VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

STEGANOGRAFICKÁ APLIKACE PRO MOBILNÍ ZAŘÍZENÍ

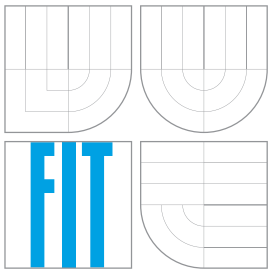
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DAVID BERAN

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

STEGANOGRAFICKÁ APLIKACE PRO MOBILNÍ ZAŘÍZENÍ

STEGANOGRAPHIC APPLICATION FOR MOBILE DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DAVID BERAN

VEDOUCÍ PRÁCE
SUPERVISOR

Mgr. KAMIL MALINKA, Ph.D.

BRNO 2015

Abstrakt

Tato práce řeší výběr vhodné steganografické metody pro mobilní platformu a její implementaci. Vybrána byla metoda F4 a byla implementována jako aplikace Stego a její rozšíření Reveal pro zařízení se systémem iOS.

Abstract

This paper discusses selection of suitable steganographic method for mobile platform and its implementation. F4 method was selected and implemented as an application Stego and its extension Reveal for iOS devices.

Klíčová slova

steganografie, kryptografie, steganogram, krycí objekt, mobilní aplikace, iOS

Keywords

steganography, cryptography, steganogram, cover object, mobile application, iOS

Citace

David Beran: Steganografická aplikace pro mobilní zařízení, bakalářská práce, Brno, FIT VUT v Brně, 2015

Steganografická aplikace pro mobilní zařízení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Mgr. Kamila Malinky, Ph.D.

.....
David Beran
20. května 2015

© David Beran, 2015.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

Potřeba utajování komunikace se v lidské společnosti vyvinula již několik staletí před naším letopočtem. Historicky první zmínky můžeme nalézt v Hérodotových Historiích z 5. století př. n. l., podle kterých spartský král Demaratus varoval Řeky před perským útokem pomocí zprávy vyryté na dřevěné destičce zalité vrstvou vosku. O něco mazanější a trpělivější byl milétský tyran Histaeus, který zosnoval povstání proti perskému králi Dariovi I., jehož tažení se účastnil. Samotnou zprávu s výzvou k povstání vytetoval Histaeus na vyholené temeno hlavy svého otroka a až vlasy znovu dorostly, poslal jej za svým synovcem. Přestože bylo povstání potlačeno a Histeaus byl později popraven královým bratrem, samotný Darius nikdy neuvěřil v Histeaovu zradu a nechal jeho hlavu slavnostně pohřbít.

Ať už jsou tyto příběhy pravdivé nebo se jedná o pouhé legendy, mnohé další příklady z naší historie potvrzují, že odhalení či neodhalení tajné zprávy může významě ovlivnit osudy mnoha lidí. Využití však steganografie nenalézá pouze během válečných konfliktů. Díky jejímu vývoji ji dnes můžeme aplikovat v mnoha oborech a může mít množství různých podob.

Cílem práce je analýza a porovnání dostupných metod obrazové digitální steganografie, výběr vhodné metody pro mobilní platformu a její implementace. Součástí práce je také návrh a implementace vhodného uživatelského rozhraní pro implementovanou metodu.

V kapitole 2 je vysvětlen rozdíl mezi kryptografií a steganografií a je v ní definován steganografický systém. Kapitola 3 se věnuje vlastnostem steganografických metod, které poslouží k jejich porovnání a výběru nejvhodnější metody. Následuje kapitola 4 věnovaná steganalýze, tedy odhalování steganografie. Dalším důležitým tématem rozebraným v kapitole 5 je krycí objekt a aspekty, na které je třeba brát ohled při jeho výběru. Rozdělení a komparace dnešních steganografických metod je v kapitole 6.

Návrh řešení aplikace, která je výsledkem této práce, je popsán v kapitole 7. V této kapitole se také zabývá práce možnostmi cílové platformy a výběrem vhodné metody k implementaci. Popis samotné implementace a uživatelského rozhraní je obsahem kapitoly 8.

Kapitola 2

Terminologie

Steganografie je vědní disciplína zabývající se utajováním zpráv tak, aby případná třetí strana v komunikaci nevěděla, že tajná komunikace vůbec probíhá. Zprávy jsou skrývány do různých médií, ať už fyzických (např. „neviditelný inkoust“ na papíře) nebo digitálních (obrazové soubory, audio, video, text atd.). Do druhého zmíněného druhu médií ukrývá informace digitální steganografie, na kterou se v této práci soustředím. Médium, do kterého je zpráva ukryta, se nazývá krycí objekt (*cover object*). Výsledné médium obsahující ukrytou zprávu je nazýváno steganogram (*steganogram* nebo *stego object*).

Disciplínou vycházející ze steganografie je staganalýza (*steganalysis*), která se zabývá odhalováním zpráv ukrytých steganografickými metodami, většinou bez znalosti původního krycího objektu. Protože steganografické metody vždy nějakým způsobem změní krycí objekt, zanechávávají za sebou více či méně odhalitelné stopy. Tyto stopy se pak steganylýza snaží najít a odhalit, že médium je ve skutečnosti steganogram obsahující zprávu.

Obecně proces steganografiického skrývání informací sestává ze dvou základních kroků. Prvním je identifikace redundantních bitů v krycím objektu, které mohou být pozměněny bez toho, aby byla významně narušena kvalita média. Druhým krokem je pak selekce podmnožiny těchto bitů, která bude posléze nahrazena bity ukrývané zprávy, čímž vznikne výsledný steganogram [16].

2.1 Steganografie a kryptografie

Přestože steganografie je svou podstatou podobná kryptografii, nesmíme tyto pojmy v žádném případě zaměňovat. Zatímco kryptografie (z řeckého *kryptós* – skrytý, tajný; *gráphein* – psát) zprávu převádí do šifrované podoby, steganografie (z řeckého *steganós* – ukrytý, zakrytý) zprávu ukrývá do jiného média. U šifrované zprávy je často očividné, že se jedná o tajnou zprávu, útočník však potřebuje znát klíč pro odhalení její původní podoby. Zpráva ukrytá steganografickou metodou však *nevypadá* jako zpráva, ale jako obyčený objekt, který plní zcela jiný účel a není nositelem tajné informace.

Ukázkovým příkladem může být v úvodu práce zmíněný otrok s vytetovanou zprávou pod dorostlými vlasy, což je zpráva ukrytá steganografickou metodou. Pokud útočníka napadne prozkoumat, zda útočník na těle nemá ukrytou zprávu, je vcelku pravděpodobné, že ji odhalí velmi rychle. Útočníka však nemusí napadnout zprávu hledat, protože vidí jen obyčejného otroka, který nevypadá v ničem podezřele. Pokud však otrok nese zprávu ukrytou kryptografickou metodou, bude vybaven například svitkem se zdánlivě nesmyslným textem či neznámými symboly. To pravděpodobně vzbudí podezření a svitek bude útoč-

níkem zabaven. I kdyby nebyla zpráva útočníkem odhalena, nesplnila svůj účel, protože nebude doručena adresátovi.

Oba obory se tedy mohou hodit pro různé účely a v různých situacích. Nedá se obecně říct, že by byl jeden lepší či horší než druhý. Nelze ani tvrdit, že by steganografie byla podoborem kryptografie. Jedná se spíše o „sesterské“ obory.

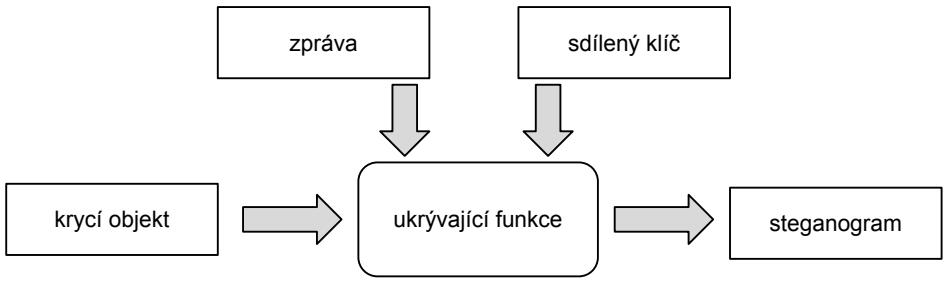
Pozornějšího čtenáře možná napadne v rámci zvýšení bezpečnosti obě tyto metody zkombinovat, tedy zprávu nejdříve zašifrovat a poté ukrýt. Příjemce zprávy pak musí zprávu odkrýt a poté rozšifrovat. Toto však nemusí být tak účinná metoda, jak by se mohlo na první pohled zdát — často může být dokonce kontraproduktivní a zvyšovat pravděpodobnost odhalení.

2.2 Steganografický systém

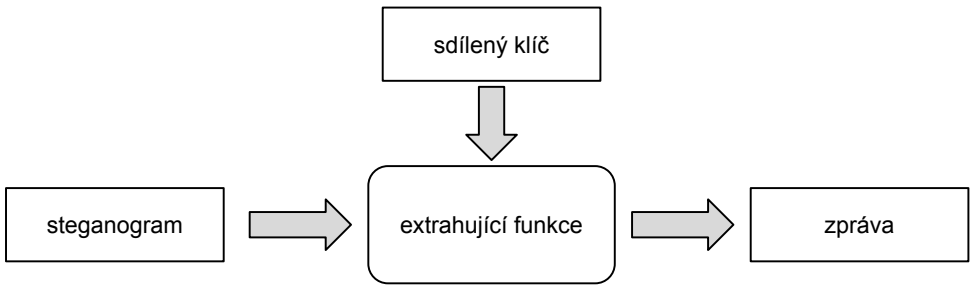
Steganografie zavádí pojem steganografický systém, který popisuje proces skrývání i odkrývání zprávy.

Steganografický systém je mechanismus skrývající zprávu m do krycího objektu c pomocí tajného sdíleného klíče k. Výsledkem je steganogram s, který nese zprávu m. Formálně definujeme steganografický systém jako pár mapujících funkcí (F, G), kde F je ukrývající funkce a G funkce extrahující [7]. Schéma skrývání a odkrývání zprávy je vidět na obr. 2.1 a obr. 2.2.

$$s = F(c, m, k)$$
$$m = G(s, k)$$



Obrázek 2.1: Vkládání zprávy



Obrázek 2.2: Extrahování zprávy

Kapitola 3

Vlastnosti steganografických metod

V předchozí kapitole byl čtenář seznámen se steganografií a jejími odlišnostmi od kryptografie. Měl by také mít alespoň elementární představu o důležitých pojmech souvisejících se steganografií. V následujících kapitolách by tyto pojmy měly nabýt konkrétnější obrysy a čtenář by tak měl získat o steganografii ucelenější představu.

Steganografické metody mají různé vlastnosti, které je třeba nejdříve definovat, aby bylo možné jednotlivé metody porovnávat a nakonec vybrat metodu vyhovující konkrétnímu účelu.

