JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 26, 1787-1799 (2010)

Blind Steganalysis of Mp3stego

JULIO C. HERNANDEZ-CASTRO[1], JUAN E. TAPIADOR[2],
ESTHER PALOMAR[3] AND ANGEL ROMERO-GONZALEZ[4]
[1]*School of Computing*
University of Portsmouth
Hampshire, PO1 2UP UK
E-mail: Julio.Hernandez-Castro@port.ac.uk
[2]*Department of Computer Science*
University of York
York, YO10 5DD UK
E-mail: jet@cs.york.ac.uk
[3]*Department of Computer Science*
Carlos III University of Madrid
Leganés, 28911 Spain
E-mail: epalomar@inf.uc3m.es
[4]*Enusa Industrias Avanzadas*
Madrid, 28040 Spain
E-mail: ang@enusa.es

In this paper we present a steganalytic attack against mp3stego, a publicly available open-source steganographic tool which uses mp3 audio files as the stego medium to embed hidden information. We first identify through extensive statistical experimentation which parameters of the medium suffer a significant deviation from normality, then we construct a very efficient blind steganalytic algorithm based on their use. We finally present the results obtained with this steganalytic approach over various sets of mp3 files, and an algorithm for the estimation of the amount of hidden data.

Keywords: steganalysis, mp3, audio, mp3stego, steganography

1. INTRODUCTION

Steganography means “covered writing”, a term coined in 1499 by the monk Trithemius, who devised a method to encode covert messages into prayers.

Along the years, many different methods of secret communication to conceal the existence of hidden data were employed. Among these: writing into shaved slave heads or wax-covered tables, knitting, invisible inks, microdots, combinations of the dots and dashes on *i*, *j*, *t* and *f* giving Morse codes, null ciphers, covert channels, and spread-spectrum communications.

Steganography is the art and science that studies how to hide information within seemingly innocuous media (*e.g.* images, audio files, text, html, *etc*.). It is important to note that steganography and cryptography, although related, are quite different disciplines: While cryptography tries to “cramble” messages so that if intercepted they cannot be understood, steganography camouflages messages to hide its existence and make them seem invisible, thus concealing the fact that a message is being sent altogether. An encrypted message may draw suspicion while an invisible message will not.

Received April 9, 2008; revised August 1, 2008; accepted November 12, 2008.
Communicated by Tsan-sheng Hsu.

Steganography should provide a way of secret communication which cannot be removed without significantly altering the data in which it is embedded. For the data to be recovered, an attacker should first find a way to detect it, only then being able to mount a classical cryptanalysis attack (it is common practice to cipher messages before hiding). The associated field which tries to detect, recover or eliminate messages hidden with steganographic techniques is called steganalysis.

Steganography is nowadays in common use, both for copyright protection of digital media and for exchanging information without raising suspicions (of law enforcement agencies and/or dictatorial governments, for example) There are many tools for hiding messages in images, audio files, video, and other not so common media such as text [17, 19, 20], tcp/ip packets [21], executable files [22], DNA strands [16], XML [18], *etc*. The huge amount of internet traffic allows for easy covering hidden messages in media where the addition of this extra information is very difficult to detect even when complex, time consuming steganalytic techniques are applied.

1.1 The Mp3stego Tool

The mp3stego tool [12] is a steganographic software that allows to insert and subsequently extract hidden information into mp3 files. The tool is freely distributed together with its source code in C. It was created by Fabien A. P. Petitcolas [5, 12, 14] in the Computer Laboratory (Cambridge) in 2002, after observing that, among the broad range of available steganographic tools, there was no one to hide information into mp3 files. Besides being mp3 an extremely popular format – it offers a perceived quality similar to that provided by a CD with an approximate compression rate of 11/1 at 128 kbps – its usage as a stego medium is not as easy as with many other audio formats, mainly because the high compression rate significantly reduces the redundance needed to embed hidden data.

In mp3stego, information is embedded during the compression process described in the mp3 specification. Before being incorporated into the file, the text to be hidden is first compressed and enciphered. Even though we will not elaborate on this feature, it is worth noting that mp3stego can be also used as a watermarking system for mp3 files.

To date, the only known attack against the scheme used by mp3stego consists in restoring (evidently with losses) the audio signal, and generating the mp3 file again. This process deletes the hidden information, but also causes a significant loss of audio quality. In [15] an in-depth study of this tool and its steganographic capabilities is presented.

