A Graph–Theoretic Approach to Steganography

Stefan Hetzl[1] and Petra Mutzel[2]

[1] Institute of Computer Languages (E185),
Vienna University of Technology, Favoritenstraße 9,
A-1040 Vienna, Austria
hetzl@logic.at

[2] Institute of Algorithm Engineering, LS11,
University of Dortmund, Joseph-von-Fraunhofer Str. 20,
D-44221 Dortmund, Germany
petra.mutzel@cs.uni-dortmund.de

Abstract. We suggest a graph-theoretic approach to steganography based on the idea of exchanging rather than overwriting pixels. We construct a graph from the cover data and the secret message. Pixels that need to be modified are represented as vertices and possible partners of an exchange are connected by edges. An embedding is constructed by solving the combinatorial problem of calculating a maximum cardinality matching. The secret message is then embedded by exchanging those samples given by the matched edges. This embedding preserves first-order statistics. Additionally, the visual changes can be minimized by introducing edge weights.

We have implemented an algorithm based on this approach with support for several types of image and audio files and we have conducted computational studies to evaluate the performance of the algorithm.

Keywords: Steganography, graph theory, information hiding.

1 Introduction

The purpose of steganography is to conceal the fact that some communication is taking place. This is achieved by embedding a secret message in some cover data. This process – the embedding algorithm – produces stego data which must not raise suspicion that the secret message exists. The intended receiver extracts the secret message from the stego data. Typically, the sender and the receiver must share a common secret, like a secret key in cryptography.

This paper presents a new graph-theoretic approach to steganography based on the idea of exchanging rather than overwriting samples. By exchanging samples, the secret message can be embedded while preserving the color frequencies, thus automatically avoiding detection by tests based on first-order statistics. We construct a graph from the cover data and the secret message. A vertex in this graph will correspond to the necessity of making a change to the cover data. Two vertices that are potential partners for an exchange will be connected by an edge. This approach has the following advantages:

J. Dittmann, S. Katzenbeisser, and A. Uhl (Eds.): CMS 2005, LNCS 3677, pp. 119–128, 2005.

1. It does not depend on the type of the cover data (e.g. image, audio,...).
2. It is easily extendable concerning the question which exchanges are allowed by defining additional restrictions on the set of edges. This allows for modular addition of visual and statistical criteria to the embedding algorithm.
3. It reduces the problem of finding a steganographic embedding to the well investigated combinatorial problem of finding a maximum matching in a graph (see e.g. [9,8]).

We have implemented an algorithm based on this approach in the system *steghide* [7]. Our computational experiments have shown that sufficiently large matchings can be found, so that first-order statistics are not changed substantially. Additionally, the visual differences can be minimized by introducing edge weights and minimizing the weights of all matched edges.

This paper is organized as follows. Section 2 contains a theoretical description of our new approach. In Section 3 we describe the implementation of an algorithm based on this approach. Section 4 contains a discussion of the steganographic security of our algorithm in comparison to other methods.

2 A Graph-Theoretic Approach

2.1 Terminology

The central concept for abstracting the embedding process from the underlying data format is that of a *sample*. A sample is the smallest data unit of a certain data format, e.g., the data making up a pixel in an image (a R/G/B triple in true-color bitmaps). The set of values a sample (of a certain data format) can have, is denoted as $\mathbb{S}$. A cover (or stego) file is an array of samples.

For cover (or stego) data $D = \langle s_1, \dots, s_N \rangle$ and a set $P \subseteq \{1, \dots, N\}$ the *frequency* of the sample value $x \in \mathbb{S}$ in the set P is $|\{i \in P \mid s_i = x\}|$. We define a function $v : \mathbb{S} \to \{0, \dots, m-1\}$ assigning an *embedded value* to every sample value, where m can be varied for different data formats. For a traditional least significant bit (LSB) embedding, m would be 2 and $v(s) = \mathrm{LSB}(s)$. Additionally we use a construction mentioned in [1]: We do not embed a value in a single sample, but instead in a set of k samples, more precisely, as modulo m sum of their embedded values. This has the advantage that we have the freedom to choose one of these k samples for modification. Let $D = \langle s_1, \dots, s_N \rangle$ be some cover (or stego) data. We define the value that is embedded in the i-th k-tuple of samples as: $V_i(D) = v(s_{k\cdot(i-1)+1}) \oplus_m \dots \oplus_m v(s_{k\cdot(i-1)+k})$, where $1 \leq i \leq \lfloor \frac{N}{k} \rfloor$. The secret message will be denoted as $E_m = \langle e_1, \dots, e_n \rangle$ where $e_i \in \{0, \dots, m-1\}$ for $i \in \{1, \dots, n\}$. The purpose of this notation is to make clear that the data that will be embedded is encoded in digits modulo m which is assumed for the construction of the graph. For $m = 2$, the values e_i are the bits of the secret message.

