Android-Based Digital Image Steganography and Steganalysis

Dominique Piens (dpiens@stanford.edu), Nathan Staffa (staffa@stanford.edu)

I. Introduction

Steganography is the discipline concerned with achieving confidential communication by hiding information in plain sight. Media for hiding this information include images, video, audio and markup languages (Papapanagiotou et al. 2005). Steganographic schemes typically exploit information redundancies which are not easily perceptible. Digital images tend to exhibit such redundancy, and thus are a popular medium for steganography. Throughout the years, many different methodologies have been proposed in the literature, of which steganographic schemes like F5, OutGuess or Yet Another Steganographic Scheme (YASS) are a sample. While some steganographic techniques operate in the primal domain, the majority of newer techniques utilize the frequency domain to conceal information. F5, OutGuess and YASS occupy this group, embedding the information to hide in the image's discrete cosine transform (DCT) coefficients and extending the range of encodable images to compressed JPEGs (Hamid et al. 2012). The inverse problem of detecting information hidden in plain sight, steganalysis, is similarly developed for digital images. For instance, Fridrich (2004) used linear discriminant analysis to achieve a detection accuracy upwards of 87% on stego-images encoded with OutGuess. Steganography has reached consumers with multiple options present in iOS and Android app stores. Typical app features are encoding and decoding of images with a private key. PixelKnot and SSE for example, use the F5 algorithm to hide text in images. However, one of the challenges of mobile platforms is limited memory and computation. The range of techniques from steganography and steganalysis that can be used reliably on mobile devices is restricted.

We present Android software to encode and decode stego-images with OutGuess using a secure key, and detect whether a digital image is a stego-image encoded with OutGuess. Decoding and encoding functions are implemented with an emphasis on reliability and quality in a mobile setting, while also prioritizing user experience. Utilizing machine learning techniques on the DCT of stego-images encoded with OutGuess, our detector acts as a classifier, from which we fit a parameterized model to also indicate a confidence in percent. To our knowledge, this work adds two novel features to consumer mobile apps: an OutGuess encoder/decoder, and a stego-image detector.

II. Methods

In order to perform both steganography and steganalysis on Android, we harness OpenCv's image processing functionality in Java which is Android compatible. Using OpenCv 2.4.13, Android libraries, and standard Java libraries, we implemented an Outguess encoder/decoder and stego-image detector.

A. Outguess Encoding/Decoding

Open source JPEG libraries are written in C. Using open source JPEG libraries to manipulate DCT coefficients and implement Outguess, then run Outguess on Android would require more focus on software pipelines than algorithmic development. In the interest of time, we chose to focus on algorithmic development and to use OpenCV to read images. We then emulate part of the JPEG pipeline. Images are first converted to YCrCb color space, then the 8-bit unsigned intensity values are centered around zero as signed 8-bit values. The DCT of each eight-by-eight pixel block is computed and the coefficients are quantized with standard quantization matrices (ITU-T T.81).

The integer-valued DCT coefficients are used for the Outguess algorithm as described by Provos and Honeyman (2003). The algorithm is depicted in figure 1. In short, a password which is shared between sender and recipient is hashed with SHA-256 to obtain a seed integer for a pseudo-random number generator (PRNG). The PRNG determines the order in which DCT coefficients are visited. If a selected coefficient has a value different from zero or one, its least significant bit will be set to that of the next bit to write from the message. In this work, a set was added to keep track of coefficient indices which have already been visited. This avoids corrupting message bits that have already been written. Once all message bits have been written, an inverse

