Zebrafish: A Steganographic System

by

Rachel Greenstadt

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2002

Author . . . .
Department of Electrical Engineering and Computer Science
May 24, 2002

Certified by . . . .
Robert Morris
Assistant Professor
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Zebrafish: A Steganographic System

by

Rachel Greenstadt

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 2002, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Computer Science and Engineering

Abstract

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. We use luminosity information 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 deleting, reordering, or blatantly altering images. Our proof of concept implementation creates web-based image galleries for the transmission of covert information.

Thesis Supervisor: Robert Morris
Title: Assistant Professor

Acknowledgments

Thanks to Roger Dingledine, David Molnar, Geoffrey Goodell, H.T. Kung, Nick Mathewson and, of course, Robert Morris for helpful comments and support.

Contents

List of Figures

List of Tables

0.1 Introduction

0.1.1 What is Steganography?

Let us assume that Alice wants to send a message to Bob. Alice naturally wants to keep the contents of the message secret (a problem which can be solved by traditional cryptographic means). However, in this instance, Alice also wants to hide the existence of the message, or the fact that she and Bob are communicating secretly, from prying eyes. Alice can no longer find a simple solution to her problem in the field of cryptography, she must resort to information hiding.

Information hiding is a field which works to hide the presence of information from the adversary, in addition to its nature. It has many applications, including enhancing privacy, overcoming censorship, digital rights management, anonymous communication and espionage. Due to increasing interest from content holders in means to protect their intellectual property and concerns about privacy in situations when cryptography is prohibited or limited, the field of information hiding has grown much in recent years. Andreas Pfitzmann demonstrated a steganographic system to the German government, which succeeded in convincing them that limiting cryptographic exports was a futile proposition[1]. Information hiding encompasses many subfields besides steganography including anonymous communications, covert channels, detection of hidden information, digital elections, subliminal channels, and watermarking. Figure 0.1.1 shows a hierarchical classification of the field of information hiding [PAK99].

Steganography is a subfield of information hiding. The word "steganography" comes from Greek words meaning covered writing. 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 cover-

[1] PET 2002, rump session

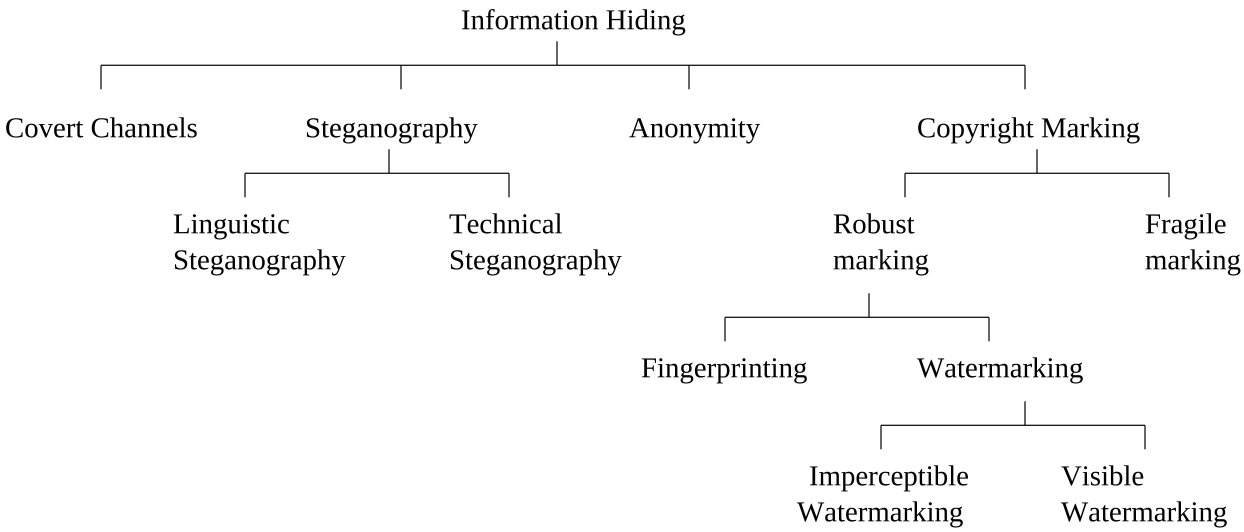

Figure 0-1: A classification of information hiding. This classification defines 'technical steganography' as steganography in which the messages are physically hid. Most digital steganography falls into the category of 'linguistic steganography.'

text subtly in order to embed a secret message. In general, the message is encrypted before embedding because otherwise the natural language structure of the message would make it detectable.

