Steganography and Digital Watermarking: a global view.

Table of Contents

Table of Contents

Steganography and Digital Watermarking: a global view.

Next: introduction

Matteo Fortini

Steganography and Digital Watermarking: introduction

Steganography comes from the Greek and literally means **"covered writing"**.[JJ98-1] It is one of various **data hiding techniques**, which aims at transmitting a message on a channel where some other kind of information is already being transmitted. This distinguishes steganography from covert channel techniques, which instead of trying to transmit data between two entities that were unconnected before.

In this paper we'll consider two different applications of steganography:

- The first is what we call **classical steganography**, and its aim is that of transmitting a message by means of another innocent or casual-looking medium
- The second is **invisible digital watermarking**, which purpose is to assess the ownership or integrity of some pieces of information, named after the watermarks we are used to see on bills or legal documents for the same purpose. It can be used also to provide additional informations on a message inside the message itself.

A sample scenario for classical steganography can be the "Prisoners' Problem", formulated by Simmons in 1983, where Alice and Bob were in jail, and were planning an escape plan. Only, all of their communication was passed through the warden, Willie, and if Willie discovered an encrypted message, he would send them into solitary confinement. Thus, they had to exchange their messages in a covert way.[AP97]

The scenario for digital watermarking is much more from the real world: someone wants to be able to enforce his copyright over some informations or to check their integrity, and embeds some data in the informations themselves for this purpose.

In the copyright protection problem, we can distinguish between proper watermarking or fingerprinting. In the former the entity that embeds the data wants only to be able to show its ownership over it, so basically it inserts the same message in every copy of the informations. In fingerprinting the same entity wants to be able to tell who between the recipient of the data misused it, so it embeds personalized messages in every issued copy.[AP97]

Data hiding techniques can also be classified with respect to the extraction process:

- *cover escrow* methods need both the original piece of information and the encoded one in order to extract the embedded data.
- *blind or oblivious* schemes can recover the hidden message by means only of the encoded data.

Blind data hiding is usually preferred nowadays, since it is usually impractical to distribute certified copies of the original medium.[MBR98]

Both classical steganography and digital watermarking are based on a fundamental assumption: *it is quite easy to foil the human senses*. This can lead to a simple formula: given an original piece of data *d*, there is a threshold *t* below which any changes to the data won't be spotted by a person. *t* is depended on the experience of the observer, but there is a minimum that's beyond the capabilities of the human senses. Thus, we can always afford to make a change *c* on *d* without being spotted, as long as

$$d + i < t$$

[JJ98-2]

If the matter is, instead, to foil some technically gifted attacker, then we can rephrase the formula using the entropy function H(): assuming that the message we insert is perfectly encrypted, then it is indistinguishable from random data, so the entropy will be strictly additive. The entropy of the stego-medium S will be given by the sum of the entropy of the cover C and of the embedded message M:

$$H(S) = H(C) + H(M)$$

And the embedded message would be undetectable if:

- The entropy H(M) is much less than the uncertainty in the attacker's measurement of H()
- H(C) is reduced by some means before processing, so it is restored by adding H(M)

As a rule of thumb, we must give the attacker as little stego-data as we can, so he won't be able to gather any good measurement of entropy. [AP97]

What distinguishes classical steganography from invisible digital watermarking in the very end is that in the former what is important is the hidden message and not what a casual observer can see, while in watermarking we are adding content to some data which are important by themselves, for the purpose of completing them or protecting them.

This distinction also arises from a basic **tradeoff** that anyone who attempts to hide data in other data faces: the one between the ratio of hidden informations over the normal ones, and the subsequent robustness of the embedded message.

Roughly, this leads to a simple formula: given a piece of information in which we want to hide some data and being sure that the data is ideally undetectable, we have

$$amount\_of\_hidden\_data \times robustness = constant$$

Now, what we have supposed is that classical steganography is used as a service which is important for both the transmitter and the receiver. Hence we'll want to push up as much as we can the amount of data hidden, and eventually losing robustness.

On the other hand, for digital watermarking we have a completely different setting: the watermark is inserted by the transmitter, and it must avoid any receiver could be able to remove, counterfeit or destroy it by any means. This assumption demands that the watermark be as robust as it can. We can satisfy this need since a watermark is usually represented by a very small amount of data, not violating our equation.

Another topic that this paper will deal with is **steganalsys**, a series of techniques aimed at either discovering and extracting a hidden message or "destroying" it. Steganalisys is very similar to criptoanalisys for what regards the technical means it uses and the different kinds of attacks. [JJ98-2]

References:

- [JJ98-1]
- [Ben96]
- [JJ98-2]
- [AP97]

Previous:cover

Next: a brief history

Matteo Fortini

A brief history of steganography

The first description of the use of steganography dates back to the Greeks. Herodotus tells how a message was passed to the Greeks about Xerses' hostile intentions underneath the wax of a writing tablet, and describes a technique of dotting successive letters in a cover text with a secret ink, due to Aeneas the Tactician.

Pirate legends tell of the practice of tattooing a secret information, such as a map, on the head of someone, so that the hair would conceal it.

Kahn tells of a trick used in China of embedding a code ideogram at a prearranged position in a dispatch; a similar idea led to the grille system used in medieval Europe, where a wooden template would be placed over a seemingly innocuous text, highlighting an embedded secret message.

During WWII the grille method or some variants were used by spies. In the same period, the Germans developed microdot technology, which prints a clear, good quality photograph shrinking it to the size of a dot.

There are rumors that during the 1980's Margareth Thatcher, then Prime Minister in UK, became so irritated about press leaks of cabinet documents, that she had the word processors programmed to encode the identity of the writer in the word spacing, thus being able to trace the disloyal ministers.

During the "Cold War" period, US and USSR wanted to hide their sensors in the enemy's facilities. These devices had to send data to their nations, without being spotted.

Today, steganography is researched both for legal and illegal reasons.

Among the first ones there is war telecommunications, which use spread spectrum or meteor scatter radio in order to conceal both the message and its source.

In the industry market, with the advent of digital communications and storage, one of the most important issues is copyright enforcement, so digital watermarking techniques are being developed to restrict the use of copyrighted data.

Another important use is to embed data about medical images, so that there are no problems with matching patient's records and images.

Among illegal ones is the practice of hiding strongly-encrypted data to avoid controls by cryptography export laws.