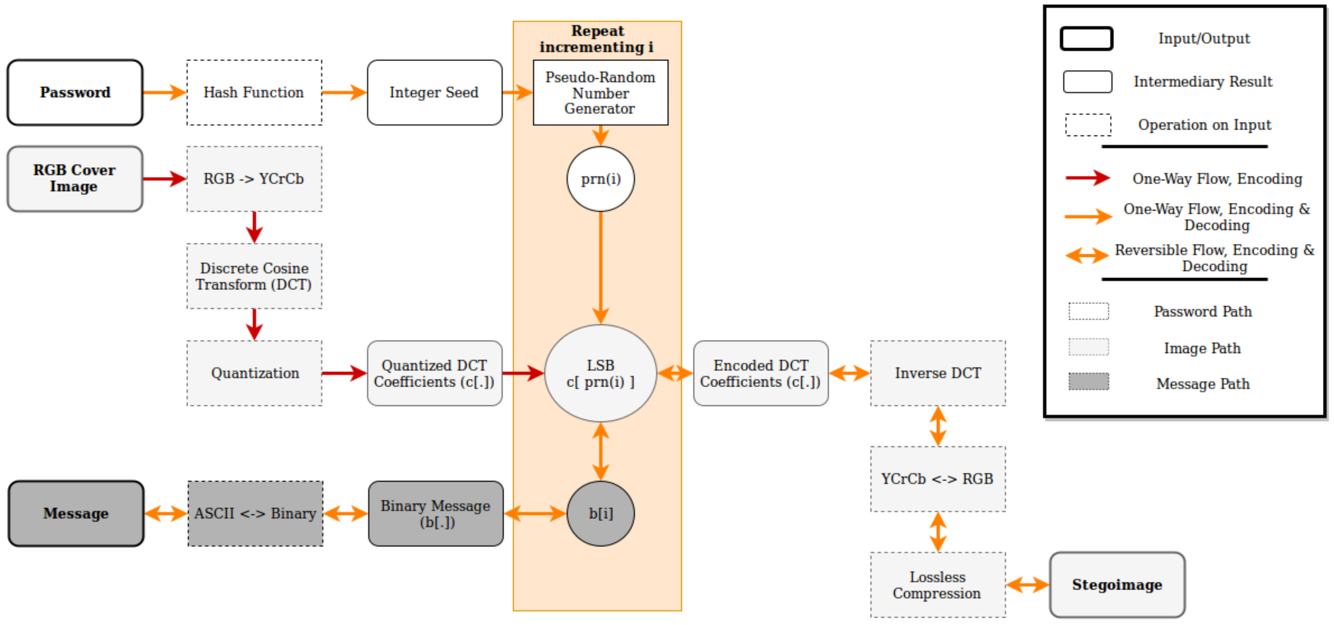

Figure 1. Block diagram of Outguess encoding and decoding.

DCT is performed and the resulting stegoimage is saved in a lossless compression format.

B. Outguess Steganalysis

General Approach. In order for steganalysis to be used on a mobile platform, our approach was to train a machine learning classifier using features which are computationally-inexpensive to obtain. We restricted the choice of model to those implemented in OpenCV in Java, namely, support vector machines (SVMs), boosted trees, random forests, naive Bayesian classifiers. Using the Outguess encoding scheme described above, a training set of 8572 images was generated, 4357 of which were stegoimages. The stegoimages were generated with passwords randomly drawn from a first text with lengths from one to 16 characters, and messages randomly drawn from a second text with lengths from one to 256 characters. The cover images were taken from the Institut National de Recherche en Informatique et en Automatique (INRIA) Holidays and Copydays datasets, as well as Libor Spacek's facial image dataset.

Feature Selection. In the literature, two pathways are taken to detect Outguess stegoimages. Farid (2002) used Fisher linear discriminant analysis to detect stegoimages. The features used were image mean, variance, skewness and kurtosis for different feature scales, and horizontal, vertical and diagonal features extracted with separable quadrature mirror filters. Provos and Honeyman (2003) expand on Farid's approach, using the distribution of squared differences rather than a wavelet decomposition, which extract directional features. Provos and Honeyman trained an SVM with the image-statistic-based features, and reported a detection accuracy ranging from about 21% to 96% on Outguess stegoimages, depending on image size.

The second approach by Fidrich et al. (2002), exploits Outguess' operating in eight by eight pixel blocks, and is based on blockiness. Blockiness is a measure of intensity discontinuity between neighboring eight by eight pixel blocks in an image. It is formally defined as

$$B = \sum_{i=1}^{\lfloor \frac{M-1}{8} \rfloor} \sum_{j=1}^{N} |g[8i,j] - g[8i+1,j]| + \sum_{i=1}^{M} \sum_{j=1}^{\lfloor \frac{N-1}{8} \rfloor} |g[i,8j] - g[i,8j+1]|$$

with M the number of rows, N the number of columns, and g the pixel intensities in a channel of the image (Provos and Honeyman, 2003). Fidrich et al. establish a linear trend in blockiness as a function of the length of messages encoded with Outguess. To establish this line, the analyzed image is encoded with the maximal length message, then a proxy for the image with no encoding is established by compressing and cropping the analyzed image. Encoding maximal length messages in the proxy for the image with no encoding twice, the process is then repeated ten times to account for the randomness introduced in Outguess by the PRNG. The length of the message encoded in the analyzed image can then be obtained accurately.

In this work, Farid's approach using image statistics was attempted. Given that there are no wavelet transform implementation in OpenCv in Java, an edge extraction operator (Scharr) was used instead to extract directional features. Scale was selected by blurring with a Gaussian kernel of different standard deviations. Lastly, image mean, standard deviation and central moments are computed for each color channel. This approach with four scales led to feature vectors of length 324. These vectors were used to train machine learning classifiers to predict whether images are Outguess stegoimages.

Fidrich et al.'s approach was also implemented, but rather than using the blockiness of five images to predict the length of the encoded message, the blockiness of each color channel of the same five images are assembled in a feature vector. Again, machine learning classifier trained on resulting features extracted from the training set.

Lastly, drawing on Fidrich et al.'s findings that blockiness can be used to detect Outguess encoding with high precision, a less computationally-expensive approach was developed. In this approach, feature vectors only contain the blockiness of the original image, and of the original image compressed with JPEG and cropped by four pixels horizontally and vertically as a proxy for the original image.

III. Results

A. Outguess Encoding and Decoding

The algorithm produces the expected results in most instances, however, there is some bit-level corruption in the conversion from DCT coefficients to an image. Testing suggests that a conversion from 8-bit signed coefficient values to 32-bit floating-point values in order to use OpenCv's DCT function is responsible for this corruption. There is likely a loss of precision at the level of the least-significant bit. To remedy this, an alternative DCT library could be used, or implemented that would operate on integers without conversion to floating-point. Overall, most messages input in the app would decode accurately on images captured in camera. The observed effect on decoded messages is that a minority of characters might be wrongly decoded.

