

Skryvat - Steganography Tool

CSF Group Number: 13

Carlos Ribeiro, Fábio Carvalho, Pedro Dias

carlos.m.ribeiro@tecnico.ulisboa.pt, fabio.r.carvalho@tecnico.ulisboa.pt, pedro.duarte.dias@tecnico.ulisboa.pt

1. Motivação do trabalho

Nos dias em que vivemos a informação e conhecimento é talvez uma das maiores riquezas a nível global. Esta encontra-se ao alcance de cada vez mais pessoas, de todas as etnias, crenças e personalidades, com a distância de um simples clique. Como tal, muita informação circula e de longe se pode considerar privada, apesar da existência de vários métodos para a proteger, essa segurança pode ser quebrada, como já pudemos verificar vezes sem conta, em diversos casos mediáticos. Usando o referido anteriormente como ponto de inspiração, resolvemos ter uma palavra a dizer, criando assim uma ferramenta que permitisse que uma pessoa pudesse comunicar com outra, ou com uma organização, de maneira que no fosse evidente o que estava a ser transmitido, escondendo numa inocente imagem. Esta uma técnica de esteganografia. A esteganografia consiste no estudo de técnicas para omissão de mensagens, de uma maneira geral à vista de todos.

2. Objectivos

De forma a ser possível concretizar o que foi referido anteriormente esperamos construir uma aplicação robusta e sólida do ponto de vistó técnico, que forneça garantias aos seus utilizadores que a informação que estão a transmitir não irá ser comprometida de forma alguma, sendo que o utilizador a quem a mensagem se destinar vai poder ver a mensagem sem nenhuma alteração.

De forma a conseguir realizar aquilo a que nos propomos terá de ser feito um trabalho prévio de investigação, estudo e pesquisa, nomeadamente que estratégias esteganográficas se podem aplicar a esconder informação dentro de imagens, como diferentes algoritmos afectam a imagem portadora, quais as diferentes vantagens e desvantagens dos diferentes algoritmos e que garantias nos do que a imagem não será detectada por terceiros como sendo suspeita.

Numa primeira etapa espera-se utilizar um algoritmo simples, de modo a termos uma ferramenta minimamente funcional, que esconda apenas uma simples mensagem numa imagem e melhorá-la então após um conhecimento mais aprofundado de algoritmos mais eficientes e robustos.

3. Solução proposta

Numa primeira fase, uma solução simples, amplamente usada no mundo da Esteganografia é o algoritmo LSB (Least Significant Bit). Este algoritmo simplesmente divide a mensagem a transmitir em bits, coloca cada um no último bit (ou últimos bits) de cada pixel da imagem e assim o ruído na imagem é praticamente indetectável a olho humano. Este algoritmo dá-nos pouca margem para o tamanho da mensagem a transmitir (maior mensagem implica mais bits para dividir e o numero de pixéis tem de acompanhar) de maneira a que no se tenha de usar mais bits dos pixéis, e com isso danificar demasiado a imagem.

Outra solução proposta e com descodificação não evidente em caso de intromissão seria utilizar as matrizes de Transformadas do Cosseno aplicadas na imagem para esconder os bits da mensagem a transmitir. Ou seja, uma matriz DCT (Discrete Cosine Transform) gerada com base na imagem original é uma composição de um conjunto de coeficientes em que todos somados indicam como se comporta a cor ao longo da imagem. A ideia seria usar os LSB desses coeficientes para inserir dados, diluindo o impacto de cada alteração pela imagem toda.

Infelizmente já tarde no desenvolvimento do projecto reparámos que a biblioteca que até ento tínhamos vindo a usar aquiria um comportamento indesejado para certos tipos de imagens e certos volumes de dados. Através da modificação de grandes números destes coeficientes por vezes eram introduzidos erros possivelmente propagados por acumulação de várias variáveis arredondadas, o que fazia com que aleatoriamente um coeficiente não especificado mudasse o seu valor, tornados os dados inconsistentes. Devido á natureza intermitente deste problema e de já ter sido detetado numa fase tardia do desenvolvimento tivemos de abandonar esta solução. De forma a podermos compensar esta falta, aumentámos o grau de complexidade da primeira solução, nomeadamente a omissão de qualquer tipo de ficheiro e melhorámos a forma de esconder o tamanho da mensagem, entre outros aspectos.

