

**科技部補助**  
**大專學生研究計畫研究成果報告**

|                 |  |
|-----------------|--|
| 計 畫<br>：<br>名 稱 | A Study on Random Grid-based Threshold Visual<br>Secret Sharing Scheme |
|-----------------|--|

報 告 類 別 ： 成果報告  
執行計畫學生： 林育如  
學生計畫編號： MOST 110-2813-C-260-005-E  
研 究 期 間 ： 110年07月01日至111年02月28日止，計8個月  
指 導 教 授 ： 阮夙姿

處 理 方 式 ： 本計畫可公開查詢

執 行 單 位 ： 國立暨南國際大學資訊工程學系（所）

中 華 民 國      111年05月10日

# **A Study on Random Grid-based Threshold Visual Secret**

## **Sharing Scheme**

### **Abstract**

Visual cryptography scheme (VCS) is topic of cryptography that deals with the encryption and decryption of images. A  $(k, n)$  visual cryptographic scheme (VCS) is a secret sharing method, which encodes a secret image  $A$  into  $n$  share in such a way that the stacking of any more than or equal to  $k$  share will recover  $A$ , while any less than  $k$  share provide no clue about  $A$ .

In this study, we will design three algorithms step by step, making this algorithm more functional and more widely used. Algorithm 1 encrypt a binary secret image to  $n$  random grid images, so that the collection of at least  $k$  pieces can be restored back to the secret image by OR operator. The contrast of this algorithm 1 will better than the algorithm proposed by Chen and Cao in 2011. Algorithm 2 we will add the function of XOR operator to decrypt on the basis of algorithm 1 so that there are two decryption methods OR operator and XOR operator can use. Algorithm 3 we will improve algorithm 2. In algorithm 2 the secret image can be seen only when  $t = n$  and  $k$ , but algorithm 3 the secret image can be seen when  $k \leq t \leq n$ . The contrast of the restored image will be improved, and then we will prove our Algorithms are safe and useful. As far as we know, this is the first algorithm with these capabilities.

**Keywords:** Visual Secret Image Sharing, Visual Cryptography, Contrast, Random grids.

# 一種基於多解密方式隨機網格並且有門檻的視覺秘密

## 摘要

視覺密碼(VCS)是密碼學的一個主題,處理影像的加密和解密。 $(k, n)$  視覺密碼(VCS)是秘密共享的一種方法,它將秘密圖像  $A$  加密為  $n$  張影像,使得任何大於或等於  $k$  張影像的堆疊將恢復  $A$ ,而任何少於  $k$  張影像都無法知道關於  $A$  的任何線索。

在本研究中,我們將循序漸進地設計三種演算法,使得演算法變得具有更多功能並且運用更加廣泛。方法一將黑白秘密影像加密為  $n$  張隨機網格影像,至  $k$  張的集合進行 OR 運算還原回秘密影像,此方法的對比度將會比 2011 年陳和曹論文所提出的方法還要好。方法二我們將在方法一的基礎上加上 XOR 運算解密的功能使得演算法有 OR 和 XOR 兩種解密方法。方法三我們將改良方法二,將原本  $t = n, k$  才能看出秘密方法改良成  $k \leq t \leq n$  都是可以看出秘密影像的,將還原圖像的對比度提高,之後我們將證明我們的算法是安全且有用的。就我們所知,這是具有這些功能的第一個演算法。

**關鍵字：**視覺秘密影像分享, 視覺密碼, 對比度, 隨機影像

# 1. 問題介紹與研究動機

秘密共享(Secret Sharing)[4]將用戶數據(data)加密成不同的片段(share)並將其分給多個參與者,使得每位參與者都會持有解密的關鍵,只有集齊一定的參與者才能解密,這引起了各界的廣泛關注。視覺密碼(Visual Cryptographic Scheme,簡稱VCS) [5]是秘密共享的主要分支。視覺密碼是將一張秘密影像加密成數張加密影像(share)並分給數個參與者,需要一定數量的參與者拿出加密影像才能解密。這樣加密影像的方式有許多種,接下來我們討論的是使用隨機網格[1](Random Grids)的方式加密,加密後的加密影像會和秘密影像大小相同且毫無意義。解密時我們採用有門檻[2](threshold  $k$  out of  $n$ ,  $n$  張加密影像中要  $k$  張或  $k$  張以上才能還原)的方式,並且可以直接疊起來,不需要任何電腦計算。本研究之動機希望改善的解密方式中,疊  $k$  張、疊  $k+1$  張, ..., 疊  $n$  張的還原影像更加清晰可見,光透率得到提升。並且加入 XOR 的解密方式,使得其運用更加廣泛。

Kafri 和 Keren 定義隨機網格中的每一個像素只有分為透明或是不透明[1]。且每一個像素是由亂數產生,所以透明像素量會等於不透明像素量。因此隨機網格的平均光透率( $L$ )為 0.5,其中光透率的定義為全部的像素量分之透明的像素量。他們並提出三種不同的演算法來加密一張黑白的秘密影像,輸入一張要加密的黑白影像  $A$ ,輸出得到隨機網格,或稱為加密影像(share)。

在 2011 年陳和曹改進了演算法[3],使得原來是能加密成兩張加密影像變成可以加密成  $n$  張並使用有門檻(Threshold) [2]的方式來還原,而此時每張隨機網格的光透率仍為 0.5。此方法必須產生一個二維陣列,依據每次新產生的隨機網格對此陣列值做修改,這是為了第  $k$  張加密影像而作準備。也就是說,前面的  $k-1$  張加密影像皆是由亂數所產生,只有第  $k$  張是根據前面的二維陣列及原圖所產生的,而後面的  $k+1$  到  $n$  張也是由亂數產生的。產生的任何一張加密影像都無法看出秘密影像為何,一定要  $k$  張或是  $k$  張以上的加密影像疊起來才能看出個秘密影像。但是此種演算法只要  $n$  越大輸出的加密影像越多就會越看不清楚,疊合後還原影像的對比度會越來越接近 0 ( $(0.5)^t$ ,  $k \leq t \leq n$ )。

因此,本研究的目的是設定出一種演算法,選定  $k$  張或  $k$  張以上的加密影像疊起來後對比度可以提升,使得多張疊合後的影像仍舊清晰可辨識。除此之外,再進一步改良以增加 XOR 解密方式[4],使得我們的演算法能夠更有彈性的被多方應用。

## 2. 相關研究

首先,先定義兩像素所疊合的所有結果,其中 1 代表黑色,0 代表白色的像素:

| $b_1$ | $b_2$ | $b_1 \oplus b_2$ |
|-------|-------|------------------|
| 0     | 0     | 0                |
| 0     | 1     | 1                |
| 1     | 0     | 1                |
| 1     | 1     | 1                |

Kafri 和 Keren 只能產生兩張隨機網格的演算法[1]如下, 其中  $A$  為秘密影像, 尺寸為  $w \times h$ :

### 演算法 1:

```

Generate a random grid  $B_1$ ,  $L(B_1) = \frac{1}{2}$ 
for(  $i = 0 ; i < w ; i++$  )
    for(  $j = 0 ; j < h ; j++$  )
        if(  $A[i][j] == 0$  )  $B_2[i][j] = B_1[i][j]$  ;
        else  $B_2[i][j] = \sim B_1[i][j]$  ;
output (  $B_1, B_2$  )

```

下表為演算法 1 疊合後的光透率, 其中  $a$  代表原始秘密影像的像素值,  $b$  代表加密影像的像素值:

|       | $a$ | $b_1$  | $b_2$  | $b_1 \oplus b_2$ | 光透率           |
|-------|-----|--------|--------|------------------|---------------|
| 演算法 1 | □   | □<br>■ | □<br>■ | □<br>■           | $\frac{1}{2}$ |
|       | ■   | □<br>■ | ■<br>□ | ■<br>■           | 0             |

