UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO

Processamento de Imagens - SCC0251
Esteganografia: Sound Spy - Relatório Final

Docentes: Dr. Moacir Antonelli Ponti

Nome: Leonardo Cesar Cerqueira NUSP: 8937483

1. Descrição do Problema

O objetivo do projeto é criar um programa que, com entrada de um arquivo .wav, uma chave de codificação e uma imagem .png, seja capaz de esconder o .wav na imagem de forma imperceptível ao olho humano, e indecifrável por força bruta. O mesmo programa também fará o processo contrário, porém somente se for usada a mesma chave, para decodificar, detectar e decifrar o .wav dentro da imagem

1.1. Arquivos exemplo

- Imagens (imagens .jpg foram convertidas para formato .png):
  - orchestra.jpg
  - collector.jpg
  - garrus.png
  - pikachu.jpg
  - mario.jpg
- Sons (todos em formato WAV 16 PCM):
  - fledermaus.wav, retirado e convertido de: https://www.youtube.com/watch?v=gPybrOxRoT4
  - collector.wav, retirado e convertido de: https://www.youtube.com/watch?v=cTdJ6FVYSR8
  - calibrations.wav, retirado e convertido de: https://www.youtube.com/watch?v=WQ5_fya5IdI
  - pikapi.wav, retirado e convertido de: https://www.youtube.com/watch?v=ESUbFL1Gaos
  - mario.wav, retirado e convertido de: https://www.myinstants.com/instant/its-a-me-mario/

Inicialmente, foram usados somente os 2 bits menos significativos da imagem para esconder as samples do arquivo de áudio. Para testar, foram usadas 3 imagens com 3 áudios, descritos no arquivo README.

2. Solução implementada

Para esconder a imagem, as samples do arquivo de áudio (16 bits) são dividas em grupos de bits, definidos pelo usuário (1, 2, 4, ou 8 bits). Esses grupos substituem o mesmo número de bits menos significativos de pixels aleatórios da imagem. É definida uma seed para o gerador aleatório, que é a "chave de codificação" necessária para recuperar o arquivo .wav.

2.1. Passos para esconder

1. A chave fornecida ao programa é usada como seed para números aleatórios
2. O valor de sample rate é divido em bits e armazenado nos bits menos significativos de posições da imagem geradas aleatoriamente
3. O mesmo é feito para o número total de samples do arquivo
4. O mesmo é feito para cada sample do arquivo
5. A imagem modificada é salva

2.2. Passos para recuperar

1. A chave fornecida ao programa é usada como seed para gerar os mesmos números aleatórios da fase de esconder
2. Primeiramente, o valor de sample rate do arquivo escondido é recuperado dos bits menos significativos das posições "aleatórias" da imagem
3. O mesmo é feito para o número total de samples do arquivo, N
4. O mesmo é feito N vezes para recuperar as samples em significativos
5. Os dados recuperados são usados para montar e salvar o arquivo .wav

3. Resultados

Primeiramente, foram testados os resultados usando 2 bits significativos com arquivos de som adequados ao tamanho da imagem. Esses resultados estão disponíveis no repositório (imagens com prefixo "hidden).

1- orchestra_clean.png x fledermaus.wav

Antes (imagem limpa):

Depois (imagem com arquivo escondido | seed = “tom&jerry”):

2- collector_clean.png x collector.wav

Antes (imagem limpa):

Depois (imagem com arquivo escondido | seed = “harbinger”)

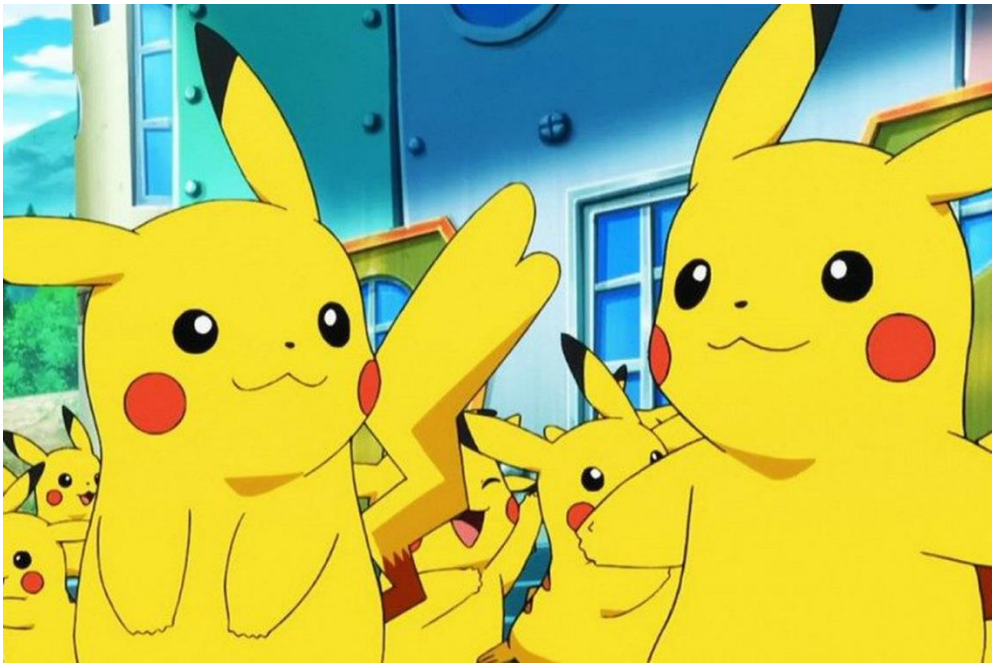
3- garrus_clean.png x calibrations.wav

Antes (imagem limpa):

