Visual Concurrent Codes

Leemon C. Baird III, Dino Schweitzer, William L. Bahn
Academy Center for Cyberspace Research,
US Air Force Academy
Colorado Springs, CO, USA

Sam Sambasivam
Department of Computer Science
Azusa Pacific University
Azusa, CA, USA

Leemon.Baird@usafa.edu ssambasivam@apu.edu

Abstract

A form of visual jam resistant coding is presented. Using *Visual BBC*, a modified form of BBC (Baird, Bahn, Collins) coding, it is shown that several images can be printed on clear plastic, such that when they are superimposed (i.e. a bitwise OR of the pixels is performed), the resulting image may look random, but the original images can still be recovered without any information about the original pictures, and without any secret. BBC is a complex subject to understand, and so Visual BBC aids the teaching of how BBC coding works, by giving students a concrete, physical model. Examples are shown, illustrating that it is possible for legitimate BBC codewords to actually look like recognizable images, rather than just random binary strings. This allows us to superimpose arbitrary pictures and separate them again in linear time without using any keys or channels specific to each picture. This is not possible in any other coding systems, such as error correcting codes, superimposed codes, or steganography systems. In addition, a number of analysis problems are described that can be given to students, which are motivated by the issues arising in Visual BBC, and which further increase student understanding of the system.

Keywords: BBC, coding theory, education, concurrent codes, visualization, visual cryptography.

Introduction

Cryptography is a complex and difficult subject for students to learn: a message is scrambled using a key and an algorithm that tends to be highly abstract. That is why many different uses of technology have been explored to make it more concrete and easier to understand. One of these is *visual cryptography*. In visual cryptography, the "key" is simply a sheet of clear plastic with apparently-random dots printed on it. The "encrypted message" is another such sheet, which also appears to be random. But when the two sheets are set on top of each other, a hidden message appears. There is an enormous literature on visual cryptography (see all of the References section below, other than the first four). It was originally invented and developed for cryptographic reasons, but the pedagogical uses are clear. It allows the student to physically manipulate the elements of the system, and visually see the decryption process in action. Although a computer must still be involved to create the two images, the student can carry out the decryption step without a computer. For example, the reader is invited to experiment with the visual cryptography system online at (Baird, 2000).

Red	0010000000001000000010000
Green	0010000100000000000001000
Blue	0000010000001000001000000

Figure 1: Example codewords for three messages.

The more recent field of *concurrent codes* is even more abstract and difficult for students to grasp. The original code developed in that field is the Baird Bahn Collins (BBC) algorithm (Baird et. al., 2007), which is complex enough that students may have difficulty grasping it at first. This suggests that it would be useful to have something analogous to visual cryptography for BBC codes. This paper proposes the first such *Visual BBC* system, and gives several variants.

Red	0010000000001000000010000
Green	0010000100000000000001000
Red+Green	0010000100001000000011000

Figure 2: Superimposing messages via a bitwise OR of their codewords.

In BBC, an algorithm is used to convert a message into a binary string of mostly zeros with only a few ones. This string is the *codeword* for that message. For example, Figure 1 shows how the three messages "Red", "Green" and "Blue" might be encoded as BBC codewords.

This is just a simple example. In an actual implementation, the codewords would be much longer, and would be generated in a more complex way.

In cryptography, encryption is used to hide messages. In BBC coding, the goal is the opposite: to make the messages clear, even when they are *superimposed*. Two messages are superimposed by taking a bitwise OR of their bits. For example, the "Red" and "Green" messages can be superimposed as in Figure 2. Both messages can be transmitted simultaneously by transmitting their superimposed codeword.

Note that there is a 1 in the "Red, Green" result in every position where there is a 1 in "Red" or "Green" or both. It is clear by inspection that given the "Red+Green" string, it can be seen to contain "Red" and "Green", but not "Blue". This decoding can be done easily when there are only 3 possible messages. But it is tedious to do without a computer when there are, say, 2^{1000} possible messages.

A superimposed combination of BBC codewords can be separated back into their original messages through a very simple algorithm. Although the BBC decoding algorithm will not be given here, it is a linear-time algorithm that separates images with very low error rates, as long as the combination of codewords has more 0 bits than 1 bits.

