StegExpose - A Tool for Detecting LSB Steganography

Benedikt Boehm

School of Computing
University of Kent, England
bb269@kent.ac.uk

Abstract

Steganalysis tools play an important part in saving time and providing new angles of attack for forensic analysts. StegExpose is a solution designed for use in the real world, and is able to analyse images for LSB steganography in bulk using proven attacks in a time efficient manner. When steganalytic methods are combined intelligently, they are able generate even more accurate results. This is the prime focus of StegExpose.

1 Introduction

Steganalysis is the practice of detecting the use of steganography. Steganography being the ancient practice of disguising secret communication behind a non suspect channel.

Proposed here is a steganalysis tool named StegExpose. The tool is built to be universal for detecting steganography in lossless images. StegExpose can be run in the background analysing multiple images without human supervision, returning a detailed steganalytic report once the tool has finished its job.

The organization of this paper is as follows. Section 2 defines how to interpret specialist terminology in this report. Section 3 reviews adopted technologies and literature. Section 4 discusses the steps taken to create an adequate testing environment for all steganalitic tests. Section 5 covers the attempts to find more accurate and faster steganalysis techniques and presents the test results. Section 6 covers StegExpose's implemented algorithms, features, and usage. Section 7 provides examples of how the tool can be used. Section 8 concludes the project and Section 9 discusses further directions.

2 Key Terminology

The following are descriptions of how certain terms are to be understood in the context of this report.

2.1 LSB steganography, the spatial domain and samples

LSB stands for 'Least Significant Bit' referring to the bit which makes a byte even or odd. LSB Steganography (also knows as LSB embedding) is a type of digital steganography where secrets are embedded in the least significant bit of a particular sample (or feature) of digital file. The spatial domain refers to a multidimensional space, such as the pixel plane in an image. "Samples" are features within a file that can collectively be used to carry hidden information. In lossless images, the most common samples are individual pixels.

2.2 Detectors and fusion techniques

The term detector or signal is used as a shorthand for a steganalytic method. Fusion techniques are a well known concept in signal processing and can be applied to steganalysis. The technique combines multiple detectors into one, with the intention of creating a new detector that is stronger.

2.3 Stego, carrier, cover and clean files

Stego files (also knows as carriers) are files that have embedded hidden information as a result of the use of steganography. Covers are files that can potentially be used as carriers (could be any file as long as there exists an embedding method that supports it). Clean files are files that are untouched from steganography.

2.4 Embedding rate

The embedding rate refers to the ration between the size of a payload and its cover file. For example if a cover image is 10 MB in size carrying 1 MB of hidden data, the embedding rate of the image would be 10%.

2.5 Detector success rate

Success rate is given a very specific meaning in this report. It refers to the rate at which a particular implementation of a detector is capable of calculating a steganalytic grade for a series of files.

3 Review of literature and technology

3.1 LSB embedding

In the spatial domain, LSB replacement is the most widely used LSB embedding method. LSB replacement is the process of embedding a secret as-is, so that the secret can be directly read from the LSB's without having to undergo any transformation. More complex LSB embedding methods would obfuscate the payload before embedding it with the intention to make it look statistically like a clean file. Examples include LSB matching (Sharp 2001) and Efficient High Payload Data Embedding Scheme or EPES (Omoomi, Samavi, and Dumitrescu 2011). Keeping a low embedding rate is key in preventing successful steganalysis. This means that it is desirable to embed only into a fraction of all samples (e.g. pixels in images) using a particular distribution method that would decide which sample to use and which to leave out. The importance of keeping embedding rates low is highlighted in (Ker et al. 2008). The image embedding tools used in this project are listed below.

- LSB-Steganography (David 2012) - LSB replacement with sequential distribution.
- OpenStego (Vaidya 2014) - LSB replacement with pseudorandom distribution.
- SilentEye (Chorein 2010) - LSB replacement with equidistribution.
- OpenPuff (EmbeddedSW.net 2014) - Proprietary method known as "nonlinear adaptive encoding LSB".