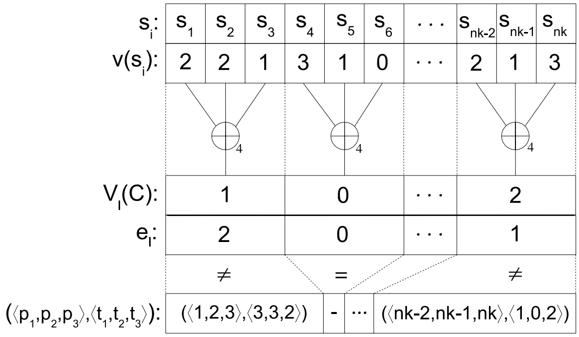
A graph G is a structure (V, E) where V is the set of vertices and $E \subseteq V \times V$ is the set of edges. We will only consider undirected graphs, so $(x, y) \in E$ and $(y, x) \in E$ are the same edge. Every edge can be assigned a weight which will be denoted by $c(e)$ for $e \in E$. A *matching* $M \subseteq E$ is a set vertex-disjoint edges, i.e. there do not exists two edges $e_1, e_2 \in M$ and a $v \in V$ such that both e_1 and e_2 are connected to v. A *maximum cardinality matching* on a given graph G is one largest (w.r.t. its cardinality) matching on G. A *maximum cardinality minimum weight matching* M is a maximum cardinality matching where $\sum_{e \in M} c(e)$ is minimal among all maximum cardinality matchings. An edge $e \in E$ is called *matched* with respect to some matching M if $e \in M$. A vertex $v \in V$ is called *matched* (in M) if there is an edge in M that is incident to v. We will sometimes write $v \in M$ for a vertex $v \in V$ to indicate that this vertex is matched in M, respectively $v \notin M$ to indicate that it is not matched. A *perfect matching* is a matching such that all vertices of the graph are matched.

2.2 Construction of the Graph

In this section we will describe the construction of the graph.

Definition 1. *Let $C = \langle s_1, \dots, s_N \rangle$ be the cover data, $k \geq 1$. A* ***vertex*** *is a structure (P, T) where $P = \langle p_1, \dots, p_k \rangle \in \{1, \dots, N\}^k$ is a k-tuple of positions in the cover file, and $T = \langle t_1, \dots, t_k \rangle \in \{0, \dots, m-1\}^k$ is a k-tuple of target values.*

The precise meaning of a vertex is as follows: Exactly one of the samples s_{p_i}, $i \in \{1, \dots, k\}$ needs to be changed to a sample value s_i^* for which $v(s_i^*) = t_i$ holds to embed a certain part of the secret message. The number k is fixed for a graph and will in the following also be called the *samples per vertex* ratio.



s_i:
s_1
s_2
s_3
s_4
s_5
s_6
...
s_{nk-2}
s_{nk-1}
s_{nk}
$v(s_i)$:
2
2
1
3
1
0
...
2
1
3
$\oplus_4$
$\oplus_4$
$\oplus_4$
$V_i(C)$:
1
0
...
2
e_i:
2
0
...
1
≠
=
≠
$(\langle p_1,p_2,p_3\rangle,\langle t_1,t_2,t_3\rangle)$:
$(\langle 1,2,3\rangle,\langle 3,3,2\rangle)$
-
...
$(\langle nk-2,nk-1,nk\rangle,\langle 1,0,2\rangle)$

Fig. 1. Example for the vertex construction with $k = 3$ and $m = 4$

After having defined the structure of a single vertex it remains to describe the construction of the set of vertices: Basically we have to create a vertex for every k-tuple of samples of which one needs to be changed. Fig. 1 shows an example of our construction for cover data $C = \langle s_1, \dots, s_N \rangle$ with $k = 3$, $m = 4$ and secret message $E_4 = \langle e_1, \dots, e_n \rangle$. The first row contains the samples of the cover data.