2011 年陳和曹將此演算法延伸出可以產生多張片段的演算法[2]如下。

## 演算法 2:

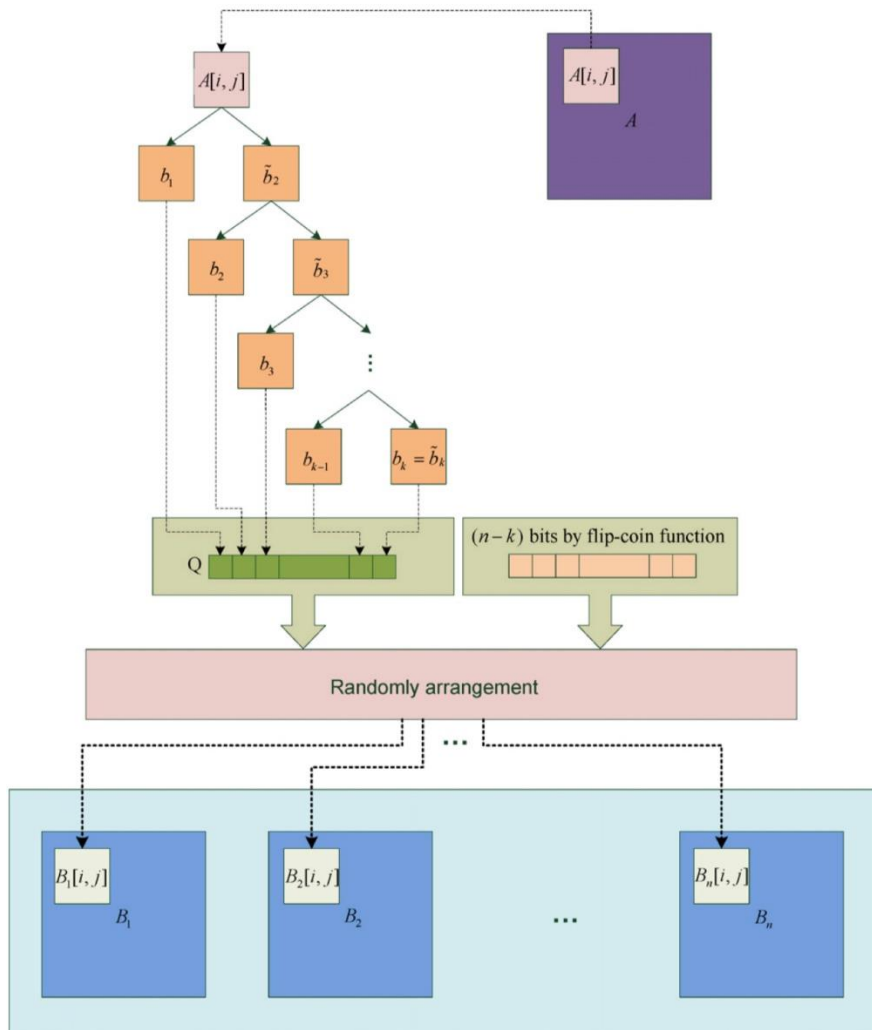
```

k-out-of-n_RandomGrids(){
   $b_1$  create by random select  $\{0,1\}$ 
  if (  $a == 0$  )  $\tilde{b}_2 = b_1$  ;
  else  $\tilde{b}_2 = \sim b_1$  ;

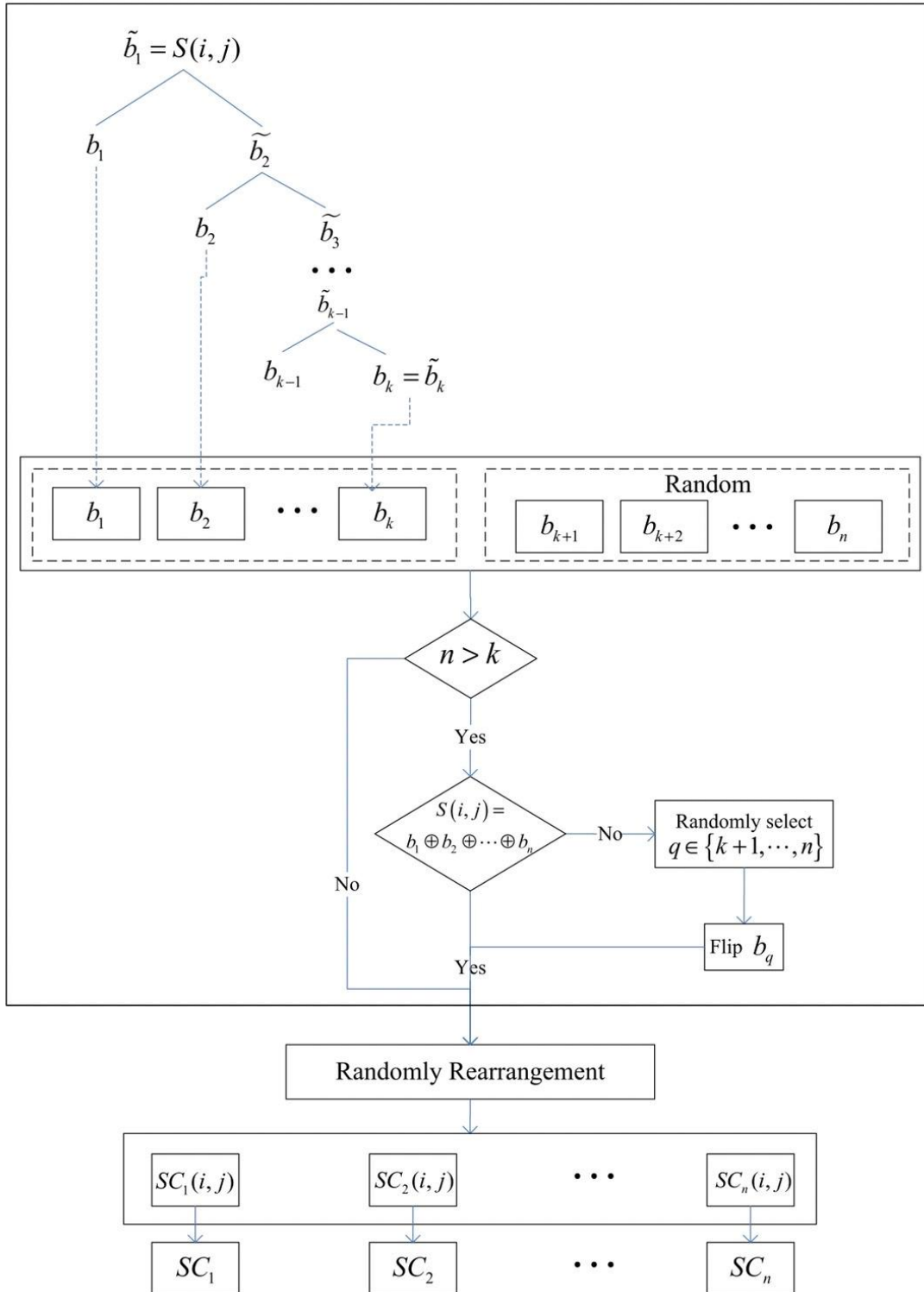
  for ( $x=2$ ;  $x < k$ ;  $x++$ ){
     $b_x$  create by random select  $\{0,1\}$ 
    if(  $\tilde{b}_x == 0$  )  $\tilde{b}_{x+1} = b_x$  ;
    else  $\tilde{b}_{x+1} = \sim b_x$  ;
  }
  for ( $x=k+1$ ;  $x < n+1$ ;  $x++$ ){
     $b_x$  create by random select  $\{0,1\}$ 
  }
  Randomly arrangement  $b_1, b_2, \dots, b_n$  to  $B_1, B_2, \dots, B_n$ 
}

```

此法所產生的多重隨機網格樹狀圖如下所示：



2015 年 Yan 等人在陳和曹的演算法延伸出可以使用 XOR 和 OR 兩種解密方式的演算法[4], 如下所示:



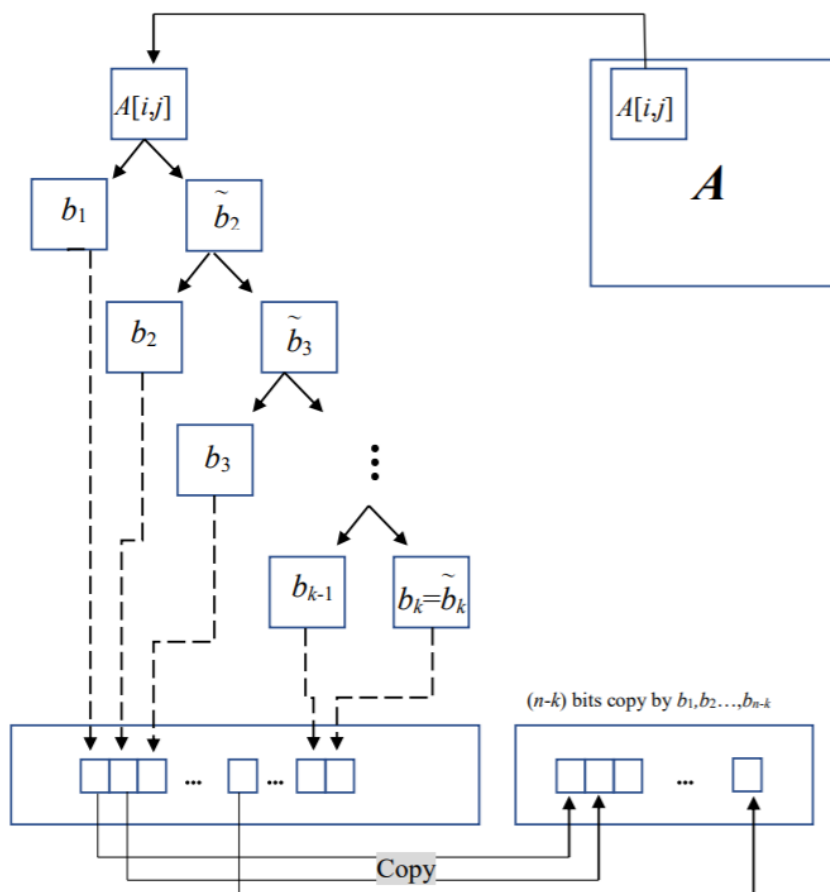
### 3. 研究成果

本研究計畫共有三個主要研究成果，以下將分為三個子節去討論：成果一改進 2011 年陳和曹論文所提出演算法的對比度；成果二在成果一的基礎上增加 XOR 運算的解密方式，使得成果二有 OR 運算和 XOR 運算兩種解密方式；方法三我們將改良方法二，將原本  $t = n$ 、 $k$  才能看出秘密方法改良成  $k \leq t \leq n$  都是可以看出秘密影像的，將還原圖像的對比度提高。

#### 3.1 成果一

觀察[2]之方法發現，當每張加密影像的光透率為 0.5，而共分為  $n$  張加密影像時，疊合  $n$  張後白色的光透率會降為  $(\frac{1}{2})^{n-1}$ ，而黑色部分仍維持為 0。由於對比度的定義為  $(\text{白色的光透率} - \text{黑色的光透率}) / (1 + \text{黑色的光透率})$ ，此法的對比度就會是  $((\frac{1}{2})^{n-1} - 0) / (1 + 0) = (\frac{1}{2})^{n-1}$ ，所以疊合越多張越不易看清楚原圖為何，對比度會越來越接近 0。

改良演算法如下：將產生第  $k+1$  到第  $n$  張加密影像的過程修改成第  $k+1$  複製第一張，第  $k+2$  張複製第二張，以此類推至第  $n$  張。演算法樹狀圖如下：





舉例如下：

1. 加密影像張數為 3 時， $k=2$ ：

| $a$ | $b_1$ | $b_2$ | $b_3$ | $b_1+b_2$ | $b_1+b_3$ | $b_2+b_3$ | $b_1+b_2+b_3$ |
|-----|-------|-------|-------|-----------|-----------|-----------|---------------|
| □   | □     | □     | □     | □         | □         | □         | □             |
|     | ■     | ■     | ■     | ■         | ■         | ■         | ■             |
| ■   | □     | ■     | □     | ■         | □         | ■         | ■             |
|     | ■     | □     | ■     | ■         | ■         | ■         | ■             |


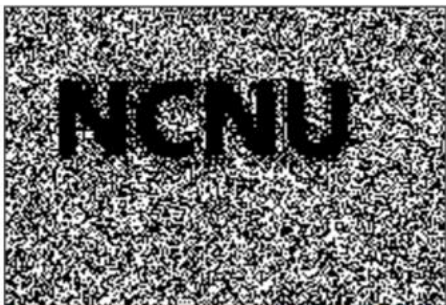
從上表可知，由上表可看出疊合兩張後白色部分的光透率  $= \frac{3}{6} = \frac{1}{2}$ ，黑色部分的光透率  $= \frac{1}{6}$ ，經過計算其對比度為  $(\frac{1}{2} - \frac{1}{6}) / (\frac{1}{2} + \frac{1}{6}) = \frac{2}{7}$ 。疊合三張後白色部分的光透率  $= \frac{1}{2}$ ，黑色部分的光透率  $= 0$ ，經過計算其對比度為  $\frac{1}{2}$ 。由此可看出，與原演算法對比還原影像的對比度得到很高的提升。

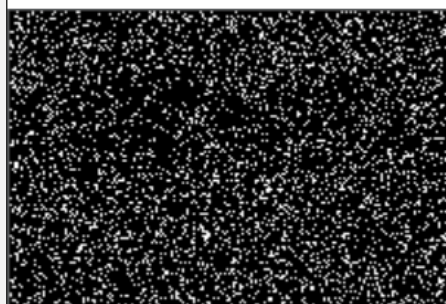



實驗結果如下：

原圖



| $n=3, k=2$ | 共分三張，疊合其中兩張後 | 共分三張，疊合其中三張後 |
|------------|--------------|--------------|
| 原演算法       |              |              |

|            |   |  |
|------------|---|--|
| 改良後<br>演算法 |  |  |
|------------|---|--|

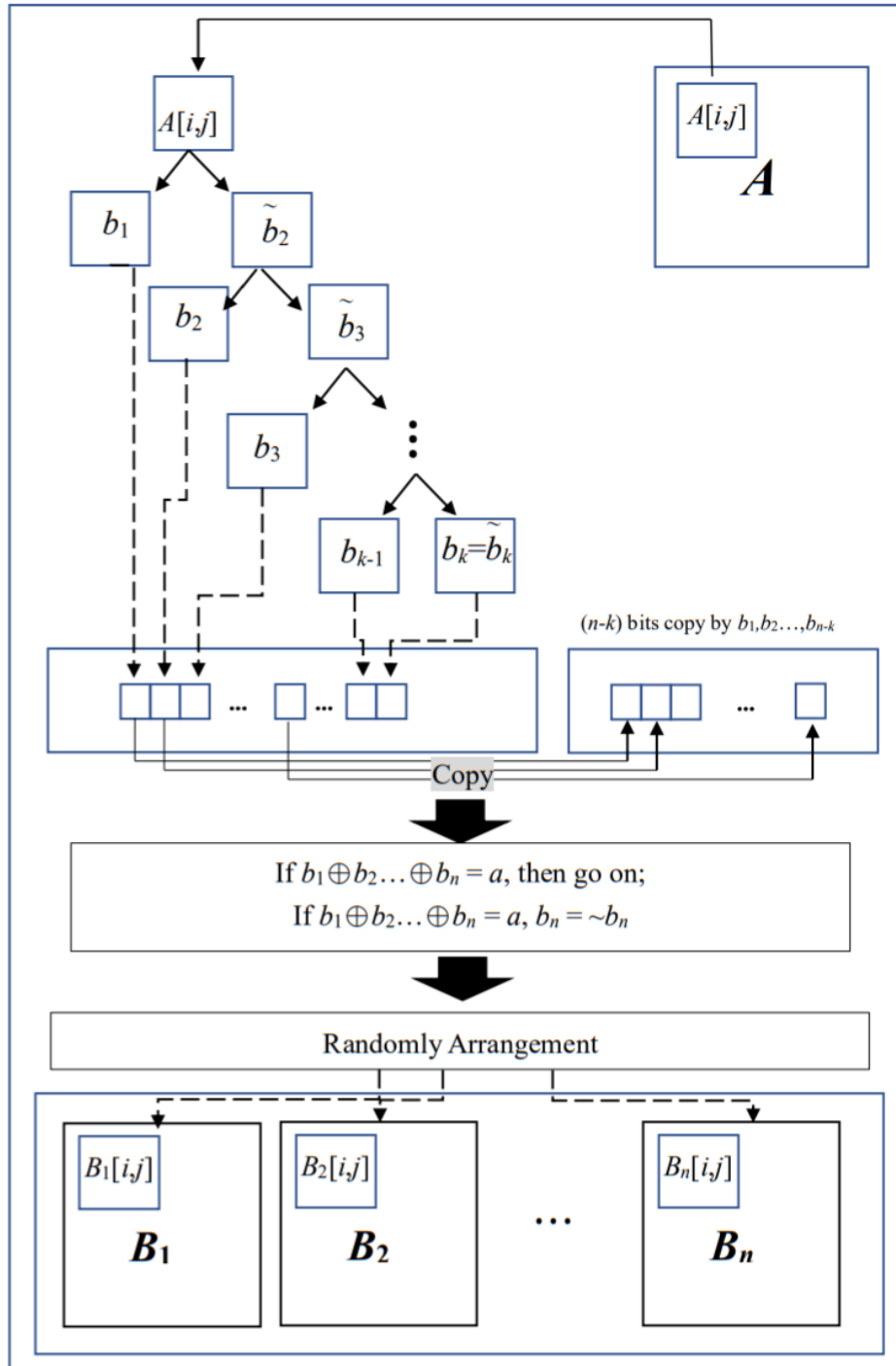
|            |   |  |
|------------|---|--|
| $n=4, k=3$ | 共分四張, 疊合其中三張後   | 共分四張, 疊合其中四張後  |
| 原<br>演算法   |   |   |
| 改良後<br>演算法 |  |  |