Mnohé zdroje uvádějí mezi tyto vlastnosti například postřehnutelnost (*perceptibility* – odhalitelnost pouhým lidským okem), odhalitelnost (*detectability* – obtížnost určit statistickými či technologickými prostředky, zda steganogram obsahuje skryté informace) či rychlost (*speed* – určuje množství času, který je potřeba na skrytí a odkrytí informace) [20]. První dvě vlastnosti jsou však značně abstraktní a lze je nahradit určením bezpečnosti metody (viz dále). S dnešním běžným výpočetním výkonem počítačů i mobilních zařízení mohu v rámci této práce zanedbat i rychlost, neboť skrývání i extrahování je vždy možné provést v přijatelném čase. Speciálním případem by mohla být potřeba výpočtu v reálném čase (například skrývání do streamovaného zvuku či videa), to však není cílem této práce.

Klíčovými vlastnostmi, které jsou skutečně důležité při popisu a výběru steganografické metody, jsou kapacita, bezpečnost a robustnost [20].

3.1 Kapacita

Kapacita (*capacity*) je množství informací, které může být do krycího objektu uschováno bez signifikantního narušení jeho ostatních vlastností.

3.2 Bezpečnost

Bezpečnost (*security*) steganografické metody určuje, jak náročné bude pro případného útočníka odhalit, že médium obsahuje zprávu. Je možné ji definovat mnoha způsoby, z nichž každý se může hodit pro různé druhy médií a steganalytických metod.

První a často citovanou definici stanovil Christian Cahin [6]. Dle této definice je steganografická metoda ϵ-bezpečná (ϵ-secure), kde $\epsilon \geq 0$, pokud relativní entropie mezi distribuční funkcí krycího objektu a steganogramu je rovna maximálně ϵ.

$$D(P_c||P_s) = \int P_c \log \frac{P_c}{P_s} \leq \epsilon$$

Metoda je považována za dokonale bezpečnou (*perfectly secure*), pokud platí, že $\epsilon = 0$. Dokonale bezpečné metody nejsou jen teoretickým pojmem a skutečně existují, nejsou však příliš praktické. Tato definice totiž popisuje pouze jedinou charakteristiku média a tou je distribuční funkce, proto ji lze použít spíš jen pro náhodně vygenerované bity. V reálném světě však média obsahují mnohem více různě závislých charakteristik a jejich narušení může být sofistikovanější steganalytickou metodou odhaleno i v případě, že je během skrývání distribuční funkce zachována a $\epsilon = 0$.

Dle [8] je možné míru bezpečnosti steganografického systému definovat vůči konkrétnímu steganalytickému systému $\mathcal{D}$. Pokud se nám podaří určit pravděpodobnost výskytu falešného poplachu $\alpha_{\mathcal{D}}$ (*false alarm probability*) a pravděpodobnost odhalení přítomnosti zprávy $\beta_{\mathcal{D}}$ (*detection probability*), pak můžeme říct, že systém je $\gamma_{\mathcal{D}}$-bezpečný vůči systému $\mathcal{D}$, pokud platí, že:

$$|\beta_{\mathcal{D}} - \alpha_{\mathcal{D}}| \leq \gamma_{\mathcal{D}}$$

kde $0 \leq \gamma_{\mathcal{D}} \leq 1$.

Dále pak můžeme říct, že steganografický systém je dokonale bezpečný vůči steganalytickému systému $\mathcal{D}$, pokud $\gamma_{\mathcal{D}} = 0$ [8].

3.3 Robustnost

Robustnost (*robustness*) určuje obtížnost zničení uschované informace manipulací s médiem. Tento faktor je jedním z nejdůležitějších v oboru zvaném watermarking. Pokud například pošle filmové studio svůj nový film kritikům, ale nepřeje si, aby jej někdo dále šířil, může každou kopii opatřit vodoznakem (*watermark*) identifikujícím kritika, kterému je určena. V případě, že se pak začne film šířit, studio může odhalit viníka přečtením vodoznaku šířené verze filmu. Pokud by však vodoznak nebyl dostatečně robustní a jednalo by se jen o pár prvních redundaních bitů, stačilo by kritikovi před šířením vymaskovat tyto bity a usvědčujícího vodoznaku se tak jednoduše zbavit.

Ideální vodoznak by měl být dostatečně robustní, aby při jeho odstranění došlo ke znehodnocení média, ve kterém je obsažen. Některé vodoznaky jsou natolik robustní, že odolají i geometrickému zkreslování (*geometrical distortion*). Jednou z takových metod je například metoda SIFT (*Scale-Invariance Feature Transform*) [14].

Kapitola 4

Steganalýza

Přestože po seznámení s vlastnostmi steganografických metod v předchozí kapitole by měla logicky následovat část věnovaná metodám samotným, dovolím si odbočku ve formě dvou kapitol věnovaných steganalýze a krycímu objektu.

Pro pochopení výhod a nevýhod steganografických metod je totiž třeba pochopit, jaké prostředky jsou používány pro jejich odhalování. Steganografie zanechává kvůli své invazivní povaze stopy ve steganogramu ve formě odchylek v charakteristikách média. Steganalýza se snaží tyto stopy nalézat pomocí tzv. steganalytických útoků. Statistické útoky lze charakterizovat více způsoby, nejčastěji jsou však rozděleny na vizuální a statistické útoky [26].

4.1 Vizuální útoky

Při vizuálním útoku jsou z média vyfiltrována jen určitá data, například nejméně významný bit (*least significant bit*, dále jako LSB) jednotlivých bytů nebo DCT (*discrete cosine transform* – viz 5.3) koeficienty. Lidským okem je pak potřeba hledat podezřelé charakteristiky v těchto datech a určit tak, zda nebyla původní data narušena vkládanou zprávou.

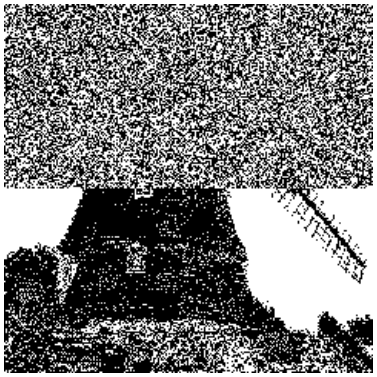
Tento typ útoku je vhodný na odhalování práce steganografických algoritmů operujících v prostorové doméně (viz 6.1), zejména pak na algoritmy využívající LSB. Mnoho autorů steganografických metod se mylně domnívá, že rozložení LSB je zcela náhodné [26]. Pokud vyfiltrujeme z obrázku LSB a zobrazíme jako obrázek, kde budou například jedničky černé pixely a nuly bílé, mohou vznikat textury podobné původnímu obrázku. Přepsáním LSB jinými daty však vznikne šum bez textur.

Znázornění vizuálního útoku je možné vidět na obr. 4.1, obr. 4.2 a obr. 4.3. Na obr. 4.1 je vidět krycí objekt bez jakékoliv analýzy. Obr. 4.2 zobrazuje LSB jednotlivých pixelů původního obrázku (bity s hodnotou 0 jsou znázorněny černou barvou, bity s hodnotou 1 bílou barvou). Obr. 4.3 pak zobrazuje stejný typ znázornění jako obr. 4.2, ovšem pro steganogram vygenerovaný metodou EzStego, který obsahuje zprávu zabírající přibližně polovinu kapacity obrázku.

V tomto konkrétním případě je odhalení porušení charakteristiky LSB obrázku vcelku triviální, avšak v praxi je často potřeba využít mnohem pokročilejších metod. Protože metody popisované v této práci operují ve frekvenční doméně (viz 6.1), kde vizuální útoky většinou pozbývají na účinnosti, nebudu toto téma rozebírat více do hloubky a čtenáře s případným zájmem o více informací odkáži na [26].

Obrázek 4.1: Krycí objekt (převzato z [26])

Obrázek 4.2: LSB krycího objektu (převzato z [26])



Obrázek 4.3: LSB steganogramu metody EzStego (převzato z [26])

4.2 Statistické útoky

Statistické útoky umožňují odhalování steganografie díky invazivním vlastnostem steganografických metod, které mohou narušovat statistické parametry krycího objektu. U steganografických metod pracujících mimo prostorovou doménu je úspěšnost vizuálních útoků prakticky vyloučena, proto se statistické útoky stávají hlavním prostředkem pro odhalování steganografie. Transformace krycího objektu na steganogram může zanechávat různé vizuálně nedetekovatelné nebo jako šum působící deformace. Tyto deformace však mohou způsobovat statistické odchylky. Jedním z prostředků pro statistické útoky je tzv. test dobré shody (*chi-square test*) [17], který porovnává statistické vlastnosti obrázku s vlastnostmi předpokládanými.

Je třeba počítat s tím, že třetí strana se může dostat nejen k samotnému steganogramu, který si účastníci komunikace posílají, ale také může získat některá další data, která mohou pomoci při analýze. Čím více informací třetí strana má, tím vyšší je pravděpodobnost úspěšného steganalytického útoku. Útoky lze rozdělit podle toho, zda může třetí strana přenášená data modifikovat (tzv. *aktivní útok*), nebo z nich pouze extrahovat informace a ponechat je beze změny (tzv. *pasivní útok*). Pasivní útok je mnohem častější, neboť jakékoliv modifikace dat mohou uvést komunikující strany v podezření.

Existuje mnoho steganalytických aplikací. Lze mezi ně zařadit například programy Snort, Crawl a Stegdetect [15].

Kapitola 5

Krycí objekt

Nyní, když má čtenář představu o steganalýze a rozdělení jejích útoků na statistické a vizuální, následuje druhá část avizované odbočky ve formě přiblížení pojmu *krycí objekt*.

Nedílnou součástí implementace steganografického řešení je výběr vhodného krycího objektu. Lze využít prakticky jakýkoliv digitální soubor, jeho struktura však silně ovlivňuje odhalitelnost. Zpráva může být uložena do obyčejného textu (například Baconova šifra), lépe lze však využít soubory binární. Protože nejpoužívanějším médiem při elektronické komunikaci (kromě textu) jsou obrazové soubory, zaměřím se na ně dále ve své práci a vyberu vhodný formát pro použití jako krycí objekt.

Obrazové soubory můžeme rozdělit do dvou základních kategorií — komprimované a nekomprimované. Principy steganografických metod se pak liší v závislosti na tom, do které z těchto kategorií krycí objekt spadá. U bezkompresních formátů jsou nositelem informace většinou nejméně důležité bity s informací o barvě pixelů. V případě komprimovaných obrázků se objevují nové možnosti skrývání informací, jejichž princip však bývá výrazně složitější. Informace nebývají uloženy v jednotlivých pixelech, ale využívají vlastnosti kompresního algoritmu.