The second row contains the values that are embedded in the (unmodified) cover data. Three (in general k) of these values are combined using addition modulo four (in general m) to form the value $V_l(C)$ which is compared to the l-th part of the secret message (e_l). If these two values are not equal (as in the first and last case but not in the second case) a vertex is created in the last line. The target values are computed by adding the difference $d = e_l \ominus_m V_l(C)$ to each $v(s_i)$. This has the effect that replacing one of the $v(s_i)$ with its corresponding target value t_1, t_2 or t_3 yields e_l as value of the vertex. This operation is called *embedding a vertex.*

Definition 2. *Let v and w be two vertices with $v = (\langle p_1, \ldots, p_k\rangle, \langle t_1, \ldots, t_k\rangle)$ and $w = (\langle q_1, \ldots, q_k\rangle, \langle u_1, \ldots, u_k\rangle)$ and let $i, j \in \{1, \ldots, k\}$. There is an* ***edge*** *connecting the i-ith sample value of v with the j-th sample value of w, written as $(v, w)_{i,j} \in E$ if*

$$v(s_{p_i}) = u_j \text{ and } v(s_{q_j}) = t_i$$

An edge connects two vertices and is labeled with the index of one sample value from each vertex. An exchange of these two sample values results in embedding both vertices. Note that it is possible that two vertices are connected by more than one edge. In this case a matching can contain only one of these edges.

The above definition alone does not prohibit exchanges that create visible distortions such as for example exchanging a black and a cyan pixel. We need to define a restriction on this set of edges that takes *visual similarity* into account. We define a distance function $d : \mathbb{S} \times \mathbb{S} \to \mathbb{R}$ meant to capture the notion of visual distance. We define a relation of visual similarity $\sim$ for $s_1, s_2 \in \mathbb{S}$ as: $s_1 \sim s_2 \Leftrightarrow d(s_1, s_2) \leq r$ for some neighborhood radius r. Now we can give the refined definition of the set of *edges restricted by* $\sim$:

$$E^{\sim} = \{(v, w)_{i,j} \in E \mid s_{p_i} \sim s_{q_j}\}$$

where $v = (\langle p_1, \ldots, p_k\rangle, \langle t_1, \ldots, t_k\rangle)$ and $w = (\langle q_1, \ldots, q_k\rangle, \langle u_1, \ldots, u_k\rangle)$. The cost $c : E \to \mathbb{R}$ of an edge is the distance of the two sample values $d(s_{p_i}, s_{q_j})$.

2.3 Finding an Embedding

The goal of the embedding process is to find a way to modify the cover file such that all vertices are embedded. For this purpose we try to calculate a perfect matching on the graph defined in the previous section. Every matching on this graph corresponds to a set of (disjoint) exchanges of sample values in the cover file. Conducting these exchanges has the effect that all matched vertices will be embedded. Of course we cannot except to reach a perfect matching for every cover file. To embed those vertices that can not be matched it is necessary to overwrite one sample per vertex. An operation like this will change the sample value frequencies, but this operation needs to be done only on the unmatched vertices and in Sec. 3.2 we show that for natural cover data we can find matchings of sufficient cardinality.

As an additional feature one can bias the algorithm towards choosing short (w.r.t. their cost) edges to minimize the overall visual impact by trying to find a minimal weight maximum matching.

3 Implementation

We have implemented an algorithm based on our new graph-theoretic approach in the system *steghide* [7]. Our implementation supports palette images, true-color images, jpeg images, waveform audio data and μ-law audio data. As additional security measure we permute the samples before embedding in a way determined by the secret key to guarantee a uniform distribution of stego samples in the stego file (as recommended e.g. in [12]).

3.1 Data Structures and Algorithms

The number of vertices in our graphs grows in $O(n)$, where n is the size of the secret message. The number of edges grows in $O(n^2)$. For example, a graph for a true-color image and embedding data of size 1KB has about $3,200$ vertices and $98,000$ edges (on average). If the size of the embedded data raises to 4KB, then the size of the graph gets up to $12,500$ vertices and $1,470,000$ edges. However, our aim is to embed an even greater amount of data. This huge size of the graphs has two important implications: First, it is impossible to store a graph using an adjacency list, because the number of edges simply becomes too high. And second, even if there are fast algorithms to solve the matching problem running in time $O(\sqrt{|V|} \cdot |E|)$ (see [8]) it is necessary to use simple heuristics that are even faster. Note that this does not result in a decrease of the quality of the obtained matchings because it is compensated by choosing data format specific values for the parameters m, k and r that result in graphs such that heuristics can find matchings of sufficient quality (see Sect. 3.2).

