# A Defenders' Guide to Steganography

Dr. Simon R Wiseman, *Deep Secure ltd*

Deep Secure Technical Report DS-2017-2

***Abstract*—Steganography concerns hiding a secret message inside another. There are many different methods for doing this and these have quite different characteristics. The methods can be classified according to how hard it is to detect the hidden message, how well the hidden messages survive routine manipulation of the carrier message and the nature of the encoding technique employed. Using this classification, methods and defences can be compared.**

***Index Terms*—Cyber Security, Steganography, Content Threat, Transformation.**

## I. Introduction

Broadly speaking, steganography is the ability to hide a secret message inside another message so that the presence of the hidden message cannot be detected by anyone other than the intended recipient [1].

Steganography differs from cryptography [2], which is about making sure a message cannot be read by anyone but the recipient. Encrypted communication can be seen and it is obvious that some message is being passed, even though the message cannot be read, whereas with steganography it isn't apparent that the hidden message is being sent.

It also differs from secret codes [3]. Here the reader can see the symbols that code the message, but does not understand their meaning or understands them to mean something else. With steganography, the symbols that encode the hidden message are not apparent.

Steganography is being used in various ways, but its adoption by cyber attackers to create stealthy attacks is a particular problem [4]. They are using it to evade defences and introduce malware into sensitive systems, to create command and control channels for directing attacks that hide inside existing business communication mechanisms, and to exfiltrate sensitive data from compromised systems without fear of triggering data loss prevention defences.

There are many ways of implementing steganography in the digital world, and different techniques have different properties. Unless these differences are understood, it is not possible to compare the effectiveness of defences that purport to reduce the risk. This article aims to help by explaining how different steganographic techniques can be categorised and showing what kinds of defence work against these categories.

Dr. Wiseman is Chief Technology Officer at Deep Secure, Malvern, WR14 1JJ, UK (simon.wiseman@deep-secure.com). First published on ResearchGate September 2017.

## II. Stealth

Steganography is about passing a carrier message that carries a hidden secret message, but how hidden it is makes a difference when it comes to a defence. A secret message that is hidden from a person inspecting the carrier message using normal desktop applications might be discoverable by someone using specialist tools. In other cases, the secret message might be so well hidden that it is impossible to find even with specialist tools, such as those from the Digital Data Embedding Laboratory in SUNY Binghampton [5].

There are two kinds of steganography that can be usefully distinguished, based on how stealthy the hidden message is. All kinds of steganography are invisible to the user who is employing the desktop applications normally used to handle the message, but we can distinguish between steganography that can be discovered by analysis with specialist tools and that which cannot.

**Discoverable Steganography**: A hidden secret message can be discovered by using specialist tools that look at the way the carrier message is constructed;

**Undetectable Steganography**: A hidden secret message cannot be distinguished from arbitrary noise, even by complex analysis.

## III. Survivability

When a message is manipulated, the changes might disrupt a hidden message it is carrying. If the structures used to carry the hidden message are altered, the hidden message may be destroyed. But if the ordinary applications used to manipulate messages preserve these structures, a hidden message is likely to survive and so be carried further through a system.

Steganographic methods can be classified as fragile if the hidden message is readily destroyed by the usual processing performed by normal applications. The most fragile methods are those where the message is destroyed simply by opening and saving the file carrying the message. Some methods may be less fragile, requiring some routine operation to be performed before saving the file, such as editing some text.

Robust methods are those that encode a message in such a way that hidden messages survive typical operations carried out

using normal operations. The strongest methods are those that survive any amount of editing a file.

**FRAGILE STEGANOGRAPHY**: The hidden secret message is destroyed when the carrier message is edited using normal desktop applications;

**ROBUST STEGANOGRAPHY**: The hidden secret message can only be destroyed by using specialist tools.

## IV. ENCODING TECHNIQUE

Different steganographic methods encode a hidden message in different ways, but broadly three classes can be identified.

The simplest case is where the hidden message is just appended to the carrier file [6]. Many applications ignore spurious data at the end of the file, so the presence of the hidden message will not be apparent to the ordinary user. The only indication that something is hidden that might be observed is that the file size has increased. But usually the user will not have the original document as a reference point and most formats produce files whose size has no direct relationship to what the user sees as their content, so the user is unlikely to notice a modest increase in size.

Although appending data is the simplest case, there are generally many ways that hidden data can be grafted onto a carrier message. Another fairly simple case is to add comment blocks or custom extensions to the carrier [7]. These are features often supported by file formats as an extensibility mechanism and are intended to be ignored by applications that do not understand them, making them an ideal vehicle for storing a hidden message.

