Hiding Data in Image Choices with the Zebrafish Steganographic System

anonymous

No Institute Given

Abstract. [1] Many steganographic methods involve embedding hidden messages inside images. These changed images are potentially detectable by statistical analysis and the message is often easily removed by an adversary able to make small changes to the image. We introduce Zebrafish, a method by which two parties can communicate undetectably by sending chosen images to each other. The choice of images conveys the hidden information. Unlike previous methods, Zebrafish uses existing images unchanged, as opposed to embedding information in preselected images. Luminosity information in is used to choose images which will transmit the desired message. Each image's average luminosity encodes a few message bits, and the sequence of images encodes the sequence of message bits. Therefore the adversary cannot remove the message without degrading the quality of the image. Our proof of concept implementation creates web-based image galleries for the transmission of covert information.

1 Introduction

Steganography traditionally employs a *covertext*, an innocuous communications medium in which the secret message (*stegotext*) is sent. In an undetectable system a *covertext* containing *stegotext* is indistinguishable from a *covertext* containing no secrets. In addition, the *covertext* itself must not arouse suspicion. In the traditional approach, the sender modifies the covertext subtly in order to embed a secret message. For example, many naive schemes [Stea,Jst,Steb] replace the low order bits of an image with the encrypted bits of the message. These methods create images that the human eye cannot distinguish from the originals. More sophisticated methods [Pro01,Wes01] attempt to resist statistical tests which could be used to determine that these images have been modified to carry extra information. There is a danger, with these systems, that there might always be a more sophisticated test which could be applied to reveal that they are hiding information. Indeed, many systems have been shown ineffective by various statistical tests which have been developed in recent

[1] Based on thesis work.

years [WP99,FGD01a,PH02]. It is extremely difficult to determine if these systems are vulnerable to such failures as such statistical properties are subtle and varied and embedding methods are often complex. They also have the weakness that if the original image is directly compared with the modification, the differences will be obvious. Such systems are usually not designed to resist an active adversary who wishes to deny service to the system. To make their changes to the image subtle, the stegotext bits must be placed in areas which will not alter the appearance or statistical properties of the cover image. However, just as the embedder has subtly modified the original image bits to encode his image, so too can an adversary subtly modify the embedding image to destroy the embedded message.

We introduce Zebrafish as a response to these problems. Our goals in designing Zebrafish were to

- Hide information undetectably using images.
- Be robust against the adversary who modifies images in order to prevent use of the system.

There has been much work done in designing and building systems which attempt to meet the first goal. The primary contribution of Zebrafish is the emphasis placed on the second goal, in the belief that without robustness, undetectability has little meaning.

The first goal, undetectability, is central to the idea of steganography. Whereas previous systems pick an image as covertext and then modify it to embed information [Pro01,Wes01], we compile a database of suitable images and select those which correspond to the message we wish to send. Each image in the database encodes c bits of information which are strongly linked to the appearance of the image. We encrypt the message and then choose $\frac{n}{c}$ images whose sequenced encodings correspond to our ciphertext. We then group these images together in `html` image galleries. The receiver can visit the sender's website and download the images in order. He uses a public function to retrieve the bits of information stored in each image and then decrypts those bits with the secret key he shares with the sender. Since we never modify an image, our system does not change any statistical properties of the images we use. We do have to be careful that the image galleries we create are indistinguishable from those which can be found "in the wild" on the web. Since we use high-level information about the image and that information directly determines its appearance, the image cannot be changed by an adversary without changing the fundamental appearance of the image.

As discussed above, most embedding techniques are fragile and an active adversary could easily remove them without degrading the images noticeably. In order to subtly modify arbitrary images, these systems modify bits in the areas where they are most easily removed. An active adversary, one who can modify all data passing through it in order to promote censorship efforts, can easily defeat these systems without going through the effort of discovering their subtle statistical weaknesses. We would like our system to be robust against this censoring adversary's attempts to destroy our covert data by modifying the image.

