Master’s Defense Speech (4/8/16)

## I. Title Slide

Good morning everyone. For those that don’t know me, my name is Allison Holt and I am a graduate student defending my Master’s project. My project topic focuses on visual cryptography. Visual cryptography is “a cryptographic technique which allows visual information (pictures, text, etc.) to be encrypted in such a way that decryption becomes the job of the person … via sight reading” [Wikipedia]. When most people think of visual cryptography they tend to picture Moni Naor and Adi Shamir’s research. With their scheme a single black and white image is broken down into multiple images that resemble static. N images are created and K must be stacked in order for an image to be revealed. When doing my initial research on visual cryptography, I found very few tools available. The tools I came across used Naor and Shamir’s scheme, so the encoded images were obviously odd because the images were static. After researching different visual cryptography schemes, like size invariant, extended, and halftone, I figured out how I wanted to approach my Master’s project. As for the name of my visual cryptography tool, I sadly had a block and could not think of clever name.

## II. Problem Statement Slide

For my project I strove to develop a tool that, given a secret message and two innocent cover images, the tool would create two new encoded images. Those two new encoded images have the same meaning as the cover images they are bases on. In other words if the two innocent cover images are a picture of a cat and a picture of a dog, then the one of the encoded images will be a cat and the other a dog. Also the encoded images cannot reveal any of the secret information. All of that data should remain invisible to the user until the decryption process has occurred.

This tool must also reveal the secret when given two encoded images, i.e. the encoded images are combined to reveal the secret. The revealed secret does not have to match the initial secret entirely as long as the message is clear to the user.

## III. Solution (System Overview) Slide

My solution has a couple of other requirements not mentioned in the problem statement. First the inputted images must have the same dimensions. Thus, your secret message must be the same size as your cover images. Also, the tool I developed only handles JPEG and PNG images at the moment.

The diagram on the slide shows how the classes in my visual cryptography tool relate to one another in terms of general dependencies. The top three classes, all ending in “frame”, are the classes controlling my Graphical User Interface (GUI). The three frames contain the welcome screen, which give a short description of the tool and a few instructions, the encoding form, and the decoding form. Both forms have required fields and optional fields with default values if left blank. We will discuss this more during the demonstration.

The Custom Filters on the lower left were added so a user could only view acceptable choices when using the browse button. On the forms, a user can type the path to an image or directory or they can select the browse button beside the text field and use a file explorer to navigate to the desired image or directory. I added to filters to help with the usability and the robustness of the tool.

The last two classes in the system are Pixel and ExtendedVCS. The Pixel class is a small class that stores the red, green, and blue concentrations for a pixel. This class is used during the encryption process. The ExtendedVCS class does all the heavy work for the tool. It includes the methods for encoding and decoding as well as their helper methods.

## IV. Solution (Encryption) Slide Words

As I mentioned earlier there are several visual cryptography schemes. My tool uses the extended visual cryptography scheme since this scheme ensures the encrypted shares have a meaning and do not look like static. My encryption technique uses the VIP synchronization analyzed by L. M. Varalakshmi, Prithy R, and Radhika Parameswari. VIP stands for visual information pixel.

At the beginning of the encryption process, the cover images are sent through error diffusion. In my tool, the error diffusion implements Floyd-Steinberg dithering, and is used to help better blend the pixels into a continuous image. Then, the secret message gets broken into three shares. One containing the red concentrations of the secret, one containing the green concentrations of the secret, and the last containing the blue concentrations of the secret. Next, the VIP synchronization occurs and results in pixel expansion. After the bit manipulation, the alpha values are calculated. Alpha controls the transparency of the pixel. If the pixel held the cover pixel, the alpha was set to 255 (100% opaque). Otherwise, the three color concentrations were examined. The average of the absolute value of the original concentration minus the encoded concentration became the alpha value. Finally, the shares in the pixel expansion are randomized before getting stored in the array representing the encoded images, i.e. the order of the red, green, blue, and cover pixels can be in any order. Keep in mind the order will be the same for both encoded images. Before writing the pixels to a PNG file, the encoded images are sent through error diffusion. Again, this is to give the image a more smooth, continuous look.

