může být stále teoreticky kdekoli od $-\infty$ do $+\infty$. Kdyby byla tajná informace rozdělena do obou souřadnic p i q, tak už by znalost jedné přímky snižovala neurčitost toho, kde se pod $[p, q]$ nachází (omezili bychom se z prostoru $\mathbb{R}^2$ na $\mathbb{R}^1$).

Jako tomu bylo v podkapitole 4.4.1, často se rozděluje tajná informace, která obsahuje m bitů, na n částí tak, že každá z částí má opět m bitů. To může být při použití hodně bitů pro tajnou informaci docela náročné na paměť, a proto části mají pouze $\frac{m}{n}$ bitů, kde tato velikost může také stačit. Navíc se ještě při tomto zmenšování částí provádí zašifrovaní původní tajné informace kryptografickým algoritmem. Až poté je tento kryptogram rozdělován.

Další informace o této metodě mohou být nalezeny v [1].

4.4.3 Reálný model principu rozdělení informace

V předešlé podkapitole bylo nastíněno řešení rozdělení informace pomocí geometrie, která pracuje s reálnými čísly. S reálnými čísly ovšem není možné pracovat v konkrétní implementaci počítačového software z důvodu nemožnosti jejich reprezentace.

Zaprvé by mohla být snaha se omezit na osách pouze na celá čísla (integer). Při zvolení tajného čísla p jako celého je potom nutno k tomuto bodu nalézt alespoň dvě přímky, kde každá je reprezentována dvěma čísly. První číslo označuje sklon přímky s osou x a druhé číslo průnik s osou y. Není možné, aby při zvolení libovolného celého p byla obě čísla celá.

Zadruhé, když by myšlenkou bylo používat typy s plovoucí řádovou čárkou, tak by docházelo k tomu, že výsledná čísla by byla pouze aproximace. I když by byla tato čísla velice přesná, různé procesory mohou tyto hodnoty vypočítat trošku jinak. Právě změna, klidně jen o jediný bit, není v kryptografii přípustná, protože právě při použití silných kryptografických funkcí změna jediného bitu znamená změnu v průměru poloviny bitů originální zprávy.

Východiskem je použití polynomů. Stačí se omezit pouze na rovinu $\mathbb{R}^2$, kde se místo přímek použije polynom řádu $k-1$, když je potřeba alespoň k částí pro získání tajné informace. Polynom se zkonstruuje tak, že první parametr polynomu p_0 bude ukrývat tajnou informaci a zbytek parametrů $p_1, p_2, \ldots, p_{k-1}$ se vygenerují náhodně. Poté je vygenerováno náhodně n bodů, které leží na polynomu, a každý z těchto bodů slouží jako podklíč. Při získání k bodů polynomu je možné polynom zrekonstruovat, a tím pádem je zjištěn i první parametr p_0, jakožto tajná hodnota.

Tento přístup má opět tu výhodu, že znalost téměř dostatečně mnoho bodů polynomu ($< k-1$), nesníží neurčitost toho, jaká je tajná hodnota, podobně jako v podkapitole 4.4.2. Další příznivou charakteristikou je fakt, že se v celém

výpočtu používají pouze celá čísla, takže je bez problému možnost přesné reprezentace v počítači. Nakonec ještě jedna výhoda od předchozího a to, že při nutnosti zvyšování počtu částí bylo v předchozím případě nutné použití prostorů vyšší dimenze, kde bylo potřeba k poddimenzí k získání bodu. Při vysoké dimenzi tyto poddimenze zabírají hodně místa pro jejich uložení v operační paměti. Naopak v případě polynomu je každá část reprezentována vždy pouhým bodem $[p, q]$.

4.4.4 Různé váhy podklíčů

Po tomto systému by bylo možné dále požadovat i to, aby některé části měly větší váhu ve smyslu toho, že určitých částí stačí méně ke kombinaci s ostatními, aby bylo možno zrekonstruovat tajnou informaci. Nejprve se opět rozdělí tajná informace na n podklíču, kterých je potřeba $k <= n$ k rekonstrukci. Nic nám ale nebrání rozdělit každý z n podklíčů na další podklíče nižší úrovně. Například uvažujme, že každý z podklíčů n_1 a n_2 je rozdělen na libovolně mnoho částí. Jsou rozděleny tak, že pro zrekonstruování n_1 je nutno zkombinovat minimálně m jeho částí a pro n_2 o jeho částí tak, že $m < o$. Je zjevné, že podklíče tvořené z n_1 mají vyšší hodnotu (stačí jich méně), než podklíče z n_2.

4.4.5 Rozdělení tajné informace v asymetrické kryptografii

Jak již bylo uvedeno v sekci 3.2.2, v asymetrické kryptografii se používají dva typy klíčů. Rozdělení tajné informace (privátního klíče) je zde velmi snadno proveditelné. Asymetrické šifry bývají dost často založeny na problému diskrétního logaritmu. Zkráceně – jde o nalezení hodnoty x takové, že známe hodnoty g, p a $g^x \bmod p$, kde x je právě soukromý klíč, g je generátor grupy a p je modul grupy (prvočíslo). Je ale možné soukromý klíč vyjádřit jako složení podklíčů $x_1, x_2, \ldots, x_n$ a z těch vypočítat veřejný klíč $a = g^{x_1} \cdot g^{x_2} \cdot \cdots \cdot g^{x_n} \bmod p$. Poté je tajná zpráva zašifrována jako $g^y \bmod p$. Při dešifrování zprávy je poté nutno každým soukromým klíčem dešifrovat zprávu $(g^y)^{x_i} \bmod p$ a nakonec se zkombinují mezivýsledky a je získána vlastní zpráva:

$$a^y = (g^y)^{x_1} \cdot (g^y)^{x_2} \cdot \cdots \cdot (g^y)^{x_n} \bmod p$$

Ve skutečnosti se a^y používá ke generování klíče pro další zašifrování zprávy (například pro AES).

Detailnější popis této metody je obsažen v [1].

4.4.6 Steganografické souborové systémy

Existuje koncept takzvaných steganografických souborových systémů. Takové systémy fungují tím způsobem, že na pevném disku je klasicky vyhrazený prostor pro soubory. Tyto soubory jsou na disku uloženy tak, že bez hesla není možno zjistit ani to, jestli se daný soubor vyskytuje na pevném disku (jinak se jeví

jako šum). Dokonce na takových systémech může existovat i hierarchie souborů taková, kde k různým úrovním náleží jiné heslo. [1, 9]

Jsou dva různé přístupy k těmto hierarchiím a heslům. První funguje tak, že když má uživatel heslo k dané úrovni, je nutné mít i všechna hesla z nižších úrovní k zobrazení souborů na aktuální úrovni. Druhý přístup je právě opačný. Ten při vlastnění hesla na dané úrovni zpřístupní i všechny soubory na nižších úrovních.

Ke každému zakódovanému souboru (úrovni hierarchie) existuje klíč, který má tolik bitů, kolik existuje souborů v systému. Když se na i-té pozici klíče daného souboru vyskytuje v bitu hodnota 1, tak to znamená, že se i-tý soubor v systému podílí na zakódování daného souboru. Zakódování souboru je často realizováno pomocí bitové operace XOR ($\oplus$) nad soubory, které se podílí na zakódování tohoto souboru.

Nejjednodušší je si takový souborový systém představit jako matici o rozměrech $m \cdot n$, kde m je počet souborů a n je velikost souborů, respektive jejich bitová reprezentace.[13] Bitové reprezentace i-tého souboru (řádek matice) se označí C_i a klíč j-tého souboru se označí K_j. Dále se definuje i-tá hodnota bitu v klíči (odpovídá začlenění i-tého souboru) jako $K_j(i)$. Konečně se může j-tý soubor v systému získat následovně:

$$\bigoplus_{i=1}^{m} K_j(i) \cdot C_i.$$

Ještě je nutno dodat, že klíče jednotlivých souborů musí být ortogonální, aby nedocházelo k poškozování ostatních souborů, když je další přidáván nebo jeden ze stávajících měněn. To znamená, že pro každou dvojici klíčů K_i a K_j musí platit:

$$K_i \cdot K_j = \begin{cases} 1 & \text{pokud } i = j \\ 0 & \text{jinak,} \end{cases}$$

kde, v tomto případě, operace $\cdot$ je skalární součin.

Více podrobností této metodě je obsaženo v [1].

4.5 Komprese dat

Soubory, nebo spíše části dat, často spadají do různých vzorů, což má za následek redundanci (nadbytečnost) dat. U takových dat je možná takzvaná komprese, což je modifikace dat takovým způsobem, že obsahují stále stejnou informaci, ale objem dat je nižší. Výjimkou je ztrátová komprese, kde se za cenu malé ztráty informace získá mnohem větší míra komprese. Samozřejmě existují různé algoritmy ke kompresi, jejichž míra komprese se liší. Efektivita komprese nezáleží pouze na algoritmu samotném, ale také na datech, na kterých je komprese prováděna.

[13] Při velkých souborech je možno rozdělit soubor do více souborů velikosti n.

Mezi techniky kompresních algoritmů pro bezeztrátovou kompresi patří statistické metody, mezi které patří například Huffmanovo kódování, dále existují slovníkové metody s konkrétní implementací LZ78 a také metody kontextové. Mezi ztrátové kompresní algoritmy patří například Kosinova transformace, která se používá pro kompresi obrazu, konkrétně pro formát JPEG.

4.5.1 Komprese ve vztahu ke steganografii

Komprese dat je další významnou kapitolou v příběhu steganografie. Jak již bylo popsáno, její hlavní charakteristikou je to, že obecně zachovává informaci a snižuje objem dat. To znamená, že je vyšší poměr informace na bit, což dále znamená, že je i vyšší entropie (popsáno v sekci 3.1). Tím je tedy snadnější ukrytí tajné informace z hlediska vyšší neurčitosti v datech. Jinými slovy, zkomprimovaná data mají blíže k šumu (vypadají více náhodně).

Další, a ještě možná i významnější, význam komprese ve steganografii je ten, že se pomocí ní dá provádět takzvaná imitace dat (mimic data). Ta spočívá v tom, že kompresní algoritmus vytvoří nějaký vzor, který poté přesněji specifikuje parametry. Je možné naše data, nesoucí tajnou informaci, upravit do takové formy, aby při kompresi byl tento vzor s další specifikací co možná nejpodobnější nějakému vzoru jiných dat. Zkomprimovaná data pak vypadají velmi podobně.

Podobný princip se dá provést i opačným směrem. Předpokládejme nějaký kompresní algoritmus, který pracuje s určitými daty velmi efektivně (má velkou míru komprese). Poté uvažujme data, která jsou právě tímto algoritmem efektivně zkomprimovatelná. To znamená, že algoritmus využívá nějaký velmi vhodný vzor právě pro tato data. Co se ale stane, provedeme-li dekompresi nad našimi daty, obsahující skrytou informaci, která nemusí být vhodná pro tento kompresní (dekompresní) algoritmus? Dekomprese uvažuje určitý vzor na našich datech, a tím určitým způsobem nabalí naše data tak, že bude imitovat právě data, která jsou k této kompresi určena. Například uvažujme specifická data jako reprezentaci černých a bílých pruhů různých šířek. Kompresní algoritmus efektivně pracuje s těmito daty, používá vhodný vzor, velmi sníží obsah dat a pouze přidá určité parametry s informací o šířkách daných pruhů. Když naše data poté určitým způsobem poupravíme a provedeme dekompresní algoritmus, tak tím vlastně „napasujeme“ černo-bílé pruhy na naše data. Tím dochází k imitaci původních specifických dat. Kompresním algoritmem opět získáme naše původní poupravená data.

4.5.2 Huffmanovo kódování

Velmi významnou kapitolou v tématu pojednávající o kompresi je bez pochyby Huffmanovo kódování. Toto kódování generuje tzv. optimální kód – takový kód, jehož kódová slova mají nejmenší průměrnou délku mezi všemi možnými kódováními.

Hlavní vlastností tohoto kódování, i všech kódování obecně, je prefixová vlastnost. To znamená, že žádné kódové slovo (bitová reprezentace přiřazena symbolu)

nesmí být prefixem jiného kódového slova. Tato vlastnost je důležitá z důvodu jednoznačného dekódování zakódované zprávy.

Vytvoření takového optimálního kódu je v zásadě docela jednoduché – nejprve je zjištěna četnost vstupních symbolů nad kterou se bude kódování provádět a poté se tyto symboly seřadí sestupně do posloupnosti podle této četnosti. Následně pracuje algoritmus rekurzivně tak, že z posloupnosti vytvoří novou posloupnost o jeden prvek kratší tím způsobem, že sloučí dva znaky s nejmenší četností do jednoho, u tohoto sloučeného znaku se uvažuje součet četností obou znaků a nakonec je ještě tento sloučený znak umístěn na správnou pozici v posloupnosti. Limitní podmínkou této rekurze je takový stav, kdy posloupnost obsahuje pouze dva znaky a až od této chvíle dochází k vytváření samotného kódu, což znamená přiřazovaní bitových kombinací jednotlivým symbolům. To se opět děje rekurzivně tak, že jednomu ze znaků se do jeho kódové reprezentace přidá na konec bit 1, druhému bit 0 a oba znaky se rozdělí na dvojici znaků reverzně tak, jak tomu bylo v první části slučování a rekurzivně se pokračuje nad každou vytvořenu dvojicí dále. K rozdělení samozřejmě může dojít jen tehdy, pokud znakem již není výchozí znak ze vstupu. Pokud jde již o vstupní znak, jde o limitní podmínku rekurze a zde vytváření kódu pro aktuální větev končí.

Příkladem vytvoření kódu podle Huffmanova kódování budiž vstupní řetězec ze sekce 3.1 o entropii (pro jednoduchost uvažováno bez mezer).

```
barbara a barbora u baru
```

V řetězci délky 20 se vyskytují znaky s následující četností:

znak	a	b	o	r	u
četnost	7	5	1	5	2

Nejprve se tedy znaky uspořádají podle četností sestupně:

symbol	a	b	r	u	o
četnost	7	5	5	2	1

V dalším kroku se sloučí poslední dva znaky, sečtou se jejich četnosti a tento znak se zařadí na správnou pozici:

symbol	a	b	r	{u, o}
četnost	7	5	5	3

Následují další kroky:

symbol	{r, u, o}	a	b
četnost	8	7	5

Po dosažení limitní podmínky rekurze následuje vlastní vytváření kódu. Pro levé symboly se bude aktuální kód pro tento symbol řetězit například s bitem s hodnotou 1 a pravý symbol s bitem s hodnotou 0. První krok této fáze odvíjení bude vypadat následovně:

symbol	{a, b}	{r, u, o}
četnost	12	8

symbol	{a, b}	{r, u, o}
četnost	12	8
bitová reprezentace	1	0

Poté se rozdělí oba symboly stejným způsobem, jako byly při fázi navíjení sloučeny a rekurzivně je na obě dvojice použit stejný proces:

symbol	a	b	r	{u, o}
četnost	7	5	5	3
bitová reprezentace	11	10	01	00

Pro symboly a, b a r bylo dosaženo limitních podmínek a rekurze zde končí a již mají přiřazené finální bitové reprezentace v rámci kódu. Ještě zbývá rozdělit a doplnit bitovou reprezentaci znaků u a o:

symbol	a	b	r	u	o
četnost	7	5	5	2	1
bitová reprezentace	11	10	01	001	000

4.5.3 GZSteg

Kompresní algoritmus GZSteg vychází z velmi oblíbeného kompresního algoritmu GZIP, který funguje tak, že místo opětovných výskytů frází ve vstupním textu jsou takovéto výskyty nahrazovány příznaky. Tyto příznaky nesou informaci o tom jaké a kolik znaků se mají na toto místo přenést z již zpracované části vstupu, aby výsledný text odpovídal vstupnímu. Například vstup:

```
Steganografie je velmi spjata s kryptografií
```

bude upraven do formy:

```
Steganografie je velmi spjata s krypt<31, 6>í,
```

kde se podřetězec *ografi* nahradí příznakem *<31, 6>*, který znamená vrať se o 31 pozic zpět a zde následujících 6 znaků doplň místo příznaku.

Zde se konečně dostáváme k tomu, v čem se GZSteg liší od GZIP. Pracuje úplně stejně jen s tím rozdílem, že u příznaků, které tímto způsobem kopírují alespoň 5 znaků, se navíc kóduje bit skryté zprávy tak, že jestli tento bit má být 1, tak se kopíruje o jeden znak méně. To znamená, že v předchozím případě, kdyby měl příznak nést skrytý bit s hodnotou 1, tak by zpráva vypadala:

```
Steganografie je velmi spjata s krypt<31, 5>ií
```

Jestliže by nesl bit s hodnotou 0, tak by příznak vypadal jako v prvním případě.