1.2 Overview

In this article we present a study that tries to find if there are statistical significant differences between the mp3 files generated by mp3stego, that carry hidden information, and the same files when no information has been inserted. So we try to determine if the insertion of this additional information by means of the technique employed by mp3stego significantly alters the cover medium. Based in the analysis of these differences, we propose a method to decide whether or not an mp3 file carries out contents hidden with mp3stego. Finally, we conclude that the proposed technique, apart from distinguishing mp3 files with steganalytic contents, is able of accurately giving an estimation of how

many bits were actually embedded. The aggregation of the results of these two algorithms (detection plus estimation) constitutes a very powerful attack on the mp3stego application.

2. PRELIMINARY EXPERIMENTATION

2.1 A Brief Note about the Mp3 Audio Format

The MPEG-1 Audio Layer 3, commonly known as mp3, is a popular encoding and compression scheme designed to reduce the amount of data required to represent a digital audio signal, yet still keeping a reasonable quality with respect to the original, uncompressed sound. Basically, the mp3 scheme uses two different compression techniques. First, a lossy filtering is applied, in which certain contents are discarded simply because they are not audible by the vast majority of listeners. A lossless compression is then applied to the filtered signal, which removes redundancies by using a semioptimal compression through a Huffman code [1].

An mp3 coder uses a psychoacoustic model based on the subjective human perception of sounds. In practice, this stage is implemented by means of a bank of filters that extract and reorder the spectral information in a number of frequency bands. The perceptual model determines which frequencies can be removed and which must remain unaltered. After the quantization stage, the signal is codified into a bitstream of compressed digital audio.

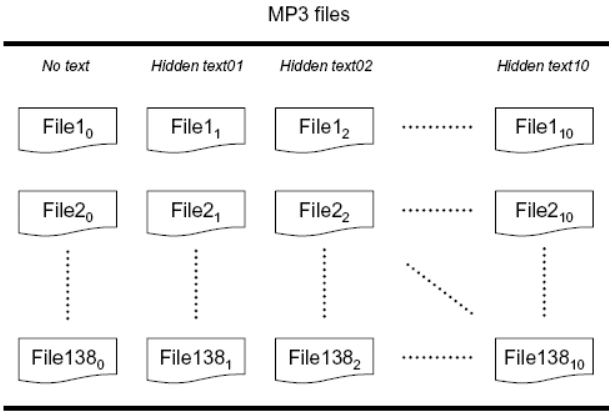
There are a huge number of sources about the details of the mp3 standard. Apart from the official documents, interested readers can find further information in [2, 3]. A more general introduction can be found in [4], while [13] provides a presentation rich in technical details.

2.2 Test-Bed

The first step in our research was to determine how relevant the differences between a file with and without hidden information embedded by mp3stego actually are. However, both the information to be inserted and the inherent features of the audio file can exert some influence in the final results.

To yield an acceptable level of generality in our conclusions, we prepared a test-bed composed of 138 sound files extracted from different music CDs. Each file was first converted to wav format, for this is the input format required by the mp3stego tool. The wav (*waveform audio file*) format is original from Microsoft Windows 3.1 and quite common nowadays. In a wav file, the samples are stored one after another, without applying any kind of data compression and using a uniform quantization. A header at the beginning of the file contains some additional information, such as the sampling frequency.

The wav version of each file has been used to generate 11 mp3 files: 1 without any embedded hidden text, and the rest embedding different messages. For these, we have used 10 fixed text files with lengths ranging from 29 bytes (the shorter) to 5507 bytes (the longer). A scheme of the test-bed preparation is shown in Fig. 1.



Hidden file	Length (bits)
Text 01	44056
Text 02	26080
Text 03	14960
Text 04	10672
Text 05	7288
Text 06	4040
Text 07	2368
Text 08	952
Text 09	632
Text 10	232

Fig. 1. Test bed.

For a given song, the 10 mp3 files containing embedding data have been compared with the mp3 file without any hidden information. Three different comparison methods have been employed:

- Subjective hearing tests.
- Tests based on comparisons of some signal parameters (mainly waveform and spectrum).
- Tests of structural parameters of the mp3 files.

