

Information Hiding to Foil the Casual Counterfeiter

Daniel Gruhl and Walter Bender

Massachusetts Institute of Technology Media Laboratory

Abstract. Security documents (currency, treasury bills, stocks, bonds, birth certificates, etc.) provide an interesting problem space for investigating information hiding. Recent advances in the quality of consumer printers and scanners have allowed the application of traditional information hiding techniques to printed materials. This paper explores how some of those techniques might be used to address the problem of counterfeiting as the capability of home printers to produce "exact" copies improves.

1 Introduction

The appearance of commercial color photocopiers in the 1970's presented treasury departments around the world with a problem. No longer were special equipment and expertise required to produce passable reproductions of most currencies. Anyone with access to a copy store was a potential counterfeiter. Steps had to be taken to limit the possibility of a counterfeiting explosion.

The U.S. Treasury Department took a multi-pronged approach to deal with this problem [11,12,13]. First, they tried to make the bills themselves more difficult to copy. A combination of features were introduced, such as very fine engraving designed to alias noticeably when undersampled, watermarks, embedded plastic strips, and more recently, special inks that change color when light strikes them at different angles. The treasury sought to make it extremely difficult to produce reasonable copies of bills using color photocopiers.

While this approach is an effective deterrent, and in the end may be the right approach to the prevention of counterfeiting, there remains a problem. The U.S. government honors at face value any bill it has ever printed. Therefore, bills printed prior to the 1970's are still good. Thus, counterfeits of older bills might still be accepted, and those bills do not have as many security features as the new bills do. This is not as much of a problem as it might first appear, since paper money only has an active life of on average 18 months [11]. Until old bills leave common circulation, however, there is still a problem.

Two steps are being taken as intermediate measures. First, many modern color photocopiers have a circuit that tries to detect when a bill is being photocopied, and refuses to do so. This circuit seems to use a simple feature recognition on the image being copied to try to determine if it is a bill. It is, however, easy to defeat this system. One must speculate that this device is meant primarily to "protect people from themselves", forcing them to think about what they are doing when they first attempt to counterfeit a bill.

A second measure has been articulated by Representative Michael Castle, Republican of Delaware and chairman of the House of Representatives Subcommittee on

Domestic and International Monetary Policy. Representative Castle has said that "practical and realistic" measures to tag scanners and printers must be considered, in order to identify the source of the counterfeit notes [14]. If copiers were to encode their serial number in continuous-tone images, possibly through modifications to the dithering algorithm, treasury agents would be able to identify which machine was used in the creation of a bogus bill. It would then be possible to "stake out" the photocopier in question and apprehend the culprits the next time they create some counterfeit cash.

The difficulty that treasury departments have been encountering in recent years is the proliferation of very good, very inexpensive color scanning and printing technology for personal computers [14]. A 720×720 color ink-jet printer lists for \$200–\$300, and a 300 DPI flatbed scanner for \$75. Using these devices, it is possible to create color reproductions that exceed the quality of modern color copiers costing \$20000–\$45000.

The problem

The problem addressed in this paper is how to find a way to bring the same kind of technologies that exist in modern color copiers to the realm of ink-jet printers. The second modification mentioned above, how can a serial number be hidden in an image, is a standard information-hiding problem. The first problem, can an encoding be placed in an image such that the printer can detect it and refuse to print it, is more challenging.

The problem space this presents differs from traditional information-hiding problems. Typically for images, an assumption is made that the image quality might be largely degraded in a signal-to-noise ratio sense (through perceptual coding such as JPEG [Joint Photographic Experts Group]), that arbitrary resampling might be done (through scaling), and that cropping is a possibility. Most commercial systems assume that the image presented to the decoder is not rotated, and often it must not have been translated. Further, there is often an assumption that the image will be in a similar color/luminance space to the original (RGB vs. CMYK for example).

For the money problem, an almost reverse set of circumstances prevails. The image quality sent to the printer is usually excellent. No one is going to make a bill that looks only a little like a real one, and then try to spend it. The size of the reproduction is fixed. Again, trying to spend a bill twice or half-normal size is unlikely to be successful. On the other hand, if someone can print the bill out at a 45 degree angle and defeat the system, it is likely that they will. The same applies to arbitrary translation. It is unlikely, however, that someone will radically crop a bill, as again this adversely impacts the chance of passing it successfully.

Additionally, since the typical consumer ink-jet printer has five to seven fixed colors to use (four to six inks and white from the paper) the image is dithered, trading spatial resolution for color depth. This process results in nonlinear modifications to the image in transferring it to paper. When the image is scanned in again, more nonlinear modifications are introduced, since the scanner acquires an RGB representation of what is scanned, usually at still another resolution, rotation, and translation.