由上面實驗結果可以看出, 此法可以將疊合後的結果會變的不清楚的問題改善已讓圖像更明顯的可辨識。

### 3.2 成果二

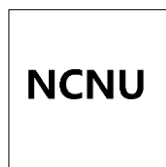
本方法將增加 XOR 解密功能議題做研究。根據 2015 年 Yan 等人發表的論文[4]進行改良。

在每張圖的 pixel 被打亂之前先進行一次 XOR, 如果 XOR 的結果與原圖  $A$  相同就進行下一步, 若是不同則翻轉最後一張。演算法樹狀圖如下:

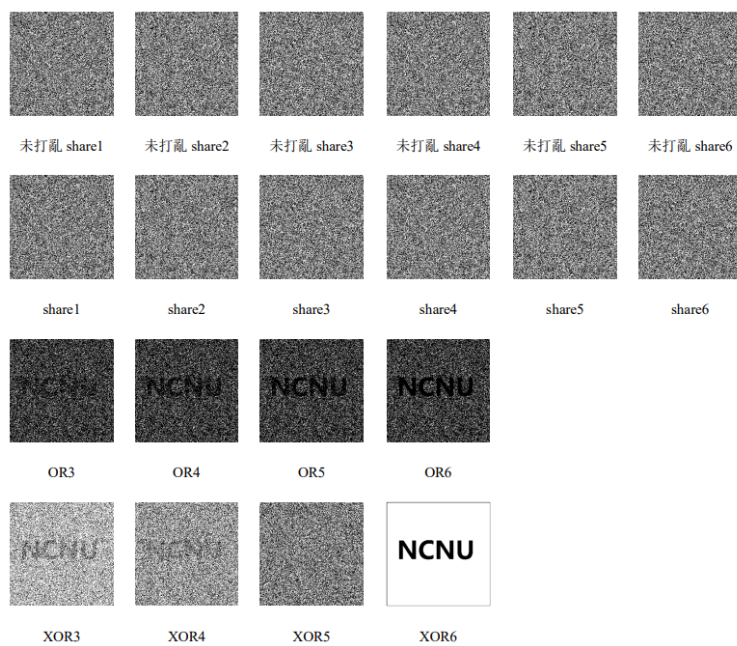


實驗結果如下：

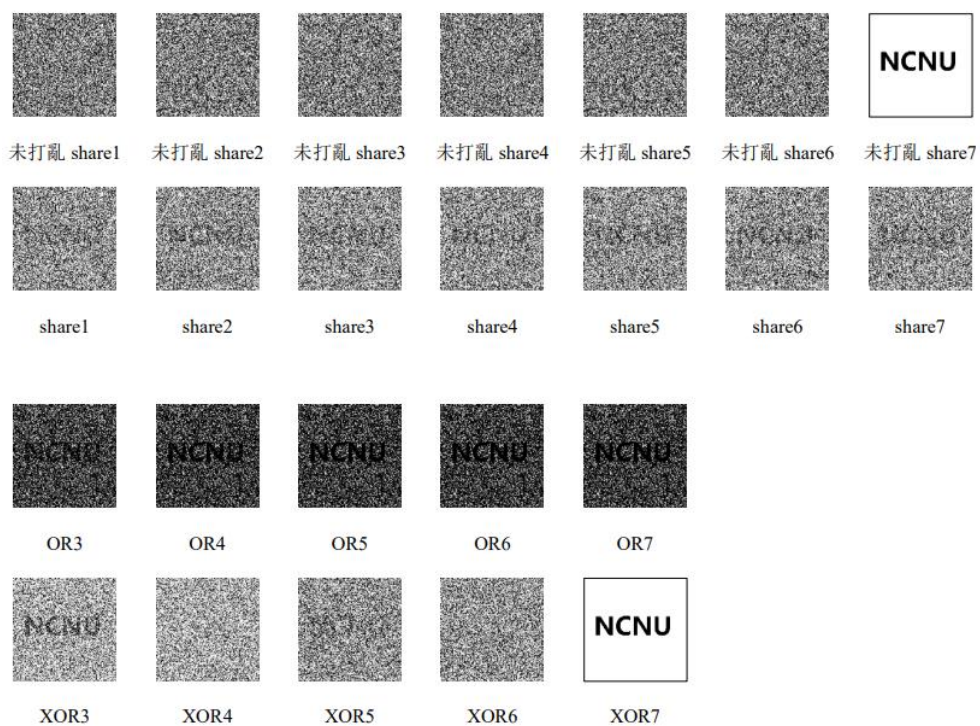
原圖



$n = 6$   $k = 3$



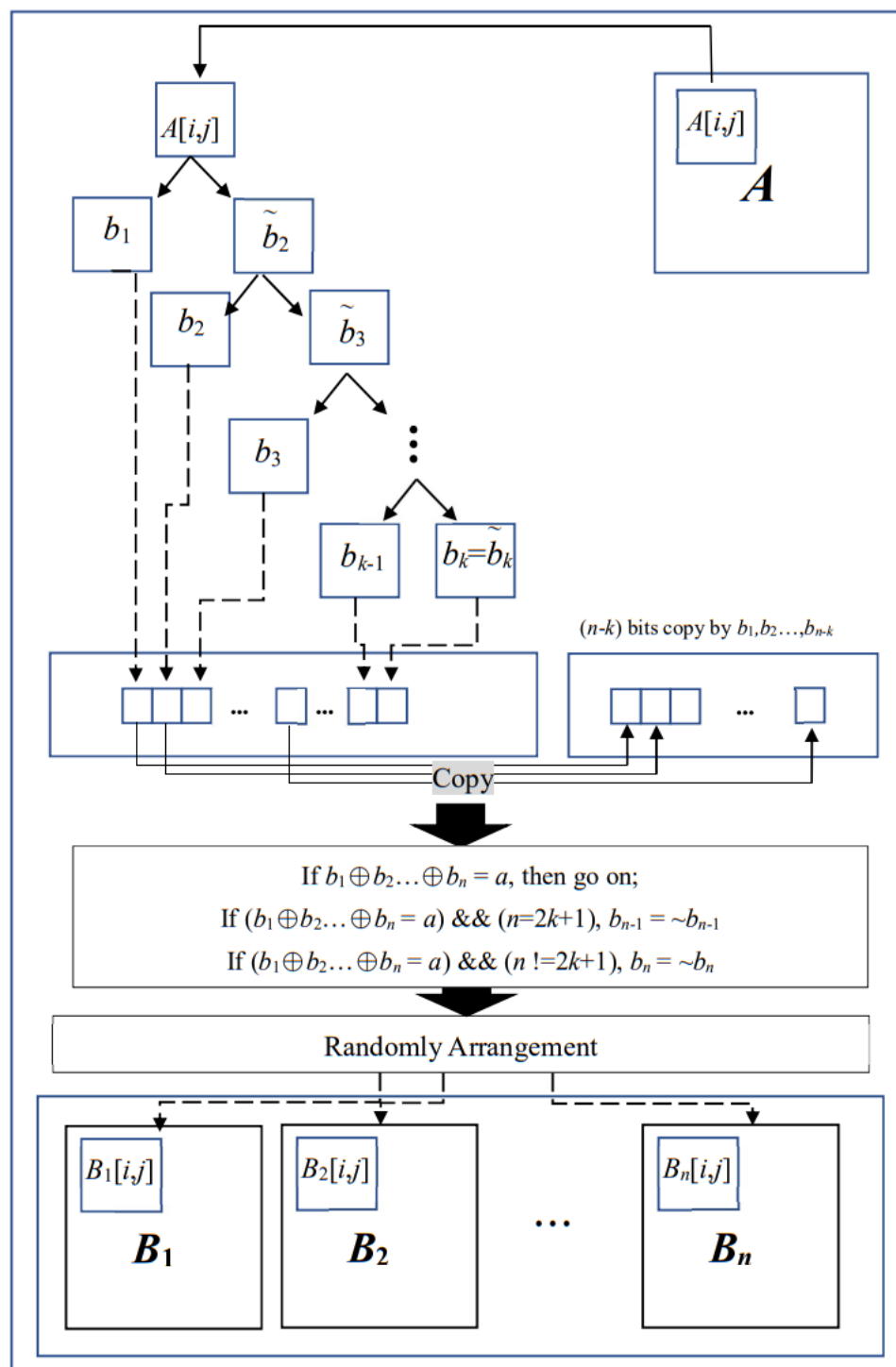
$n = 7$   $k = 3$





