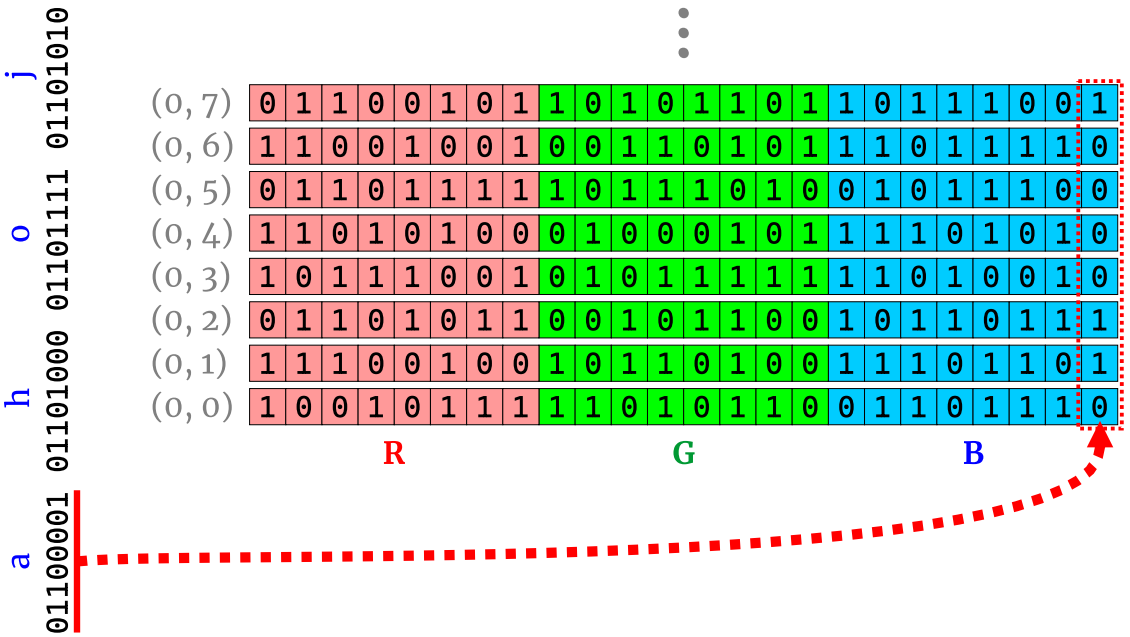
ZADÁNÍ SEMESTRÁLNÍ PRÁCE
DIGITÁLNÍ STEGANOGRAFIE

Zadání

Naprogramujte v ANSI C přenositelnou[1] **konzolovou aplikaci**, která ukryje/obnoví soubor s libovolným užitečným obsahem (tzv. *payload*) do/z digitálního snímku (fotografie) ve formátu PNG nebo BMP. Payload bude nejprve zkomprimován a následně uložen do bitmapové reprezentace snímku po jednotlivých bitech, a to tak, že n-tý bit payloadu bude uložen do LSB modrého kanálu n-tého pixelu tak, jak ukazuje (zjednodušeně, bez komprese) obr. 1. Tím je tedy dáno, že jak v případě formátu PNG, tak BMP bude nutné zajistit, že je bitmapa uložena jako sekvence 8-bitových hodnot intenzity červeného, zeleného a modrého kanálu, tj. tzv. 24-bit RGB. Program



Obrázek 1: Ukázka způsobu steganografického kódování textu do RGB bitmapy

se tedy před skrýváním textu do obrázku nejprve přesvědčí, že je obrázek (PNG nebo BMP) v subformátu 24-bit RGB (není-li toto splněno, ukončí se s chybovým hlášením), pak provede kompresi payloadu bezeztrátovou kompresní technikou LZW, ověří, že se zkomprimovaný obsah „vejde“ do pixelů obrázku (tj. zda je jich dostatek – pokud ne, opět se ukončí s chybovým hlášením) a nakonec zkomprimovaný obsah uloží naznačeným způsobem do bitmapy, kterou zapíše na disk.