The first two groups of tests did not offer any concluding result. On the contrary, the analysis of structural parameters of the mp3 files reveals that these variables can be essential to perform an steganalysis.

2.3 Variational Analysis of Mp3 Parameters

With the aim of performing an analysis of the structural features of the mp3 files included in the test-bed, we have used two different tools. First, the software EncSpot Ba-

sic 2.0 allows us to extract the data contained in an mp3 file. Moreover, the package Minitab 14 has been used to carry out the statistical analysis.

We have analyzed the set of mp3 parameters of each couple of files (one with and another without hidden data). During our preliminary experimentation, we observed that, in the vast majority of the cases, there were no statistically significant differences between the values of the parameters contained in a "normal" mp3 file and those of an mp3 file with embedded information. The two most notable exceptions are the *Max. Reservoir* and the *Av. Reservoir* fields.

The function of these fields is the following:

- The *Max. Reservoir* field contains the maximum length of the reservoir field within a frame of the mp3 file. The reservoir is the free space in an mp3 data frame. It could be used by another frame which does not have space to store its data. The length of the reservoir varies between 0 and 511 bytes.
- The *Av. Reservoir* is the average length of the reservoir field computed over all the frames contained in the mp3 files.

In the case of the *Av. Reservoir* field, the analysis has been carried out taking into consideration different amounts of frames of the mp3 file. In particular, we have analyzed segments of the mp3 files containing the first 50, 100, 150, 200, 250, 300, 350, 400, …, 1024 frames. The reason behind this procedure is simple: the mp3stego tool codifies in each frame just 1 bit of the message to be hidden, in a consecutive manner, and starting at the first frame of the file. Therefore, when a short message is embedded in an mp3 file, just the first few frames are actually affected by a variation in the *Av. Reservoir* field. If we embed an 80-bit message into an mp3 file, this will be inserted in approximately the first 80 frames. If the file has 2000 frames, the variation of the reservoir length – computed over all the frames – will not reflect any anomaly. Therefore, it is convenient to limit our observation to the first frames. This rule is heuristic, for not all the frames will have space available (*i.e.* a *Max. Reservoir* field greater than 0) to embed data.

An additional important point to keep in mind is that, as a general rule, the length of the message will not coincide with the length of the embedded message, as the original one is both compressed and ciphered before being inserted among the frames.

Next we provide the results obtained when measuring the value of the *Av. Reservoir* parameter in the first 50, 100, 150, 250, 500, 1000, 1500, and 2000 frames. At the end of the *x* axis we also provide the value corresponding to the average when all the frames in the file are considered (value *All*). We have analyzed in detail three messages in increasing order of length, with the aim of verifying if, as the amount of embedded data increases, the number of frames affected by a high value of the *Av. Reservoir* field also increases. This phenomenon is clearly observed in all the graphs shown in Fig. 2. To get a general view of this effect, graph Fig. 2 (d) shows the average result when all the messages are considered.

Next we discuss in detail four illustrative cases.