3.2 Steganalysis methods

The following LSB steganalysis methods have been investigated and tested as part of this project. RS analysis (Fridrich, Goljan, and Du 2001) detects randomly scattered LSB embedding in grayscale and colour images by inspecting the differences in the number of regular and singular groups for the LSB and 'shifted' LSB plane. Sample pair analysis (Dumitrescu, Wu, and Wang 2003) is 'based on a finite state machine whose states are selected multisets of sample pairs called trace multisets' (Dumitrescu, Wu, and Wang 2003). The chi-square attack (Westfeld and Pfitzmann 2000) is a statistical analysis of pairs of values (PoV's) exchanged during LSB embedding. PoV's are groups of binary values within a object's LSB's. Primary sets (Dumitrescu, Wu, and Memon 2002) is based on a statistical identity related to certain sets of pixels in an image. The difference histogram analysis (Zhang and Ping 2003) is a statistical attack on an image's histogram, measuring the correlation between the least significant and all other bit planes.

3.3 Fusion techniques

The use of fusion techniques within steganalysis is still largely unexplored. (Kharrazi, Sencar, and Memon 2006) proved how steganalyis methods can be combined or 'fused' in order to create a stronger detector. Different approaches to fusion are covered such as employing different classification stages and fusion rules. Classification stages include pre and post classification. In pre-classification individual detectors are classified as clean or stego before any further processing is done. In post-classification, various detector outputs (usually percentage values) are combined before classifying an object. Finally, a fusion rule needs to be chosen in order to derive the final indicator. The fusion rule is simply a statistical property that is taken from a set of detectors. (Kharrazi, Sencar, and Memon 2006) compares and contrasts the mean and maximum rules. More rules are covered by (Kittler et al. 1998), a paper on signal processing, which is also relevant to steganalysis.

4 Providing a test environment

In order to achieve quality test results for StegExpose, we generated a pool of 5,200 stego files and 10,000 clean files. All files were sourced from flickr.com, a large image hosting web site. Flickr.com searches were composed of keywords that were likely to return a high diversity of photographic images in terms of colours and textures. Names of countries were most commonly used as keywods. Images in the pool vary in size between 0.04 and 1.02 megapixels, averaging at 0.21. Due to the purpose of flickr.com, most images will be photographic, however non-photographic images will occur on rare occasion.

Flickr.com hosts only lossy images that are compressed using JPEG. Lossless versions (BMP and PNG) of all images were obtained. After the conversion, images were ready to form part of the pools clean portion. The stego portion had to undergo an embedding operation via SilentEye, OpenStego, OpenPuff or LSB-Steganography where each tool embedded into 1,300 images. All payloads are compressed using the zlib compression library for SilentEye and the .ZIP archive file format for all other stego tools before embedding.

Stego files created with OpenStego, SilentEye and LSB-Steganography embed the same information into all files, resulting in varying embedding rates, as all carrier files have different sizes. Stego files created with OpenPuff use batch steganography (Ker 2007). Batch steganography is when a single payload is embedded into several files using a uniform embedding rate.

The table below provides an overview of the resulting embedding rates.

	Total files	Hidden data per cover (bytes)	Average Embedding rate
LSB Steg	1,300	31,816	5.37%
OpenPuff 4.00	1,300	Variable	2.52%
OpenStego 0.6.1	1,300	163,632	25.26%
SilentEye 0.3.1	1,300	106,912	18.83%
TOTAL	5,200	n.a.	13.81%

Figure 1: Embedding rates used in the test pool

5 Experimentation and Results

The detectors used in this project were all sourced from other open source steganalysis tools. RS analysis and Sample Pair attack was sourced from Digital Invisible Ink Toolkit (Hempstalk 2006). Primary Sets and the Chi Square attack were sourced from simple-steganography-suite (Faure 2013). All detectors are automatic and return a percentage reflecting the likelihood of a file being a carrier.