Program se bude spouštět příkazem

`stegim.exe`[2] ⟨*obrázek*[`.png`|`.bmp`]⟩ ⟨*-přepínač směru*⟩ ⟨*payload*⟩↵

Symbol ⟨*obrázek*⟩ zastupuje první povinný parametr – název souboru s bitmapovým obrázkem ve formátu PNG nebo BMP, subformátu 24-bit RGB, do kterého bude steganograficky ukryt nebo

[1] Je třeba, aby bylo možné Váš program přeložit a spustit na PC s operačním prostředím Win32/64 (tj. operační systémy Microsoft Windows NT/2000/XP/Vista/7/8/10/11) a s běžnými distribucemi Linuxu (např. Ubuntu, Debian, Red Hat, atp.). Server, na který budete Vaši práci odevzdávat a který ji otestuje, má nainstalovaný operační systém Debian GNU/Linux 11 (bullseye) s jádrem verze 5.10.0-20-amd64 a s překladačem gcc 10.2.1.

[2] Přípona `.exe` je povinná i při sestavení v Linuxu, zejm. při automatické kontrole validačním systémem.

z něj bude naopak extrahován příslušný užitečný obsah (podle toho, jaký bude uveden přepínač směru).

Symbol ⟨*-přepínač směru*⟩ pak představuje druhý povinný parametr, kterým uživatel určuje, zda se bude payload ze souboru do obrázku zakódovávat nebo se bude z obrázku extrahovat a ukládat na disk. Je-li tento přepínač předán programu ve tvaru '`-h`' (jako ***h**ide*), bude payload specifikovaný třetím povinným parametrem komprimovat a ukrývat do obrázku. Při zadání ve tvaru '`-x`' (jako *e**x**tract*) bude payload vyjímat z bitmapového obrázku, dekomprimovat a ukládat zpět na disk do zvoleného souboru (daného třetím povinným parametrem).

Třetí povinný parametr zastoupený symbolem ⟨*payload*⟩ je specifikace souboru na disku, ze kterého program vezme užitečný obsah určený k ukrytí do obrázku nebo do něj naopak užitečný obsah vyjmutý z obrázku uloží.

U obou předávaných souborů (obrázku a payloadu) berte v potaz možnost, že uživatel uvede jak krátké jméno, tak také úplnou specifikaci včetně úplné cesty (a ta může obsahovat i např. mezery).

Úkolem Vámi vyvinutého programu v případě spuštění s parametrem *přepínač směru* ve tvaru '`-h`' tedy bude:

1. Načíst soubor s obrázkem ve formátu PNG nebo BMP, subformátu 24-bit RGB, a zjistit jeho rozměry, resp. počet pixelů n_{pix}. Není-li obrázek ve vhodném formátu, ukončit s chybovým hlášením a návratovým kódem `2`.
2. Načíst soubor s payloadem (nemusí být nutně jen textový, může být obecně libovolný).
3. Zkomprimovat payload bezeztrátovou kompresní technikou LZW. Tím je dána délka zkomprimovaného obsahu l_{comp} v bytech.
4. Zkontrolovat, zda počet pixelů je dostatečný pro ukrytí payloadu, tj. zda $n_{\text{pix}} \geq 8 \cdot l_{\text{comp}}$. Není-li toto splněno, ukončit s chybovým hlášením a návratovým kódem `3`.
5. Ukrýt payload bit po bitu do LSB (nejméně významného bitu) modré složky pixelů obrázku.
6. Uložit modifikovaný obrázek (se zakódovaným payloadem) zpět na disk ve stejném formátu (tedy PNG nebo BMP), v jakém byl načten.

V případě spuštění s parametrem *přepínač směru* ve tvaru '`-x`' to pak bude:

1. Načíst soubor s obrázkem ve formátu PNG nebo BMP, subformátu 24-bit RGB, a zjistit, zda je v něm ukryt payload[3]. Pokud ne, ukončit s chybovým hlášením a návratovým kódem `4`. V případě předání obrázku v nevyhovujícím formátu vraťte již dříve stanovený návratový kód `2` pro případ nevhodného formátu obrázku.
2. Vyjmout bit po bitu z LSB modré složky pixelů obrázku zkomprimovaný payload. Ověřit, zda není poškozen (např. výpočtem kontrolního součtu CRC32[4] a porovnáním s uloženou hodnotou). Je-li payload poškozen, ukončit s chybovým hlášením a návratovým kódem `5`.
3. Dekomprimovat payload a uložit do souboru předaného třetím parametrem.

