Compression-Based Integral Prior Classification for Improving Steganalysis

Viktor Monarev[1], Ilja Duplischev[2], and Andrey Pestunov[3(✉)]

[1] Institute of Computational Technologies SB RAS, Novosibirsk, Russia
viktor.monarev@gmail.com
[2] Novosibirsk State University, Novosibirsk, Russia
amfipter@gmail.com
[3] Novosibirsk State University of Economics and Management, Novosibirsk, Russia
pestunov@gmail.com

Abstract. We propose the integral prior classification approach for binary steganalysis which imply that several detectors are trained, and each detector is intended for processing only images with certain compression rate. In particular, the training set is splitted into several parts according to the images compression rate, then a corresponding number of detectors are trained, but each detector uses only an ascribed to it subset. The testing images are distributed between the detectors also according to their compression rate. We utilize BOSSbase 1.01 as benchmark data along with HUGO, WOW and S-UNIWARD as benchmark embedding algorithms. Comparison with state-of-the-art results demonstrated that, depending on the case, the integral prior classification allows to decrease the detection error by 0.05–0.16.

Keywords: Information hiding · Steganalysis · Support vector machine · Compression · HUGO · UNIWARD · WOW · Prior classification · SRM · PSRM

1 Introduction

The classic problem of steganalysis consists in distinguishing between empty and stego images via a bare detector; at that, all images are subject to processing. Recently, it was introduced an approach of how to exploit a prior classification in steganalysis [10], and, within it, there were proposed three possible methods of selecting a portion of the testing set such that a detection error, calculated over this subset, may be lower than that calculated over the whole set. In their paper, the authors also discussed an possibility of splitting the testing set into several subsets containing images with common (in some sense) properties and training an individual detector for each subset in order to decrease the detection error calculated over the whole testing set.

In this paper, we propose a compression-based method of how to turn this idea in practice. We suggest to split the training image set into several subsets according to their compression rate, then obtain a corresponding number of

K.-Y. Lam et al. (Eds.): ICICS 2016, LNCS 9977, pp. 134–144, 2016.
DOI: 10.1007/978-3-319-50011-9_11

non-trained detectors, and train each of them utilizing a separate subset. During the testing phase, images with a certain compressing rate should be send to the detector, which has been trained on the images that have the close compression rate. The idea of using the compression rate as an indicator for the integral prior classification came from a well-known fact that noisy images are harder to steganalyze than plain ones, but noisiness is usually tightly correlated with entropy, and therefore with the compression rate. So, we guessed that the detectors for the noisy images should be better trained with noisy images, and the ones for the plain images should be trained with plain images.

The main hypothesis, which motivated our work, assumes that the detector preceded by the integral prior classification would be more accurate than the detector alone. It is worth mentioning, that compression is a very universal tool, which has been already used in steganalysis, see e.g. [2,8]; however, these papers are devoted to distinguishing either the basic LSB steganography or to creating quantitative steganalyzers, while the current paper focuses on binary detection of the content-adaptive embedding. Moreover, in the earlier papers, the data compression was exploited for developing the stego detectors themselves, while in the current paper we do not touch the detectors, and use the compression methods in order to perform the integral prior classification.

The principle difference between the single prior classification (introduced in [10]) and the integral prior classification (being introduced in this paper) consists in the fact that the single prior classification allows to select only *a part of the testing set* which would provide higher accuracy, while the integral prior classification enhances the accuracy estimated over *the whole testing set*. Thus, although the single prior classification may obtain a rather large subset, which would be sent into the steganalyzer, all the same, it discards other images. At the same time, the integral prior classification assumes that all the testing images are subject to processing by the detector, keeping us in the traditional scenario.

Using BOSSbase 1.01 [1] as benchmark data along with content-adaptive embedding methods HUGO [12], S-UNIWARD [5] and WOW [4] as benchmark embedding algorithms we compare our results against state-of-the-art due to Holub and Fridrich [6]. Our experiments have confirmed the above hypothesis and demonstrated that prepending the integral prior classification allows to decrease the detection error of the bare detector by 0.05–0.16. For the sake of clarity, we want to emphasize that the results of the current paper are compared against accuracy of the best bare detectors, and not with that of the detectors accompanied by the single prior classification, because the latter deals only with the part of the testing set, while the integral prior classification assumes that the detection error is calculated over the whole testing set.

