Computer Science 9544a
Visual Cryptography

December 9, 2009

Student: Jeffrey Shantz <jshantz(5-1)@csd.uwo.ca>
Instructor: Roberto Solis-Oba

Contents

1 Introduction

Visual cryptography (VC) is a novel concept first introduced by Naor and Shamir [1] in 1994, allowing one or more images to be encrypted and subsequently decrypted. However, it is distinguished from traditional cryptographic techniques in that the decryption of an image encrypted by a visual cryptography scheme requires no mathematical computations or knowledge of cryptography. Instead, the original image becomes visible to the naked eye simply by overlaying cipher transparencies – known as *shares* – created during the encryption process.

Naor and Shamir [1] establish visual cryptography as a visual variant of the k out of n secret sharing problem. In a secret sharing scheme, one wishes to randomly divide a secret amongst a group of n individuals in such a way as to allow any $k < n$ of them (or, in certain cases, only a qualified subset of them), to recover the secret from their individual shares. However, any number of individuals $k' < k$ should be prevented from obtaining any information about the original secret by combining their individual shares [2].

Similarly, in a VC scheme, shares are generated from the original image according to rules of the scheme in such a way as to provide no information about the encrypted image individually, but to produce a reasonable depiction of the original image when superimposed. While the original image should be visible after overlaying its shares, visual cryptography schemes are typically lossy and produce decrypted images that are often noisy or suffer from diminished contrast and resolution.

A number of factors can affect the quality of the resulting decrypted image in a VC scheme. Typically, as the number of shares n is increased, the contrast of the resulting decrypted image worsens. Furthermore, many schemes produce shares in which each pixel of the original image is represented by multiple pixels in each share, diminishing the resolution of the decrypted image. This paper presents algorithms that both do and do not require pixel expansion to highlight the differences between the two.

It can be tempting to think of visual cryptography as a form of *steganography*, but it is important to understand the distinction between the two. In steganography, one seeks to conceal the *existence* of a message, perhaps by composing the message using invisible ink. By contrast, visual cryptography – like its true cryptographic counterparts – seeks only to conceal the message itself. It is, however, possible to combine steganography and visual cryptography to produce two benign-looking images that, when superimposed, reveal a third hidden image [3].

This paper presents several visual cryptography algorithms along with a time complexity analysis and proof of correctness for each algorithm. The rest of this paper is organized as follows: section 2 describes the original 2 out of 2 algorithm presented by Naor and Shamir [1] to demonstrate the basic concepts of visual cryptography. Section 3 discusses several random grid algorithms due to Kafri and Keren [4] and Chen and Tsao [5], which present a form of visual cryptography in which the resolution of the original image is preserved during the encryption and decryption process.

Section 4 presents related work in the area of visual cryptography, while section 5 discusses the details of the software implemented for this project, which implements the algorithms discussed in this paper. Finally, section 6 contains a user manual detailing the use of this software.

2 2 out 2 Algorithm

The simplest VC algorithm was given by Naor and Shamir [1] in their introductory paper on visual cryptography. They presented a 2 out of 2 scheme, in which 2 shares would be generated ($n = 2$) for each image encrypted, while decryption would require these 2 shares ($k = 2$) to be super-imposed. At its most basic level, the 2 out of 2 algorithm works by representing each pixel in the original image by 2 pixels in each share. Each pixel in the original image is read and, if a white pixel is encountered, one of the first two rows in Figure 1 is selected with equal probability, and each share is assigned a 2 pixel block as shown in the third and fourth columns. Similarly, if a black pixel is encountered, one of the last two rows is selected with equal probability, from which a subpixel is assigned to each share.

If two white pixels overlap when two shares are superimposed, the resulting pixel will be white. By contrast, if a black pixel in one share overlaps with either a white or black pixel in the other share, the resulting pixel will be black. This implies that the superimposition of the shares represents the Boolean OR function. The last column in Figure 1 shows the resulting subpixel when the subpixels of both shares in the third and fourth columns are superimposed.

As demonstrated in Figure 1, if a pixel in the original image was black, the subpixel in the superimposition of the two shares will be fully black. Similarly, if a pixel in the original image was white, the subpixel in the superimposition of the two shares will be black and white. However, because the pixels are small and situated very close together, the human eye averages the relative contributions of the black and white pixels, resulting in a grey pixel.

Original Pixel	Probability	Share 1 Sub-Pixel	Share 2 Sub-Pixel	Share 1 || Share 2
□	0.5	■□	■□	■□
□	0.5	□■	□■	□■
■	0.5	■□	□■	■■
■	0.5	□■	■□	■■

Figure 1: 2 out of 2 using 2 subpixels per original pixel

Figure 2 shows the encryption and decryption of the University of Western Ontario logo using Naor and Shamir's 2 out of 2 algorithm, in which 2 subpixels are used for each original pixel. Neither share generated reveals any information about the original image, but when the two are superimposed as shown in Figure 2(d), a representation of the original image can be seen. The aspect ratio of the original image is distorted in the decrypted version due to the fact that the use of 2 subpixels per original pixel doubles the width of the decrypted image while retaining its original height.

