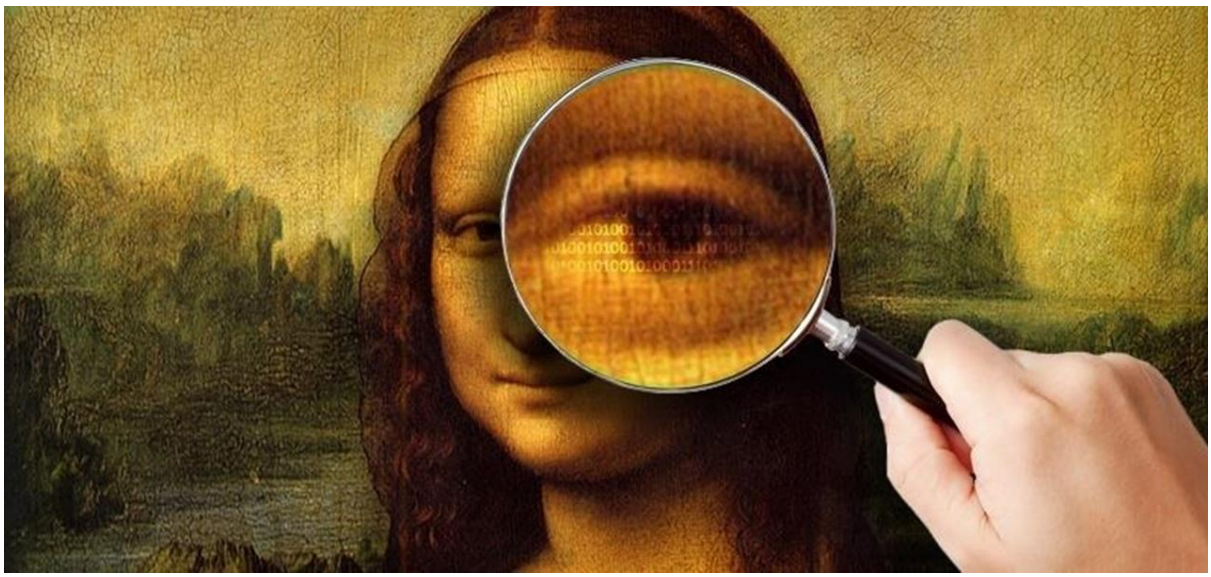
Introdução a Técnicas de Programação - 2017.1

Descrição de projeto

Esteganografia - Programando para espiões



O termo **esteganografia** deriva do grego, no qual ***estegano*** significa “esconder ou mascarar”, e ***grafia***, “escrita”. Assim, esse termo pode ser definido como a arte de esconder informações, tornando-as ocultas. Seu principal objetivo é que esses dados não sejam percebidos por terceiros. Somente o receptor da mensagem tem conhecimento de sua existência, assim como da maneira como extraí-la.

Muitos são os meios utilizados para a aplicação da esteganografia. Esta técnica foi difundida entre espiões durante o período da Guerra Fria. A literatura específica, a sua maioria da época da Guerra Fria, relata exemplos de esteganografia usando marcação de caractere, tinta invisível, perfurações, fita corretiva de máquina de escrever, arranjo de caracteres (*character arrangement*), assinaturas digitais, canais escondidos (*covert channels*), comunicações por espalhamento de espectro (*spread spectrum communications*), entre outras técnicas. Com o avanço e proliferação da computação, mensagens podem ser escondidas em imagens, por exemplo, utilizando-se de algumas técnicas específicas, como a do bit menos significativo, que será aplicada a este projeto e é explicado a seguir. Além de imagens, arquivos de áudio também podem ser usados para ocultar mensagens, de maneira que estas não sejam percebidas por quem estiver ouvindo o som. De forma análoga, qualquer tipo de arquivo digital pode facilmente transportar mensagens secretas, através do uso da esteganografia.