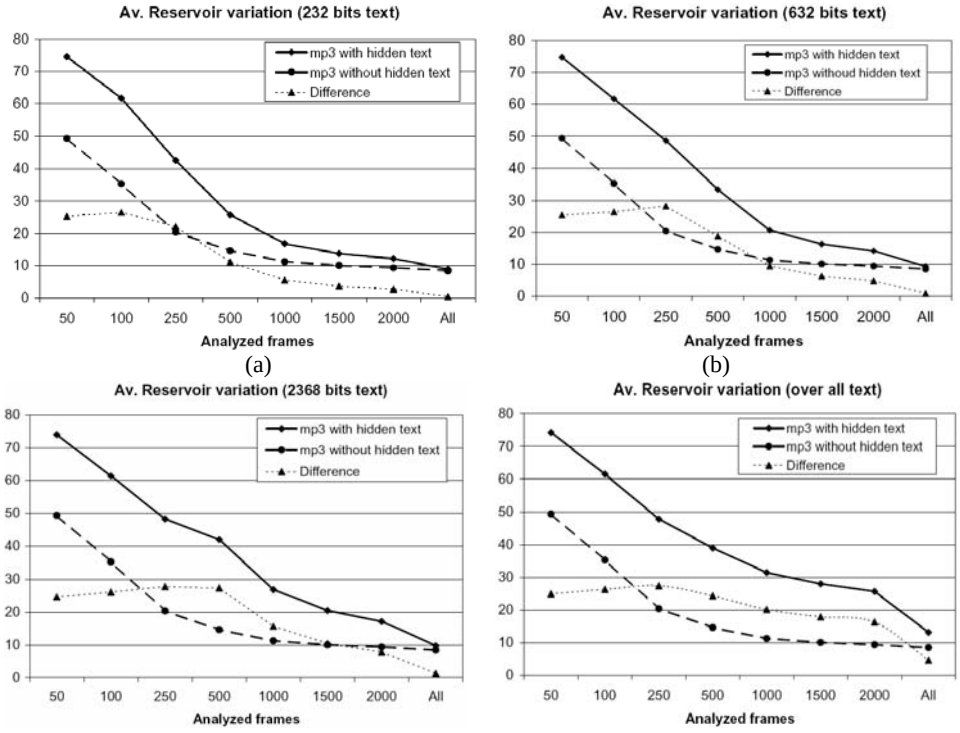


Fig. 2. Variation of the (*Av. Reservoir*) value with hidden texts of different sizes.

2.3.1 Case study: text10 (232 bits)

This first analysis corresponds to embedding the message labeled “text10”, which is 232 bits length. In Fig. 2 (a) it is observed that the value of the *Av. Reservoir* field is clearly higher in the case of mp3 files that hide this message than in those which does not. In particular, the numerical difference in this parameters is around 22-26 bits during the first 250 frames. From that point on, the difference is gradually decreasing. As the original message has a size of 232 bits, it is expected that it will be embedded approximately among the first 232 frames. Therefore, just the first 232 frames will suffer an increase in the *Av. Reservoir* parameter. Experimentally, it is observed that the number of affected frames is around 250.

2.3.2 Case study: text09 (632 bits)

The analysis concerning the message of 632 bits length (“text09”) is analogous to the previous one. Fig. 2 (b) reflects a behavior similar to that of Fig. 2 (a). In this case, however, the differences in the size of the reservoir ranges between 19 an 28 bits during the first 500 frames. From that point on, the difference decreases until being negligible. These values suggest that the original message has been compressed in, approximately, 500 bits.

2.3.3 Case study: text07 (2368 bits)

The results for the case of a text of length 2368 bits ("text07") are similar to those commented above. In Fig. 2 (c) it is observed that difference ranges between 16 and 28 bits during the first 1000 frames (so 1000 seems to be the size of the message after its compression).

2.3.4 Case study: analysis with all the messages

For this experiment, we have carried out the following task: Each of the 10 messages shown in Fig. 1 has been embedded into each of the 138 mp3 files, resulting a set of 1380 mp3 files with hidden information, and the original 138 mp3 files. Next, we have computed the difference between the size of the reservoir between the original mp3 file an each of the mp3 files with embedded data, as it has been performed in previous experiments. Finally, we have computed the average of the difference. The result is shown in Fig. 2 (d).

The difference in the size of the reservoir keeps in the range of 16-29 bits, except in the case when all frames are analyzed. Even in this last case, the difference is still around 5 bits. This phenomenon is due to the fact that, since all messages have been included in the computation, long messages have increased the average even at a high number of frames.

3. A DISTINGUISHER ALGORITHM

Previous results allow us to formulate a general method to decide whether an mp3 file has or not hidden embedded information. The analysis will rely on an inspection of the first T frames. According to our previous results, it is expected that, if an mp3 file contains hidden data, there will be significant differences in the average of the reservoir length of the first T frames, for some value of T.

In practice, however, the most usual scenario is what is known as a blind steganalysis. This implies that we have at our disposal just one mp3 file. In this situation, our approach is to obtain a "clean" copy of the mp3 file, *i.e.* without any hidden information, if this was present in the file. This second mp3 file will allow us to perform the comparisons. There are various methods according to which hidden information can be removed. Our approach consists in codifying the mp3 file into a WAV file. During this process, the waveform is extracted, but any other information is removed. Then, the WAV file is again codified as an mp3 file, serving this copy for our analysis.

