MC920 – Trabalho 4

Rafael Sartori M. Santos, 186154

31 de outubro de 2019

1 Introdução

O objetivo do trabalho é a codificação e decodificação de mensagens dentro de imagens e verificar se as novas informações produzem algum artefato visual na imagem. Como exigido pelo enunciado, haverá 2 programas: um para codificação e outro para decodificação.

Faremos esses programas utilizando Python com as bibliotecas *OpenCV*, *NumPy*, *pycrypto* (sua implementação atualizada, *pycryptodome*) e padrão.

2 Metodologia

No programa de codificação, precisamos receber uma imagem e um arquivo a ser codificado. Então abrimos essa imagem, alteramos um único *bit* para cada ponto e camada da imagem, escrevendo a mensagem que desejamos, e salvamos a imagem de saída.

No de decodificação, recebemos a imagem com a mensagem codificada, abrimos e isolamos o *bit* desejado, acumulando num vetor binário que será posteriormente convertido em um arquivo de texto decodificado.

Com esses passos gerais, começamos preparando a imagem, alterando seu formato em um programa externo.

2.1 Preparação da imagem

É um requisito trabalhar de forma *lossless* (sem perdas) para manter todas as informações da imagem intactas, garantindo a integridade da mensagem. Isto é, algoritmos de compressão *lossy* podem degradar a imagem e possivelmente causar a corrupção da mensagem.

Apesar disso, a imagem carregada pelo *OpenCV* não precisa ser de um formato *lossless* pois a biblioteca consegue abrir em um formato e salvar em outro. Há, portanto, apenas uma importância: salvar a imagem sem perdas.

Para isso, o programa sempre exigirá o formato `PNG` para a saída, adicionando a extensão "`.png`" caso não esteja mencionado na entrada.

2.2 Preparação da mensagem

O programa será capaz de encriptar a mensagem de forma segura utilizando uma senha e o algoritmo `AES`, produzindo um vetor de *bytes*. Caso a senha não seja fornecida, a mensagem é interpretada apenas pelos *bytes* que compõem a *string*.

Esses *bytes* serão divididos em uma matriz de mesmo tamanho da imagem, contendo apenas um *bit* por posição correspondendo a imagem. Ou seja, dividiremos cada *byte* em 8 *bits* na matriz que possui dimensão igual à da imagem.

A matriz precisa ser menor ou no máximo de mesmo tamanho à imagem e cada valor precisa ocupar no máximo um *bit*. Caso isso não for verdade, não conseguiremos incluir toda mensagem na imagem sem maiores perdas, pois nos comprometemos pelo enunciado a utilizar apenas um plano de *bits* da imagem. O efeito de qual plano de *bits* devemos escolher para diminuir artefatos visuais será discutido na seção 2.2 com os resultados.

Um problema que haverá com esta metodologia é que, como os *bytes* retornados pelo `AES` não representam um texto em `ASCII`, não podemos confiar que haverá um caractere terminal `\0`. Devemos, portanto, incluir alguma forma de saber onde a mensagem secreta acaba. Fazemos isso através de um inteiro a ser colocado no preâmbulo: o número de *bytes* da mensagem a ser escrita na imagem.