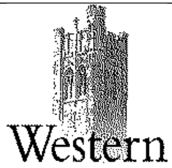
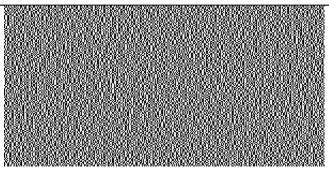
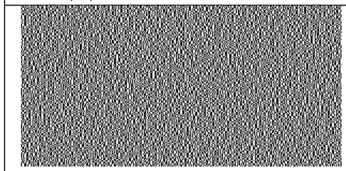
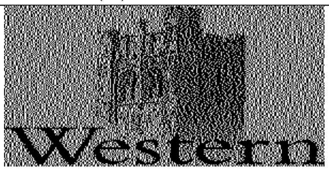
	
(a) Original Image	(b) Share 1
	
(c) Share 2	(d) Shares Super-Imposed

Figure 2: 2 out of 2 encryption/decryption using 2 subpixels per original pixel

To compensate for the distortion of the aspect ratio of the original image, Naor and Shamir [1] recommend using a 2 × 2 subpixel block to represent each original pixel. This produces an image that is four times the size of the original image, but retains the aspect ratio of the original image. Figure 3 shows the subpixels used in this new variant of the 2 out of 2 algorithm.

Original Pixel	Probability	Share 1 Sub-Pixel	Share 2 Sub-Pixel	Share 1 || Share 2
□	0.5	■□ □■	■□ □■	■□ □■
□	0.5	□■ ■□	□■ ■□	□■ ■□
■	0.5	■□ □■	□■ ■□	■■ ■■
■	0.5	□■ ■□	■□ □■	■■ ■■

Figure 3: 2 out of 2 using 4 subpixels per original pixel

Figure 4 shows an encryption and decryption cycle on the same image used in Figure 2, this time using the 4 subpixel variant of the 2 out of 2 algorithm. It is clear that while the image is four times as large as the original, its original aspect ratio has been preserved, producing a clearer and more natural looking result.

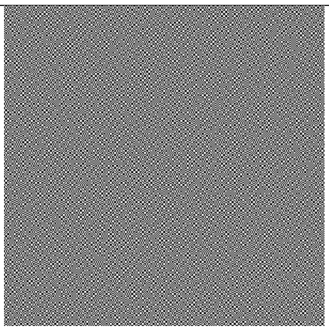
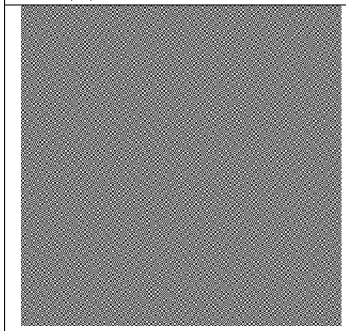
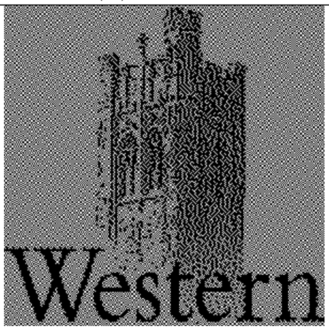
	
(a) Original Image	(b) Share 1
	
(c) Share 2	(d) Shares Super-Imposed

Figure 4: 2 out of 2 encryption/decryption using 4 subpixels per original pixel

2.1 Analysis of Naor and Shamir's 2 out of 2 Algorithm

Let w be the width of the original image, and h be its height. Then, $n = w \times h$ is the number of pixels in the original image. To encrypt an image using the 2 out of 2 algorithm, each pixel in the original image must be read, and a block of m subpixels must then be written to each share. Thus, for each pixel in the original image, $2m$ subpixels are written, and each share contains $n \times m$ pixels. As such, a total of $2(n \times m)$ pixels are written in the encryption process. As long as $m < n$, we thus have a linear time complexity of $O(n)$ for the encryption algorithm.

When superimposition is done by a computer, each subpixel in each share must be read sequentially, computing the Boolean OR of the subpixels from each share as they are read. This computation requires O(m) time, and there are n such computations. Once again, as long as $m < n$, decryption takes place in linear time.

2.2 Correctness of Naor and Shamir's 2 out of 2 Algorithm

Let W represent a white pixel and B represent a black pixel. An examination of the subpixels used in Figure 3 reveals that only two types of subpixels exist: $BWWB$ and $WBBW$. Suppose that we encrypt an image and distribute one of its shares to Alice and the other to Bob. However, Eve steals Alice's share and wishes to decrypt the original image from this share. She examines the

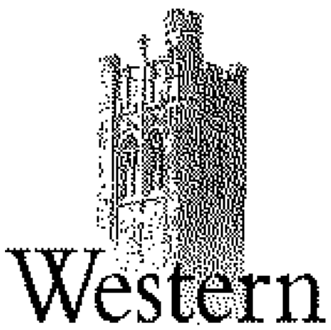


Figure 5: Result of the decrypted image contrast repair algorithm

first pixel and finds that it is $BWWB$. A cursory glance at the *Share 1* column of Figure 3 shows that if the subpixel $BWWB$ is found, we have $P(W) = \frac{1}{2}$ and $P(B) = \frac{1}{2}$. Thus, the original pixel could be white or black with equal probability. The analysis is symmetric for $WBBW$. As such, Eve can obtain no information from a given pixel in the image. She can only guess a black or white value for each pixel and hope to be right. However, since there are n such pixels, the probability of guessing all pixels correctly is $\left(\frac{1}{2}\right)^n$. Given that a standard image size is $800 \times 600 = 480,000$ pixels, her odds of doing so are approximately $\frac{1}{2.49 \times 10^{198}}$. The analysis is symmetric if Eve should steal Bob's share instead.

