國立交通大學

資訊管理研究所

碩　士　論　文

以圖片攪亂提升 DCT 浮水印演算法的安全性

Enhancing the Security of DCT Watermarking Scheme by Image Scrambling

研 究 生： 朱 國 鼎

指導教授： 蔡 銘 箴 博士

中華民國九十五年七月

以圖片攪亂提升 DCT 浮水印演算法的安全性

學生：朱國鼎　　指導教授：蔡銘箴

國立交通大學 資訊管理研究所 碩士班

摘要

數位浮水印技術乃是利用人眼無法察覺圖片細微變化的弱點，將浮水印訊息直接嵌入原數位圖片中，作為聲明版權之用。

Das 和 Maitra 把浮水印系統當成一個加解密系統，對 CKLS 及 CHU 浮水印演算法進行密碼分析（Cryptanalysis）攻擊，只需降低部份圖片的畫質卻具有相當顯著的攻擊效果。他們利用浮水印數列皆為標準常態分配的特徵，找出如何判斷相異數列的特徵值，接著再以迴歸方程式進行浮水印係數的重建，找出特徵值相符的數列後，再嵌回原來的浮水印係數當中。

由於 Das 和 Maitra 的密碼分析攻擊是針對 CKLS 及 CHU 浮水印演算法，因此在考量不更動原始演算法的情形下，我們便對圖片作改變，先把原始圖片進行圖片攪亂的動作後，再以原始的浮水印演算法嵌入浮水印，嵌入完成後，再將嵌有浮水印的圖片以相同的亂數種子還原。

實驗結果證實，經過攪亂前處理，縱然受到 Das 和 Maitra 所提出的密碼分析攻擊，仍然能夠取出高相關度（correlation）的浮水印數列，證明本研究的設計，可增加以 DCT 為轉換手法的 CKLS 和 CHU 的安全性。

關鍵字

數位浮水印、密碼分析、圖片攪亂

Enhancing the Security of DCT Watermarking Scheme by Image Scrambling

Student: Kuo-Din Chu Advisor: Min-Jen Tsai

Institute of Information Management
National Chiao Tung University

ABSTRACT

Watermarking is a technique where the inserted signal is too subtle to be detected for the copyright protection in video, image or music files.

Das and Maitra regard the watermarking system as an encryption/decryption system. Therefore, they can utilize the cryptanalysis approach to attack CKLS and CHU's schemes which are basically DCT based watermarking. Their approach only slightly affects the image quality but effectively remove the watermark signal. In essence, they focus on the fact that the watermark signal is the random sequence with standard normal distribution. By differentiating the characteristic of different watermark signals, the regression function is utilized to rebuild and substitute the original DCT coefficients. By using this technique, cryptanalysis can decrease the correlation value significantly for CKLS and CHU's techniques.

Since Das and Maitra focus on CKLS and CHU's schemes, the purpose of this study is to modify the image by scrambling techniques to resist the cryptanalysis attacks, without changing CKLS and CHU's methods. The first step is to scramble the image before embedding the watermark. We then embed the watermark in the scrambled image and restore it.

From the experiment results, our approach can still retrieve high correlation values from the watermarked images which are under the cryptanalysis attacks. Therefore, we can conclude our approach can effectively enhance the robustness and security of CKLS and Chu's methods.

Keywords

Digital watermarking, cryptanalysis, image scrambling.

目錄

圖目錄

表目錄

一、序論

1.1 研究背景與動機

隨著數位化時代的來臨，大量的文字、圖片、視訊與音樂等，皆能夠轉換成數位檔案儲存，加上各類消費性電子產品如手機、MP3 Player、DVD Player 的便利性，人們也逐漸習慣使用不佔空間、傳遞快速的數位化檔案。但就另一方面來看，媒體數位化也等於更容易遭受盜版、不法使用，因此，數位浮水印這樣一個保護版權的技術便應運而生。如同密碼學的領域，有加密者與破解者，數位浮水印亦是如此，新的浮水印演算法不斷被提出，各類型的攻擊也層出不窮，有鑑於此，本文便試圖對於 CKLS 及 CHU 以 DCT 為主的演算法進行補強，以期能夠提供更佳的防護效果。