4. Arquitectura da solução

Para a injeção de dados numa imagem, neste projeto decomponemos cada imagem nas suas três componentes RGB, codificando os dados, segundo um critério de LSBs em cada uma das variáveis destas componentes. O espaço disponível numa dada imagem é calculado com base neste valor (LSBs) em conjunto com a resolução da própria imagem, quanto maior qualquer um deles, maior será o espaço disponível. A aplicação disponibiliza 4 valores de operação para o LSB:

- Low - 1 bit de LSB
- Medium - 2 bits de LSB
- High - 3 bits de LSB
- Insane - 6 bits de LSB

4.1 Organização e estrutura da metadata

Por forma a torna viável a recuperação de dados injetados nos LSBs tivemos de desenvolver uma estrutura para armazenar de forma consistente todas as informações necessárias a essa recuperação. A metadata foi inserida nos pixéis iniciais e substitui um número standard de LSBs especificado na constante METADATALSB = 2. Esta informação pode ser decomposta da seguinte forma:

- 32 bits especificam os tamanho em bits do ficheiro escondido
- 12 bits o tamanho do nome do ficheiro

- 8 bits o numero de bits menos significativos a serem usados

Respetivamente especificadas pelas seguintes constantes:

- FILE SIZE HEADER BITS = 32
- FILE NAME SIZE HEADER BITS = 16
- LSB SIZE BITS = 8

4.2 Organização dos dados binários

Posteriormente à metadata e de forma sequencial está armazenado primeiro o nome do ficheiro, seguido do seu corpo binário, segundo as métricas especificadas na metadata, sendo que para o primeiro é feita uma conversão de formato UTF-8 para binário e para o segundo é utilizado o módulo Python *binascii* para converter directamente os bytes do ficheiro em binário.

5. Resultados de avaliação da solução

Por forma a testar a viabilidade da nossa solução decidimos editar uma imagem resultante da nossa aplicação utilizando um simples software de edição de imagem, o *paint.net*. O objectivo destes testes será através da alteração de valores como exposição contraste e valores gama tentar identificar quais as posições da imagem em que a sequência de bits do ficheiro escondido termina.

Os testes de seguida apresentados foram realizados com uma imagem original de dimensões 3000x2000 e com um executável codificado de 4.22MB

Figure 1. Imagem original



A figura 1 representa a imagem original sem qualquer ficheiro inserido. Nas figuras 2 e 3 podemos ver o resultado final da inserção do ficheiro para os settings High e Insane (3 e 6 bits menos significativos respectivamente). Como é possível observar na figura 2 a olho nu é impossível observar qualquer fronteira onde a sequência de bits escritos termina, no entanto com o setting insane, já é possível fazer esta distinção sem editar a imagem. No entanto alterando alguns valores de exposição e contraste é possível observar esta fronteira para o setting High, como demonstrado na figura 4. Para settings inferiores a High, (Low e Medium) tanto a olho nu como através da edição da imagem esta fronteira é invisível.

6. Conclusão

Com a realização deste trabalho ficámos a conhecer o conceito de esteganografia e em que consiste, observando uma parte das suas inúmeras potencialidades. O trabalho permitiu-nos ainda uma evolução significativa no uso da linguagem Python, pois foi a linguagem na qual implementámos a solução assim como de alguns algoritmos desta área, como foi o caso do uso de DCTs e do LSB.

Figure 2. Imagem Setting High



Figure 3. Imagem Setting Insane



Figure 4. Imagem Setting High editada



References

DCTs and a Case Study

LSB algorithm