A consideration for any encoding method intended to be embedded in a printer is how the printer "sees" what it is printing. This is usually a small number of lines at a time. A single pass for an ink-jet printer is typically 0.25"×8.5". Any processing

should require only image data from one pass at a time.

Lastly, since such a method might be embedded in a \$200–\$300 consumer product, the encoder cannot be as expensive as one being placed in a \$20000–\$45000 color photocopier. Thus the method needs to be computationally inexpensive.

Paper overview

The rest of this paper addresses potential solutions to the problems addressed above. Much of the work described in this paper is based upon a statistical method of data hiding, Patchwork [9]. This method is detailed in Section 2. In Section 3, Patch Track, a method of encoding a tracking number in an image is discussed. Section 4, Tartan Threads, deals with the more difficult problem of having a printer detect when it should refuse to print a document. This method employs direct-sequence spread spectrum (DSSS). In Section 5, experimental results are presented. The paper concludes with some observations of how these applications of information hiding differ from the more traditional ones and what the future might hold for these techniques.

2 Patchwork

Patchwork is a statistical method that will allow the detection of a single, specific bit in an image. Patchwork imperceptibly embeds in a host image a specific statistic, one that has a Gaussian distribution. This is usually interpreted as a watermark, in the sense “Is IBM’s watermark in this image, yes or no?” By a single, specific bit, it is meant that the algorithm, when given a certain password, can tell if an encoding using that password has been embedded in the image. There are actually two possible encodings that can be made for a given password, a positive one and a negative one. For each encoding, it is also possible to assign a “confidence”, or a measure of how certain it is that the image has been encoded with a particular bit, in a positive or negative sense. Patchwork is independent of the contents of the host image. It shows reasonably high resistance to most nongeometric image modifications.

The Patchwork algorithm [9,10] is detailed here. The following simplifying assumptions are made for the analysis presented here (these assumptions are not limiting, as is shown later): Patchwork is operating in a 256 level, linearly quantized system starting at 0; all brightness levels are equally likely; all samples are independent of all other samples.

The Patchwork algorithm proceeds as follows: take any two points, A and B , chosen at random in an image. Let a equal the brightness at point A and b the brightness at point B . Now, let

$$S = a - b \quad (1)$$

The *expected* value of S is 0, i.e., the average value of S after repeating this procedure a large number of times is *expected* to be 0.

Although the *expected* value is 0, this does not reveal much about what S will be for a specific case. This is because the variance is quite high for this procedure. The variance of S , σ_S is a measure of how tightly samples of S will cluster around the expected value of 0. To compute this, make the following observation: Since $S = a - b$ and a and b are assumed independent, σ_S^2 can be computed as follows (this, and all

Standard Deviations Away	Certainty	n
0	50.00%	0
1	84.13%	679
2	97.87%	2713
3	99.87%	6104

Table 1. Degree of certainty of encoding given deviation from that expected in a Gaussian distribution ($d = 2$)

other probability equations are from Drake¹²):

$$\sigma_s^2 = \sigma_a^2 + \sigma_b^2 \quad (2)$$

where σ_a^2 for a uniform S is:

$$\sigma_a^2 \approx 5418 \quad (3)$$

Now, $\sigma_a^2 = \sigma_b^2$ since a and b are samples from the same set, taken with replacement. Thus:

$$\sigma_s^2 = 2 \times \sigma_a^2 \approx 2 \times \frac{(255 - 0)^2}{12} \approx 10836 \quad (4)$$

which yields a standard deviation $\sigma_s \approx 104$. This means that more than half the time, S will be greater than 43 or less than -43 . Assuming a Gaussian clustering, a single iteration does not tell much. However, this is not the case if the procedure is performed many times.

Repeat this procedure n times, letting a_i and b_i be the values a and b take on during the i th iteration, S_i . Now let S_n be defined as:

$$S_n = \sum_{i=1}^n S_i = \sum_{i=1}^n a_i - b_i \quad (5)$$

The *expected* value of S_n is:

$$S_n = n \times S = n \times 0 = 0 \quad (6)$$

This makes intuitive sense, since the number of times a_i is greater than b_i should be offset by the number of times the reverse is true. Now the variance is:

$$\sigma_{S_n}^2 = n \times \sigma_S^2 \quad (7)$$

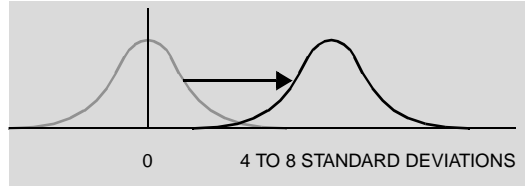


Fig. 1. As δ or n increases, the distribution of S'_n shifts further to the right.