Many file formats contain internal structures that are essential but convey no information. Most prevalent is the use of padding to align structures in memory, for example adding bytes to the end of records to align the start of the next record on a word boundary [8]. These padding bytes are an essential part of the structure, but their values do not contribute to the information content of the file. Such unused data is irrelevant and so can be used to store a hidden message. In these cases, the hidden data is living in a symbiotic relationship with the carrier.

A more complex method of storing a hidden message is to encode information in the order of some objects within it. As the number of orderable objects increases, more information can be stored on a near exponential scale. The hidden message is encoded by selecting one of the many permutations of the order of the objects. With n objects, there are n! permutations, which can encode log2(n!) bits – for example, with 10 objects 21 bits can be encoded, while with 256 objects this rises to 1683 (210 bytes).

**GRAFTED STEGANOGRAPHY**: The hidden secret message is stored in spurious data within the carrier message;

**SYMBIOTIC STEGANOGRAPHY**: The hidden message is encoded in data that is an unused part of the carrier message;

**PERMUTATION STEGANOGRAPHY**: The hidden secret message is encoded in the way data is ordered within the carrier message.

## V. EXAMPLES

### *STUFFING*

This technique works with many different file formats, including JPEG. It involves nothing more than appending the hidden message to a document file. This can simply be done using the command line copy command run in binary mode to concatenate the original image and the hidden message file.

In the case of image files like JPEG, viewer applications simply ignore any data following the image data, and the ordinary user will see nothing unusual – the increase in file size will not be noticed unless it is excessive. This makes the technique an example of Grafted Steganography.

The appended spurious data will also be ignored by an image editor, so opening and saving the image will remove the hidden message. Thus, the technique is Fragile Steganography.

Applications that create images will not add spurious data to the end of the file. Any JPEG with data appended can be considered unusual and possibly carrying a hidden message. There will be some false positives here, because small amounts of appended data can arise for innocent reasons – for example a carriage-return / line-feed might accidentally be appended by some application that processes the image. Nonetheless, stuffing should be considered a case of Discoverable Steganography.

### *PALETTE ORDERING*

This algorithm works with images that use a colour palette. The palette contains a small number of colours (up to 256) and each pixel is represented by an index into the colour palette. Each colour could take up 24 bits but each index is only 8 bits, so the image is well compressed though can only have a limited number of different colours. GIF images always use a colour palette while BMP and PNG image formats may be palettised.

The order of colours in the palette is important, because the pixels reference them by position, but the order does not contribute to the information content of the image. So the order can be changed, if the pixel indexes are updated to reflect the changes, without modifying the image. Palette Ordering is therefore an example of Permutation Steganography. A typical palette of 256 colours can encode 210 bytes, which is sufficient for a cyber attacker to encode a command or a stolen password [9].

Most image editing software will actually order the colour palette in some way, so encoding a hidden message in the ordering will produce images with unusual colour palettes. The colour palette is not normally seen by the user so it is unlikely that an unusual order will be noticed, which makes it Steganography. But analysis of the palette will reveal that it is non-standard and so the technique is Discoverable Steganography.

Opening and saving an image will usually preserve the ordering of colours in the palette, so this is an example of Robust Steganography. However, it should be noted that the hidden message does not of course survive conversion of the image to a TrueColour format, which does not use a colour palette, and some editors will do this conversion by default.

*LSB REPLACEMENT*

TrueColour images are generally able to store more colours than can be perceived by the human eye. Also, natural images have a lot of minor variation in the colours, for example a photograph of the sky will contain many shades of blue and white, but the exact shade of individual pixels is not important to the scene. This means the least significant bits (LSBs) of the pixel values are unused and so can carry a hidden message.

LSB Replacement (LSB-R) [10] works with TrueColour images that are not stored using lossy compression – so it works with PNG but not JPEG. It replaces the least significant bit of each pixel with one bit of the hidden message. The order in which pixels are selected is important. Different variations of LSB-R choose different orders, for example the first n pixels may store the n bits of the hidden message in turn, or some pseudo-random sequence determined by a secret password might be used. As the hidden message is stored in bits that are unused, LSB-R is an example of Symbiotic Steganography.

Regardless of the order, changing the least significant bit of a natural image is not usually noticeable even when the original and modified image are available for visual comparison. However, there is some correlation between the least significant bit and the other bits in each pixel value. Randomly changing the least significant bit to encode the hidden message upsets this correlation and this can be detected by a statistical test. Thus this form of steganography is Discoverable.

Opening and saving a TrueColour image preserves the pixel values (when there is no lossy compression), so the least significant bits are preserved and consequently the hidden message is preserved, making LSB-R Robust Steganography. However, the hidden message will be destroyed if the image is scaled, as the pixel values will be averaged out to fit onto the new canvas size, so it is not fully robust.