Je zřejmé, že pokud by byla původní zpráva zkomprimována oběma algoritmy, a ty by poté byly dekomprimovány, byly by obě dekomprimované zprávy totožné. Nicméně je snadné odhalit ve zprávě, že byla zkomprimována právě GZSteg velmi jednoduchou analýzou.

4.6 Imitace dat (Data mimicry)

Imitací dat se rozumí vytvoření nových dat na základě vstupních tak, že mají se vstupními daty určitou podobnost. Například můžeme uvažovat nějaký článek o steganografii jakož to vstupní data. Data, která imitují tento článek, se poté vytvoří tak, že se budou generovat slova, fráze a věty, které se statisticky často vyskytují ve vzorovém článku.

První způsob, jak takový imitovaný text vytvořit, je pomocí takzvaných imitačních funkcí n-tého řádu. Tyto funkce využívají statistický model, což znamená, že si vytváří statistickou četnost všech řetězců délky n, které se vyskytují ve vstupním textu. Poté se imitovaný text vytváří tak, že se začne libovolnou n-ticí ze vstupního textu. Další znaky se generují tím způsobem, že se najdou všechny n-tice ze statistiky, které mají prvních $n - 1$ znaků shodných s posledními $n - 1$ znaky již vytvořeného textu. Z těchto n-tic je vybrána jedna podle pravděpodobnostního rozdělení daného vytvořenou statistikou. Znak na konci vybrané n-tice je přidán na konec vytvářeného textu. Tento proces přidávání znaku na konec se provádí tak dlouho, dokud není text dostatečně dlouhý.

Následuje algoritmus pro vytvoření imitovaného textu n-tého řádu:

1. Analýza imitovaného textu – Vytvoří se seznam všech řetězců délky n vyskytujících se v textu a zaznamená se jejich četnost, čímž je vytvořená statistika.

2. Vytvoření začátku imitovaného textu – Zvolí se libovolná n-tice ze statistiky jako počátek imitovaného textu, a to buď náhodně nebo podle pravděpodobnostního rozdělení n-tic podle vytvořené statistiky.

3. Vytváření zbytku imitovaného textu dokud není vygenerován dostatečně dlouhý text:

   (a) Vyhledají se všechny n-tice ze statistiky, u kterých prefix délky $n - 1$ odpovídá postfixu (také samozřejmě délky $n - 1$) prozatím vytvořeného imitovaného textu.

   (b) Podle pravděpodobnostního rozdělení vybraných n-tic se zvolí právě jedna.

(c) Poslední znak ze zvolené n-tice se přidá na konec imitovaného textu.

Mimické funkce 3. řádu a nižší produkují relativně špatnou imitaci originálního textu, ale funkce vyšších řádů už produkují text, který vypadá na první pohled lépe. Avšak samozřejmě takový text nemusí dávat úplně smysl. Nicméně čtenář, který není do dané tématiky dostatečně zasvěcen a není schopen porozumět významu, by mohl být ošálen při funkci dostatečně vysokého řádu.

Co se týče automatického rozpoznávání textu, tak jsou na tom mimické funkce ještě lépe. Algoritmy rozpoznávající text podle statistického výskytu znaků v textu jsou ošáleny při použití libovolné mimické funkce, protože i ta vychází právě ze statistického modelu originálního textu. Například písmeno *e*, které se v anglickém textu vyskytuje nejvíce, se bude statisticky nejvíce vyskytovat v imitovaném textu a podobně.[14] Algoritmy rozpoznávající text na úrovni gramatiky je mnohem náročnější ošálit, ale ani samotný člověk se ne vždy drží zásadami gramatiky a může používat různé slangy, nespisovné výrazy a idiomy.

Takovéto řešení pomocí mimických funkcí by mohlo být ještě vylepšeno například tím způsobem, že by pro generování textu používalo tzv. genetické algoritmy, které by vytvářely velmi vhodné vzory odpovídající originálnímu textu.

4.6.1 Ukrytí informace v imitovaných datech

Dosud bylo popisováno pouze to, jak vytvořit imitaci dat, ale nikoli jak takováto data mohou ukrývat skrytou informaci. Podstata ukrytí informace tkví v náhodném výběru následujícího znaku z předchozího algoritmu. Když tento výběr nebude náhodný, tak právě určitým výběrem je možnost ukrýt informaci.

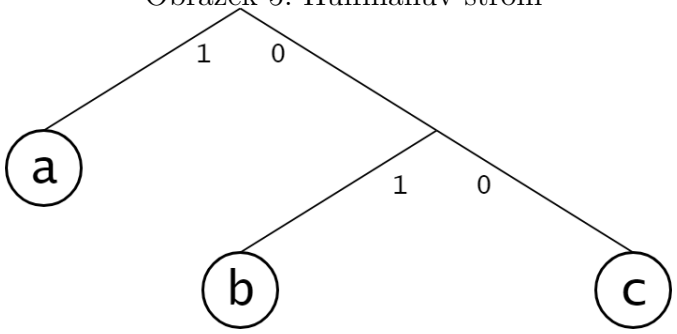
K tomu se využije dosti netradičního přístupu ke kompresi, konkrétně přístupu k Huffmanovu kódování. To bude fungovat naopak, než k čemu je normálně tento algoritmus využit. Nejdříve se klasicky sestrojí Huffmanův strom z aktuálních znaků a jejich statistických četností. Následně z proudu bitů, které budou reprezentovat tajnou zprávu, se budou dekompresí podle vytvořeného Huffmanova stromu vytvářet znaky imitovaných dat.

Tajnou informaci z takovýchto dat je možno extrahovat podobným způsobem, jakým byla vytvořena. Je nutno mít zdrojová data, na kterých byla vytvářena statistika četnosti n-tic, a právě přesně takovou statistiku si také vytvořit a samozřejmě je nutno mít také imitovaná data se skrytou informací. Pro získání bitů, reprezentujících tajnou informaci, se provádí naopak komprese jednotlivých znaků imitovaného textu podle odpovídajících Huffmanových stromů.

Uvažujme například, že se algoritmus pro vytváření imitace dostal do bodu, kde má podle statistiky na výběr ze tří různých znaků *a*, *b* a *c*, které může přidat do imitovaného textu, kde *a* má nejvyšší pravděpodobnost výskytu. Odpovídající Huffmanův strom pro tento případ vypadá (až na izomorfismus) právě jako na obrázku 5. Takže když budeme mít aktuálně na vstupu bit 1, tak to znamená,

[14] Samozřejmě také záleží na tom, jakou tématikou se vstupní text zabývá, a podle toho jak frekventovaně volí určitá specifická slova. Například text často obsahující slovo *queue* bude mít mnohem častější frekvenci písmena *u* než jiné texty.

Obrázek 5: Huffmanův strom



že se do imitované zprávy přidá znak a, jestli je na vstupu dvojice bitů 01, tak znak b a při sekvenci bitů 00 znak c.

4.6.2 Problémy ukrývání tajné informace v imitovaných datech

Velkým nedostatkem této metody pomocí Huffmanova kódování je to, že tento přístup již nedodržuje statistické vlastnosti přirozeného jazyka. Znak a se totiž bude ve zmíněném případě objevovat právě s pravděpodobností $\frac{1}{2}$ a znaky b a c každý s pravděpodobností $\frac{1}{4}$, což obecně neodpovídá statistické charakteristice jazyka.[15] Tuto vlastnost má takto imitovaný text z důvodu binární struktury Huffmanova kódu.

Tato špatná vlastnost Huffmanova stromu, kde jednotlivé znaky se vyskytují vždy s pravděpodobností právě $\left(\frac{1}{2}\right)^n$, se dá částečně upravit. Provede se to tím způsobem, že se v Huffmanově stromu uvažuje tentýž znak vícekrát a jeho součet pravděpodobností je roven jeho pravděpodobnosti v originálním Huffmanově stromu. Touto granularitou dojde k většímu zpřesnění pravděpodobností výskytu jednotlivých znaků, ale dochází tím k neefektivitě, jelikož tímto je Huffmanův strom větší a jednotlivé znaky se tak kódují více bity. To ale neznamená, že vznikne více prostoru pro tajnou informaci. Tyto bity, které jsou „navíc“, slouží právě jen k zpřesnění pravděpodobností jednotlivých znaků. Navíc je problém se znaky, které mají vyšší pravděpodobnost než $\frac{1}{2}$ a nejdou jednoduše řešit tímto způsobem. V těchto případech již nezbývá nic jiného, než neuvažovat znaky samostatně, ale uvažovat je s jejími pravděpodobnostmi po dvojicích nebo obecně n-ticích. Těchto „znaků“ je pak sice více, ale pravděpodobnost se mezi ně rozloží a nedochází ke zmíněnému problému, nebo alespoň ne v takové míře.

Když algoritmus má zrovna na výběr pouze jediný znak, tak se tímto způsobem nedá zakódovat žádná informace, jelikož není možné nijak reprezentovat to, že na vstupu je bit 1 nebo 0. Nicméně při dostatečně dlouhém zdrojovém textu tento případ nastává minimálně, jestli vůbec.

[15] Například pokud by šlo o konkrétní znaky e, j a q v anglickém jazyce, tak ty budou mít rozdělení pravděpodobnosti velmi odlišné od zmíněného příkladu.

4.6.3 Imitace dat v obrázcích

Do teď byla imitace vztahována pouze na data textové povahy. Stejný princip se dá použít na data reprezentující obrázky.

Konkrétně se hodí při používání již zmíněné techniky LSB v podkapitole 4.7.1, která ukrývá tajnou informaci do nejméně významných bitů v pixelech. Tento přístup trpí tím nedostatkem, že mnoho obrázků mívá (alespoň z části) oblasti, kde jsou hodnoty nejméně významného bitu stejné (nejpravděpodobněji se jedná o přesně tutéž barvu). Po přepsání těchto bitů tajnou zprávou je následně velmi jednoduché analyzovat obrázek a zjistit, že s ním mohlo být určitým způsobem manipulováno.

Tento problém se dá ošetřit opět využitím imitace. Původní obrázek může sloužit pro vytvoření statistik výskytu n-tic nejméně významných bitů po sobě jdoucích, vytvořit odpovídající Huffmanovy stromy a do obrázku zanést tajnou informaci pomocí těchto statistik a Huffmanova stromu.

4.6.4 Imitace dat za pomoci formálních gramatik

Jak již bylo popsáno v sekci 4.6.2, pouze za pomoci statistik je sice možno generovat imitovaný text, který ale gramaticky nedává smysl, což čtenář ihned odhalí. Proto je možno metodu imitace vylepšit za použití formálních gramatik, konkrétně bezkontextových gramatik, které jsou například hojně používány v překladačích programovacích jazyků.

Formální gramatika je nástroj pomocí které se vytváří tzv. věty gramatiky. Ty nejsou nic jiného než určité řetězce znaků nad danou abecedou symbolů. Jedna z komponent formální gramatiky je právě abeceda symbolů, ze kterých jsou složeny její věty gramatiky. Ty se ve formální gramatice nazývají terminální symboly. Dále do formální gramatiky patří neterminální symboly, které slouží k odvozování nových řetězců (větných forem), které mohou obsahovat další neterminální symboly, které se opět odvozují. Samotné odvozování je prováděno podle množiny odvozovacích pravidel formální gramatiky. Odvozovací pravidlo je uspořádaná dvojice reprezentující, že se řetězec levé strany pravidla má přepsat na řetězec pravé stany pravidla. Pro potřeby této kapitoly postačí tzv. bezkontextové formální gramatiky, které jsou charakteristické tím, že na levé straně pravidla obsahují vždy právě jeden neterminální symbol. Pravidla bezkontextové formální gramatiky se zapisují ve tvaru:

$$A \ \rightarrow b|c,$$

kde:

- $A \ \in N$ ... neterminální symbol
- $b, c \in (N \bigcup \Sigma)^*$ ... větné formy
- Znak | odděluje alternativy přepisu neterminálu A (dvě různá pravidla).

Při odvozování větných forem (řetězce terminálů a neterminálů) se začíná větnou formou obsahující pouze startovní neterminál (typicky označován jako **S**). Dále také uvažujeme při odvozování podle pravidel gramatiky takzvanou nejlevější derivaci. To znamená, že když větná forma obsahuje více neterminálních symbolů, bude se jako první odvozovat vždy ten, který je nejlevější.

Z názorných důvodů budou terminály i neterminály označovány celými slovy s tím rozdílem, že neterminály budou tučným písmem. Z toho důvodu stačí pro jednoznačnost psát již pouze pravidla gramatiky a není nutno definovat dané dvě množiny.

Konkrétní gramatika může vypadat například takto:

- **S** → **kdo co jak**
- **kdo** → Alice | Bob
- **co** → šifruje | dešifruje
- **jak** → symetricky | asymetricky

Pomocí takové gramatiky můžeme vygenerovat například větu „Alice šifruje symetricky“.

4.6.4.1 Kódování tajné informace

Co se týče ukrývání tajné informace do takto generovaného textu, tak je možno kódovat obdobně, jako tomu bylo při využívání statistických vlastností zdrojového textu. Tedy tajná informace se kóduje možností odvozovat z konkrétního neterminálu více způsoby.

S využitím právě zmíněné gramatiky uvažujme, že při použití prvního pravidla při odvozování neterminálů **kdo**, **co** a **jak** je kódován bit 0 a při použití druhého pravidla kódujeme 1. Když bychom chtěli kódovat bitové slovo 101, tak odvozování bude vypadat následovně. Nejdříve se odvodí ze startovního neterminálu **S** trojice neterminálů **kdo**, **co** a **jak**. Zde máme na výběr pouze jediné pravidlo, takže zde není možno kódovat žádnou informaci.

S → **kdo co jak**

Následně se bude odvozovat nejlevější neterminál **kdo** podle jeho druhého pravidla, jelikož chceme kódovat bit 1.

kdo co jak → Bob **co jak**

Dále je na řadě nejlevější neterminál **co**. Ten se bude odvozovat podle prvního pravidla, z důvodu kódování 0.

Bob **co jak** → Bob šifruje **jak**

Na závěr kódujeme bit 1, takže poslední neterminál **jak** odvodíme podle druhého pravidla.

Bob šifruje **jak** → Bob šifruje asymetricky

Tím je vytvořena věta gramatiky a navíc je v ní zakódováno kódové slovo 101 vzhledem k této gramatice. Samozřejmě čím více bude pravidel přepisu konkrétního neterminálu, tím bude možno zakódovat více informace při jeho odvozování. Například kdyby existovaly 4 pravidla pro odvození konkrétního neterminálu, bylo by možno kódovat všechny možné kombinace dvojic bitů 00, 01, 10 a 11.

4.6.4.2 Parsování imitovaných dat a získání tajné informace

Když už je imitovaný text se zakódovanou tajnou zprávou podle nějaké gramatiky vygenerovaný, zbývá ještě popsat proces pro opětovné získání tajné zprávy z textu. Jelikož jde o text vytvořený pomocí formální gramatiky, tak se tento proces nazývá syntaktická analýza, nebo parsování. Právě tento proces je velmi hojně užíván při překladu (kompilaci) programovacích jazyků.

První důležitou vlastností, jakou musí mít gramatika, aby byla tímto způsobem analyzovatelná, je její jednoznačnost. Jednoznačná gramatika je charakteristická tím, že každá její věta má pouze jedinou derivaci (odvození) z dané gramatiky. Jinými slovy – pro to, aby gramatika vygenerovala danou větu gramatiky, má v každém odvozovacím kroku jasně dáno, které pravidlo má použít jako další. Za předpokladu, že by gramatika nebyla jednoznačná, mohla by daná věta gramatiky kódovat různé tajné zprávy, a proto by byla v tomto případě nepoužitelná.

Dále je pro jednoduchou analýzu zapotřebí, aby daná gramatika byla v tzv. Greibachově formě. Taková gramatika musí mít všechna pravidla ve tvaru, kdy existuje rozdělení řetězce na pravé straně pravidla na dvě části, kdy levá strana se skládá jen z terminálů a pravá strana pouze z neterminálů. Navíc každá bezkontextová gramatika je jednoduše převoditelná na gramatiku v Greibachově formě upravováním pravidel, které danou vlastnost nesplňují a přidáváním nových pravidel. Greibachova forma zajišťuje to, že jak je věta gramatiky analyzována zleva doprava, tak se tím dekódují bity tajné zprávy také postupně zleva doprava. Kdyby byla použita gramatika, která není v Greibachově formě, obecně by se získávaly bity tajné zprávy v různém pořadí.