2 Description of Integral Prior Classification

2.1 General Scheme

A general scheme of how to train the detectors using the integral prior classification is represented at the Fig. 1. At first, we need to split the training set

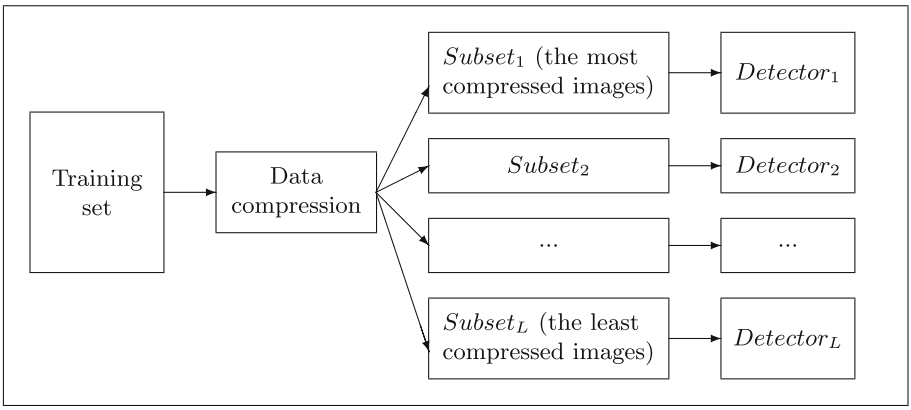

Fig. 1. The integral prior classification scheme

into subsets according to the compression rate in order to combine images with close compression rate in the same subsets. For instance, the first class contains the most compressed images, and the last class contains the least compressed images. The number of subsets would define the number of detectors for training. After the splitting, the detectors are trained: each detector is trained using the images from a certain subset. In particular, the first detector is trained using the first subset, the second detector is trained using the second subset etc.

During the testing phase, images are distributed between the detectors according to their compression rate, and each image is processed by the corresponding detector. Thus, unlike traditional bare detection scheme, which employs the detector alone, the integral prior classification assumes that the given image is sent to the detector which (we expect so), would provide the least detection error. Certainly, there may be some other ways of distributing the images (not only basing on the compression rate).

2.2 Detailed Description

In high-level steps, the detection with the integral prior classification is depicted at the Algorithm 1. The algorithm is injected with the training set $\mathcal{X}$ and the testing set $\mathcal{Y}$. Then a compression method `Compress(·)`, a number of subsets L, along with their sizes, $size_1, \ldots, size_L$ ($size_1 + \ldots + size_L = |\mathcal{X}|$), and $size'_1, \ldots, size'_L$ ($size'_1 + \ldots + size'_L = |\mathcal{Y}|$), are chosen.

The Split-Set function (see Algorithm 2) returns L non-intersected subsets which constitute a partition of the set Z. As you can see (Algorithm 1), this function is called twice: for splitting the training set $\mathcal{X}$ and for splitting the testing set $\mathcal{Y}$. The first step of this function is compressing every image $z \in \mathcal{Z}$ in order to obtain its size after compression $|\texttt{Compress}(z)|$. At the second step, the images are sorted according the this value (the sorted row we denote as follows: $z_{(1)}, z_{(2)}, \ldots, z_{(|\mathcal{Z}|)}$). And at last, the subsets are formed. The first $size_1$ images

DETECTION($\mathcal{X}$, $\mathcal{Y}$)
$\mathcal{X}$ — the training image set, $\mathcal{Y}$ — the testing image set.