Red	00100	Green	00100	Blue	00000	Red+Green	00100
	00000		00100		10000		00100
	00100		00000		00100		00100
	00000		00000		00010		00000
	10000		01000		00000		11000

Figure 3: Codewords wrapped around to form a rectangle of bits.

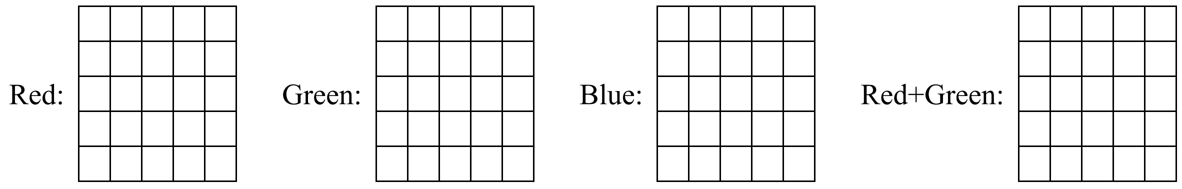


Figure 4: Codewords wrapped and colored to form images

At this point, it would be natural to go on and show the students the details behind encoding and decoding. But first, it would be useful to put the above examples in a form that the students can understand more easily. In the example above, each string contains 25 bits, and is written as a long row of bits. The example might be clearer if they were instead written as a 5 by 5 array of bits. In that case, the example would look like Figure 3.

This might be clearer if the digits 1 and 0 were replaced with black and white squares, respectively, as in Figure 4.

In this form, a type of visual BBC system could be created. Print out the pattern for Red on a sheet of clear plastic, and similarly for Green and Blue. Then the student can form Red+Green by laying the Red and Green sheets on top of each other. The other combinations Green+Blue, or Red+Blue, or even the complete Red+Green+Blue would be created similarly.

This is an improvement, but there are still several issues that aren't made clear to the student. If a student is shown the superposition of Red+Green, it won't be immediately obvious that it is simply a superposition of the Red and Green patterns. Nor is it obvious that it would be difficult to extract the identities of Red and Green given only the pattern Red+Green. These two points would be clearer if the patterns for Red, Green, and Blue were all images that are recognizable to the eye. It would then be clear that the superposition of two or more pictures can be very difficult for the human visual system to pull apart.

Unfortunately, the original BBC system does not usually generate recognizable images for messages that would typically be encoded. When a typical message is encoded and the codeword is transformed into an image, the image typically looks like a white field covered with a random distribution of black dots. The next section explains the BBC coding algorithm in detail, then proposes a method for finding messages that will produce codewords that look like recognizable pictures.

Pictures from BBC

The BBC coding algorithm is shown in Figure 5, for a simple example of encoding the message 111010.

Message
111010

Message Prefixes		Hash of each prefix
1	→	23
11	→	17
111	→	1
1110	→	19
11101	→	5
111010	→	24

Codeword (with a 1 in those positions chosen by hash)
1000100000000000101000110
= 10001 00000 00000 01010 00110

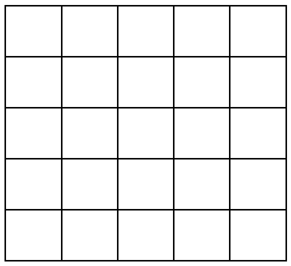
=


Figure 5: The BBC coding algorithm, with an example of encoding the message 111010 as a 5 by 5 black and white image

First, the message must be given in binary, such as by converting each letter in a string to its ASCII code in binary.

Next, all prefixes of the message are taken. So we take the first bit, the first two bits, and so on, giving the prefixes 1, 11, 111, 1110, etc.

Next, each prefix is passed through a hash function. A hash function is any function that maps a prefix to an integer, scrambling it in the process in a way that looks random. For example, if the codeword will have 25 bits, then the hash function takes in the prefix, and converts it to an integer between 1 and 25, inclusive. This function can be almost anything. Common examples are algorithms such as SHA-1 or MD5. In this example, we assume the hash maps the prefix 1 to the integer 23, maps the prefix 11 to the integer 17, and so on.

Finally, the codeword is defined to be all zeros except for those positions that were chosen by the hash. So in this example, the codeword is all zeros, except for a 1 in positions 23, 17, 1, 19, 5, and 24 within the codeword.