5.1 Nekomprimované obrazové soubory

Nekomprimované obrazové soubory lze reprezentovat jako posloupnost pixelů, kde každý je uložený na určitém počtu bitů (tento počet je fixní pro celý soubor). Takovým formátem je například BMP. Soubor s 24bitovým kódováním barev v systému RGB má pro každou ze složek (červená, zelená, modrá) 1 byte, celkem tedy 3 byty na pixel. Pokud u pixelu změníme LSB každé barevné složky, bude změna barvy tak malá, že je prakticky nemožné ji poznat pouhým okem. To umožňuje zapsat 3 bity vlastní informace do každého pixelu. Například do obrázku o rozlišení 1024×768 pixelů s 24bitovým kódováním barev tak můžeme získat 294 912 bytů pro zápis vlastních informací zapsaných do LSB. Taková kapacita umožňuje skrytí 163 normostran textu v ASCII kódování či černobílého obrázku s 256 úrovni šedi, též o rozlišení 1024×768.

Nekomprimované soubory umožňují jednoduše implementovatelné steganografické metody, mají však řadu nevýhod:

Velikost
: Nekomprimované obrazové soubory jsou často příliš velké — jejich ztrátovou kompresí s únosnou ztrátou kvality je můžeme zmenšit cca na 5–10% původního počtu bajtů [21].

Malá rozšířenost

Nekomprimované soubory mají díky zachovávání kvality obrazu využití například ve fotoaparátech a bitmapové grafice určené pro tisk, pro přenos během komunikace jsou však zcela nevyhovující díky své velikosti. Proto nejsou na internetu nekomprimované soubory příliš rozšířené. Jejich použití v komunikaci může vzbudit podezření a dát impuls ke steganalýze.

Jednobarevné plochy

Pokud je obrázek bez šumu (nejedná se tedy o fotografii, ale například reklamní leták vytvořený v grafickém editoru), budou pravděpodobně na obrázku celé plochy tvořené jednou barvou. Pokud však zasáhneme do LSB, mohou se na těchto plochách vytvořit pixely s mírně odlišnou barvou. Tyto pixely sice nejsou vidět pouhým okem, jsou však poměrně snadno odhalitelné statistickou analýzou i vizuálními útoky (viz 4).

5.2 Komprimované obrazové soubory

Kvůli výše zmíněným nevýhodám nekomprimovaných obrazových souborů je vhodné při výběru krycího objektu nalézt vhodnější formát. Nabízí se tak využití komprimovaných formátů. Díky své dominanci na internetu je vhodný formát JPEG. Protože JPEG je ztrátový kompresní algoritmus, je velmi nevhodný pro metody pracující v prostorové doméně. Kvůli ztrátě informací během kvantizace (viz 5.3) je těžké predikovat, jakým způsobem budou ovlivněny samotné pixely, protože změna každého koeficientu může ovlivnit všechny pixely v bloku. Metoda by musela být natolik robustní, aby tyto modifikace ustála.

Tímto se zužuje výběr použitelných steganografických metod v této práci na metody pracující nad obrazovými soubory formátu JPEG operujícími ve frekvenční doméně.

5.3 Komprese JPEG a formát TFIF

V roce 1982 vznikla výzkumná skupina Joint Photographics Experts Group (JPEG), která byla tvořena převážně ze členů skupiny Photographics Experts Group (PEG) a členů komise ISO [21]. Tato skupina se zabývala vývojem metody pro kompresi obrazových souborů. Výsledkem byla standardizace JPEG schématu v roce 1992 pod označením ISO/IEC 109181. Schéma bylo též popsáno jako standard pro e-mailové přílohy v RFC 1341 [5].

Pojem JPEG označuje nejen pracovní skupinu, ale i samotný postup komprese. Pro formát obrazových souborů (běžně s příponou .jpeg nebo .jpg) je korektní název JFIF (*JPEG File Interchange Format*, často je však používán pojem JPEG, který se vžil [21].

Celý proces komprese je složen z několika kroků [21]:

1. transformace z barevných prostorů RGB, CMYK či jiných do prostoru YCbCr,
2. podvzorkování barvonosných složek C_b a C_r,
3. kosinová transformace jednotlivých barevných složek aplikovaná na bloky 8×8 pixelů,
4. kvantování bloků pomocí vypočtených kvantizačních tabulek,
5. kódování kvantovaných koeficientů aritmetickým nebo Huffmanovým kódováním,
6. přidání hlavičky, patičky a dalších dat, vytvoření výsledného JFIF obrázku.

Kapitola 6

Steganografie

6.1 Klasifikace obrazové digitální steganografie

Metody pro obrazovou steganografii mohou operovat buď v prostorové doméně (*spatial domain steganography*), kde se nositelem informace stávají pixely, nebo ve frekvenční doméně (*frequency domain steganography*), kde jsou nositelem informace zpravidla DCT koeficienty.

Jak bylo řečeno v 5.2, metoda použitá v aplikaci, která je cílem této práce, pracuje ve frekvenční doméně. Proto se budu v následujícím textu soustředit pouze na tento typ metod.

6.1.1 Metody obnovující statistické metriky

Metody obnovující statistické metriky (*statistical restoration methods*) [10] nevyužívají všechny redundantní bity pro ukrývání informace. Místo toho využívají některé z těchto bitů pro tzv. korekce koeficientů tak, aby byl histogram DCT koeficientů steganogramu stejný jako histogram krycího objektu. Nevýhodou těchto metod je, že jsou většinou snadno detekovatelné, neboť zachování histogramu neznamená zachování dalších důležitých statistik jak v rámci jednotlivých bloků, tak mezi nimi. Aby byly tyto metody použitelné v praxi, musely by obnovovat všechny důležité charakteristiky, nikoliv pouze DCT histogram. Dodatečné korekce způsobují distorze, které ostatní charakteristiky naopak narušují a paradoxně tak snižují bezpečnost těchto metod [19].

Typickým zástupcem metod obnovujících statistické metriky je metoda OutGuess [18].

6.1.2 Metody zachovávající statistické metriky

Metody zachovávající statistické metriky (*statistics-preserving methods*) [10] jsou podobné metodám popsaných v předchozí podkapitole. Pro zachování metrik však nevyužívají korekcí, ale provádí takové úpravy, aby nebyly metriky vůbec narušeny. Možností, jak tohoto dosáhnout, je více.

První možností je záměna DTC koeficientů, které provádí například algoritmus Steghide. Další možností je zachování modelu DCT koeficientů místo zachování statistik. Zachování modelu využívá tzv. *Model Based Steganography* (MBS) [10].

6.1.3 Metody s minimálním zkreslením

Metody s minimálním zkreslením (*minimal distortion methods*) [10] se snaží využít informace, které jsou dostupné pouze odesílateli k tomu, aby minimalizovaly zkreslení charakte-

ristik obrázku. Často využívají dvojí kvantizace, kde při první kvantizaci získají potřebné informace pro redukci zkreslení. Při druhé kvantizaci se tato data z obrázku ztratí, kodér je má však stále k dospozici a využije je ke zpětné korekci koeficientů.

Mezi zástupce metod s minimálním zkreslením patří metoda upraveného maticového vkládání a metoda s perturbovanou kvantizací.

6.1.4 Heuristické metody

Heuristické metody (*heuristic methods*) [10] jsou neadaptivní metody, které jsou navržené tak, aby vkládání zprávy co nejméně modifikovalo charakteristiky obrázku. Na rozdíl od ostatních popsaných druhů metod se nesnaží po vložení zprávy o žádné zpětné korekce koeficientů, ale spoléhají se na vlastní snížený invazivní charakter.

Charakteristickým znakem heuristických metod bývá snaha o provedení co nejmenšího počtu změn koeficientů a co nejefektivnější využití kapacity krycího objektu. Typickými představiteli jsou metody JP Hide&Seek, F3, F4, F5 a jejich modifikace.

6.2 Steganografické metody

Ve své práci budu implementovat steganografickou metodu, která využívá jako krycí objekt JPEG obrázek. Pro tento účel jsou velmi vhodné metody pracující ve frekvenční doméně, neboť při převodu do JPEG je do frekvenční podoby obrázek převáděn. Všechny metody, které zde uvedu, vkládají informace do krycího média manipulací s LSB kvantizovaných DCT koeficientů. V procesu převodu na JPEG popsaném v kapitole 5.3 tedy probíhá vkládání mezi kroky 4 a 5. Diskrétní kosinová transformace je aplikována na čtverce 8×8 pixelů, stejný rozměr mají tedy i matice koeficientů a kvantizační tabulky.

6.2.1 Steghide

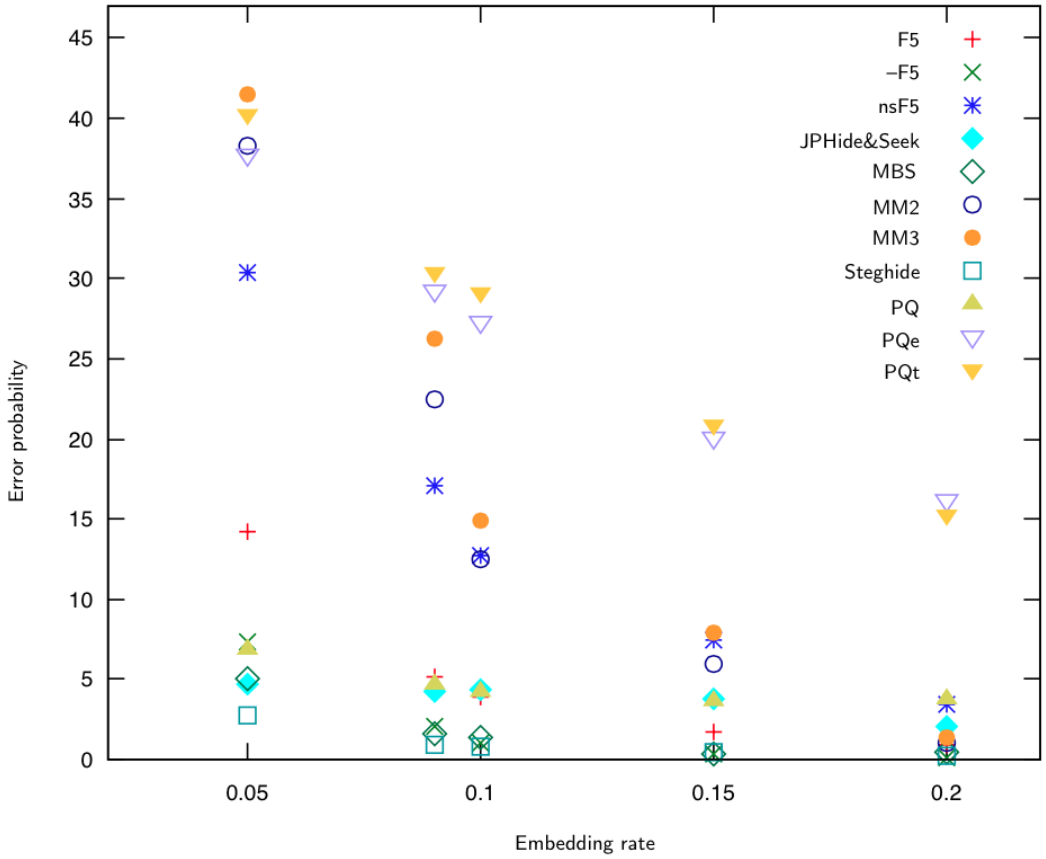
Steghide [12], jehož autorem je Stefan Hetzl, je steganografický program založený na záměně hodnot složek, do kterých je zpráva ukrývána. Dokáže pracovat s audioformáty (AU a WAV) a obrazovými formáty (BMP a JPEG).