1.2 論文架構

本論文其餘各章節內容摘要說明如下：

第二章

此章為文獻探討，介紹數位浮水印的源起、內涵、類型與研究方向等，並解釋說明何謂密碼分析、密碼分析的類型。

第三章

設計浮水印時必須均衡考量不可視性、強韌性、安全性等要素，任何一環出現弱點，就容易淪為攻擊者攻擊的對像，學者 Das 與 Maitra 便運用浮水印數列為一標準常態分佈的特質，針對 CKLS 及 CHU 浮水印演算法進行密碼分析攻擊，只需降低許些的圖片品質或甚至維持和原圖相似的高品質，就能達顯著地攻擊效果，本章即是在說明此一密碼分析攻擊。

第四章

本章主要針對第三章所提出來的攻擊方式，進行 CKLS 與 CHU 演算法本身的補強，補強的方法便是加入攪亂前處理，此一方法不但有效提升了 CKLS 及 CHU 演算法的安全性，並增加了 CKLS 及 CHU 演算法對於影像處理攻擊的強韌性。

第五章

總結本文內容以及未來可行的研究方向。

二、文獻探討

2.1 數位浮水印

2.1.1 數位浮水印簡介

音樂、圖片、視訊等產品數位化的時代已然來臨。由於數位產品可被無限制的完美複製，又能透過有網際網路傳播，因此，若缺乏完善的數位智慧財產權保護機制，數位產品的著作權將無從防護與確認。圖片數位浮水印技術乃是利用人眼無法察覺圖片細微變化的弱點，將浮水印訊息直接嵌入原數位圖片中，並經由公開的演算法將嵌入的資訊依偵測出來。

為了將一些秘密的訊息發佈出去，通常採用的方式有二種：

一、資訊偽裝（Information Hiding, Steganography）：

藉由其它多媒體（Multimedia）物件將這些訊息掩飾起來，此種方法通常稱之為「資訊偽裝」（Information Hiding, Steganography），而這此負責掩飾秘密訊息的多媒體物件即稱之為「掩護媒體」（Cover Media）。掩護媒體常見的有圖片（Image）、影片（Video）或音訊（audio）。

二、資訊加密（Information Encryption）：

即是直接將卻傳送的訊息打亂，形成密文，使他人無法閱讀，古今中外皆有相當多實際的案例。資訊偽裝有別於資訊加密，資訊偽裝的目的不在於限制對於掩護媒體的存取，而是要保護隱藏的資訊。

數位浮水印可視為資訊偽裝的一種，數位浮水印與一般資訊偽裝最大的差別在於，數位浮水印強調「強健性」（Robustness），即是能夠承受第三方有意或無意的攻擊與破壞。常見的破壞有壓縮、尺寸變化或量化等。

關於數位媒體的資訊隱藏技術，其分類架構圖如下：

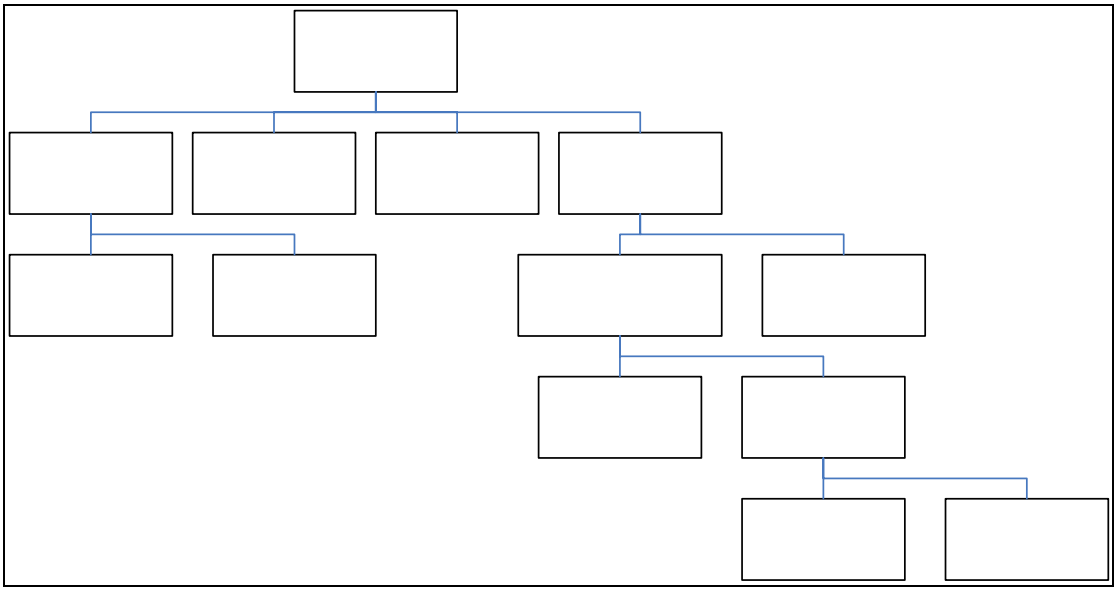


圖 2- 1 資訊隱藏分類架構圖[4]