We encode information based on images' appearances rather than their low-level representation. Assuming the websites generated by Zebrafish users are indistinguishable from those of usual `www` users, if an adversary wishes to prevent Zebrafish users from communicating, he will have to either deny all other Internet users in his power the ability to use images or alter the way the images used by these other users look in a noticeable way. This is because the information is encoded in high level properties of the image, so small modifications will not destroy it. Most conceivable adversaries lack the power and/or the desire to implement this level of control and censorship. A secondary benefit of this use of high level information which also sets it apart from other steganographic systems is its extreme generality. Zebrafish will work on virtually all image formats. Images can be resized, compressed, or have their low order bits randomized. You could use a variety of other steganographic systems on top of it and still have the images retain their message.

Zebrafish keeps the capacity of each image low, relying on large groups of images to convey meaning rather than any single image. In many systems, capacity is determined by the maximum number of bits that can be packed into an image without triggering whatever specific statistical attacks the author of the system is trying to avoid. In the case of Zebrafish, images need to be found to match the bits which are going to be sent in the Stegotext. The more bits each image must match, the more difficult the search for appropriate cover images will be. Another capacity constraint relates to robustness. The more bits we maintain per image, the less they can represent the core appearance of the image. If we use too many bits, the adversary will be able to make nonnoticeable changes which alter these bits. Lastly, by not attempting to get large capacities out of our images, our system remains simple and we have some hope of analyzing its security.

Steganography is most useful when the message would otherwise be censored. and when it is important that the adversary not know or suspect that the two parties are communicating in a way that they cannot detect. Both of the above situations exist in this world in the form of powerful, oppressive governments who wish to limit the dissenting elements of their populace by suppressing their views and ability to communicate. Such governments, and other similarly powerful entities, are exactly the adversary that systems like Zebrafish are intended to thwart. Such an adversary controls all the ISPs in the country and passes all data through its own proxy, allowing it to both gather and modify all data that passes through it.

Zebrafish will work well in this situation because it is unlikely that the governments would wish to eliminate images on the web in their countries. It is also unlikely that the governments would wish to make vast changes to the appearance of images on the web as citizens are likely to (1) place high personal value on the image quality and integrity and/or (2) have invested large amounts of time and money on graphic designers and market research to determine the appearance and placement of such images. Here, as a designer of steganographic system, we have some leeway. We can manipulate these images on our web pages because our priorities are such that we can create whatever covertext is necessary to suit our hidden message.

Zebrafish distinguishes itself by being a system that aims for robustness against an active adversary, rather than the passive adversary most systems assume. In addition, it avoids detection in a way that can be analyzed because it does not modify the images it sends. In the rest of this paper we will describe our adversary model (Section 2), present background information and related work (Section 3), describe the requirements of the system (Section 4), describe how the system works (Section 4), present analysis of the security of the system (Section 5) and then present conclusions and future directions (Section 6).

2 Adversary and Threat Model

Steganographic adversaries are often described in terms of the prisoners' problem, first formulated by Simmons[Sim84]. In this situation, Alice and Bob are attempting to plan an escape, but an adversarial Warden is watching their communication and will foil escape attempts she notices. Many systems discuss their security in terms of a *passive* Warden, who is able to watch messages but not alter them. Zebrafish's Warden is *active*: able to modify messages as they pass through. The Warden is *global*, able to watch all traffic at all times.

There are some important limitations on our adversary. First of all, we assume that the adversary does not have access to the storage on Alice and Bob's system. Secondly, we assume Alice and Bob have some shared secret with which to communicate. However, Zebrafish could be modified to allow key exchange using techniques described in previous work[AP98]. Lastly, the adversary will only make changes which do not alter the perceived value of the data sent. In the case of image traffic, we interpret this to mean that the data will not be changed sufficiently to affect human perception of the images. In figure 1, M must be of equivalent perceivable quality to M'.