References:

- [JJ98-1]
- [AP97]

Previous: introduction

Next: definitions

Matteo Fortini

Definitions

Steganography

We give some definitions common to the steganography field:

cover medium

This is the medium in which we want to hide data, it can be an innocent looking piece of information for steganography, or some important medium that must be protected for copyright or integrity reasons.

embedded message

This is the hidden message we want to put in the cover. It can be some data for steganography and some copyright informations or added content for digital watermarking.

stegokey

This is represented by some secret information, which is needed in order to extract the embedded message from the stego-medium

stego-medium

This is the final piece of information that the casual observer can see.

We can define this simple formula:

$$cover\_medium + embedded\_message = stego\_message$$

Steganalisys

As the name may suggest, steganalisys is the science of breaking steganography. Its attacks are of two different types: one is aimed at detecting and/or extracting an embedded message from a stego-medium. The second tries to destroy or render unrecoverable that embedded message. Note that these two tasks can be accomplished separately, since we can destroy a message just applying some transformations to the stego-medium depending on the steganographic method used or supposed, even if we don’t know what the message is.

The different steganalisys methods derive their names from their counterparts in cryptoanalisys:

- A **stego only** attack is one where we only have the stego-medium, and we want to detect and/or extract the embedded message.
- A **known cover** attack is used when we have both the stego-medium and the cover medium, so that we can make comparisons between the two
- A **known message** attack assumes that we know the message and the stego-medium, and we want to find the method used for embedding the message
- A **chosen stego** attack is used when we have both the stego-medium and the steganography tool or algorithm
- Lastly, a **chosen message** attack is one where the steganalyst generates a stego-medium from a message using some particular tool, looking for signatures that will enable him to detect other stego-media.

References:

[JJ98-2]

Previous: a brief history

Next: steganography

Matteo Fortini

Classical Steganography

As we said in the introduction, the sample problem of steganography is "The Prisoners Problem" by Simmons, 1983, where Alice and Bob are in jail, and are planning an escape plan.

All of their communications pass through the warden, Willie. Willie will attempt to find any hidden communication between Alice and Bob, and if he finds one, he will separate them and foil their plans.

Thus, the two prisoners must find the better way they can to exchange hidden undetectable data.

Moreover, Willie can be passive, so that he simply tries to detect the presence of the messages, or active, in which case he’s going to try to insert or modify the embedded messages for his purposes. [AP97]

Steganography has developed a lot in recent years, because digital techniques allow new ways of hiding informations inside other informations, and this can be valuable in a lot of situations.

The first to employ hidden communications techniques -with radio transmissions- were the armies, because of the strategic importance of secure communication and the need to conceal the source as much as possible.

Nowadays, new costraints in using strong encryption for messages are added by international laws, so if two peers want to use it, they can resort in hiding the communication into casual looking data. This problem has become more and more important just in these days, after the international Wassenaar agreement, with which around thirty of the major - with respect to technology - countries in the world decided to apply restrictions in cryptography export similar to the US’s ones.

Another application of steganography is the protection of sensitive data. A filesystem can be hidden in random looking files in a harddisk, needing a key to extract the original files. This can protect from physical attacks to people in order to get their passwords, because maybe the attacker can’t even know that some files are in that disk. [ANS98]

The major concern of steganography is stealth, because if an attacker, either passive or active, can detect the presence of the message, from that point he can try to extract it and, if encrypted, to decrypt it.

The resistance to attempt at destruction or noise is not required, since we consider the sender and the receiver equally interested in exchanging messages, so that they will try to transmit the stego-medium in the best way they can. If the stego-data can be transmitted over the selected channel, and this is usually the case with all the media that are used, like images or sounds, then the embedded data will be preserved along with them.

Thus, data hiding techniques for steganography must focus on the maximum strength against detection and extraction.

As a second request, we would prefer a high data rate, because we will usually want to be able to exchange any amount of data, from simple messages to top secret images. [JJ98-1]

Looking at the equation presented in the introduction about the basic steganography tradeoff, these two constraints will lead to losing in robustness against attacks aimed at destroying the message, but we have already said that we can accept it.

Steganography as an aid to cryptography

While cryptography today can provide good security to the exchange of informations, it is also true that in most occasions a really determined attacker can find ways to defeat even the most secure cipher, given enough resources (ultimately money) and/or time.

Steganography comes to aid in this problem since if a message can’t even be grabbed by the attacker, then the cipher can be considered really secure. The probability of finding what the message is is the same as finding it by random search.

In this way, we can say that steganography completes cryptography, and actually there are usually two ciphers to break when trying to extract the embedded message: one is the one with which the message was embedded, and the other is the one with which the message was enciphered.

The problem with this approach is how not to arise suspicion on the malicious observer. In fact, there must already be an "innocent" communication channel between the source and the destination, since a big exchange of naturalistic pictures between two governmental agencies, for example, would be spotted as strange by even the dumber of all the attackers.

We will talk about steganography applied to different media:

- Still images [FBS98] [WW98] [MBR98] [JJ98-1] [Ben96] [GB98]
- Moving images [WW98]
- File systems [ANS98]
- Audio files [Ben96]
- Text files [Ben96]

Our description will be more in depth for still images, being these pieces of data one of the most frequently exchanged on the Internet, and because the methods for information hiding in them are quite mature, given their interest for both steganography and digital watermarking.

Previous: definitions

Next: still images

Matteo Fortini

Data hiding in still images

We discuss the various techniques for data hiding in still images:

- LSB insertion [JJ98-1]
- spread spectrum [MBR98]
- texture block [Ben96]
- patchwork [Ben96] [GB98]
- Orthogonal projection coefficients manipulation [WW98] [FBS98]
- other methods: dithering manipulation, perceptual masking, DCT coefficients manipulation. [MBR98]

To remain invisible to the eye, these techniques must exploit "holes" in the Human Visual System (HVS):

- the masking effect of edges
- the varying sensitivity to contrast as a function of spatial frequency
- the low sensitivity to small changes in luminance for random patterns
- the low sensitivity to very low spatial frequencies, such as continuous changes in brightness through an image.

An advantage in using still images for data hiding is that they represent a noncausal medium, since it is possible to access any pixel of the image at random.