And the standard deviation is:

$$\sigma_{S'_n} = \sqrt{n} \times \sigma \approx \sqrt{n} \times 104 \quad (8)$$

Now, compute S_{10000} for a picture, and if it varies by more than a few standard deviations, it is fairly certain that this did not happen by chance. In fact, since as will be shown later S'_n for large n has a Gaussian distribution, a deviation of even a few σ_s 's indicates to a high degree of certainty the presence of encoding (see Table 1).

The Patchwork method artificially modifies S for a given picture, such that S'_n is many deviations away from expected. To encode a picture, we:

1. Use a specific key for a known pseudo-random number generator to choose (a_i, b_i) . This is important, because the encoder needs to visit the same points during decoding.
2. Raise the brightness in the patch a_i by an amount δ , typically in the range of 1 to 5 parts in 256.
3. Lower the brightness in b_i by this same amount δ (the amounts do not have to be the same, as long as they are in opposite directions).
4. Repeat this for n steps (n typically $\sim 10\,000$).

Now, when decoded, S'_n will be:

$$S'_n = \sum_{i=1}^n (a_i + \delta) - (b_i - \delta) \quad (9)$$

or:

$$S'_n = 2\delta n + \sum_{i=1}^n (a_i - b_i) \quad (10)$$

So each step of the way an expectation of $2 \times \delta$ is accumulated. Thus after n repetitions, S'_n is expected to be:

$$\frac{2\delta n}{\sigma_{S'_n}} \approx 0.028\delta\sqrt{n} \quad (11)$$

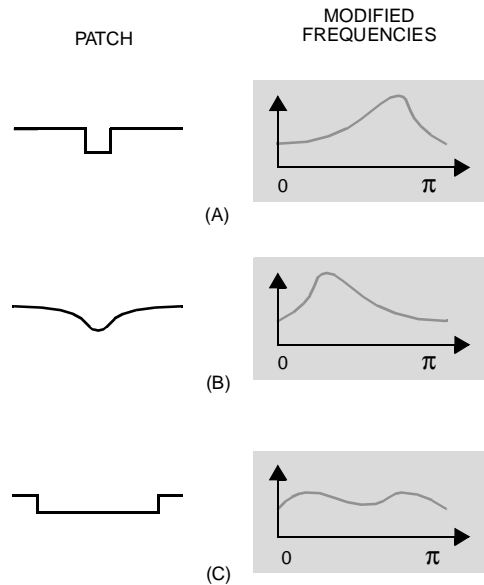


Fig. 2. The contour of a patch largely determines which frequencies will be modified by the application of Patchwork.

As n or δ increases, the distribution of S'_n shifts over to the right (Figure 1 and Table 1). In Figure 1, as δ or n increases, the distribution of S'_n shifts further to the right. If shifted far enough, any point that is likely to fall into one distribution is highly unlikely to be near the center of the other distribution.

While this basic method works well by itself, a number of modifications have been made to improve performance including:

1. Treating *patches* of several points rather than single points. This has the effect of shifting the noise introduced by Patchwork into the lower spatial frequencies, where it is less likely to be removed by lossy compression and typical Finite Impulse Response (FIR) filters. Additionally, it makes alignment easier.
2. Making Patchwork more robust by using a combination with either affine coding (described later) or some heuristic based upon feature recognition (e.g., alignment using the interocular line of a face). Patchwork decoding is sensitive to affine transformations of the host image. If the points in the picture visited during encoding are offset by translation, rotation, or scaling before decoding, the code is lost.
3. When decoding large patches, sampling all the points around the center point.
4. Using a visibility mask to avoid putting patches where they would be noticeable.
5. Superimposing a random mask on top of a cone-shaped patch to mask visibility (see Figure 4).

Patch shape

The shape of the patches deserves some comment. Figure 2 shows three possible one-dimensional patch shapes, and next to them a very approximate spectrum of what a

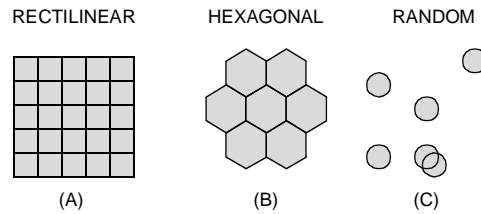


Fig. 3. Patch placement affects patch visibility.