Vkládaná data jsou nejdřív zkomprimována a zašifrována. Poté je pseudonáhodným generátorem vygenerována sekvence koeficientů, do kterých bude zpráva vkládána. Steghide poté vyřadí koeficienty, jejichž hodnota se nemění (mají již takovou hodnotu, jako by do nich byla vkládána zpráva). Poté nalezne všechny páry koeficientů, které mohou být zaměněny tak, aby měly LSB stejný, jako bit zprávy. Ve zbylých koeficientech je LSB nahrazen bitem zprávy.

6.2.2 Perturbovaná kvantizace (PQ)

Metoda perturbované kvantizace (*Perturbed Quantisation*) [9] — dále jen PQ — využívá při kódování zprávy informace, které má kodér k dispozici, ale ve výsledném steganogramu se již nenachází. Proces kvantizace DCT koeficientů neprobíhá přesně jako u klasického JPEG kodéru — zaokrouhlování kvantizovaných koeficientů na celočíselnou hodnotu je doplněno o algoritmus, který rozhoduje, zda zaokrouhlit k nižší nebo vyšší celočíselné hodnotě. K tomuto rozhodování využívá informace, které z obrázku získal ještě před kvantizací.

Pro vkládání zprávy do koeficentů získaných upraveným kvantizátorem je použita metoda tzv. *wet paper codes* [9].



Obrázek 6.1: Komparace steganografických metod (převzato z [10])

6.2.3 MMx

Metoda upraveného maticového vkládání (*Modified Matrix Encoding*) [13] — dále jen MMx — využívá pro skrývání zprávy algoritmus maticového vkládání pomocí Hammingova kódu, které však může provést i více než jednu změnu bitu. Tato metoda umožňuje zamezit zkracování (viz 6.2.5), což sice zvyšuje efektivitu vkládání, ale snižuje bezpečnost [10].

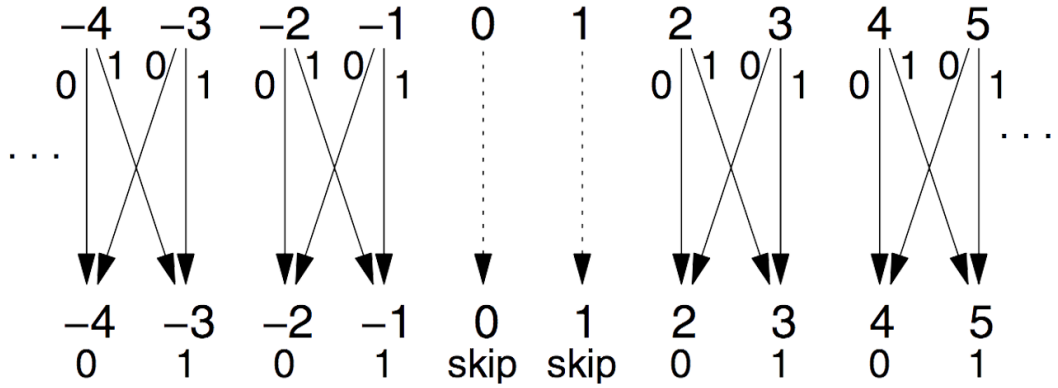
6.2.4 JSteg

Jedním z nejjednodušších algoritmů ve frekvenční doméně je algoritmus JSteg [23], který vytvořil Derek Upham.

Metoda JSteg vybere z krycího objektu všechny kvantizované koeficienty, které nenabývají hodnot 0 nebo 1. LSB každého z těchto koeficientů je nahrazen jedním bitem ze vkládané zprávy. Kapacita této metody se pohybuje okolo 12 % velikosti steganogramu [24].

Na obrázku 6.2 je názorně vidět, jak se mění koeficienty v případě vkládání 0 nebo 1 ze zprávy. Horní řada čísel jsou vkládané koeficienty. Pod nimi je šipkou znázorněna změna na výsledný koeficient podle aktuální hodnoty bitu ze zprávy.

JSteg je sice odolná vůči vizuálnímu útoku, je však snadno odhalitelná statistickým útokem — stačí pohled na histogram výskytu koeficientů (viz obrázek 6.6). Symetrie histogramu podle nuly je narušena a počty koeficientů (kromě 0 a 1, které algoritmus vynechává) se po dvojicích vyrovnávají — vytváří jakési dvouhodnotové schody.



Obrázek 6.2: Kódování LSB v koeficientech u metody JSteg (převzato z [25])

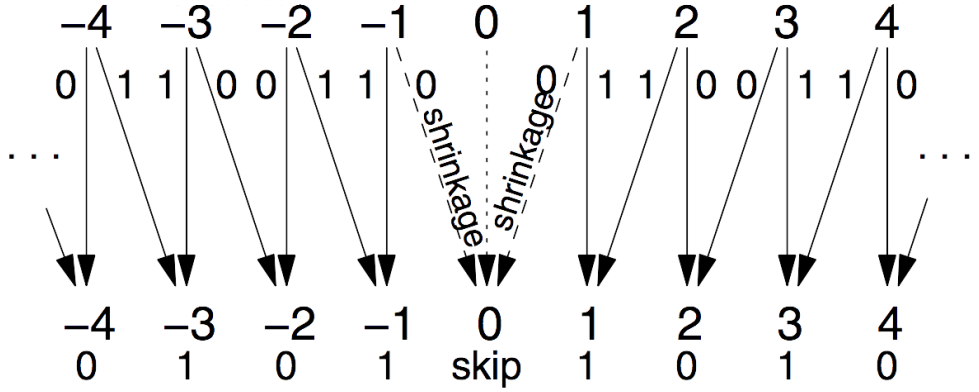
6.2.5 F3

Metoda F3 [24], jejímž autor je Andreas Westfeld využívá podobného principu, jako JSteg. Místo nahrazování LSB koeficientů dekrementuje jejich absolutní hodnotu (posouvá je směrem k nule).

Algoritmus vybere všechny nenulové koeficienty (tedy i koeficenty s hodnotou 1). Vezme první koeficient a první bit zprávy. Pokud je LSB koeficientu stejný jako bit zprávy, ponechá koficient beze změny a pokračuje na další bit zprávy a další koeficient. V opačném případě však *sníží absolutní hodnotu* koeficientu. To znamená, že kladný koeficient je dekrementován a záporný inkrementován.

Pokud je původní koeficient rovný hodnotě −1 nebo 1 a jeho LSB se neshoduje s bitem zprávy, vznikne snížením absolutní hodnoty nulový koeficient. Dekodér však čte bity jen z nenulových koeficientů a takto vložený bit je při odkrývání ignorován, proto musí dojít k opětovnému vložení bitu do dalšího nenulového koeficintu. Tomuto jevu se říká zkracování (*shrinkage*). Zkracování snižuje efektivitu vkládání a v histogramu redukuje počet koeficientů −1 a 1.

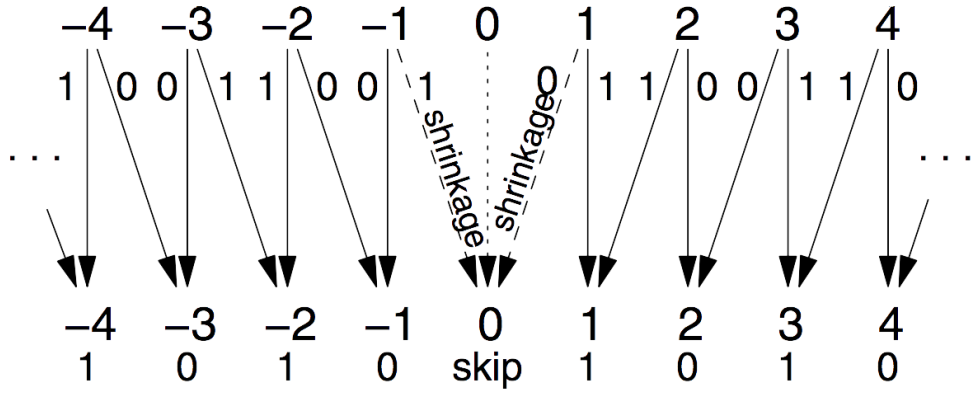
I když tato metoda řeší některé neduhy algoritmu JSteg, stále zanechává histogram, který neodpovídá Gaussově rozložení. Místo aby se počet koeficienů směrem od nuly snižoval, dochází u sudých hodnot k jejich zvyšování.



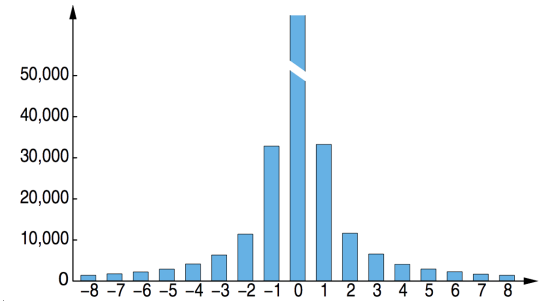
Obrázek 6.3: Kódování LSB v koeficientech u metody F3 (převzato z [25])

6.2.6 F4

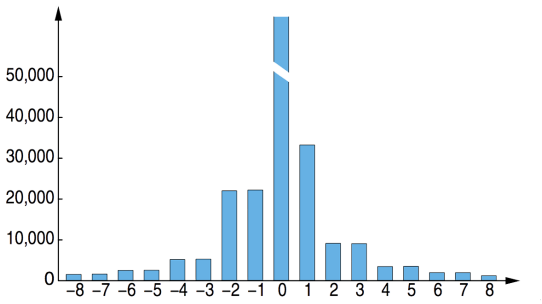
Aby byl histogram podobný Gaussově rozložení v podobě klasického „zvonu“, je potřeba jen mírná modifikace u záporných koeficientů. To dělá metoda F4 [24], která se chová podobně jako F3, ale u záporných koeficientů porovnává LSB s invertovaným bitem zprávy. Pomocí této malé změny se podaří dosáhnout požadované minimalizace změn tvaru histogramu, tedy statistických vlastností.