*LSB MATCHING*

LSB Matching (LSB-M) [11] is similar to LSB-R, except it uses a slightly different way of encoding the hidden message. The least significant bit of a pixel colour value still encodes one bit of the hidden message, but rather than just replacing the least significant bit with the required value, the original value is either incremented or decremented to make the least significant bit have the right value. The choice between incrementing and decrementing is made randomly.

With LSB-R, the more significant bits of the pixel values are unchanged, while with LSB-M they may be modified. So while LSB-R de-correlates the least significant bit and the more significant bits, LSB-M does not. This means the straightforward statistical tests for LSB-R do not detect LSB-M. More advanced tests have been proposed for the detection of LSB-M, but none appear sufficiently effective and accurate to be used in general. This makes LSB-M Undetectable Steganography that, like LSB-R, is Robust and Symbiotic.

*F5*

The F5 algorithm [12] works with JPEG files. JPEG files use a lossy compression method that converts the TrueColour representation of the image into sets of small integer coefficients of a two-dimensional wave equation that computes an approximation to the original colours. These coefficients are then compressed with a loss-less algorithm.

Some coefficients have more impact on the computed colours than others. The F5 algorithm encodes bits of the hidden message by reordering pairs of some of the least important coefficients – coefficient A > coefficient B encodes a zero, coefficient A < coefficient B encodes a one. Only relatively small amounts of information can be hidden in this way, but JPEGs tend to be large so it is still possible to use F5 to hide significant amounts of data.

In effect, the ordering of the coefficients is irrelevant to the final image, so this makes F5 an example of Symbiotic Steganography.

If any modification is made to a JPEG image, it is highly likely that the coefficients will be recomputed and any hidden message will be destroyed when the changes are saved. Even just opening and saving a JPEG can be a lossy process. However, an attacker can choose to hide messages only in JPEG images that will not be modified when they are opened and saved. Thus there is no guarantee that a hidden message will be destroyed, and consequently F5 should be considered Robust Steganography.

While the changes made by F5 make no discernible difference to the final image, even when the original is available for comparison. They also have no effect on the statistical

properties of the coefficients. This means it is not possible to easily analyse an image and decide if it contains a hidden message or not. This makes F5 Undetectable Steganography.

*SUMMARY*

The table below summarises the examples given.

| ALGORITHM | ENCODING TECHNIQUE | SURVIVABILITY | STEALTH |
|---|---|---|---|
| STUFFING | GRAFTED | FRAGILE | DISCOVERABLE |
| PALETTE ORDERING | PERMUTATION | ROBUST | DISCOVERABLE |
| LSB REPLACEMENT | SYMBIOTIC | ROBUST | DISCOVERABLE |
| LSB MATCHING | SYMBIOTIC | ROBUST | UNDETECTABLE |
| F5 | SYMBIOTIC | ROBUST | UNDETECTABLE |

## VI. COUNTERMEASURES

A hidden message that can be discovered can be stopped. But even with Discoverable Steganography it is not necessarily possible to make the detection reliable.

With Grafted Steganography, the spurious data can be identified and largely assumed to be a hidden message. However, some account needs to be taken of applications that do put spurious data in files to avoid false-positives (where a file is identified as carrying a hidden message when it does not). A defence can therefore be built, though a complete knowledge of the file format is needed to ensure there are no false-negatives (where a file containing a hidden message is not identified as such) because a message is hidden in spurious data that is overlooked. In complex file formats this is not simple task as spurious data can appear deep inside the file format's structures.

Symbiotic Steganography can in some cases be discovered. Where the unused data is normally set to a fixed value (typically zero), it will be obvious when a hidden message is stored in it. However, even when unused data is supposed to be a fixed value, applications will often leave it unset. This gives rise to possible false-positives. Discovery also requires complete knowledge of the file format, otherwise hidden messages can be stored in unused data that is overlooked, leading to false-negatives.

In some cases, though, it is normal for unused data to have an arbitrary value – for example the least significant bits of image pixel values. In these cases, it is not possible to discover a hidden message directly, though it may be possible to examine the statistical properties of the unused data to find anomalies. Statistical methods cannot guarantee to detect all instances of a hidden message. Also, they may indicate that some messages contain a hidden message when they do not, so in practice there are likely to be high rates of these false-negatives and false-positives

Some cases of Permutation Steganography can actually be discovered, because applications tend to use a particular order even though an arbitrary ordering is allowed. But, in general, it is not possible to say whether the order is normal or unusual. That is, most Permutation Steganography is Undetectable.

Only in limited cases can Discoverable Steganography be discovered with any degree of certainty, whereas by definition Undetectable Steganography cannot be discovered. Thus, the strategy of detecting messages that carry a hidden message in order to block their use in a cyber-attack does not work. Attackers generally find ways of evading attempts to discover their activities, and this will surely be the case with detecting steganography.