The project underwent two rounds of experimentation, namely an accuracy and a speed round. The accuracy round focuses on optimizing the accuracy of a fusion detector, whereas the speed round focuses on finding a detector that provides an acceptable trade-off between accuracy and time. The rounds where necessary in equipping StegExpose with two fusion techniques, standard fusion and fast fusion. The motivation behind this is to make StegExpose relevant for academic as well as practical forensic applications.

Constants for both rounds include the test pool described in the section 'Providing a test environment'. Additional constants for the speed round include the number of time trials taken by each detector. There are three trials and the average will be used as a speed benchmark. The machine used for running the speed tests will also remain the same. The machine's specifications include a 3.40GHz Intel Core i7-2600 processor with 6 GB of RAM available.

5.1 Accuracy: finding standard fusion

The accuracy of all detectors was compared using the area under their ROC (receiver operating characteristic) curve known as the AUC. Where the true positive rate (sensitivity) is plotted over the false positive rate (fall-out). Please note that all AUC values are based on integrating high order polynomial estimates based on 23 ROC coordinates.

Three different fusion techniques were compared, one which considers only the highest scoring detector, one which considers the arithmetic and one which considers the geometric mean of all detectors.

The arithmetic mean showed the largest AUC and from this point on will be referred to as standard fusion. Standard fusion is more powerful than any of its component detectors, beating runner up RS analysis by 1.43 percentage points in AUC. Figure 2 shows a ROC curve plotting standard fusion and its component detectors (fast fusion is also plotted and will be discussed in the next section). Figure 3 gives a table of AUC values, providing a quntitative comparison of all detectors. Fast fusion is also featured in these figures and will be discussed in the next section.

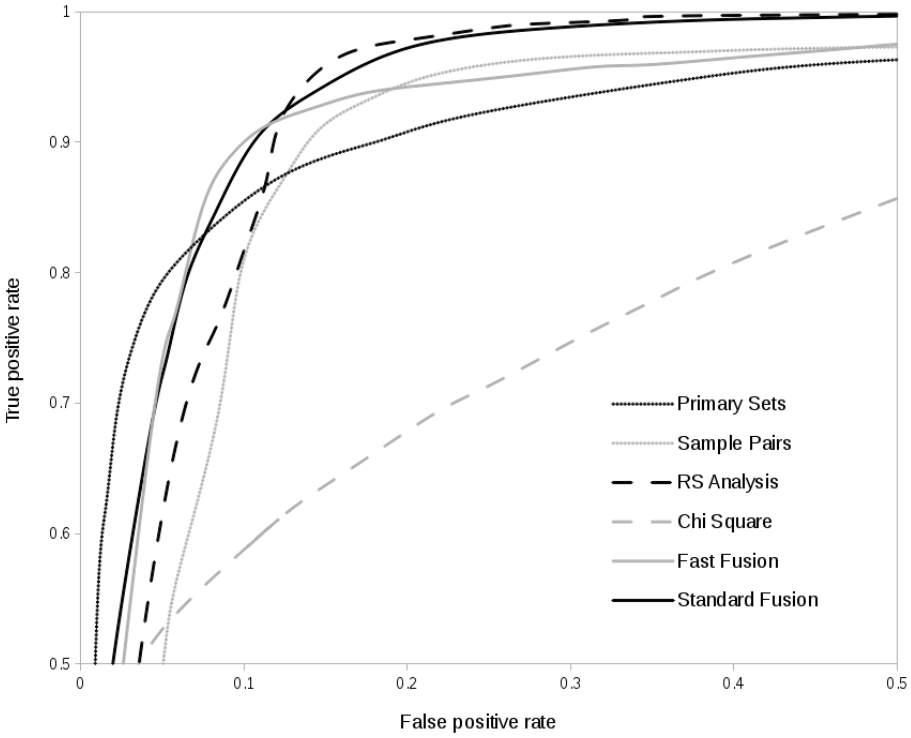

Figure 2: ROC curves for fusion detectors and their components

Primary Sets	RS Analysis	Sample Pair	Chi Square	Standard Fusion	Fast Fusion
93.27%	94.43%	91.36%	81.94%	95.86%	94.24%

Figure 3: AUC table for fusion detectors and their components

Note that if a particular implementation of a detector fails to return a result for a particular file, that file and its detection result is disregarded from the arithmetic mean in the standard fusion algorithm.