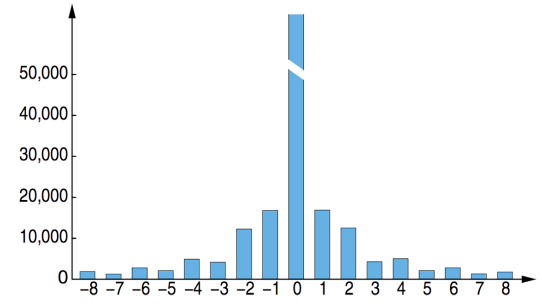
Obrázek 6.4: Kódování LSB v koeficientech u metody F4 (převzato z [25])



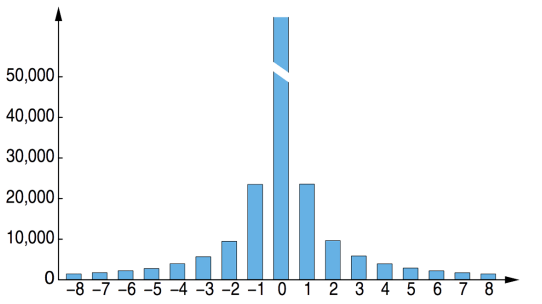
Obrázek 6.5: Histogram krycího objektu (převzato z [24])



Obrázek 6.6: Histogram steganogramu JSteg (převzato z [24])



Obrázek 6.7: Histogram steganogramu F3 (převzato z [24])



Obrázek 6.8: Histogram steganogramu F4/F5 (převzato z [25])

6.2.7 F5

Andreas Westfeld se při tvorbě algoritmu F5 zaměřil na lepší efektivitu kódování a na distribuci změn koeficientů.

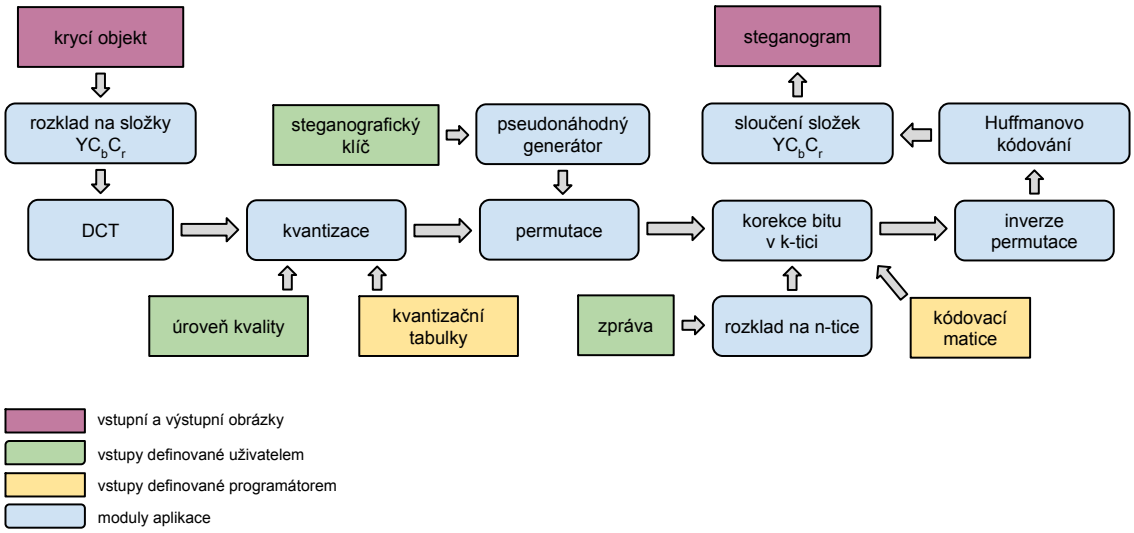
Vylepšené distribuce je docíleno metodou permutativního spojování (*permutative straddling*), které koeficienty před jejich změnami zamíchá pomocí pseudonáhodného generátoru a steganografického klíče. To navíc zvyšuje zabezpečení metody, protože i když bude znát třetí strana inverzní algoritmus, bez znalosti hesla nebude schopna určit pořadí čtení koeficientů. Po vložení zprávy jsou koeficienty znovu seřazeny do původního pořadí, takže změny nejsou umístěny na začátku souboru, ale jsou nepravidelně distribuovány do celého obrázku.

Efektivita vkládání zprávy je v metodě F5 vylepšena díky maticovému vkládání, jak uvádí samotný autor metody: „*Ron Crandall popsal možnost maticového vkládání jako novou techniku k vylepšení efektivity vkládání. F5 je asi první implementace maticového vkládání. Pokud je většina kapacity steganogramu nevyužita, maticové vkládání snižuje počet potřebných změn.*“[24]

Teoretická efektivita kódování metody F5 je určená následujícím vzorcem [10], kde p značí počet pixelů v obrázku.

$$e = p/(1 - 2^{-p})$$

Kvůli zkracování (viz 6.2.5) však v praxi této efektivity nelze dosáhnout, reálná hodnota se jí vždy bude jen limitně blížit.



Obrázek 6.9: Znázornění průběhu algoritmu F5

Kapitola 7

Analýza řešení

Cílem mé práce je vytvoření mobilní aplikace pro platformu Apple iOS, která uživateli umožní za pomocí steganografické metody ukrýt zprávu do libovolného obrázku a tento obrázek odeslat e-mailem. Aplikace rovněž umožňuje inverzním algoritmem této steganografické metody z obrázku zprávu znovu získat. Díky této mobilní aplikaci mohou uživatelé e-mailem posílat ve formě obrázku tajné zprávy, jejichž přítomnost nebude odhalena třetí stranou.

V nadcházející kapitole se budu věnovat cílové platformě a popisem knihoven a technologií, které jsou v aplikaci použity. Závěrečná část kapitoly je věnována výběru vhodné steganografické metody a návrhu struktury aplikace.

7.1 Apple iOS

Operační systém iOS je odlehčenou verzí systému OS X od firmy Apple Inc. pro mobilní zařízení iPhone a iPad. Tento systém je po Androidu druhým nejrozšířenějším mobilním operačním systémem.

Součástí systému iOS je množství frameworků, které vývojářům zpřístupňují mnohé funkce pro práci s 2D obrazem, zvukem, systémovými komponentami apod. Z těchto frameworků využívám ve své aplikaci *UIKit*, *Foundation* a *Core Graphics*.

7.2 Frameworky užitečné pro aplikaci

Tři výše zmiňované frameworky obsahují třídy a funkce, které v mé aplikaci využívám, proto s nimi čtenáře podrobněji seznámím. Informace jsou čerpány z oficiální dokumentace společnosti Apple [4]. Výčet frameworků je doplněn také o knihovnu *OpenCV*, která však není vydaná firmou Apple — jedná se o open-source projekt.

7.2.1 Framework UIKit

Framework UIKit poskytuje zásadní infrastrukturu potřebnou pro vytváření a správu aplikací pro iOS. Obsahuje funkce pro vytváření uživatelského rozhraní, řízení událostí aplikace a aplikační model potřebný k obsluze hlavní běhové smyčky. Funguje také jako vrstva umožňující vývojářům přístup ke komponentám systému jako je fotoaparát, geolokace atd.

7.2.2 Framework Foundation

Framework *Foundation* definuje základní vrstvu tříd pro Objective-C. Poskytuje vývojářům sadu tříd jako např. `NSString`, `NSArray`, `NSDictionary` apod., které obsahují funkce pro práci s pokročilými abstraktními datovými typy. Ze všech těchto tříd vychází také jejich měnitelné verze (*mutable*), které lze za běhu modifikovat (přidávání prvků, mazání prvků atd.). Měnitelnými třídami jsou např. `NSMutableString`, `NSMutableArray` či `NSMutableDictionary`.

7.2.3 Framework Core Graphics

Framework *Core Graphics* je postavený na vrstvě systému OS X a iOS jménem *Quartz*, která zajišťuje vykreslování grafiky. Obsahuje funkce pro nízkoúrovňovou práci s obrazem.

7.2.4 Knihovna OpenCV

OpenCV z anglického *Open Souce Computer Vision* [11] je (jak název napovídá) otevřená knihovna s licencí BSD dostupná pro jazyky C, C++, Python a Java. Jejím autorem je Willow Garage z firmy Intel a hlavním cílem je zjednodušit vývojářům především zpracování a rozpoznání obrazu v reálném čase. Pro platformu iOS je dostupná ve formě statické knihovny (viz 7.3.1).

7.3 Verze systému iOS 8

V září 2014 byla vydána verze systému iOS 8. Přišla se třemi velkými novinkami pro vývojáře — možnost vytvářet (a případně distribuovat) vlastní frameworky, vytvářet rozšíření aplikací a především pak nový programovací jazyk Swift. Poslední stabilní verze systému v době psaní této práce je verze 8.3 vydaná v dubnu 2015 [3].

7.3.1 Vlastní frameworky

Protože vývojáři často potřebují některé části kódu využívat vícekrát napříč několika aplikacemi, přišel Apple s možností vytvářet vlastní knihovny, které jsou ve formě frameworků zkompilované do jediného souboru. To přináší vývojářům řadu výhod [1].

Rychlost kompilace
: Protože zdrojový kód frameworku je kompilován a jeho veřejné funkce jsou pak dostupné přes vývojářem definované rozhraní, stačí jej zkompilovat jednou a při jeho využívání v dalších aplikacích pak již není kompilace potřebná. Vývojáři tak mohou ve svých aplikacích využívat masivní frameworky, přičemž kompilují jen zdrojové kódy své aplikace. Tento přístup samozřejmě není ve světě IT žádnou novinkou, většina programátorů jistě zná například předkompilované části kódu v jazyce C umožňující jeho modularitu.

Ochrana proti plagiátorství
: Protože frameworky je možné distribuovat i jiným vývojářům, je každý framework během kopilace šifrován unikátním klíčem vývojáře. To sice znamená, že vývojář vytvářející framework musí být registrovaným vývojářem v komunitě Apple Developers (a platit si tak každý rok licenci), jeho kód je však mnohem lépe chráněný před

reverse-engineeringem. Každý, kdo má k dispozici jeho framework, jej sice může využívat ve své aplikaci, ale může jej však využívat pouze jako celek chráněný autorskými právy a nemůže získat zdrojový kód, který by pak mohl zneužívat. Nikdo tak nemůže odhalit, jak fungují obsažené algoritmy ani vydávat části zdrojového kódu za své.

Redukce duplikace kódu
Vývojáři často opakovaně používají svůj zdrojový kód ve více aplikacích, což však bez využití frameworků vede k duplikaci kódu. Ta snižuje čitelnost i udržovatelost kódu, což se daří řešit díky možnosti verzování frameworků.