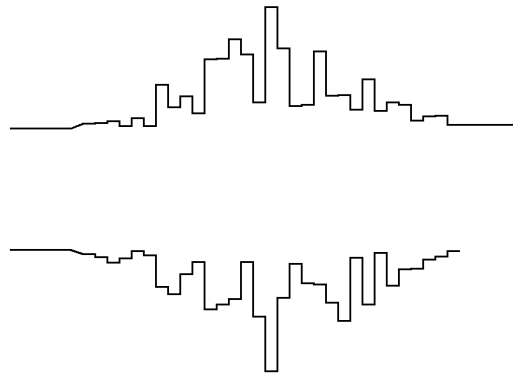


Fig. 4. Random cone-shaped patch used with Patch Track method. The depth across the patch is chosen at random, but constrained to a cone of radius 15 and depth 10 for most of the experiments.

line with these patches dropped onto it pseudo-randomly would look like. In Figure 2A, the patch is very small, with sharp edges. This results in the majority of the energy of the patch being concentrated in the high frequency portion of the image spectrum. This makes the distortion hard to see, but also makes it a likely candidate for removal by lossy compressors and for the non-linear transforms introduced by dithering. If one goes to the other extreme, as in Figure 2B, the majority of the information is contained in the low-frequency spectrum. The last choice, Figure 2C, shows a wide, sharp-edged patch, which tends to distribute the energy around the entire frequency spectrum.

The optimal choice of patch shape is dependent upon the expected image modifications. If JPEG encoding or dithering is likely, then a patch that places its energy in the low frequencies is preferable. If contrast enhancement is to be done, placing energy in higher frequencies would be better. If the potential image modifications are unknown, then spreading the patch energy across the spectrum would make sense.

The arrangement of patches has an impact on patch visibility. For illustration, three possibilities are considered (Figure 3). The simplest method is shown in Figure 3A, a simple rectilinear lattice. While simple, this arrangement is often a poor choice if a high n is to be used. As the grid is filled in, continuous edges of gradient are formed. The human visual system (HVS) is very sensitive to such edges. A second

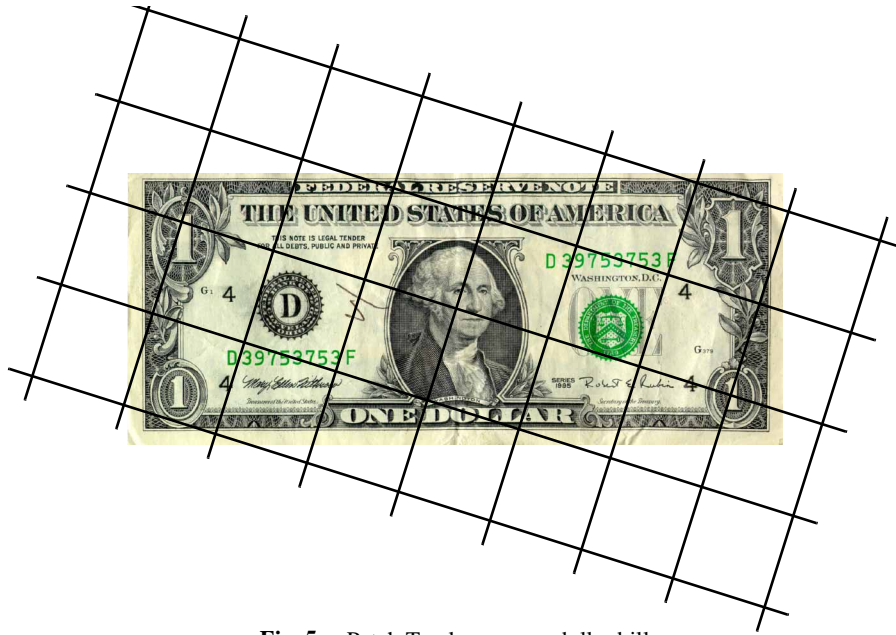


Fig. 5. Patch Track on a one-dollar bill

choice, Figure 3B, breaks this symmetry by using hexagons for the patch shape. A preferred solution, shown in Figure 3C, is a completely random placement of patches. An intelligent selection of patch shape in both the horizontal and vertical dimensions will enhance the effectiveness of patchwork for a given picture.

Variance

In order to evaluate the likelihood that a bill is encoded, some idea of the variance of S_n is needed. In currency all luminance values are not equally likely. In U.S. currency there tend to be many white or black regions and few mid-tone regions.

While the variance differs for each sample, for decoding purposes it helps to choose a typical variance rather than recomputing the variance for each bill examined. S_n was calculated for 10000 seeds, using 10000 patch pairs and a 3×3 decoding area on both a one-dollar bill and a five-pound note. In both cases the full bill was examined by the decoder, rather than just a small patch. The variance of these S_n was then computed.

The variance of the dollar bill is 77621. The variance of the pound note is 79105. Allowing for the consideration of nine times as many points, it was found that these notes have a variance about 60% higher than an assumption of uniform variance would suggest. This can be corrected by simply encoding with 60% more depth.

Summary

There are several limitations inherent to the Patchwork technique. The first is the extremely low embedded data rate it yields, usually a one-bit signature per image.