5.2 Speed: finding fast fusion

The StegExpsose project was interested in finding a second fusion technique that would offer time savings. The technique will be known as fast fusion. Instead of skipping slow detectors completely, StegExpose proposes an algorithms that tries to speed up the classification of clean files, only investing time on suspicious looking ones. This decision was made because in practical applications, clean files are a lot more abundant.

Any speed results for fast fusion will always be biased towards to the test pool, due to the nature of its algorithm. However, a conservatively high proportion of stego files (a third) in the pool should render results that would rather underestimate the speed of fast fusion.

Fast fusion consists of four stages (one for each component detector). At every stage a new component detector is added and the arithmetic mean of all currently introduced detectors is evaluated. After the evaluation, the result is compared to a specified threshold. If the result is below the threshold, all other stages are skipped and the file is immediately classified as clean. If the result is above threshold, the algorithm passes to the next stage. A file will only be classifies as stego if it passes to the final round and is still above threshold. If a component detector fails to produce a percentage value, the algorithm moves to the next stage giving the failed detector a zero weighting. Figure 4 demonstrates fast fusion's framework in a flow chart.

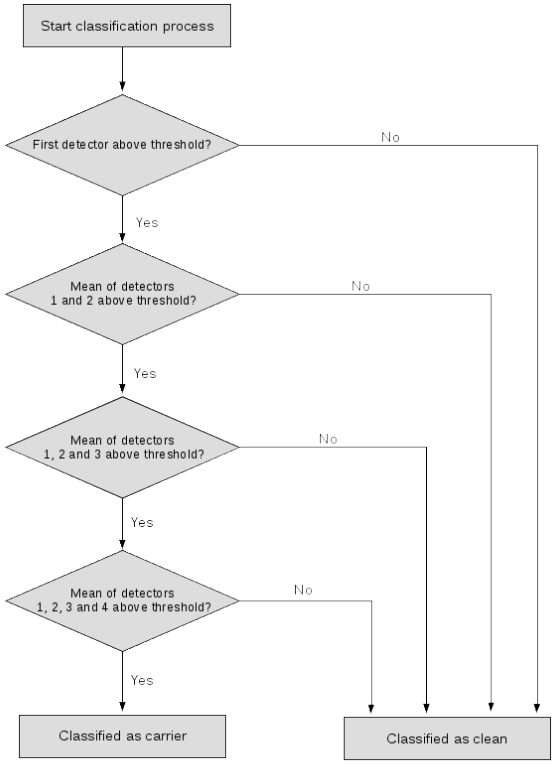

Figure 4: Fast fusion flow chart

To complete the described framework, an order of component detectors needs to be established. Initially, the order was solely based on the detector speed. This order proved to be fast but very inaccurate. After testing different orders of component detectors, a particular order proved to be fast as well as accurate. The order takes into consideration the speed as well as the accuracy of the component detectors and goes as follows: 1st Primary Sets, 2nd Sample Pairs, 3rd Chi Square and 4th RS analysis. This order has been chosen for the fast fusion which is 0.19 percentage points (in terms of AUC) less accurate that the strongest component detector (RS analysis), but therefore 3.16 times faster. Figure 3 and 5 demonstrate fast fusion's accuracy and speed respectively compared to standard fusion and all component detectors.

	Chi Square	RS Analysis	Sample Pair	Primary Sets	Standard Fusion	Fast Fusion
Detector success rate	95.59%	99.89%	42.28%	53.96%	100.00%	100.00%
Detector Speed (seconds per file)	0.073	1.082	0.142	0.054	1.204	0.342

Figure 5: Detector speed table

6 Implementation and usage of StegExpose

StegExpose is an open source Java 1.6 program available under https://github.com/b3dk7/StegExpose. There are two main aspects of StegExpose, namely detector fusion and steganalytic reporting.

