FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

BIT

STEGANOGRAFIE FORMÁTŮ AVI A PNG

Martin Hamet
30. listopadu 2017
Doba přípravy $\simeq$ `80h`

Obsah

1 Zadání

Ve zvoleném programovacím jazyce navrhněte a implementujte program, který pomocí steganografie dokáže ukrýt a následně extrahovat zprávu ze souborů formátu `.avi` a `.png`.

2 Analýza problému a návrh řešení

Steganografie se zabývá ukrýváním komunikace, v našem případě zprávy, do média tak, aby nebylo poznat že nosné medium bylo pozměněno. V případě obrázku nebo videa by nemělo dojít k výrazné ztrátě kvality. Z praktického hlediska se nejčastěji využívá steganografie při prokazování autorských práv, tedy tvorba vodoznaků a podobně.

Naším úkolem je najít způsob ukrytí zprávy do výše zmíněných formátů. Bude tedy nejprve nutné zjistit jakou strukturu mají uložené formáty, následně do jaké části umístit vlastní data zprávy tak, aby nedošlo k velkým změnám ve výsledku. Dále nás bude zajímat kapacita daných formátů vzhledem ke skryté zprávě a rychlost zápisu i čtení.

2.1 Zpráva a její uložení

Obecně se nezdá vhodné snažit se ukrýt zprávu mimo vlastní data daného formátu, pokud nechceme ukrývat zprávy pouze minimální délky. Vlastní data daných formátů tvoří největší část celého souboru je tedy logické se snažit ukrýt zprávu v nich. Z čehož vyplývá problém s výše zmíněnou kvalitou. Pokud by naše zpráva výrazně ovlivnila vlastní data souboru bylo by zřejmé že je něco v nepořádku.

Za předpokladu že budeme ukládat zprávu do vlastních dat daného formátu můžeme snadno využít běžných postupů pro práci s daným formátem ve smyslu získání bitmapy z obrázku, streamu z videa atp. Budeme tedy zkoumat především strukturu dat a nikoliv nastavení hlavičky.

2.2 Portable Network Graphics (PNG)

K datům formátu `PNG` se dostaneme snadno s využitím `Bitmapy`, která poskytne přístup k jednotlivým pixelům obrázku. Aby nedocházelo k výrazným změnám bude vhodné modifikovat pouze nejméně významné bity (LSB) jednotlivých pixelů, respektive jejich barevných složek. Zprávu tedy nejprve převedeme do bitové podoby a jednotlivými bity zprávy budeme postupně nahrazovat LSB.

Z hlediska kapacity by bylo možné nepřepisovat pouze jeden bit, ale použít jich víc. Například tři nejméně významné bity z každé složky čímž by se výrazně zvýšila kapacita pro naši zprávu na úkor kvality obrázku.

Aby jsme nemuseli rozlišovat mezi různými formáty uložení pixelů (`GrayScale`, `TrueColor`, `TrueColor and Alpha`, etc) budeme načítat bitmapu v jednotném formátu. Dojde tedy ke změně formátu pixelů pokud bude zdrojový obrázek např. `GrayScale`.

Jednotlivé zasažené pixely nemusejí nutně následovat za sebou. Pomocí klíče nebo podle velikosti zprávy v poměru k velikosti obrázku by jsme mohli změněné pixely "roztáhnout". Například pokud by zpráva byla poloviční velikosti než kapacita obrázku, modifikovali by jsme pouze každý druhý pixel. Snížil by se tím celkový efekt případného zhoršení kvality obrazu.

Pro jednoduchost a z důvodu snadné kontroly efektu na výsledek budeme ukládat informaci postupně od prvního pixelu a pouze změnou jednoho bitu z každé barevné složky.

Konec zprávy se dá označit více způsoby pro tento formát nejsnadnější bude použití ukončovacího znaku na konci zprávy.

2.3 Audio Video Interleaved (AVI)

Tento formát je ve skutečnosti multimediální kontejner který může obsahovat několik datových streamů obvykle pouze audio a video, které lze přehrávat současně. Z pohledu ukrývání zprávy je zde více možností, ale omezíme se pouze na video. Samotné video je tvořené sekvencí obrázků. Při získání jejich streamu bude možné použít stejné techniky jako u formátu `PNG`. Video stream je ovšem komprimovaný podle zvoleného kodeku.