Frameworky byly již dříve dostupné pro platformu OS X. Pro iOS byly k dispozici pouze frameworky vytvořené přímo firmou Apple. Od verze iOS 8 je však může vytvářet každý licencovaný vývojář, což umožňuje vytvořit framework s jádrem aplikace, který pak bude využívat kodér a dekodér. Před příchodem frameworků měli vývojáři k dispozici pouze možnost vytváření statických knihoven (tzv. *Cocoa Touch Static Library*), které jsou však náročnější na zprovoznění — je potřeba je při kompilaci složitě linkovat.

7.3.2 Rozšíření aplikací

Od verze iOS 8.0 mohou vývojáři psát rozšíření, která je možné spouštět i mimo aplikace. Je tak možné implementovat funkce, které mohou být spouštěny i z jiných částí systému mimo rámec rozšiřované aplikace. Mohou tak vznikat rozšíření umožňující sdílení na sociální sítě, vlastní klávesnice atd. V dekodéru mé aplikace využívám *Action Extension*, které může být aktivováno v e-mailové aplikaci telefonu a zobrazí zprávu odkrytou z vybrané přílohy.

7.3.3 Programovací jazyky

Hlavním programovacím jazykem pro iOS do verze 7 včetně byl jazyk Objective-C, který je objektovou formou jazyka C doplněnou o systém zasílání zpráv inspirovaný jazykem Smalltalk. Jeho derivátem je pak jazyk Objective-C++, který podporuje i jazykové konstrukce z C++.

Objective-C byl využíván od roku 1983 pro psaní aplikací pro systém Mac OS (nyní OS X) a iOS. Na výroční konferenci WWDC 2014 však Apple představil zcela nový programovací jazyk Swift, který by měl jazyk Objective-C nahradit. Swift se drží modernějších programovacích paradigmat a svou syntaxí připomíná spíše skriptovací jazyk. Smalltalkový systém zpráv je nahrazen klasickou objektovou tečkovou notací a jazyk je doplněný o syntaktický cukr umožňující uživateli provádět běžné operace mnohem kratším a čitelnějším zápisem [2].

Velkou výhodou jazyka Swift je, že stejně jako Objective-C je kompilován LLVM kompilátorem, díky čemuž mohou vývojáři v jedné aplikaci kombinovat C, C++, Objective-C i Objective-C++. Kombinovat všechny tyto jazyky zároveň samozřejmě není ideální postup, jde spíše u umožnění postupného přestupu na Swift. Jádro aplikace tak může zůstat napsané například v Objective-C a vývojáři nemusí plýtvat čas přepisem do Swiftu. Ve Swiftu pak mohou psát nové komponenty.

Všechny aplikace pro platformu iOS je potřeba psát ve vývojovém prostředí Xcode (až na výjimky jmenované dále), které je však dostupné pouze pro platformu OS X. Uživatelé jiných systémů mají k dispozici alternativy jako například kompilace Flash aplikací pro iOS či open-source framework PhoneGap vytvářející iOS aplikace z aplikací napsaných v HTML, CSS a JavaScriptu, jejich využívání však není ze strany firmy Apple příliš podporované

a kdykoliv se může stát, že budou v licenčním ujednání zakázány. Z těchto a mnoha dalších důvodů považuji za lepší využívat prostředí Xcode pro psaní nativních iOS aplikací.

7.4 Volba implementované metody

Z možných známých steganografickýcm metod operujících ve frekvenční doméně s obrázky ve formátu JPEG považuji za nejvhodnější heuristické metody.

Metody snažící se o korekci charakteristik obrázku mohou sice určité charakteristiky (například statistické rozložení DCT koeficientů) opravit tak, aby nebylo vkládání patrné, často u nich však hrozí riziko zkreslení jiných charakteristik [10].

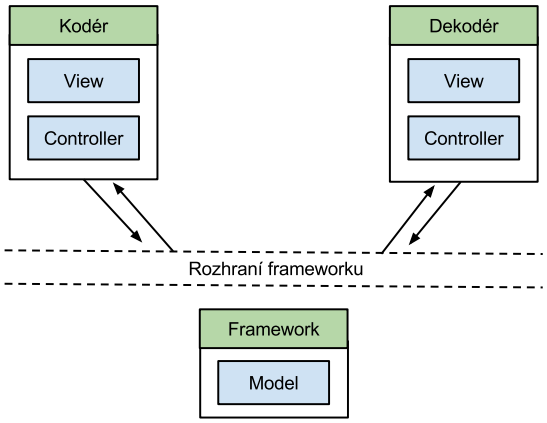
Metody s minimálním zkreslením potřebují pro své fungování informace nedostupné příjemci, což však vyžaduje buď obrázek v nekomprimované podobně nebo dvojí kvantizaci [10]. Systém iOS nenabízí možnost přístupu k nekomprimované formě obrázku a dvojí kvantizací ztrácí obrázek na kvalitě.

Metoda F4 vyhovuje svou odhalitelností i kapacitou [24] požadavkům mé aplikace. Její nástupce F5 sice vylepšuje efektivitu vkládání, avšak za cenu vyšší spotřeby výpočetního výkonu, který je na mobilních zařízeních cenným zdrojem. Proto v aplikaci implementuji metodu F4.

7.5 Struktura aplikace

Protože celá aplikace sestává ze dvou hlavních částí — kodéru a dekodéru, je je rozdělen na tyto části i celý projekt. Obě tyto komponenty však využívají množinu podobných či dokonce stejných funkcí, takže je jádro aplikace odděleno do samostatného celku — frameworku (viz 7.3.1).

Aplikace je rozdělena podle schématu MVC (*Model-View-Controller*), které umožňuje oddělit logiku aplikace (model), rozhraní pro uživatele (view) a interakci s uživatelem (controller). Protože logika je od kodéru i dekodéru separována do frameworku, vzniká hybridní struktura zobrazená na obrázku 7.1.



Obrázek 7.1: Znázornění hybridní MVC struktury aplikace

Kapitola 8

Implementace

8.1 Hlavní komponenty aplikace

8.1.1 Frameworky

Jak bylo řečeno v kapitole 7.5, množina funkcí, které tvoří algoritmus vkládání a odkrývání zprávy, je oddělena do frameworku, který je dostupný v kodéru i dekodéru. Toto uspořádání je však pouze programátorská konvence zvyšující přehlednost a udržitelnost zdrojového kódu a na samotný algoritmus nemá vliv. V následující části jsou popsány jednotlivé kroky skrývání, resp. extrahování zprávy tak, jako by byly programátorsky součástí kodéru, resp. dekodéru, přestože jsou ve skutečnosti v samostatném celku mimo tyto komponenty. Tento framework jsem pojmenoval *StegoKit* (koncovka *Kit* je konvencí společnosti Apple).

Jak kodér, tak i dekorér využívají kromě *StegoKit*u také v 7.2 popisované frameworky *UIKit*, *Foundation*, *Core Graphics* a statickou knihovnu *OpenCV*.

8.1.2 Kodér

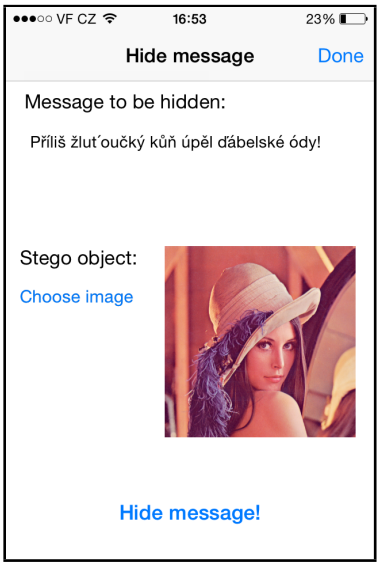
Klíčová komponenta celé aplikace, která uživateli umožňuje vložit zprávu, vybrat obrázek a vygenerovat výsledný steganogram, je vytvořena jako samostatná aplikace pro iOS s názvem *Stego*.

Aby bylo možné použít aplikaci na mobilním zařízení, je nezbytné, aby měla grafické uživatelské rozhraní. K tvorbě uživatelského rozhraní je použit systém tzv. storyboardů (*storyboards*). Storyboardy definují prvky grafického rozhraní (*view*) a jejich napojení na controller. Storyboard aplikace kodéru (viz obr. 8.4) je pojmenován `Main.storyboard`, controller obrazovky se zadáváním vstupů se jmenuje `HideViewController.m` a controller pro obrazovku s výsledným steganogramem nese název `HideResultViewController.m`.

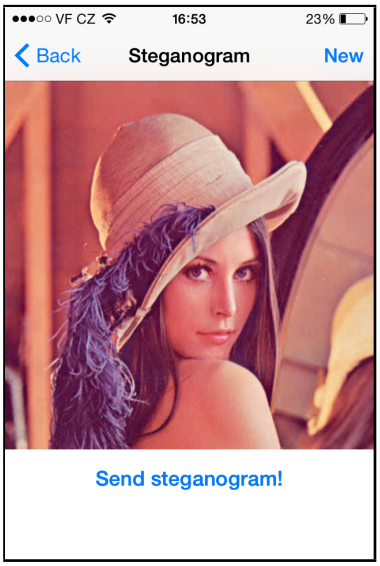
Průběh aplikace *Stego* je velmi podobný průběhu klasického JPEG kodéru popsaného v 5.3 a probíhá následovně.

Načtení vstupu

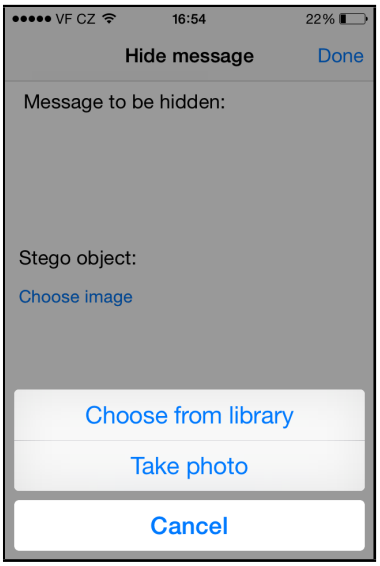
Vstupem zvolené metody F4 je zpráva a krycí objekt. Pro zprávu je určen prvek `UITextView` z frameworku *UIKit*, ze kterého jsou použity i ostatní prvky uživatelského rozhraní. Všechny dále uvedené třídy s předponou `UI` jsou součástí *UIKit*u. Pro výběr krycího objektu je uživateli po kliku na tlačítko třídy `UIButton` nabídnuta možnost vybrat obrázek z jeho knihovny nebo vyfotografovat nový snímek (viz obr. 8.3). Obě tyto akce pak zajišťuje třída `UIImagePickerController`.