References:

- [FBS98]
- [WW98]
- [MBR98]
- [JJ98-1]
- [Ben96]
- [GB98]

Previous: steganography

Next: LSB insertion

Matteo Fortini

Least Significant Bits (LSB) insertion

Technique basics

Today, when converting an analog image to digital format, we usually choose between three different ways of representing colors:

- 24-bit color: every pixel can have one in 2^24 colors, and these are represented as different quantities of three basic colors: red (R), green (G), blue (B), given by 8 bits (256 values) each.
- 8-bit color: every pixel can have one in 256 (2^8) colors, chosen from a palette, or a table of colors.
- 8-bit gray-scale: every pixel can have one in 256 (2^8) shades of gray.

LSB insertion modifies the LSBs of each color in 24-bit images, or the LSBs of the 8-bit value for 8-bit images.

Example:

The letter ’A’ has an ASCII code of 65(decimal), which is 1000001 in binary.

It will need three consecutive pixels for a 24-bit image to store an ’A’:

Let’s say that the pixels before the insertion are:

10000000.10100100.10110101, 10110101.11110011.10110111, 11100111.10110011.00110011

Then their values after the insertion of an ’A’ will be:

*1000000**1**.10100100.1011010**0**, 1011010**0**.1111001**0**.1011011**0**, 1110011**0**.10110011.00110011*

(The values in **bold** are the ones that were modified by the transformation)

The same example for an 8-bit image would have needed 8 pixels:

10000000, 10100100, 10110101, 10110101, 11110011, 10110111, 11100111, 10110011

Then their values after the insertion of an ’A’ would have been:

*1000000**1**, 10100100, 10110100, 10110100, 11110010, 10110110, 11100110, 10110011*

(Again, the values in **bold** are the ones that were modified by the transformation)

From these examples we can infer that 1-LSB insertion usually has a 50% chance to change a LSB every 8 bits, thus adding very little noise to the original picture.

For 24-bit images the modification can be extended sometimes to the second or even the third LSBs without being visible. 8-bit images instead have a much more limited space where to choose colors, so it's usually possible to change only the LSBs without the modification being detectable.

Data Rate

The most basic of LSBs insertion for 24-bit pictures inserts 3 bits/pixel. Since every pixel is 24 bits, we can hide

3 hidden_bits/pixel / 24 data_bits/pixel = 1/8 hidden_bits/data_bits

So for this case we hide 1 bit of the embedded message for every 8 bits of the cover image.

If we pushed the insertion to include the second LSBs, the formula would change to:

6 hidden_bits/pixel / 24 data_bits/pixel = 2/8 hidden_bits/data_bits

And we would hide 2 bits of the embedded message for every 8 bits of the cover image. Adding a third-bit insertion, we would get:

9 hidden_bits/pixel / 24 data_bits/pixel = 3/8 hidden_bits/data_bits

Acquiring a data rate of 3 embedded bits every 8 bits of the image.

The data rate for insertion in 8-bit images is analogous to the 1 LSB insertion in 24-bit images, or 1 embedded bit every 8 cover bits.

We can see the problem in another light, and ask how many cover bytes are needed to send an embedded byte.

For 1-LSB insertion in 24-bit images or in 8-bit images this value would be 8/1*8 = 8 Bytes, for 2-LSBs insertion in 24-bit pictures it would be 8/2*8 = 4 Bytes, for 3-LSBs insertion it would be 8/3*8 = 21.33 Bytes.

Robustness

LSB insertion is very vulnerable to a lot of transformations, even the most harmless and usual ones.

Lossy compression, e.g. JPEG, is very likely to destroy it completely. The problem is that the "holes" in the Human Visual System that LSB insertion tries to exploit - little sensitivity to added noise - are the same that lossy compression algorithms rely on to be able to reduce the data rate of images.

Geometrical transformations, moving the pixels around and especially displacing them from the original grid, are likely to destroy the embedded message, and the only one that could allow recovery is a simple translation.

Any other kind of picture transformation, like blurring or other effects, usually will destroy the hidden data.

All in all, LSB insertion is a very little robust technique for data hiding.

Ease of detection/extraction

There is no theoretical outstanding mark of LSB insertion, if not a little increase of background noise.

It’s very easy, instead, to extract LSBs even with simple programs, and to check them later to find if they mean something or not.

Suitability for steganography or watermarking

First of all, since it is a so vulnerable technique even for simple processing, LSB insertion is almost useless for digital watermarking, where it must face malicious attempts at its destruction, plus normal transformations like compression/decompression or conversion to analog (printing or visualization)/conversion to digital (scanning).

Its comparatively high data rate can point it as a good technique for steganography, where robustness is not such an important constraint.

Problems and possible solutions

Having stated that LSB insertion is good for steganography, we can try to improve one of its major drawbacks: the ease of extraction. We don’t want that a malicious attacker be able to read everything we are sending.

This is usually accomplished with two complementary techniques:

- Encryption of the message, so that who extracts it must also decrypt it before it makes sense
- Randomizing the placement of the bits using a cryptographical random function (scattering), so that

it’s almost impossible to rebuild the message without knowing the seed for the random function.

In this way, the message is protected by two different keys, acquiring much more confidentiality than before.

This approach protects also the integrity of the message, being much more difficult (we could say at least computationally infeasible) to counterfeit the message.

Anyway, since we don’t want our message to be only an encrypted and scrambled message, we must go back to the purpose of making the communication hidden.

The two most important issues in this problems are:

- the choice of images
- the choice of the format (24-bit or 8-bit, compressed or not)

The cover image first of all must seem casual, so it must be chosen between a set of subjects that can have a reason to be exchanged between the source and the receiver.

Then it must have quite varying colors, it must be "noisy", so that the added noise is going to be covered by the already present one. Wide solid-color areas magnify very much any little amount of noise added to them.

Second, there is a problem with the file size, that involves the choice of the format. Unusually big files exchanged between two peers, in fact, are likely to arise suspicion.

Let’s calculate, for instance, what the size would be for a 500x300 image (150,000 pixels), quite common for pictures on the Internet, with the different color representations:

- 24-bit color: 150,000 pixels x 24 bits/pixel / 8 bits/byte = 90,000 Bytes ~= 440KB
- 8-bit color / grayscale (the occupancy is the same): 150,000 pixels x 8 bits/pixel / 8 bits/byte = 150,000 bytes ~= 146KB

Looking at the size, we can see that a 24-bit uncompressed picture is of a quite uncommon size, because it’s very strange that the sender didn’t compress it, a practice that’s widely used and wouldn’t have worsened the image quality so much.