The correctness of the decryption routine relies on the correctness of the Boolean OR function and the ability of the human eye to average relative contributions made by neighbouring colours. For each white pixel in the original image, a $BWWB$ or $WBBW$ is written to each share with equal probability. When two identical pixels from each share are superimposed, the resulting subpixel will not change, and will be 50% black. The human eye therefore averages this as a grey pixel. Similarly, for each black pixel in the original image, complementary pixels are written to each share. When these complementary subpixels are superimposed, the resulting pixel will be 100% black, resulting in a black pixel in the decrypted image.

2.3 Repairing Decrypted Image Contrast

Given the discussion at the end of the previous section, the author realized that it is possible to completely repair the contrast of a decrypted image using a very simple algorithm. The algorithm works by scanning each subpixel in the decrypted image. If a subpixel is found to be 100% black, then it should remain a black pixel. However, if it is found to be 50% white, then it should be written as a white pixel. Figure 5 shows the result of applying this contrast repair algorithm to the image decrypted in Figure 4, demonstrating that its contrast matches that of the original image.

3 Visual Cryptography using Random Grids

While the approach by Naor and Shamir [1] offers perfect security when one possesses only a single share, it suffers from the need to represent each pixel in the original image by multiple pixels in each share, resulting in a decrypted image that is 2–4 times larger than the original image (depending on the number of subpixels used). As one can see in Figure 5, this produces an image with poorer resolution and clarity than the original. Furthermore, additional time is required to encrypt and decrypt images – as well as to transfer encrypted images across a network – than would be required in a scheme that did not require the use of pixel expansion.

Such a scheme was proposed by Kafri and Keren [4] through the use of *random grids* (RG). Like the 2 out of 2 algorithm presented in section 2, an RG scheme takes an input image and transforms it into multiple *cipher-grids* that provide no information on the original image. However, RG schemes have the additional benefit that they require no pixel expansion, and thus each share – along with the resulting decrypted image – retains the size of the original image. The following sections detail two algorithms employing random grids.

3.1 2 out of 2 using Random Grids

Kafri and Keren [4] proposed three similar 2 out of 2 algorithms employing random grids in 1987[1]. For brevity, only the first of these algorithms is discussed here, but all three were implemented in the software developed for this project. Qualitative testing revealed that the first algorithm they present produces results superior to those produced by the others, and thus this algorithm was selected for discussion. A pseudo-code listing of this algorithm is presented in Figure 6.

The algorithm takes an input image of size $height \times width$. It then initializes two cipher-grid images R_1 and R_2 with the same dimensions as the input image. In lines 6 - 8, the algorithm randomizes the contents of R_1, producing an image of random black and white pixels. R_2 is next generated based on the input image and R_1 in lines 11 - 15. This process occurs by scanning each pixel of the input image. If a pixel at location $[x, y]$ in the input image is found to be white, then the pixel $R_2[x, y]$ is set to be the same as $R_1[x, y]$. If, instead, the pixel at $[x, y]$ in the input image is black, then the pixel $R_2[x, y]$ is set to be the complement of $R_1[x, y]$.

Figure 7 shows the possibilities for the pixels written to each share based on a given pixel in the input image. As the table shows, if a white pixel is encountered in the input image and a white pixel is randomly selected for R_1 (with probability 0.5), a white pixel will also be written to R_2. If, instead, a black pixel is selected for R_1, then a black pixel will be written to R_2. Thus, when the shares are overlaid, the share pixels generated from a white input image pixel will be correct only 50% of the time. This also implies that, on average, 50% of the white pixels in the input image

[1]Curiously, an explanation could not be found as to why Naor and Shamir are credited with introducing visual cryptography in 1994 when such techniques were presented 7 years earlier by Kafri and Keren.

will appear unaltered in each share. While this may pose a security problem for images with a high proportion of white pixels, this should not cause a problem in images where the ratio of white to black pixels is relatively even.

```
KAFRI-KEREN-ALGORITHM-1($I$)
  ▷ Input   : Input image $I$ of size $height \times width$
  ▷ Output  : Two cipher-grids $R_1, R_2$, both of size $height \times width$

▷ Setup constants
WHITE ← 0
BLACK ← 1

▷ Randomize the first cipher-grid $R_1$
for $row \leftarrow 1$ to $height$
    do for $col \leftarrow 1$ to $width$
        do $R_1[row, col] = \text{RANDOM}(\text{WHITE}, \text{BLACK})$

▷ Create the second cipher-grid $R_2$ based on $I$ and $R_1$
for $row \leftarrow 1$ to $height$
    do for $col \leftarrow 1$ to $width$
        do if $I[row, col] = \text{WHITE}$
            then $R_2[row, col] = R_1[row, col]$
            else $R_2[row, col] = 1 - R_1[row, col]$

return $R_1, R_2$
```

Figure 6: Random Grid algorithm 1 proposed by Kafri and Keren [4]

$I[x, y]$	$R_1[x, y]$	$P(R_1[x, y])$	$R_2[x, y]$	$R_1[x, y] \vert\vert R_2[x, y]$
□	□	0.5	□	□
□	■	0.5	■	■
■	□	0.5	■	■
■	■	0.5	□	■

Figure 7: Pixel table for Kafri and Keren's first random grid algorithm

Examining Figure 7 for black pixels in the input image, one can see that, regardless of the pixel selected for R_1 – white or black, with equal probability – the resulting pixel when both shares are superimposed will be black. This is because, if a black pixel is selected for R_1, the resulting superimposed pixel will be black regardless of the pixel selected for R_2 due to the properties of the Boolean OR operation. By contrast, if a white pixel is selected for R_1, then the resulting superimposed pixel will also be black, since R_2 is always chosen as the complement of R_1 and will thus be assigned a black pixel. Thus, all black pixels in the input image will be black in the decrypted image.