## V. Solution (Encryption) Slide Pictures

This slide gives a better visual of how the encryption process works. In the upper left corner, you see one of the first steps. The secret message gets split into three “new” images holding only the red, green, or blue concentrations. The image in the lower right corner is a zoomed in view of one of the encode images. When the VIP synchronization occurs, the red, green, and blue concentrations are examined on a binary level. If the secret’s digit for that color is one and the cover digits in that same place are one, then the program randomly selects a cover pixel (either the first or second) and makes the second encoded pixel the complement of the first. For instance, if the first secret digit is one and both the first digits in the cover pixels are one, then, if the computer randomly selects the second cover image, then the second encoded image will have a one in the first digit while the first encoded image will have a zero in the first digit. Otherwise, if the secret digit is zero or if the cover digits are not equal, then one of the covers is randomly selected and the other cover gets set to be equal to it. For example, if the secret digit is zero, the first cover’s digit is one, and the second cover’s digit is zero, then, if cover two is selected, then both the encoded digits become zero. The alpha values are calculated for each pixel. These red, green, and blue shares are then randomly ordered and placed into the pixel expansion. Afterwards, the error diffusion helps smooth out some of the rough spots.

## VI. Solution (Decryption) Slide Words

When decrypting two encoded images, the pixels have their RGB values of the related pixels (from the pixel expansion) stored into an array. Then each related pixel for one image is analyzed to determine which pixel most likely holds the red, green, and blue information. In my code, I check to see absolute value of 127 minus the encoded color is less than 15. When a single color was stored in a pixel the other colors received 127 as there place holder. Since the Floyd-Steinberg dithering technique has a denominator of 16, I thought 15 was a good buffer. The red, green, and blue colors determined early are sent through a reverse VIP synchronization. The stored decoded red, green, and blue colors are converted to a color, and that color’s RGB value are stored in the array of the revealed secret. Finally, the revealed secret is sent through error diffusion. The next slide gives a little more of a visual to the decryption process.

## VII. Solution (Decryption) Slide Pictures

The two images on this slide have two images that have been encoded with a secret message. The four related pixels are analyzed to determine which pixels hold the red, green, and blue visual information. Once the visual information pixels have been determined, then the VIP synchronization is reversed. As shown in the code snippet, if the digits from the encoded pixels are the same, then the decoded pixel digit is one. Otherwise, the value for the decoded pixel is randomly selected to be zero or one. After the image has been decoded, the image gets sent through error diffusion before getting printed to a PNG file.

## XIII. Demonstrations Slide

Now that I have talked about how the tool works, let me demonstrate it for you. I have exported the source files to a JAR file to run my project.

In terminal:

cd \Volumes\AMHOLT

java –jar HCVT.jar

Files for demo:

secretMessage.png, innocent1.png, innocent3.png

## IX. Process Slide

The image on the screen is a sample of the process for one iteration. The process usually did not change between iterations. Each iteration lasted four weeks. The requirements analysis was performed during the first iteration. At the beginning of the iteration, I would update the project plan and the iteration plan based on the progress of the previous iteration. The iteration plan included all the tasks to be completed during that four week period. The design portion included any changes to the class relations and my planned strategies for tackling a particular iteration goal. The construction phase was where I implemented the strategies in an ad hoc manner. Testing and construction happened in a mini loop. Often I would implement a strategy, test it, and make some tweaks before testing it again. I also wanted to add JUnit tests for the methods in the Pixel and ExtendedVCS classes. Finally, I ended the iteration with the Post Mortem. During this part of the iteration, I would review my progress during the iteration and brainstormed solutions for any issues that appeared.

About midway through the project, I started planning how many hours a day I would allocate to working on my project. Balancing my other commitments with the project had become troublesome. Being able to keep track of the hours I had actually spent on the project verses how many I planned, helped keep me accountable.

## X. Iteration 0 Slide

Before I had fully settled on focusing my project on visual cryptography, I spent this past summer just messing around with two of the visual cryptography schemes. The size invariant scheme was like Naor and Shamir’s research. Given a single image, you broke the pixels up into N images where you needed at least K images to reveal the secret. The method I used only worked with black and white images. I also experimented with an extended scheme as well. The secret and cover images were also strictly black and white, i.e. not gray scale.

