Stegra – STEganoGRAphy CLI utility

Matouš Dlabal
ČVUT–FIT
dlabamat@fit.cvut.cz

2021-01-02

1 Úvod

Tento report informuje o semestrální práci vytvořené pro předmět BI–PYT v zimním semestru B201.

Semestrální práce je na téma steganografie – disciplíny zabývající se ukrýváním dat v jiných datech. Na rozdíl od kryptografie není cílem data šifrovat a tím znemožnit jejich čtení, ale zamezit zjištění, že nějaká taková data vůbec existují.

Cílem této práce je vytvořit CLI program, který umožní ukrývat data do obrázků (a následně je z nich extrahovat). Způsob jakým se data do obrázku vloží závisí na použitém algoritmu. Zadání požaduje alespoň 3 různé metody ukrytí dat. Program dále nabízí data před vložením zašifrovat pomocí blokové šifry AES (a samozřejmě také data po extrahování dešifrovat). Po ukrytí dat do obrázku program zobrazí hodnoty MSE[1] a PSNR[2], které indikují rozdílnost originálního a nově vytvořeného obrázku. Krom ukrývání dat program umožňuje výpočet kapacity obrázku (v závislosti na jednotlivých steganografických algoritmech). Poslední žádaná funkcionalita zadání je detekce ukryté zprávy v obrázku.

2 Steganografické algoritmy

Zde krátce popíši myšlenku za jednotlivými algoritmy. První dva postupy (LSB a PVD) přímo zmiňuje článek přiložený k zadání [1]. Poslední z algoritmů vznikl spojením metody LSB a principu využívání jen komplexních částí obrázku.

2.1 Least Significant Bit (LSB)

Nejjednodušší způsob jak ukrýt data do obrázku. Již z názvu[3] vyplývá, že se využije málo významných bitů jednotlivých pixelů, jejichž změna ovlivní celkový vzhled velmi málo. Čím více bitů se použije pro schování dat, tím více se projeví změna obrázku. Program využívá jen jeden poslední bit každého pixelu a to samostatně pro jednotlivé barevné složky (RGB). Kapacita obrázku, nosiče, v bytech je tedy

$$\left\lfloor \frac{\text{výška} \times \text{šířka} \times \text{kanály}}{8} \right\rfloor.$$

2.2 Pixel Value Differencing (PVD)

Tento algoritmus také využívá nejméně významný bit, ale neukládá do něj data přímo. Místo toho jsou data rozdílem dvou po sobě jdoucích pixelů. Máme-li nejméně významné bity b_0 a b_1 dvou takových pixelů, má bit $b_{\text{data}} = b_0 \oplus b_1$ hodnotu, kterou pixely skrývají.[4] Máme-li bit b_{data} do bitů uložit, stačí nastavit bit $b_1 = b_0 \oplus b_{\text{data}}$.

Jelikož je možné dvojice bitů překrývat, využijeme postupně $n - 1$ dvojic

$$(b_0, b_1),\ (b_1, b_2),\ \ldots,\ (b_{n-1}, b_n).$$

Implementačně je jednodušší pracovat s kanály samostatně, proto odečteme jeden nevyužitý bit za každý kanál obrázku a dostaneme kapacitu

$$\left\lfloor \frac{(\text{výška} \times \text{šířka} - 1) \times \text{kanály}}{8} \right\rfloor.$$

2.3 LSB na komplexních částech obrázku (CLSB)

Tato metoda využívá agresivnějšího algoritmu LSB (využívá více bitů pixelu). Aby jsme si to mohli dovolit a nezměnili obrázek natolik, že by byla změna oproti originálu viditelná, budeme ukládat data jen do komplexních částí obrázku. Na takových segmentech totiž není ani významnější změna nijak zásadně znatelná.

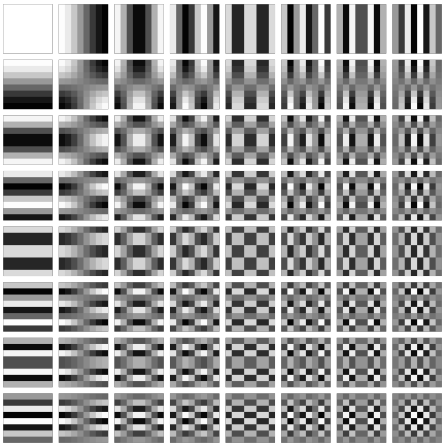
2.3.1 Komplexnost

Jako metriku komplexnosti jsem zvolil aritmetický průměr dolního pravého trojúhelníku DCT[5].

Po vzoru komprese JPEG obrázků jsem zvolil segmenty o rozměrech 8×8. Po aplikaci DCT na takový segment dostáváme pole stejných rozměrů naplněné koeficienty frekvencí.