To solve this problem, it has been studied a modification to the JPEG algorithm that inserts LSBs in some of the lossless stages or pilots the rounding of the coefficients of the DCT used to compress the image to encode the bits.

Since we need to have small image file sizes, we should resort in using 8-bit images if we want to communicate using LSB insertion, because their size is more likely to be considered as normal.

The problem with 256 colors images is that they make use of an indexed palette, and changing a LSB means that we switch a pixel from a position to an adjacent one. If there are adjacent contrasting colors in the palette, it can happen that a pixel in the image changes its color abruptly and the hidden message becomes visible.

To solve this problem different methods have been studied, like rearranging the palette so that adjacent colors

don’t contrast so much, or even reducing the palette to a smaller number of colors and replicating the same entry in the table in adjacent positions, so that the difference after the embedding of the message is not visible at all. Moreover for most images the reduction of colors from, for instance, 256 to 32 is hardly visible.

Most of the experts, anyway, advise to use 8-bit grayscale images, since their palette is much less varying than the color one, so LSB insertion is going to be very hard to detect by the human eye.

References:

[JJ98-1]

Previous: still images

Next: spread spectrum

Matteo Fortini

SSIS - Spread Spectrum Image Steganography

We point out this technique as an example for spread spectrum data-hiding methods. Spread spectrum techniques are now widely used in military radio communications, due to their very high robustness to detection and extraction.

SSIS is a quite mature process, and its aim is to achieve low detectability, ease of extraction, high data rate and good robustness to removal.

It is based on spread spectrum techniques, but it enhances them by adding other encoding steps, acquiring better performance.

Technique basics

The core of SSIS is a spread spectrum encoder. These devices work by modulating a narrow band signal over a carrier. The carrier's frequency is continually shifted using a pseudorandom noise generator feeded with a secret key.

In this way the spectral energy of the signal is spread over a wide band, thus decreasing its density, usually under the noise level.

To extract the embedded message, the receiver must use the same key and noise generator to tune on the right frequencies and demodulate the original signal.

A casual observer won't be able even to detect the hidden communication, since it is under the noise level.

the SSIS encoder adds more steps in order to push spread spectrum to its limits:

1. It optionally encrypts the message *m* to be embedded with *key1*, getting *e*
2. The data stream passes through a Low-Rate ECC (Error Correction Code) encoder, to acquire better robustness against destruction attacks and unwanted noise, becoming *c*.
3. Spread spectrum modulation, using a pseudorandom noise generator fed with *key2*, and get *s*
4. An interleaver and spatial spreader processes *s* using *key3* obtaining *i*
5. The output of the interleaver is added to the image *f*, getting *g*
6. A quantization process is used to preserve the initial dynamic range of the cover image. We'll call it still *g*

We assume that the stego-image is sent through a noisy channel to the receiver and will become *g'*

The decoding process fairly repeats the same steps backwards:

1. It gets an optimal approximation *f'* of the original image *f* using image restoration techniques
2. *f'* is subtracted from the stego image *g'* to reveal an estimate of the embedded data *i'*.
3. *i'* is fed into a keyed deinterleaver, that uses *key3* to construct an approximation of the hidden signal, *s'*.
4. *s'* is demodulated with *key2* to get an estimate of the encoded message, c'

5. c’ is decoded through the low-rate ECC to get e’
6. if m was encrypted, then e’ is decrypted with *key1* and this will give m’

Data Rate

The data rate for this technique can be fairly high, but it depends on the choices made for the different parameters of the encoding.

We can assume that the message will be compressed before embedding to allow for a higher capacity.

The ECC encoder instead is going to insert redundant data into the stream to be able to correct the errors. The more errors we want to correct, the more bits will be added. Then, we have a tradeoff between good retrieval and capacity. If we can allow for small glitches in the recovered message, then we can use a weaker encoding.

Moreover, the more data we want to insert in the image, the more noise we are going to add to it. Then, if our cover is not noisy, we will be able to hide very little data, while if we choose a noisy one, its capacity will be higher.

Experiments with 512x512 grey scale images (256 KB) could embed from 500 bytes to 5KB, depending on the cover. These experiments used a spread spectrum signal powerful enough to give almost total error-free retrieval, because the compression method adopted didn’t allow for any errors.

This means a data rate varying from 1 hidden_bytes/50 cover_bytes to 10 hidden_bytes/50 cover_bytes, a rate suprassed only by LSB insertion.

Robustness

Spread spectrum techniques are usually quite robust. Every transformation that adds noise to the image isn’t able to destroy the message. Anyway, a determined attacker can quite easily compromise the embedded data using some digital processing, like for example noise reduction filters, the same that are used in decoding to estimate the original cover.

Ease of detection/extraction

Spread spectrum encoding is widely used in military communications for its robustness against detection. An attacker can’t usually even know if the message was embedded, and anyway it will be very hard for him to extract it without knowing the right *key2* and *key3*.

Suitability for steganography or watermarking

Due to its fairly high capacity and low ease of detection and extraction, SISS is very good for steganography.

Problems and possible solutions

The basic tradeoff in using SSIS is between the error rate we can afford and the amount of informations we want to embed, that varies in turn the power of the added noise. The ECC is used to allow for a lower power without increasing the Bit Error Rate as well.

Further improvements will deal with improving the original cover estimate stage, so that it'll lead to a lower Bit Error Rate in the recovered signal, allowing to use less redundant ECCs.

Alexander Herrigel et alii have developed additions to spread spectrum techniques that can give higher robustness against cropping and geometrical modifications [HRP98].

The first is redundant encoding by dividing the cover into blocks, and embedding the same message in each of them, so that the hidden data can be extracted even from a part of the image as big as one block, but the more of it we have, the more certain we can be about the result.

Moreover, they added to the spectrum a template that can, through a log-polar transform applied to the spectrum of the stego-image, determine the original scale factor and orientation of the image, rendering the stego-message virtually immune to scaling and rotation. [HRP98].

Finally, spread spectrum techniques can add an adaptive perceptual masking filter before the insertion of the signal, so that the added noise is quite sure to be under perceptual limits. This, however, will increase the error rate in the retrieval, because it reduces the power of the embedded signal. [HP98].