Other systems hide information in images as a means of solving 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. The difference is one of priorities. 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. 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 for the robustness of steganographic and watermarking systems.

0.1.2 General Properties of Steganography

A secure steganographic system should provide the following properties:

- 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.

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.

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.

0.1.3 Attacks Against Steganographic Systems

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.

There have been several potential classes of attacks which have been described which an adversary can employ against a steganographic system. The following human and statistical attacks were described in the third Information Hiding Workshop by Westfeld and Pfitzman[WP99].

- Human_attacks are attacks in which the presence of hidden information is detectable by the human senses somehow. This includes images that look wrong (for instance, if there is variation in what ought to be a monochrome field) or audio files with noticeable artifacts. These changes might be able to be sensed by using sophisticated visual/audio aids like a microscope. An example of a visual attack which works on all of the lsb type images is to strip all but the least significant bits from the image and look at it. The characteristics of the overall image should still be recognized in the lsbs. If not, chances are they have been altered as the lsbs of an image are not random.

- Statistical attacks are the kind most likely to be used by the kind of adversary we are concerned about. The adversary may not be able to carefully scrutinize every piece of data which it collects. However, she will have a large amount of innocent covertext to gather statistics about and to utilize to discover anomalies. These tests will be able to be performed automatically by a computer. One such test is the Chi-squared test which will catch simple schemes. Other tests might use expected compression artifacts to determine if changes have been made.

- Non-steganalytical Attacks are the main type of other attack which steganographic programs are subject to is a denial of service attack. In this type of attack, the image is changed such that the secret information is destroyed. In order to use this kind of attack without being able to use steganalysis, this would have to be done in a manner that would not destroy or appreciably degrade the covertext. This sort of attack is closely related to research done on the persistence of watermarks. Typical attacks involve using various compression techniques on the data. Stirmark[PAK98, A.P00] is a program which has been used to destroy the marks in many watermarking schemes which made claims of robustness.

One approach to understanding the security of steganographic systems is to use game theory. Information hiding can be seen as a two player contest between a data-hider and a data-attacker. A game described by Ettinger [Ett98] analyzes the robustness (or resistance to the third of the listed attacks) of a steganographic system. It is likely that detection could also be modelled this way. Usually detection is modelled by comparing the output of the system to unmodified cover media. In unsophisticated systems, they often only consider human attacks and consider media which appear similar to the naked eye (or ear, etc) to be sufficiently secure for their purposes. Other systems claim that the statistical distributions of the outputs of their system are indistinguishable from unmodified cover media.

0.1.4 History of Steganography

Historically, steganography has been a form of security through obscurity. The method in which the message is hidden is known only to sender and receiver and this is where the security of the system lies. This is in violation of Kirchoff's principle, which states that the security of a system should lie in its key alone. Nonetheless, for many centuries steganographic systems have been designed and use which depend on the secrecy of the algorithm.

Several historical examples of this sort of system are presented in David Kahn's "Codebreakers,"[Kah67] a work which chronicles codes, ciphers, and other information hiding techniques throughout history. One of the classic examples of early steganography was the use of invisible inks. These consist of organic fluids such as milk, vinegar and urine and chemical inks. Many of the organic types can be exposed by simple heating. Invisible inks have fallen out of favor since the development of universal developers which expose the inks. A popular use of steganography in World

War II was the microdot, a small photograph reduced to the size of a period on a page and overlaid over such punctuation marks. One of the most famous instances of the early usage of steganography comes from the histories of Herodotus. A revolt against Persia was set in motion by Histiaeus, who wished to urge his son-in-law, Aristagoras at Miletus, to revolt. He shaved the skull of a slave and tattooed his message thereon. He waited for the hair to regrow then sent the slave to Aristagoras with instructions to shave the slave's head. In addition to these examples, hiding information on one's person (by swallowing it, perhaps) or in secret compartments within objects are also forms of steganography.