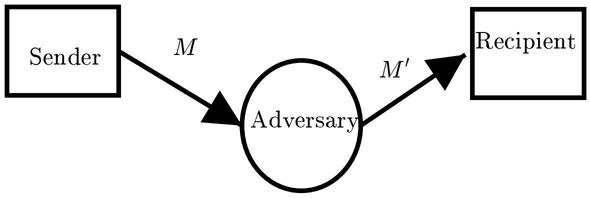


Fig. 1. The Adversarial Proxy

3 Related Work

There have been many steganographic systems designed and built. Almost all involve modifying an image to embed data in it. Many systems such as `steganos` [Stea], `Jsteg` [Jst], `steghide` [Steb] and many others rely on techniques which are known to be detectable, [WP99,FGD01a] such as encoding information in the low order bits of images. These systems rely on the adversary to not perform these statistical tests and as such rely on security through obscurity.

A few systems deserve some more attention. These systems take active measures to prevent statistical steganalysis. For instance, Outguess [Pro01] performs transforms on modified images so that they will retain their statistic properties. There are no statistical tests known which can detect Outguess. Another system which chooses bits in such a way that it avoids the statistical pitfalls that the simpler systems all victim to is F5 [Wes01]. These systems are potentially vulnerable to more sophisticated statistical tests that the adversary has thought of (but we haven't). All of these systems, both simple and more subtle, are intended to work against passive adversaries and make the assumption that the images arrive intact at their destination. Zebrafish does not make this assumption.

There has been quite a lot of work done on detecting systems which hide information inside images (by modifying these images). In 1998, Andreas Pfitzmann and Andreas Westfeld presented techniques for breaking the systems currently out there [WP99]. Recently, Niels Provos, acting on media speculation that terrorists were hiding information on images in eBay [Sie01], has built a framework for detecting steganographic images on the web on a large scale. [PH02] He has not had much success in finding "wild" steganographic images. Recently Jessica Fridrich [FGD01b,FGD01a] has built a system for detecting changes to JPEG images. This is particularly promising because it is not specific to any particular hiding system. Technology for detecting modifications to images is moving forward. However, none of these detection frameworks will detect covert content in our system since it does not modify images.

The problem of storing information robustly in images is closely related to the watermarking problem. Watermarking is traditionally used in order to prevent counterfeiting or protect intellectual property. This problem is subtly different from the steganography problem. In watermarking, the covertext is the valuable piece of data and the stegotext exists merely to protect it. In steganography, the message is the important part and the covertext is simply the packaging. The watermarking problem is generally believed to be very difficult, and some maintain that it is altogether impossible. We do not attempt to solve it in this paper. Instead, we take advantage of the difference in priorities between steganography and watermarking to make our problem tractable. Zebrafish achieves robustness by using high level information about the images to transmit the data. We can do this because we select images to suit our message. Using zebrafish to watermark images would be silly because the marker could not choose which images to mark.

A great number of systems which claim to be robust watermarking systems have been presented, [KP99], however, techniques for analyzing this robustness are varied and often suspect. The StirMark program [PAK98,A.P00] was designed to act as a benchmark in this case and we utilize it to argue the robustness of my system.

Zollner [J.Z98] and Cachin [Cac98] have presented work on an information theoretical framework for steganography. However, using their models requires a probability model of the cover media. This can be difficult to obtain in the case of the complex media often used in steganography [KP02]. Other theoretical ideas are discussed in "On the Limits of

Steganography." [AP98] This paper surveys work in steganography and discusses the interaction between stegography compression and entropy.

Scott Craver presented theoretical work on superliminal channels [Cra98] which could resist an active adversary. This work presented the inspiration for my work by suggesting that high level information in the cover medium could be used to construct a channel which could resist the active adversary.

4 Description of the System

This system consists of a server who sends images to the client machine (or machines). The images are chosen in order to convey a secret message m_s. The client wants to obtain the images and reconstruct m_s.