經實驗我們發現當  $n = 2k+1$  時會發生秘密洩露的情況。之後我們進行演算法的分析,發現當  $n = 2k+1$  時因為  $k+1, k+2, \dots, n$  都是複製前面的 pixel, 所以在 XOR 時就會發生兩兩互相抵消的情況, 導致最後的第  $n$  張就會被翻成秘密圖像, 打亂之後的每張 share 都有些秘密圖像的影子。

有了上述結論我們再繼續將演算法進行修改, 當  $n = 2k+1$  時將原來的翻轉最後一張改成翻轉最後第二張。演算法樹狀圖如下:

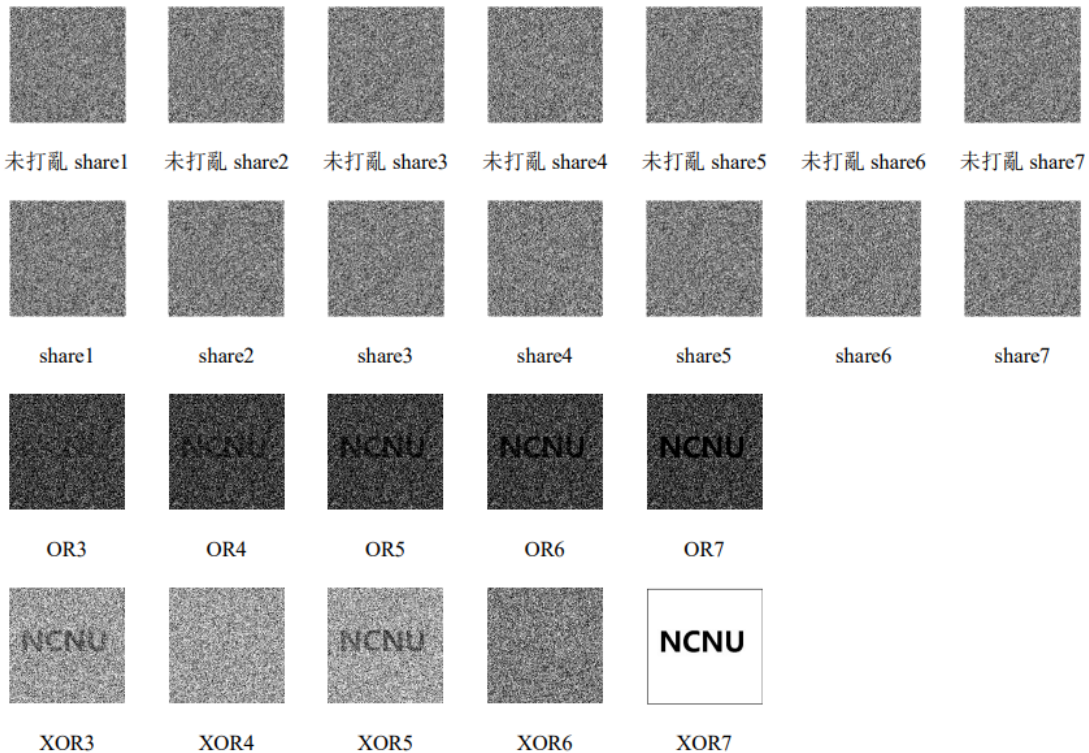


實驗結果如下：

原圖



$n = 7$   $k = 3$

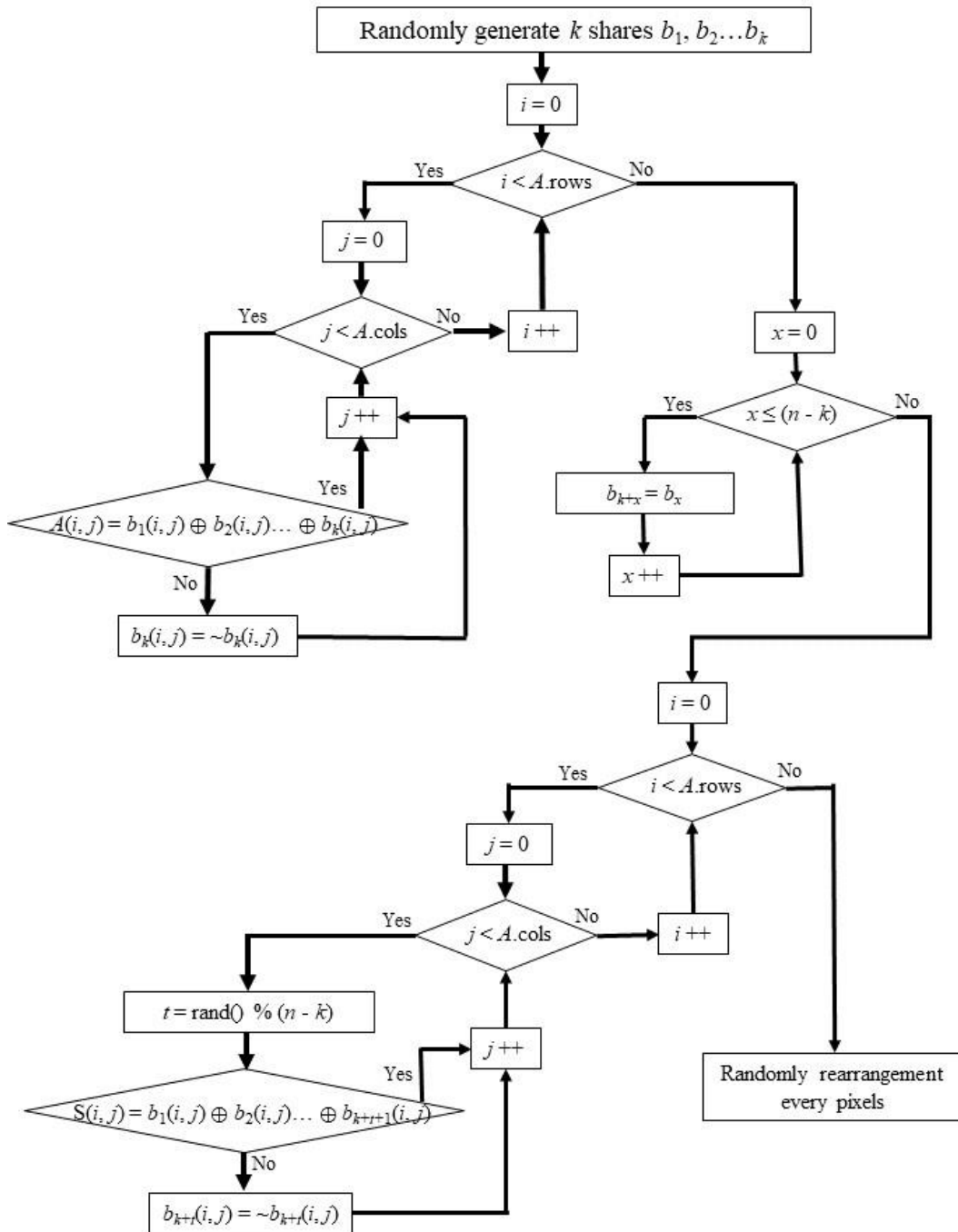


經上述實驗結果我們發現秘密洩露的問題已經解決,但是因為翻轉了一張圖片,在 OR 的還原上作出了一些犧牲,所以在 OR 疊三張的時候對比度會降低。在 XOR 方面,我們可以發現除了疊  $k$  張和  $n$  張之外,疊其他張數都不太能看出秘密圖像,對比度也很低。