Obrázek 8.1: Rozhraní aplikace Stego s vloženou zprávou a krycím objektem



Obrázek 8.2: Zobrazení výsledného steganogramu s ukrytou zprávou



Obrázek 8.3: Možnost výběru fotografie knihovny nebo pořízení nového snímku

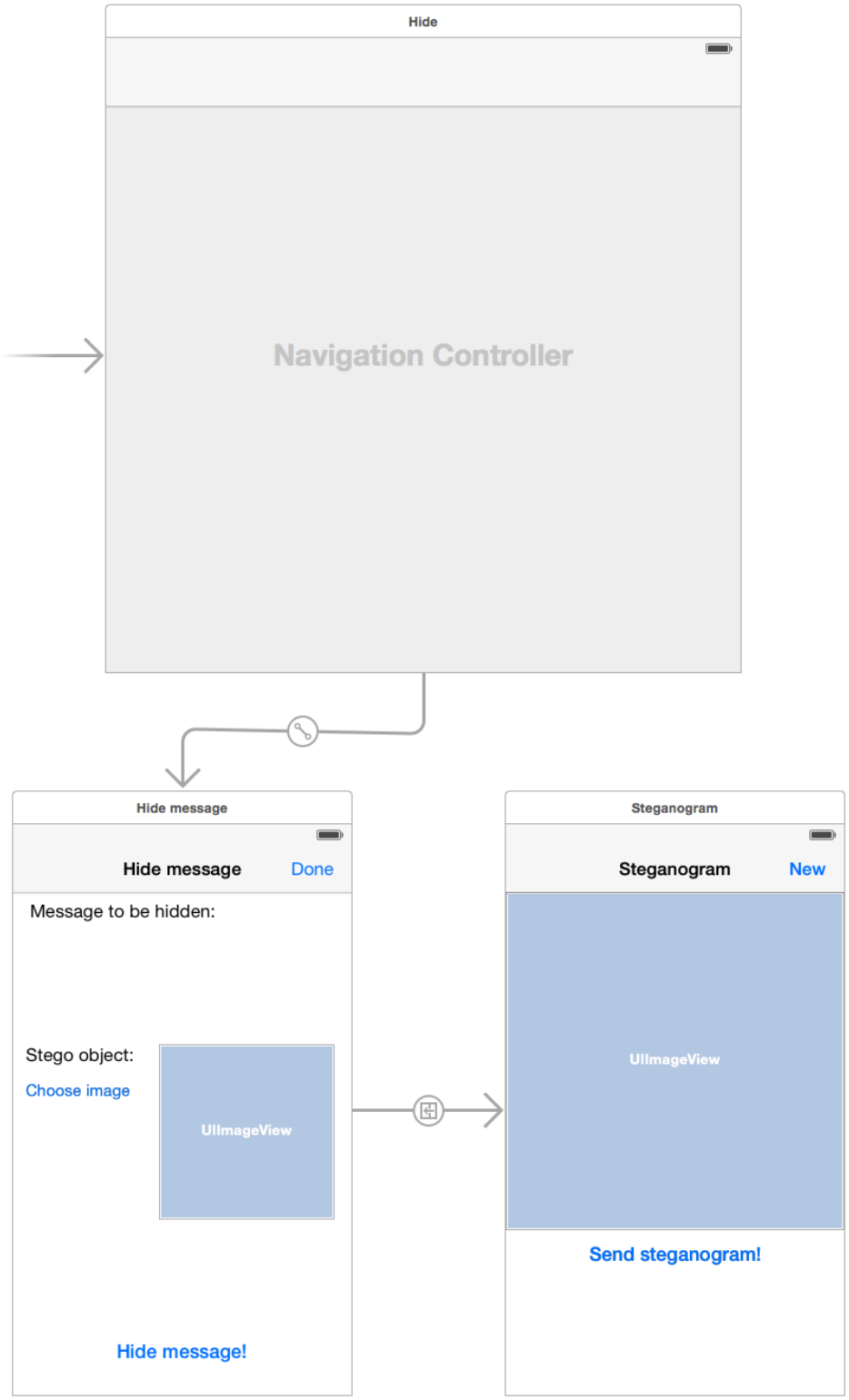
Konverze krycího objektu

Protože algoritmus kodéru využívá knihovny OpenCV, která neumí pracovat s datovými typy obsaženými ve standardních Apple frameworcích, je potřeba obrázek nejdříve konvertovat na datový typ `cv:Mat`. O to se stará funkce `UIImageToMat` ze *StegoKit*u, která krycí objekt převede z typu `UIImage` na `cv:Mat` pomocí funkcí `CGBitmapContextCreate` a `CGContextDrawImage` z frameworku *Core Graphics*.

Během konverze dochází také ke zmenšení obrázku, pokud je jeho šířka nebo výška větší než 1600 pixelů. Velikost obrázků pořízených fotoaparátem iPhonu totiž může být až 8 Mpix, což by při zachování této velikosti způsobovalo příliš dlouhý průběh skrývání a při odesílání steganogramu by mohl systém iOS uživateli nabídnout zmenšení odesílaného souboru. Výzva ke zmenšení souboru je od určité velikosti obrázku automatická a nelze ji programátorsky potlačit. Pokud by uživatel se zmenšením obrázku souhlasil, došlo by k opětovné kompresi a tím i ke zničení zprávy ještě před jejím odesláním.

Transformace barevného prostoru

Protože načtený steganogram je vždy definován v barevném prostoru RGB a komprese JPEG pracuje se luminiscenčními a chrominančními složkami, je potřeba provést transformaci barevného prostoru z RGB do YCbCr. Tuto transformaci mezi barevnými prostory zajišťuje funkce `cv:cvtColor` z knihovny *OpenCV*.



Obrázek 8.4: Storyboard s definicí grafického rozhraní kodéru

Kvantování bloku

Důležitým krokem každého JPEG kodéru je kvantování, které nejvíce ovlivňuje míru komprese a tím i velikost výsledného obrázku. Kvantizace probíhá tak, že DTC koeficienty jsou celočíselně děleny hodnotami v kvantizačních tabulkách. V tomto kroku je velké množství koeficientů vynulováno. To umožňuje efektivní zápis Huffmanovým kódováním.

Rozdělení na bloky 8×8 pixelů

Komprese JPEG zpracovává obrázek po blocích 8×8 nebo 16×16 pixelů. V kodéru jsem zvolil velikost 8×8, což vyžaduje zpracovávat obrázek postupně po blocích o této velikosti. Následující popsané kroky až po Huffmanovo kódování jsou umístěny uvnitř cyklu, který ze steganogramu postupně kopíruje bloky 8×8 do matice, se kterou pak jednotlivé kroky pracují. Bloky jsou z krycího objektu kopírovány nejdříve z levého horního rohu a kopírování pokračuje až pravému hornímu rohu. Takto cyklus pokračuje po řádcích, až k pravému dolnímu rohu.

Pokud šířka, resp. výška krycího objektu není celočíselně dělitelná osmi, je obrázek doplněn o spodní, resp. pravý okraj, ve kterém mají všechny pixely nulovou hodnotu luminiscenční i chrominanční složky. Před vytvořením výsledného steganogramu je tento okraj oříznut, aby byla zachována velikost původního obrázku.

Kosinová transformace bloku

Po transformaci barevného prostoru na YCbCr jsou hodnoty všech prvků zpracovávané matice v intervalu $\langle 0, 1 \rangle$, proto je před kosinovou transformací potřeba nejdříve všechny prvky matice vynásobit konstantou 255 (čímž dojde k rozšíření intervalu na $\langle 0, 255 \rangle$) a odečíst konstantu 128 (posunutí intervalu na $\langle -128, 127 \rangle$) [22]. Poté je možné provést kosinovou transformaci, kterou zajišťuje funkce `cv:dct` z knihovny *OpenCV*. Výstupem kosinové transformace jsou DTC koeficienty.

Skrytí zprávy

Kvantizované DTC koeficienty jsou nositelem zprávy, proto následuje vkládání zprávy, které klasický JPEG kodéru neprovádí. Implementovaná metoda F4 prochází kontinuálně všechny nenulové kvantované DCT koeficienty od prvního po poslední a porovnává jejich hodnoty LSB s bity vkládané zprávy. Pokud se bity neshodují, je dekrementována absolutní hodnota koeficientu, díky čemuž se hodnota LSB a vkládaného bitu začnou rovnat. V případě záporných koeficientů probíhá porovnávání s invertovaným bitem zprávy.

Pokud je během vkládání koeficient vynulován (jeho hodnota byla -1 nebo 1 a absolutní hodnota byla dekrementována), dochází ke zkracování (viz 6.2.5) a bit zprávy musí být vložen znovu do dalšího koeficentu.

Huffmanovo kódování

Efektivní uložení kvantovaných DTC koeficientů (obohacených o vloženou zprávu) probíhá pomocí Huffmanova kódování. Během inicializace kodéru si aplikace vytváří slovník s kódovými slovy a jejich Huffmanovým kódem. Hodnoty Huffmanových tabulek jsou převzaty z doporučených tabulek skupiny JPEG a jsou ukládány do objektu třídy `NSDictionary` z frameworku *Foundation*.

Vytvoření výsledného steganogramu

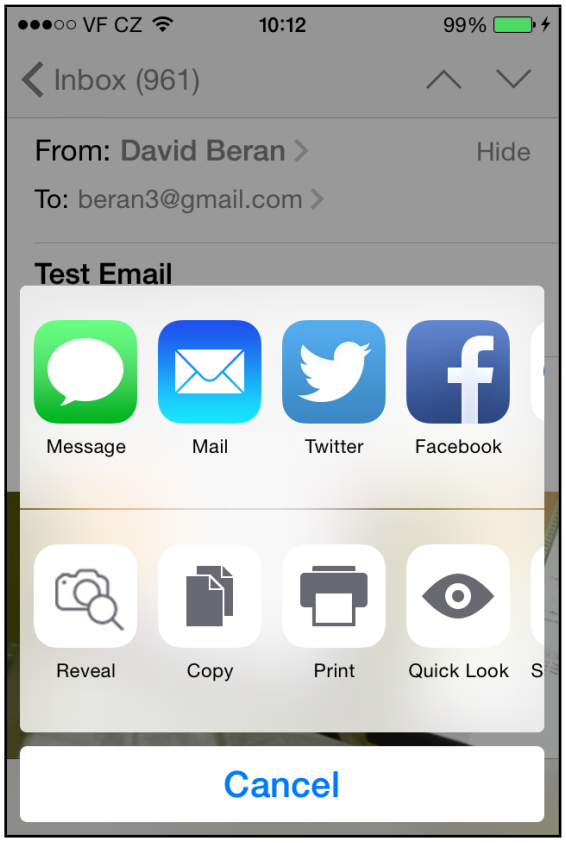
Aby bylo možné výsledný steganogram přečíst a zobrazit, je třeba vytvořit binární soubor, který splňuje standard JFIF. Pro tvorbu tohoto souboru využívám vlastní třídu `BitReader`, která soubor generuje do proměnné typu `NSMutableData` (třída z frameworku *Foundation* pro binární data) a obsahuje funkce pro připojování JFIF značek a dat v binární podobě do této proměnné.