4.6.4.3 „Zkreslení“ formální gramatiky

Při použití jednoduché gramatiky, jako byla například ta z podkapitoly 4.6.4, dochází k tomu, že se prvním bitem kóduje, jestli je osobou Alice nebo Bob, druhým bitem se kóduje, jestli šifruje nebo dešifruje a poslední bitem, jakým způsobem se kóduje. Jinak řečeno n-tý bit je přímo určen právě jedním terminálem věty gramatiky. To je sice na první pohled hezké, ale z důvodu bezpečnosti

je tato vlastnost velmi špatná. Důmyslný útočník, který by měl přístup k takovémuto šifrovacímu zařízení, by i bez znalosti gramatiky mohl vypozorovat tyto souvislosti a toho využít k prolomení systému.

Je nutno tedy nějakým způsobem zajistit vlastnost, která se používá například při šifrování DES 3.2.1, a to skrýt souvislost mezi daty a tajnou informací. Tato vlastnost se nazývá konfúze a v tomto případě se provádí operacemi expanzí, kontrakcí a permutací gramatiky, konkrétně jejích pravidel, přičemž výsledná gramatika bude ekvivalentní s gramatikou původní.

Expanzí se rozumí přidávání pravidel za pomocí libovolných pravidel, které obsahují alespoň jeden neterminál na pravé straně. Nové pravidlo pak vypadá stejně jen s tím rozdílem, že libovolný neterminál na pravé straně je nahrazen přepisem na základě jiného pravidla, kde se tento neterminál vyskytuje na levé straně. Neterminál je vlastně v tomto pravidle expandován na svůj přepis.

Kontrakce je inverzní operace k expanzi, tedy když je na pravých stranách více pravidel stejný vzor, tak je tento vzor nahrazen novým neterminálem a je přidáno pravidlo, kdy je nově vzniklý neterminál přepsán na daný vzor. Tím se původní pravidla zkracují.

Permutace je proces, při kterém pouze uvažujeme jiné pořadí pravidel a tím používání těchto pravidel bude kódovat tajnou zprávu jinými hodnotami.

4.6.4.4 Bezpečnost a efektivita

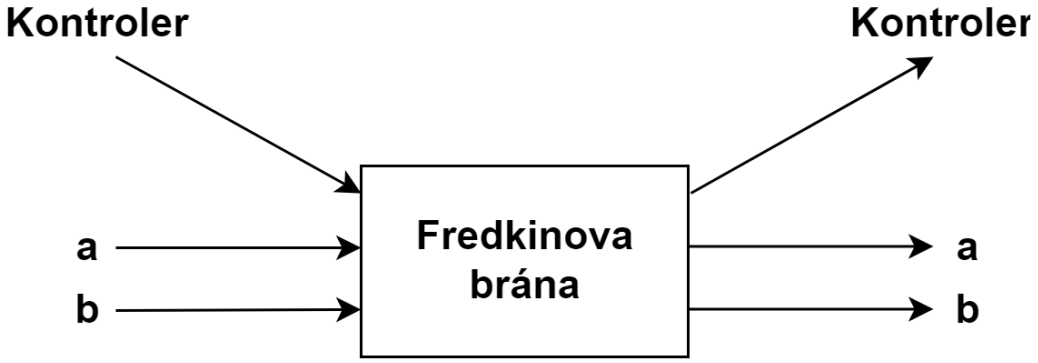
Co se bezpečnosti týče, systémy založené na imitaci formální bezkontextovou gramatikou jsou stejně těžké na prolomení, jako prolomení systému založeném na RSA. To znamená, že tento problém spadá do kategorie NP-úplných a nalezení algoritmu, který by daný problém řešil v polynomickém čase by znamenalo, že by se dal řešit například i RSA problém v polynomickém čase.

Samozřejmě je zde důležité, jako i v případě RSA, mít dostatečně velký klíč, kterým je v tomto případě samotná gramatika a mít zde zanesený princip konfúze.

Další otázkou je efektivita tohoto řešení ve smyslu, jaký je poměr množství vygenerovaného textu ku množství tajné informace. Zde velmi záleží na navržené gramatice. Obecně čím více bude existovat různých pravidel v gramatice, tím více bude možno při procesu odvozování kódovat více bitů tajné informace. Tohle může být zavedeno až do extrémů, jelikož jediná zábrana je pouze představivost. Na druhou stranu poté narůstá velikost této gramatiky (klíče) a navíc to jde také na úkor efektivity rychlosti kdy vytváření nové věty gramatiky s kódováním tajné informace a poté naopak analyzování trvá déle.

Také samozřejmě záleží na velikosti vstupní abecedy gramatiky, kdy pro malou velikost této množiny stačí tyto znaky kódovat méně bity, takže samotný vygenerovaný text zabere méně dat.

Obrázek 6: Fredkinovo hradlo



4.6.5 Použití konceptu zpětného výpočtu a Turingova stroje

Na začátku této sekce je potřeba upozornit, že sekce popisuje princip zpětného výpočtu (Reversible computing), který patří do kategorie tzv. nekonvenčních výpočetních procesů (Unconventional computing). To znamená, že se zde pohybujeme spíše v teoretické rovině.

4.6.5.1 Zpětný výpočet

Hlavní myšlenkou zpětného výpočtu je schopnost nikoli pouze začít výpočetní proces vpřed a poté vrátit výsledek, ale navíc provést výpočet, který začíná se vstupem ve formě výstup dopředného výpočtu a naopak skončit opět s výstupem jako s původním vstupem. Takový zpětný výpočet není možný například při použití operace AND, která při dvou vstupních hodnotách vrátí jednu. Při pokusu invertovat tuto operaci nelze získat z jediné hodnoty ony dvě původní. Například operace negace tuto vlastnost má.

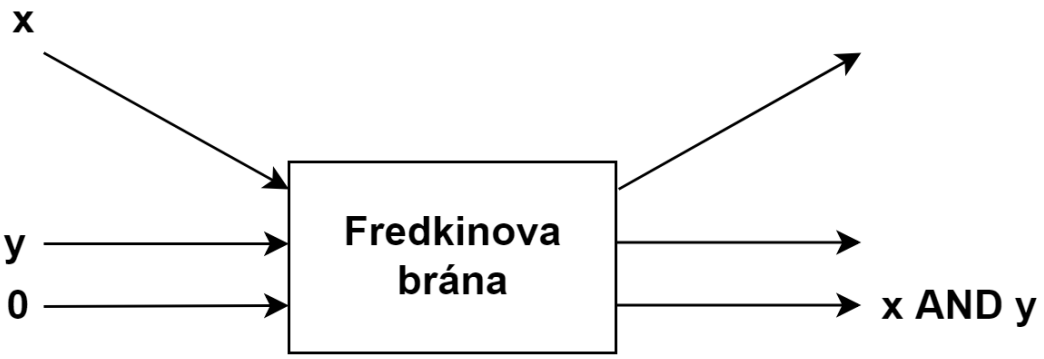
Děje se vlastně to, že se v určitém smyslu ztrácí informace (zvyšuje se entropie), která ale již není potřeba při dopředném výpočtu, avšak při reverzibilním výpočtu je naprosto nezbytná.

Možností, jak tento problém řešit, je použitím tzv. Fredkinových hradel. Toto hradlo má dva běžné vstupy (jako například u AND) a jeden „systémový“ vstup navíc. Tomuto dalšímu vstupu se říká kontroler. Fredkinovo hradlo funguje velmi jednoduše a to tak, že při vstupu hodnoty kontroleru jako 0, jsou oba další vstupy na výstupu stejné a při hodnotě kontroleru 1 jsou hodnoty vstupů na výstupu prohozeny. Schéma Fredkinova hradla je na obrázku 6.

Pomocí takového hradla se dají implementovat klasické binární operace. Například operace AND by vypadala tak, že hodnota kontroleru by byla jedním ze vstupních hodnot ANDu, první běžný vstup by odpovídal druhému vstupu ANDu a druhý běžný vstup by byl nastaven na konstantu 0. Poté by výstupní hodnota ANDu odpovídala druhému běžnému výstupu Fredkinova hradla. Schéma je zobrazeno na obrázku 7.

Podobně se dá implementovat OR a to pouze s tím rozdílem, že druhý běžný parametr bude nastaven na konstantu 1.

Obrázek 7: Fredkinovo hradlo implementující AND



4.6.5.2 Reverzibilní Turingův stroj

Turingův stroj je výpočetní formalismus, který je velmi blízký soudobým počítačům, co se týče výpočetních možností. To je myšleno nikoli z hlediska rychlosti výpočtu, ale z hlediska toho, co je možno tímto strojem vypočítat. Bylo dokázáno, že většina problémů, které řeší Turingův stroj, se vstupem odpovídajícímu kódu Turingova stroje (nebo také neformálně počítačového programu), je nerešitelných[16]. To znamená, že neexistuje Turingův stroj, který by pro všechny přípustné vstupy rozhodl, zda je kód Turingova stroje instancí problému či nikoli. Příkladem může být problém zastavení (*Halt problem*).[17]

Jelikož je o tomto konceptu formálně dokázáno, jak složité je rozhodování, když je instancí právě kód Turingova stroje, dalo by se toho využít pro velmi bezpečný systém k šifrování a dešifrování tajné informace. Při dopředném výpočtu se může ukrýt tajná informace, která může být znovu získána jen schopností provést zpětný výpočet. Hlavním problémem je zde ta skutečnost, že běh Turingova výpočtu není obecně reverzibilní. Je to dáno tím, že Turingův stroj umožňuje definovat například přechody do nového stavu ze dvou různých stavů při stejném čteném symbolu. V takovém případě je zpětný krok nejednoznačný. Případů, při kterých k nejednoznačnosti mezi stavy může dojít je více.

Nicméně bylo dokázáno, že ke každému Turingovu stroji existuje ekvivalentní Turingův stroj, který je reverzibilní. Převod se provede tím způsobem, že ke každému novému stavu, do kterého se můžeme dostat přechodovou funkcí z více stavů, a zároveň by při zpětném průchodu docházelo k nejednoznačnosti, je nutné vytvořit nové stavy. Do těchto stavů poté postupně po jednom přesměrovat přechody z předešlých stavů, kvůli kterým docházelo k nejednoznačnosti. Takto se postupuje pro všechny stavy stroje a nakonec se získá reverzibilní Turingův stroj. Úskalím může být ale to, že v některých případech může počet stavu vzrůst až exponenciálně.

Nakonec takový Turingův stroj může být sestrojený právě tak, aby generoval

[16]Až na triviální problémy, kde je odpověď buď vždy ano nebo ne.

[17]Když se na vstup díváme jako na program, tak se jedná o takový problém, ve kterém je rozhodováno, jestli daný program skončí nebo ne.

bezkontextovou gramatiku, která byla řešena v předchozí podsekci. A ta právě může skrývat onu tajnou informaci.

4.7 Užití ruchu v digitálním souboru

Jak již bylo nastíněno v sekci 4.1.1, digitální soubory (nejčastěji obrázky, hudební soubory, ale také jiné soubory se složitější datovou strukturou) jsou ideálním místem pro skrytí tajné informace. Dá se totiž využít ruchu (nebo také šumu), kterého v takových souborech bývá hodně. Tudíž je možno si dovolit změnit hodnoty určitých bitů, a tím se výsledný obrázek nebo zvuková nahrávka bude jevit pro lidské oko (ucho) jako totožná. Tato metoda steganografie je jednou z nejoblíbenějších a má největší potenciál co se týče použitelnosti, jelikož v dnešním světě internetu je všude spoustu obrázků a hudby, a každý z nich může klidně obsahovat tajné zprávy.

V zásadě je princip fungování této metody docela jednoduchý. U základních formátů bity souboru reprezentují míru intenzity barvy v dané části obrázku (nebo u hudby intenzitu v daném časovém momentu). Soubor obsahující obrázek se tak dá představit jako matice, která ve svých buňkách obsahuje intenzitu barev pixelů jim odpovídajících. U obrázku vzniklý vyfotografováním nejsou hodnoty těchto buněk úplně přesné, jelikož když fotka vzniká, tak se vytváří pouze jakási aproximace obrazu reálného světa, která je dále reprezentována pouze přirozenými čísly v počítači, kterých je samozřejmě konečně mnoho. Dokonce se většinou intenzita jedné barevné složky často reprezentuje pouhým jedním bytem. Nicméně tato aproximace pro lidské oko bohatě postačí. Ke stejnému problému dochází, když se nahrává zvuk z mikrofonu.

Prostor pro tajnou informaci v takovýchto souborech je relativně velký. Za předpokladu, že se využívá jeden byte pro reprezentaci intenzity dané složky barvy a pro každý takový byte, reprezentující intenzitu, se použije jeden bit pro tajnou informaci, dostáváme $\frac{1}{8}$, což je 12,5% souboru. Například pro obrázek, který má rozlišení 3072×2048, což znamená přibližně 18MB místa, máme přibližně 2MB pro tajnou informaci. Text této diplomové práce má přibližně 163 kB, takže by mohl být do takového obrázku zakódován dokonce 12krát! Tento konkrétní přístup je znám jako metoda nejméně významného bitu (Least Significant Bit).

Taková tajná informace se nemusí vyskytovat pouze v uložených souborech jako takových, ale klidně se k tomu může využít „prázdné“ místo na disku. Více sofistikovanější způsob pak může spočívat v tom, že se do obrázku uloží skrytá informaci a tento soubor se poté „vymaže“. Souborový systém pouze dealokuje dané místo paměti, ale daná data zde přetrvají. Samozřejmě nevýhoda tohoto přístupu je ta, že při ukládání nových dat na disk se může daná část paměti alokovat systémem a přepsat.

4.7.1 Metoda nejméně významného bitu (Least Significant Bit; LSB)

Největším představitelem digitální steganografie je Metoda nejméně významného bitu (*Least Significant Bit*; LSB). Ta kóduje tajnou zprávu do rastrového obrázku (zvukové nahrávky) pomocí změny nejméně významného bitu v bytech, které reprezentují barvu daného pixelu v obrázku (intenzitu zvukového impulsu). Jelikož se mění nejméně významný bit barvy, která má velikost jeden byte, je tato změna barvy pouhým okem nepozorovatelná. Navíc statisticky se v polovině případů vlastně ani tento nejméně významný bit nemění jelikož, bit nabývá pouze dvou hodnot. Takže průměrně v polovině případů budou tyto nejméně významné bity obsahovat již správnou hodnotu naší zakódované ukrývané zprávy.

Například u 32-bitového obrázku může barevná složka pixelu nabývat hodnot intenzity od 0 do 255. Průměrné lidské oko není schopno ani zdaleka zaregistrovat změnu této intenzity o 1, což znamená 1 bit pro skrytou zprávu. Změna intenzity daného odstínu barvy se změní o 1/256, což dělá přibližně změnu o 0,39%.[18] To je opravdu málo, aby lidské oko zaregistrovalo rozdíl.

Samozřejmě by bylo možné používat více bitů než pouze ten nejméně významný. V tom případě by čím dál víc docházelo ke zkreslení v obrázku, ale vešla by se do něj delší zpráva. Konkrétně posunutí intenzity barvy maximálně[19] o tři již znamená 2 bity pro tajnou informaci, posunutí o sedm 3 bity a tak dále. Tedy obecně, když se intenzita změní maximálně o $2^n - 1$, tak je dostupných n bitů pro informaci. Z toho plyne, že lineární zvětšování místa pro tajnou informaci vede k exponenciálnímu nárůstu šumu v obrázku. Změna bitů v intenzitě barvy je brána samozřejmě od nejméně významného bitu k více významným.

Na obrázku 8 je konkrétní příklad LSB techniky, která využívá k uložení tajné informace posledních dvou bitů z každé intenzity pixelu. Každý znak skryté zprávy je kódován ve standardním formátu ASCII. Z každé intenzity pixelu jsou tedy brány 2 bity, a jelikož jsou znaky v ASCII kódovány 8 bity, nestačí pouze jediný pixel, ale musí se použít byte reprezentující intenzitu z dalšího pixelu. Pixely se uvažují v horizontálním pořadí.

Když by kódovaná zpráva neobsahovala všechny ASCII znaky a obsahovala by například pouze malá písmena, kterých je 26, bylo by možno znaky kódovat pouze pomocí 5 bitů[20], čímž by vzniklo místo pro delší zprávu.