This limits its usefulness to low bit-rate applications such as the digital watermark. Second, it is necessary to *register* where the pixels in the image lie. While a number of methods have been investigated, it is still somewhat difficult to decode the image in the presence of severe affine transformations. These disadvantages aside, without the key for the pseudo-random number generator, it is extremely difficult to remove the Patchwork coding without degrading the picture beyond recognition.

The Patchwork method is subject to cryptographic attack if it is used to encode a large number of identically sized images using the same key. If the images are averaged together, the *patches* will show up as lighter or darker than average regions. This weakness is a common one in cryptography, and points to the truism that for a static key as the amount of traffic increases, it becomes easier to “crack” the encryption. One solution is to use multiple pseudo-random patterns for the patches. Even the use of just two keys, while increasing decoding time, will make Patchwork much more robust to attack. Another solution is to use the same pattern, but to reverse the polarity of the patches. Both solutions deter cryptographic attack by averaging.

3 Patch Track

If an image is printed on a particular printer, is it possible to figure out which printer was used given only a printed sample? This problem is addressed in color copiers by using a dither pattern that encodes the copier’s serial number. This is not as practical for ink jet-type printers, where the dithering is usually done in software on a host computer, or on a plug-in card such as a Postscript processing module.

In seeking an alternative method, there are several constraints: such a method needs to be able to encode on the order of 32 bits in the target image to hold a serial number. The encoding needs to not impact image quality adversely. For analysis purposes, it should be possible to “find” these bits using a flatbed scanner of typical consumer resolution (600 DPI or less). And lastly, the encoding should be one that does not require extensive computational resources, since ultimately the goal would be to embed such an encoding method in the printer itself. (Perhaps the algorithm should be executable on a Microchip PIC16C84 microprocessor or similar chip). A simple modification of an existing information hiding technique, Patchwork meets these requirements.

Tracking bits are encoded in an image using a sequence of these positive and negative watermarks to encode the ones and zeros. For example, the “password” for the first bit might be “IBM0”, the second bit “IBM1”, etc. Since when using the Patchwork method, the embedded watermarks corresponding to different passwords are nearly orthogonal, many bits can usually be encoded in an image before either visibly degrading the image quality, or interfering with the ease of data recovery.

This method can serve a dual role. Traditional watermarking is used to identify whether an image has been encoded with a particular mark at all. Image tracking tries to encode a tracking number in an image, but the decoder is just trying to recover the bits with high accuracy, not identify if the image has been encoded in the first place.

In Patch Track, the first bit can serve as a “marker”, by encoding that bit to a much higher level of certainty. This is important because if a bit is only encoded to the 99% certainty level, there will be on average one false positive for every hundred images or orientations examined. For a watermarking application used on the Internet,