資訊隱藏包括有隱藏式通道（Covert Channels）、偽裝（Steganography）、匿名（Anonymity）與版權標記（Copyright Marking），依次分述如下：

- 隱藏通道：

  隱藏式通道是指非正式拿來用作通訊的傳輸通道，在網路安全議題當中，常被駭客、木馬程式用於竊取電腦上的資訊或取得控制權。

- 偽裝：

  偽裝是將所要傳遞的機密訊息藏入選定的掩護媒體中，唯有正確的金鑰才能取出其中的秘密訊息。

- 匿名：

  匿名是指訊息傳送時，不洩露出真實傳送者的相關訊息，傳送者能以假名（Pseudonym）傳送隱密的訊息，也能以假名收到回覆。

- 版權標記：

  在數位媒體中藏入所有權相關資訊的方法即稱為版權標記，在目前，最具代表性的就是數位浮水印（Digital Watermarking）。數位浮水印與偽裝的研究上最大的差異在於，偽裝法著重於所嵌入的資訊量，較少討探遭受攻擊的影響，而數位浮水印則強調所嵌入的浮水印數列在受到攻擊後的殘留情況。

2.1.2 良好數位浮水的要素

數位浮水印可以是一群隨機數列或是一張圖片，一個良好的數位浮水印技術通常具有以下幾種要素：

- 不可視的（Invisible）：

  製作浮水印時最基本的要求，就是透過人眼視覺檢視、比較原始圖片與嵌有浮水印圖片時，無法辨別出二張圖片有何不同點，換句話說，就是讓人感覺與原圖幾乎是一模一樣。將浮水印以不可視方式嵌入，除了降低對原圖的品質影響外，也能減少被遭受惡意竄改的機率。在客觀的檢測標上，即是使信號雜訊比（Peak Signal to Noise Ratio, PSNR）大於或等於 30 db（256×256 的圖片）。計算方式如下：

$$PSNR = 10\log_{10}\frac{255^2}{MSE}$$

  其中的均方差（Mean Square Error, MSE）計算方式如下：

$$MSE = \left(\frac{1}{m \times n}\right)\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left(I_{ij} - I'_{ij}\right)$$

  I_{ij} 表示原圖在 (i, j) 位置上的像素值，I'_{ij} 表示處理過後的影像，在 (i, j) 位置上的像素值，m 與 n 表分別代表影像的長與寬。如果二張圖片運算出來的 PSNR 愈高，則表示這二張圖片愈相似。

- 強韌性（Robustness）：

  浮水印的強韌性代表著對於攻擊的忍受能力，禁得起攻擊的浮水印，才能達成它保護所有權的目的。數位媒體的一大特性就是容易遭到修改、複製，因此要如何能達到所嵌入的資訊在傳輸過程中，或是經過過濾處理、有意或惡意的攻擊後，仍不遭受攻擊的影響，實為數位媒體所有權保護的最主要訴求。

- 安全性（Security）：

  浮水印必須具備不可偵測的特性。即使浮水印演算法是公開的，惡意攻擊者也無法藉此找出浮水印資訊。一般來說，安全性常藉由使用私鑰／公鑰（Private key／Public key）來完成，只有合法擁有者才知道金鑰，使用正確的金鑰才能獲得嵌入的訊息。

- 盲擷取（Blind Extraction）：

  非盲擷取的浮水印技術是當針對已含有浮水印保護的媒體，作擷取浮水印的動作時，需要參考到原始媒體，才能順利地擷取出浮水印。而盲擷取乃是指在浮水印取出過程中，不需要原始媒體訊號的協助就可順利取出浮水印。盲擷取的浮水印技術能節省儲存原始媒體的空間，不必特別為原始媒體保存安全性作任何考量。