The detection engine used by StegExpose features standard and fast fusion, which work exactly as described in the previous section. In order to classify images as clean or stego, a threshold must be chosen. A table linking thresholds to ROC values can be found for both fusion detectors under Figure 6 and 7. From these tables one can gather that the best trade off between fall-out and sensitivity is given at a threshold of 0.2 for both standard and fast fusion. Due to this, StegExpose will use a default threshold of 0.2 unless the user specifies otherwise. Reasons to change the threshold could be to either keep false positives at bay, in which case the threshold would be set slightly higher than the default or to reduce false negatives, in which case the threshold would be set lower. Another benefit of increasing the threshold is that the fusion algorithm will run faster, due to more frequent early classifications taking place. The threshold tables in Figure 6 and 7 can be used for guidance here. Note that for fast fusion, all decision points in Figure 4 use the same threshold i.e. the default or user specified threshold. Both fusion algorithms are implemented as modes i.e. standard and fast mode, collectively known as the fusion modes. A decision needs to be made whether to use standard of fast fusion every time StegExpose is run.

Threshold	True Negative Rate (specificity)	False Positive Rate (fall-out)	True Positive Rate (sensitivity)	Sensitivity – Fall-out
0	0	1	1	0
0.025	0.1949	0.8051	1	0.1949
0.05	0.3708	0.6292	0.9990384615	0.3698384615
0.075	0.5183	0.4817	0.9963461538	0.5146461538
0.085	0.5666	0.4334	0.9951923077	0.5617923077
0.1	0.634	0.366	0.9928846154	0.6268846154
0.125	0.7264	0.2736	0.9861538462	0.7125538462
0.15	0.8024	0.1976	0.9713461538	0.7737461538
0.175	0.8552	0.1448	0.9398076923	0.7950076923
0.2	0.8919	0.1081	0.9038461538	0.7957461538
0.225	0.9177	0.0823	0.8461538462	0.7638538462
0.25	0.9358	0.0642	0.7936538462	0.7294538462
0.275	0.9473	0.0527	0.7351923077	0.6824923077
0.3	0.9571	0.0429	0.6809615385	0.6380615385
0.33	0.9646	0.0354	0.621068879	0.585668879
0.36	0.9704	0.0296	0.5774647887	0.5478647887
0.4	0.9763	0.0237	0.5363461538	0.5126461538
0.5	0.9865	0.0135	0.4463461538	0.4328461538
0.6	0.9966	0.0034	0.3544230769	0.3510230769
0.7	0.9996	0.0004	0.2398076923	0.2394076923
0.8	0.9996	0.0004	0.1515384615	0.1511384615
1	1	0	0	0

Figure 6: Threshold table for standard fusion

Threshold	True Negative Rate (specificity)	False Positive Rate (fall-out)	True Positive Rate (sensitivity)	Sensitivity – Fall-out
0	0.0003571854	0.9996428146	1	0.0003571854
0.015	0.2349089177	0.7650910823	0.99267258	0.2275814978
0.025	0.3493273009	0.6506726991	0.9868877748	0.3362150756
0.04	0.4731515657	0.5268484343	0.9776320864	0.450783652
0.05	0.5381593047	0.4618406953	0.9710759738	0.5092352785
0.062	0.5986426956	0.4013573044	0.9649055148	0.5635482104
0.075	0.6511489463	0.3488510537	0.9595063633	0.6106553096
0.085	0.6831765686	0.3168234314	0.9579637486	0.6411403172
0.1	0.7287772354	0.2712227646	0.9517932896	0.680570525
0.125	0.784260031	0.215739969	0.9444658696	0.7287259006
0.15	0.8267650911	0.1732349089	0.9375241034	0.7642891944
0.175	0.8585545898	0.1414454102	0.9259544929	0.7845090827
0.2	0.8878437909	0.1121562091	0.9116853066	0.7995290975
0.225	0.9097511609	0.0902488391	0.8873891246	0.7971402854
0.25	0.9270151208	0.0729848792	0.8441959121	0.7712110329
0.275	0.9427312775	0.0572687225	0.7662938681	0.7090251456
0.3	0.9580902488	0.0419097512	0.6679521789	0.6260424278
0.4	0.9894035004	0.0105964996	0.33089086	0.3202943604
0.5	0.9967853316	0.0032146684	0.2487466255	0.2455319571
0.6	0.9996428146	0.0003571854	0.1866563826	0.1862991972
0.7	1	0	0.1214809101	0.1214809101
0.8	1	0	0.0532202083	0.0532202083
1	1	0	0	0