A esteganografia possui inúmeras aplicações. Ela é uma das técnicas utilizadas para implementar mecanismos de verificação de direitos autorais em imagens e outras mídias. Além disso, pode ser utilizada para a divulgação de mensagens sem o conhecimento da existência dessas mensagens por parte de outros interessados. No entanto, é relevante notar

que as técnicas também possuem algumas restrições. Por exemplo, o tamanho das informações a serem escondidas é limitado pelo tamanho do próprio meio que será utilizado. Quanto menos essas informações degradarem a aparência do objeto hospedeiro, maior é o potencial das técnicas esteganográficas não serem detectadas. Geralmente, mensagens muito grandes acabam ferindo a integridade do objeto hospedeiro, o que permite uma fácil detecção de que uma possível mensagem foi nele escondida (Wikipédia).

Esteganografia em Imagens Digitais

Muitas técnicas modernas possibilitam esconder informações dentro de imagens. A forma mais utilizada emprega a técnica denominada LSB (Least Significant Bit), que consiste em utilizar o bit menos significativo de uma determinada informação para armazenar um bit de uma nova informação. No caso de uma imagem com profundidade de cor de 24 bits, um bit de uma nova informação pode ser armazenado no bit menos significativo de cor dos pixels, ou seja, o bit menos significativo dos 24.

Considere o valor dos caracteres em binário da palavra "Wikipedia": W(01110111), i(01101001), k(01101011), i(01101001), p(01110000), e(01100101), d(01100100), i(01101001), a(01100001). Na forma apresentada, a palavra "Wikipedia" é representada utilizando 72 bits. Sendo assim, precisaremos de uma imagem com no mínimo 72 pixels. Para armazenar a letra 'W', iremos utilizar o bit menos significativo de cor dos 8 primeiros pixels. O primeiro bit do caracter 'W' é 1, se o bit menos significativo do primeiro pixel for 1, o valor é mantido, caso contrário é trocado para 1. Caso o bit a ser armazenado tenha valor 0, por exemplo o quarto bit do caractere 'W', a mesma regra é usada, se o bit menos significativo do quarto pixel for 0, o valor é mantido, caso contrário é trocado. Este procedimento deve ser repetido por todos os bits de cada caractere. No fim teremos uma imagem armazenando a palavra "Wikipedia" com ruído de 1 bit por pixel. Para extrair a informação da imagem, basta fazer o processo reverso. Leia o valor de cor cada pixel e armazene apenas o bit menos significativo. Provavelmente, será necessário armazenar o pixel que termina a informação.

É possível armazenar mais que um bit por pixel, porém isto aumentará o ruído da imagem. Uma maneira de armazenar mais informação, por exemplo, é armazenar um bit da nova informação no bit menos significativo de cada cor do pixel. Numa imagem RGB de 24 bit, as cores vermelho, verde e azul são representadas utilizando 8 bit cada uma. O mesmo processo apresentado anteriormente pode ser realizado para cada cor do pixel. Assim serão armazenados 3 bits por pixel.

Formato de imagem

Consultar o documento "Formatos de Imagem", texto elaborado por João Manuel Brisson Lopes para a disciplina Computação Gráfica do curso Licenciatura em Engenharia Informática e de Computadores no Departamento de Engenharia Informática do Instituto Superior Técnico - Universidade Técnica de Lisboa. Este documento foi publicado em Janeiro de 2003 e reeditado em Dezembro de 2008 e Abril 2013. Disponível em: http://disciplinas.ist.utl.pt/~leic-cg.daemon/textos/livro/Formatos%20de%20Imagem.pdf

Requisitos funcionais do programa

O nome do binário/executável deverá ser "**steg**" e o mesmo deverá ser capaz de processar um conjunto de parâmetros, como indicado a seguir.