Finally, the codeword is converted into an image as before, wrapping it around to fill a rectangle, and representing 0 as a white square and 1 as black.

If the hash function does a good job of scrambling its inputs in a way that looks random, then the image generated by a typical message will look random. It will look like a white field with a random spattering of black dots.

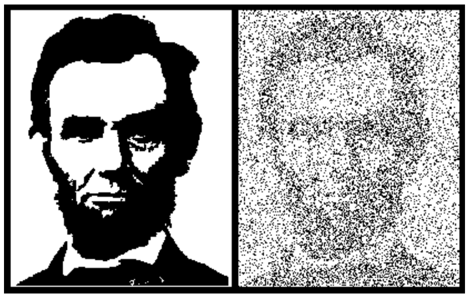


Figure 6: An image of Abraham Lincoln with roughly equal areas of black and white (left), and a BBC codeword that approximates it (right).

But why use a typical message? If the goal is to get a desired image for a codeword, then the choice of message doesn't matter. So instead of starting with a message of 111010 and seeing what image it produces, we could start with a desired image, and try to find a message that generates that image, as closely as possible.

For example, we could choose an image whose top half is black and whose bottom half is white. Can we find a message that will generate that image? Not exactly, but it is possible to get close.

This method will work by walking through the decoding process and, at each step, choosing the next message bit whose encoding is most consistent with the desired codeword image.

So should the message start with a 0 or 1? To choose, we would calculate the hash of 0 and the hash of 1, and look at the two positions chosen by the hash function.It may be that the hash of 0 is a position in the region we have decided should be black, and the hash of 1 is in a position in the region we have decided should be white. In that case, we should start the message with a 0. That will put one black dot in the region that we want to be black. Similarly, if the hash of 1 is in the black region and the hash of 0 is in the white, then we should start the message with a 1 bit. On the other hand, if both 0 and 1 hash to the same color region, then we can pick the first bit arbitrarily. It won't matter. Once the first bit of the message has been chosen, say it is chosen to be a 1, we continue in the same way with the second bit of the message, choosing it to be 0 or 1 based on the color that the hash of 10 and 11 falls in.

This method will actually give a recognizable image. That is because at each step there are 4 different outcomes for the position of hashing 0 and 1. They can be in regions that are desired to be (black, black) or (black, white) or (white, black) or (white, white). If the hash function looks random, and the black and white regions have equal area, then each of those 4 outcomes is equally likely. In the first 3 cases, we can always choose the bit so that the black dot will be placed in the region that we want to be black. Only in 1 case are we forced to put a dot in the white region. So, after choosing many bits of the message, the encoding will be an image that has a density of dots in the dark areas that is 3 times higher than in the light areas. Thus, ordinary BBC can actually have codewords that are recognizable images, without modifying the algorithm at all. We simply start with the desired codeword, and then use a greedy method to choose the message.

The analysis in the previous paragraph makes one simplifying assumption. It shows that 3 times as many marks will be put in the dark region as in the white, and therefore assumes the dark region will be 3 times darker than the white. But this ignores the fact that sometimes a mark is added on top of an existing mark, and so doesn't make the region any darker. Since this will happen more often in darker regions, the contrast ratio will be slightly less than 3. But for very light pictures (with a low final density), the contrast ratio will approach 3.

An example is given in Figure 6. The image on the left is a black and white image of Abraham Lincoln. The image on the right is an actual BBC codeword. The codeword was generated by a message that was chosen specifically to give that image, using the algorithm just described.