4.1 The Encoding Function

At the heart of the system is a function F which maps an image to c bits.

$$F(image) \rightarrow c \tag{1}$$

F maps images from the domain of images found on the world wide web to the codomain of integers in the range from 0 to 2^c. This function must have the following properties:

1. Changes to the image which do not change the perceived value of the image should not change $F(image)$. This will help achieve the goal of robustness against the active adversary.
2. In order to achieve undetectability, outputs of F concatenated together must be indistinguishable from the encryption of a message. Since such encryptions have high entropy, these outputs must be equally distributed among the codomain of the function.

In order to achieve the first property, we used the average luminosity of the image as a basis for the encoding function. To determine this, we convert the image to `ppm` format, obtain the 8 bit `rgb` values of each pixel, add them up and divide by the total number of pixels multiplied by three. This gives us a real number between 0 and 256 bits. We now need to map these numbers into integers in the range $\{0, 2^c\}$. We divide the range of the real numbers $\{0, 256\}$ into 2^c intervals in that range. In order to achieve the second property, we divide sets of images drawn from the distribution of images on the web equally into each interval. In using the luminosity of the whole image, we limit our capacity c. We cannot use too many bits or small changes in the image might change the output of F and we would lose robustness. We chose $c = 4$. The mapping of average luminosity interval to 4 bit number is shown in the table below.

Range	0-46	46-62	62-73	73-83	83-92	92-100	100-108	108-115
Value	0	1	2	3	4	5	6	7
Range	115-123	124-132	132-142	142-154	154-169	169-188	188-214	214-255
Value	8	9	10	11	12	13	14	15

In order to determine the intervals used, we needed to understand the distribution of images on the web. To do this, we collected images by doing successive altavista image searches of words in `/usr/dict/words`. For each image, we determined the average luminosity. After a while we achieved a distribution which did not change significantly when more images were added to it. Most images appeared in the middle of the spectrum, with a bias toward bright values. We believe this is due to the preponderance of white backgrounded images on the web. The graph in figure 2 shows the distribution as measured using 24,211 images. The x axis is average luminosity values up to 256. The y axis represents the number of images with average luminosity equal to $\lfloor x \rfloor$.

Zebrafish can use images in any format, as long as luminosity information can be extracted. In our implementation, we convert images into `ppm` format to easily extract the luminosity information. Images can be culled from almost any source, including easily recognizable images on the web since the images are not modified.

We considered extending the capacity of Zebrafish by considering the average luminosity of regions of the image. However, in real world images there simply was not enough variance in the different regions of a single image. As a result these images would be hard to find. If users managed to find such varied objects they would cause the system to be detectable by their very strangeness.

4.2 Server

In order to send images the sender must have either a shared key with the clients or the public key of intended recipients. In addition, the server needs access to large amounts of images. Lastly, the server will need a mechanism to get these images to the client in a way that will not provoke