Since, on average, 50% of the white pixels in the original image will appear as black in the decrypted image, while all black input pixels will appear correctly, one can expect an average of 75% of the original input pixels to appear in the decrypted image, assuming a relatively even ratio of white to black pixels in the input image. Figure 8 shows the result of encrypting and decrypting the University of Western Ontario logo using Kafri and Keren's first random grid algorithm.

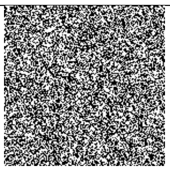
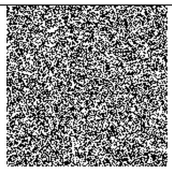
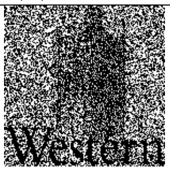
	
(a) Original Image	(b) Share 1
	
(c) Share 2	(d) Shares Super-Imposed

Figure 8: 2 out of 2 encryption/decryption using Kafri and Keren's first random grid algorithm

3.1.1 Analysis of Kafri and Keren's First Random Grid Algorithm

As before, let $n = width \times height$ be the number of pixels in the original image. To encrypt an image using this random grid method, the cipher-grid R_1 of size n must first be created by randomly assigning an integer value in the range $[0, 1]$ to each of its pixels. Since 0 and 1 can be represented in 1 bit, generating a random value requires constant time. Thus, R_1 is generated in $O(n)$ time.

To generate R_2, each pixel of the original image must be read, and a constant time comparison decides the next pixel value to be written to R_2. As such, the creation of R_2 also takes place in linear time. Thus, the encryption algorithm runs in time linear in the number of pixels in the input image.

The decryption algorithm for an image generated by this random grid method is identical to that of Naor and Shamir's 2 of 2 algorithm, and is therefore linear as well.

3.1.2 Correctness of Kafri and Keren's First Random Grid Algorithm

Let W represent a white pixel and B represent a black pixel. Suppose again that we encrypt an image using Kafri and Keren's algorithm and distribute one of its shares to Alice and the other to

Bob. Eve knows that the Alice's share is generated randomly and thus will provide no information about the original image. Instead, she decides to steal Bob's share and hopes to decrypt the original image from this share. She examines the first pixel and finds that it is B; that is, $R_2[x, y] = B$ Looking at Figure 7, we see that given a black pixel in the second share, the original pixel could be black or white with equal probability. That is, $P(I[x, y] = B|R_2[x, y] = B) = P(I[x, y] = W|R_2[x, y] = B) = 0.5$. Thus, Eve can obtain no information about the original image from this pixel. The argument is symmetric when Eve encounters a white pixel.

As previously indicated, the decryption of an image generated by this random grid method is identical to that of Naor and Shamir's 2 of 2 algorithm. See section 2.2 for a discussion of its correctness.

3.2 n out of n using Random Grids

The final algorithm presented in this paper is due to Chen and Tsao [5]. They proposed an n out of n random grid algorithm based on Kafri and Keren's algorithms [4], in which any number of shares can be generated for a given input image. To decrypt the image, all shares must be superimposed. A pseudo-code listing of the algorithm is presented in Figure 9.

```
CHEN-TSAO-ALGORITHM-4($I, n$)
 ▷ Input   : Input image $I$ of size $height \times width$; Number of shares $n$ to generate
 ▷ Output  : $n$ cipher-grids $R_1, R_2, \ldots, R_n$, all of size $height \times width$

1  ▷ Generate the starting grids
2  $R_1, \Re_2 \leftarrow$ KAFRI-KEREN-ALGORITHM-1($I$)
3
4  if $n > 2$
5     then for $i \leftarrow 2$ to $n - 1$
6              do $R_i, \Re_{i+1} \leftarrow$ KAFRI-KEREN-ALGORITHM-1($\Re_i$)
7  $R_n \leftarrow \Re_n$
8  return $R_1 \ldots R_n$
```

Figure 9: n out of n random grid algorithm proposed by Chen and Tsao [5]

The algorithm works by creating a chain of cipher-grids, where each successive cipher-grid can decrypt the previous. Thus, when all cipher-grids are superimposed, the original image will be visible. It starts by creating R_1 and $\Re_2$ from the original input image using the KAFRI-KEREN-ALGORITHM-1 algorithm presented earlier. R_1 will be generated randomly, while $\Re_2$ will be generated based on the pixels of R_1 and I.

Subsequent shares are generated in lines 5 - 7. This process occurs by again executing KAFRI-KEREN-ALGORITHM-1, this time passing in the last $\Re_i$ generated. R_i will then generated by the algorithm randomly, while $\Re_{i+1}$ will be generated based on the pixels of R_i and $\Re_i$. This will repeat n - 2 times, after which R_n is simply assigned the value of $\Re_n$.

An interesting phenomenon appears in this algorithm. Observe that when the algorithm terminates, the only cipher-grid that is not fully randomized will be R_n. This is because each cipher-grid from 2 to $n-1$ will be assigned the first cipher-grid returned by Kafri-Keren-Algorithm-1, which is always randomly generated. To understand how this works, suppose that $n = 3$. R_1 is generated randomly, with $\Re_2$ generated from the pixels of R_1 and I. $\Re_2$ is then passed to Kafri-Keren-Algorithm-1, generating R_2 randomly, and $\Re_3$ from the pixels of R_2 and $\Re_2$. Finally, R_3 is set as $\Re_3$. This series of assignments is illustrated in Figure 10.