In our implementation, we have not stored the graph using an adjacency list, instead we use data structures that allow an on-the-fly construction of the edges: The *sample value adjacency list* is an array indexed by the sample values that for each sample value s contains a list of those sample values that have a distance $\leq r$ to s. These lists are sorted by distance in ascending order. We also need a data structure called *sample value occurrences* that is an array indexed by the sample values that for each sample value s contains a list of pointers to those vertices that contain s. Using these two data structures we can iterate through the edges of a given vertex in order of ascending distance without the need to store an adjacency list.

As mentioned above, we use heuristics to speed up the calculation of the maximum cardinality minimum weight matching. We use the greedy construction heuristic formally described in Algorithm 1 (see, e.g., [9]).

This heuristic adds vertices ordered by their degree[1] and - for equal degrees - with their shortest edge to the matching. Sorting by degree significantly increases

[1] The degree of a vertex is the number of edges incident to this vertex.

```
Input    : Graph $G = (V, E)$
Output   : Matching $M$ on $G$

Sort all vertices by degree in ascending order into $\langle v_1, \ldots, v_n \rangle$;
Initialize $M = \emptyset$;
Mark all vertices as active;
for $i = 1, \ldots, n$ do
    if $v_i$ is active and degree($v_i$) $> 0$ then
        Set $e = (v_i, w)$ = shortest edge of $v_i$;
        Set $M = M \cup \{e\}$;
        Mark $v_i$ and $w$ as inactive and delete all of their edges from $E$;
    end
end
return $M$;
```

Algorithm 1. The static minimum degree construction heuristic (SMD)

the cardinality in comparison to a random selection because vertices that have a lower number of possible partners are matched first. Additionally sorting vertices with equal degrees by shortest edge biases the non-determinism in the algorithm towards choosing shorter edges.

After this construction heuristic, for some data formats we use a heuristic depth-first search for *augmenting paths* as postprocessing step. Augmenting paths then can be used to increase the cardinality of the matching. The interested reader is referred to [9] for details.

3.2 Computational Studies

To evaluate the performance of the implementation empirically, we created a test set of cover files for every data format. The image test sets have been created by digitizing 50 images showing natural scenes with an Epson flatbed scanner, which then have been converted to the appropriate file format: to bitmaps using 256 color palettes, to uncompressed true-color bitmaps (16.7 million colors) and to color jpeg images. The audio data has been taken from different CDs and then stored as 16 bit waveform data and converted to the μ-law format. The tests have been conducted with different amounts of random data as secret messages.

These test sets have been very useful to determine good values for the data format specific parameters: the samples per vertex ratio k, the modulus m and the radius r. As distance function we use the euclidean distance in the RGB cube for palette and true-color images, for the other formats we use the absolute distance between the sample values. Table 1 shows values for the parameters s.t. an embedding rate as high as possible is reached while still allowing nearly perfect matchings and thus preservation of first-order statistics. The embedding rate is the rate of the size of the secret message to the size of the cover file and can be calculated as $r = \frac{1}{\frac{2}{m} \cdot k \cdot s}$ where s is the size of a sample in bits. A sample in a jpeg file is a coefficient of the discrete cosine transform. Coefficients with the value 0 are not used to embed data, because setting a coefficient that is 0

Table 1. Data format specific definitions

	palette	true-color	jpeg	waveform	μ-law
radius r	20	10	1	20	1
samples/vertex k	3	2	3	2	2
modulus m	4	4	2	2	2
embedding rate	8.33%	4.16%	*5.86%*	3.13%	6.25%
algorithms	SMD	SMD	SMD	SMD,DFS	SMD,DFS

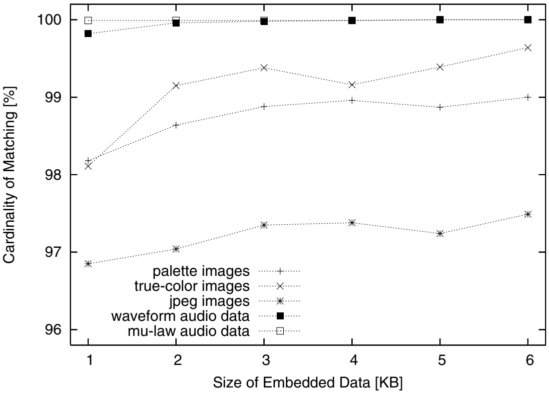


Fig. 2. Matching cardinality

in the cover file to 1 in the stego file can result in visually detectable distortion of the image. For this reason, the embedding rate of a jpeg file can be given only empirically, so the exact value given in the table is only valid for our image database, however for other natural images the value will be similar. The last row shows which algorithms are used on a specific file format. SMD refers to the static minimum degree construction heuristic (as described in Algorithm 1) and DFS refers to the heuristic depth first search for augmenting paths presented in [9]. For the audio data formats it is useful to apply the DFS postprocessing step, because the SMD heuristic already produced high quality solutions (98.2%–99.7% matched vertices for waveform data and 99.8%–99.9% for μ-law data) and the DFS postprocessing step is rather fast when applied to such high quality solutions but still yields even better solutions.