Po zapsání všech dat i značek do proměnné získám její neměnitelnou verzi třídy `NSData`, která je pak konvertována funkcí `initWithData` z frameworku *UIKit* na datový typ `UIImage`.

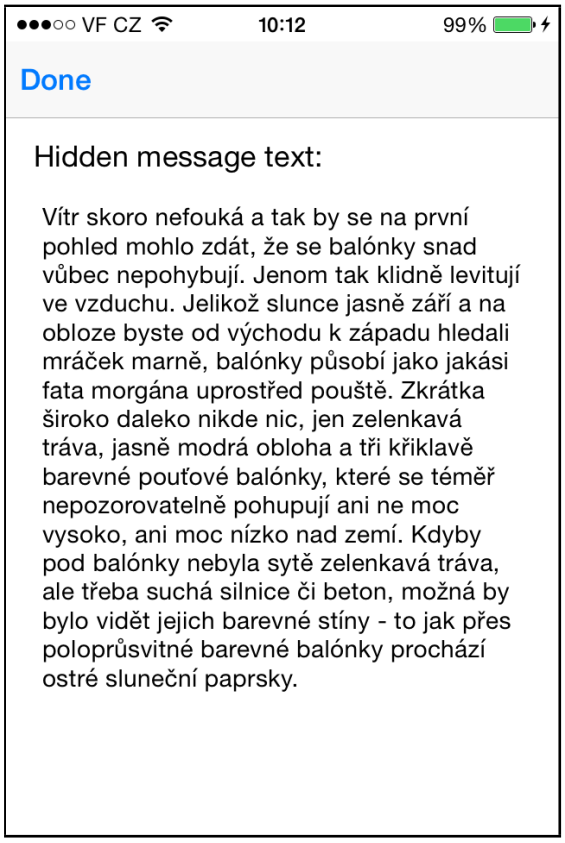
Zobrazení výstupu

Data ve formátu `UIImage` jsou v systému iOS klasickým datovým typem pro reprezentaci obrázků. Steganogram je tak možné zobrazit v uživatelském rozhraní a uživateli je nabídnuta možnost obrázek odeslat e-mailem.

Třída `MFMailComposeViewController` z frameworku *Foundation* umožňuje vyvolání okna pro vytvoření a odeslání e-mailu. Voláním funkce `addAttachmentData`, které je předán jako parametr výsledný steganogram datového typu `NSData`, je připojen steganogram jako příloha. Uživatel pak může vyplnit libovolného příjemce, předmět i obsah zprávy a celý e-mail odeslat.



Obrázek 8.5: Nabídka spustitelných zařízení z e-mailové aplikace



Obrázek 8.6: Rozšíření Reveal s odkrytou zprávou

8.1.3 Dekodér

K extrahování zprávy ze steganogramu slouží rozšíření aplikace (viz 7.3.2) pojmenované *Reveal*. Je možné jej spustit v e-mailové aplikaci delším podržením na obrázek v příloze a výběrem rozšíření Reveal. Nabídka spustitelných rozšíření je vidět na obr. 8.5.

Algoritmus odkrývání zprávy vyžaduje nenulové koeficienty získané z obrázku. Pro získání těchto koeficientů jsem použil JPEG dekodér v jazyce C++, který vytvořil Rich Geldreich a zveřejnil pod licencí *public domain*. Modifikoval jsem tento dekodér tak, aby byly všechny nenulové koeficienty při dekódování obrázku ukládány do datového typu `vector<int>` a tento vektor pak předávám funkci `decode` ve frameworku *StegoKit*. Přečtením LSB těchto bitů a jejich uložením do proměnné třídy `NSData` získám binární hodnotu se zprávou. Bity ze záporných koeficientů jsou před zápisem invertovány.

Data se zprávou jsou následně konvertována na datový typ `NSString` a zobrazena v grafickém rozhraní uživateli. Toto rozhraní i se zprávou je vidět na obr. 8.6.

Kapitola 9

Závěr

Tato práce analyzuje a porovnává dostupné metody obrazové digitální steganografie. Cílem je vybrat vhodnou metodu pro mobilní platformu a provést její implementaci. Po srovnání vlastností dostupných metod a možností mobilní platformy iOS jsem zvolil implementaci metody F4, jejímž autorem je Andreas Westfeld. Navazuji tak na jeho práci a metoda dosud dostupná pouze pro desktopová zařízení s interpretem jazyka Java je díky mé práci dostupná i pro systém iOS.

Jedním z vytyčených cílů je také vytvoření vhodného uživatelského rozhraní. Aplikace Stego nabízí uživatelům intuitivní grafické uživatelské rozhraní, díky kterému mohou jednoduše skrývat zprávy do uložených i nově pořízených fotografií. Pomocí aplikačního rozšíření mohou uživatelé zprávu odkrývat přímo z libovolné e-mailové aplikace na jejich mobilním zařízení. Steganografická metoda F4 je tak nejen dostupná širšímu spektru uživatelů, ale její použití pro komunikaci je i mnohem komfortnější.

Algoritmus F4 aplikace Stego a jejího rozšíření Reveal je možné vylepšit na algoritmus F5 či některou z jeho modifikací. Musí však být brán ohled na nižší výpočetní výkon mobilních zařízení. Pravděpodobně by tedy společně s úpravami měly být provedeny i náležité optimalizace pro zlepšení rychlosti skrývání.

Literatura

[1] Apple Inc.: Přednáška „Building Modern Frameworks“ na konferenci WWDC 2014. 2014.

[2] Apple Inc.: *The Swift Programming Language*. 2014.

[3] Apple Inc.: iOS SDK Release Notes for iOS 8.3 [online]. `https://developer.apple.com/library/ios/releasenotes/General/RN-iOSSDK-8.3/`, 2015-04-08 [cit. 2015-04-27].

[4] Apple Inc.: iOS Technology Overview [online]. `https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iOSTechOverview.pdf`, [cit. 2015-04-15].

[5] BORENSTEIN, N.; FREED, N.: MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies [online]. `http://www.ietf.org/rfc/rfc1341.txt`, 1992 [cit. 2013-10-23].

[6] CACHIN, C.: An Information-Theoretic Model for Steganography [online]. `http://www.zurich.ibm.com/~cca/papers/stego.pdf`, 2004-05-04 [cit. 2015-03-05].

[7] CHALLITA, K.; FARHAT, H.: Combining Steganography and Cryptography: New Directions [online]. `http://www.sdiwc.net/digital-library/web-admin/upload-pdf/00000017.pdf`, 2011 [cit. 2014-09-20].

[8] CHANDRAMOULI, R.; MEMON, N.: Steganography Capacity: A Steganalysis Perspective [online]. `https://isis.poly.edu/~steganography/pubs/spie03.pdf`, 2003-07-20 [cit. 2015-03-05].

[9] FRIDRICH, J.; GOLJAN, M.; SOUKAL, D.: Perturbed Quantization Steganography with Wet Paper Codes. In *Proceedings of the 2004 Workshop on Multimedia and Security*, New York, NY, USA: ACM, 2004, ISBN 1-58113-854-7, s. 4–15.

[10] FRIDRICH, J.; PEVNÝ, T.; KODOVSKÝ, J.: Statistically undetectable JPEG steganography: Dead Ends, Challenges, and Opportunities [online]. `http://www.ws.binghamton.edu/fridrich/research/fraction03.pdf`, 2007 [cit. 2014-02-15].

[11] Garage, W.: OpenCV [online]. `http://opencv.org/`.

[12] HETZL, S.: Steghide – manual [online]. `http://steghide.sourceforge.net/documentation/manpage.php`, [cit. 2015-01-20].

[13] KIM, Y.; DURIC, Z.; RICHARDS, D.: Modified Matrix Encoding Technique for Minimal Distortion Steganography. In *Proceedings of the 8th International Conference on Information Hiding*, IH'06, Berlin, Heidelberg: Springer-Verlag, 2007, ISBN 978-3-540-74123-7, s. 314–327.

[14] LUO, H.; SUN, X.; YANG, H.; aj.: A Robust Image Watermarking Based on Image Restoration Using SIFT. *Radioengineering*, ročník 20, č. 2, 2011, ISSN 1210-2512.

[15] OREBAUGH, A. D.: Steganalysis: A Steganography Intrusion Detection System [online]. `http://www.securityknox.com/Steg_project.pdf`, [cit 2014-10-23].

[16] PROVOS, N.: Defending Against Statistical Steganalysis. In *10th USENIX Security Symposium*, 2001, s. 323–335.

[17] PROVOS, N.; HONEYMAN, P.: Detecting Steganographic Content on the Internet. Technická zpráva, In ISOC NDSS'02, 2001.

[18] PROVOS, N.; HONEYMAN, P.: Hide and Seek: An Introduction to Steganography. *IEEE Security and Privacy*, ročník 1, č. 3, Květen 2003: s. 32–44, ISSN 1540-7993.

[19] SHI, Y. Q.; CHEN, C.; CHEN, W.: A Markov Process Based Approach to Effective Attacking JPEG Steganography. In *Proceedings of the 8th International Conference on Information Hiding*, IH'06, Berlin, Heidelberg: Springer-Verlag, 2007, ISBN 978-3-540-74123-7, s. 249–264.

[20] STANLEY, C. A.: Pairs of Values and the Chi-squared Attack [online]. `http://orion.math.iastate.edu/dept/thesisarchive/MSCC/CStanleyMSSS05.pdf`, 2015-05-01 [cit. 2015-03-09].

[21] TIŠNOVSKÝ, P.: Grafické formáty: JPEG - král rastrových grafických formátů? [online]. `http://www.root.cz/clanky/jpeg-kral-rastrovych-grafickych-formatu/`, 2006-12-07 [cit. 2013-10-03].

[22] TIŠNOVSKÝ, P.: Programujeme JPEG: Kvantizace DCT koeficientů [online]. `http://www.root.cz/clanky/programujeme-jpeg-kvantizace-dct-koeficientu/`, 2007-01-11 [cit. 2015-01-19].

[23] UPHAM, D.: JSteg README [online]. `http://zooid.org/ paul/crypto/jsteg/README.jsteg`, [cit. 2014-10-21].

[24] WESTFELD, A.: F5 — A Steganographic Algorithm: High Capacity Despite Better Steganalysis [online]. `http://f5-steganography.googlecode.com/files/F5%20Steganography.pdf`, 2001 [cit. 2014-02-03].

[25] WESTFELD, A.: F5 — A Steganographic Algorithm [online]. `http://www2.htw-dresden.de/~westfeld/publikationen/f5.pdf`, 2001 [cit. 2014-02-03].

[26] WESTFELD, A.; PFITZMANN, A.: Attacks on Steganographic Systems. In *Proceedings of the Third International Workshop on Information Hiding*, IH '99, Londýn, UK: Springer-Verlag, 2000, ISBN 3-540-67182-X, s. 61–76.