Dále obrázek 8 obsahuje pro ukázku vpravo nahoře pod šedou šipkou čtvereček bílé barvy, ve které jsou všechny tři intenzity sníženy o hodnotu 3 (nejnižší 2 bity jsou převrácené). Zde je již změna barvy lehce viditelná díky čistě bílému podkladu.[21] Je ale nutno pamatovat, že samotné pixely v běžném obrázku jsou velmi malé, a i kdyby obrázek obsahoval spoustu čistě bílé barvy, tak v průměru

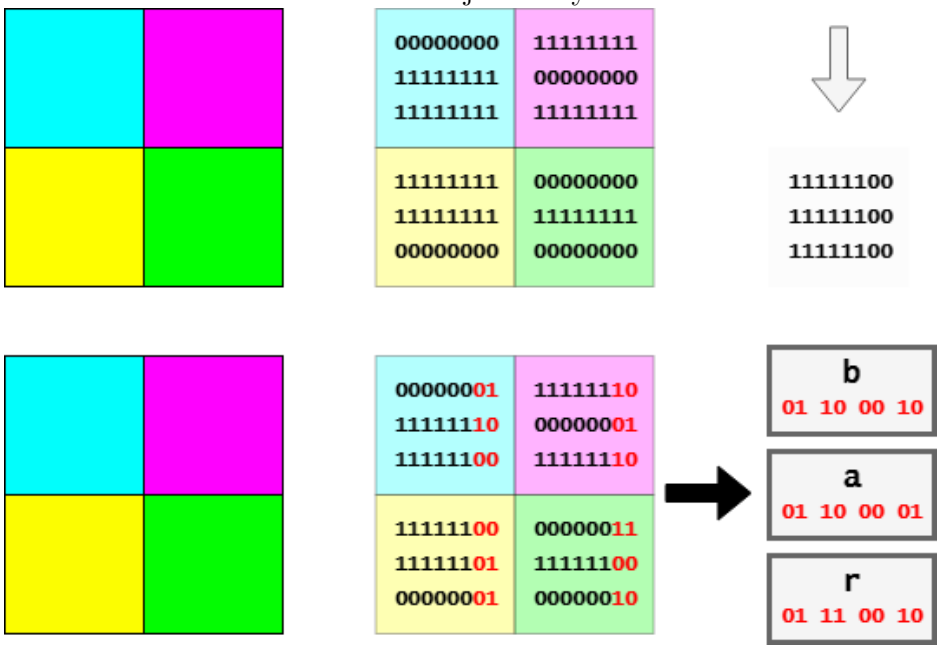
[18] A to ještě navíc v nejhorším případě, kdy se mění právě každý nejméně významný bit.

[19] „Maximálně“ je tu opět právě pro to, že při zakódování zprávy, mohou zůstat některé bity stejné.

[20] Nejmenší vyšší číslo, které je mocninou čísla 2, k číslu 26 je $32 = 2^5$.

[21] Změna je viditelná hlavně v digitální verzi práce na počítači. V tištěné verzi je rozdíl hůře viditelný.

Obrázek 8: Metoda nejméně významného bitu



budou všechny pixely s čistě bílou barvou lehce změněny[22], takže změna barvy bude pouhým okem hůře pozorovatelná.

4.7.2 Úskalí při užívání ruchu k ukrytí tajné informace

Bez pochyby největší překážkou pro ukrývání tajné informace v ruchu digitálních souboru je ztrátová komprese. Jak již bylo popsáno v sekci 4.5, ztrátová komprese modifikuje soubor tím způsobem, že zmenší jeho velikost za cenu toho, že například obrázek, který soubor reprezentuje, se stane do určité míry méně kvalitní, přičemž zpětnou dekompresí můžeme, ale většinou nezískáme původní soubor, který byl před kompresí. Problém je v tom, že po kompresi se změní struktura souboru, ve kterém již není tajná informace čitelná, jako před dekompresí, nebo je dokonce zničena. Navíc po dekompresi jsou v souboru na místech, která jsme užívali pro ukrytí informace, obecně jiná data. Jistým, avšak ne velmi vhodný řešením, je ukládat tajnou informaci nikoli do šumu v souboru, ale do těch částí, které nesou důležité informace, které se nesmí při běhu komprese zničit. Tento způsob pak není z podstaty steganografie příliš vhodný, jelikož hlavním úkolem steganografie je tajnou zprávu co možná nejefektivněji skrýt.

Další komplikace mohou být s obrázky, které mají oblasti obsahující úplně stejné intenzity daných odstínu barev. Většinou to bývají právě „úplně“ černá nebo bílá barva. Příkladem může být odraz Slunce, který na fotce takové oblasti vytváří. Navíc u obrázků, které nejsou vytvářeny jako fotografie, ale vznikly na počítači, se přirozeně častěji vyskytují tyto oblasti stejných barev. To znamená, že tyto obrázky mají tu vlastnost, že v těchto okolních pixelech obsahují všechny

[22]Nebude čistě bílý podklad, jako na tomto papíře.

bity stejné hodnoty. Při použití metody LSB nad takovými obrázky není možné zpozorovat pouhým okem použití steganografie, ale při hlubší analýze nad obrázkem je použití steganografie zjevné. Analýza odhalí, že byl porušen určitý vzor, který by měl v obrázku přirozeně být. Jinak řečeno na těchto obrázcích není šum tak moc velký, aby se dal použít pro ukrytí informace. Proto je metodu LSB nejlepší použít s obrázkem, který nemá tyto oblasti stejných barev, respektive na pozicích nejméně významného bitu se vyskytuje šum a nikoli nějaký vzor.

Na obrázcích 9 a 10 je znázorněn tento problém – obrázek 10 tvoří masku [23] nejméně významných bitů modré intenzity obrázku 9. V místě tekoucí řeky je vidět, že zde není velmi kvalitní šum. Při použití metody LSB by právě tato struktura řeky byla změněna na úplný šum.

Nicméně i u takovýchto obrázků se dá občas tento problém řešit pomocí imitace dat, jak již bylo popsáno v sekci 4.6.3. Dalším řešením může být to, že budeme používat jen malou část nejméně významných bitů k ukrytí informace. Tím se nám prostor pro skrytou informace sice velmi zmenší, ale dokážeme tím ošálit analýzu, která může akceptovat malou odchylku od vzoru.

4.7.3 Souborový formát GIF

Není jednoduše možné obrázek, obsahující tajnou informaci, zkomprimovat do jiného formátu, a zároveň tím zachovat tajnou informaci. Je ale možno vložit tajnou informaci při samé komprimaci obrázku. Tímto problémem se bude zabývat právě tato podkapitola.

Jedním ze souborových formátů, zajišťující bezeztrátovou kompresi, je souborový formát GIF. Existuje více standardů tohoto formátu, ale zde bude uvažován formát, který používá 8-bitové barvy pro reprezentaci pixelů. Podle klasického 24-bitového obrázku je vytvořena meta-tabulka, která obsahuje pouze 256 různých barev, které nejlépe reprezentují originální obrázek. Pro každý pixel originálního obrázku je vybrána jedna z 256-ti barev, která je mu nejblíže. Je nutno uvést, že barvy v této tabulce nejsou obecně nijak uspořádány. To znamená, že dvě barvy, které se liší pouze v nejméně významném bitu, nemusí být vůbec podobné.

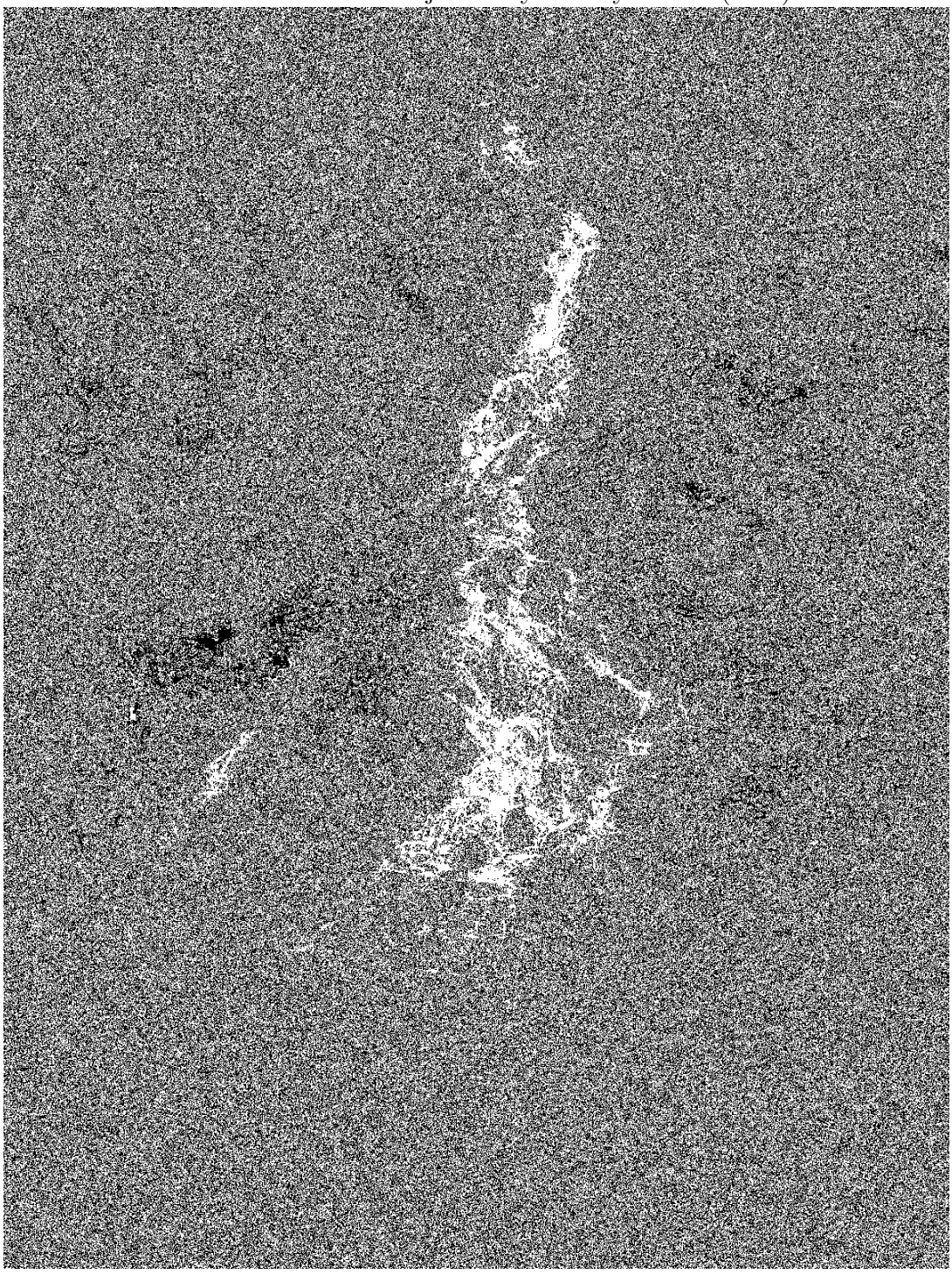
Při snaze zde uložit tajnou informaci se dá postupovat například tak, že se nejprve omezí výběr pouze na 128 barev do meta-tabulky a druhá polovina barev se zvolí tak, že se ke stávajícím barvám vyberou do páru velmi podobné barvy. Navíc se barvy v páru budou lišit vždy v nejméně významném bitu[24]. Mezi barvami v každém páru se nebude uvažovat žádný rozdíl až do toho momentu, kdy budeme chtít ukrýt bit 1 nebo 0 tajné zprávy. To znamená, že při převádění původní barvy z 24-bitového obrázku se vybere jemu nejbližší barva (pár) ve vytvořené meta-tabulce a poté se použije pro reprezentaci ta barva, která má

[23] Více detailu o této masce je uvedeno v sekci 5.3.1.

[24] Z podstaty toho, že hodnoty bytů zde nereprezentují míru zastoupení určité barevné složky, ale fungují pouze jako identifikátor barvy, není nutno používat právě ten nejméně významný bit. Ke kódování informace je možno používat bit na libovolné pozici, ale tato pozice musí být používána pro celou meta-tabulku.

Obrázek 9: Obrázek s oblastí nekvalitního šumu (řeka)

Obrázek 10: Maska nejméně významných bitů (řeka)



na nejméně významném bitu tu hodnotu, která je potřeba. Příklad části takové meta-tabulky by mohl vypadat například jako v tabulce 1.

Tabulka 1: Část meta-tabulky formátu GIF

Binární rep.	Červená složka	Zelená složka	Modrá složka
00000000	62	193	132
00000001	60	192	132
00000010	149	73	37
00000011	150	75	36
00000100	173	241	87
00000101	180	238	79
⋮	⋮	⋮	⋮

Jako při technice LSB byly možné varianty, kdy se použilo více bitů pro tajnou informaci, i zde je možnost toto řešení podobně škálovat. Opět ale dojde k čím dal silnějšímu zkreslování obrázku. To se dá provést tak, že se opět omezíme na ještě méně základních barev. Konkrétně při dalším jednom kroku by to bylo 64 základních barev (tentokrát je to $\frac{1}{4}$ barev z celého množství) a zbytek ($\frac{3}{4}$ barev) je opět podobných k vybraným do čtveřic. Při výběru barvy je vybírána jedna ze 64 čtveřic a poté se použije ta barva ze čtveřice, která je reprezentována tou správnou kombinací bitů, ve kterých se liší od ostatních barev ve čtveřici. Tím jsou na jednu barvu uchovány dva bity tajné informace.

Obecné je tedy možno uchovat m bitů informace v dané barvě, když se vytváří n-tice barev o velikosti $n = 2^m$. Opět jako v případě metody LSB zde dochází k exponenciálnímu nárůstu zkreslení při lineárním navyšováním tajné informace.

4.7.3.1 EzStego

Příkladem software ukrývající tajné zprávy do obrázku ve formátu GIF je EzStego, který je naprogramován v programovacím jazyku Java.

Tajná informace se ukládá opět do nejméně významného bitu barev v meta-tabulce. Co se týče vytváření meta-tabulky, tak EzStego jej řeší tak, že nechá vytvořit všech 256 barev, tak jak to dělá klasický algoritmus GIFu. Dále se však ale snaží o co nejlepší uspořádání v této meta-tabulce. Při zpracování obrázků, které používají jen jednu barevnou hloubku (tzn. jsou černo-bíle), je tento proces třídění triviální. Ale pokud má obrázek tři barevné hloubky, dostáváme se k velmi obtížnému problému, který je na úrovni NP-úplných problémů. EzStego tento problém převádí na známý NP-úplný problém obchodního cestujícího, pro nějž je nutno použít aproximační algoritmus. Bijekce s problémem obchodního cestujícího je ta, že jednotlivé barvy, které mají tři složky, jsou městy v tří-dimenzionálním prostoru a hledá se zde nejkratší cesta skrze všechna tato města (barvy).

To by samo o sobě ještě nestačilo, protože by pro steganoanalýzu bylo velmi podezřelé, že barvy v meta-tabulce jsou právě v tomto „téměř uspořádaném“

pořadí. Proto se před samotným tříděním zapamatuje původní pořadí barev v meta-tabulce a po určení barev všech pixelům, které již obsahují skrytou informaci, jsou řádky v meta-tabulce vráceny do původního stavu. Takový obrázek již může být poslán přes nezabezpečený kanál. Za předpokladu, že příjemce používá tentýž algoritmus jako odesilatel, jsou záznamy v meta-tabulce vytříděny do stejného pořadí a můžou se začíst číst bity tajné zprávy pomocí LSB.

4.7.4 Rozdělení a rozprostření tajné informace

Již sekce 4.4 popisuje myšlenku rozdělení tajné informace do více různých částí, kde poté ke zrekonstruování tajné informace jsou opět potřeba všechny tyto části nebo jen určitá podmnožina.

Při použití techniky LSB pouze nad jedním obrázkem, který se poté uloží na webovou stránku vlastníka, je svým způsobem velmi nebezpečné. Kdokoli, koho by napadlo dešifrovat LSB nad tímto obrázkem, získá skrytou zprávu a navíc si bude jist, že vlastníkem zprávy je právě majitel webové stránky. Rozdělením tajné informace nezvýšíme nikoli pouze míru ukrytí tajné zprávy. Můžeme tím dokonce zajistit i to, že když dojde k odhalení zprávy, útočník nebude mít jistotu, že zpráva je zrovna naše. Zpráva je totiž zrekonstruována v tomto případě z více různých obrázků a kterýkoli majitel libovolného z těchto obrázku může být podezřelý z ukrytí zprávy.