There are also limitations due to the choice of cover image. Testing Outguess using a cover image with a digital checkerboard pattern resulted in systematic errors in decoding. This is due to the image intensities being near their extremal values, and some DCT coefficient manipulations resulting in intensity changes that are clipped as 8-bit unsigned integers. The information being lost, it cannot be retrieved when performing the DCT for decoding.

The limitations outlined here have no significant impact on the training set generated for steganalysis since the only signal that needs to be detected is least-significant bit manipulation of the DCT coefficients. The test set consists of natural images, and DCT coefficients have been manipulated when intended. Thus, the classification of the training set is not impacted by the limitations of the implementation of Outguess encoding.

B. Outguess Steganalysis

The method mirroring Farid's image-statistic based steganalysis did not result in accurate detection of Outguess stegoimages (Farid, 2002). The poor results obtained with these features here could be explained by the substitution of wavelet-based filtering by a directional edge-detection operator or by the choice of classifier. However, the feature vectors were large, requiring moderate computation time, and one of the classifiers trained here (SVM) was the same classifier used by Provos and Honeyman (2003).

Using the blockiness for five images as described by Fidrich et al. resulted in feature extraction which was too computationally-expensive to used in a mobile setting (Fidrich et al. 2002). Using the features of a subsample of 100 images from the training set, machine learning classifiers showed low prediction accuracy.

Finally, using the method developed in this work with the blockiness of two color images, a SVM with a radial basis function kernel was trained to obtain a classification accuracy (true-positive and true-negative rates) of 68.5% and a false-negative rate of 16.3%. Accuracy was determined using non-parametric Bootstrapping, which is known to overestimate prediction error (Borra and Di Caccio, 2010). Thus, 68.5% accuracy is likely a lower bound for the SVM's detection accuracy. Further testing the classifier on 100 images with fixed DCT coefficient saturation (ratio of DCT coefficients that are encoded to those that could be), coefficient saturations of 0.5% and 1% yielded accuracies greater than 92%. Nonetheless, testing images encoded on a mobile phone resulted in poor detection accuracy. The lower accuracy is likely due to messages sizes being very small (less than 20 characters on average) in relatively large images. We estimate the saturation ratio of those images was below 0.1%. Future work would characterize more fully the relation between classification accuracy and DCT coefficient saturation.

C. User Experience

The initial pre-release version of the Android application was presented for user testing during a live poster session. Users found the app intriguing, and enjoyed mostly stable performance, while recommending some potential improvements to intuitiveness. While most of the observed difficulty in use was due to lack of familiarity with the Android environment, a fatal bug was found as well when confirming a photo taken from the camera in a different orientation. While the solution to this error is known, it has not been implemented in the final application. This, along with improvements to thread management to allow for a clearer representation of the app's working on encoding and decoding, should allow for a much cleaner experience in a later version.

IV. Conclusion

An Android app featuring Outguess encoding, decoding, and detection was implemented in this work using OpenCv and Java standard libraries only. Outguess encoding and decoding is sometimes unreliable in the app, but might easily be remedied by using a different DCT library. Future work would also involve further characterization of the SVM trained with two-image-blockiness features.

References

[1] Borra, Simone, and Agostino Di Ciaccio. "Measuring the prediction error. A comparison of cross-validation, bootstrap and covariance penalty methods." Computational statistics & data analysis 54, no. 12 (2010): 2976-2989.

[2] Farid, Hany. "Detecting hidden messages using higher-order statistical models." In Image Processing. 2002. Proceedings. 2002 International Conference on, vol. 2, pp. II-905. IEEE, 2002.

[3] Fridrich, Jessica, Miroslav Goljan, and Dorin Hogea. "Attacking the outguess." In Proceedings of the ACM Workshop on Multimedia and Security, vol. 2002. Juan-les-Pins, France, 2002.

[4] Fridrich, Jessica. "Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes." In International Workshop on Information Hiding, pp. 67-81. Springer Berlin Heidelberg, 2004.

[5] Hamid, Nagham, Abid Yahya, R. Badlishah Ahmad, and Osamah M. Al-Qershi. "Image steganography techniques: an overview." International Journal of Computer Science and Security (IJCSS) 6, no. 3 (2012): 168-187.

[6] Papapanagiotou, Konstantinos, Emmanouel Kellinis, Giannis F. Marias, and Panagiotis Georgiadis. "Alternatives for multimedia messaging system steganography." In International Conference on Computational and Information Science, pp. 589-596. Springer Berlin Heidelberg, 2005.

[7] Provos, Niels, and Peter Honeyman. "Hide and seek: An introduction to steganography." IEEE Security & Privacy 1, no. 3 (2003): 32-44.

Appendix

Author Contributions

Dominique Piens: Outguess encoding/decoding development and improvement, steganalysis development and characterization.

Nathan Staffa: User Experience development, Outguess encoding/decoding improvement, steganalysis characterization.