References:

[MBR98]

Previous: LSB insertion

Next: texture block coding

Matteo Fortini

Texture Block Coding

Texture Block Coding was developed by Bender et alii and published on the IBM Systems Journal in 1996 [Ben96].

It is a very low bit rate technique, fairly resistant to affine manipulations of the image, but its major drawback is that it needs human intervention for the coding process.

Technique Basics

Texture block coding works by copying a region from a random texture pattern found in a picture to an area that has similar texture.

The coding process works by manually choosing the region on which to operate, and then using some mask to choose the area for copying, for example a graphic text, so that after decoding the mask can become visible.

The decoding process has this steps:

1. Autocorrelate the image with itself. this will produce peaks at every point in the autocorrelation where identical regions of the image overlap. If large enough areas are copied, this will produce an additional large autocorrelation peak at the correct alignment for decoding.
2. Shift the image as indicated by the peaks in Step 1.
   Now subtract the image from its shifted copy, padding the edges with zeros as needed.
3. Square the result and threshold it to recover only those values quite close to zero. The copied regions will be visible as theses values.

Data Rate

One can easily see that since the copied areas must be fairly large (at least 16x16 pixels), and they usually bear some mask on them, then the amount of hidden information is small. In fact, we can’t modify in this way many parts of an image without it becoming visible.

Moreover, not all parts of an image are suitable for this technique, like solid color areas, because they would magnify their changes.

Robustness

If the block size for the copying operation is large enough (16x16 pixel at least), then the blocks’ inner part changes in the same way as the image under most nongeometric transformations, such as filtering, compression and rotation. The embedded mask will still become visible, only it will be rotated or filtered with respect to the original.

Cropping will destroy texture block coding if one of the two copies is out of the cropped area. Affine transformations will usually transform the mask, too, maybe rendering it unintelligible.

If only one of the two equal regions is modified by a malicious attacker, then the embedded data can be destroyed.

Ease of detection/extraction

The decoding algorithm is quite easy to apply and doesn’t include any key-coded step, so the embedded data can be extracted by anyone.

Suitability for steganography or watermarking

Due to its very low data rate, its fair robustness to non malicious modifications and its ease of extraction make this technique almost only suitable for watermarking images.

Problems and possible solutions

The major drawback with this technique is the need for human support in choosing the areas from which to copy and paste. A solution would be to instruct a computer to do the same job, maybe with human monitoring.

Further studies include the possibility of cutting an pasting blocks from only part of the image frequency spectrum, moving around less noticeable blocks and rendering it more robust to various compression algorithms.

References

[Ben96]

Previous: spread spectrum

Next: patchwork

Matteo Fortini

Patchwork

Technique Basics

Patchwork is a data hiding technique developed by Bender et alii and published on IBM Systems Journal, 1996. It is based on a pseudorandom, statistical model.

It works by invisibly embedding a specific statistic, with a Gaussian distribution, into the host image. Two sets of pixel, or *patches*, of the image are chosen, the first A and the second B. Then the algorithm works by slightly brightening points in A, while darkening of the same factor those in B.

To determine the points we have to touch in the image, a pseudorandom number generator is used, feeded with a secret key shared by both the transmitter and the receiver, because the decoding algorithm has to visit the very same points in the same order to extract the embedded data.

What happens in brief, is that while the expected value of the sum

$$S(n) = sum\ (a[i] - b[i],\ i = 1..n)$$

where a[i] and b[i] are the i-th point in patch A and B respectively, is equal to zero in a normal image.

After doing n times the operations:

$$a[i] = a[i] + deltab[i] = b[i] - delta$$

The expected value is going to be shifted to the right of

$$2 \times delta \times n$$

with respect to the average one of 0.

So if n is big enough, and we calculate the value of S(n) for an image, we are going to have a quite strong certainty if the image was watermarked or not by patchwork.

Actually, we can assign a "certainty level" related to n to the coding process, representing the trust we can give to the decoding answers. [GB98] The problem with increasing n to get a stronger mark, is that the more we repeat the process, the more the watermark is likely to become visible.[Ben96]

Data Rate

We have seen that the only answer that this technique can give is if an image was encoded with a particular patch or not, given the stego-image and the key used for the pseudorandom number generator. From this

observation follows immediately that patchwork has an extremely low bit rate.

However, it must be said that usually multiple applications of this algorithm don’t interfere with each other, because patches made using different keys are almost orthogonal.[GB98]

Robustness

One of the most important characteristics of patchwork is its resistance to cropping and to gamma and tone scale corrections.

The former will lead only to a logarithmic reduction of accuracy as the picture size diminishes. The second usually change in a regular way the luminance of pixels, not disturbing patchworks, that uses a difference measure to work.

Patchwork is destroyed by any affine transformation, like tranlation, rotation or scaling.[Ben96]

Successful attacks against it were accomplished by Anderson and Petitcolas using a "synchronization breaking" attack. Basically, they slightly displaced the data in order to change them in an invisible way, meanwhile changing the positions of the points sampled by patchworks. [AP97]

Ease of detection/extraction

Patchwork is nearly invisible, since there’s the need for the key and the pseudorandom function in order to find the changed points. Anyway, if the same key is used to encrypt a large number of identical sized images, then by averaging them the patch will become visible. [Ben96]

Suitability for steganography or watermarking

Given its fair robustness and its extremely low data rate, patchwork is easily spotted as suitable for digital watermarking.

The usual procedure for applying it to copyright protection systems is to embed a first very strong bit (for instance with a 99.9999% certainty level), that works as a "marker", telling whether the image was watermarked or not at all. Then it uses other keys to embed the other bits, but in a much weaker way, so that the watermark doesn’t compromise the image so much.

The decoding algorithm will look first for the strong bit, and if it finds it, then it will go on decoding the rest of the watermark. [GB98]

Problems and possible solutions

The first problem is that, using points as the units of patches, the noise we add to the image is in the high frequencies. These are likely to be filtered or disturbed by a lot of processes, from a simple lossy compression

to DA/AD conversions. A solution is to use patches of small areas instead of points, with a distribution of the *deltas* that spreads the noise to a wide band or moves it to low frequencies, making it less likely to be filtered out.