1. Choose the compression function `Compress(·)`;
2. Choose the number of detectors L;
3. Choose the sizes of the training subsets $size_l$, $l = \overline{1, L}$ $(size_1 + \ldots + size_L = |\mathcal{X}|)$;
4. Choose the sizes of the testing subsets $size'_l$, $l = \overline{1, L}$ $(size'_1 + \ldots + size'_L = |\mathcal{Y}|)$;
5. $(Subset_1, \ldots, Subset_L)$:= SPLIT-SET($\mathcal{X}$, `Compress(·)`, $size_1, \ldots, size_L$);
6. $(Detector_1, \ldots, Detector_L)$:= TRAIN-DETECTORS($Subset_1, \ldots, Subset_L$);
7. $(Subset'_1, \ldots, Subset'_L)$:= SPLIT-SET($\mathcal{Y}$, `Compress(·)`, $size'_1, \ldots, size'_L$);
8. $\mathcal{Y}_{Empty} = \{\}$; $\mathcal{Y}_{Stego} = \{\}$;
9. `ForEach` $y \in \mathcal{Y}$
   (a) $detectorNumber$:= DETECTOR-NUMBER(y, $Subset'_1, \ldots, Subset'_L$);
   (b) $detectionResult := Detector_{detectorNumber}(y)$;
   (c) `If` $(detectionResult = Empty)$ $\mathcal{Y}_{Empty} := \mathcal{Y}_{Empty} \cup \{y\}$;
   `Else` $\mathcal{Y}_{Stego} := \mathcal{Y}_{Stego} \cup \{y\}$;

Result: $\mathcal{Y}_{Empty}$ — empty images (according to the detector);
$\mathcal{Y}_{Stego}$ — images with embedded information

Algorithm 1. High-level scheme of the detector with integral prior classification

are ascribed to the first subset, the next $size_2$ images are ascribed to the second subset, and so on; the last $size_L$ images are ascribed to the last subset.

Then the TRAIN-DETECTORS function (see Algorithm 3) receives L subsets and trains L detectors to distinguish between the empty and stego images. The detectors may be of different types, but, in steganography, usually the support vector machine or the ensemble classifier are employed. They utilize image features such as SRM [3], PSRM [6], SPAM [11] etc.

The testing phase is now divided into two stages: the prior classification stage and the detection stage. The prior classification stage consists in calling the DETECTOR-NUMBER function (see Algorithm 4), which returns the detector number for each image from the testing set $\mathcal{Y}$. This number is obtained according to the testing set splitting (see step 7 from the Algorithm 1) but there may be other ways of implementing it.

3 Experimental Results

We performed two types of experiments. Adjustment experiments aimed at searching for the better parameters, and benchmark experiment were performed with parameters, chosen during the adjustment experiments, and intended for comparing our results the state-of-the-art ones.

SPLIT-SET($\mathcal{Z}$, `Compress(·)`, L, $size_1$, ..., $size_L$)
$\mathcal{Z}$ — the image set to be splitted (the training or the testing set);
`Compress(·)` — the compression function;
L — the number of the subsets;
$size_l$, $l = \overline{1, L}$ — sizes of the subsets ($size_1 + \ldots + size_L = |\mathcal{Z}|$).

1. Compress every image $z \in \mathcal{Z}$ and for each image obtain $|\texttt{Compress}(z)|$.
2. Sort the images according to the value $|\texttt{Compress}(z)|$ so, that $i < j$ means $|\texttt{Compres}(z_{(i)})| < |\texttt{Compress}(z_{(j)})|$.
3. Split the set $\mathcal{Z}$ into L subsets as follows:
   $Subset_1 = \{z_{(1)}, \ldots, z_{(size_1)}\}$;
   $Subset_2 = \{z_{(size_1+1)}, \ldots, z_{(size_1+size_2)}\}$;
   $Subset_3 = \{z_{(size_1+size_2+1)}, \ldots, z_{(size_1+size_2+size_3)}\}$;
   ...
   $Subset_L = \{z_{(size_1+\ldots+size_{L-1}+1)}, \ldots, z_{(|\mathcal{Z}|)}\}$.

Result: Subsets: $Subset_1$, $Subset_2$, ..., $Subset_L$.

Algorithm 2. Splitting the set into the several subsets according to their compression rate

TRAIN-DETECTORS(L, $Subset_1$, ..., $Subset_L$)
L — the number of the detectors;
$Subset_1$, ..., $Subset_L$ — the training subsets.