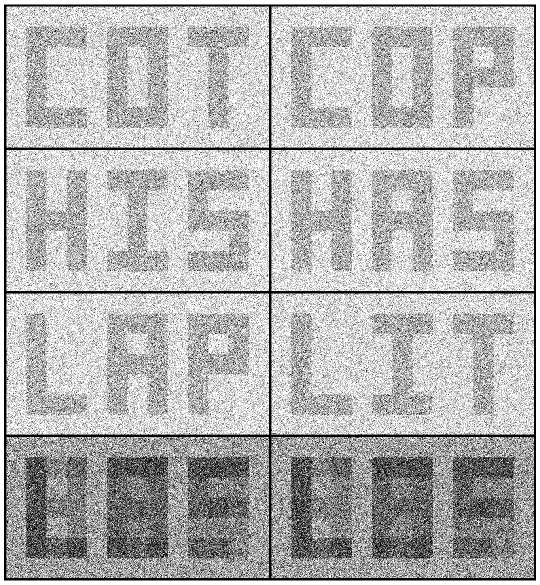


Figure 7: Eight images that can be printed on separate transparencies to illustrate BBC coding. In the left column, the 3 separate images for COT, HIS, and LAP are superimposed to give the image at the bottom, and similarly for the right column using images for COP, HAS, and LIT. In both cases, the human eye has difficulty in recognizing that the first third of the result is a superposition of C+H+L, that the second position is O+I+A, and the third is T+S+P. But even if that can be discovered, the human has no way to know the specific words in each image, because the same letters are superimposed in each case. The two superimposed images look practically identical to a human. But the BBC algorithm allows a computer to quickly and easily decode the lower-left image as COT+HIS+LAP and the lower right as COP+HAS+LIT. This visual, concrete example gives the students an intuitive understanding of the power of the BBC algorithm.

This system can now be used to illustrate BBC coding in a way that is accessible to students. Print the BBC codeword form of the Abraham Lincoln image on clear plastic. Print a different image on another sheet of plastic. Laying one on top of the other will give an image that is difficult to for a human to visually interpret. Laying 3 or 4 on top of each other will likely give an image that is totally incomprehensible to the human eye. Yet the BBC algorithm will easily decode the combined image, as long as the combined image has a density less than one half (i.e. there are more white pixels than black in the combined image).

Using Visual BBC

BBC codewords are designed so that a computer can easily recover them after several have been superimposed (by combining the bits with a bitwise OR). Visual BBC can aid in teaching that concept by printing several visual codewords on separate transparencies, and allowing students to see them combined on top of each other. The students can then try to guess what the individual pictures are, then separate them to check that guess. Figure 7 gives an excellent example to print out and use for this purpose.

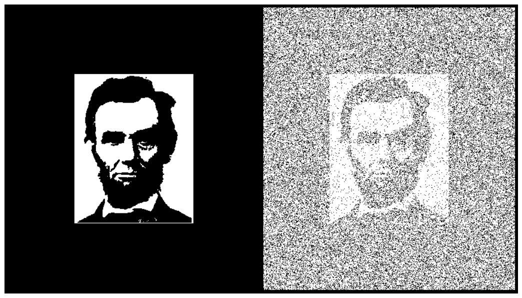


Figure 8: The algorithm applied to a target image with a black frame, to increase the contrast ratio.

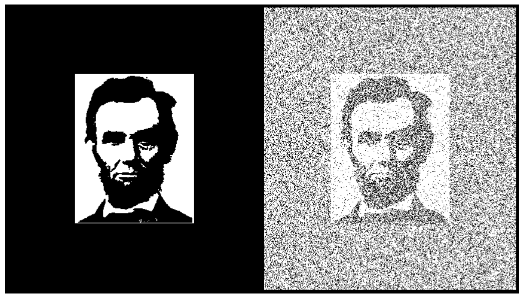


Figure 9: The algorithm applied to a target image with a black frame, with the additional tweak that when given a choice between a black region of the picture or a location in the frame, it always chooses the former. This improves the contrast ratio in the picture, so the dark parts of the picture are even darker than the frame.

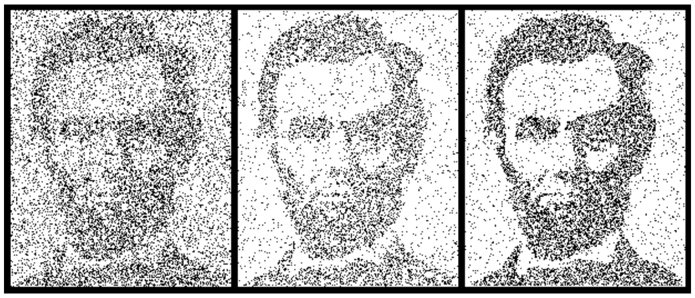


Figure 10: The original algorithm applied to an arbitrary target (left), with a frame added (middle), and with both a frame and a preference for putting marks in the picture itself (right).