Fig. 3 sketches a distinguisher algorithm for the mp3stego tool. As exposed above, the parameter T indicates the number of frames to be considered in the analysis. From the experiments carried out, and taking into consideration the length of the messages embedded, it has been experimentally found that an optimal value is $T = 250$. On the other hand, the parameter P serves as threshold for the variable *VarAvR*, *i.e*. values above P indicate that the mp3 file contains hidden data. Experimentally, we have determined that an optimal value is a variation equal or higher to 20% ($P = 0.2$).

Input: *f*: mp3 file.
T: Number of frames to analyze.
P: Detection threshold.
Output: Yes, if *f* contains hidden information.
No, otherwise.
Algorithm:
1. Compute:
   AvR1 = average of the Av. Reservoir field considering the *T* first frames of *f*.
2. Obtain
   fw = *f* codified in WAV format.
3. Obtain
   f′ = *fw* codified in mp3 format, using mp3stego and the same codification parameters that *f*.
4. Compute:
   AvR2 = average of the Av. Reservoir field considering the *T* first frames of *f′*.
5. Obtain the variation of AvR1 with respect to AvR2:
   VarAvR = (AvR1 – AvR2)/AvR1
6. If (VarAvR > P) then
   f contains hidden information.
   Else
   f does not contain hidden information.

Fig. 3. A distinguisher algorithm for mp3stego.

In any case, it is important to keep in mind that values for *T* and *P* depend on factors such as the codification parameters used when generating the mp3 file or the intrinsic redundancy of the hidden data (*i.e.* compression ratio), among others. Therefore, any practical steganalysis following the proposed method will require a preliminary study such as the one briefly discussed in section 2 to set appropriate values for these parameters. Once obtained, the distinguisher algorithm can be employed.

3.1 Results

The distinguisher algorithm which is the basis of the steganalytic technique proposed in this work has been applied to a battery of mp3 files just described, hiding texts of different length. In Fig. 3 we can observe the variation of the *Av. Reservoir* value, as computed in step 5 of the algorithm.

It is easy to see how its value oscillates around 0% for files with no embedded information. On the other hand, information hiding is clearly visible because in most of the cases the variation is above 50%. Only when all frames are considered in the analysis, this variation decreases to around 20%.

The maximum value in the difference plot (65%) is reached when analyzing the first 250 frames, so a value of $T = 250$ is set. We also fixed a conservative threshold of $P = 0.2$, which corresponds to the minimum observed value.

3.1.1 Method validation

The proposed method has been validated with the following tests. We have generated a validation set made of:

- 100 mp3 files without hidden contents; and
- 100 mp3 files, different from those above, with hidden texts.

The mp3 files used for testing have been obtained from the Internet, using the eMule application and performing multiple pseudorandom searches for very general patterns such as '**a**', '**e**', '**i**', '**o**', '**u**', *etc.* We have begun with audio files in wav format. The 200 files thus obtained have been codified in the mp3 format by using the mp3stego application. In 100 of them, the hidden text was *texto08.txt*, of 119 bytes, which we consider to be the minimum needed for interchanging a significant message (that amounts to, approximately, two lines of text). The other 100 files were codified with no hidden information at all.

In the case of the audio files with hidden information, the proposed test had a 100% of success, correctly identifying the presence of embedded data. When working on files without hidden data, the results of the test were correct in 99% of the cases, with only one false positive. Additionally, in the correct detections we have observed an average variation of the *Av. Reservoir* value of a 69%. In the case of files without hidden data, this variation is of – 14%. This reinforces the correctness of choosing this parameter as a base for a distinguisher algorithm.