1. Obtain L non-trained detectors $Detector_1$, $Detector_2$, ..., $Detector_L$;
2. Train $Detector_l$ using the images from $Subset_l$, $l = \overline{1, L}$.

Result: The L trained detectors $Detector_1$, $Detector_2$, ..., $Detector_L$.

Algorithm 3. Detectors training scheme

DETECTOR-NUMBER(y, L, $Subset'_1$, $Subset'_2$, ..., $Subset'_L$)
y — the image from the testing set;
L — the number of the testing subsets;
$Subset'_l$, $l = \overline{1, L}$ — the testing image set splitting. `For` $l = \overline{1, L}$

`If` $y \in Subset'_l$
$detectorNumber := l$;
Result: $detectorNumber$ — the detector which should process y.

Algorithm 4. Obtaining the number of a detector for the given image

3.1 Common Core of the Experiments

Images. During the adjustment experiments the image set from the Break Our Watermarking System 2 (BOWS2) contest [15] was utilized, and during the benchmark experiments—the BOSSbase 1.01 from the Break Our Steganographic System (BOSS) contest [1]. The BOWS2 image set consists of 10000 grayscale images in PGM format; the size of the images is 512 × 512. The well-known benchmark database BOSSbase 1.01 contains 10000 images captured by seven different

cameras in RAW format. These images had been converted into 8-bit grayscale format, resized and cropped to the size 512 × 512 pixels.

Preparing the Training and the Testing Sets. The both bases, BOWS2 and BOSSbase, include 10000 images, therefore we prepared the corresponding training and testing sets in the same way. In order to prepare the training set $\mathcal{X}^p$ and the testing set $\mathcal{Y}^p$, where p identifies the embedding rate in bpp, the whole database was divided into two subsets $\mathcal{X}_0$ and $\mathcal{Y}_0$, where $|\mathcal{X}_0| = 7500$ and $|\mathcal{Y}_0| = 2500$. Then by random embedding p bpp into all the images from $\mathcal{X}_0$ and $\mathcal{Y}_0$ we obtained $\mathcal{X}_1^p$ and $\mathcal{Y}_1^p$ correspondingly. The training set was $\mathcal{X}^p = \mathcal{X}_0 \cup \mathcal{X}_1^p$ and the testing set $\mathcal{Y}^p = \mathcal{Y}_0 \cup \mathcal{Y}_1^p$. Thus, $|\mathcal{X}^p| = 15000$ and $|\mathcal{Y}^p| = 5000$. Both sets contain a half of empty images and a half of stego images. Further in the paper we omit the payload index p (it will not confuse the reader) and designate the training set as $\mathcal{X}$ and the testing set as $\mathcal{Y}$.

Compression Methods. We employed well-known lossless compression methods LZMA and PAQ. LZMA (Lempel-Ziv-Markov chain-Algorithm) is a method which uses a dictionary compression scheme [13]. We launched this archiver with the following script: "lzma -k -c -9". PAQ is based on the context mixing model and prediction by partial match [14]. The launching script in our experiments was "paq -11".

Detector. We employed a support vector machine as a detector of steganography. The Python implementation was taken from [16], where the default parameters were used except for the following: the linear kernel, shrinking—turned on, and the penalty parameter $C = 20000$.

Embedding Algorithms. In the benchmark experiments, we employed three embedding algorithms: HUGO, WOW and S-UNIWARD, because exactly these algorithms were used by Holub and Fridrich in their state-of-the-art paper [6]. HUGO (Highly Undetectable Steganography) is a content-adaptive algorithm based on so-called syndrome-trellis codes [12]. WOW (Wavelet Obtained Weights) uses wavelet-based distortion [4], and S-UNIWARD [5] is a simplified modification of WOW. In the adjusting experiments only HUGO was used.

Feature Set. We utilize Spatial Rich Model (SRM) features [3] as one of the most popular instruments for steganalysis. The newer Projection Spatial Rich Model features (PSRM) [6] only slightly decrease the detection error, but significantly increase complexity. SRM features have a total dimension of 34,671.