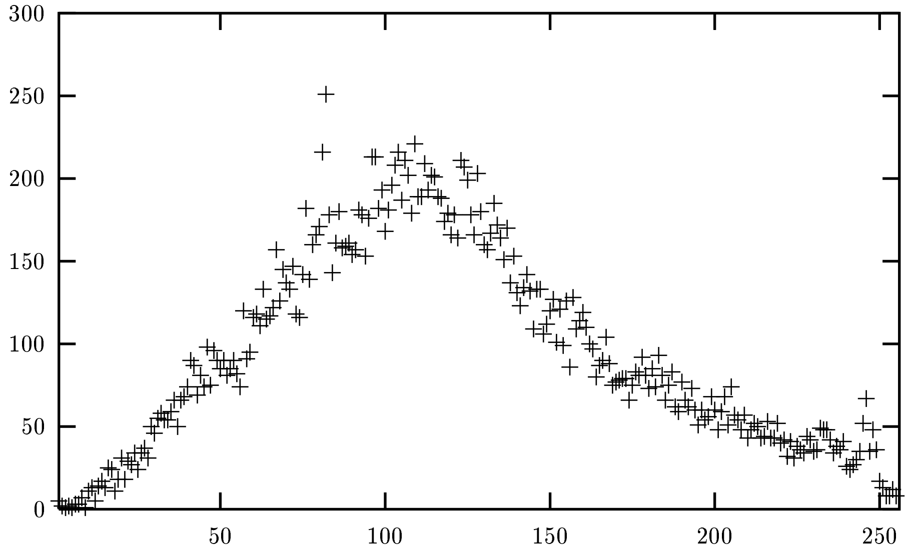


Fig. 2. Distribution of average luminosity of images on the web

suspicion. One way of doing so is to serve the images on a webpage. This is the mechanism we will be assuming is in use for most of this paper.

The server will need access to a large number of images in order to construct messages. In order to send an n bit message, the server needs $\frac{2^c n}{c}$ images from which to choose, where c is the number of bits stored in each image (in our implementation $c = 4$). This is because each of the $\frac{n}{c}$ images sent could require any of the 2^c encodings so we need this many images to ensure we can send any n bit message. This may seem like a large number of images but disk space is cheap these days and we can store the images in an easily searchable manner. The server can create a database of images as follows:

- Gather lots of images which are plausible[2]
- Apply the encoding function to the images to obtain a c-bit representation of the image ($c = 4$ in this case)
- Store the images in an easily searchable manner.

Once our server has a source of messages, it can now send messages. The steps involved in this process are as follows:

- The server decides to send an n bit message M.

[2] the concept of plausibility in the context of this system is further described in section 5.2

- The server encrypts M. This can be done under the symmetric key shared with the recipient or under the recipient's public key.
- The server groups bits of the ciphertext $E(M)$ into c-bit blocks, each of which can be represented by a single image.
- The server finds images in the database which correspond to each block of bits in the previous step.
- These images must be made accessible to the client in the order of the message. One way to do this is to serve them on a webpage.

4.3 Recipient

Our recipient basically needs two things: access to the images and a key with which to decrypt the message. The first requirement might be satisfied with a web browser, knowledge of the server's location, and access to the server's webpage (the webpage is not blocked by the adversary's proxy). The second by a shared key with the server or a public key known to the server.

In order to receive messages, the client machine:

- Visits the Server's webpage, downloading the images (or obtains them through some other manner) as well as the `html` page which indicates the order of the images.
- Uses the encoding function to retrieve c bits from each image.
- Concatenates the c-bit blocks into a ciphertext string.
- Decrypts the ciphertext string.

5 Evaluation

Evaluation is one of the most difficult tasks in steganographic system design. A steganographic system should be evaluated based on how well it provides certain desired properties in the face of a threat model. The threat model for this system is described in section 2. These properties are:

- Confidentiality - The adversary should not be able to gain any information about the covert data being sent in the system.
- Undetectability - The adversary should not be able to distinguish cover media created by the system from similar cover media created by legitimate users which does not hide information.
- Robustness - The adversary should not be able to prevent messages from getting to their destination.

5.1 Confidentiality

In general the confidentiality property is a fail-safe one. If the adversary is somehow able to suspect that a certain message has hidden data, he should not be able to determine what that hidden message is.

Zebrafish achieves confidentiality by encrypting the message M before dividing it into c bit blocks and choosing images to send. The security depends on the strength of the encryption system and the key. To avoid detectable repeat messages, the encryption system should be randomized.

5.2 Undetectability

The undetectability constraint is much more difficult. In general there are two types of attacks on such systems. Attacks which are done automatically by machines and attacks which are performed by humans looking for strange content. Generally, the attacks which are performed by machines are based on statistical analysis of the cover media. Images which are used in steganographic systems tend to have higher entropy than those which do not. [Pro01] Often, modifications to the images can alter the expected output of a compression algorithm such as the JPEG format and give the game away.[FGD01b]. Our images should be absolutely resistant to this kind of scrutiny since we have not modified them at all.