[1] Mean Squared Error
[2] Peak Signal to Noise Ratio
[3] Česky nejméně významný bit
[4] Operátor $\oplus$ má význam bitového XORu
[5] Diskrétní kosínová transformace

$$\begin{bmatrix} c_{0,0} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & c_{1,7} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & c_{2,6} & c_{2,7} \\ \cdot & \cdot & \cdot & \cdot & \cdot & c_{3,5} & c_{3,6} & c_{3,7} \\ \cdot & \cdot & \cdot & \cdot & c_{4,4} & c_{4,5} & c_{4,6} & c_{4,7} \\ \cdot & \cdot & \cdot & c_{5,3} & c_{5,4} & c_{5,5} & c_{5,6} & c_{5,7} \\ \cdot & \cdot & c_{6,2} & c_{6,3} & c_{6,4} & c_{6,5} & c_{6,6} & c_{6,7} \\ \cdot & c_{7,1} & c_{7,2} & c_{7,3} & c_{7,4} & c_{7,5} & c_{7,6} & c_{7,7} \end{bmatrix}$$



Obrázek 1: Vzory odpovídající frekvencím 2D DCT

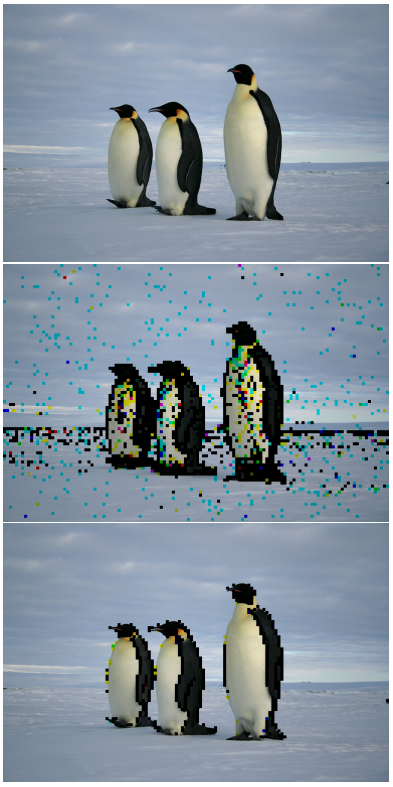
Z obrázku 1 je vidět, že v dolním pravém trojúhelníku se nacházejí vyšší frekvence, které přispívají k rozmanitosti segmentu. Pokud v segmentu nejsou zastoupeny, je komplexnost nízká a pokud by segment reprezentoval jen jednobarevnou plochu, koeficient $c_{0,0}$ by byl jediný nenulový.

2.3.2 Zvolení parametrů

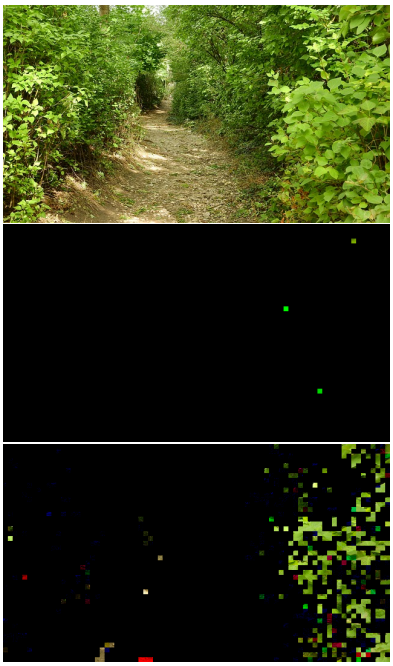
Pro přesnou specifikaci algoritmu je potřeba určit, kolik nejméně významných bitů každého pixelu se využije a jaký bude práh rozlišující komplexní a nekomplexní segmenty.

Metodou pokus-omyl jsem se rozhodl pro dvě konfigurace (počet bitů, práh komplexnosti): (2, 4) a (3, 32). V programu jsou pro jednoduchost označeny jako CLSB–2 a CLSB–3. Pro ilustraci prahů komplexnosti jsem do obrázků vložil nulová data a počet bitů nastavil na 8 (celá hodnota pixelu). Segmenty vyhodnocené jako nedostatečně komplexní nejsou nijak změněné. Komplexní segmenty mají hodnotu vynulovanou (ideálně černá barva), protože ale pracujeme s barevnými kanály samostatně (a ne vždy jsou pro stejnou oblast segmenty všech kanálů komplexní), projevují se částečně komplexní oblasti jen jako barevně zkreslené.

Na obrázku 2 vidíme, že i s velmi benevolentním prahem 4 se značná část obrázku nevyužije. Situace na obrázku 3 je už ale jiná. Původní obrázek neobsahuje moc jednolitých ploch. Díky tomu se s prahem 4



Obrázek 2: Originál, CLSB(8, 4) a CLSB(8, 32)



Obrázek 3: Originál, CLSB(8, 4) a CLSB(8, 32)

využije téměř celý a s přísnějším prahem 32 se stále využije jeho většina.