Recently the field of steganography has focused on its use in digital media. Messages have been hidden in images, audio files, video conferencing sessions, and even text. Many of the initial system were based upon hiding information in the least significant bits of images, however, recent research [WP99, FGD01a] has shown that these systems are insecure. Security in steganographic systems has traditionally meant undetectability. An adversary should be unable to know, or even have grounds to suspect, that a secret message is embedded in a cover medium. This property has been very hard to evaluate in steganographic systems, for reasons which will be elaborated upon in Section 0.4. Most existing steganographic systems are *fragile*, meaning that the secret embedded message is easy to remove. In World War II, the censors would make small changes to telegrams and letters in order to prevent coded messages. In one instance, a message stating "Father is dead" was altered to say "Father is deceased." The suspicious reply was "Is Father dead or deceased?" This example illustrates the important relationship between robustness and undetectability. If a message does not get through because it is removed by a censor, the information it contains will still need to be conveyed somehow. The way in which this occurs, if the sender and recipient are not very careful, may alert the adversary to the presence of secret content.

0.1.5 Recent Image-Based Systems

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 encode information in the low order bits of images. This method create images that the human eye cannot distinguish from the originals. However, this technique is known to be detectable by statistical tests, [WP99, FGD01a] 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] uses various transforms to correct for changes in entropy created by the introduction of the hidden data. Currently, 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 fall victim to is F5 [Wes01].

0.1.6 Attacks on Recent Systems

There is a danger, with existing 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 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.

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.

Existing systems 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.

0.1.7 Zebrafish Overview

In response to the issues discussed in 0.1.6, we introduce Zebrafish. 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. These bits are obtained from the average luminosity of the image. As a result, the encoded bits are strongly linked to the appearance of the image. We encrypt the message and then choose $\frac{n}{c}$, where n is the length of the message, 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.

As discussed in Section 0.1.6, 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 steganographic 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 censor, 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.

Scott Craver presented theoretical work on superliminal channels [Cra98] which could resist an active adversary. A supraliminal channel is a communications channel that has the property that it is impossible to modify the secret message embedded within it without significant changes to the cover object. This work presented inspiration for Zebrafish by suggesting that information which was necessarily linked to the cover medium could be used to construct a channel which could resist the active adversary.

In the case of Zebrafish, this necessary information is the overall appearance of the image rather than inconsequential details of their 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 overall properties of the images, 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 and still retain their message. The sender may wish to do this for practical reasons such creating an esthetically pleasing and plausible webpage which does not use too much bandwidth or the adversary may do this in the hopes of destroying the message. 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.

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 0.2), describe the requirements of the system (Section 0.3), describe how the system works (Section 0.3), present analysis of the security of the system (Section 0.4) and then present conclusions and future directions (Section 0.6).

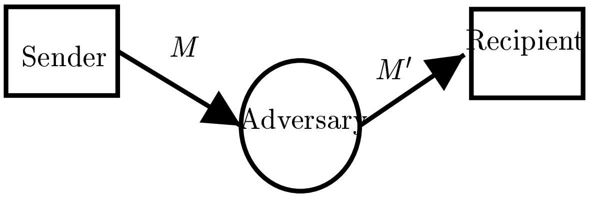

Figure 0-2: The Adversarial Proxy

0.2 Adversary and Threat Model

In terms of the prisoners' problem described in Section 0.1.3, 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, because the adversary must make these changes without knowledge of whether the sender of the data is using steganography, we assume 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 0-2, M must be of equivalent perceivable quality to M'.

0.3 Description of the System

Zebrafish consists of a server that sends images to the recipient machine (or machines). The images are chosen in order to convey a secret message m_s. The recipient wants to obtain the images and reconstruct m_s.

0.3.1 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 (in many cases this will be any change to the appearance of the image) 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 must have the same statistical properties regardless of whether the sequence of input images is the output of Zebrafish or an innocent image gallery. As a result, if the input to F is drawn from the distribution of images from the web (D_{WEB}), the output of F must be indistinguishable from the encryption of a message sent by Zebrafish. We assume the output of our encryption function is random, all possible sequences of c bits are equally likely. As a result, on web image inputs, all outputs of F must be equally likely.