Dá se postupovat tak, že se na internetu náhodně najde nějaká množina obrázků, a provede se operace XOR nad jejich nejméně významnými bity a bity tajné zprávy. Výsledek se poté uloží do nejméně významných bitů našeho obrázku, který se potom může nahrát na vlastní web. Při dešifrování LSB nad naším obrázkem se získá pouze šum. Je zde nutné provést LSB nad všemi použitými obrázky, nad nimi provést XOR a až poté se získá skrytá zpráva. Navíc v tomto případě nikdo nezjistí, kdo je původcem zprávy. Co se může nabízet jako odhalující důkaz může být to, čí obrázek má nejnovější datum vytvoření, ale tato informace se dá kdykoli jednoduše změnit. Nevýhodou této techniky je beze sporu ta skutečnost, že při dešifrování je potřeba všechny (nebo určité množství) obrázky, které ale můžou být z internetu smazány, nebo můžou být modifikovány. Ale dají se také vytipovat takové obrázky, u kterých tato situace hrozí s velmi malou pravděpodobností.

4.7.5 Více kanálů v jednom souboru

Pro vyšší bezpečnost steganografie při použití metody LSB je možné nepoužívat všechny nejméně významné bity každého bytu. Když se použijí jen některé, například podle nějakého pseudo-náhodného vygenerovaného klíče, zlepší se tím velmi ukrytí tajné zprávy. Nevýhodou je, že některé nejméně významné bity zůstanou nevyužité.

Nastupuje zde myšlenka toho, využít takového volného místa pro další zprávy, a tak klidně vytvořit hned více kanálů. To může fungovat klidně i za předpokladu, že do obrázku ukládají svá data různí lidé nekooperativně. To znamená, že se

v určitých místech mohou zprávy překrývat. Tento problém se dá elegantně řešit použitím samoopravných kódů. Tím dojde k dalšímu úbytku místa pro data a navíc také není úplně zaručeno zotavení z kolizí i za použití samoopravného kódu, když chybovost přesáhne určitou mez. Pravděpodobnost, že kolize nastane je velmi závislá na tom, kolik kanálů v souboru existuje a kolik chyb je schopný samoopravný kód opravit. Samozřejmě s určitou pravděpodobností může k neopravitelným kolizím dojít kdykoli.

K mnohem vyšší efektivitě dojde, když lidé, vytvářející kanály, spolupracují (nebo prostě jde jen o jednoho člověka). Je možno zde organizovaně rozdělit prostor pro zprávy, takže není vůbec potřeba samoopravný kód. Pro vytvoření n kanálů se rozdělí celý soubor do bloků bytů velikosti n. Každý takový byte v jednom bloku obsahuje jeden nejméně významný bit, a ten je přiřazen jednomu z kanálů. Kanálům jsou bity rozdělovány buď postupně podle určitého uspořádání kanálů, nebo může být u každého bloku vytvořena permutace a bity jsou přiřazeny podle této permutace, což sice zvyšuje výpočetní režii, ale zvyšuje sílu steganografie. K dosažení ještě vyššího ukrytí zpráv, ale opět snížení výkonu, není nutné uvažovat rozdělení souboru přímo po blocích velikosti n, ve kterých každý z kanálů má právě jeden bit, ale toto rozdělení provést rovnoměrně mezi kanály napříč celým souborem.

4.7.6 Souborový formát JPEG

Ukrýt skrytou zprávu přímo do souboru ve formátu JPEG je velmi náročné. Nicméně se tento formát dá využít k nalezení šumu v obrázku, a ten pak následně použít pro ukrývání informace variantou metody LSB, kdy se bude v každém bytu užívat různý počet nejméně významných bitů.

4.7.6.1 Využití formátu JPEG k efektivnější steganografii

Komprese obrázku do souboru JPEG je ztrátová. To znamená, že při kompresi obrázku a následně jeho zpětné dekompresi nedostaneme přesně stejný obrázek (alespoň tedy v drtivé většině případů), ale obrázek velmi podobný. Takový dekomprimovaný obrázek můžeme porovnat s naším originálním obrázek, čímž zjistíme, které bity algoritmus považoval za nevýznamné a při komprimaci je „znehodnotil“. Právě tyto bity se dají následně použít pro ukrytí tajné zprávy, a zároveň nedojde k většímu zhoršení kvality obrázky, než při kompresi JPEGem.

Otázkou zbývá, jak zjistit, které bity jsou právě ty, které se dají využít pro steganografii. Při porovnávání bitů odpovídajících bytů mezi originálním a zkomprimovaným-dekomprimovaným obrázkem, procházíme bity od nejvíce významného po méně významný. Když narazíme na bit ve kterém se liší, je jasné, že algoritmus počínaje tímto bitem, považoval všechny zbylé za zbytečné (jsou méně významné). Právě všechny takové bity se pak můžou využít pro ukrytí informace. Je ale také nutné poukázat na to, že i předchozí bit, ač má třeba stejnou hodnotu, jako odpovídající bit originálního obrázku, mohl být považován za zbytečný, ale to se nedozvíme. Jde o to, že bit může nabývat pouze dvou hodnot

a my si se 100% pravděpodobností můžeme být jisti, že byl bit považován za zbytečný, když se odpovídající dvojice bitů liší, ale pouze s 50% pravděpodobností si můžeme být jisti, že bit je považován za zbytečný (nebo za důležitý), když odpovídající dvojice bitů má stejnou hodnotu. Proto bity, u kterých si nejsme jisti, nebudeme používat k ukrytí informace.

Tato metoda je velmi efektivní z hlediska poměru množství ukryté informace ku míře zkreslení obrázku. Na druhou stranu je nutno, aby příjemce takového obrázku se skrytou informací měl ještě navíc i originální obrázek, aby následně pomocí algoritmu JPEG zjistil, ve kterých bitech je ukryta tajná zpráva.

4.7.6.2 Metoda JSteg

Jak již bylo popsáno, ukrývání tajné informace v souborech JPEG vzhledem k jeho ztrátovým vlastnostem je velmi náročné, ale existují určité metody, které jsou toho schopny. Příkladem toho mohou být algoritmy JSteg a F4, které jsou dále popsány.

Historicky první metoda se nazývá JSteg. Jednou z částí algoritmu JPEG je fáze, kdy se obrázek rozdělí do bloků po 56-ti hodnotách (8×8) pro každou barevnou složku zvlášť. Nad těmito bloky je poté zvlášť provedena diskrétní kosinova transformace typu 2. Touto transformací jsou hodnoty reprezentující intenzitu dané barvy převedeny na hodnoty frekvencí kosinových funkcí, které se použijí jako koeficienty funkcí aproximující daná data.[25] Tyto vypočítané koeficienty se musí zaokrouhlovat, což je právě to, kam tato metoda ukrývá skrytá data. Pokud se nepoužijí standardní pravidla pro zaokrouhlování, ale budeme si zaokrouhlovat jak se nám to hodí, jsme tímto schopni do těchto koeficientů zakódovat tajnou informaci. To můžeme například tak, že když zaokrouhlíme hodnotu dolů, tak je tím reprezentovaný bit 0 a při zaokrouhlení nahoru bit 1. Detailnější popis kosinovy transformace je uveden v [2, 7].

Tato metoda může mít samozřejmě negativní vliv na výsledný obrázek z důvodu nepřesného zaokrouhlování, zejména když se vyskytují nízké koeficienty ve velkém množství, což je právě případ kosinovy transformace. Nicméně oproti metodám klasického LSB, ať už v 32-bitovém obrázku nebo GIFu, je steganoanalýza náročnější.

4.7.6.3 Metoda F4

Metoda JSteg má nedostatky v tom, že z důvodu „náhodného“ zaokrouhlování koeficientů dochází ke zkreslení křivky, která má reprezentovat výslednou kosinovou funkci. To narušuje statistický profil metody, což se dá při podrobné steganoanalýze objevit a je možno zjistit, že s koeficienty bylo manipulováno. Přichází tedy nová generace této metody a to algoritmus F4.

[25] Některé hodnoty frekvencí mohou být velmi nízké a tím pádem velmi málo ovlivňují výsledek, proto mohou být tyto frekvence vypuštěny, což je jedna z kompresních a také ztrátových vlastností, ale nikoli tou nejvýznamnější.

Hlavní charakteristika kosinovy transformace je ta, že při ní vznikají nižší koeficienty ve vyšší míře než ty vyšší. Při použití metody JSteg je zasahováno do zaokrouhlování této statistiky. Je možno metodu vylepšit tím způsobem, aby bylo zachováno pravidlo, že nižších koeficientů je statisticky více než těch vyšších.

Algoritmus F4 přichází s tou myšlenkou, že místo „podvádějícího“ zaokrouhlování dojde ke klasickému zaokrouhlení. Výsledná hodnota se buďto sníží nebo zůstane stejná na základě parity koeficientu a hodnotě bitu, který chceme zakódovat. Konkrétně algoritmus funguje tak, že:

- Když má koeficient hodnotu 0, tak se nepoužije pro kódování.
- Když je koeficient lichý, tak při zakódování bitu 1 koeficient zůstane a při zakódování bitu 0 se koeficient sníží o hodnotu 1.
- Když je koeficient sudý, tak při zakódování bitu 0 koeficient zůstane a při zakódování bitu 1 se koeficient sníží o hodnotu 1.

Koeficientů s hodnotou 0 je zde sice statisticky nejvíce, ale v celkovém poměru nezaujímají nijak obrovskou část.

Tento algoritmus sice také nezachová přesně stejný statistický profil, ale vede si mnohem lépe než algoritmus JSteg.

Další verze tohoto algoritmu známá jako F5 je vylepšena o takzvané maticové kódování, které přináší ještě nižší míru zkreslení. Jak metoda JSteg, tak metoda F4 a F5, jsou detailněji popsané v [1, 2].

4.8 Klíče

Tak jako drtivá většina kryptografických algoritmů používá pro své šifrování a dešifrování klíče založené na různém principu (symetrické, asymetrické), stejně tak mohou být použity klíče i ve steganografických algoritmech. Nicméně u steganografických šifer nejsou až tak důležité, jelikož klíče nenapomohou až tak markantně k vyššímu ukrytí komunikace. U kryptografických šifer je ale klíč hlavní bezpečnostní prvek, bez kterého by se tyto šifry neobešly.

Navíc je potřeba aby obě komunikující strany měly stejný klíč (symetrický) nebo aby odesilatel šifroval veřejným klíčem (asymetrickým). S tím může být problém, jelikož hlavním účelem steganografie je to, aby komunikace nevzbudila podezření. Když je nutno si před zahájením komunikace nějakým způsobem vyměňovat či generovat klíče, již jen to může být podezřelé. Nicméně mohou existovat případy, kdy dopředu vygenerování nebo vyměnění klíčů před samotným zahájením komunikace nemusí být problém.

Kromě šifrování symetrickými a asymetrickými klíči, které jsou detailněji popsané v sekcích 3.2.1 a 3.2.2, je možné ve steganografii použít ještě takzvanou techniku *zero-knowledge proof*, která je opět popsaná v sekci 3.3. *Zero-knowledge proof* je způsob, který se užívá spíše u vodoznaků sloužící pro copyright a to tak, že autor díla je schopen dokázat, že dílo obsahuje ochranou známku, přičemž její

umístění nebo obsah není nucen odhalit. Kdyby byla tato pozice v díle a obsah odhalen, tak by to mohli útočníci zneužít pro vytvoření nelegálních kopií.

Na konec se ještě zbývá zmínit o technice, která zabraňuje vytváření nelegálních kopií pomocí slučování vodoznaků. Takový vodoznak může v souboru vypadat například v jedné kopii souboru jako 010111 a v druhém souboru 010100, kde se tyto dvě kopie liší právě jen v posledních dvou bitech vodoznaku. Útočník může vytvořit novou kopii tím způsobem, že všechny identické bity obou kopií zanechá a jen v těch bitech, které se liší, zvolí nějakou kombinaci bitů, která neodpovídá vodoznaku ze žádné z originálních kopií. Tímto způsobem může velmi pravděpodobně vzniknout nová legitimní kopie s vodoznakem, například 010110.

Tomuto útoku se dá jednoduše zabránit využitím samoopravných kódů. Takový samoopravný kód může vypadat například tak, že jediné povolené vodoznaky délky n budou obsahovat jediný bit s hodnotou 1. To znamená, že pro případ $n = 3$ jsou to vodoznaky 100, 010 a 001. Kdyby si útočník počínal podobným způsobem, tak mu například při slučování kopií s vodoznaky 100 a 001 může vzniknout kopie s vodoznakem 101, která je ovšem neplatná. Navíc se dá podle analýzy tohoto vodoznaku odhalit, že tato nelegitimní kopie vznikla z první a třetí legitimní kopie za předpokladu, že si útočník počínal popsaným způsobem. Tato metoda se nazývá *collision control codes*.

4.9 Změna pořadí

Mnoho objektů může mít svá data nějakým způsobem uspořádaná, přičemž na tomto pořadí ani zas tak nezáleží a nikomu přeuspořádání nemusí přijít nijak podezřelé. Například určitý textový dokument může obsahovat kapitoly, odstavce a věty, které na sobě (např. chronologicky) nijak nezávisí, a proto mohou být určitým způsobem zaměněny. Právě tímto přeuspořádáním je možno ukrýt informaci do takového dokumentu.

Když existuje n různých prvků, dá se nad nimi uvažovat $n!$ různých pořadí (tzv. permutace prvků). Každému z tohoto pořadí se dá přiřadit různá bitová reprezentace (dají se očíslovat), a proto je díky těmto $n!$ různým pořadím možno zakódovat $log_2 n!$ bitů tajné informace. To znamená, že s rostoucím počtem prvků v tomto seznamu velmi narůstá místo pro ukrytou informaci.[26]

Pro to, aby bylo možno do pořadí prvků zakódovat tajnou informaci, musí samozřejmě být na této množině prvku zavedena relace uspořádání. V případě slov, nebo kapitol dokumentu je to velmi jednoduché a intuitivní – zvolí se prosté abecední pořadí. Nicméně pro jiné objekty, kde již takové implicitní uspořádání nelze zvolit, je potřeba nějaká funkce s definičním oborem odpovídajícímu prvkům seznamu, která vrací jeho pořadí (pozici). Tato funkce slouží vlastně jako klíč a není vůbec špatné takovou funkci použít i v případě, kde je pořadí intuitivní z důvodu většího zabezpečení.

[26]Když se v seznamu použijí dva nebo tři prvky, tak se ukryje méně informace, než je počet prvků seznamu, ale při delších seznamech to již neplatí. Například pro seznam velikosti 9 je možno kódovat dvakrát více informace.

Kromě samotného zakódování tajné informace do specifického pořadí částí daného objektu je možné také zajistit vyšší bezpečnost před útokem při klasickém ukládání tajné informace přímo do dat, například do nejméně významných bitů v obrázku. Klasická metoda funguje tak, že první tři bity tajné informace jsou uloženy v nejméně významných bitech třech barevných složek prvního pixelu obrázku, další tři bity jsou uloženy v druhém (sousedícím) pixelu a tak dále. Záměnou pořadí je možné tajnou zprávu nekódovat do pixelů obrázku sekvenčně, ale v libovolném pořadí.

Při zakódování tajné informace se postupuje tak, že se objekt (vstupní data) rozdělí na stejně velké části. Právě nad tímto rozdělením se bude uvažovat určité pořadí. Každá z těchto částí se pak rozdělí na další dvě. První část zůstane neměnná (zde se nebude kódovat tajná informace) a v druhé části se bude ukládat tajná informace, která se bude měnit (například nejméně významné bity pixelů). Je zásadní, aby neměnné části dat byly natolik velké, aby byly různé, jelikož by zde nebylo možno zavést jednoznačné uspořádání. Nicméně kdyby došlo k tomu, že jsou některé z neměnných částí stejné, tak poté také musí nést tutéž tajnou informaci (proměnlivou část) a budou mít tedy tutéž pozici. Tudíž v případě obrázkové bitmapy by nebylo ideální rozdělit obrázek po pixelech, jelikož bude jistě existovat v obrázku mnoho pixelů identické barvy. Je tedy nutno zvolit hrubší rozdělení, kde konkrétní variace pixelů nebudou totožné s ostatními variacemi.

5 Steganoanalýza

Již zde byla popsána steganografie, jakož to vědní oblast, která se snaží vynalézt techniky, které skrývají proces komunikace mezi dvěma komunikujícími stranami. Samozřejmě také existuje věda, která se snaží proti této praktice bojovat, a ta se nazývá steganoanalýza (steganalysis). V této kapitole bylo čerpáno bylo z [1, 2].