Detection Error. We measured detection accuracy in a standard manner via calculating the detection error $P_E = \frac{1}{2}(P_{FA} + P_{MD})$, where P_{FA} is the probability of false alarms, and P_{MD} is the probability of missed detections (see e.g. [3,7,9,11]).

3.2 Adjusting Experiments

The goal of this experimental phase is to choose parameters which will be used in the benchmark experiments in order to compare our results with the state-of-the-art. The task consists in choosing the following parameters: a compression method; a number of splitting classes (L); sizes of these subsets.

Due to a long training process of SVM it was infeasible to work over many possible values of the prior classification parameters in order to search for the very best of them. That is why we have chosen three reasonable numbers of the subsets, equal to 2, 3 and 5. The thresholds for the compression rate are determined by their sizes. Here 5 subsets are of the same size, and 2 or 3 subsets have been formed by aggregation of the least compressed subsets. Trying $L = 2$ and $L = 3$ we had hoped that training the detector on images which are harder to steganalyze would provide better accuracy. However, the Table 1 demonstrate that the best accuracy is provided by $L = 5$.

Table 1. The HUGO detection error (P_E) over the whole testing set and over the subsets separately. BOWS2 image set. Search for the best parameters

		LZMA compression			PAQ compression		
	Image set	0.1 bpp	0.2 bpp	0.4 bpp	0.1 bpp	0.2 bpp	0.4 bpp
$L = 2,\ size_1 = 3000,\ size_2 = 12000,\ size'_1 = 1000,\ size'_2 = 4000$							
Integral prior classification	$Subset'_1$	0.75	0.94	0.99	0.85	0.97	0.99
	$Subset'_2$	0.50	0.54	0.69	0.49	0.57	0.69
	Whole testing set	0.55	0.62	0.75	0.56	0.65	0.75
$L = 3,\ size_1 = size_2 = 3000,\ size_3 = 9000,\ size'_1 = size'_2 = 1000,\ size'_3 = 3000$							
Integral prior classification	$Subset'_1$	0.75	0.94	0.99	0.85	0.97	0.99
	$Subset'_2$	0.67	0.85	0.95	0.65	0.89	0.97
	$Subset'_3$	0.50	0.55	0.67	0.51	0.54	0.64
	Whole testing set	0.58	0.69	0.79	0.60	0.69	0.77
$L = 5,\ size_1 = size_2 = size_3 = size_4 = size_5 = 3000,\ size'_1 = size'_2 = size'_3 = size'_4 = size'_5 = 1000$							
Integral prior classification	$Subset'_1$	0.75	0.85	0.99	0.74	0.92	1.00
	$Subset'_2$	0.67	0.71	0.91	0.54	0.74	0.95
	$Subset'_3$	0.57	0.64	0.84	0.53	0.62	0.84
	$Subset'_4$	0.56	0.55	0.76	0.52	0.54	0.71
	$Subset'_5$	0.49	0.52	0.61	0.51	0.51	0.58
	Whole testing set	0.61	0.66	0.82	0.57	0.67	0.82
No prior classification		0.525	0.60	0.77	0.525	0.60	0.77

3.3 Benchmark Experiments

The goal of this section is to demonstrate that prepending the prior classification stage (aimed at choosing the appropriate detector for each image), enhances the stego detectors accuracy. We compare the detection error with the state-of-the-art data provided by Holub and Fridrich in [6]. Unlike us, they employed the ensemble classifier [7], which is known to be faster but slightly less accurate than support vector machine. In order to be more persuasive, we calculated the detection errors for our support vector machine implementation (without prior classification) and show that they are close to that for the ensemble classifier. Anyway, integral prior classification allows to exceed both results.

See Tables 2, 3 and 4, where this comparison is provided for HUGO, WOW and S-UNIWARD embedding algorithms correspondingly. Prior classification parameters are as follows (they were chosen during the adjusting experiments): PAQ compression method; $L = 5$; $size_1 = size_2 = size_3 = size_4 = size_5 = 3000$; $size'_1 = size'_2 = size'_3 = size'_4 = size'_5 = 1000$.