Figure 7: Threshold table for fast fusion

There are two types of reports that StegExpose is capable of producing, namely the standard and the full report. The standard report prints out to console all files classified as stego and includes an estimate of the size of the embedded data known as quantitative steganalysis. The quantitative steganalysis is derived by mutilpying the fusion detector result by the file size and dividing by three. This method was not included in the 'Experimentation and Results' section, as there was not enough scope to test this thoroughly. However, brief testing showed that the formula is seemingly accurate for covers using embedding rates above 10%.

The full report prints out the following information on all files to a csv (comma separated value) file: file name, classification (stego or clean), quantitative steganalysis (payload size in bytes - same technique as for the standard report), Primary Sets result, Chi Square result, Sample Pair result, RS analysis result and fusion result (standard or fast fusion depending on configuration). The steganalityc results for each file are flushed to the report file once fully analysed. This has the effect that any steganalytic progress is not completely lost in case the program crashes.

In order to run StegExpose, Java 1.6 or later, needs to be installed on the users machine and the StegExpose executable needs to be obtained by creating it from source or directly downloading it from the project repository, where it is saved under 'StegExpose.jar'. Below is an overview of how to run the program and a description of the arguments. Only the first argument is compulsory, however in order to set any of the optional arguments (arguments 2, 3 and 4), all arguments preceding it, must be set.

```
java -jar StegExpose.jar [directory] [speed] [threshold] [csv file]
```

Where

[directory]

Directory containing images to be diagnosed. The directory does not have to exclusively contain images, however only image files will be processed. Beware, that lossy images will be processed as well for which the implemented detectors are not designed.

[speed]

The second argument sets the speed mode. Argument *standard* will run the standard fusion algorithm and *fast* will run the fast fusion algorithm. If the argument is left out, StegExpose will default to standard fusion.

`[threshold]`

Sets the threshold, taking a floating point value between 0 and 1. If the argument is out of range, not numeric or left out then a default threshold of 0.2 is applied.

`[csv file]`

Leaving this argument out will generate a *standard* report outputted to console. Using this argument will generate a *full* report, saving it in the current directory and naming it after the given argument.

7 Examples of usage

Following are some examples in which StegExpose can be used. All examples analyse a directory containing 3 stego files (generated with OpenStego) and 13 clean files available in the project repository under the directory named 'testFolder'.

```
java -jar StegExpose.jar testFolder
```

Basic usage of Stegexpose, providing a directory of images as the only argument. As no other arguments are set, StegExpose defaults to the standard (speed) mode with a threshold of 0.2 and produces the standard report outputted to console.

```
java -jar StegExpose.jar testFolder standard default steganalysisOfTestFolder
```

Same as above but producing a full report named 'steganalysisOfTestFolder' saved under the current directory.

```
java -jar StegExpose testFolder fast 0.3
```

Increasing the threshold and running the program in fast mode.

8 Conclusion

StegExpose is a steganalysis tool heavily geared towards bulk analysis of lossless images. Two new fusion detectors, standard and fast fusion were derived from four well known steganalysis methods and successfully implemented in the tool. Standard fusion is more accurate than any of the component detectors it is derived from. Fast fusion is 0.2% weaker in accuracy than its strongest component but 316% faster. Note, that these figures are specific to the detector implementations of (Hempstalk 2006) and (Faure 2013) as well as the test pool which has a stego to clean ration of one to three. In a real world setting, the proportion of stego files will be usually a lot lower, causing fast fusion to run even faster.

9 Further work on StegExpose

Optimizing quantitative steganalysis in StegExpose is an obvious area for further work, as there has been minimal testing thus far in contrast to it forensic value.