2.1.3 數位浮水印的類型

數位浮水印就視覺辨識的角度來看，可分為「可視」（Visible）與「不可視」（Invisible）二種，即嵌入後的浮水印，若能為人類肉眼所輕易辨識出來者，稱為可視浮水印，若無法輕易被辨識者（即以上述資訊偽裝方式嵌入），稱為不可視浮水印。可視浮水印的優點在於處理快速，但會有明顯的畫質影響。

由於可視浮水印對於畫面品質影甚大，因此多數的研究都集中於不可視浮水印，但這不代表不可視浮水印不會破壞掩護媒體，要視嵌入位置與嵌入強度而定，設計不良，往往會帶來比可視浮水印更多的失真。

而就破壞忍受程度來看，數位浮水印可分為「強韌型」（Robutness）與「易碎型」（Fragile）。強健型浮水印多半應用於版權保護，將數位資料嵌入浮水印，若它日遭到不法使用時，可將浮水印擷取出來，聲明版權。由於掩護媒體只要受到輕微的破壞，易碎型浮水印就能夠反應出來，因此易碎型浮水印常應用於竄改測偵。

就嵌入與偵測的方式來看，浮水印的嵌入、偵測若是採用直接更改圖片亮度或灰階值，通稱為「空間域機制」（Spatial Domain Mechanism）；而更改以數學轉換過的係數，則稱為「轉換域機制」（Transform Domain Mechanism），或是稱之為「頻率域機制」（Frequency Domain Mechanism）。

所謂空間域（Spatial Domain）的技術是以直接改變或調整圖片的像素值來達成嵌入的目的。最典型的代表技術就是 LSB（Last Significant Bit）嵌入。以一張灰階圖片為例，圖片中的每個像素值佔 1 個 Byte（8 個 Bit）。每 1 個 Byte 當中最右邊的 Bit 稱為 LSB。此做法為直接修改像素值的較不重要的部份，更換為我們的浮水印數列，以達到藏入數位浮水印之目的，方法快速，同時因為修改的是位元組中最後面的幾個位元，對整個位元組的影響不大，因此對於畫質的影響

也很小。

頻率域技術不是直接修改像素值以藏入數位浮水印，而是先將圖片以「離散餘弦轉換」(Discrete Cosine Transform, DCT)或「離散小波轉換」(Discrete Wavelet Transform, DWT)，解析出低頻、中頻與高頻訊號後，再行嵌入浮水印。在頻率域嵌入浮水印的優點是能選擇合適的頻率來嵌入浮水印。一般而言，頻率域浮水印相較於空間域浮水印，更能夠忍受雜訊、壓縮等影像處理的攻擊。

由圖 2-2 可看出，一張圖片如果我們用 DCT 或 DWT 轉換後，其頻率域圖形會所有差異。

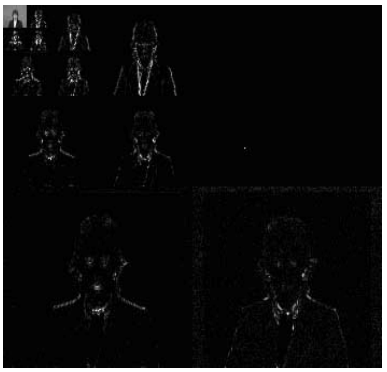
原始圖片	經 DCT 後之頻率域	經 4 階 DWT 後之頻率域
		

圖 2- 2 DCT 與 DWT 頻率域圖形差異圖

如圖 2-3 所示，一個圖片中，若某個區域的內容較平滑，則該區域為低頻；反之，區域中的像素彼此間差異甚大時，該區域為高頻，因此如果我們能夠只在高頻區嵌入浮水印，肉眼就無法輕易的察覺出來。