Fig. 2 shows the cardinalities of the calculated matchings. The cardinality of the obtained matchings is very high (more than 97% of the vertices have been matched). This is due to the high density of the graphs obtained using our parameter settings. For the audio data formats almost 100% of the vertices have been matched. In contrast to the audio data formats, for jpeg files only 97% can be reached. This data format is more difficult to handle, because the coefficients of the discrete cosine transform do not have a uniform distribution, instead sample values with smaller absolute values occur more often (see, e.g.

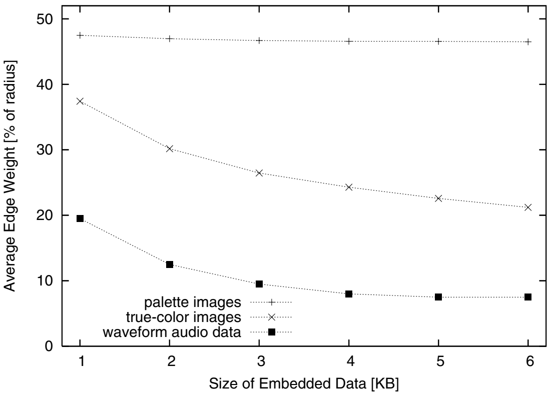


Fig. 3. Average edge weight

[12]). These results show that embeddings can be found that do not modify first-order statistics substantially.

Fig. 3 shows the average weight in % of the radius. An unbiased algorithm would choose edges with random weight with the result of an average edge weight of 50% of the radius. Biasing the algorithms towards choosing shorter edges has a significant effect for true-color images and waveform audio data.

We have observed that true-color images need considerable more running time than the other data formats, e.g. for embedding 6KB data the construction heuristic takes approximatly 12 seconds to complete compared to less than one second for the other data formats. This is due to the high number of different sample values. The problem is that, in general, creating the sample value adjacency list needs $O(|\mathbb{S}'|^2)$ time (where $\mathbb{S}'$ is the number of sample values actually occurring in the graph).

4 Steganalysis

The number of unmatched vertices is an upper bound on the number of changes to first-order statistics. Our experiments have shown that sufficiently good matchings ($< 3\%$ unmatched) can be reached for natural cover data. This makes our approach practically undetectable by tests that look only at first-order statistics such as the χ^2-attack [14]. Furthermore it is not possible to specify a set of groups partitioning $\mathbb{S}$ s.t. exchanges occur only inside a group making our approach also undetecable by the generalized χ^2-attacks in [10,4,13].

It would be interesting to run the blind steganalysis scheme [5] against our implementation to compare its detectability to the other tested algorithms [12,10,11], in particular to Salle's model-based approach. His algorithm does not only preserve the global histogram of jpeg files but also the frequencies of each individual DCT coefficent. Note that our approach could easily be extended by adding a restriction to the set of edges which allows only exchanges of sample within one DCT coefficient. This would have the effect that also the frequencies of the individual coefficients will be preserved. In fact, any restriction that can be expressed by allowing or disallowing single sample value exchanges can be added. It remains to be investigated how powerful this really is.

For a targeted steganalysis of our approach for jpeg files the blockiness measure seems to be a candidate. The blockiness measure gives an indication of the discontinuities at the 8x8 boundaries of jpeg blocks and was used in [6] to break outguess [10].

5 Summary and Outlook

We have presented a graph-theoretic approach to steganography that is based on exchanging rather than overwriting samples. Thus it preserves first-order statistics without the need for additional changes as in [10]. A graph is constructed from the cover data and the secret message where each vertex corresponds to the necessity of making a certain change and each edge represents a possible sample exchange. The embedding is found by solving the well-investigated combinatorial problem of finding a maximum cardinality minimum weight matching in this graph. The maximality of the cardinality ensures that a maximal amount of data is embedded by exchanges. The unmatched vertices need to be embedded in a way that does not preserve first-order statistics. However, as demonstrated in our computational studies, the number of unmatched vertices is negligibly low ($0\% - 3\%$) for natural cover data. Additionally the minimality of the edge weights ensures that the visual changes introduced by the embedding are as small as possible. We have implemented the algorithm with support for true-color images, palette image and jpeg images as well as waveform and μ-law audio data.