In order to achieve the first property, we used the average luminosity (or brightness, measured in 8 bits) of the image as a basis for the encoding function. To determine this, we obtain the 8 bit `rgb` values of each pixel, add them up and divide

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

Table 1: The Encoding Function F: 16 intervals from 0 to 255 and the 4-bit number each maps to

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 all outputs of F must be equally likely. We determine the distribution of images on the web, and divide it up into our 2^c intervals such that equal numbers of images fall into each interval. An image drawn at random from images on the web will map to one of the 2^c values. Without knowledge of which image was chosen, this output should be indistinguishable from a random number.

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

Since the ciphertexts created by the encryption system are randomly distributed, each c bit string of the ciphertext will be equally likely to be any of the possible values. Therefore, the encoding of a set of images found on the web should be indistinguishable from a ciphertext generated under the encryption system.

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 Table 1.

In order to map luminosity information to 4 bits and 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.

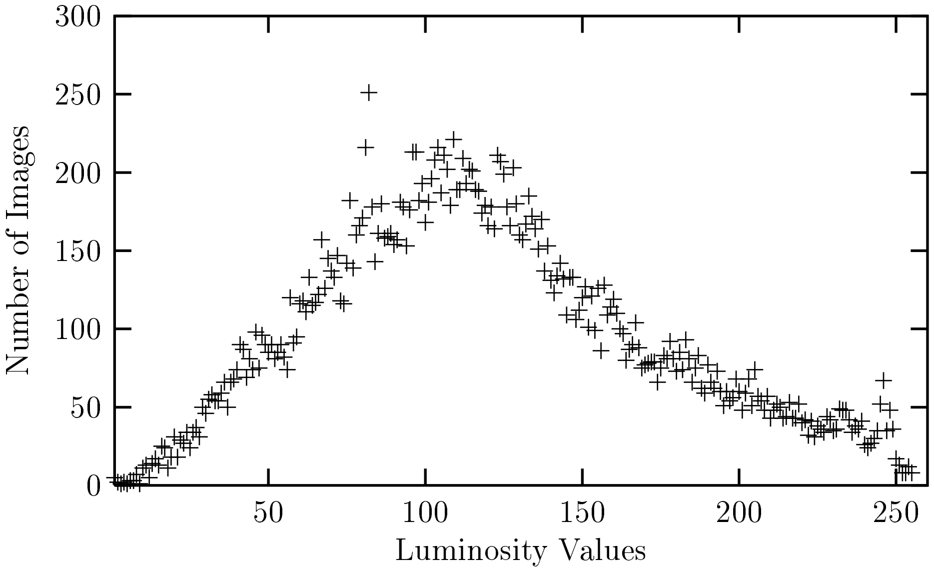

Figure 0-3: Distribution of average luminosity of images on the web

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 0-3 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 the floor of average luminosity equal to x.

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.

0.3.2 Choosing Image Galleries

The encoding function will ensure that our web pages are indistinguishable from normal web pages with respect to the average luminosities of the images they contain. However, we require that all statistical properties of Zebrafish web pages conform to the distribution of normal web pages.

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 web pages just as any other webdesigner might: by hand. If we design the web page (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 web pages 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.

0.3.3 Server

In order to send images the sender must have either a shared key with the recipients 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 recipient in a way that will not provoke 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 arbitrary n bit message without reusing images, 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. M includes redundance in the form of error correcting codes.
- 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. The server should avoid reusing images in order to avoid detectability. If all images in the database could be combined in order to make a plausible group of images, the server can choose any corresponding image. If not, other user-determined constraints may apply.
- These images must be made accessible to the recipient in the order of the message. One way to do this is to serve them on a webpage.

0.3.4 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 recipient's machine:

[2]the concept of plausibility in the context of this system is described in section 0.3.2

- 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.

0.4 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. 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.

The threat model for this system is described in section 0.2.

0.4.1 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.

0.4.2 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 want the web pages we create using Zebrafish to be indistinguishable from image galleries found on the Internet.

The method we used in choosing our encoding function (described in Section 0.3.1) ensures that these web pages will be indistinguishable with regard to average luminosity. Thus, the encoding of the images is indistinguishable from the encoding of the message. In order to show that Zebrafish web pages are indistinguishable from "wild" web pages, 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. In creating web pages which send our message, we may unknowingly create web pages which differ from the distribution of "wild" web pages in some other respect. We use the techniques described in Section 0.3.2 to create web pages which do not differ from normal web pages in these secondary respects.

