Relatório referente ao Trabalho I de Programação de Software Básico

Bernardo de Cesaro[*], Gustavo Machado Possebon[†]
Faculdade de Informática — PUCRS

17 de Maio de 2019

Resumo

Este relatório descreve uma alternativa de solução para o primeiro problema proposto na disciplina de Programação de Software Básico no semestre 2019/1, trata-se de aplicar uma técnica muito comum e conhecida como Esteganografia, onde é possível ocultar informações em diversos tipos de mídias.

Introdução

Dentro da disciplina de Programação de Software Básico, o paradigma de programação desenvolvido durante todo o semestre é o C que tem por propósito ser estrutural, imperativo e procedural. Principais vantagens mediantes as outras linguagens são desde os compiladores mais eficientes até o acesso e manipulação irrestrita da memória, o que torna a melhor opção para desenvolvimento de software base a outros.

Sobre a Esteganografia acredita-se que é utilizada por espiões em todo mundo pois tem como principal objetivo mascarar as informações obtidas. Para realizar o processo de Esteganografia podem haver diversos graus de complexidades, de forma que optamos pelas alterações nos bits menos significativos de cada pixel, basicamente se dá pela substituição de um ou mais bits por uma parte da mensagem.

Desenvolvimento

Dividiu-se o desenvolvimento do algoritmo de Esteganografia em duas partes fundamentais, a classe para codificar a senha e mensagem inseridas na primeira versão da imagem, posteriomente a classe para decodificar a mensagem mediante a mesma senha definida. Utilizamos como base de embaralhamento de palavras o código de César ou mais conhecido como a cifra de César que faz a troca de uma letra mediante a um número fixo de repetições posteriores, como por exemplo de A se substitui para D e B para E, e assim por diante.

Classe *Main*

Após todo o processo de compilação executado pelo GCC(*GNU Compiler Collection*), a *main.c* é a responsável pelo restante do trabalho pois é onde se encontra todo o código fonte do algoritmo.

[*] b.cesaro@edu.pucrs.br
[†] gustavo.possebon@edu.pucrs.br

Nas primeiras linhas são feitas as inclusões das bibliotecas tanto padrões da linguagem quanto as desenvolvidas por terceiros. Em seguida, os dois tipos de estruturas que serão utilizadas durante todo funcionamento, algumas variáveis de escopo global. Os protótipos são parte importante do código em C pois são declarados apenas os nomes e tipos de cada função a partir da compilação ja se faz a verificação se são válidos, caso não, ja não permite prosseguir. Funções mais utilizadas sem os argumentos:

- load() : por meio da biblioteca SOIL carrega imagem se possível para variável pic da estrutura Img
- testbit() : uma função comum conhecido como Get
- setbit() : substituí o bit desejado
- clearbit() : limpa o bit enviado
- blink() : executa um pulo baseado no tamanho da frase divido pela letra
- toLetter() : transforma a posição do pic em binario

No final da classe é onde acontece as entradas e saídas, impressões de tela com os valores e verificações quanto a senha e texto.

Função *Encrypter*

O cabeçalho do protótipo da função *Encrypter* é composta pela senha, um texto que é transformado em vetor e os argumentos. É criado também um vetor de caracteres auxiliar para onde serão enviados os caracteres após a codificação, um char para se comparar. É feito uma verificação se os argumentos são válidos após isso é feito a cifragem de César(soma-se mais dois para cada letra) em cima do texto recebido. Pega-se cada posição de cada pixel internamente para se pegar em binário, e logo após é modifica-se o bit menos signficativo da estrutura Img, executado 8 vezes para cada pixel, e o processo se repete para a proxima letra. Também executa-se um pulo para atualizar a posição do vetor baseado no tamanho da frase e a letra.

Após feito a inserção do texto em meios aos bits menos significativos, zera-se os bits criados no escopo global, e salva-se a imagem em *saida.bmp* e libera o espaço de memória utilizado para a estrutura de Img.

Função *Decrypter*

O cabeçalho do protótipo da função *Decrypter* é baseado pela senha(mesma passada anteriormente), calculo do tamanho do texto e o argumento da imagem de saida gerada antes. Algo crucial é a utilização do método *calloc* que permite alocar um espaço especial em memória para a mensagem codificada. Carrega-se a imagem para *pic* e realiza a varredura pegando o bit menos significativo de cada pixel escolhido e transformando-o em letra. Após o proximo nos 8 bits de cada R, G e B é feito a inversão dos reais valores das posições, tirando-se 2 de cada, imprime a mensagem criptografada e descripritografada, por último é feito a limpeza na memória.

Conclusões

Encontramos dificuldades na implementação pois é necessário compreender desde uma estrutura de dados até a manipulação de memória em baixo nível(bit a bit). Aproveitamos uma biblioteca antiga

chamada SOIL para manipular a imagem utilizada durante todo o programa. Outra dificuldade enfrentada foi na forma como iriamos aplicar o salto a ser feito entre os pixels da imagem, bem como em uma implementação fácil e de bom entendimento do algoritmo para criptografar a mensagem a ser gravada.

Quanto a estrutura do programa, acreditamos que a eficiência possa melhorar a partir da criação de classes separadas, uma para codificar e outra para decodificar, assim apenas instânciando-as na classe principal. Um dos pequenos problemas de se manipular uma imagem em formato "jpg"é que no momento da compressão para outro formato ("bmp"), este tipo de arquivo perde uma certa quantidade de dados.

Concluimos que a implementação atende os requisitos propostos mas não na forma como nós gostaríamos e que a falta de tempo/conhecimento não permitiu uma maior eficiência neste trabalho.