### 3.3 成果三

繼續修改演算法, 試著解決上述實驗中  $k+1 \sim n$  張圖片疊起來會看不見的問題。

我們在之前的演算法後再加上一步, 每個像素都從  $k+1$  到  $n$  的範圍內隨機取一個數字  $t$ , 把前  $t$  張用 XOR 疊起來檢查是否和原圖相符, 一樣的話不變, 否則把第  $k+t$  張的像素翻轉。演算法流程圖如下:

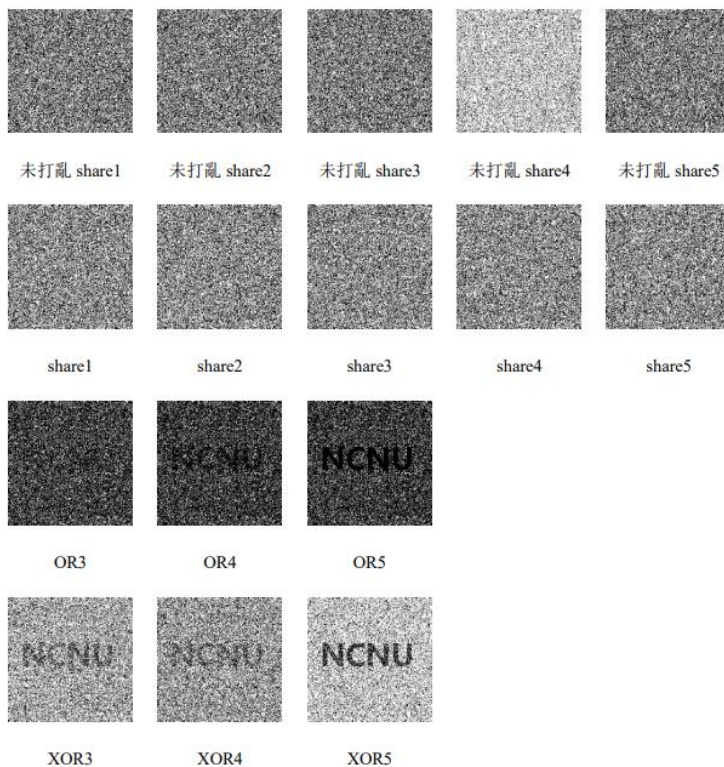


實驗結果如下：

原圖



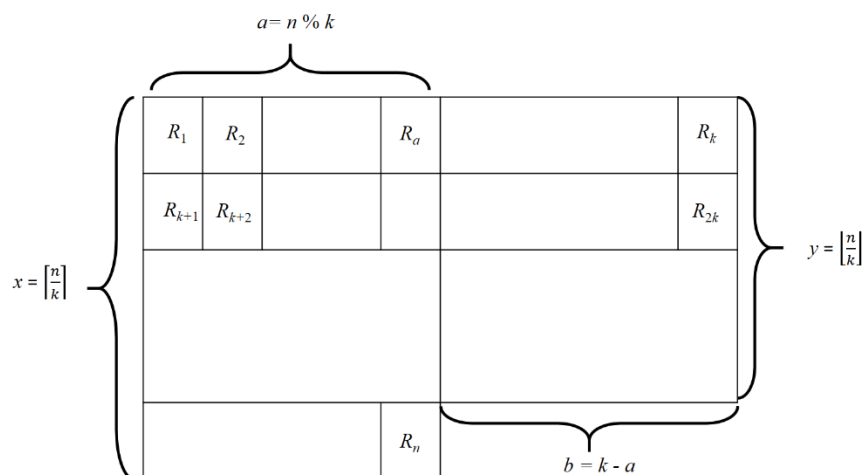
$$n = 5 \quad k = 3$$



從這個結果我們可以發現和成果二演算法的結果相比 XOR 每張都可以看的清楚, 對比度也有所提高。接下來我們將從光透率來分析成果三演算法的正確率及對比度(Alpha)。

首先, 我們先要看  $t$  張運用 XOR 運算疊起來的正確率是多少, 其中  $t$  必須大於等於  $k$ 。第一種情況是  $t$  等於  $k$ , 根據演算法我們知道在像素未打亂前, 前  $k$  張疊起來會完全還原成原圖, 第  $k+1$  張會與第 1 張圖片相同, 第  $k+2$  張圖片與第 2 張圖片相同..... 第  $k+p$  張會與  $p \% k$  張圖片相同,  $p \% k = 0$  時第  $k$  張相同, 其中  $p \leq (n-k)$ 。在這個階段就會有  $x^a y^b$  ( $a$ 、 $b$ 、 $x$ 、 $y$  如下圖所示) 種情況可以完全還原。在下一個階段中, 第  $k+1$  到第  $n$  張圖片的每一個像素都會有  $1/(n-k)$  的機率會被選中, 並且有  $1/2$  的機率會被翻轉, 所以除了  $t$  選擇 1 到  $k$  的這組不會被翻轉, 剩餘的  $(a+b)y-1$  的  $n-k$  個像素都會有  $(1/2)(1/(n-k))$  的機率會翻轉。而剩下的  $C(n, k) - x^a y^b$  種情況中都是隨機的一半會對一半會錯。





第二種情況是  $t$  大於  $k$ ，與第一種情況不同的是沒有  $t$  選擇第 1 到  $k$  張的選擇，所以在  $k < t \leq n$  會每個  $t$  還原成原圖的機率只有  $1/(n-k)$ 。

但實際上我們在實驗的時候發現，白色像素和黑色像素不是我們想像中正確率相等的樣子。分析後發現，是因為在演算法第一部分的時候把前  $k$  張做 XOR 運算已經會完整還原成原圖  $A$  了，演算法的第二步是將第  $k$  張以上的圖片都會有  $1/(n-k)$  的機率做 XOR 看有沒有與原圖相等，所以第  $k+1$  張會有  $1/(n-k)$  的機率永遠是白色的。

在  $n=4, k=2, t=2$  的例子中白色的正確率為 88%，黑色的正確率為 50%，並不是想像中的白色黑色的正確率相等。如下圖的運算表所示：

| $A$ | 1 | 2 | 3 | 4 | 12 | 13 | 14 | 23 | 24 | 34 |
|-----|---|---|---|---|----|----|----|----|----|----|
| 0   | 0 | 0 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
|     | 0 | 0 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  |
|     | 1 | 1 | 0 | 1 | 0  | 1  | 0  | 1  | 0  | 1  |
|     | 1 | 1 | 1 | 1 | 0  | 0  | 0  | 0  | 0  | 0  |
| 1   | 0 | 1 | 0 | 1 | 1  | 1  | 1  | 0  | 0  | 0  |
|     | 0 | 1 | 0 | 0 | 1  | 0  | 0  | 1  | 1  | 0  |
|     | 1 | 0 | 0 | 0 | 1  | 1  | 1  | 0  | 0  | 0  |
|     | 1 | 0 | 1 | 1 | 1  | 0  | 0  | 1  | 1  | 0  |

|       |     |    |   |   |   |   |   |   |
|-------|-----|----|---|---|---|---|---|---|
| 白色正確率 | 88% | 白對 | 4 | 3 | 4 | 3 | 4 | 3 |
| 黑色正確率 | 50% | 黑對 | 4 | 2 | 2 | 2 | 2 | 0 |

|       |       |       |       |
|-------|-------|-------|-------|
| 正確率   | 0.688 | 白色光透率 | 0.875 |
| Alpha | 0.25  | 黑色光透率 | 0.5   |