Opět jak tomu bylo s historickými příklady u „nedigitální“ steganografie, má steganoanalýza v historii také své zastoupení. Například dokumenty, které potenciálně nesly schovanou zprávu, byly osvětlovány UV světlem, nebo opticky zvětšovány a důkladně zkoumány. Dále také metody, kdy za doby druhé světové války ostraha ve vězení dávala vězňům speciální papír, který byl určen pro psaní, a který měl tu vlastnost, že s ním nešly provádět techniky s neviditelným inkoustem, jak bylo popisováno v kapitole 2. Takové metody prevence, kdy se již dopředu zabraňuje potenciální steganografii, patří do skupiny tzv. pasivní steganoanalýzy.

Tak jako steganografie, je i steganoanalýza velmi rozšířená hlavně v digitálních formách. Je to vědní oblast která nad objekty, které jsou podezřelé pro ukrývání tajné informace, zkouší detekovat různé známe steganografické šifry nebo také provádět různé statistické analýzy. Tím je tato oblast velice časově, ale nejen časově, náročná.

Nejjednodušší analýza může být prostě jen porovnání dvou stejně vypadajících souborů (například obrázků), kde o jednom víme, že je originál a neobsahuje skrytou zprávu. Taková analýza dokáže pouze zjistit, že soubor pravděpodobně obsahuje skrytou zprávu, ale obsah samotné zprávy se nezjistí. Další běžné analýzy se provádí různými statistickými metodami. Samozřejmě soubory, které obsahují velkou míru komprese, jsou pro analýzu náročnější.

5.1 Obtížnost odhalení steganografických šifer

Ve steganografii je poměrně náročné zkoumání toho, jak moc je daná steganografická šifra silná. Je těžké odpovědět na otázku, jak moc musí být informace skryta, aby nešla najít. Oproti kryptografii, steganografie postrádá jasně definovaný model, na kterém by se dalo zkoumat, jak moc je která šifra silná.

Jedním způsobem, jak měřit sílu steganografické šifry, může být ten, že se uvažuje co možná největší množství steganoanalytických útoků na tuto šifru a pozoruje se, kolik z nich bylo úspěšných. Útok na steganografické šifry je velmi podobný, jako útok na kryptografické šifry.

5.2 Typy útoků na steganografii

Útoky se dělí podle toho, co všechno má útočník dostupné k provedení útoku, nebo o co mu vlastně při útoku jde.

- Pouze zašifrovaný soubor – V tomto případě se snaží útočník rozlišit, zda je v daném souboru skrytá informace, či nikoli. U tohoto typu se využí-

vají hlavně statistické analýzy pro odhalení skryté informace. Stejně jako v podobném případě v kryptografii jde spíše o „alchymii“, jelikož neexistuje přímo osvědčený způsob, jak by měl algoritmus soubor zkoumat, protože ten, kdo ukrývá zprávu, může statistické vlastnosti souboru upravit.

- Zašifrovaný soubor a originál – Může se stát, že útočník má kromě souboru, který nese skrytou informaci, ještě i originální soubor bez skryté informace. Zde je samozřejmě triviální odhalení toho, zda soubor nese extra informaci, či nikoli, jednoduchým porovnáním souborů. Tudíž nároky jsou zde větší. Útočník se snaží buď ukrytou informaci pouze odhalit, odstranit, nebo ji dokonce nahradit svou vlastní informací.

- Několik stejně zašifrovaných souborů – Podobné jako u minulého případu, jen s tím rozdílem, že souborů se skrytou informací může být více. Tyto soubory jsou stejné až na informaci, kterou ukrývají a předpokládá se použití stejného algoritmu pro ukrytí zprávy. Takové kopie souboru mohou být například různé kopie zvukového souboru s různými vodoznaky pro autentizaci. Účel útoku je opět tyto většinou restriktivní informace vymazat, nebo je nahradit vlastními takovým způsobem, aby zvukový soubor bylo možno přehrát. Často se to provádí tak, že se vytvoří jeden nový soubor, který obsahuje zkombinované části informací z ostatních souborů.

- Zašifrovaný soubor a algoritmus – Útočník má k dispozici zašifrovaný soubor a algoritmus, pomocí kterého do něj byla zašifrována informace. Tímto útokem projdou úspěšně už jen opravdu sofistikované algoritmy, u kterých je nutno držet nějakou část algoritmu nebo nějaké hodnoty v tajnosti. Tento tajný prvek se nazývá klíč a je nutný k zašifrování a také k dešifrování souboru.

- Náhodné malé změny – Útok může vypadat i tím způsobem, že se útočník nesnaží vyloženě zjistit, jestli daný soubor obsahuje nějakou skrytou informaci, ale preventivně do všech souborů přidává malé změny. Tím docílí toho, že za předpokladu, že soubor obsahuje nějakou skrytou informaci, tak je určitá šance, že informace bude poškozena. Nicméně některé steganografické algoritmy jsou vůči tomuto útoku imunní díky tomu, že využívají opravy kódu při získání informace ze zprávy.

- Změna formátu – Existuje mnoho různých druhů formátu, jak ukládat soubory. Už jen obrázkové soubory jich mají mnoho, a každý z těchto formátů ukládá hodnoty barevných hloubek pixelů trošku jinak. To vede často ke zničení skryté informace, ale i přes to existují sofistikované algoritmy, které si s určitými typy přeformátování dokáží poradit.

- Komprese – Konkrétně jde o kompresi, která je ztrátová a po dekompresi souboru je často zpráva zničená. Tento případ úzce souvisí s předchozím a stejně jako v předchozím případě existují algoritmy které si s kompresí určitých druhů formátů ví rady.

Poslední tři body jsou metodou tzv. aktivní steganoanalýzy – útočník se snaží preventivně dělat malé změny do všech souborů, které by mohly potenciálně nést skrytou informaci, přičemž nemusí jít ani o soubory ukrývající tajemství. V opačném případě, když útočník nezasahuje do komunikace a pouze komunikaci monitoruje, jde o tzv. pasivní steganoanalýzu. V případě že jde o pasivní steganoanalýzu a útočníkovi se podaří odhalit tajnou komunikaci, může odpovídající komunikační kanál přerušit, nebo se uchýlit opět k aktivnímu přístupu, kdy může drobnými změnami tajnou informaci ničit. Jestliže je útočník schopen dokonce i zjistit obsah tajné informace, může tajné informace dále odposlouchávat a toho následně zneužít. Poslední způsob útočníka je tzv. zlomyslný (*malicious*)[2].

5.3 Přístupy k odhalení steganografie

Tak jako je mnoho způsobů, jak steganografií kódovat tajnou informaci do objektů, musí existovat i mnoho přístupů, jakým způsobem odhalit tajnou komunikaci. Mezi hlavní zastupitele patří nejméně náročná analýza a to pouhým okem/uchem, avšak za pomoci již nějakého předzpracování. Dále také strukturální analýza, kde, jak již název napovídá, velká pozornost je zde věnována struktuře formátu objektu, do kterého může být tajná informace zakódována. A poslední uvedená metoda je asi nejdůležitější, a to statistická analýza, při které se využívá určitých charakteristických statistik, které objekty obsahují.

5.3.1 Vizuální (poslechová) analýza

Tento způsob spočívá v tom, že je z obecně velmi složitého objektu, například obrázku, odstraněna velká část dat a analýza se dále soustředí pouze jen na ten daný zbytek, kde je již zpozorovatelné podezření pouhý okem/uchem.

V případě černo-bílé bitmapy to může být například to, že se jednotlivé pixely budou reprezentovat pouze nejméně významným bitem a to tím způsobem, že jestliže je nejméně významný bit 1, tak pixel bude úplně černý a jestliže má hodnotu 0, tak pixel bude úplně bílý. Příkladem je obrázek 10 ze sekce 4.7.2. Většina obrázků pořízena jako snímek z fotoaparátu velmi často obsahuje místa totožné barvy, a tím pádem by po podobném testu mohla jít tato stejná místa vidět. Ale v případě, že všechny nejméně významné bity pixelů byly použity k uchování tajné informace, jeví se výsledek z testu jako šum, což je podezřelé. Samozřejmě také velice záleží na obrázku, který byl k ukrytí tajné informace použit. Ten již může být vytvořený tak, že neobsahuje oblasti stejných barev, a proto by nebylo adekvátní takový obrázek podezřívat.

Podobný útok by mohl být použit nad libovolnými jinými objekty, například u zvukových nahrávek, a tudíž by se jednalo o poslechový útok. V tomto případě by se opět ze zvukové nahrávky odstranila určitá část zvuku a člověk s hudebním, nebo dokonce absolutním, sluchem by takto mohl objevit podezřelou zvukovou nahrávku.

5.3.2 Strukturální analýza

Steganografické metody v mnoha případech vytvářejí určitou strukturovou charakteristiku v ukrývaném objektu. To se dá přirovnat například k charakteristickému podpisu každého člověka, který je jedinečný.

Jako příklad může posloužit obrázkový formátu GIF, jehož rozbor a steganografické použití je rozebráno v sekci 4.7.3. Pro zopakování, tento formát používá v záhlaví souboru meta-tabulku pro definici barev v obrázku včetně jejich bitových reprezentací. Tyto barvy nejsou obecně v meta-tabulce nijak seřazené, a proto v nich není tak jednoduché provést steganografickou metodu nejméně významného bitu, jelikož změna jediného bitu barvy může vést k obecně úplně jiné barvě. Avšak je zde možnost změnit pořadí barev v meta-tabulce, aby bylo možno informaci uchovávat do bitů barev, čímž je vlastně manipulováno se strukturou tohoto formátu. V takto přestrukturovaném souboru je jednoduché nalézt podezření, avšak steganografický algoritmus může být opatřen ještě o přeuspořádání tohoto pořadí barev a znovu uspořádání je možno jen za pomoci nějakého klíče nebo dalšího specifického algoritmu.

5.3.3 Statistická analýza

Velmi často objekty obsahují různé vzory a tím mají i určité statistické vlastnosti. Opět vhodným příkladem je zde, jako v podkapitole 5.3.1 o vizuální analýze, metoda nejméně významných bitů v obrázku. Pokud obrázek obsahuje určité oblasti stejné barvy, tak se tento fakt obvykle promítne i ve statistických četnostech hodnot jednotlivých bitů. Avšak když jsou nejméně významné bity nahrazeny tajnou informací stávají se více náhodnými, což je pro statistickou analýzu velké varování, že zde mohlo dojít k manipulaci s bity. Statistická analýza v tomto případě může být mnohem efektivnější než pouhá vizuální analýza, protože při malém zkreslení lidské oko nemusí zaregistrovat nic podezřelého. Analýza již může o neobvyklých hodnotách varovat, avšak může být náročnější na implementaci.

Dalším příkladem, kde již je vizuální analýza krátká, je analýza obrázku ve formátu JPEG. Opět je tato problematika detailněji popsána v sekci 4.7.6. Ke kódování tajné informace se používá zaokrouhlování koeficientu figurujících v rámci kosinovy transformace, která je součástí algoritmu JPEG. V případě nepoužití žádné steganografické metody mají tyto koeficienty tu vlastnost, že nižších koeficientů je vždy alespoň dvakrát více než těch vyšších. A právě při použití steganografie dochází ke zkreslení této statistiky a statistická analýza tento problém odhalí.

Klasickou a jednoduchou statistickou metodou je metoda χ^2 (chí-kvadrát), která se také nazývá jako test nezávislosti. Tento test určuje jakousi míru náhodnosti výskytu určitých událostí (jevů). Vzorec χ^2:

$$\chi^2 = \sum \frac{(e_i - E(e_i))^2}{E(e_i))},$$

kde:

- e_i ... uvažovaná události i, která může nastat
- $E(e_i)$ ... očekávaná hodnota i-té události

V případě, kdy se zajímáme o statistické rozložení nejméně významných bitů, je množina událostí velmi jednoduchá – jedna z událostí je ta, že hodnotou bitu je hodnota 1 a druhá událost je, že hodnota bitu je 0. Tímto způsobem se prochází co možná nejvíce bitů, aby byla statistika co nejpřesnější. Na druhou stranu může být také velmi užitečné postupně aplikovat χ^2 pouze na určité menší oblasti.

Výsledná hodnota χ^2 značí, jak velká nezávislost je mezi událostmi – čím je hodnota vyšší, tím více událostí nastává předvídatelně. V našem případě dominuje jedna hodnota bitu nad druhou, což je velmi charakteristické pro obrázky. Na druhou stranu nízká hodnota znamená, že hodnoty bitů se vyskytují se stejnou nebo velmi podobnou pravděpodobností, a proto je zde podezření, že byla použita nějaká steganografická metoda.

O něco důmyslněji definovaný χ^2 může obsahovat více událostí pro specifičtější statistické charakteristiky. Například nemusí být zkoumán pouze poměr výskytu bitů 1 a 0, ale také může být zkoumána závislost mezi hodnotami sousedících bitů. Taková množina událostí může vypadat například:

Událost	Bit	Následující bit
e_0	0	0
e_1	0	1
e_2	1	0
e_3	1	1

Následující test bude mít zvýšenou výslednou hodnotu i v případě, kdy poměr výskytu hodnot bitů bude sice statisticky v pořádku, ale statistika sousedících bitů v pořádku nebude. U obrázků je totiž velmi časté, že sousedící pixely mají stejnou hodnotu nejméně významného bitu.

6 Programátorská dokumentace

Aplikace je naprogramována v jazyce C# v prostředí Microsoft Visual Studio Community 2017. Je implementována jako okenní WPF (Windows Presentation Foundation) aplikace, která kombinuje XAML (Extensible Application Markup Language) kód s již zmíněným kódem v C#.

6.1 Microsoft Ribbon

V nejnovějších verzích programů od firmy Microsoft se v hojné míře vyskytuje ovládací prvek tzv. pás karet (ribbon), který poskytuje pohodlnou práci s aplikací pomocí pásu přepínacích karet, které obsahují vzájemně související funkce. Tento prvek je používán ve verzích Microsoft Office 2007 a vyšších a je také použit v průzkumníku souborů Windows v operačních systémech Windows 8 a 10. [12] Tato funkcionalita je šířena pomocí knihovny .dll (System.Windows.Controls.Ribbon).

Aplikace této diplomové práce využívá tento ovládací prvek.

6.2 Metoda LSB Swap

Šifra LSB Swap nejprve převede textovou zprávu do proudu bitů a následně ji šifruje tak, že v určitých případech zamění pořadí dvou nejméně významných bitů v bytech téže barevné složky sousedících pixelů v bitmapě. Pro zakódování bitu tajné informace s hodnotou 1 se zvolí například dvojice sousedících bitů v pořadí 10 a pro zakódování bitu s hodnotou 0 se zvolí dvojice 01[27]. Pokud jsou v obrázku dva sousedící bity s hodnotami 11 nebo 00, tak zde samozřejmě žádná informace zakódovat nelze. Konkrétní definice chování, které je implementováno v aplikaci, je následující:

Když se narazí na dva nejméně významné bity sousedících pixelů dané barevné složky, které mají různou hodnotu, tak:

- Při kódování bitu tajné zprávy s hodnotou 1, kdy aktuální hodnoty nejméně významných bitů jsou popořadě 1 a 0, se neprovede žádná záměna.
- Při kódování bitu tajné zprávy s hodnotou 1, kdy aktuální hodnoty nejméně významných bitů jsou popořadě 0 a 1, se prohodí hodnoty těchto bitů.
- Při kódování bitu tajné zprávy s hodnotou 0, kdy aktuální hodnoty nejméně významných bitů jsou popořadě 1 a 0, se prohodí hodnoty těchto bitů.
- Při kódování bitu tajné zprávy s hodnotou 0, kdy aktuální hodnoty nejméně významných bitů jsou popořadě 0 a 1, se neprovede žádná záměna.

Dekódování zprávy je potom ještě jednodušší. Prochází se nejméně významné bity stejné barevné složky sousedních pixelů po dvojicích a pokud jsou různé,

[27] Duálně lze samozřejmě hodnoty pro oba případy prohodit.

tak se vezme první z nich jako součást tajné zprávy. Když jsou stejné, tak se pokračuje bez obdržení tajného bitu.

Tento přístup je velmi vhodný hlavně ze dvou důvodů. Zaprvé je zachován poměr četností obou hodnot bitu, který odpovídá originálnímu obrázku, proto je tento přístup nedetekovatelný pro běžnou statistickou analýzu.[28] Zadruhé, jelikož tato metoda přeskakuje místa stejných nejméně významných hodnot bitů dané intenzity barvy v sousedících pixelech, oblasti se stejnými hodnotami nejméně významného bitu, jsou tímto zachovány téměř ve stejném stavu. To znamená, že pro vizuální analýzu s předzpracováním je tento přístup velmi náročné odhalit.[29]

Na druhou stranu má tento přístup v průměru pouze čtvrtinovou kapacitu pro uložení tajné informace[30] než klasická metoda nejméně významného bitu. To je dáno tím, že dvojic různých sousedících bitů je v průměru polovina, a právě pouze ty kódují tajnou informaci. Dvojice stejných hodnot bitů se úplně přeskakují. Z důvodu toho, že bity nabývají pouze dvou hodnot, není možné pro kódování pořadí používat delší sekvence bitů, které by vedly k mnohem většímu množství zakódované informace.