Another issue is the big sensitivity to affine transformations, that can be reduced adding some heuristic based upon feature recognition, like aligning along the interocular line of a face, or using affine coding, or both. The last one uses a known pattern placed on the image so that it is possible to reconstruct the affine transformations applied to it, and invert them before decoding. [Ben96]

References:

[Ben96] [GB98]

Previous: texture block coding

Next: Orthogonal projection coefficients manipulation

Matteo Fortini

Orthogonal projection coefficients manipulation

This method for data hiding is a generalization of DCT coefficients manipulation [WW98].

Its value is especially for digital watermarking of images.

The basic approach is to manipulate the greater coefficients of a projection of the image over an orthonormal basis, so that the hidden information are stored in the most important part of the data and they are less likely to be destroyed by normal processing and lossy compression.

This technique is similar to some extent to LSB insertion in its philosophy, because it changes the greater coefficients in a relatively small way, but the target of the modification is less trivial.

Technique basics

The method we describe here briefly is an improvement over the classical DCT approach, and it conveys better security.

We said it is a generalization because the orthonormal basis used for projection is not made of cosine functions anymore. Its vectors are generated from a secret key and then the other steps are the same as the DCT ones.

An abstract for the encoding process can be:

1. Generate a set of *n* orthonormal vectors (two-dimensional maps) from *key*
2. Project the image over the basis obtaining a set of coefficients *{c[i], i = 1..n}*
3. Modify the biggest of the coefficients encoding the message bits *m(k)*with the wanted strength *a*: *c'[j] = c[j] (1 + a * m(k))*
4. Reconstruct the image inverting the projection

The decoding process will work backwards, obtaining the differencies between the coefficients calculated over the stego-image and the original ones. This points this method as a cover escrow one, since the original image is usually needed in order to extract the embedded message.

The key choice for this method is the generation algorithm for the orthonormal basis, which must be dependent enough on *key*. The authors used a generator of maps with a pseudorandom function, then they othonormalized the set using a Gramm-Schmidt algorithm. The interesting property of this technique is that it doesn't need to have a full basis for the space, only the more vectors in the basis, the safer the embedding will usually be.

Data Rate

The data rate for these techniques is very dependent on the nature of the original image and the basis used, because we need big coefficients to be able to hide data. Thus, an image can convey a different amount of

data using different *keys*.

Robustness

The method illustrated has proven to be resistant to a lot of attacks, including high lossy compression rates, filtering by photo manipulation packages and deliberate attempts at removal with dedicated software such as Stirmark and Unzign.

Resistance to cropping can be achieved using a redundant encoding, embedding the same message in multiple blocks of the image.

Ease of detection/extraction

Since the basis' vectors for this technique are always changing, then the extraction process is very hard without knowing the *key*.

Obiouvsly, an attacker can gain precious statistical informations when the same key is used to embed messages in different images, thus the need to change *key* frequently.

Suitability for steganography or watermarking

Due to its robustness and the need for the original medium in order to extract the embedded data, this technique is useful especially for digital watermarking.

However, DCT techniques, a subset of these algorithms, are used for steganography because of their relatively high data rate.

References

[WW98]

[FBS98]

Previous: patchwork

Next: other methods for still images

Matteo Fortini

Other methods for information hiding in still images

Dithering Manipulation

Dithering manipulation chooses the different patterns used to dither images to encode a message. It can achieve a high data rate, 1 hidden_byte/4 cover_bytes. Its major drawback is to rely only on dithered images and to be extremely sensitive to errors in the stego-image.

Perceptual Masking

This approach tries to embed data into regions that are masked by others with respect to the Human Visual System. The embedding can be done in the spatial or in the frequency space. The amount of data that can be hidden depends on the cover image and usually the choice of the areas to be modified has to be done under human control.

DCT coefficients manipulation

These techniques are used every time a lossy compression algorithm is on the signal path. Since those algorithms usually process the image using a DCT (Discrete Cosine Transform), which produces a matrix of floating point coefficients, while the same coefficients will be stored as ints, a rounding process is involved. The embedding of data is acquired piloting the rounding function.

These methods are sometimes considered as variants to the LSB method.

[WW98]

References:

[MBR98]

Previous: Orthogonal projection coefficients manipulation

Next: Moving images

Matteo Fortini

Data hiding in moving images

Andreas Westfeld and Gritta Wolf [WW98] have made an interesting proposal for data hiding in a videoconferencing system.

Basically, the technique used is DCT coefficient manipulation, which becomes very valuable in the context of videoconferencing.

To achieve high frame rates on narrow band digital networks, videoconferencing applications compress every frame with a differential lossy compression, which means that only differencies between successive stills are first compressed and then broadcast.

The advantages in manipulating DCT coefficients in this situation is that the overall effect is very similar to the modification due to slight movements of the camera. This renders the embedding technique almost invisible, and the authors claim that one can differentiate between the original and the stego images, but can't tell which is which.

Moreover, one of the threat of detection is comparison between successive similar frames that would magnify the presence of encoding. Since videoconferencing systems only broadcast the differences between successive frames, then this threat is automatically voided.

Other attacks are very unlikely to succeed in detecting or extracting the stego-message, since the added noise is very similar to the original one, and the data are encrypted before embedding, making them more difficult to find.

The data rate for the described technique has been reported as up to 8KB/s (a GSM conversation) embedded into an ISDN videoconference.

This points it out as a valuable classical steganography scheme, due to the high data rate and good stealth.

References:

[WW98]

Previous: other methods for still images

Next: other media

Matteo Fortini

Data hiding in other media

We describe briefly the data hiding techniques developed for other media than digital images.

The Steganographic File System

Ross Anderson et alii have developed a proposal for a steganographic file system. [ANS98]

Such a file system basically hides the user’s documents into other seemingly random files. To extract a document, the user must provide the system its name and a password. Otherwise, the presence of any data is undetectable.This technique relies on encryption schemes and pseudo-random number generators to achieve stealth.

A steganographic file system can protect from some threats:

- Soldiers and intelligence agents can be captured and tortured into revealing cryptographic keys and other secret data.
- When conducting delicate negotiations, such as between a company and a trade union, informal offers may be made which will be denied in the event of later litigation. However, the other side might obtain court orders for access to documents.
- Police power may be abused. An individual may be arrested on "suspicion" of a crime, found with an encrypted hard disk partition, and told that if he doesn’t supply the password, this will be taken as an evidence of guilt. But the encrypted files can well contain confidential business information sought by people who have bribed the police;
- Private individuals have been tortured by robbers into revealing informations such as the secret codes for their bank cards and the location of safes.