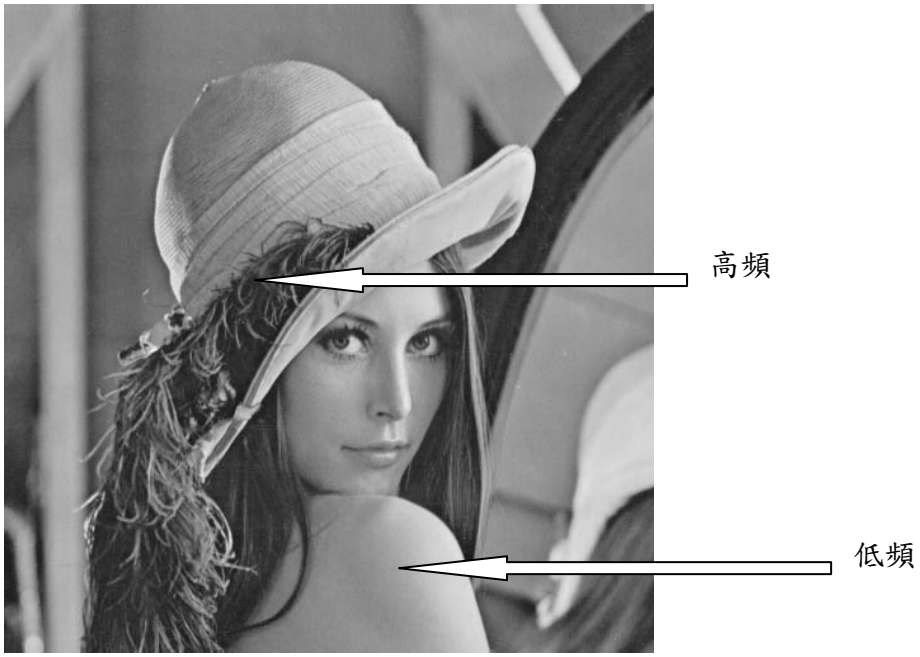


圖 2- 3 高低頻區示意圖

圖 2-4 為一般浮水印系統的架構圖，首先，我們將浮水印數列（w）以可視或不可視的方式嵌入到掩護媒體（CM）當中，使掩護媒體藏有浮水印訊息（CMw）。接著，掩護媒體在傳輸或散佈過程中，可能會受到有意或無意的破壞或攻擊，使浮水印數列產生變化（CMw’），最後可經由人眼或浮水印擷取演算法，聲明出浮水印數列的存在。

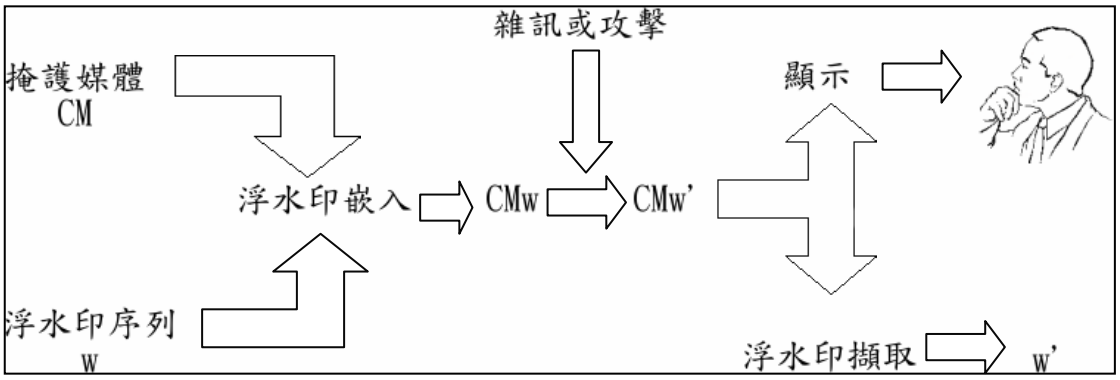


圖 2- 4 一般嵌入與擷取模型[5]

2.1.4 數位浮水印的應用

影像數位浮水印的應用如下：

1、版權保護與認證（Copyright Protection and Authentication）：

版權持有者可在產品中嵌入專屬的資訊，例如圖片或是以特定方式產生的亂數。當有著作權糾紛時，原作者即可透過公開的偵測演算法從產品中偵測出與版權資訊相關的浮水印，聲明其版權。