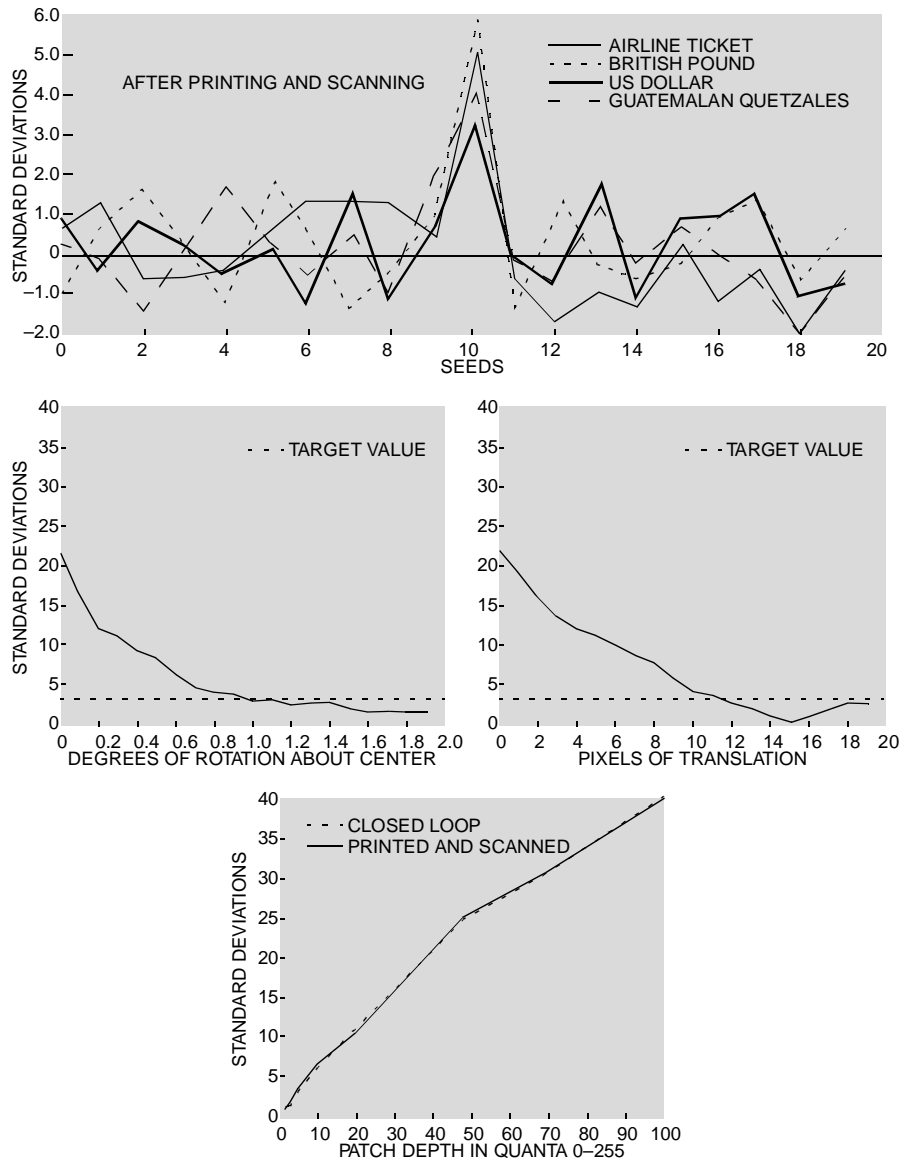


Fig. 6. Patch Track results. A variety of documents, encoded with a password of 10 are decoded after printing and scanning with various passwords (top), decoded after rotation (middle, left), decoded after translation (middle, right), and encoded at various pixel depths (bottom). Note that printing and scanning had almost no impact on data recovery.

the marker bit realistically needs to have an encoding level that returns one false positive in 100000 (99.999%) or better, or a method will be swamped with false positives.

If this lead marking bit is detected, it is known that the image has watermarks in the particular orientation being tested. The rest of the bits which define the tracking number can be encoded in a much "lighter" (less certain and less degrading to the image) fashion, since there is no question of whether there are bits in the image, only if those bits are positive or negative.

Encoding algorithm

The serial number tracking algorithm presented here is an asymmetric algorithm. The encoding is easy to place in a document, but at the expense of being difficult to recover, due to potential variations in orientation of the encoded document. This is a reasonable assumption if the encoding is going to be embedded in nearly every document printed while the decoding is done only in the few cases where a crime has been committed. Thus, no special effort is taken to make the Patch Track data easy to find, since it is assumed that many orientations can be searched on a counterfeit bill to identify the printer used.

However, in order to detect features placed in a bill using the Patchwork algorithm, it has been experimentally determined that they need to be on the order of a tenth of an inch in size. This allows for the drop in effective resolution because of dithering, as well as slight misalignments when the bill is scanned.

Since the bill is 2.5 inches across on the short side, this suggests using patch zones that are around 0.88 inches, guaranteeing that at least three of them will fall completely on the bill. Each of these regions, then, will need to hold about 12 bits (See Figure 5).

Taken together, these constraints equate to encoding an image in area approximately 760×760 pixels of printer resolution, using patches around 90 pixels in size. One bit in each region will have to be strongly encoded, and the rest weakly. Strong encoding corresponds to about 10000 patches, and weak encoding to about 2000 patches. An encoding depth of 20 out of 256 is used with a simple visibility mask. The encoding is done using the random cone method.

Decoding

The decoding process is more computationally intensive. There is first a grid search of the image in a variety of rotations. Once a target region has been located, a gradient search may be done to find the exact alignment of the system. The "weak" bits (that is, the tracking numbers) are then decoded, and the alignment of the system tells where other blocks might be found. If desired, the overlap of several copies of the same encoded regions may allow multiple copies of the region to contribute to the decoding process.

4 Tartan Threads

The second half of the counterfeiting problem involves preventing bills from being printed in the first place. The ink-jet printer only "seeing" a print line a quarter inch wide complicates this problem. This limits the size of the features that the printer can examine when determining whether or not to print.

As a result, the approach considered here involves inserting a chosen feature in



Fig. 7. The region of the bill used in the Tartan Thread experiment

the document to be printed which can later be detected, even when looking at a region as small as a single print line.

Approach

The approach we have tried involves the scattering of “threads” of encoding throughout the image. These “threads” can be created by any of a number of spread spectrum techniques, including Patchwork. The particular technique chosen depends on the medium.

For detecting a code using a simple piece of hardware, a simple encoding method is needed. Given the constraints of this problem, traditional linear Direct Sequence Spread Spectrum (DSSS) [4, 5] seems to be a good choice.