2.3.3 Kapacita

Kapacita závisí na konkrétním obrázku a získá se vynásobením počtu komplexních segmentů kapacitou jednoho segmentu v bytech, což je

$$\left\lfloor \frac{8 \times 8 \times (\text{využitých bitů pixelu})}{8} \right\rfloor .$$

Obrázek	CLSB–2	CLSB–3
Tučňáci	27,9 %	9,1 %
Cesta	199,9 %	268,3 %
Ideální obrázek	200,0 %	300,0 %

Obrázek 4: Kapacita obrázků oproti základnímu algoritmu LSB

V tabulce 4 jsou v procentech uvedené poměry kapacit v závislosti na zvoleném algoritmu. Ideální obrázek pro algoritmus CLSB je všude komplexní s rozměry celočíselně dělitelnými 8.

3 Problém s délkou dat

Jeden z problémů, na které jsem narazil bylo zjištění kolik dat se do obrázku uložilo. Původně jsem k datům ukládal jejich délku. To by ale bohužel umožňovalo odhadnout, zda byla nějaká data do obrázku vložena.

Délku by bylo potřeba ukládat v bytech a abych zbytečně neomezil možnost ukládání většího množství dat, bylo by nutné pro reprezentaci délky použít 4 B. Použití algoritmu CLSB–3 na ideální (plně komplexní) RGB obrázek o rozměrech 4000×4000 totiž umožňuje uložit

$$2^{24} < \frac{4000 \cdot 4000 \cdot 3 \cdot 3}{8} \approx \underline{17 \cdot 2^{20} \text{ B}} < 2^{32}.$$

Pravděpodobnost, že první 4 B náhodných vyextrahovaných dat obsahují číslo $\leq 17 \cdot 2^{20}$ je

$$\frac{17 \cdot 2^{20}}{2^{32}} \approx 0,415\ \%.$$

V případě nezašifrovaných dat je to jedno, ty je možné prostě zkusit extrahovat a zjistit, zda jsou validní. Pro zašifrovaná data už ale tato nízká hodnota význam má. Kdybychom navíc zkoumali obrázek výrazně menších rozměrů, dejme tomu 1920×1080, zjistíme, že pravděpodobnost klesne na pouhých 0,049 %. Maximální kapacita takového menšího obrázku je totiž přibližně 8 krát nižší.

3.1 Řešení

Abych se tomuto problému vyhnul, délku ukládaných dat do obrázku nevkládám. Při procesu vkládání dat uživateli vypíšu ukládanou délku v bytech. K tomu případně i to, že jsou vkládaná data moc velká a do obrázku se vloží jen jejich část. Při extrahování má potom uživatel možnost zadat délku uschovaných dat a tím dostat přesně stejná data, která byla do obrázku vložena. Pokud délku nezadá, extrahuje se celá kapacita obrázku. Pokud obrázek nějaká data obsahoval, vyextrahuje se s nimi navíc i „šum“ obrázku.

Jak jsem již jednou zmínil, v případě nezašifrovaných dat se toto opatření nijak neprojeví, v případě zašifrovaných dat se tím ale odstraní účinný způsob odhalení jejich přítomnosti.

4 Výstupní obrázek

Všechny tři postupy pro uschování dat jsou náchylné na jakékoli změny obrázku – nosiče. Proto je nutné volit takový výstupní formát, který neprovádí ztrátovou kompresi (např. PNG). Dále je potřeba si dávat pozor, aby při přenosu nosiče nedošlo k jeho kompresi. Nelze využít např. Facebook Messenger.

5 Steganoanalýza

Steganoanalýza se zabývá detekcí ukrytých zpráv. V zadání byl zmíněn požadavek detekce, ale nebyl nijak blíže specifikován. Jelikož se obecně jedná o složitější oblast, než je samotná steganografie, zaměřil jsem se jen na jednoduchou situaci *known-carrier* – máme k dispozici originální i zkoumaný obrázek.

K porovnání obrázků jsem využil hodnot MSE a PSNR. Obrázky jsou stejné, pokud se PSNR rovná jedné.

6 Závěr

Přestože se práce zabývá velmi jednoduchými technikami steganografie a rozhodně bych nedoporučoval využívat tento program pro nic konfidenčního, je program schopný ukrýt a zase zpětně získat textová i libovolná jiná data. Program navíc nabízí možnost data před ukrytím šifrovat, čímž poskytuje další vrstvu ochrany.

Přestože je hodně místa k vylepšení, jsem s výsledkem práce spokojený. Téma steganografie mě velmi bavilo a přestože jsem narazil na spoustu drobných i větších problémů, jsem rád, že jsem si toto téma vybral.

Reference

[1] Kaur, Harpreet a Rani, Jyoti. A survey on different techniques of steganography. *MATEC Web of Conferences*, 57:02003, 2016. Dostupné z: `https://www.matec-conferences.org/articles/matecconf/pdf/2016/20/matecconf_icaet2016_02003.pdf`.

[2] Pound, Mike a Riley, Sean. JPEG DCT, Discrete Cosine Transform (JPEG Pt2)- Computerphile. online, 2015. [cit. 2020–12–31] `https://www.youtube.com/watch?v=Q2aEzeMDHMA`.

[3] Padgham, Lin. Emperor penguins. online (CC BY 2.0), 2007. [cit. 2020–12–31] https://www.flickr.com/photos/linpadgham/2589167851.