Improving Visual BBC

It is useful that this hands-on teaching aid can be built using ordinary BBC. However, the images don't have the contrast ratio that might be desired. In other words, the dark regions are less than 3 times darker than the light regions (i.e. the fraction of pixels that are black is less than 3 times greater in the dark regions than in the light regions). So one might ask whether there is a way to improve this. It turns out that the contrast ratio in the final image can be improved by making the target image have a higher fraction of black pixels. It is difficult to achieve this by modifying the image itself without destroying it, but it is easy to accomplish by adding a large, black border.

Figure 11: The original Lena image in grayscale, which cannot be used directly in the Visual BBC algorithm.

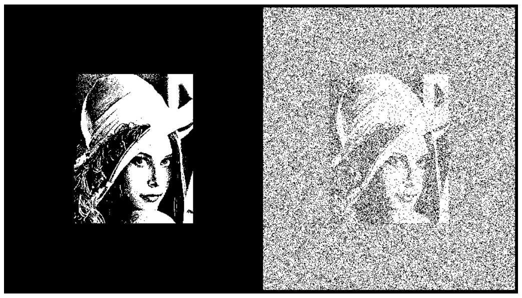


Figure 12: Grayscale Lena converted to pure black and white by thresholding (left), and the BBC codeword generated when that is used as the target image (right).

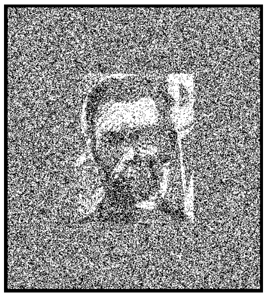


Figure 13: A superposition of Lena and Abraham Lincoln. An observer may be able to recognize that two faces have been superimposed, though it may be difficult to recognize the identity of at least one of the two faces.

For example, suppose a square target image is N by N pixels, and has an equal number of black and white pixels. The analysis in the previous section shows that the final image will have dark areas less than 3 times darker than the light areas. Now add a black border around all sides of the target image, with the border being N/2 pixels wide. The framed picture now has ¾ of its pixels in the frame, and only ¼ of its pixels in the actual image, with only half of those being white. So a full 7/8 of all the target pixels will be black. Now, when two possible bits are considered for the next bit of a message, the probability of them being in a pair of regions with colors (black, black), (black, white), (white, black) and (white, white) is 49/64, 7/64, 7/64, and 1/64, respectively. So the final codeword will have 49+7+7=63 black pixels in the dark region for every 1 black pixel in the light region. Since the dark region is 7 times the area of the white region, the resulting contrast ratio is 9:1, a marked improvement. Of course, this again assumes the image is very light overall. As it becomes darker, the rate at which new marks fall on top of existing marks will increase, particularly in the dark regions, and the contrast ratio will decrease. Figure 8 shows the result when a picture frame is used.

There is another tweak to the algorithm that can improve the contrast ratio. Recall that when adding an additional bit, it can be either a 0 or a 1, and that with probability 49/64, both of those will hash into the black region. In that case, the algorithm arbitrarily chooses which of the two to use. However, sometimes one of the marks will be in the black region of the frame, with the other one in the actual picture. In those cases, the algorithm could be biased to consistently choose the mark inside the picture. If that is done, the dark regions in the picture will end up darker than the frame, and the picture will look better than it would with arbitrary choices. Figure 9 shows the results of this tweak to the algorithm. Note that the dark regions of the picture are indeed darker than the frame. Figure 10 compares the three algorithms. It is clear that the final result is much better than the original.

In addition to illustrating BBC for students, these images can serve as motivation for performing a mathematical analysis of the behavior of BBC. This paper has analyzed the contrast ratio for 3 different situations. Students can be asked to perform that analysis themselves. This is a much easier analysis than having them analyze the security of the system. Also, the question itself is much more intuitive. After seeing the various images, a student will naturally wonder why some have better contrast ratio than others, and exactly how much better they are. This is the perfect motivation for having them perform such an analysis.