0.4.3 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, we assume our adversary is unwilling to lower the perceived value of the images by modifying their appearance.

This is a very similar to the problem of robustness of watermarking schemes.[3] 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 encoding was changed on only 9 of 194 images tested. This will be insufficient to break our system since we add redundancy into our plaintext in the form of error correcting codes.

There is a worry that the adversary might attempt to destroy the message 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 0-4.

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) [McG01]. It is important to realize that we *must* use these borderline images or their absence will make our system detectable.

[3] It is useful to note that Zebrafish is unsuitable as a system for watermarking images. This is primarily because it does not modify or mark images in any way. It could potentially mark groups of images, but the content holders would not be able to decide which groups. This would not be useful as protecting a specific covertext is the priority of watermarking as mentioned in Section 0.1.1.

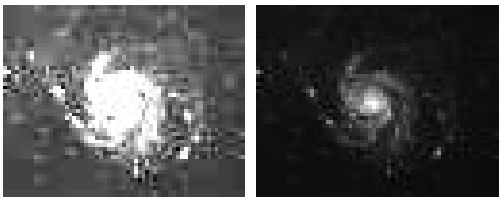

Figure 0-4: Changing the luminosity threshold often alters the appearance of the image significantly

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.

0.5 Implementation

The current implementation of Zebrafish is intended as a proof of concept. It consists of a variety of perl scripts. Principle among these are `putmakedb.pl`, which creates a database of images, `chooseim.pl`, which chooses images to send corresponding to a plaintext message, and `parsewget.pl` which downloads a webpage and extracts a hidden message.

0.5.1 Creating the Database

In order to create a database, the user first assembles a directory of `jpeg` images. These images can come from the web or a digital camera or any other source of digital images. The user points `putmakedb.pl` at the directory of `jpeg` images and it outputs a sorted file with the four bit number corresponding to each pathfilename. The script converts each file to `ppm` format[4] and runs it through the encoding function to assign a 4 bit number. Currently the code only works for `jpeg` images, though it would not be difficult to extend it to include other formats.

0.5.2 Sending Messages

The perl script which is used to send messages (`chooseim.pl`) takes as arguments a directory d of `jpeg` images and a plaintext message m. The message m is encrypted using Blowfish in CBC mode. CBC mode is not optimal for Zebrafish as an adversary can attack the system by making changes to only borderline images. In a real implementation the underlying cryptosystem ought to deal with redundancy in the plaintext to foil this attack. After encryption, the script takes four bit chunks of the ciphertext and searches through the database file for matching images. It outputs this set of images as both a list of filenames and also as rudimentary `html`.

[4] A very simple but inefficient image format

0.5.3 Receiving Messages

In order to receive messages, the script `parsewget.pl` takes in a `url` and an output filename. The script calls the `wget` program to download the images on the site. It then uses the output file to apply the encoding function to each image in order, extracting 4 bits. These encodings are then concatenated together and the message is decrypted.

0.5.4 Example

In order to demonstrate the system, we created a database of images grabbed from an Altavista image search for "cats." As you will see, many of the images had apparently little to do with "cats." An excerpt of the datafile can be seen in Figure 0-5. We then decided to see what would happen if we decided to send the message "meeting at noon" with Zebrafish. The encryption of "meeting at noon" was $52616e646f6d49565cb6ff5e4c6151eab57c56fa5c5bd7da439bdf48f6d4d4ce$. This will require 64 images. In general, the number of images required to send a given message is twice the number of characters, plus whatever extra is required to get the correct block cipher size. The images chosen are shown in Figure 0-6.