由於白色像素黑色像素正確率不相等的原因，正確率很難用一般式算出來，所以我們使用上述  $n=4, k=2, t=2$  時相同方式一一進行分析將正確率算出來，之後運用白色的光透率與黑色的光透率算出對比度。結果如下圖所示：

| XOR白色像素正確率 |        |        |        |        |        |        |
|------------|--------|--------|--------|--------|--------|--------|
| $(k, n)$   | $t=2$  | $t=3$  | $t=4$  | $t=5$  | $t=6$  | $t=7$  |
| (2,4)      | 0.8750 | 0.6875 | 0.7500 |        |        |        |
| (3,4)      |        | 0.3500 | 1      |        |        |        |
| (4,4)      |        |        | 1      |        |        |        |
| (3,5)      |        | 0.6000 | 0.7000 | 0.7500 |        |        |
| (4,5)      |        |        | 0.8000 | 1      |        |        |
| (5,5)      |        |        |        | 1      |        |        |
| (3,6)      |        | 0.6670 | 0.6222 | 0.5830 | 0.6667 |        |
| (4,6)      |        |        | 0.7778 | 0.5830 | 0.7500 |        |
| (5,6)      |        |        |        | 0.5830 | 1      |        |
| (6,6)      |        |        |        |        | 1      |        |
| (4,7)      |        |        | 0.5900 | 0.5556 | 0.5480 | 0.6670 |
| (5,7)      |        |        |        | 0.5670 | 0.5540 | 0.7813 |
| (6,7)      |        |        |        |        | 0.5540 | 1      |
| (7,7)      |        |        |        |        |        | 1      |

| XOR黑色像素正確率 |        |        |        |        |        |        |
|------------|--------|--------|--------|--------|--------|--------|
| $(k, n)$   | $t=2$  | $t=3$  | $t=4$  | $t=5$  | $t=6$  | $t=7$  |
| (2,4)      | 0.5000 | 0.5625 | 0.7500 |        |        |        |
| (3,4)      |        | 0.3500 | 1      |        |        |        |
| (4,4)      |        |        | 1      |        |        |        |
| (3,5)      |        | 0.5500 | 0.7000 | 0.7500 |        |        |
| (4,5)      |        |        | 0.8000 | 1      |        |        |
| (5,5)      |        |        |        | 1      |        |        |
| (3,6)      |        | 0.5667 | 0.5110 | 0.5278 | 0.6670 |        |
| (4,6)      |        |        | 0.7778 | 0.5830 | 0.7500 |        |
| (5,6)      |        |        |        | 0.5830 | 1      |        |
| (6,6)      |        |        |        |        | 1      |        |
| (4,7)      |        |        | 0.5430 | 0.5400 | 0.5480 | 0.6670 |
| (5,7)      |        |        |        | 0.5600 | 0.5700 | 0.7500 |
| (6,7)      |        |        |        |        | 0.5540 | 1      |
| (7,7)      |        |        |        |        |        | 1      |

| XOR平均正確率 |        |        |        |        |        |        |
|----------|--------|--------|--------|--------|--------|--------|
| $(k,n)$  | $t=2$  | $t=3$  | $t=4$  | $t=5$  | $t=6$  | $t=7$  |
| (2,4)    | 0.6875 | 0.6250 | 0.7500 |        |        |        |
| (3,4)    |        | 0.6250 | 1      |        |        |        |
| (4,4)    |        |        | 1      |        |        |        |
| (3,5)    |        | 0.6250 | 0.6000 | 0.7500 |        |        |
| (4,5)    |        |        | 0.6000 | 1      |        |        |
| (5,5)    |        |        |        | 1      |        |        |
| (3,6)    |        | 0.6167 | 0.5667 | 0.5560 | 0.6670 |        |
| (4,6)    |        |        | 0.5830 | 0.5830 | 0.7500 |        |
| (5,6)    |        |        |        | 0.5830 | 1      |        |
| (6,6)    |        |        |        |        | 1      |        |
| (4,7)    |        |        | 0.5667 | 0.5476 | 0.5480 | 0.6670 |
| (5,7)    |        |        |        | 0.5600 | 0.5700 | 0.7500 |
| (6,7)    |        |        |        |        | 0.5700 | 1      |
| (7,7)    |        |        |        |        |        | 1      |

| XOR Alpha |        |        |        |        |        |        |
|-----------|--------|--------|--------|--------|--------|--------|
| $(k,n)$   | $t=2$  | $t=3$  | $t=4$  | $t=5$  | $t=6$  | $t=7$  |
| (2,4)     | 0.2500 | 0.1740 | 0.4000 |        |        |        |
| (3,4)     |        | 0.1820 | 1      |        |        |        |
| (4,4)     |        |        | 1      |        |        |        |
| (3,5)     |        | 0.1820 | 0.1430 | 0.4000 |        |        |
| (4,5)     |        |        | 0.1423 | 1      |        |        |
| (5,5)     |        |        |        | 1      |        |        |
| (3,6)     |        | 0.1628 | 0.0900 | 0.0760 | 0.2500 |        |
| (4,6)     |        |        | 0.1170 | 0.1170 | 0.4000 |        |
| (5,6)     |        |        |        | 0.1170 | 1      |        |
| (6,6)     |        |        |        |        | 1      |        |
| (4,7)     |        |        | 0.0920 | 0.0650 | 0.0660 | 0.2500 |
| (5,7)     |        |        |        | 0.0830 | 0.0980 | 0.4000 |
| (6,7)     |        |        |        |        | 0.1    | 1      |
| (7,7)     |        |        |        |        |        | 1      |

接下來我們用上述相同方式分析 OR 運算的正確率，結果是否和成果二預期的一樣， $t$  疊越多的時候對比度越高，不會出現和 2011 年陳和曹論文所提出演算法一樣越疊越黑的情況。結果如下圖所示：

| OR白色像素正確率 |        |        |        |        |        |        |
|-----------|--------|--------|--------|--------|--------|--------|
| $(k,n)$   | $t=2$  | $t=3$  | $t=4$  | $t=5$  | $t=6$  | $t=7$  |
| (2,4)     | 0.5000 | 0.5000 | 0.5000 |        |        |        |
| (3,4)     |        | 0.1313 | 0.2500 |        |        |        |
| (4,4)     |        |        | 0.1250 |        |        |        |
| (3,5)     |        | 0.2750 | 0.2500 | 0.2500 |        |        |
| (4,5)     |        |        | 0.1250 | 0.1250 |        |        |
| (5,5)     |        |        |        | 0.0625 |        |        |
| (3,6)     |        | 0.2625 | 0.2500 | 0.2500 | 0.2500 |        |
| (4,6)     |        |        | 0.1500 | 0.1250 | 0.1250 |        |
| (5,6)     |        |        |        | 0.0625 | 0.0625 |        |
| (6,6)     |        |        |        |        | 0.0313 |        |
| (4,7)     |        |        | 0.1500 | 0.1310 | 0.1250 | 0.1250 |
| (5,7)     |        |        |        | 0.0804 | 0.0625 | 0.0625 |
| (6,7)     |        |        |        |        | 0.0346 | 0.0313 |
| (7,7)     |        |        |        |        |        | 0.0156 |

| OR黑色像素正確率 |        |        |        |        |        |       |
|-----------|--------|--------|--------|--------|--------|-------|
| $(k,n)$   | $t=2$  | $t=3$  | $t=4$  | $t=5$  | $t=6$  | $t=7$ |
| (2,4)     | 0.7083 | 0.8750 | 1      |        |        |       |
| (3,4)     |        | 0.4500 | 1      |        |        |       |
| (4,4)     |        |        | 1      |        |        |       |
| (3,5)     |        | 0.7875 | 0.9000 | 1      |        |       |
| (4,5)     |        |        | 0.9000 | 1      |        |       |
| (5,5)     |        |        |        | 1      |        |       |
| (3,6)     |        | 0.8125 | 0.9000 | 0.9583 | 1      |       |
| (4,6)     |        |        | 0.8708 | 0.9375 | 1      |       |
| (5,6)     |        |        |        | 0.9479 | 1      |       |
| (6,6)     |        |        |        |        | 1      |       |
| (4,7)     |        |        | 0.8714 | 0.9226 | 0.9643 | 1     |
| (5,7)     |        |        |        | 0.9271 | 0.9643 | 1     |
| (6,7)     |        |        |        |        | 0.9394 | 1     |
| (7,7)     |        |        |        |        |        | 1     |