In the image processing community, it is considered important to use standard images so that algorithms can be compared. A famous standard image is known as Lena, and its colorful history has been documented in (Rosenberg, 2001) and (Kandangath, 2003). Though the original image was in color, it is often converted to grayscale (Figure 11); however, neither of these is suitable for the Visual BBC algorithm. To obtain a useable target image, the grayscale image was cropped to the same dimensions as the Lincoln picture and then every pixel above a certain threshold was set to black while all others were set to white. The threshold chosen yielded an image with approximately 50% fill. Figure 12 shows the target monochrome image and the resulting Visual BBC codeword. Figure 13 then shows the results of superimposing the Lena and Lincoln codewords, which is useful pedagogically because, while it is reasonably easy to see it as a superposition of two faces, it can be hard to discern whose faces they are.

Finally, Figure 14 shows the results of one last technique for improving the image. The algorithms so far have all been greedy: we construct a message by adding 1 bit at a time such that the new message has the best contrast ratio and density possible. It would seem that better results could be obtained by looking ahead more than 1 bit. For example, it could look ahead 3 bits at a time. In that case, the message would be extended at each step by appending one of eight patterns: 000, 001, 010, 011, 100, 101, 110, 111. The pattern would be chosen to maximize the contrast ratio and density of the image. In Figure 14, the top row adds only 1 bit at a time. Going down, the rows show the results of looking ahead 1, 2, 4, 8, or 16 bits. The image clearly improves with increased bits. Although the encoding time is exponential in the number of bits, the decoding time is not affected. The result is simply an ordinary BBC codeword, and so it is decoded in the usual way. Figure 14 also shows the results for various sizes of frames. The left column is for no frame at all. The columns from left to right are for frames whose areas are 0, 0.5, 1, 3, 5 times the area of the image, respectively. It is clear from Figure 14 that both look ahead and frames help the final image, and that combining the techniques gives an even better result than either alone.

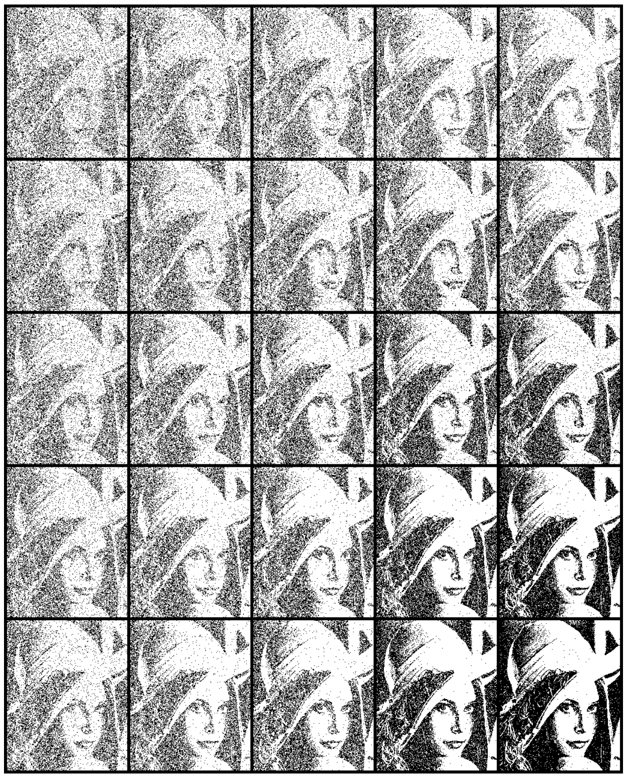


Figure 14: Lena images shown for various combinations of parameters (frame not shown). The rows from top to bottom correspond to looking ahead 1, 2, 4, 8, 16 bits. The columns from left to right use a frame-to-core-image-ratio of 0, 1/2, 1, 3, 5. The upper-left corner gives a contrast ratio of only 2.87 between the black and white regions. The contrast ratio in the lower left (no frame, 16 bit lookahead) increases to 11. The upper right (border is 5 times the area of the picture, 1 bit lookahead) increases to 17, and the lower right corner (frame is 5 times the area, 16 bit lookahead) increases to 450, which is practically perfect. In that image, 95.5% of the target-black pixels are black, and 99.8% of the target-white pixels are white.