2、複製控制（Copy Protection）：

將可否複製標籤直接作為浮水印嵌入產品中，燒錄機可憑此標籤判定能否複製[12]。

3、廣播監控（Broadcast Monitoring）：

將浮水印嵌入影片當中，監控系統可監視所有播放的情形，以便對播放有浮水印的影片的頻道收取費用 [8]。

4、篡改偵測（Tampering Detection）：

將產品本身的重要資訊，如圖片的特徵，當成浮水印嵌入產品中，或是以易碎型浮水印技術嵌入浮水印，當無法偵測出浮水印時，就能知道圖片已遭到篡改。

5、影片品質預測（Video Quality Estimation）：

藉由浮水印嵌入至影片當中，藉由浮水印被破壞的情形來反應影片品質[7]。

2.2 密碼分析

2.2.1 密碼學

戰爭是密碼學（Cryptography）的推手，西元前五世紀，斯巴達人便發明了「移轉密碼」(Transposition Cipher)。移轉密碼的基本原理是將原文中的字元調動位置，使得訊息的攔截者無法正確解讀當中的意義。在加密方式不斷推陳出新後，開始有人反其道而行，研究如何破解這些被加密的訊息，這種技術的學問便被稱為「密碼分析」(Cryptanalysis)。頻率分析法（Frequency Analysis）便是最早的密碼分析法。若現有一密文是以單一字元替代法（Mono alphabetic substitution cipher）加密，即是將原本的字元以另一個字元或是符號來替代，假設已知原文的語言結構為英語，在英語當中最常出現的字元是 E，因此在加密的訊息中，最常出現的符號或字元，有非常大的機率是 E(12.7%)，其次是 T(9%)、

A（8.2%）及 O（7.5%）…等[15]，依此類推，即有機會發現出原始訊息的涵義。

2.2.2 密碼系統的要素

一個完整的密碼系統，不外乎要達到下面的四個目標：

1. 私密性（Privacy）：

   確保在通訊的過程之中，別人無法偷窺到資料的內容。

2. 完整性（Integrity）：

   確保在從來源到最終目的地的通訊過程之中，資料不會被改變。

3. 身分確認（Authentication）：

   檢驗並確定通訊的雙方之真正身分。

4. 不可否認（Non-Repudiation）：

   不可否認資料之發送。

數位浮水印與密碼學要求可以說是一致的，同樣都是為了保護秘密而存在，同樣也要求了私密性（不可視性）與完整性（強韌性），但數位浮水印提供了更佳的保護措施，不似一般密文訊息，經過解密後就不再受到保護，但是數位浮水印即使經過擷取，還是依然能夠承擔保護版權的作用。

2.2.3 密碼分析（Cryptanalysis）

傳統加密機制的破解有二種方法：

1、密碼分析（Cryptanalysis）：

這種型式的破解是利用加密演算法的特性，試圖推算出原文或加密使用的金鑰。

2、暴力破解（Brute-force Attack）：

密碼分析者嘗試所以與隱密訊息有關的鑰匙，直到可以把密文轉為有意義的明文為止。

依分析者擁有明文資訊的多寡，可將密碼分析分為表 2-1 所列之類型：

表 2- 1 密碼分析的類型

類型	分析者擁有的資訊
Cipher-text only	加密演算法 密文
Known plaintext	加密演算法 密文 一組或多組的密文－明文組合
Chosen plaintext	加密演算法 密文 分析者自選的明文及某個金鑰所產生的密文

表 2-1 所示各類型當中，尤以「只知密文」(Cipher-text only)是最難破解的。如果暴力解是不可行的，分析者就只能從分析密文本身來著手，通常會運用統計的方式來嘗試破解。想要用這種方式破解時，破解者必須對被保護的明文有一些基本的概念。

三、浮水印密碼分析攻擊

上一章節提及，一個完善的浮水印演算法必須具備的特性有：不可視(Invisible)、強韌性(Robustness)、明確性(Unambiguous)、安全性(Security)…，等。其中的不可視性、強韌性有相互取捨的考量。合理的情況下，當嵌入的資訊量增加之後，自然能增加浮水印的強韌性，卻也因為藏入的資訊量大，使得其他人容易察覺到浮水印的存在。相反地，當嵌入的資訊量減少時，其他人自然不容易察覺出浮水印的存在，卻也因為藏入的資訊量較少，因而降低浮水印的強韌性。在設計浮水印演算法時，不可視性和強韌性這二項因素不斷被考慮，而明確性因為是法律判定上的重要依據，也會在進行「浮水印存在與否」的判別上被提及。