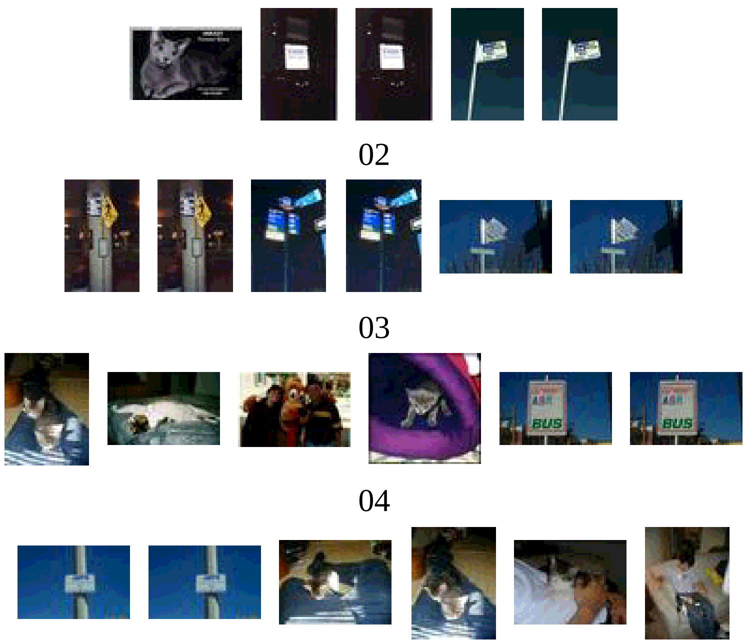

Figure 0-5: An excerpt from the database for the cats pictures (output of `putmakedb.pl`: these are some of the brighter images scoring 13 and 14.

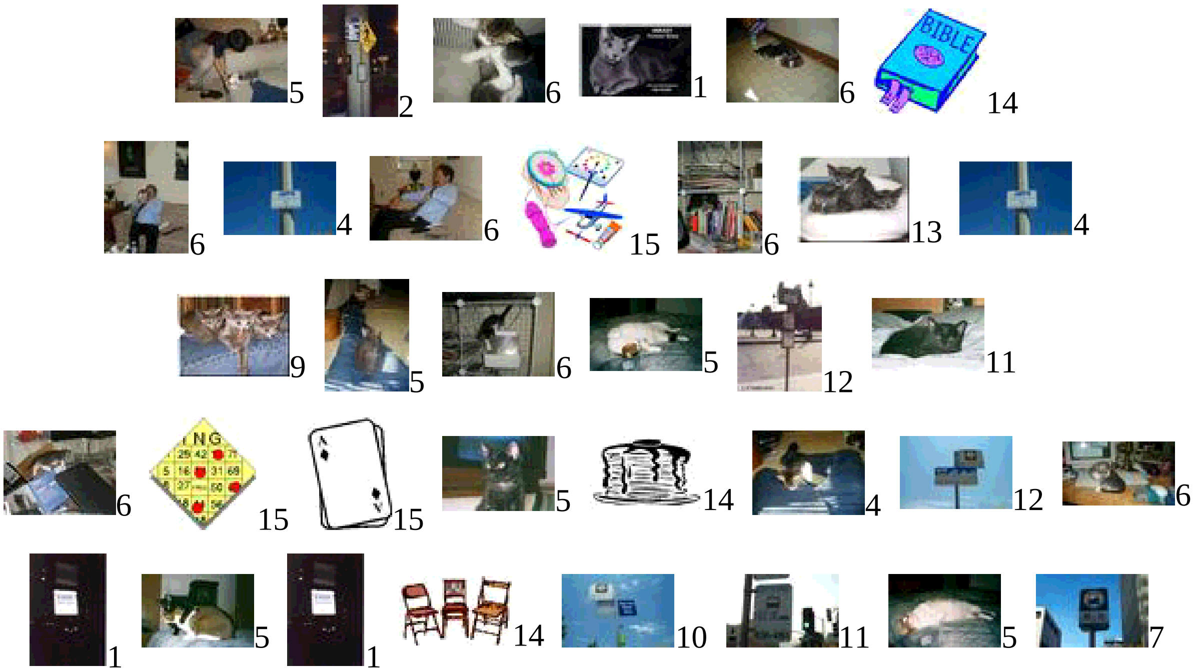

Figure 0-6: A subset of the images in the image gallery, and the numbers they decode to

0.6 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.

Bibliography

[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.

[Ett98] M. Ettinger. Steganalysis and game equilibria. In *Information Hiding*, pages 319–328. 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.

[Kah67] D. Kahn. *The Codebreakers*. The Macmillan Company, 1967.

[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.

[McG01] D. A. McGrew. Integer counter mode. `http://www.ietf.org/internet-drafts/draft-mcgrew-saag-icm-00.txt`, 2001.

[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.

[PAK99] F. A. Petitcolas, R. J. Anderson, and M. G. Kuhn. Information hiding – a survey. In *Proceedings of the IEEE*. 1999.

[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.