As an interesting endnote, by looking ahead 20 bits at a time and using a target image whose border was 10 times the picture area, a codeword image was produced in which every single target black image was black and only 13 of the 26,000 target white pixels (0.05%) were black, yielding a contrast ratio of 2000:1. Producing this codeword required 15 hours of processing time on a fairly modern laptop computer but, as pointed out previously, the resulting codeword is a standard BBC codeword that would be decoded in the normal amount of time. From these results comes an interesting question for advanced students to explore: Is it possible, with reasonable processing effort, to find a codeword that exactly reproduces the core picture portion of the target image.

Conclusions

We have proposed a system for visual BBC, which is a visual form of BBC coding analogous to visual cryptography. Visual BBC has the same pedagogical advantages as visual cryptography: it gives the student a visual, hands-on way to explore the subject early on, before getting bogged down in mathematical detail. Several forms were given, allowing the final image to be remarkably clear. In addition, they motivate a number of analysis questions that can be posed to the students, so they can calculate and discover the contrast ratios in each case. Finally, the difficulty that humans have in separating superimposed pictures gives the students a feeling for the power of BBC, because it is capable of doing such separations easily.

Acknowledgements

This work was sponsored in part by the Air Force Information Operations Center (AFIOC), Lackland AFB, TX, and was performed at the Academy Center for Cyberspace Research (ACCR) at the United States Air Force Academy.

References

Baird, L (2000). Visual Cryptography Demonstration. Retrieved 15 Nov 2009 from http://leemon.com/crypto/VisualCrypto.html

Baird, L., Bahn, W. & Collins, M. (2007). Jam-Resistant Communication Without Shared Secrets Through the Use of Concurrent Codes, Technical Report, U. S. Air Force Academy, USAFA-TR-2007-01, Feb 14.

Kandangath, A. (2003). A Brief History of Lena, Retrieved 15 Nov 2009 from http://www.ecogito.net/articles/lena.html

Rosenberg, C. (2001). The Lenna Story, Retrieved 15 Nov 2009 from http://www.lenna.org

Ateniese, G., Blundo, C., De Santis, A., & Stinson, D. (1996). Visual cryptography for general access structures. *Information and Computation*, 129, issue 2:86–106.

Ateniese, G., Blundo, C., De Santis, A., & Stinson, D. (2001). Extended capabilities for visual cryptography. *Theoretical Computer Science*, 250(1–2):143–161.

Biehl, I. & Wetzel, S. (1997). Traceable visual cryptography. *Proc. of ICICS'97*, LNCS 1334, Springer-Verlag, pages 63–71.

Blundo, C., Bonis, A., & De Santis, A. (2001). Improved schemes for visual cryptography. *Designs, Codes and Cryptography*, 24(3):255–278.

Blundo, C., D'Arco, P., De Santis, A., & Stinson, D. (1999). On the contrast in visual cryptography schemes. *J. of Cryptology*, 12, issue 4:261–289.

Blundo, C., D'arco, P., De Santis, A., & Stinson, D. (2003). Contrast optimal threshold visual cryptography. *SIAM J. of Discrete Math.,* 16(2):224–261.

Blundo, C., De Santis, A., & Stinson, D. (1999), On the contrast in visual cryptography schemes, *J. Cryptol*., vol. 12, no. 4, pp. 261–289.

Blundo, C., De Santis, A., and Naor, M. (2001). Visual cryptography for gray-level images. *Information Processing Letters*, 75, issue 6:255–259.

Blundo, C.& De Santis, A. (1998), Visual cryptography schemes with perfect reconstructions of black pixels, *i*., vol. 22, no. 4, pp. 449–455.

Brickell, E.&Stinson, D. (1991), The detection of cheaters in threshold schemes, *SIAM J. Discrete Math*., vol. 4, no. 4, pp. 502–510.

Cimato, S., De Santis, A., Ferrara, A., & Masucci, B. (2005), Ideal contrast visual cryptography schemes with reversing, *Inf. Process. Lett*., vol. 93, no. 4, pp. 199–206.

De Bonis, A.& De Santis, A. (2004), Randomness in secret sharing and visual cryptography schemes, *Theoret. Comput. Sci*., vol. 314, no. 3, pp. 351–374.

Droste, S. (1996). New results on visual cryptography. *Advances in Cryptology-CRYPTO'96*, LNCS 1109, Springer-Verlag, pages 401–415.