## XI. Iteration 1 Slide

Shortly after the fall semester started, my first iteration began. I used this iteration to set the groundwork for my entire project. Since I’ve been using Java heavily, I decided to use that language for my project. After doing some research I saw that NetBeans, an integrated development environment (IDE), would make it easier to design the GUI for my tool as well as develop the methods essential to perform the encryption and decryption. Thus, I created my project and set the necessary preferences for my project. I also created the three frames mentioned earlier for better human interaction and usability. Then, I ported the relevant code from my summer experimentations into the project. Finally, I tested that the GUI and the ExtendedVCS class worked with black and white images.

## XII. Iterations 2 & 3 Slide

During the second iteration, I researched halftone visual cryptography. Halftoning simulates a continuous tone through the use of dots, varying in either size or in spacing. I was hoping that halftone visual cryptography would give me some guidance on how to modify my tool so that it could handle gray scale images. Thus, I also searched for ways to combine the extended visual cryptography scheme with the halftone visual cryptography scheme.

During the third iteration, I implemented my strategy for processing gray scale images. I also added success message to the GUI user interaction. The images would either encrypt or decrypt and just jump back to the main window without notifying the user to a successful action. The problem with my gray scale encryption was the secret was visible in the encoded images. During the Post Mortem, I decided it would be more beneficial for me to continue forward and start looking into color images instead of continuing to fiddle with the gray scale images.

## XIII. Iterations 4 & 5 Slide

Iteration four fell on Winter Break, so I was not able to commit as much time to my project as I had in previous iterations. Thus, I focused on different strategies for handling colorful images with extended visual cryptography. By the end of the iteration I had settled on wanting to implement VIP synchronization.

During the fifth iteration, I applied the Floyd-Steinberg dithering as my error diffusion. Plus, I implemented the VIP synchronization without pixel expansion. I was hoping to keep that “size invariant” quality of my encryption technique. However, the encoded images did not have the same meaning as the covers they inherited from. In other words, if the cover image was a flower, the encoded image resembled static and revealed portions of the secret message. Also, when decrypted, the secret was not revealed.

## XIV. Iterations 6 & 7 Slide

Thus, during Iteration 6, I added the pixel expansion to the VIP synchronization. Once the pixel expansion was finalized, I also modified the process so the placement of the related pixels would be random in order to increase the security of the image. Towards the end of the iteration, testing revealed the encoded images appeared really dark. Since the unspecified colors were set to have concentrations of 0, the images were dark. Thus, I set those concentrations to be 128, though in iteration seven I switched it to 127. Another problem was the decrypted secrets were not very clear.

During my final iteration, I fixed the problem with the decryption. Initially, I wanted the user to be able to print the encoded images on transparencies and decrypt them by stacking transparencies if necessary. After dropping that constraint, I ended up just reversing the encryption process as mentioned earlier. This greatly improved the visibility of the secret after decryption. During the middle of the iteration, I went through and added more robustness to the tool with three things. First, all of the required fields had to have data in them before the encode or decode button would enable. Second, I fixed the code so the default directory for the tool should work. It stored the shares on the user’s desktop. The third fix involved putting more details in the success message, and that was the names of the images and the folder in which they were saved. The last change to the tool revolved around using the alpha property of a pixel. The alpha allowed the secret information to better blend with the cover pixels in the encoded images. Plus, the alpha value has no impact on the decryption process.

## XV. Lessons Learned Slide

Upon wrapping up the project, I reflected on some of the lessons learned. In terms of process, it is vital to give hard deadlines for your construction. Otherwise, you end up having less time testing your code. I, also, should have researched techniques on how to test images. Adding in the weekly planning had a significant impact on helping me balance my time with my project workload.

In general, if I could go back and do this over with what I know now, I think I would have started working with the colorful images from the get-go instead of baby-stepping (black and white to gray scale to color). Then I would have had more time to work on improving the efficiency of my tool. Also, documenting the methods and tests as you make progress helps the iteration write-ups. It’s easier to look back at well documented code and discuss what strategies worked compared to scrolling through trying to figure out what you thinking three weeks ago. Fortunately, I figured that out early in the project. Another helpful tool for tracking changes to the code was GitHub. Writing detailed commit messages was also very helpful.

## XVI. Questions Slide

This concludes my presentation. Thank you so much for your time. Does anyone have any questions?

## XV. Sources (Footnotes)

[1] Varalakshmi, L., R, P. and Parameswari, R. 2013. Extended Visual Cryptography for Color Images and its PSNR Analysis. International Journal of Computer Applications. 67, 17 (2013), 17-22.