However, we do have to worry that the grouping of images might be statistically unusual. We would like for the "ciphertexts" obtained from the high level information in image galleries on the web to be indistinguishable from those generated by our algorithm. In other words, we want the output of our encoding function F, when applied to web image galleries in the general case to have high entropy.

This should be the case because of the way the indexing function was chosen. The function was chosen by looking at the distribution of average luminosities of images on the web (D_{WEB}).

More specifically, the encoding function maps an 8 bit value specifying the average luminosity of the image to a 4 bit string. This output should be uniformly distributed on the codomain of F. An image drawn at random from images on the web will map to one of the 16 values. Without knowledge of which image was chosen, this output should be indistinguishable from a random number.

$$D_{WEB}(8) \rightarrow D_{RAND}(4) \tag{2}$$

Since the ciphertexts created by the encryption system have high entropy, each four bits of the ciphertext will also be equally likely to be any of the 16 values as well. Therefore, the encoding of a set of images found on the web should be indistinguishable from a ciphertext generated under the encryption system.

This shows that the encoding of the images is indistinguishable from the encoding of the message. Ideally, we would like to show that the images encoding messages are indistinguishable from images on the web. We have thus far only shown that they are indistinguishable with regard to average luminosity. In order to show this we would need to thoroughly characterize the distribution of groupings of images on the internet.

The problem that we face is that any time we attempt to hide information within complex, nonrandom covermedia, we are going to change the distribution of that media to suit our purposes. This is because we are making a choice which is determined by none other than our hidden plaintext message. Such choices are not generally made by the designers of the content in which we hide our messages. Take for example, a cane with a secret compartment. The makers of the cane recognized a redundancy, as all designers of steganographic systems do. The cane does not have to be solid inside to serve its purpose. If we hollow out parts of the cane, it will look identical to any other cane. There are plenty of people in our society who use canes which do not have secret compartments so the presence of a cane, by itself, should not cause suspicion. We can assume that we have some "key" without which the cane will not open and reveal its secrets. By some measures, our job as a designer of a steganographic system might be done. However, when we removed part of the inside of the cane and replaced it with something else, we may have changed the weight of the cane, we may have also changed its center of gravity and a number of other things which may not be immediately obvious. If the cane is to be used for a specific purpose we may be able to compensate for this, by carefully conforming to every possible measure that might distinguish our cane from a legitimate cane to an observer. We may still miss some. Our job becomes more difficult if we wish to make our cane a general tool that can be used to carry a wide variety of secret items. Now we have a choice: we must either publish to our cane carrier the statistics of the normal cane and instruct that he is only to include things in the cane which conform to the specifications given or we can assume that our adversary will not examine the cane sufficiently closely to distinguish the subtle differences between our cane and the expected cane. Neither solution is terribly satisfactory.

We nonetheless attempt to solve this problem. Images are created in general by cameras, editing programs and compression algorithms. Their statistical properties are extremely complex and we feel that it would be difficult to modify images undetectably. Groups of images, in contrast are created by simpler algorithms (such as those run by search engines) and human beings (such as those on personal web pages). Knowing this, we can design our system such that the web pages produced in our system will produce ciphertext strings which are indistiguishable from web pages found "in the wild" of the world wide web. However, we leave the secondary characteristics unspecified, so that they can be decided by humans in the ways that normal web pages are determined, by human selection and human designed algorithms.

As a result, a user of the system will have to put effort (relative to his level of paranoia) into creating a web page which could plausibly be a "normal" webpage with no hidden content.