Otázkou ale zůstává, jak při dešifrování poznat, kde zašifrovaná zpráva končí a kde začíná šum. Jednou z možností by mohlo být toto rozeznávání ignorovat a zobrazit zprávu s následným šumem, což ale není velmi hezké a praktické. Další možností by bylo vymyslet speciální řetězec jako oddělovač zprávy a šumu. V tomto případě by ale vstupní zpráva nemohla obsahovat tento řetězec, jinak by se zpráva zakódovala jen jako její prefix po tento oddělovací řetězec. Aplikace tento problém řeší tak, že do prvních 32 nejméně významných bitů uloží v binární podobě celé číslo, které označuje, kolik bytů zpráva obsahuje.[31] To sice narušuje statistický profil dané metody, ale nepatrně.

Tato šifra je založena na neznalosti principu jejího utajení (tzv. *security through obscurity*). Nicméně bylo by ji možno rozšířit o šifrování s klíčem tak, že by hodnoty tohoto klíče určitým způsobem měnily pravidla při jejím šifrování. Přitom by byly zachovány dříve zmíněné výhody.

6.3 Metoda LSB Vigenère

Podobně jako předešlá šifra, i tato kóduje informaci pomocí nejméně významných bitů v bitmapě, nikoli ale pomocí záměn, nýbrž jejich přímým přepisováním, jako klasická LSB šifra.

Tato metoda je navíc vylepšena o myšlenku kryptografické Vigenèrovy šifry. Stručně řečeno, Vigenèrova šifra šifruje tajnou zprávu pomocí klíče, který se skládá ze sekvence nezáporných celých čísel. Při procesu šifrování se postupně prochází znaky šifrované zprávy společně s hodnotami klíče, kde je každý znak

[28] Sofistikovanější statistické analýzy mají větší šanci na úspěch.

[29] Více o metodách analýzy steganografických metod je uvedeno v kapitole 5.

[30] A to dokonce ještě v tom případě, kdy nejméně významné bity bitmapy obsahují šum v celém obrázku (masky bitmapy neobsahují oblasti stejných barev).

[31] Obsahuje tolik bytů, kolik obsahuje znaků, jelikož se kóduje podle standardu ANSI.

zaměněn za znak, který je posunut o tolik pozic, kolik je aktuální hodnota obsažená v klíči. Když se dorazí na konec sekvence klíče, prochází se opět od začátku, dokud není zakódována celá zpráva.[32] Šifra LSB Vigenère nejprve provede Vigenèrovo šifrování[33] nad vstupním textem a poté, pomocí stejného klíče, šifruje zprávu do bitmapy tak, že přeskakuje tolik nejméně významných bitů bitmapy, kolik je aktuální hodnota obsažená v klíči a do následujícího nejméně významného bitu uloží bit zprávy.

Princip Vigenèrova šifrování je ještě dále vylepšen o tu myšlenku, že se nemusí stále dokola používat stejná sekvence obsažená v klíči, ale mohou se používat její různé permutace. Tudíž klíč neobsahuje pouze sekvenci hodnot pro posuv, ale navíc obsahuje ještě další sekvenci, která obsahuje hodnoty udávající, jaká permutace originální sekvence posuvů je momentálně uvažována. Je samozřejmě nutno vygenerovat všechny permutace v nějakém jednoznačném uspořádání, aby mohly být očíslovány. Tato očíslování jsou poté použita v druhé části klíče.[34]

Vylepšení v podobě permutací sekvence posuvů má význam hlavně ten, že je celé řešení bezpečnější vůči klasickým kryptoanalytickým postupům pro rozšifrování šifry i bez znalosti klíče. Nejběžnější je použití testu Kasiského následovaný Friedmanovým testem. Kasiského test dokáže odhadnout délku klíče Vigenèrovi šifry a Friedmanův test následně ověřuje, zdali je tato délka správná.[35] Zjištění této délky je důležité v tom, že sekvence posuvů se po této délce opět opakuje. Následně je možné provádět frekvenční analýzu pro posuvnou šifru nad znaky, které jsou vzdáleny o tuto délku. Díky tomu, že se nepoužije pouze originální sekvence posuvů, ale zřetězení jejich permutací, je tahle délka, za kterou se bude celá sekvence znovu opakovat, mnohem delší. Konkrétně když sekvence posuvů bude mít délku n a sekvence permutací bude mít délku m, tak celá sekvence posuvů, než se začnou posuvy opět opakovat, je $n \cdot m$. To znamená, že při lineárním zvětšování obou sekvencí narůstá celková délka sekvence kvadraticky. Dalším možným vylepšením by bylo neuvažovat pouze jedinou sekvenci permutací, ale rovnou celou její množinu sekvencí.

Například pro sekvenci posuvů $5, 8, 3$ se předpokládá následující uspořádaní permutací:

Index	Permutace
0	$5, 8, 3$
1	$5, 3, 8$
2	$8, 5, 3$
3	$8, 3, 5$
4	$3, 8, 5$
5	$3, 5, 8$

[32]Podrobnosti o Vigenèrově šifře jsou obsaženy v [5].

[33]Ve skutečnosti nejde o klasické Vigenèrovo šifrování, viz dále.

[34]Tento princip je použit jak při šifrování v rámci kryptografické části, tak i při šifrování během steganografické fáze.

[35]Více o těchto testech je uvedeno v [5].

Poté, když se uvažuje klíč, který obsahuje tutéž sekvenci posuvů se sekvencí permutací $3, 5, 0, 2, 1$, bude celá sekvence posuvů vypadat následovně:

$$8, 3, 5|3, 5, 8|5, 8, 3|8, 5, 3|5, 3, 8$$

Dešifrování pracuje velmi podobně. Nejprve samozřejmě dochází k dešifrování steganografické části a následně k dešifrování kryptografické. Jako u klasické Vigenèrovy šifry, v kryptografické části se posuvy berou se zápornými hodnotami pro posuvy. Ve steganografické části tomu tak samozřejmě není.

Jako v případě LSB Swap, tak i tato metoda zapisuje do prvních 32 nejméně významných bitů celé číslo, které označuje délku zprávy, aby mohlo dojít k dešifrování zprávy bez následného šumu.

6.4 Metoda Comments cypher

Na rozdíl od dvou předešlých metod, šifruje tato metoda zprávu do sekvence komentářů, kde je nad množinou těchto komentářů sestaven Huffmanův kód[36].

Nejprve je podle souboru, obsahující seznam komentářů s jejich váhami, vytvořen Huffmanův kód. Následně je vstupní zpráva převedena na proud bitů, ve kterém jsou hledány prefixy, které odpovídají kódovým slovům vytvořeného Huffmanova kódu.[37] Komentář, který odpovídá právě nalezenému kódovému slovu, je uložen do sekvence a proud bitů zprávy se zkrátí o použitý prefix. Tento proces se provádí tak dlouho, dokud se nevyčerpá celý proud bitů.

Na konci tohoto šifrování mohou nastat dvě možnosti. Buď zbytek proudu bitů odpovídá přesně nějakému kódovému slovu, poté je jeho odpovídající komentář přidán na konec sekvence a sekvence kóduje přesně identickou zprávu s originální zprávou. Mnohem častěji ale dojde k případu, kdy zbytek proudu bitů obsahuje řetězec kratší než kódová slova. V tomto případě je možno vybrat libovolné kódové slovo, jehož prefix se shoduje s řetězcem ve zbytku proudu. Tím nastane to, že se na konec zašifrované zprávy dostane ještě nějaký postfix, který ale neobsahuje originální zpráva.

Opět velmi podobně vypadá dešifrování – sestaví se Huffmanův kód podle seznamu komentářů, komentáře ve vygenerované sekvenci jsou postupně převáděny na kódová slova, ze kterých je budován proud bitů. Ten je na závěr převeden na textovou zprávu.

Šifru by bylo možno vylepšit po uživatelské stránce, co se týče vytváření seznamu komentářů, tak, že by seznam komentářů byl implementován jako formální gramatika. Uživatel by tak nemusel psát každý komentář zvlášť, ale definoval by pravidla této gramatiky, čímž by se efektivněji tvořilo velké množství vět gramatiky (komentářů). Na aktuální implementaci se dá pohlížet jako na speciální

[36] Huffmanův kód je detailněji popsán v sekci 4.5.2.

[37] Z důvodu toho, že Huffmanův kód je prefixový, existuje vždy právě jedno kódové slovo, které je prefixem zbývajícího proudu bitů. Jediná výjimka je ta, kdy zbytek proudu bitů je již kratší než kódová slova (viz dále).

případ gramatiky, kdy se ze startovního neterminálu ihned odvozuje jediný terminální symbol, a tím i věta gramatiky.

7 Uživatelská dokumentace

Praktickou částí diplomové práce je aplikace implementující tři šifry – LSB Swap, LSB Vigenère a Comments cypher. První dvě metody šifrují vstupní zprávu do obrázkové bitmapy a poslední zmíněná metoda šifruje zprávu do sekvence komentářů. Více detailů o implementačních aspektech daných šifer je uvedeno v kapitole 6 (Programátorská příručka).

Aplikace byla vytvořena pro operační systém Windows na platformě Microsoft .NET Framework verze 4.7.1 a operačním systému Windows 7 Ultimate. Aplikace zpracovává vstupní textové soubory ve standardním kódování ANSI (8-bitů pro znak).

7.1 Spuštění aplikace

Pro spuštění aplikace není požadována její instalace. Před spuštěním je ale nezbytné překopírovat adresář *bin/* na pevný disk počítače a spustit aplikaci z nově vzniklé složky. Aplikace se spustí souborem *StegoTools.exe*, který se nachází v adresáři *bin/*.

7.2 Práce s aplikací

Po spuštění aplikace se otevře okno s pásem karet (ribbon) s označenou kartou Hlavní, která vypíše hlavní informace o programu.

Jak již bylo zmíněno, program nabízí tři steganografické nástroje, ke kterým se přistupuje skrze záložky nahoře okna aplikace. Každá záložka dané metody obsahuje její detailnější popis pro práci s ní a dále samotné funkce pro šifrování a dešifrování. Na obrázku 11 je zobrazeno hlavní okno po zapnutí aplikace.

V levém horním rohu okna lze rozbalit nabídku menu, která obsahuje tlačítko pro vypnutí aplikace a tlačítko pro zobrazení podrobné nápovědy[38]. Na obrázku 12 je ukázáno toto menu. Dále všechna tlačítka v aplikaci jsou opatřena tooltipy – po najetí kurzorem na ně se zobrazí stručná nápověda.

7.2.1 Metoda LSB Swap

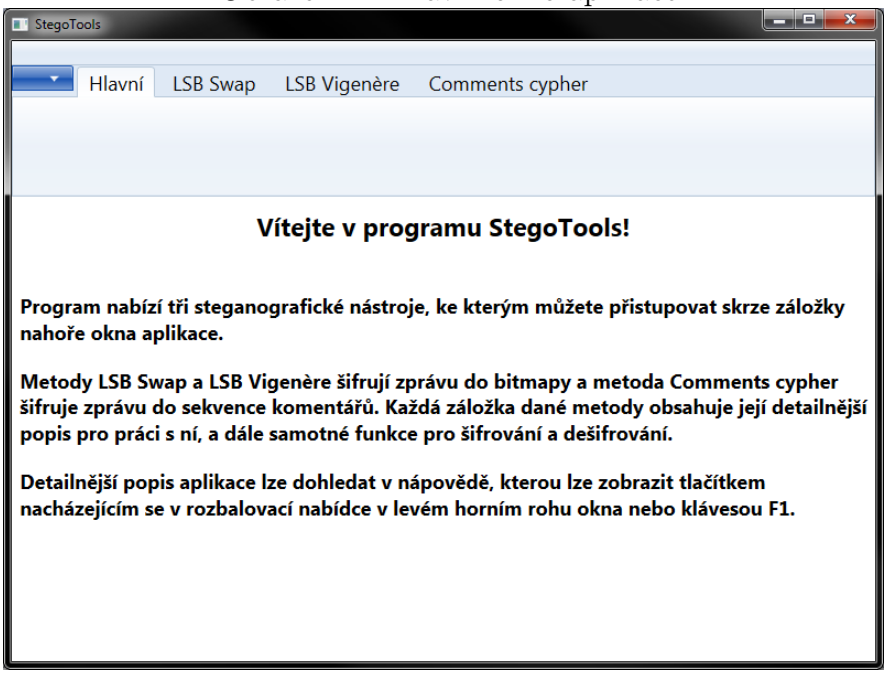
Záložka pro metodu LSB Swap nabízí tři formuláře. První obsahuje návod k použití této metody a další dvě nabízí funkce k samotnému šifrování a dešifrování.

7.2.1.1 Šifrování

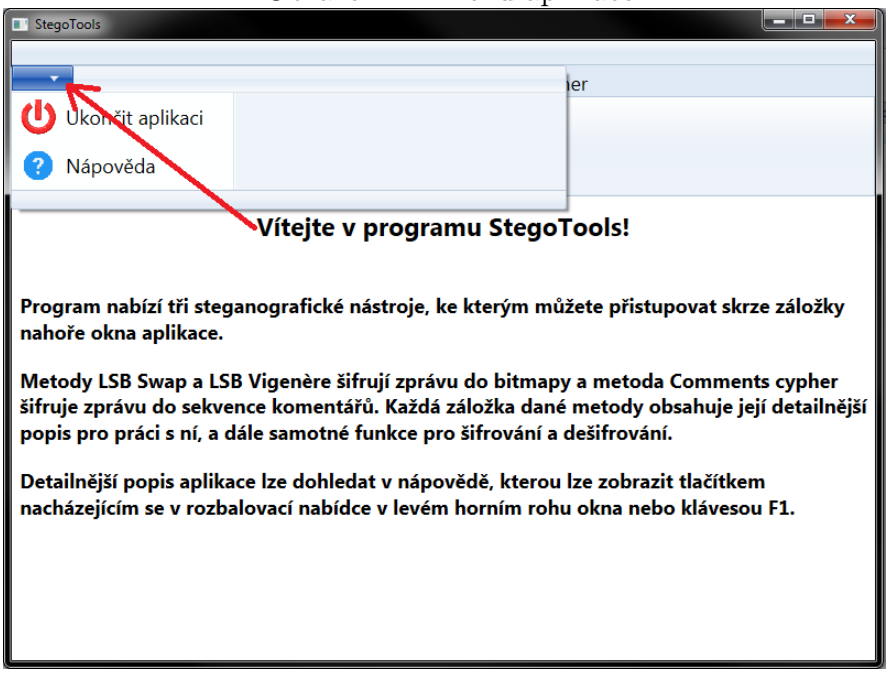
Vstupní data pro šifrování metodou LSB Swap tvoří textová zpráva (soubor ve formátu .txt) a bitmapa (soubor ve formátu .bmp). Proto jsou na formuláři pro šifrování dvě textová pole pro vyplnění cest k těmto dvěma souborům. Pole se naplňují tlačítkem vpravo (...), které otevře okno průzkumníka souborů v místě,

[38]Nápověda je přiložena ve formě souboru ve formátu .pdf (v adresáři *bin/*), proto je nutno mít nainstalovaný program pro prohlížení tohoto typu souborů.

Obrázek 11: Hlavní okno aplikace



Obrázek 12: Menu aplikace



kde již existuje několik vzorových souborů. Je ovšem možné se průzkumníkem navigovat libovolně po datovém úložišti a zvolit jakýkoli soubor daného formátu. Tlačítko pro vyplnění cesty k textovému souboru otevře průzkumníka ve složce `bin/Messages` a tlačítko pro vyplnění cesty k bitmapě otevře průzkumníka ve složce `bin/Bitmaps`.

Po vyplnění cesty k souboru se zprávou je možné ji zobrazit tlačítkem Zobrazit, které se nachází vpravo pod textovým polem cesty. V případě bitmapy také existuje tlačítko na její zobrazení, které se nachází dole v sekci Zobrazit s konkrétním popisem Originál.

Alternativní naplnění cesty k textovému souboru je možné pomocí tlačítka Zadat ručně. Po stisknutí tlačítka se objeví okno, do kterého je možno napsat vlastní zprávu. Ta je po uložení uložena do složky `bin/Messages` a je naplněna cesta pro textovou zprávu.