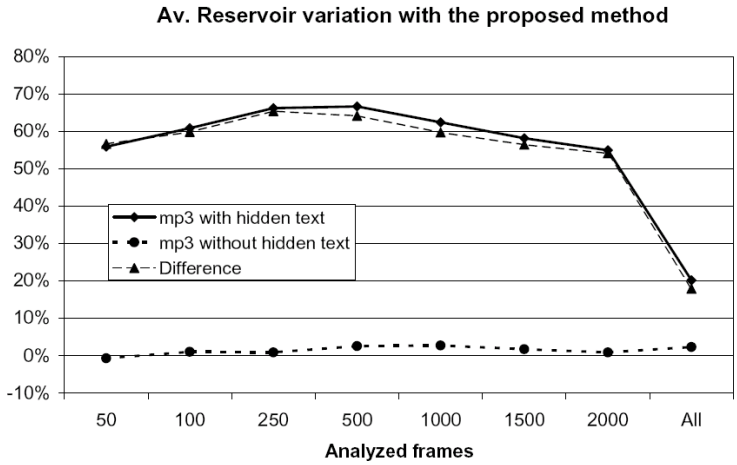
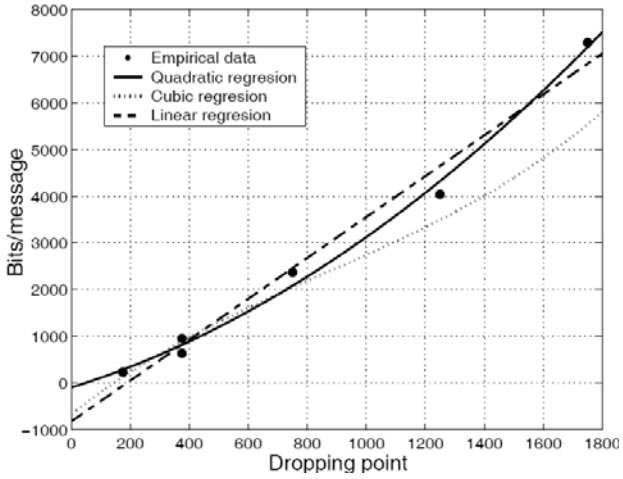


Fig. 4. Bit reservoir variation after applying the proposed method.

3.2 Estimation of the Hidden Data's Length

As implicitly shown in some of the previous figures and analysis, the amount of frames affected by the variation of the reservoir bit can be used to estimate the length of the hidden data.



Empirical data

Dropping point (DP)	175	375	375	750	1250	1750
Estimated bits/message (EBM)	232	632	952	2368	4040	7288

Regression

Linear: $EBM = -825.5 + 4.378 * DP$
Quadratic: $EBM = -97.7 + 1.961 * DP + 0.00126 * DP^2$
Cubic: $EBM = -674.3 + 4.991 * DP - 0.002575 * DP^2 + 0.000001 * DP^3$

Quadratic differences

Linear	736542
Quadratic	244429
Cubic	3567685

Fig. 5. Regression data to estimate the number of embedded bits.

The experiments showed that the relation between the number of embedded bits and the number of frames affected is not necessarily linear, because the hidden data is compressed before insertion. However, there is empirical evidence that a certain relationship exists between these two variables, such as more embedded data implies more affected frames. Intuitively, the amount of affected frames is computed by finding the *dropping point* (DP) of the difference curve (*i.e.* the value where the difference between the two curves decreases significantly).

In our experiments (see Fig. 5), the model that better fits these two quantities is:

$$EBM = -97.7 + 1.961 * DP + 0.00126 * DP^2 \quad (1)$$

where *EBM* is an estimation of the number of embedded bits in the message. Fig. 5 shows how this expression adjusts the experimental data found in our work.

4. CONCLUSIONS

After the analysis of the structural parameters of mp3 files, we have observed that the best characteristic for distinguishing an mp3 file with contents hidden with mp3stego is the average value of the reservoir bits in the first frames. In particular, files with embedded data present an average value of this parameter much higher than those without hidden information.

A possible reason for this is that the mp3stego tool, in order to hide the contents, increases the quanta size, thus using less bits for codifying every text. This forces that the affected frames need less space and, paradoxically even when including additional information, the size of the unused data in the final file (*Reservoir bits*) is bigger than in an mp3 without embedded contents.

Taking advantage of this, in the present work a complete analysis method is proposed that, in the light of the tests performed, has an overall success ratio of over 0.99. We have additionally proposed a new scheme for estimating the length of the embedded information. When adding these two results together, this constitutes quite a powerful steganalytic result over the mp3stego tool. These results the some improvements are needed in order to make mp3stego secure enough.