The steganographic file system would allow the owner in such situations to give the names and passwords for some of the files, while keeping secret the existence of the others. The enemy won’t be able to tell if other documents are present or not. For example, one could give access to his love letters and tax records, but keep quiet about the secret plans he is developing.

The authors didn’t resort in using classical data hiding in audio, video or still images because those techniques have a very low bit rate and are hard to implement in a transparent way for the OS. [ANS98]

Data hiding in audio

Bender et alii describe various techniques for data hiding in digital audio. [Ben96]

These techniques can’t rely on many of the methods described for images because of the better performances of the Human Auditory System (HAS) with respect to the Human Visual System (HVS).

The authors have sought coding schemes resistant to most of the usual manipulation of sounds, like A/D-D/A conversion, radio broadcast, "over the air" playing, resampling, lossy compression.

The first proposed method is the easy and high bit rate LSB insertion, that is easily destroyed by anything else than pure digital transmission.

The second technique is based on the HAS sensitivity only for differential phase variation, but relative insensitivity to initial phase. Thus, the sound file is divided into blocks and each block's initial phase is modified using the embedded message, preserving the subsequent relative phase shifts. This is one of the better techniques with respect to perceived signal to noise ratio.

Third, spread spectrum schemes can be used, even if they usually add perceivable noise to the sound. However, the embedded signal can be filtered through a perceptual mask, so that the most audible components of the added noise are reduced in power.

Last, the authors hid data by adding echo to the audio signal, using two different delays to encode bits. Both of these delays were chosen small enough to be heard by the naked ear as enrichments in sound, and not as distortions. This technique had a good robustness and fairly high data rate. [Ben96] Moreover, is the only one that can resist a jitter attack. [PAK98]

Data Hiding in text

Bender et alii showed some of the techniques used to hide data in text files. [Ben96]

These methods are divided in open-space, syntactic and semantic.

The formers use white space to encode hidden informations. They can modify the spaces between words, or the spaces at the end of sentences. The problem with these approaches is that most of the word processors clean up or reformat spaces in files transparently, so the message can easily be destroyed.

Syntactic methods are based on modifying the structure of text by changing punctuation or word order. They are quite difficult to implement in an automatic way without being visible. In fact mispunctuation is usually readily visible after a certain extent, and the order of the words in a sentence can change its tone in a strange way.

Semantic methods exploit equivalences between words (synonyms) to encode a hidden message, by choosing one word or the other in a synonyms pair. The problem here is one of style, since sometimes words can't be exchanged for "equivalent" ones without changing profoundly the meaning of a sentence.

A common limitation to all of these techniques for data hiding in texts is the very low data rate they can achieve. [Ben96]

References:

[ANS98]

[Ben96]

[PAK98]

Previous: Moving images

Next: Invisible Digital Watermarking

Matteo Fortini

Invisible Digital Watermarking

Invisible digital watermarking (IDW) is a type of steganography that aims at concealing information in a medium to prove ownership, integrity or provide additional information.

If it's used to support copyright, its first priority is robustness against destruction and spoofing.

We can define two different needs of copyright: proof of ownership and secure distribution.

In the former, the digital watermark will usually be unique, and it will try to convey an amount of information that suffices in binding the medium with the legal owner.

In the case of distribution, we refer to watermarking as fingerprinting, and the embedded data state a distribution contract between the owner and the receiver. In this way, the watermark will have a variable part to accomodate identity information about the receiver or the issuer of a licence of use, such as a retailer.

If watermarking is used to determine integrity of a piece of data, then its main purpose is to be very weak, so that any minor modification to the medium will destroy it. However, it will have to be highly concealed, too, since an attacker could try to rebuild it after tampering with the data.

These two applications are the ones that gave IDW its name, after the techniques used to protect bills or important documents from tampering or copying.

A less legal-bound application of watermarking is the embedding of additional information in an image. This doesn't need as much robustness as the former two, because the receiver of the medium will usually be interested in the enrichment of the informations. Among the different fields in which it can be useful are:

- Adding identity information in medical images, so that there is no need of external procedures to insure the right binding between identities and pictures
- Captioning of pictures, useful for professional photographers and news agencies
- Feature tagging in digital images, so that parts of them can be highlighted and described briefly.
- Indexing of stream data, as audio files, without the need of an additional communication channel.

Additional requests for watermarking algorithms can be ease of encoding or decoding. The former is needed if some retailer has to embed information about the customer on the fly while he is selling the medium. The last is useful if we want to be able to test automatically broadcast or Internet data to detect copyright infringements.

Watermarking techniques have developed enormously in the recent years, as one can see comparing the papers in the '96 and '98 Information Hiding Workshops. The majority of efforts, pushed by money from big firms, has been put on copyright support, since if practical solutions could be found to this problem, then the Internet and new digital channels could be used to distribute data in a quick and inexpensive way.

Another problem is the ease with which a normal user can make very good copies of money bills and other valuable papers using off-the-shelf quite cheap devices, such as scanners and color ink-jet printers. Countries have introduced features that can't be reproduced by these devices in new bills, but since they have to deal with older money, too, they are interested in a way to prevent or discover the authors of such crimes. [GB98]

Compared to classical steganography, digital watermarking adds the problem that usually copyrights last much more than the useful life for secret informations, typically 70 years, so that we can’t be satisfied with "computationally infeasible" proofs for resistance to attacks, because we can’t know what the capabilities of future attackers will be. [AP97]

We have already provided a description of the various methods for steganography in digital images and other media.

Among them, we can point out as most suitable for invisible digital watermarking:

- patchwork [Ben96] [GB98]
- spread spectrum (in the more robustness/less capacity flavors) [MBR98]
- orthogonal projection coefficients manipulation [WW98] [FBS98]

References:

[Ben96]

[GB98]

[MBR98]

[WW98]

[FBS98]

[AP97]

Previous: Steganography in other media

Next: Attacks

Matteo Fortini

Attacks on steganographic and watermarking systems

Johnson and Jajodia have attempted various attacks against steganographic software, both for detection and destruction of the embedded message.

Detection schemes can be known-cover and chosen-message attacks, where the added informations can be emphasized by subtraction, or we must analyze a number of images to get an "average" image with respect to some properties changed by steganography, and then proceed by subtraction again.