Váš program může být během testování spuštěn například takto (v operačním systému Windows):

```
stegim.exe "E:\My Work\Test\Holiday 01.png" -h secret_text.txt↵
```

nebo takto:

```
stegim.exe "E:\Download\Beach.bmp" -x "E:\Documents\Tajná zpráva.txt"↵
```

nebo pouze takto:

[3]To tedy nutně znamená, že musíte kromě payloadu do obrázku zakódovat i nějakou vhodnou signaturu, podle které budete schopen/schopna toto ověřit. Rozumné je spojit tuto signaturu s kontrolním součtem pro ověření neporušenosti zakódovaného obsahu.

[4]Cyclic Redundancy Check, 32-bit Checksum

```
stegim.exe img001.png -h message.txt↵
```

Spuštění v UNIXových operačních systémech (GNU/Linux, FreeBSD, macOS, atp.) může vypadat např. takto:

```
stegim.exe /home/user/images/img001.png -h /home/user/doc/secret.txt↵
```

Uvažujte všechny možné specifikace jak souboru s obrázkem, tak souboru s payloadem, které jsou přípustné v daném operačním systému. Pokud nebudou na příkazové řádce uvedeny právě 3 argumenty (soubor s obrázkem, přepínač směru a soubor s payloadem), vypište chybové hlášení, stručný návod k použití programu (v angličtině) a ukončete program s návratovým kódem `1`.

Dovolte také uživateli uvést přepínač směru na libovolném pozici, tj. nejen jako druhý parametr v pořadí, ale i jako první nebo třetí (to bude snadné, protože vždy začíná znakem '`-`'). Naopak v případě specifikace souboru je ten první předaný vždy obrázek a druhý payload.

Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Postupujte podle instrukcí uvedených na webu předmětu. Archiv nechť obsahuje všechny zdrojové soubory potřebné k přeložení programu, **makefile** pro Windows i Linux (pro překlad v Linuxu připravte soubor pojmenovaný `makefile` a pro Windows `makefile.win`) a dokumentaci ve formátu PDF vytvořenou v typografickém systému TEX, resp. LATEX. Bude-li některá z částí chybět, kontrolní skript Vaši práci odmítne.

Některé detaily činnosti programu

Základní princip činnosti byl již popsán výše. Zde budou jen doplněny některé podrobnosti. Obecně platí, že činnost programu je tzv. dávková, tj. neočekává se (a nedochází) v průběhu činnosti k žádné interakci s uživatelem. Jakmile uživatel program se správně zadanými povinnými parametry spustí, program již doběhne sám, bez dalších dotazů či potvrzování čehokoliv.

Práce s grafickými formáty PNG a BMP

Formát PNG je poměrně složitý a rozhodně nelze požadovat v rámci semestrální práce implementaci funkcí pro načítání a ukládání obrázků v tomto formátu. Pro práci s ním **použijte knihovnu libpng**, kterou lze nalézt na adrese URL `http://www.libpng.org/pub/png/libpng.html`. Tato knihovna je stabilní, multiplatformní a funguje velmi dobře. Navíc lze bez potíží přeložit jak ve Windows, tak v Linuxu či macOSu (ověřeno vyučujícím dne 6. 10. 2023, verze knihovny libpng 1.6.40, překladač gcc verze 12.2.0). **Nepoužívejte však jinou knihovnu pro práci s formátem PNG**, protože pak by Vaše aplikace nemusela být přeložitelná na validačním serveru – na tom je totiž nainstalovaná právě knihovna **libpng**, a tudíž přepínač linkeru '`-lpng`' při překladu na validačním serveru zajistí její přilinkování a správnou činnost, stejně jako na Vašem domácím počítači.

Naopak formát BMP je celkem jednoduchý (vizte URL `https://en.wikipedia.org/wiki/BMP_file_format`) a zvláště jeho subformát '`BM`' 24-bit RGB bez komprese (tj. offset 30 hlavičky má hodnotu `0` čili `BI_RGB`). Proto **funkce pro načítání a ukládání v tomto formátu naprogramujte sami**.

Signatura, kontrolní součty

Váš program musí být schopen jednoznačně určit, zda obrázek urývá nějaký steganograficky zakódovaný obsah. Tzn. není možné připustit situaci, kdy prostě vyjme všechny LSB bity z modrého kanálu a ty prohlásí za ukrytý obsah (ono by to tedy ani nešlo korektně dekomprimovat LZW, protože při použití algoritmu LZW existuje jednoznačná korespondence mezi nezkomprimovanými a zkomprimovanými daty – jinými slovy nejde dekomprimovat to, co nebylo LZW zkomprimováno).