The only viable strategy for dealing with steganography is to ensure all the places a hidden message can be stored are either removed or overwritten. But there are two problems with a process of straightforward identification and modification of these places. The first is that it fails if the designer does not identify all the places where data can be hidden. The second is that the implementation must handle the complex file structures passed to it by an attacker, and this means the attacker is presented with a large attack surface.

The only effective way of destroying hidden messages is to take the information that is found in the carrier data and build entirely new normalised data to carry it forward. This is the strategy behind Content Threat Removal [ctr]. It transforms the way information is carried, converting potentially unsafe data to known clean safe data.

Such a transformation process comprises two separate parts – the extraction of the information and the building of new data. While the extraction part must handle the complex file structures presented to it by an attacker, the second part does not – it only handles the extracted information, and the interface can be designed to eliminate any complexity. With appropriate separation mechanisms ensuring the build phase cannot be bypassed, the effective attack surface is that of the build phase, which is small.

The process of extracting the information naturally discards any spurious data, so defeating Grafted Steganography. A process of normalisation is needed when building the new data to defeat Symbiotic and Permutation Steganography – this sets unused data to fixed values and sorts orderable objects in a fixed way. Since building new data requires a deep understanding of the

file format, it is straightforward to identify all the places where unused data occurs and where ordering can be exploited. So this makes it possible to be confident that no opportunities are left for messages to be hidden inside the file.

## VII. Future Work

Other than documenting how more steganography algorithms can be categorised, there is scope to refine and expand the classifications identified. In particular, the classifications are presented as dichotomies, but they are clearly ends of a spectrum and it may prove useful to define a scale for the spectrum. Also, the computing resources required and the detectability of those resources are worthy of further consideration.

## VIII. References


[1] Kahn D. (1996) The history of steganography. In: Anderson R. (eds) Information Hiding. IH 1996. Lecture Notes in Computer Science, vol 1174. Springer, Berlin, Heidelberg

[2] Bruce Schneier. 1995. *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (2nd ed.). Phil Sutherland (Ed.). John Wiley & Sons, Inc., New York, NY, USA.

[3] CHURCHHOUSE, R. F. (2002). *Codes and ciphers: Julius Caesar, the Enigma, and the internet*. Cambridge, Cambridge University Press. http://site.ebrary.com/id/10001842.

[4] Lauren J. Young, The Dark Side of Steganography, Aug 2015, https://spectrum.ieee.org/tech-talk/telecom/security/the-dark-side-of-steganography

[5] Digital Data Embedding Laboratory, Department of Electrical and Computer Engineering, SUNY Binghamton, http://dde.binghamton.edu/download/

[6] Zhe-Ming Lu and Shi-Ze Guo, Lossless Information Hiding in Images, Syngress, 2016, ISBN: 978-0-12-812006-4

[7] Parasites, Hacker Factor Blog, October 2014, https://www.hackerfactor.com/blog/index.php?/archives/642-Parasites.html

**[8]** Bitmap Image File (BMP), Version 5, https://www.loc.gov/preservation/digital/formats/fdd/fdd000189.shtml

[9] J.Fridrich, A new steganographic method for palette-based images, Procs IS&T PICS,1999

[10] Zhao Z., Liu F., Luo X., Xie X., Yu L. (2013) LSB Replacement Steganography Software Detection Based on Model Checking. In: Shi Y.Q., Kim HJ., Pérez-González F. (eds) The International Workshop on Digital Forensics and Watermarking 2012. Lecture Notes in Computer Science, vol 7809. Springer, Berlin, Heidelberg

[11] Jiaohua Qin, Xuyu Xiang and Meng Xian Wang, 2010. A Review on Detection of LSB Matching Steganography. *Information Technology Journal, 9: 1725-1738.*

[12] Westfeld A. (2001) F5—A Steganographic Algorithm. In: Moskowitz I.S. (eds) Information Hiding. IH 2001. Lecture Notes in Computer Science, vol 2137. Springer, Berlin, Heidelberg

[13] Martin Kuppinger, Executive View: Deep Secure Content Threat Removal Platform, https://www.kuppingercole.com/report/ev71311



**Dr. Simon R. Wiseman** received a BSc in Computer Science from the University of Reading in 1979 and a PhD from the University of Newcastle upon Tyne, UK, in 1988. From 1979 to 2010 he worked on cyber security at RSRE, DRA, DERA and QinetiQ in Malvern. In 2010 he joined Deep Secure in Malvern to continue working on cyber security products that protect against highly sophisticated content based attacks, developing the principle of Content Threat Removal into a commercial reality.