Step	Share	Value Assigned
1	R_1	Random
2	$\Re_2$	$R_1 \oplus I$
3	R_2	Random
4	$\Re_3$	$R_2 \oplus \Re_2$
5	R_3	$\Re_3$

Figure 10: Encryption steps in the n out of n random grid algorithm (n=3)

To see how the decryption process takes place, consider that when R_3 and R_2 are superimposed, they will decrypt $\Re_2$ just as in the last section discussing Kafri-Keren-Algorithm-1. Next, when R_1 and $\Re_2$ are superimposed (that is, R_1 is stacked on top of R_2 and R_3), they will decrypt the original image I. This series of decryption steps is illustrated in Figure 11.

Step	Share 1	Share 2	Share 1 || Share 2
1	$R_3 = \Re_3$	R_2	$\Re_2 = R_1 \oplus I$
2	$\Re_2$	R_1	I

Figure 11: Decryption steps in the n out of n random grid algorithm (n=3)

Figure 12 shows an encryption and decryption of the University of Western Ontario logo using 3 shares. Observe that superimposing any strict subset of the 3 shares does not produce the original image. However, when all 3 shares are superimposed, the original image becomes visible. Of course, the contrast of the resulting image is quite poor, but the silhouette of the tower can be made out, along with the text contained in the image.

3.2.1 Analysis of Chen and Tsao's n out of n Random Grid Algorithm

Let $p = width \times height$ be the number of pixels in the original image. To encrypt an image using this random grid method, the algorithm first executes Kafri-Keren-Algorithm-1 to generate R_1 and $\Re_2$. As seen in section 3.1.1, this requires $O(p)$ time.

Next, assuming $n > 2$, the loop in lines 4 - 6 runs $n - 2$ times. On each iteration of the loop,

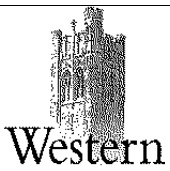
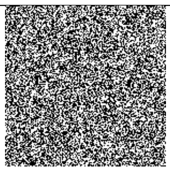
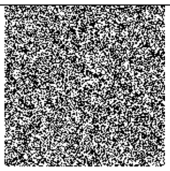
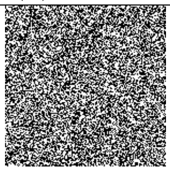
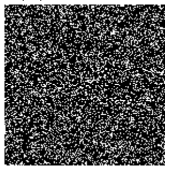
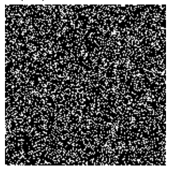
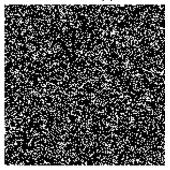
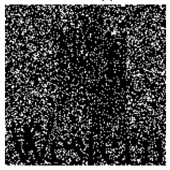
	
(a) Original Image	(b) Share 1
	
(c) Share 2 	(d) Share 3 
(e) Share 1 || Share 2 	(f) Share 1 || Share 3 
(g) Share 2 || Share 3	(h) Share 1 || Share 2 || Share 3

Figure 12: n out of n encryption/decryption using $n = 3$

Kafri-Keren-Algorithm-1 is executed once, requiring $O(p)$ time. Thus, the loop runs in $O(np)$ time. Since the number of shares n is much less than the number of pixels p in the input image, this can be viewed as time linear in p.

As seen in section 2.2, decrypting two shares requires $O(p)$ time. Since this decryption process must occur $n - 1$ times, the decryption process for the n out of n algorithm requires $O(np)$ time, which can again be viewed as being linear in the number of pixels p.

3.2.2 Correctness of Chen and Tsao's n out of n Random Grid Algorithm

Suppose that we encrypt an image using Chen and Tsao's algorithm to produce 3 shares. We then distribute these shares to Alice, Bob, and Victor. If Eve steals the shares possessed by Alice and Bob (R_1 and R_2) and superimposes them, she will not obtain any information about the original image since R_1 and R_2 were both randomly generated, and thus $R_1||R_2$ will also be random. If, instead, she steals the shares possessed by Alice and Victor (R_1 and R_3) and superimposes them, she will obtain the following:

$$\begin{aligned}
&R_1||R_3\\
&= R_1||\Re_3\\
&= R_1||R_2 \oplus \Re_2\\
&= R_1||R_2 \oplus (R_1 \oplus I)
\end{aligned}$$

However, Eve does not have R_2, and thus cannot decrypt $R_2 \oplus (R_1 \oplus I)$, so she does not obtain any information about the original image. Similarly, if Eve steals the shares possessed by Bob and Victor (R_2 and R_3) and superimposes them, she will obtain the following:

$$\begin{aligned}
&R_2||R_3\\
&= R_2||\Re_3\\
&= R_2||R_2 \oplus \Re_2\\
&= R_2||R_2 \oplus (R_1 \oplus I)
\end{aligned}$$

However, Eve does not have R_1, and thus cannot decrypt $(R_1 \oplus I)$, so she does not obtain any information about the original image. Only when Eve obtains all three shares does she obtain the original image, as seen below:

$$\begin{aligned}
&R_1||R_2||R_3\\
&= R_1||R_2||\Re_3\\
&= R_1||R_2||R_2 \oplus \Re_2\\
&= R_1||R_2||R_2 \oplus (R_1 \oplus I)\\
&= R_1||R_2 \oplus (R_1 \oplus I)\\
&= R_1||(R_1 \oplus I)\\
&= I
\end{aligned}$$

This argument extends for general n, since all $R_1 \dots R_{n-1}$ will be randomly generated when the algorithm terminates, while R_n will require all $n-1$ shares to be superimposed upon it in order to decrypt the original image, as seen above.