Vzhledem k tomu že je jedná o stream obrázků velikost značně roste používají se proto kodeky pro jeho ztrátovou komprimaci. Obyčejný zápis do LSB jednotlivých snímků by se tedy poškodil. Existují metody pro omezení poškození zprávy ztrátovou kompresí např. (Swaping Algorythm) nejedná se už ovšem o triviální problém. Proto budeme pracovat pouze s videem bez komprese. Respektive pro vytvoření výsledného videa nepoužijeme kodek se ztrátovou kompresí.

Protože video stream nepodléhá kompresi očekáváme že velikost videa značně vzroste. Taková metoda pro ukrytí zprávy by byla vhodná pouze pro videa v `raw` formátu, nicméně pro demonstraci steganografie nám bude stačit.

Konec zprávy v tomto případě bude vhodné určit z velikosti zprávy kterou uložíme na začátek, protože díky známé velikosti bude možné snadno připravit datové struktury a urychlit tím běh programu. Očekávaná doba zpracování by měla být delší vzhledem k objemu dat ve video streamu.

Postup ukrytí zprávy:

- Otevření kontejneru `AVI` a získání video streamu.
- Dekomprese podle použitého kodeku.
- Získání bitmapy jednotlivých snímků.
- Ukrytí zprávy v bitmapách.
- Zápis upravených bitmap do výsledného video streamu.
- Uložení vytvořeného streamu bez komprese.
- Uložení nově vytvořeného avi.

3 Implementace (popis řešení)

V této sekci budou popsány hlavní moduly programu a jejich vzájemná funkce.

`Program` Zajišťuje pouze základní předání funkce programu podle zvoleného formátu v parametrech programu a zobrazení základní nápovědy.

`MsgLoader` Zajišťuje načtení zprávy pro ukrytí ze souboru.

3.1 PNG

Samostatná třída `PngHandler` poskytuje veškeré nástroje pro práci se soubory `.png`. Využívá především funkcionality `System.Drawing` pro práci s obrázky respektive použití `Bitmap`.

`pngMod()` Jediná veřejně přístupná metoda, která zajistí volání jednotlivých funkcí podle zvolených parametrů. Tedy ukrytí/extrakci zprávy (`pngInsert()`, `pngExtract()`). Případně zobrazí nápovědu pro korektní zadání parametrů.

`insertMessage()` Projde předanou bitmapu po pixelech a pomocí maskování přepisuje LSB jednotlivých složek podle zprávy která má být v obrázku ukryta. Na konec zprávy je přidán speciální znak značící konec zprávy. Pro modifikaci pixelů využíváme metod `setPixel()`, `getPixel()`.

`extractMessage()` Podobně jako u `insertMessage()` průchodem bitmapou a extrakcí jednotlivých LSB postupně tvoří zprávu spojováním znaků do řetězce, dokud nenarazí na koncovou značku.

3.2 AVI

Samostatná třída `AviHandler` poskytuje veškeré nástroje pro práci se soubory `.avi`. Podobně jako `pngHandler` využívá `Bitmap` a dále knihovny `AForge.Video.FFMPEG` a jejích závislostí. Zejména kvůli přístupu k sadě kodeků a tím i snadnému získání dekomprimovaného video streamu, nebo jeho vytvoření.

`aviMod()` Jediná veřejně přístupná metoda, která zajistí volání jednotlivých funkcí podle zvolených parametrů. Tedy ukrytí/extrakci zprávy (`aviInsert()`, `aviExtract()`). Případně zobrazí nápovědu pro korektní zadání parametrů.

`writeMessage()` S využitím `VideoFileReader` z knihovny `AForge` prochází jednotlivé snímky video streamu a podobně jako v případě `PngHadler` v nich modifikuje LSB jednotlivých pixelů. V prvním pixelu prvního snímku je nejprve uložena velikost celkové zprávy počtem znaků. První pixel je tedy zcela přepsán touto informací (nikoliv jen jeho LSB).

Pro úpravu bitmapy zde využíváme přímého přístupu ke kopii dat bitmapy (pole pixelů) což by mělo znatelně urychlit modifikaci jednotlivých pixelů.

`readMessage()` Stejným průchodem jako u `writeMessage()` získáváme postupně zprávu z jednotlivých pixelů. Ovšem tentokrát jsou jednotlivé bity konvertovány rovnou do pole znaků (z důvodu vyhnutí se spojování řetězců) což značně urychlí získání zprávy oproti implementaci pro `PNG`.