As written, StegExpose only utilizes one processor core. Featuring multi threading capabilities could significantly increase the speed of running detectors and improve the project as long as it does not introduce any bugs.

The source code from the project's detector dependencies have remained unchanged. However based on the test pool, the detector success rate (described in the key terminology section) of the implementation of Sample Pair (Hempstalk 2006) and Primary Sets (Faure 2013) analysis is 42% and 54% respectively. These figures are very low and are caused by bugs in both dependencies. Fixing these bugs will generate more complete reports and most likely speed up fast mode as well as improve the accuracy of both fusion modes.

A long term goal for StegExpose would be to introduce image steganalysis in the transform domain (used by the popular JPEG format), as well as other media types such as digital documents, plain text, video and audio. Most importantly, reliable and fast bulk processing needs to be maintained in order to preserve relevance in the practical forensic field.

10 Acknowledgements

I would like to thank Julio Hernandez-Castro for supervising this project, proposing the idea behind it and providing invaluable advice. I would also like to thank the authors that proposed the steganalysis methods used in this project as well as Bastien Faure and Kathryn Hempstalk for making their source code freely available.

References

Chorein, Anselme (2010). *SilentEye 0.3.1*. `http://www.silenteye.org`.

David, Robin (2012). *LSB-Steganography*. `https://github.com/RobinDavid/LSB-Steganography`.

Dumitrescu, Sorina, Xiaolin Wu, and Nasir Memon (2002). "On steganalysis of random LSB embedding in continuous-tone images". In: *Image Processing. 2002. Proceedings. 2002 International Conference on*. Vol. 3. IEEE, pp. 641–644.

Dumitrescu, Sorina, Xiaolin Wu, and Zhe Wang (2003). "Detection of LSB steganography via sample pair analysis". In: *Signal Processing, IEEE Transactions on* 51.7, pp. 1995–2007.

EmbeddedSW.net (2014). *OpenPuff 4.00*. `http://embeddedsw.net/openpuff.html`.

Faure, Bastien (2013). *simple-steganalysis-suite 0.2*. `https://code.google.com/p/simple-steganalysis-suite/`.

Fridrich, Jessica, Miroslav Goljan, and Rui Du (2001). "Reliable detection of LSB steganography in color and grayscale images". In: *Proceedings of the 2001 workshop on Multimedia and security: new challenges*. ACM, pp. 27–30.

Hempstalk, Kathryn (2006). *Digital Invisible Ink Toolkit 1.5*. `http://diit.sourceforge.net/`.

Ker, Andrew D (2007). "Batch steganography and pooled steganalysis". In: *Information Hiding*. Springer, pp. 265–281.

Ker, Andrew D et al. (2008). "The square root law of steganographic capacity". In: *Proceedings of the 10th ACM workshop on Multimedia and security*. ACM, pp. 107–116.

Kharrazi, Mehdi, Husrev T Sencar, and Nasir Memon (2006). "Improving steganalysis by fusion techniques: A case study with image steganography". In: *Transactions on Data Hiding and Multimedia Security I*. Springer, pp. 123–137.

Kittler, Josef et al. (1998). "On combining classifiers". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20.3, pp. 226–239.

Omoomi, Masood, Shadrokh Samavi, and Sorina Dumitrescu (2011). "An efficient high payload±1 data embedding scheme". In: *Multimedia Tools and Applications* 54.2, pp. 201–218.

Sharp, Toby (2001). "An implementation of key-based digital signal steganography". In: *Information hiding*. Springer, pp. 13–26.

Vaidya, Samir (2014). *OpenStego 0.6.1*. `http://www.openstego.info/`.

Westfeld, Andreas and Andreas Pfitzmann (2000). "Attacks on steganographic systems". In: *Information Hiding*. Springer, pp. 61–76.

Zhang, Tao and Xijian Ping (2003). "Reliable detection of LSB steganography based on the difference image histogram". In: *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*. Vol. 3. IEEE, pp. III–545.