4 Related Work

Visual cryptography is currently an active field, having enjoyed the introduction of a number of new algorithms and techniques in recent years. Although it may seem like a *toy problem*, visual cryptography schemes have been proposed as solutions to real-world problems, such as the confidential electronic transmission of financial documents [6], and the digital watermarking of images to protect copyrighted material [7]. The following sections outline a number of other novel visual cryptography techniques proposed in the literature.

4.1 Visual Cryptography for an Access Structure

Ateniese et al. [8] present an extended visual cryptography scheme (EVCS) capable of generating n shares from an input image in such as a way as to allow only certain subsets of recipients to decrypt the original image by combining their shares. To see the utility of such a scheme, suppose that we wish to hide an image containing nuclear launch codes in a set of 3 shares. We distribute a share to each of Alice, Bob, and Victor. In a time of emergency, all three members should be able to combine their shares to obtain the launch codes. Because Alice is an extremely trusted member of the team, we wish to allow her to decrypt the launch codes by combining her share with either Bob's share or Victor's share. This way, if either Bob or Victor is absent in the event of an emergency, the codes can still be obtained. However, we suspect that Bob and Victor might have malicious motives and do not wish to allow them to decrypt the launch codes simply by combining their shares.

The algorithm proposed by Ateniese et al. [8] defines the following sets:

- $P = \{1, 2, \ldots, n\}$, the shares to be generated
- 2^P, the powerset of P.
- $\Gamma_{Qual} = \{X | X \in 2^P, X$ is a set of shares that decrypt the original image when combined$\}$
- $\Gamma_{Forb} = 2^P - \Gamma_{Qual}$

Thus, Γ_{Forb} is the set of all subsets of shares that will not decrypt the original image when combined. In the scenario discussed above, we set $\Gamma_{Qual} = \{\{1,2\},\{1,3\},\{1,2,3\}\}$, while $\Gamma_{Forb} = \{\{1\},\{2\},\{3\},\{2,3\}\}$. The algorithm then generates the shares in such a way that if a set X of shares is superimposed, then the original image will be decrypted if and only if $X \in \Gamma_{Qual}$.

4.2 Hiding Multiple Secrets

Most visual cryptography schemes presented to date allow only one image to be encrypted across a number of shares. Fang [9] presents an innovative reversible visual cryptography algorithm for hiding two images within two shares. The method starts by reading the pixel $[1,1]$ and $[1, width]$ (that is, the the top left and top right pixels) from each input image. Based on these pixels, the top left and right pixels for the two shares are generated in such a way that when the two shares are stacked, the top left and right pixels of the first original image are decrypted. However, when the second share is flipped horizontally, the top left and right pixels of the second original image are decrypted. This process repeats for each pixel in the input images, moving inward, with each pair of pixels on the left and right sides of each input image used to generate corresponding subpixels in the generated shares.

This method makes use of a 3×3 subpixel block for each pixel in the original images. As a result, it results in pixel expansion of the decrypted images. By contrast, Fang [10] presents a new algorithm that applies the random grid method discussed earlier to allow reversible encryption without pixel expansion. While this technique is quite innovative in its ability to store a great deal of information in only two shares, images decrypted using this method typically exhibit poor contrast and a high level of noise.

In a related technique, Wu and Chang [11] present an algorithm in which 2 images are encrypted over 2 circular shares. When the shares are superimposed, the first image is revealed. By rotating one of the shares by a specified number of degrees, the first image disappears and the second image is then revealed.

5 Implementation

The application written for this project was designed in the Ruby programming language. As a command-line tool, the application provides the ability to encrypt and decrypt black and white images stored in the Portable Bitmap (PBM) format. The following algorithms are implemented:

- 2 out of 2 with (2x2)-Pixel Expansion (Naor and Shamir [1])
- 2 out of 2 with random grids, algorithms 1–3 (Kafri and Keren [4])
- n out of n with random grids, algorithm 4 (Chen and Tsao [5])
- 2 out of n with random grids, algorithm 6 (Chen and Tsao [5])

The two reversible encryption algorithms discussed earlier due to Fang [9, 10] were also partially implemented but later removed due to time constraints.

5.1 PBM Library

The PBM file format was selected for use in this project as it is a simple format, which is easy to parse. While a Ruby library to manipulate PBM files was sought, a custom library ultimately had to be designed, as it appears that no such library yet exists in Ruby. Thus, a PBM library was implemented allowing PBM images to be read, written, and manipulated.

The PBM format is actually an umbrella format for two different subformats – *raw* and *plain*. In raw format, 8 pixels are encoded in each byte of the file (since a pixel can be encoded in 1 bit).

By contrast, the plain format stores each pixel as an ASCII 1 or 0, allowing it to be read by the human eye. In order to be as flexible as possible, the library was designed to read and write in both formats.