Our approach can easily be extended by adding further restrictions on the set of edges (beyond the simple visual restriction $s_{p_i} \sim s_{q_j}$). An example would be a restriction for jpeg files to preserve also the frequencies for each individual DCT coefficient. Another example is the method described in [3] (and recently broken in [2]) to determine all pairs of stochastically independent sample values. Any restriction that can be expressed by allowing or disallowing single sample value exchanges can be added.

From the point of view of combinatorics the following two extension seem interesting: 1) to allow exchanging more than one sample per vertex and 2) to allow not only (disjoint) exchanges but arbitrary permutations. Both would provide more flexibility but would amount to more difficult combinatorial problems.

Acknowledgements

The authors would like to thank the anonymous referees for important suggestions for the improvement of this paper.

References

1. Ross J. Anderson. Stretching the Limits of Steganography. In Ross J. Anderson, editor, *Information Hiding, First International Workshop*, volume 1174 of *Lecture Notes in Computer Science*, pages 39–48. Springer, 1996.
2. Rainer Böhme and Andreas Westfeld. Exploiting Preserved Statistics for Steganalysis. In Jessica J. Fridrich, editor, *Information Hiding, 6th International Workshop*, volume 3200 of *Lecture Notes in Computer Science*. Springer, 2004.
3. Elke Franz. Steganography Preserving Statistical Properties. In F.A.P. Petitcolas, editor, *Information Hiding, 5th International Workshop*, volume 2578 of *Lecture Notes in Computer Science*, pages 278–294. Springer, 2003.
4. Jessica Fridrich, Miroslav Goljan, and David Soukal. Higher–order statistical steganalysis of palette images. In *Proceedings of the Electronic Imaging SPIE Santa Clara, CA, January 2003*, pages 178–190, 2003.
5. Jessica J. Fridrich. Feature-based Steganalysis for JPEG images and Its Implications for Future Design of Steganographic Schemes. In Jessica J. Fridrich, editor, *Information Hiding, 6th International Workshop*, volume 3200 of *Lecture Notes in Computer Science*, pages 67–81. Springer, 2004.
6. Jessica J. Fridrich, Miroslav Goljan, Dorin Hogea, and David Soukal. Quantitative steganalysis of digital images: estimating the secret message length. *Multimedia Systems*, 9(3):288–302, 2003.
7. Stefan Hetzl. Steghide, `http://steghide.sourceforge.net/`.
8. Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|}|E|)$ Algorithm for Finding Maximum Matching in General Graphs. In *21st Annual Symposium on Foundations of Computer Science*, pages 17–27, Syracuse, New York, October 1980. IEEE.
9. R. Möhring and M. Müller-Hannemann. Cardinality Matching: Heuristic Search for Augmenting Paths. Technical Report 439, Fachbereich Mathematik, Technische Universität Berlin, 1995.
10. Niels Provos. Defending Against Statistical Steganalysis. In *10th USENIX Security Symposium, Proceedings*, 2001.
11. Phil Sallee. Model-based Steganography. In Ton Kalker, Ingemar J. Cox, and Yong Man Ro, editors, *Digital Watermarking, Second International Workshop*, volume 2939 of *Lecture Notes in Computer Science*, pages 154–167. Springer, 2003.
12. Andreas Westfeld. F5–A Steganographic Algorithm: High Capacity Despite Better Steganalysis. In Ira S. Moskowitz, editor, *Information Hiding, 4th International Workshop*, volume 2137 of *Lecture Notes in Computer Science*, pages 289–302. Springer, 2001.
13. Andreas Westfeld. Detecting Low Embedding Rates. In F.A.P. Petitcolas, editor, *Information Hiding, 5th International Workshop*, volume 2578 of *Lecture Notes in Computer Science*, pages 324–339. Springer, 2003.
14. Andreas Westfeld and Andreas Pfitzmann. Attacks on Steganographic Systems. In Andreas Pfitzmann, editor, *Information Hiding, Third International Workshop, IH'99, Dresden, Germany, 1999, Proceedings*, volume 1768 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2000.