A “thread” might be arranged as a grid of long, narrow regions. The thread itself is broken up into several regions, typically between one and two hundred of them. Within each region some algorithm is used to hide a “bit”. Since these regions are very small, it is assumed that the accuracy of these bits will be suspect. It is important that the method chosen makes it equally likely that an unencoded region returns a positive or negative encoding. So, to encode a “thread”, a pattern of positive and negative encodings is chosen. The regions of the thread are then encoded with those bits.

To check if a certain region is encoded, one decodes each region of the suspect thread. This decoded bit stream is then compared to the encoded bit stream, and a binomial probability distribution function can be calculated to check for the likelihood of encoding.

5 Experiments

For all of the trials done, the printer was an Epson Color Stylus 800™ operating at 720×720 DPI, with Microweave enabled. The images were translated from Postscript to Epson printer control language by Ghostscript, version 5.04. They were dithered by the Floyd-Steinberg algorithm operating in a CMYK color space model. Printing was done on Epson photo-quality ink-jet paper to achieve the highest possible resolution.

For scanning, an HP ScanJet 4C™ flatbed scanner was used, operating at a variety

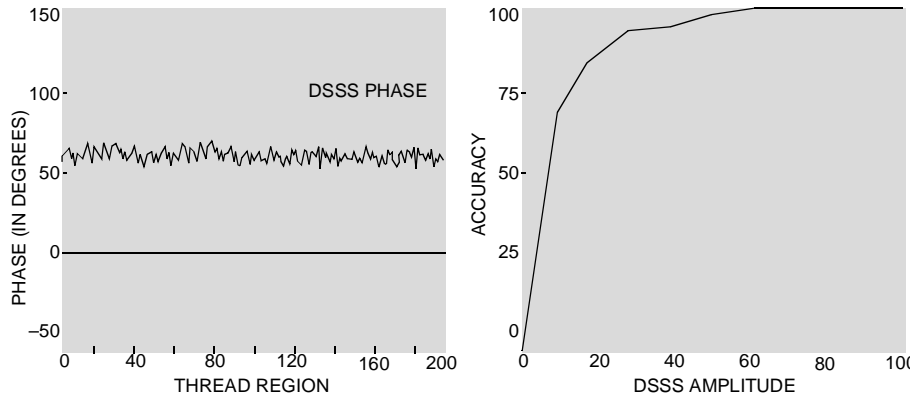


Fig. 8. Tartan Thread results: phase for each 200 thread elements (left) and error rate versus encoding amplitude (right)

of resolutions (all less than 300 DPI). Images were acquired directly into Adobe PhotoShop™ operating in RGB mode for saving to 24 bit TIFF files.

The limiting factor for most of these techniques was the available memory on the analysis machines. Operating on a 2000×3000 full color image stored in high precision format requires ~138 megabytes of RAM, per copy of the image being stored. Continued work in this area necessitates either moving to disk-based memory methods, or as was done for this paper, working with smaller than maximum resolution images. Our standard image size for these experiments was around 1000×450 for encoding and two to three times that for scanning and decoding (printed at 100 dpi and scanned at 100–300 dpi).

Patch Track results

Tests were performed using a U.S. five-dollar bill, a British five pound bank note, a Guatemalan one hundred quetzales bank note, and an airline ticket. The encoded area is spread throughout each document. Each document is printed and then scanned. A random cone-shaped patch of maximum depth 10 (out of 256) and radius 15/100 of an inch was used in the experiments. To test the Patch Track method, the bill is “decoded” with a number of different password values, using a 3×3 decoding block. The encoding was done with the password of 10.

As can be seen in Figure 6 (top), Patch Track returns a significant result for the encoded password value (10) and a much smaller response for the other values in variety of documents, include currencies from several nations and an airline ticket. In interpreting these results, it is important to remember that the σ calculated has a Gaussian distribution, therefore the target value of 4 represents a degree of certainty that exceeds 99.98%. As can be seen in Figure 6 (bottom), the encoding can be increased to any desired value without much difficulty by varying either the patch depth or number of patches.

Also shown in Figure 6 are examples of the impact of affine transformations on the Patch Track method. A Barbados five-dollar bank note was subject to rotation and

translation during the scanning process. The one degree of rotation resistance is equivalent to misalignment of up to a 12th of an inch, an easily achieved target. The shape of the translation data reflects the shape of the cone-shaped patches used in the experiment.

Tartan Thread results

Tests were performed using a “weathered” U.S. one-dollar bill. For this test, the encoding was restricted to the densely printed region (See Figure 7) of the bill. All of the thread-area phases were encoded to 90 degrees. After encoding, the bill was printed. This print out was then scanned in. This scanned image was used for decoding. Each thread area was decoded and the phase of its carrier was found. The results are shown in Figure 8.