After this process the embedded message is seen as exaggerated noise compared to the normal or "average" image.

Tools that manipulate LSBs usually leave a signature in the palette, because of their need to change it to avoid the changed colors to become visible. Some programs reduce the palette to 32 colors, and each of them is replicated 8 times, giving a clear mark of their action.

One attack related to watermarking is a kind of spoofing, in which the malicious entity inserts a new watermark in the place of the old one. This is usually possible due to the orthogonality of watermarking techniques, which allow more than one embedded message to be included.

Attempts at destruction, instead, are possible due to the same assumption that we made in the introduction about the limits imposed to steganography. If an embedding algorithm can make an invisible change to a medium and conceal some data, then another algorithm can make another invisible change and usually destroy the first embedded message. [JJ98-2]

The collusion attack aims at fingerprints, where we have copies of the same medium that differ only in the watermark. The algorithm works by comparing the different media and modifying all the bits that are found to change, while keeping the constant ones. In moving images, the same attack would reconstruct the sequence taking frames from different copies, so that the fingerprint is destroyed. [LQR98]

When the watermarking uses a "black-box" device to block the use of unauthorized material, the attacker can use an adaptive attack to gain knowledge of the bits for which the algorithm is sensitive and maybe create a fraudulent mark. [LQR98]

Studies have proved that such attacks are far from being "computationally infeasible", since their complexity is O(N), with N being the number of bytes in the data. [LD98]

The jitter attack has been developed by Petitcolas, Anderson and Kuhn [PAK98] and has proved to work against all of the watermarking techniques, except echo insertion in audio data. It basically tries to break the synchronization needed by the decoding processes by replacing tiny parts of the data with other ones. With audio the system works by splitting the whole file in small chunks and deleting or adding one sample to each of them. With images, it takes one small row or column of pixels from a part of the picture, and places it somewhere else. The result is a file of the same size, but the watermark can’t be retrieved anymore.

The same two authors have written a sample program, too, called Stirmark, that can be used to test the robustness of digital watermarking techniques. Its algorithms apply a resampling to the images similar to the one that occurs after printing and scanning them with good devices. Moreover, they make small geometric

manipulations of the image, similar to those applied by morphing techniques, changing the grid of the picture. These changes, however invisible, usually destroy watermarks. Because they displace pixels and change their values.

Another technique that works against soft-bots that try to find copyrighted images on the web is the mosaic attack, and it exploits the habit of internet browsers to join together juxtaposed images. The original picture is broken in many smaller ones, and they are placed on the web page in the same order, so that the splitting is invisible, but the decoder has a much harder job to extract the watermark. [PAK98]

References:

[PAK98]

[LD98]

[LQR98]

[JJ98-2]

Previous: watermarking

Next: conclusion

Matteo Fortini

Conclusion

In this paper we tried to give an all-round view of steganography, both used to exchange messages and watermarking.

First we gave an outline of the problem, telling also some of the history of this quickly developing field.

Then we showed the different techniques invented, from the simplest to the more complex ones, trying to evaluate them under many points of view. Major emphasis was put on data hiding in images, for the techniques involved are usually more mature than the corresponding ones for other kinds of informations. Image encoding algorithms can also be representative for manipulation of other types of media.

Then we gave an outline of the problems involved with watermarking, a field that has come into light after the development of broad band worldwide digital networks.

In the end, we showed some of the attacks that can be attempted against steganography and watermarking, and their degree of success.

Steganography and digital watermarking are undergoing a development process similar to that of encryption, where there was a continuous invention of new techniques for encryption followed by successful breakings and new improvements of them. [PAK98]

Next: Bibliography

Matteo Fortini

Bibliography

[JJ98-1] Neil F. Johnson, Sushil Jajodia, George Mason University, *"Exploring Steganography: Seeing the Unseen"*, IEEE Computers, February 1998, pp. 26-34

[Ben96] W. Bender, D. Gruhl, N. Morimoto, A. Lu, *"Techniques for Data Hiding"* IBM Systems Journal, Vol. 35 Nos 31996

[JJ98-2] Neil F. Johnson, Sushil Jajodia, Center for Secure Information System, George Mason University, *"Steganalisys of Images Created Using Current Steganography Software"*, http://isse.gmu.edu/~csis

[AP97] Ross J. Anderson, Fabien A.P. Petitcolas, *"On the limits of steganography"*

[GB98] Daniel Gruhl and Walter Bender, *"Information Hiding to Foil the Casual Counterfeiter"*, 2nd Information Hiding Workshop, 1998

[MBR98] Lisa M. Marvel, Charles G. Boncelet, and Charles T. Retter, *"Reliable Blind Information Hiding for Images"*, 2nd Information Hiding Workshop, 1998

[ANS98] Ross Anderson, Roger Needham, Adi Shamir, *"The Steganographic File System"*, 2nd Information Hiding Workshop, 1998

[WW98] Andreas Westfeld and Gritta Wolf, Dresden University of Technology, Germany, *"Steganography in a Video Conferencing System"*, 2nd Information Hiding Workshop, 1998

[HRP98] Alexander Herringel, Joseph Ó Ruanaidh, Holger Petersen, Shelby Pereira, Thierry Pun, *"Secure Copyright Protection Techniques for Digital Images"*, 2nd Information Hiding Workshop, 1998

[FBS98] Jiri Fridrich, 2 Lt Arnold C. Baldoza and Richard J. Simard, *"Robust Digital Watermarking based on Key-dependent Basis Functions"*, 2nd Information Hiding Workshop, 1998

[HP98] Juan Ramón Hernández and Fernando Pérez-González, *"Throwing More Light on Image Watermarks"*, 2nd Information Hiding Workshop, 1998

[PAK98] Fabien A. P. Petitcolas, Ross J. Anderson, and Markus J. Kuhn, *"Attacks on copyright marking systems"*, 2nd Information Hiding Workshop, 1998

[LQR98] Jack Lacy, Schuyler R. Quackenbush, Amy Reibman, James H. Snyder, *"Intellectual property protection systems and Digital Watermarking"* , 2nd Information Hiding Workshop, 1998

[LD98] Jean-Paul M. G. Linnartz and Marten van Dijk, *"Analisys of the Sensitivity Attack against Electronic Watermarks in Images"*, 2nd Information Hiding Workshop, 1998

Matteo Fortini