In order to do this effectively the user will want to have lots of images, any grouping of which is a plausible part of their webpage. Practically speaking, they will probably want to create their own images which they can then use to create a plausible pretext for having a web page. An example of this is a large set of "vacation photos" among which some subset are chosen to exist on the website. Another is "My Favorite Animal Pics of the Week" which can select among any animal pictures we can find, or "New and Adorable Clip Art." One of our favorite examples might be a series of webcams looking at various things in our living space and then have a place on our webpage which "randomly" selects between the different cams each of which takes pictures in a place with different luminosities. While manufacturing these pretexts might be hard for many end users, the basic idea of "create a set of images which would make sense together" should be fairly simple. If these images are chosen in the same manner that a user might choose them for a "normal" website (just out of a larger subset) then we ought to avoid both statistical and human attacks. On the surface, this might look a bit like security through obscurity. However, apart from the luminosity values (which we have encoded to be indistinguishable from normal web images) we are designing our webpages just as any other webdesigner might: by hand. If we design the webpage (in general terms) before we choose the message, we ought to end up with a webpage which is indistinguishable from any other of the myriad innocuous webpages which clutter up the world wide web.

This notion of plausibility is inextricably linked to steganography. If Alice and Bob could just send random bits around then they do not need steganography; they can simply use cryptography. However, in many situations, this is not the case. The distribution which Alice and Bob are covertly sending around must be the same as the one the adversary expects to see from normal users.

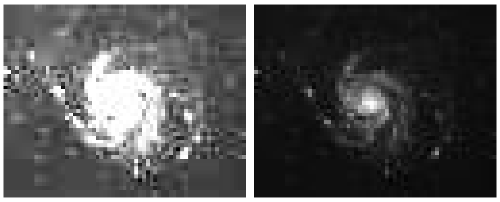
5.3 Robustness

The zebrafish system encoding function is very simple and general. As long as the average brightness of the image can be determined, a user can obtain the required bits from it. These bits remain constant even when the image is recompressed or converted to a different format. This gives the user a great deal of flexibility in using the system.

However, what if our adversary is modifying images as they pass through a proxy? Is our system robust against such an attack? Clearly if the adversary is willing to block image content altogether or to modify images beyond all recognition he will be able to block the system. However, our adversary is unwilling to lower the perceived value of the images.

This is a very similar to the problem of robustness of watermarking schemes. A popular benchmark for robustness is the StirMark program[PAK98,A.P00]. The StirMark program simulates printing out an image on a high quality printer and then rescanning it. Our system performs quite well against StirMark (although not perfectly). The mark was changed on only 9 of 194 images tested.

There is a worry that the adversary might attempt to destroy the mark by actually changing the luminosity of the images outright. The reason this is not a viable attack is that it distorts the image as perceived by the user. In order to determine that this was the case we took a sample set of images and flipped a coin to determine whether they would be brightened or dimmed. Then the luminosity of the image was modified the minimum amount necessary to change the encoding function. When the modified images were placed next to the unmodified images an observer could tell which image had been modified (on a dirty, low quality laptop lcd screen) with about 80% probability. While in some cases detection was difficult, in many instances it was very easy as shown in figure 5.3.



Some images will have luminosities that are very close to the borderline between intervals. Since an adversary knows are algorithm, he could modify these images safely. As a result, it is a good idea to put some redundance in the underlying plaintext message and use a cipher which is able to handle this (eg counter mode) [**?**]. It is important to realize that we *must* use these borderline images or their absence will make our system detectable.

In almost all instances, differences between the two images were easily perceptible to the naked eye. As a result, it is unlikely that an adversary would thus distort all images going through their proxy. If for some reason the adversary was willing to distort images to this degree the bandwidth could always be reduced to the level such that modification of the image would cause massive distortion.

Another possibility is that an adversary might reorder the images. If the images are not of uniform size, doing this to the webpage might seriously adversely affect the "look and feel" of the site. As a result, this attack is also unlikely. One way to subvert this attack is to embed order information in the images.

6 Conclusion and Future Directions

Zebrafish is a system which provides unobservable communication at a low data rate. The system is robust against an active adversary. A proof of concept implementation of the system exists in the form of a handful of perl scripts which make databases of images, encrypt plaintext and choose images.