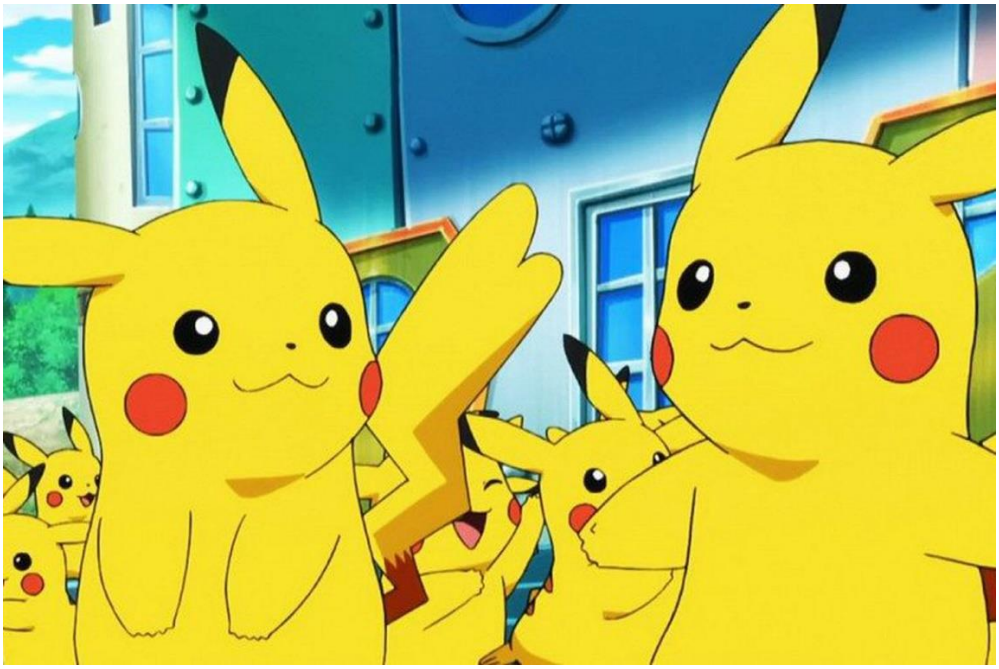
Depois (imagem com arquivo escondido | seed = “vakarian”)

4- pikachu_clean.png x pikapi.wav

Antes (imagem limpa):



Depois (imagem com arquivo escondido | seed = “the_very_best”):



5- mario_clean.png x mario.wav

Antes (imagem limpa):

Depois (imagem com arquivo escondido | seed = “italian_plumber”):

Pode-se observar que, visualmente, as imagens não aparentam ser diferentes de maneira alguma. Isso era esperado, pois apenas 2 bits menos significativos são utilizados. Porém, a diferença é melhor visualizada calculando o erro RMSE entre os pares de imagens:

1- orchestra_clean x orchestra_hidden → 0.7398
2- collector_clean x collector_hidden → 1.1231
3- garrus_clean x garrus_hidden → 0.7102
4- pikachu_clean x pikachu_hidden → 0.7954
5- mario_clean x mario_hidden → 0.9271

Em seguida, para testar melhor os efeitos na imagem, foi usado somente o maior arquivo de áudio (fledermaus.wav), com as imagens orchestra_clean, garrus_clean e mario_clean. Foram usadas todas as opções de número de bits para observar a diferença.

1- orchestra_clean

1.1- 1 bit:

RMSE: 0.4636

1.2- 2 bits:

RMSE: 0.7398

1.3- 4 bits:

RMSE: 1.9547

1.4- 8 bits

RMSE: 2.3349

2- garrus_clean

2.1- 1 bit:

Não coube na imagem

2.2- 2 bits:

RMSE: 1.5663

2.3- 4 bits:

RMSE: 4.4574

2.4- 8 bits:

RMSE: 4.8597

3- mario_clean

3.1- 1 bit:

Não coube na imagem

3.2- 2 bits:

Não coube na imagem

3.3- 4 bits:

RMSE: 5.2956

3.4- 8 bits:

RMSE: 6.4102

Desta vez, com um arquivo de áudio maior, é possível observar artefatos nas imagens menores (garrus e mario), mais destacado em locais de transições suaves de cores nas imagens originais, que acabaram ficando mais súbitas. Alguns testes não couberam nas imagens menores, pois não haviam pixels suficientes para armazenar todos os grupos de bits.

Os exemplos de 8 bits mostram como a distribuição dos bits do arquivo de áudio é uniforme através da imagem, graças à distribuição aleatória.