Vyplněním pole pro cestu k souboru zprávy se také zobrazí, jakou má zpráva velikost a podobně – v případě bitmapy – aplikace zobrazí, jakou má obrázek kapacitu pro uloženou zprávu. Informace, že zpráva má větší velikost, než může obrázek pojmout, je označena červeným textem velikostí zprávy a kapacity bitmapy. Nicméně je možné i v tomto případě provádět šifrování, viz dále.

Až jsou obě pole vyplněna, zpřístupní se tlačítko Šifrovat, kterým se provádí samotné šifrování. Je-li zpráva delší než kapacita bitmapy, zobrazí se varovné hlášení s upozorněním, že zpráva se nezašifruje celá, avšak je možné pokračovat. Po provedení šifrování se uloží výsledná bitmapa do složky `bin/LSBSwap/Encoded` a je ji možné zobrazit pomocí tlačítka dole v sekci Zobrazit s popiskem Zašifrovanou.

Poslední a velmi důležitý popis si zaslouží šestice tlačítek v sekci Zobrazit s popisky R, G a B, které přísluší jak originálnímu, tak i zašifrovanému obrázku. Každým z těchto tlačítek se vytvoří takzvaná maska v dané barevné intenzitě příslušného obrázku. Písmeno R odpovídá červené, G zelené a nakonec B modré barevné intenzitě. Tyto masky zobrazují, jaké mají hodnoty nejméně významné bity v pixelech daných barevných intenzit. Po stisknutí tlačítka se zobrazí maska dané barevné intenzity příslušné bitmapy.[39] Při zobrazení obou masek (originálního a zašifrovaného obrázku) konkrétní barevné intenzity, je možné pozorovat jejich rozdíly výměnou zobrazovaných oken s maskami pomocí *ALT + TAB*. Masky se taktéž ukládají na disk, a to do složky `bin/LSBSwap/Masks`.

Na obrázku 13 je ukázán formulář pro šifrování metodou LSB Swap.

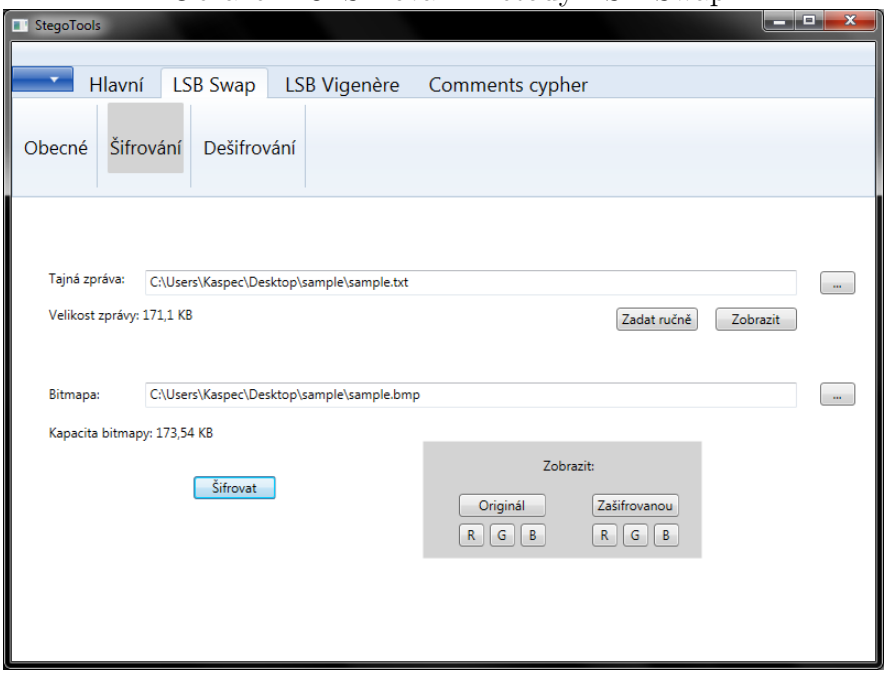
7.2.1.2 Dešifrování

Ve formuláři pro dešifrování pomocí metody LSB Swap je pouze jediné pole pro vyplnění, a to pole pro vyplnění cesty k zašifrovanému obrázku. Toto pole se vyplňuje stejně, jako již bylo uvedeno v sekci 7.2.1.1.

Po vyplnění pole se zpřístupní tlačítko Dešifruj, které po úspěšném dešifrování uloží dešifrovanou zprávu na disk do složky `bin/LSBSwap/Decoded` se

[39] Upozornění – u velkých bitmap může vytváření masek trvat poměrně dlouho.

Obrázek 13: Šifrování metody LSB Swap



jménem zašifrovaného obrázku a také se tato zpráva ihned zobrazí uživateli.

7.2.2 Metoda LSB Vigenère

Záložka pro metodu LSB Vigenère opět nabízí tři formuláře. První obsahuje návod k použití této metody a další dvě nabízí funkce k samotnému šifrování a dešifrování.

7.2.2.1 Šifrování

V případě šifrování metodou LSB Vigenère vypadá formulář velmi podobně jako formulář se šifrováním metodou LSB Swap. Avšak je tu jeden rozdíl.

Tato metoda kromě vstupních souborů ve formě šifrované zprávy a bitmapy obsahuje ještě klíč, který je používán při šifrování zprávy do bitmapy. Tudíž je zde obsaženo ještě jedno pole navíc k vyplnění cesty právě k tomuto textovému souboru obsahující klíč. Po stisknutí tlačítka pro zobrazení průzkumníka souborů (...) je aktuální složkou `bin/LSBVigenere/Keys`, která opět obsahuje vzorové klíče.

Struktura souboru, obsahující klíč, je taková, že se skládá ze dvou řádků celých nezáporných čísel oddělenými čárkami, přičemž druhý řádek může obsahovat pouze čísla, která jsou menší než $n! - 1$, kde n je počet čísel v prvním řádku[40]. Více o významu tohoto klíče je uvedeno v programátorské příručce v sekci 6.3.

[40]Druhý řádek může být ponechán prázdný. V tomto případě jde o klasické Vigenèrovo šifrování.

Po stisknutí tlačítka Šifrování je zpráva zašifrována do obrázku, který se uloží do složky
`bin/LSBVigenere/Encoded` a objeví se informace o úspěšném šifrování.

Oproti předešlé metodě LSB Swap tato metoda počítá kapacitu bitmapy nejen podle její samotné velikosti, ale také podle klíče. To znamená, že kapacita bitmapy se zobrazí až po vyplnění obou těchto polí.

7.2.2.2 Dešifrování

Dešifrování je opět velmi podobné jako v případě LSB Swap, které je ovšem rozšířeno stejné, jako v případě šifrování – o vyplnění cesty k souboru obsahující klíč.

Po stisknutí tlačítka k dešifrování se zobrazí dešifrovaná zpráva, která je opět uložena do složky `bin/LSBVigenere/Decoded` pod stejným názvem jako obrázek použitý k dešifrování.

7.2.3 Metoda Comments cypher

Jako dvě předchozí záložky, tak i tato nabízí tři formuláře. První obsahuje návod k použití této metody a další dvě nabízí funkce k samotnému šifrování a dešifrování.

7.2.3.1 Šifrování

Šifrování v případě metody Comments cypher se velmi liší od předchozích popsaných šifer z toho důvodu, že se tajná zpráva nešifruje do bitmapy, nýbrž podle souboru, který obsahuje seznam frází (komentářů).[41] Při procesu šifrování je určitým způsobem vytvářena sekvence komentářů podle vstupního seznamu komentářů a tajné zprávy.[42]

Formulář vypadá opět velmi podobně jako v předchozích případech – místo pole pro vyplnění cesty k souboru obsahující bitmapu je zde velmi podobné pole pro cestu k souboru obsahující seznam komentářů. Tlačítko pro plnění pole cesty otevře průzkumníka souborů ve složce
`bin/CommentsCypher/CommentsLists`.

Stejně jako v sekci 7.2.2.1, kde má soubor obsahující klíč určitou strukturu, má i soubor s komentáři specifickou strukturu. Komentáře se zde dělí do skupin, kde celá skupina má určenou váhu – s jak velkou mírou se budou komentáře v této skupině objevovat ve vygenerované sekvenci komentářů šifrující zprávu.[43] Nejdříve je tedy na samotném řádku uvedena kladným číslem váha skupiny a následují komentáře, opět každý na novém řádku. Skupiny se oddělují řetězcem

[41] Seznam komentářů může reprezentovat například různé komentáře k nějaké fotce, která byla nahrána na sociální síť.

[42] Více detailů v sekci 6.4.

[43] Obor hodnot vah jsou přirozená čísla. Vyšší číslo znamená častější zastoupení komentářů v generované sekvenci

„\\\“, který je taktéž na samotném řádku. Po tomto oddělovači následuje další skupina. Žádný komentář se v rámci celého souboru nesmí objevovat vícekrát. Ovšem vygenerovaná sekvence komentářů již může (a obecně bude) obsahovat duplicitní komentáře. Prázdné řádky v souboru, obsahující seznam komentářů, se ignorují.

Po vyplnění obou polí na formuláři je možné použít tlačítko pro šifrování. To uloží vygenerovanou sekvenci komentářů do souboru, který se uloží do složky `bin/CommentsCypher/Encoded` a také vygenerovanou sekvenci ihned zobrazí uživateli.

7.2.3.2 Dešifrování

Jako v předchozích dvou případech, dešifrování opět nabízí dvě pole pro vyplnění cesty k souborům obsahující vygenerované komentáře šifrující zprávu a seznam komentářů. Poté se zpřístupní tlačítko pro dešifrování, po jehož stisknutí se zobrazí zašifrovaná zpráva, která je uložena do složky `bin/CommentsCypher/Decoded`.

Závěr

Výsledkem této diplomové práce je aplikace, která implementuje tři steganografické šifry: LSB Swap, LSB Vigenère a Comments cypher. První dvě z nich šifrují tajnou zprávu do obrázkových bitmap, konkrétně do nejméně významných bitů bytů reprezentující barevnou intenzitu, a poslední zmíněná šifra šifruje tajnou zprávu do generovaného textu (jednotlivých komentářů) podle vstupního seznamu frází (opět tzv. komentářů).

Oproti klasické variantě LSB si LSB Swap vede velmi dobře proti statistickým, i vizuálním analýzám. Nicméně je pomocí ní možno zakódovat méně informace. Toto množství informace je tím větší, čím vyšší je šum v nejméně významných bitech.

V porovnání šifry LSB Vigenère vůči klasické metodě LSB, co se týče statistických a vizuálních analýz, je na tom LSB Vigenère opět lépe, nicméně ve srovnání s LSB Swap si vede hůře. I přes to má výhody v tom, že ke svému šifrování používá klíč a navíc kromě samotného steganografického šifrování je kombinována s kryptografickou metodou Vigenèrovy šifry. Co se týče kapacity pro tajnou zprávu, ta je velmi závislá na zvoleném klíči, respektive na velikosti hodnot posuvů v klíči. Obecně však bude ještě nižší než v případě LSB Swap. Hodnotami v klíči se také řídí i to, jak dobře bude šifra ukryta před analýzami.

Šifra Comments cypher se dá jen velmi těžko s předešlými šiframi srovnávat vzhledem k principu jejího fungování. Co se týče délky vygenerovaných komentářů vzhledem ke vstupní tajné zprávě, tak ta velmi závisí na velikosti vstupního seznamu komentářů při jejím šifrování. V určitých případech by se dokázalo vymyslet opravdu mnoho různých komentářů, například reakce na fotku, která byla nahrána na sociální síť. V případě tohoto šifrování často nastává, že se komentáře ve vygenerované sekvenci opakují. To ale v uvažovaném případě s fotkou není problém, jelikož běžné komentáře nemusí vzbudit podezření při jejich opakování. V seznamu komentářů se dá u jednotlivých komentářů definovat, s jak velkým zastoupením se mají generovat do výstupní sekvence komentářů.

Tento dokument slouží k tomu, aby čtenáře uvedl do základů kryptografie, seznámil s historií steganografie. Dále popisuje steganografické metody a metody s ní související a zmiňuje principy steganoanalýzy. Na závěr obsahuje uživatelskou a programátorskou dokumentaci k vytvořené aplikaci.

Conclusions

The main outcome of this master's thesis is an application which implements three steganographic ciphers: LSB Swap, LSB Vigenère and Comments cypher. The first two encrypt a secret message in picture bitmaps, specifically to the least significant bits of bytes representing color intensity, and the last mentioned cypher encrypts a secret message in the generated text (individual comments) according to a list of phrases (also comments).

Compared to the classic variant of LSB, LSB Swap is much more effective against statistical and also visual analysis. It, however, encrypts less information. The quantity of information gets greater with an increasing amount of noise in the least significant bits.

When comparing the LSB Vigenère cypher and classic LSB in regards of a statistical and visual analysis, LSB Vigenère is better, but in comparison with LSB Swap, this cypher is worse. Nevertheless, it has an advantage in the fact that it uses in its encryption process a key, and it is, besides the steganographic encryption itself, combined with a cryptographic method of the Vigenère cypher. The capacity of this method is very dependent on the chosen key, respectively on the size of the offset values in the key. Generally, the capacity of this method should be smaller in comparison with LSB Swap. The values in the key also affect how well the cipher is hidden from analyses.

The Cypher Comments cypher cannot be compared with the other cyphers because it works according to a different principle. As far as the length of the generated comments in regards to the input secret message is concerned, it depends on the size of the input list of comments during the encyphering process. In some cases, it is easy to devise a lot of different comments, for example reactions to a photo uploaded on a social network. In the case of this encyphering, the comments in the generated sequence are often repeating themselves. In the case with the photo, however, this is not a problem, because common comments do not need to rouse suspicion when repeated. In the list of comments, it is possible to define the probability with which each comment will appear in the output sequence of comments.

The purpose of this document is to introduce the reader to the basics of cryptography, and familiarize them with the history of steganography. Furthermore, this document describes steganography and the methods related with it, and mentions the basics of steganoanalysis. At the very end, there is the user's and programmer's documentation for the created application.

A Obsah přiloženého DVD

Součástí diplomové práce je i DVD obsahující následující adresáře a soubor:

`bin/`
: Adresář obsahuje soubor StegoTools.exe ke spuštění aplikace, složky pro ukládání nově vytvořených textových souborů a bitmap, které již obsahují několik vzorových souborů, a soubor s nápovědou (obsahuje uživatelskou a programátorskou dokumentaci).

`doc/`
: Adresář obsahuje text práce ve formátu PDF včetně zdrojových souborů.

`src/`
: Adresář obsahuje zdrojové kódy aplikace.

`readme.txt`
: Soubor readme.txt obsahuje postup pro spuštění aplikace. Dále obsahuje odkazy na materiály, ze kterých bylo čerpáno při programování aplikace.

Reference

[1] Peter Wayner, *Disappearing cryptography: information hiding: steganography & watermarking,* Cambridge University Press, 2002, ISBN-10: 1-55860-769-2.

[2] Jessica Fridrich, *Steganography in digital media: principles, algorithms, and applications,* Cambridge University Press, 2010, ISBN-13: 978-0-521-19019-0.

[3] John Talbot, *Complexity and cryptography: an introduction,* Cambridge University Press, 2006, ISBN-13: 978-0-521-85231-9.

[4] Christof Paar, *Understanding cryptography: a textbook for students and practitioners,* Springer, 2010, ISBN-13: 978-3-642-04101-3.

[5] Douglas R. Stinson, *Cryptography: theory and practice,* Chapman & Hall/CRC, 2006, ISBN-13: 978-1-58488-508-5.

[6] Robert B. Ash, *Information theory,* Dover Publications, 1990, ISBN-10: 0-486-66521-6.

[7] David Salomon, *Data compression: the complete reference,* Springer, 2007, ISBN-13: 978-1-84628-602-5.

[8] Manindra Agrawal, Neeraj Kayal, Nitin Saxena *PRIMES is in P*
https://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf

[9] Ross Anderson, Roger Needham, Adi Shamir *The Steganographic File System*
https://www.cl.cam.ac.uk/~rja14/Papers/stego-fs.pdf

[10] YouTube *Admiral Jeremiah Denton Blinks T-O-R-T-U-R-E using Morse Code as P.O.W.*
https://www.youtube.com/watch?v=rufnWLVQcKg

[11] YouTube *Zero Knowledge Proofs – Computerphile*
https://www.youtube.com/watch?v=HUs1bH85X9I

[12] *Microsoft ribbon – Pás karet*
https://docs.microsoft.com/en-us/windows/desktop/uxguide/cmd-ribbons

[13] Wikipedia *Kemps (card game)*
https://en.wikipedia.org/wiki/Kemps_(card_game)