Aby toto bylo možné, je třeba do obrázku kromě samotného payloadu zakódovat také nějaký unikátní vzorek, podle kterého program pozná, že jím byl obrázek modifikován a tento vzorek nebude moci vzniknout náhodně pořízením obrázku (nebo alespoň pravděpodobnost takového náhodného vzniku musí být dostatečně nepatrná). Řešení tohoto dílčího problému je ponecháno zcela Vaší fantazii a programátorským schopnostem.

Současně je ale také nutné zajistit, aby program dokázal detekovat situaci, kdy byl ukrytý obsah poškozen (např. v důsledku manipulace se steganograficky modifikovaným obrázkem v nějakém fotoeditoru). K tomuto účelu je doporučeno (nikoliv nařízeno) použít mechanismus kontrolního součtu CRC32. Výpočtem CRC32 pro daný payload (jasně, že ten už zkomprimovaný) a jeho uložením spolu s payloadem do obrázku je možné později při extrakci snadno ověřit, že extrahovaný obsah má stejnou hodnotu CRC32, jaká byla dříve uložena do obrázku a tak nabýt jistoty, že payload je v pořádku a shoduje se s tím, který byl do obrázku dříve steganograficky ukryt.

Kontrolní součet navíc může být vhodně spojen s unikátní signaturou zmíněnou výše, přičemž pravděpodobnost náhodného výskytu obou dvou těchto bezpečnostních komponent ve validním stavu se pak jistě limitně blíží nule.

Výstup programu

Výstupem programu při ukrývání payloadu (zadán přepínač '`-h`') je modifikovaný obsah na začátku zpracování načteného obrázku, který je uložen do téhož souboru (v tomtéž formátu). Původní obsah obrázku je tedy přepsán modifikovanou verzí (bez varování, bez dotazů na uživatele).

Výstupem při extrakci payloadu (zadán přepínač '`-x`') je pak ukrytý užitečný obsah (pokud se ovšem v obrázku nalézá), který se uloží do souboru daného třetím povinným parametrem. Pokud tento soubor již v pracovním adresáři existuje, je přepsán (opět bez dotazu či potvrzování ze strany uživatele).

Chybové stavy

Jakmile dojde při vykonávání programu k chybě, která nedovoluje pokračování, nechť program vypíše do konzole srozumitelné chybové hlášení (dle Vašeho uvážení) v angličtině a ukončí se s nenulovým návratovým kódem. Tabulka 1 shrnuje nejdůležitější chybové stavy a návratové kódy jim odpovídající. Tyto uvedené je nutné bezpodmínečně dodržet, neboť ty bude ověřovat i validační systém. Pokud při implementaci narazíte i na jiné, zde nepopsané chybové stavy, zvolte si pro ně

Tabulka 1: Návratové kódy a jim odpovídající chybové stavy programu

Návratový kód	**Popis chybového stavu**
0	bez chyby
1	nesprávně zadané či chybějící parametry na příkazové řádce (také neexistující vstupní/výstupní soubory)
2	nevhodný formát obrázku (tj. není to PNG či BMP, 24-bit/pixel, RGB)
3	obrázek není dostatečně velký pro ukrytí zadaného payloadu a potřebných doprovodných informací
4	obrázek určený k extrakci payloadu neobsahuje žádný ukrytý obsah
5	ukrytý obsah byl poškozen, neodpovídá kontrolní součet obsahu či nelze obsah korektně dekomprimovat

libovolné (v tabulce neuvedené, tedy „neobsazené“) návratové kódy podle Vaší potřeby.

Užitečné techniky a odkazy

Níže uvedené techniky je možné (ale nikoliv nezbytně nutné) využít při řešení úlohy. Protože se jedná o postupy v informatice víceméně standardní, lze k nim nalézt velké množství dokumentace např. za pomoci vyhledávače Google:

1. **steganografie** – `https://en.wikipedia.org/wiki/Steganography`
2. knihovna **libpng** – `http://www.libpng.org/pub/png/libpng.html`
3. kompresní technika **LZW** – `https://en.wikipedia.org/wiki/Lempel-Ziv-Welch`
4. kontrolní součet **CRC32** – `https://en.wikipedia.org/wiki/Cyclic_redundancy_check`
5. bitové (bitwise) operátory jazyka C

Řešení úlohy je zcela ve vaší kompetenci – zvolte takové algoritmy a techniky, které podle vás nejlépe povedou k cíli.