While there is an obvious phase alignment error, it is clear that most of the thread areas decode to within a very tight tolerance of each other. Several bits per area could be sent with a tolerance this tight. Assuming all phases are equally likely, a clustering this tight (± 10 degrees out of 360) will occur less than 1 in 10^{251} times. There is little likelihood of unintentional triggering.

Discussion

Information hiding of this type presents a challenge, mainly because the size of the target data space (i.e., the region the printer sees at any one time) is so small. With current technologies the data space does not exceed 180 pixels in width. When considering the aliasing effects caused by scanning and resampling, this effective data space drops to 90 pixels. To be able to compensate for possible misalignments, the necessary redundancy drops the data space to 45 pixels. The shortest dimension on a dollar bill is approximately 2.5 inches. Quartering results in length of around 450 pixels. Thus, the largest data space that could be reasonably hoped for on a dollar bill is about 20250 pixels. This is less than a third of the pixels in a 320×240 image found on a typical web page. Furthermore, since the typical print head is using only four to six inks, it has at most a “4 bit” image depth. In practice, dithering reduces the spatial resolution even further.

Another bottleneck to consider is with the scanner. A 300 DPI scanner has uniquely distinguish features at 150 DPI. This means that a 2.5×0.25 " area will give the potential counterfeiter less than 20000 pixels to start with. Once this is reduced by dithering and aliasing, the data space is comparable to a 40×50 thumbnail. This is simply too small an area to hope to do the kind of robust, effective information hiding one would like.

These are transitory problems. As scanning and printing resolutions continue to increase, there will soon be adequate data space to do more interesting experiments.

6 Conclusions

The problem space of security documents (high-value certificates such as airline tickets, stock certificates, etc.) presents an intermediate step between the geometrically constrained world of today’s commercial watermarking techniques and that of translation, rotation, and cropping independent information hiding.

Fixed scale allows the migration of many traditional time-domain spread-spectrum techniques into a “2D” environment. Exploiting the “printer/scanner” data path

presents the opportunity to explore the highly non-linear and relatively poorly understood “transmission medium.” It also takes information hiding “out of the computer” and allows its application to tangible media (i.e., physical objects).

7 References

1. E. Adelson, *Digital Signal Encoding and Decoding Apparatus*, U.S. Patent No. 4,939,515 (1990).
2. D. L. Hecht, “Embedded Data Glyph Technology for Hardcopy Digital Documents,” *SPIE* **2171** (1995).
3. K. Matsui and K. Tanaka, “Video-Steganography: How to Secretly Embed a Signature in a Picture,” *IMA Intellectual Property Project Proceedings* (1994).
4. R. C. Dixon, *Spread Spectrum Systems*, John Wiley & Sons, Inc., New York (1976).
5. S. K. Marvin, *Spread Spectrum Handbook*, McGraw-Hill, Inc., New York (1985).
6. G. B. Rhoads, *Method and apparatus responsive to a code signal conveyed through a graphic image*, U.S. Patent No. 5,710,834 (1995).
7. I. Cox, J. Kilian, T. Leighton, and T. Shamon, “Secure Spread Spectrum Watermarking for Multimedia,” *NECI Technical Report* 95-10, NEC Research Institute, Princeton, NJ (1995).
8. A. V. Drake, *Fundamentals of Applied Probability*, McGraw-Hill, Inc., New York (1967).
9. W. Bender, D. Gruhl, N. Morimoto, and A. Lu, “Techniques for Datahiding,” *IBM Systems Journal* **35** 3&4 (1996).
10. W. Bender, D. Gruhl, and N. Morimoto, *Method and Apparatus for Data Hiding in Images*, U.S. Patent No. 5,689,587 (1996).
11. “Fundamental Facts About Money,” *Federal Reserve Bank of Atlanta*, <http://www.frbatlanta.org/publica/brochure/fundfac/money.htm> (1997).
12. “Genuine or Counterfeit?,” *Federal Reserve Bank of Atlanta*, <http://www.frbatlanta.org/publica/brochure/counter/counterf.htm> (1997).
13. “Currency Page,” *The Department of the Treasury*, <http://www.treas.gov/whatsnew/newcur/currency.html> (1997).
14. “Ink-jet counterfeiting on the rise,” *Reuters*, <http://www.zdnet.com/zdnn/content/reut/0401/302907.html> (April 1, 1998).

8 Acknowledgments

This work was sponsored in part by the MIT Media Laboratory’s News in the Future research consortium and IBM. The authors would like to thank Fernando Paiz and Raymond Hwang for their help in conducting the characterization experiments.