REFERENCES

1. D. A. Huffman, "A method for the construction of minimum redundancy codes," in *Proceedings of Institute of Radio Engineers*, Vol. 40, 1952, pp. 1098-1101.
2. S. Hacker, *MP3: The Definitive Guide*, O'Reilly Media, Inc., 2000.
3. B. Fries, M. Fries, and C. Finnm, *The Mp3 and Internet Audio Handbook*, Teamcom Books, Washington, 1999.
4. M. Robertson and R. Simpson, *The Official Mp3.Com Guide to Mp3*, MP3.com, Inc., San Diego, 1999.
5. F. A. P. Petitcolas and S. Katzenbeisser, *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House Publishers, London, 1999.
6. P. Wayner, *Disappearing Cryptography, Second Edition – Information Hiding*: *Steganography and Watermarking*, Morgan Kaufmann Publishers, San Francisco, 2002.
7. E. Cole, *Hiding in Plain Sight*: *Steganography and the Art of Covert Communication*, Wiley, New Jersey, 2003.
8. G. Kipper, *Investigator's Guide to Steganography*, Auerbach Publications, Kentucky, 2003.
9. C. S. Lu, *Multimedia Security*: *Steganography and Digital Watermarking Techniques for Protection of Intellectual Property*, Idea Group Inc., PA, 2005.
10. N. F. Johnson, Z. Duric, and S. Jajodia, *Information Hiding*: *Steganography and Watermarking – Attacks and Countermeasures*, Kluwer Academic Publisher, Heidelberg, 2001.
11. R. J. Anderson, *Security Engineering*: *A Guide to Building Dependable Distributed Systems*, Wiley, New Jersey, 2001.
12. F. Petitcolas, "mp3stego website," http://www.petitcolas.net/fabien/steganography/

mp3stego/index.html.
13. MPEG Audio Layer-3, Frauenhofer Institut Integrierte Schaltungen, 2001, http://www.iis.fhg.de/amm/techinf/layer3/index.html.
14. R. J. Anderson and F. A. P. Petitcolas, "On the limits of steganography," *IEEE Journal on Selected Areas in Communications*, Vol. 16, 1998, pp. 474-481.
15. J. Monk, "An exploration in the detection of hidden data in audio bit streams," University of Colorado at Colorado Springs Technical Report, http://cs.uccs.edu/cs525/projs2002/jmmonk/doc/jmmonk.ppt.
16. A. Leier, C. Richter, W. Banzhaf, and H. Rauhe, "Cryptography with DNA binary strands," *Biosystems Journal*, Vol. 57, 2000, pp. 13-22.
17. M. Chapman and G. Davida, "Hiding the hidden: A software system for concealing ciphertext as innocuous text," in *Proceedings of International Conference on Information and Communication Security*, Vol. 1334, 1997, pp. 335-345.
18. O. Takizawa, A. Yamamura, H. Nakagawa, *et al*., "A proposal of steganography on plain text and XML," *The 7th Annual Meeting of the Association for Natural Language Processing*, 2001, pp. 135-138.
19. The SNOW Home Page, http://www.darkside.com.au/snow/.
20. Spammimic Home Page, http://www.spammimic.com/index.shtml.
21. K. Ahsan and D. Kundur, "Practical data hiding in TCP/IP," in *Proceedings of the ACM Workshop on Multimedia and Security*, 2000, pp. 63-70.
22. R. El-Khalil and A. D. Keromytis, "Hydan: hiding information in program binaries," in *Proceedings of the 6th International Conference on Information and Communications Security*, Vol. 689, 2004, pp. 187-199.

Julio C. Hernandez-Castro is currently a Senior Lecturer at the School of Computing of the University of Portsmouth, UK. He has a B.S. in Mathematics, a M.S. in Coding Theory and Network Security, and a Ph.D. in Computer Science. His interests are mainly focused in cryptology, network security, steganography and evolutionary computation.

Juan E. Tapiador is Research Associate at the Department of Computer Science of the University of York, UK. He holds a M.S. in Computer Science from the University of Granada (2000), where he obtained the Best Student Academic Award, and a Ph.D. in Computer Science (2004) from the same university. His research interests are mainly on cryptography and computer security.

Esther Palomar is Assistant Professor at the Computer Science Department of Carlos III University of Madrid. She holds a Ph.D. (2008) and a M.S. in Computer Science (2004) from Carlos III University of Madrid. Before joining the University, she worked as forensic analyst for KPMG. She belongs to the Cryptography and Information Security Group, and her research interests are focused on security in peer-to-peer and ad hoc networks.

Angel Romero Gonzalez got his B.S. in Computer Science from Carlos III University in 2003, and a MBA at IDE-CESEM in 2004. He has worked for IBM and ISOFT and is with ENUSA from 1989. He is a member of ALI. His main interests are in software development and security.