Codificando um segredo em uma imagem:
./steg -e -i <input-file> -f <format> <image>
-e indica que o programa deve rodar em modo de codificador (*encoder*)
- i <input-file> indica o arquivo de entrada (*input*) a ser codificado na imagem
-f <format> indica o formato da imagem a ser usada como hospedeira para a mensagem. Os valores possíveis para <format> são: **bmp** ou **ppm** (outros valores poderão ser adicionados por você, caso o seu programa suporte novos formatos de imagem).
<image> imagem a ser usada como hospedeira para a mensagem

Exemplo: ./steg -e -i segredo.txt -f bmp imagem.bmp

Decodificando um segredo em uma imagem:
./steg -d -o <output-file> -f <format> <image>
-d indica que o programa deve rodar em modo de decodificador (*decoder*)
-o <output-file> indica o arquivo de saída (*output*) para onde será gravada a mensagem decodificada
-s indica que a mensagem decodificada deve ser mostrada na saída padrão (stdout). O uso desta opção invalida o uso da opção -o <output-file>
-f <format> indica o formato da imagem a ser usada como hospedeira para a mensagem. Os valores possíveis para <format> são: **bmp** ou **ppm** (outros valores poderão ser adicionados por você, caso o seu programa venha a suportar novos formatos de imagem).
<image> imagem hospedeira da mensagem (.ppm, .bmp ou outros formatos)

Exemplo: ./steg -d -o segredo.txt -f ppm imagem.ppm

Por questões de simplicidade, seu programa deverá suportar OBRIGATORIAMENTE[1] os seguintes formatos de imagem:

- PPM (Portable PixMap): imagens a cores com 24 bits e em formato binário (P6);
- BMP (BitMaP): imagens a cores com 24bits e sem compressão.

Descrição do projeto

[1] Isso significa que você deverá implementar estes formatos, mas pode melhorar seu trabalho e, consequentemente, sua nota ao adicionar o suporte a outros formatos de imagem.

O projeto de Esteganografia deve ser desenvolvido em linguagem C, a ser executado, em sua versão mais simples, através de linha de comando (entrada e saída em um console/terminal). Cada projeto deve apresentar um codificador e um decodificador. O projeto deve atender também os seguintes critérios de programação:

1. **Alocação dinâmica de arranjos e/ou matrizes;**
2. **Uso de registros (`struct`) e enum para representar/ler/gravar uma imagem;**
3. **Definição de novos tipos de dados através de `typedef`;**
4. **Leitura/escrita de imagens a partir de arquivos;**
5. **Modularização do programa em diferentes funções (modularização interna) e arquivos (uso de diferentes arquivos .c e .h, cada um com sua funcionalidade - modularização externa);**
6. **Boas práticas de programação: Definição de um padrão de indentação do código fonte e de nomenclatura das sub-rotinas e variáveis;**
7. **Emitir mensagens para stderr em situações de erro: falha na alocação de memória, falha na abertura do arquivo, quantidade de pixels insuficientes para codificar a mensagem.**
8. **Documentação adequada do código-fonte (uso de comentários).**

Observações:

- O projeto deve ser desenvolvido **individualmente ou em duplas** (grupos de dois alunos). Não serão permitidos grupos com três ou mais alunos;
- Para facilitar o acompanhamento do projeto, cada dupla deve estar associada a uma única turma de PTP e a uma única turma de ITP. Ou seja, não serão permitidas duplas formadas por alunos matriculados em turmas diferentes;
- Cada dupla deve desenvolver sua solução de forma independente das demais. Soluções idênticas serão consideradas plágios e, portanto, sanções serão devidamente aplicadas em todas as duplas com soluções similares;
- Códigos e algoritmos podem ser utilizados da web desde que devidamente referenciados. Caso sejam encontrados trechos de código na web equivalentes aos apresentados pelo grupo sem a devida citação, o código será igualmente considerado plágio e sanções serão aplicadas ao grupo. Vale salientar que a avaliação será realizada unicamente sobre o código produzido pela dupla. Códigos retirados da web, apesar de permitidos com a devida citação, serão desconsiderados dos critérios de pontuação.