Eisen, P. & Stinson, D. (2002). Threshold visual cryptography schemes with specified whiteness levels of reconstructed pixels. *Designs, Codes and Cryptography*, 25(1):15–61.

Hofmeister, T., Krause, M., & Simon, H. (2000), Contrast-optimal k out of n secret sharing schemes in visual cryptography, *Theoret. Comput. Sci.*, vol. 240, no. 2, pp. 471–485.

Horng,, G., Chen, T., & Tsai, D. (2006), Cheating in visual cryptography, *Designs, Codes, Cryptog.*, vol. 38, no. 2, pp. 219–236.

Hu, C. & Tzeng, W. (2005). Compatible ideal contrast visual cryptography with reversing, in *Proc. 8th Information Security Conf.*, vol. 3650, LNCS, pp. 300–313.

Ito, R., Kuwakado, H., & Tanaka, H. (1999). Image size invariant visual cryptography. *IEICE-Trans. Fundamentals*, E82–A(10):2172–2176.

Krause, M. & Simmon, H. (2000). Determining the optimal contrast for secret sharing schemes in visual cryptology. *LATIN'00*, LNCS 1776, Springer-Verlag, pages 280–291.

Krause, M. & Simon, H. (2003), Determining the optimal contrast for secret sharing schemes in visual cryptography, *Combin. Probab. Comput.*, vol. 12, no. 3, pp. 285–299.

Naor, M. & Pinkas, B. (1997), Visual authentication and identification, in *Proc. Advances in Cryptology*, vol. 1294, LNCS, pp. 322–336.

Naor, M. & Shamir, A. (1994). Visual cryptography, in *Proc. Advances in Cryptology*, vol. 950, LNCS, pp. 1–12.

Tzeng, W. & Hu, C. (2002). A new approach for visual cryptography. *Designs, Codes and Cryptography*, 27(3):207–227.

Viet, D. & Kurosawa, K. (2004). Almost ideal contrast visual cryptography with reversing, in *Proc, Topics in Cryptology*, vol. 2964, LNCS, pp. 353–365.

Yan, H., Gan, Z., & Chen, K. (2004), A cheater detectable visual cryptography scheme, (in Chinese) *J. Shanghai Jiaotong Univ.*, vol. 38, no. 1.

Authors

Leemon Baird is the Senior Research Scientist at the Academy Center for Cyberspace Research, a research center housed within the Department of Computer Science at the United States Air Force Academy. He has a long history of research in jam resistant coding theory, machine learning, computer security and cellular automata, and has been Professor of Computer Science at the Air Force Academy as well as co-founder and Chief Technical Officer for two successful startups. He received his PhD in Computer Science from Carnegie Mellon University in 1999.

Dino Schweitzer is the Director of the Academy Center for Cyberspace Research, a research center housed within the Department of Computer Science at the United States Air Force Academy. Dino has a long history in CS education, research, and military service, including many publications in the areas of computer graphics, visualization, and computer security education. He received his PhD in Computer Science from the University of Utah in 1983.

William L. Bahn received B.Sc. (Engineering Physics) and M.E. (Applied Mechanics) degrees from the Colorado School of Mines and is presently completing M.Sc. (Computer Science) and Ph.D. (Electrical Engineering) degrees at the University of Colorado at Colorado Springs. He has been affiliated with the Academy Center for Cyberspace Research since 2006, working in the area of keyless jam-resistance.

Dr. Samuel Sambasivam is the chairman of the Department of Computer Science of Azusa Pacific University. Dr. Samuel Sambasivam served as a Distinguished Visiting Professor of Computer Science at the United States Air Force Academy during the academic year 2008-09. Professor Sambasivam has done extensive research, publications, and presentations in both computer science and mathematics. His research interests include optimization methods, expert systems, fuzzy logic, client/server, Databases, and genetic algorithms. He has taught computer science and mathematics courses for over 27 years. Professor Sambasivam has run the regional Association for Computing Machinery (ACM) Programming Contest for six years. He has developed and introduced several new courses for computer science majors. Professor Sambasivam teaches Database Management Systems, Information Structures and Algorithm Design, Microcomputer Programming with C++, Discrete Structures, Client/Server Applications, Advanced Database Applications, Applied Artificial Intelligence, Java and others courses.