The results demonstrate that, depending on the case, the integral prior classification substantially increases the accuracy. The most impressing results (see HUGO 0.1 bpp, WOW 0.1 bpp, WOW 0.2 bpp, S-UNIWARD 0.1 bpp, S-UNIWARD 0.2 bpp, S-UNIWARD 0.4 bpp) provide the accuracy decrease for more than 0.1. In the Tables 2, 3 and 4, we mark out and type in bold those values which are compared against each other. In particular, we compare the detection errors obtained for the integral prior classification against the least errors among the errors of our support vector machine (SVM) implementation and two Fridrich and Holub results. For example, in the Table 2 for HUGO 0.1 bpp we compare 0.24 against 0.35, and in the Table 3 for WOW 0.4 bpp we compare 0.08 against 0.17. As you can see, if two implementations provide the same error they are both marked out.

Table 2. The HUGO detection error (P_E). BOSSbase 1.01

	Image set	0.1 bpp	0.2 bpp	0.4 bpp
Integral prior classification	$Subset'_1$	0.01	0.01	0.00
	$Subset'_2$	0.17	0.05	0.01
	$Subset'_3$	0.21	0.11	0.03
	$Subset'_4$	0.34	0.18	0.08
	$Subset'_5$	0.46	0.30	0.19
	Whole testing set	**0.24**	**0.13**	**0.06**
No prior classification (whole testing set)	Our implementation: SVM + SRM	**0.35**	0.27	0.15
	Holub and Fridrich [6]: Ensemble + SRM	0.36	0.25	0.12
	Holub and Fridrich [6]: Ensemble + PSRMQ1	**0.35**	**0.23**	**0.11**

Table 3. The WOW detection error (P_E). BOSSbase 1.01

	Image set	0.1 bpp	0.2 bpp	0.4 bpp
Integral prior classification	$Subset'_1$	0.02	0.01	0.00
	$Subset'_2$	0.20	0.08	0.01
	$Subset'_3$	0.25	0.13	0.06
	$Subset'_4$	0.30	0.18	0.11
	$Subset'_5$	0.44	0.29	0.20
	Whole testing set	**0.24**	**0.13**	**0.08**
No prior classification (whole testing set)	Our implementation: SVM + SRM	**0.38**	**0.29**	0.21
	Holub and Fridrich [6]: Ensemble + SRM	0.39	0.31	0.19
	Holub and Fridrich [6]: Ensemble + PSRMQ1	**0.38**	**0.29**	**0.17**

Table 4. The S-UNIWARD detection error (P_E). BOSSbase 1.01

	Image set	0.1 bpp	0.2 bpp	0.4 bpp
Integral prior classification	$Subset'_1$	0.01	0.00	0.00
	$Subset'_2$	0.21	0.04	0.00
	$Subset'_3$	0.29	0.14	0.02
	$Subset'_4$	0.33	0.20	0.11
	$Subset'_5$	0.41	0.37	0.16
	Whole testing set	**0.25**	**0.15**	**0.06**
No prior classification (whole testing set)	Our implementation: SVM + SRM	**0.37**	**0.30**	**0.17**
	Holub, Fridrich [6]: Ensemble + SRM	0.41	0.31	0.20
	Holub, Fridrich [6]: Ensemble + PSRMQ1	0.39	**0.30**	0.18

4 Conclusion

In this paper we have proposed the integral prior classification approach aimed at increasing the stego detectors accuracy. Although the basic idea of this approach is rather definite, it may have many possible implementations. For instance, it may be interesting (and, what is more important, it might lead to constructing even more accurate detectors) to classify images not according to their compression rates but some how else.

In the adjusting experiments we considered only three variants of the training image set splitting. Nevertheless, it was enough to reach the goal of our research and to demonstrate that prepending the integral prior classification before

detection allows to exceed the accuracy of the state-of-the-art detectors. Thus, one of the possible future work directions may consist in conducting some theoretical research in order to elaborate recommendations of how to choose the number of subsets along with their size which would provide the better accuracy without necessity of heavy adjusting experiments.