Processo de Avaliação

O processo de avaliação desta tarefa compreende a execução e desenvolvimento do projeto em **4 semanas**. A cada semana, o grupo deve apresentar uma etapa do projeto desenvolvido, seguindo o calendário abaixo:

- **CHECKPOINT 1 - 06/06/2017**
    1. Tipos de dados necessários (`typedef`, `structs` e `enums`) para manipular cada formato de imagem a ser suportado;
    2. Modularização externa do programa (quais os arquivos .c e .h);
    3. Leitura de imagens PPM de arquivo e correta leitura dos pixels;
- **CHECKPOINT 2 - 08/06/2017**
    1. Leitura de imagens BMP de arquivo e correta leitura dos pixels;
    2. Codificador: Manipulação dos pixels e inserção de mensagens simples
- **CHECKPOINT 3 - 13/06/2017**
    1. Codificador: Inserção de mensagens de um arquivo texto qualquer
    2. Documentação
- **CHECKPOINT 4 - 20/06/2017**
    1. Decodificador
    2. Testes
- **CHECKPOINT "Atrasados é a sua última chance!" - 22/06/2017**
    1. O que falta apresentar.

Sobre as duplas

Os alunos têm **ATÉ o dia 01/06/2017** para comunicar aos respectivos professores de ITP e PTP **(SOMENTE POR E-MAIL)** se farão o trabalho em dupla (e a composição da mesma) ou se farão individualmente. O assunto do e-mail deve ser: "**Trabalho IMD0012 - 2017.1: Definição dupla**" e no corpo do e-mail deve conter os nomes completos da dupla.

Critérios de pontuação

O desenvolvimento do projeto aqui descrito vale 100% da nota da terceira unidade de PTP e ITP. A pontuação da avaliação seguirá os critérios e distribuição abaixo:

- Atendimento dos requisitos funcionais: 50%
- Uso dos recursos da linguagem C: 20%

  A dupla demonstrou saber usar de forma adequada os recursos da linguagem C (arranjos, structs, typedefs, recursividade, etc)?
- Organização do código e documentação: 10%

O código está documentado? a indentação e uso de { } seguem um padrão (indentação)? o programa está devidamente modularizado em diferentes arquivos?

- ❑ Funcionalidades avançadas: 20%
  As funcionalidades extras desenvolvidas pela dupla foram suficientemente complexas?

A pontuação a ser dada pelas funcionalidades extras não é definida *a priori*. Cada caso será avaliado em função da complexidade envolvida. Itens extras de baixa complexidade serão desconsiderados na pontuação.

Entrega do projeto

O projeto deve ser submetido pelo SIGAA até a data **27/06/2017 (até o meio dia)** em um arquivo comprimido **(.zip)** contendo os arquivos fontes do projeto **(.c e .h)** e um arquivo **README.TXT**. Este arquivo deve ter as seguintes informações:

- ❑ O que foi feito (o básico e as funcionalidades extras implementadas);
- ❑ O que não foi feito (caso não tenha sido possível implementar alguma funcionalidade);
- ❑ O que seria feito diferentemente (quando aprendemos à medida que vamos implementando, por vezes vemos que podíamos ter implementado algo de forma diferente. Assim, esse tópico é para vocês refletirem sobre o que vocês aprenderam durante o processo que, se fossem fazer o mesmo projeto novamente, fariam diferente);
- ❑ Como compilar o projeto. Atenção, não será permitido o uso de bibliotecas externas. Vocês devem usar APENAS a biblioteca padrão de C e a biblioteca C POSIX.
- ❑ Em caso de duplas:
  - Identificação dos autores;
  - Contribuição de cada integrante no desenvolvimento do projeto (quem fez o quê).

Recomendações diversas:

1. Solução de backup: não será tolerada a desculpa de que um disco rígido falhou nas avaliações. Assim, é importante manter várias cópias do código-fonte desenvolvido ou usar um sistema de backup automático como o Dropbox, Google Drive, Box ou similares. Uma solução melhor ainda é fazer uso de um sistema de controle de versões como git, e um repositório externo como Github ou Bitbucket.
2. Especificar precisamente a interface e o comportamento de cada sub-rotina. Usar esta informação para guiar o desenvolvimento e documentar o código desenvolvido.

Funcionalidades Avançadas

1. Leitura e manipulação de imagens em outros formatos, como por exemplo: PNG, GIF, JPEG.
2. Implementação de outras técnicas de esteganografia diferentes do Least Significant Bit, como por exemplo: Bit-Plane Complexity Segmentation, Espalhamento de Informação, Ordenação.