In general, steganographic systems which embed data in complex cover media are vulnerable to detection. One way to deal with this limitation is to attempt to have most of the statistical characteristics of your distribution determined as they would be in a normal system and take heed to make the things that have to be determined by the system correlate to the statistics of the cover distribution.

Some future directions might be:

- Further analyse the probability distribution of images on the Internet in order to conform to them more absolutely
- Find a means to get more bandwidth out of each image, possibly by using high level information not related to average luminosity.
- Work on public key and broadcast publishing in steganography
- Streamline the user interface for the implementation.

References

[AP98] R. Anderson and F. A. Petitcolas. On the limits of steganography. *IEEE Journal on Selected Areas in Communications*, 16:474–481, 1998.

[A.P00] F. A.P.Petitcolas. Watermarking schemes evaluation. *I.E.E.E. Signal Processing*, 17(5):58–64, 2000.

[Cac98] C. Cachin. An information-theoretic model for steganography. In D. Aucsmith, editor, *Information Hiding, 2nd International Workshop, volume 1525 of Lecture Notes in Computer Science*, pages 306–318. Springer, 1998. Revised version, March 2001, available as Cryptology ePrint Archive, Report 2000/028, http://eprint.iacr.org/.

[Cra98] S. Craver. On public-key steganography in the presence of an active warden. In D. Aucsmith, editor, *Information Hiding, 2nd International Workshop, volume 1525 of Lecture Notes in Computer Science*, pages 355–368. Springer, 1998.

[FGD01a] J. Fridrich, M. Goljan, and R. Du. Reliable detection of lsb steganography in color and grayscale images. In *ACM Workshop on Multimedia and Security*, pages 27–30. 2001.

[FGD01b] J. Fridrich, M. Goljan, and R. Du. Steganalysis based on jpeg compatibility. In *SPIE Multimedia Systems and Applications IV*. 2001.

[Jst] Jsteg. http://linkbeat.com/lb/files.

[J.Z98] e. a. J.Zollner. Modeling the security of steganographic systems. In D. Aucsmith, editor, *Information Hiding, 2nd International Workshop, volume 1525 of Lecture Notes in Computer Science*, pages 344–354. Springer, 1998.

[KP99] M. Kutter and F. Petitcolas. A fair benchmark for image watermarking systems. In *SPIE: Security and Watermarking of Multimedia Content, vol. 3657*, pages 219–239. 1999.

[KP02] S. Katzenbeisser and F. A. Petitcolas. Defining security in steganographic systems. In *SPIE vol. 4675, Security and Watermarking of Multimedia*. 2002. To appear.

[PAK98] F. A. Petitcolas, R. J. Anderson, and M. G. Kuhn. Attacks on copyright marking systems. In D. Aucsmith, editor, *Information Hiding, 2nd International Workshop, volume 1525 of Lecture Notes in Computer Science*, pages 219–239. Springer-Verlag, 1998.

[PH02] N. Provos and P. Honeyman. Detecting steganographic content on the internet. In *ISOC NDSS'02*. 2002.

[Pro01] N. Provos. Defending against statistical steganalysis. In *10th Usenix Security Symposium*. 2001.

[Sie01] D. Sieberg. Bin laden exploits technology to suit his needs. *CNN.com*, 2001. http://www.cnn.com/2001/US/09/20/inv.terrorist.search/.

[Sim84] G. Simmons. The prisoners' problem and the subliminal channel. In *CRYPTO '83*, pages 51–67. Plenum Press, 1984.

[Stea] Steganos 3 security suite. http://www.steganos.com.

[Steb] Steghide 0.3, release 1. http://steghide.sourceforge.net.

[Wes01] A. Westfeld. High capacity despite better steganalysis: F5- a steganographic algorithm. In *Fouth Information Hiding Workshop*, pages 301–315. 2001.

[WP99] A. Westfeld and A. Pfitzmann. Attacks on steganographic systems. In *Information Hiding, Third International Workshop, IH'99*, pages 61–76. 1999.