It is worthwhile to notice, that in order to determine which detector would process which image, in the current implementation the testing set was splitted into several equal-size parts. However, it is not quite convenient if the testing images arrive one by one, unless we are able to wait until a sufficient quantity accumulates. That is why, in such a case the detector's number can be established according to the compression rates thresholds, instead of the testing set splitting.

The integral prior classification approach extends the single prior classification approach [10], which is intended for only selecting images which can be reliably detected and discards other images, though the selected images may constitute a rather large subset. The main idea of our extension is employing several detectors, each of which processes a certain testing subset or, in other words, images with special properties.

The efficiency of the integral prior classification has been demonstrated for HUGO, WOW and S-UNIWARD utilizing the BOSSbase 1.01 images. Depending on the payload and the embedding algorithm, the detection error decrease, comparing to the state-of-the-art, amounted to 0.05–0.16.

References

1. Bas, P., Filler, T., Pevný, T.: "Break our steganographic system": the ins and outs of organizing BOSS. In: Filler, T., Pevný, T., Craver, S., Ker, A. (eds.) IH 2011. LNCS, vol. 6958, pp. 59–70. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24178-9_5
2. Boncelet, C., Marvel, L., Raqlin, A.: Compression-based steganalysis of LSB embedded images. In: Proceedings of SPIE, Security, Steganography, and Watermarking of Multimedia Contents VIII, vol. 6072, pp. 75-84 (2006)
3. Fridrich, J.: Rich models for steganalysis of digital images. IEEE Trans. Inf. Forensics Secur. **7**(3), 868–882 (2012)
4. Holub, V., Fridrich, J.: Designing steganographic distortion using directional filters. In: Proceedings of 4th IEEE International Workshop on Information Forensics and Security, pp. 234–239 (2012)
5. Holub, V., Fridrich, J.: Digital image steganography using universal distortion. In: Proceedings of 1st ACM Workshop, pp. 59–68 (2013)
6. Holub, V., Fridrich, J.: Random projections of residuals for digital image steganalysis. IEEE Trans. Inf. Forensics Secur. **8**(12), 1996–2006 (2013)
7. Kodovsky, J., Fridrich, J., Holub, V.: Ensemble classifiers for steganalysis of digital media. IEEE Trans. Inf. Forensics Secur. **7**(2), 434–444 (2011)
8. Monarev, V., Pestunov, A.: A new compression-based method for estimating LSB replacement rate in color and grayscale images. In: Proceedings of IEEE 7th International Conference on Intelligent Informationa Hiding and Multimedia Signal Processing, IIH-MSP, pp. 57–60 (2011)

9. Monarev, V., Pestunov, A.: A known-key scenario for steganalysis and a highly accurate detector within it. In: Proceedings of IEEE 10th International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIH-MSP, pp. 175–178 (2014)
10. Monarev, V., Pestunov, A.: Prior classification of stego containers as a new approach for enhancing steganalyzers accuracy. In: Qing, S., Okamoto, E., Kim, K., Liu, D. (eds.) ICICS 2015. LNCS, vol. 9543, pp. 445–457. Springer, Heidelberg (2016). doi:10.1007/978-3-319-29814-6_38
11. Pevny, T., Bas, P., Fridrich, J.: Steganalysis by subtractive pixel adjacency matrix. IEEE Trans. Inf. Forensics Secur. **5**(2), 215–224 (2010)
12. Pevný, T., Filler, T., Bas, P.: Using high-dimensional image models to perform highly undetectable steganography. In: Böhme, R., Fong, P.W.L., Safavi-Naini, R. (eds.) IH 2010. LNCS, vol. 6387, pp. 161–177. Springer, Heidelberg (2010). doi:10.1007/978-3-642-16435-4_13
13. LZMA SDK (Software Development Kit). http://www.7-zip.org/sdk.html/
14. Large Text Compression Benchmark. http://mattmahoney.net/dc/text.html
15. Break Our Watermarking System, 2nd edn. http://bows2.ec-lille.fr/
16. scikit-learn: Machine Learning in Python. http://scikit-learn.org/