Furthermore, because efficiency was a priority in designing the library, PBM files can be read and written in one of two ways. If one so desires, an entire image can be read into a memory-resident matrix, allowing random access to all pixels of the image. In the same way, the entire matrix can then be written to disk when needed.

However, because storing an entire image can be resource-intensive as image sizes increase, the library also supports the ability to read and write single pixels sequentially. This is useful, for instance, when decrypting two shares. In this process, a pixel is read from one share, and the corresponding pixel is read from the second. The Boolean OR of the two pixel intensities is then computed, and this resulting pixel is written to the output file. In this way, only 3 pixels need ever be resident in memory at a given time.

5.2 `vcrypt`

The command-line tool implemented for this project is known as `vcrypt`. By passing a set of options from the command line to the program, one can encrypt or decrypt a set of images with the desired algorithm. `vcrypt` attempts to be as user-friendly as possible, requiring little information from the user. For instance, when encrypting images, one need not specify the filename of each share to produce. Instead, a default share filename prefix is used which depends on the algorithm being employed (and can be changed using a command line option, if desired), and each share is given a filename having the format `PREFIXi.pbm`, where $i = 1 \ldots n$ is the share number. Similarly, when decrypting an image, only the algorithm to use and an output filename need be specified. The program will then search for files of the form `PREFIXi.pbm`, and will decrypt these. For example, if one specifies that the Naor and Shamir algorithm should be used for decryption, the program will search for the shares `ns1.pbm` and `ns2.pbm`, since `ns` is the default prefix for this algorithm. If these files are found, it will then superimpose them and write the result to the specified output file.

The exception to this is when encrypting and decrypting with k out of n schemes (such as algorithms 4 and 6), where k or n might change. In this case, the user must specify the number of shares to produce when encrypting an image. When decrypting an image produced by algorithm 4 (n out of n), the user must specify the shares to overlay. However, rather than requiring the user to specify filenames, the application allows the user to specify a comma-separated list of share numbers. For instance, specifying `--shares 1,2,3` would instruct `vcrypt` to overlay shares 1 - 3. This allows the user to experiment with various combinations of shares to see how the original image cannot be viewed unless all n shares are superimposed.

The following section presents a full user manual for `vcrypt`, complete with examples of its usage and operation.

6 User Manual

6.1 Requirements

`vcrypt` was implemented and tested using Ruby 1.8.7, and should be compatible with more recent versions as well. While the software was only tested on openSUSE Linux 11.1 and Solaris 9, it should theoretically run on any system having Ruby installed, since Ruby is an interpreted language.

6.2 Installation

The software is provided as a *tarball* which must be extracted at the command line. Executable permissions on the main `vcrypt.rb` file must then be granted to allow execution of the software. The following steps should be followed to install `vcrypt`:

- Open a command line and copy the provided `vcrypt.tar.gz` to the current directory
- Extract the archive: `gtar zxvf vcrypt.tar.gz`
- Change to the extracted `vcrypt` subdirectory: `cd vcrypt`
- Set the executable bit on `vcrypt.rb`: `chmod +x vcrypt.rb`

The program can then be run simply by typing `./vcrypt.rb` at the command line.

Important: In order to run a Ruby program, the Ruby executable must exist in the current path. On `bonnie` in the UWO Computer Science Research Network, Ruby can be found in `/opt/csw/bin`. If one is using the `bash` shell, the following command will add Ruby to the current path:

```
export PATH=$PATH:/opt/csw/bin
```

6.3 Getting Help at the Command Line

At any time, assistance can be obtained at the command line by running the program with no arguments, or by specifying one of the following switches as an argument to the program:

`--help` Displays the help screen presented when the program is run with no arguments

`--help-ns` Displays usage examples for the Naor and Shamir 2 out of 2 algorithm

`--help-ct` Displays usage examples for the Chen and Tsao family of algorithms.

6.4 Options

The syntax of an invocation of `vcrypt` is roughly defined as follows:

```
Usage:  vcrypt.rb MODE ALGORITHM [OPTIONS]
```

The elements of the syntax above are discussed in the sections below.

6.4.1 Mode

The software operates in either encryption or decryption mode, as shown below. These options are mutually exclusive.

Short Option	Long Option	Argument	Description
`-e`	`--encrypt`	IMAGE	Encrypt the specified image file
`-d`	`--decrypt`	OUTFILE	Decrypts shares to the specified output file

6.4.2 Algorithm

The following algorithms are supported by the software for use in encryption and decryption:

Short Option	Long Option	Argument	Description
`-a`	`--naor_shamir`		Use the Naor and Shamir's 2 out of 2 algorithm
`-c`	`--chen_tsao`	ALGORITHM	Use the Chen and Tsao family of algorithms

The following arguments to the `-c` option are accepted. **Note:** While algorithms 1 - 3 are actually attributed to Kafri and Keren [4], they are placed in the Chen and Tsao option for simplicity and because they are used extensively by the Chen and Tsao family of algorithms.

Algorithm Option	Description
1	Use the first random grid algorithm discussed in this paper [4]
2	Use a variant of algorithm 1 (poorer results) [4]
3	Use a variant of algorithm 1 (poorer results) [4]
4	Use the n out of n random grid algorithm discussed in this paper [5]
6	Use the 2 out of n random grid algorithm presented in [5]

6.4.3 Share Options

The following options control how shares are generated or interpreted. Note that the `-n` option is required when encrypting with Chen and Tsao's algorithms 4 or 6, while the `--shares` option is required when decrypting files encrypted by these algorithms.