`setMsgLength()` Použije se pro první snímek video streamu, kde na první pixel uloží hodnotu, určující počet znaků skryté zprávy, přes všechny barevné složky pixelu.

4 Uživatelská příručka

Pro správný běh programu je nutné, aby se spustitelný soubor nacházel ve stejném adresáři jako configurační soubor `Steganography.exe.config` a složka `lib`. Dále program vyžaduje nainstalovaný `Microsoft.NET Framework`.

4.1 Spuštění programu

Program se spouští z příkazové řádky s následujícími parametry
(bez znaků `[` a `]`.

- `[-png][-avi]` Typ souboru pro zpracování.
- `[-i][-e]`

  -i Vložení zprávy do zdrojového souboru.

  -e Získání zprávy ze zdrojového souboru.
- `[vstup]` Název zdrojového souboru.
- `[výstup]` Název výstupního souboru.
- `[zpráva]` Název souboru se zprávou pro ukrytí.

Příklady spuštění:

- `Steganography.exe`
  Zobrazí základní nápovědu.
- `Steganography.exe -png`
  Provede zápis a extrakci zprávy podle základního nastavení viz 4.1.1
- `Steganography.exe -avi`
  Provede zápis a extrakci zprávy podle základního nastavení viz 4.1.1
- `Steganography.exe -png -i zdroj.png vystup.png zprava.txt`
  Provede zápis zprávy ze souboru `zprava.txt` do obrázku `zdroj.png` a vytvoří výsledný soubor `vystup.png`
- `Steganography.exe -avi -e zdroj.avi`
  Získá skrytou zprávu ze souboru `zdroj.avi` a vypíše ji do konzole.

4.1.1 Základní nastavení

Program provede ukrytí zprávy a její následnou extrakci.

PNG

Zdrojový soubor:	`source.png`
Výstupní soubor:	`output.png`
Soubor se zprávou:	`msg.txt`

AVI

Zdrojový soubor:	`drop.avi`
Výstupní soubor:	`out.avi`
Soubor se zprávou:	`msg.txt`

5 Závěr

Program je připraven pro další rozšiřování o formáty. Zvolené algoritmy byly vybrány tak, aby byly snadno implementovatelné a dali se dále rozšiřovat. Původní snahy o implementaci pomocí základní knihovny `avifil32.dll` se nepodařili kvůli problémům s kodeky a dekopresí videa. Původní verze aplikace byla velmi pomalá při čtení zprávy z videa. Pomocí profilovacího nástroje jsem vysledoval problém k tvoření zprávy jako postupně se zvětšujícího řetězce. Tento problém byl odstraněn náhradou za bytové pole. Aplikace se tím zrychlila z řádů desítek minut pro zprávy o velikosti statisíců slov do řádů desítek sekund.

Vlastní ukrývání zprávy by šlo vylepšit například zmíněným roztroušením modifikovaných pixelů podle poměru kapacity a velikosti zprávy a kapacita by šla snadno navýšit použitím více méně významných bitů z každého pixelu.

K ukrývání zprávy do souborů formátu `AVI` by bylo potřeba použít složitější algoritmus, který by zachoval zprávu i po kompresi videa. Nekomprimované video nabývá svou velikostí skutečně rychle, proto jsem zvolil ukázkové video v malém rozlišení pouze o velikosti 700 KB.

Použitím pouze jednoho nejméně významného bitu se kvalita videa ani obrázku nijak nezhoršila.

Je vidět že `PNG` zaostává což je způsobeno menší optimalizací kódu. Použití řetězce a metod `setPixel()`, `getPixel()`, které ze zkušenosti s grafikou vím že jsou pomalejší než přímý přístup k datům bitmapy. Časy běhu programu byly měřeny pouze pro zápis zprávy časy čtení poměrově odpovídají mimo čtení `PNG`, kde čtení nadměrných zpráv je, z důvodu použití řetězců, neúnosně dlouhé a poukazuje na nutnost optimalizace takových operací, kterým jsem se od začátku vyhýbal u formátu `AVI`.

Počet slov zprávy	PNG čas [ms]	AVI čas [ms]
$1k$	218	409
$50k$	2648	518
$200k$	–	1006