| OR平均正確率 |        |        |        |        |        |        |
|---------|--------|--------|--------|--------|--------|--------|
| $(k,n)$ | $t=2$  | $t=3$  | $t=4$  | $t=5$  | $t=6$  | $t=7$  |
| (2,4)   | 0.6875 | 0.6875 | 0.7500 |        |        |        |
| (3,4)   |        | 0.5313 | 0.6250 |        |        |        |
| (4,4)   |        |        | 0.5625 |        |        |        |
| (3,5)   |        | 0.5313 | 0.5750 | 0.6250 |        |        |
| (4,5)   |        |        | 0.5125 | 0.5625 |        |        |
| (5,5)   |        |        |        | 0.5313 |        |        |
| (3,6)   |        | 0.5375 | 0.5750 | 0.6042 | 0.6250 |        |
| (4,6)   |        |        | 0.5104 | 0.5313 | 0.5625 |        |
| (5,6)   |        |        |        | 0.6063 | 0.5313 |        |
| (6,6)   |        |        |        |        | 0.5156 |        |
| (4,7)   |        |        | 0.5107 | 0.5268 | 0.5446 | 0.5625 |
| (5,7)   |        |        |        | 0.5037 | 0.5134 | 0.5313 |
| (6,7)   |        |        |        |        | 0.5022 | 0.5156 |
| (7,7)   |        |        |        |        |        | 0.5078 |

| OR Alpha |        |        |        |        |        |        |
|----------|--------|--------|--------|--------|--------|--------|
| $(k,n)$  | $t=2$  | $t=3$  | $t=4$  | $t=5$  | $t=6$  | $t=7$  |
| (2,4)    | 0.1613 | 0.3333 | 0.5000 |        |        |        |
| (3,4)    |        | 0.0526 | 0.2500 |        |        |        |
| (4,4)    |        |        | 0.1250 |        |        |        |
| (3,5)    |        | 0.0515 | 0.1364 | 0.2500 |        |        |
| (4,5)    |        |        | 0.0227 | 0.1250 |        |        |
| (5,5)    |        |        |        | 0.0625 |        |        |
| (3,6)    |        | 0.0632 | 0.1364 | 0.2000 | 0.2500 |        |
| (4,6)    |        |        | 0.0185 | 0.0588 | 0.1250 |        |
| (5,6)    |        |        |        | 0.0099 | 0.0625 |        |
| (6,6)    |        |        |        |        | 0.0313 |        |
| (4,7)    |        |        | 0.0190 | 0.0497 | 0.0862 | 0.1250 |
| (5,7)    |        |        |        | 0.0069 | 0.0259 | 0.0625 |
| (6,7)    |        |        |        |        | 0.0043 | 0.0313 |
| (7,7)    |        |        |        |        |        | 0.0156 |

根據對比度(Alpha)的理論結果,接下來我們用以下六張圖來實際實驗我們的成果三是否和我們理論算出來的一樣。



a



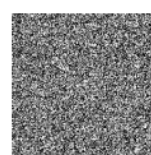
b



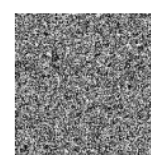
c



d



e



f

| XOR實驗平均Alpha |        |        |        |        |        |        |
|--------------|--------|--------|--------|--------|--------|--------|
| $(k, n)$     | $t=2$  | $t=3$  | $t=4$  | $t=5$  | $t=6$  | $t=7$  |
| (2,4)        | 0.2852 | 0.1739 | 0.4174 |        |        |        |
| (3,4)        |        | 0.1815 | 1      |        |        |        |
| (4,4)        |        |        | 1      |        |        |        |
| (3,5)        |        | 0.1786 | 0.1429 | 0.3984 |        |        |
| (4,5)        |        |        | 0.1435 | 1      |        |        |
| (5,5)        |        |        |        | 1      |        |        |
| (3,6)        |        | 0.1639 | 0.0913 | 0.0768 | 0.2520 |        |
| (4,6)        |        |        | 0.1173 | 0.1190 | 0.3992 |        |
| (5,6)        |        |        |        | 0.1151 | 1      |        |
| (6,6)        |        |        |        |        | 1      |        |
| (4,7)        |        |        | 0.0928 | 0.0633 | 0.0662 | 0.2508 |
| (5,7)        |        |        |        | 0.0834 | 0.0991 | 0.4007 |
| (6,7)        |        |        |        |        | 0.1019 | 1      |
| (7,7)        |        |        |        |        |        | 1      |

| OR實驗平均Alpha |        |        |        |        |        |        |
|-------------|--------|--------|--------|--------|--------|--------|
| $(k, n)$    | $t=2$  | $t=3$  | $t=4$  | $t=5$  | $t=6$  | $t=7$  |
| (2,4)       | 0.1620 | 0.3342 | 0.5009 |        |        |        |
| (3,4)       |        | 0.0540 | 0.2513 |        |        |        |
| (4,4)       |        |        | 0.1256 |        |        |        |
| (3,5)       |        | 0.0520 | 0.1359 | 0.2499 |        |        |
| (4,5)       |        |        | 0.0227 | 0.1251 |        |        |
| (5,5)       |        |        |        | 0.0624 |        |        |
| (3,6)       |        | 0.0643 | 0.1374 | 0.2008 | 0.2501 |        |
| (4,6)       |        |        | 0.0194 | 0.0588 | 0.1250 |        |
| (5,6)       |        |        |        | 0.0099 | 0.0625 |        |
| (6,6)       |        |        |        |        | 0.0311 |        |
| (4,7)       |        |        | 0.0189 | 0.0500 | 0.0861 | 0.1248 |
| (5,7)       |        |        |        | 0.0069 | 0.0258 | 0.0627 |
| (6,7)       |        |        |        |        | 0.0045 | 0.0313 |
| (7,7)       |        |        |        |        |        | 0.0160 |

經過實驗我們可以知道我們的理論值和實驗值是相符的。並且在  $n, k, t$  不同的情況下，每個例子的 Alpha 都是大於 0 的，比成果二的實驗結果好，尤其是  $t=n$  的情況下。

在分析演算法、對比度的時候，我們發現由於成果三演算法的第二部分是在做 XOR 運算，當  $n$  超過  $2k$  的時候，前  $k$  張和第  $k+1 \sim 2k$  張做 XOR 運算會互相抵消，導致第  $2k+1$  張加密影像會被翻成秘密影像，以致洩露秘密。所以我們的演算法的限制是  $n \leq k \leq 2n$ 。

#### 4. 結論

本研究的成果三使得每張加密影像依舊是隨機網格且不能看出秘密影像，疊合之後與秘密影像的光透率能夠提高，更加清晰地辨識出秘密影像，同時增加多一種 XOR 的解密功能，使得此演算法的運用範圍更加廣泛。由於本研究的成果三只能運用在  $n$  小於等於  $2k$  的情況下，所以未來的研究方向將會針對  $n$  大於  $2k$  時設計新的方法。

#### 5. 參考文獻

- [1] O. Kafri and E. Keren, "Encryption of pictures and shapes by random grids," *Optics Letters*, vol. 12, no.6, pp.377-379 (1987).
- [2] Chen, T. H., & Tsao, K. H. (2011). Threshold visual secret sharing by random grids. *Journal of Systems and Software*, 84(7), 1197-1208.
- [3] T.-H. Chen, K.-H. Tsao, Threshold visual secret sharing by random grids, *J. Syst. Software* 84 (7) (2011) 1197–1208.
- [4] Yan, X., Wang, S., Niu, X., & Yang, C. N. (2015). Random grid-based visual secret sharing with multiple decryptions. *Journal of Visual Communication and Image Representation*, 26, 94-104.
- [5] Naor, M., & Shamir, A. (1995). Visual cryptography. *Advances in Cryptology EUROCRYPT'94 Lecture Notes in Computer Science*. In Workshop on the Theory and Application of Cryptographic Techniques, May 9C12 (pp. 1-12).
- [6] Yan, X., Liu, X., & Yang, C. N. (2018). An enhanced threshold visual secret sharing based on random grids. *Journal of real-time image processing*, 14(1), 61-73.