Short Option	Long Option	Argument	Description
`-n`	`--numshares`	`NUM`	Number of shares to generate when encrypting with Chen & Tsao algorithms 4 or 6
`-s`	`--shares`	`SHARES`	Comma-separated integers specifying shares to decrypt when using C&T algorithms 4 or 6
	`--fix-contrast`		Fixes the contrast of the decrypted image when decrypting with Naor & Shamir as discussed in this paper
	`--sharedir`	`DIR`	Directory in which shares are found (defaults to `./shares`)
	`--shareprefix`	`PREFIX`	Share filename prefix (defaults to `ns` or `ct` based on the algorithm used)

6.5 Usage Examples

6.5.1 Encryption and Decryption using the Naor & Shamir 2 out of 2 Algorithm

Example 1: Encrypting `image1.pbm` with the Naor & Shamir 2 out of 2 algorithm

The following will generate two shares (`ns1.pbm` and `ns2.pbm`) stored in `./shares`. This directory will be created if it does not already exist.

```
./vcrypt.rb --encrypt image1.pbm --naor_shamir
```

Example 2: Decrypting `ns1.pbm` and `ns2.pbm` produced in Example 1

The following will search for two shares `ns1.pbm` and `ns2.pbm` stored in `./shares`. If found, the shares will be superimposed, with the result written to `decrypted.pbm`. By adding the `--fix-contrast` option, a second output file `decrypted-fixed.pbm` will be generated according to the algorithm discussed in section 2.3.

```
./vcrypt.rb --decrypt decrypted.pbm --naor_shamir --fix-contrast
```

6.5.2 Encryption and Decryption using Chen & Tsao's Algorithms

Example 3: Encrypting `image1.pbm` with the Kafri and Keren 2 out of 2 random grid algorithm

The following will generate two shares (`ct1.pbm` and `ct2.pbm`) stored in `./shares`. This directory will be created if it does not already exist.

```
./vcrypt.rb --encrypt image1.pbm --chen_tsao 1
```

Example 4: Decrypting `ct1.pbm` and `ct2.pbm` produced in Example 3

The following will search for two shares `ct1.pbm` and `ct2.pbm` stored in `./shares`. If found, the shares will be superimposed, with the result written to `decrypted.pbm`.

```
./vcrypt.rb --decrypt decrypted.pbm --chen_tsao 1
```

Example 5: Encrypting `image1.pbm` with the Chen and Tsao n out of n algorithm

The following encrypts `image1.pbm` over 5 shares using the n out of n random grid method discussed earlier. Shares `ct1.pbm` through `ct5.pbm` will be generated and stored in `./shares`. Notice that the `-n` option is required to indicate the number of shares to generate.

```
./vcrypt.rb --encrypt image1.pbm --chen_tsao 4 -n 5
```

Example 6: Decrypting shares produced in Example 5

The following superimposes shares `ct1.pbm`, `ct2.pbm`, and `ct3.pbm`, writing the resulting file to `decrypted.pbm`

```
./vcrypt.rb --decrypt decrypted.pbm --chen_tsao 4 --shares 1,2,3
```

Similarly, the following superimposes all shares (`ct1.pbm` through `ct5.pbm`):

```
./vcrypt.rb --decrypt decrypted.pbm --chen_tsao 4 --shares 1,2,3,4,5
```

References

[1] Moni Naor and Adi Shamir. Visual cryptography. In *Proceedings of Advances in Cryptology – EUROCRYPT 94, LNCS Vol. 950*, pages 1–12. Springer-Verlag, 1994.

[2] Gustavus J. Simmons. How to (really) share a secret. In *CRYPTO '88: Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology*, pages 390–448, London, UK, 1990. Springer-Verlag. ISBN 3-540-97196-3.

[3] Chang-Chou Lin and Wen-Hsiang Tsai. Secret image sharing with steganography and authentication. *J. Syst. Softw.*, 73(3):405–414, 2004. ISSN 0164-1212. doi: http://dx.doi.org/10.1016/S0164-1212(03)00239-5.

[4] O. Kafri and E. Keren. Encryption of pictures and shapes by random grids. *Optics Letters*, 12:377 – 379, 1987.

[5] Tzung-Her Chen and Kai-Hsiang Tsao. Visual secret sharing by random grids revisited. *Pattern Recognition*, 42(9):2203 – 2217, 2009. ISSN 0031-3203.

[6] L. W. Hawkes, A. Yasinsac, and C. Cline. An application of visual cryptography to financial documents. Technical report, Florida State University, 2000.

[7] R.-J. Hwang. A digital image copyright protection scheme based on visual cryptography. *Tamkang Journal of Science and Engineering*, 3(2):97–106, 2000.

[8] Giuseppe Ateniese, Carlo Blundo, Alfredo De Santis, and Douglas R. Stinson. Extended capabilities for visual cryptography. *Theoretical Cpmputer Science*, 250(1–2):143 – 161.

[9] Wen Pinn Fang. Visual cryptography in reversible style. In *IIHMSP 2007: Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 519–524, Kaohsiung, Taiwan, 2007.

[10] Wen-Pinn Fang. Non-expansion visual secret sharing in reversible style. *IJCSNS International Journal of Computer Science and Network Security*, 9(2):5 pages, 2009.

[11] Hsien-Chu Wu and Chin-Chen Chang. Sharing visual multi-secrets using circle shares. *Comput. Stand. Interfaces*, 28(1):123–135, 2005. ISSN 0920-5489.