一般來說，數位浮水印可能遭遇到的攻擊分述如下：

- 訊號處理：

  訊號處理包含了數位－類比訊間的相互轉換、重新取樣（Re-sampling）、量化（Quantization）等。

- 幾何變形：

  幾何變形包括旋轉、切割、放大或縮小等。

- 共謀攻擊（Collusion attack）：

  將許多嵌有浮水印的副本不斷的組合起來，破壞當中的浮水印。

多數浮水印演算法提出者都會以上述的各項攻擊方式來測試自己浮水印演算法的強韌性與可靠度，卻都較少以密碼分析（Cryptanalysis）的方式對演算法進行安全性測試。針對到浮水印的安全性議題，2004 年，Das 和 Maitra[1]把浮水印系統當成一個加解密系統，對浮水印演算法進行密碼分析，該文主要是針對CKLS[2]的浮水印架構進行分析。

3.1 CKLS 浮水印演算法：

1997 年，Cox、Kilian、Leighton 與 Shamoon 四位學者共同提出「類似展頻」（Spread spectrum like）的浮水印嵌入演算法。由於圖片的高頻區訊號容易在經過一般濾波、壓縮後除去，因此 CKLS 演算法將浮水印訊息嵌入至圖片的低頻區當中。每個頻率域當中的係數都有視覺容量（Perceptual capacity），即是能夠容忍改變而又不產生畫質衰減的程度。他們將圖片視為一個通訊通道，而浮水印則

是一個在此通道中傳輸的訊號。

嵌入程序：

圖 3-1 為嵌入流程示意圖，首先我們必須將原始圖形，經由 DCT 轉換，由空間域轉換成頻率域，接下來再將浮水印加到圖形頻率域中絕對值最大的前 N 個 AC 係數值(不包括 DC 值)。最後再將此圖形利用 Inverse DCT 轉換回空間域，完成整個加浮水印的程序。

$$V_i^{'} = IDCT\left(DCT\left(V_i\right)+\alpha X_i\right)$$

$V_i^{'}$：加入浮水印後的影像

α：強度係數

X_i：浮水印

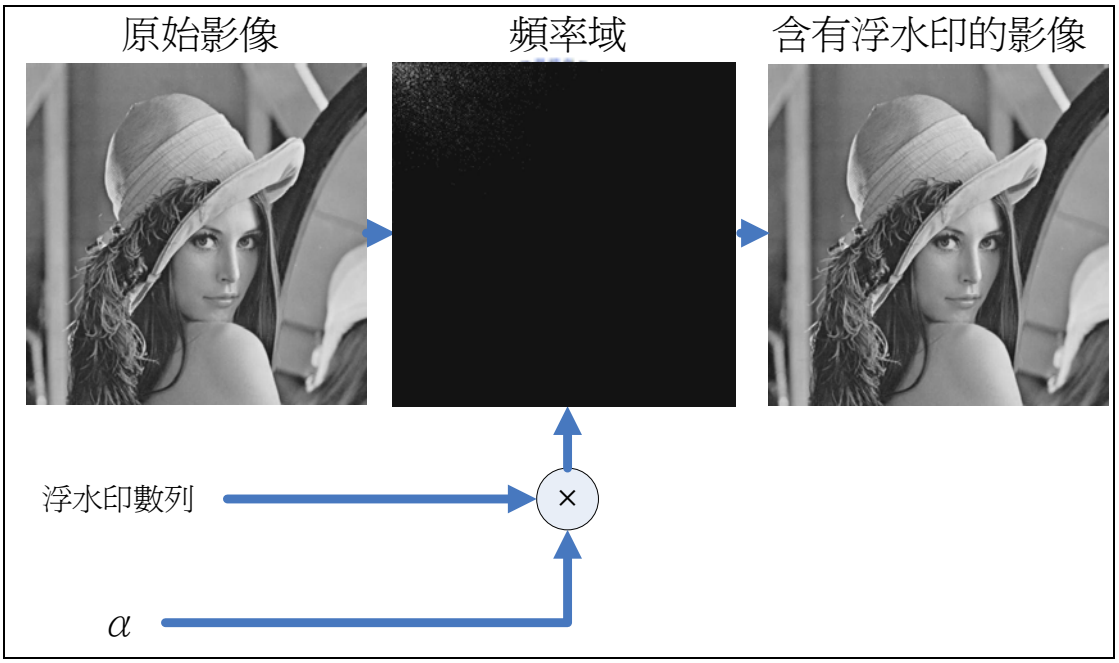


圖 3- 1 CKLS 嵌入流程圖

浮水印數列：

浮水印是一串實數的數字序列，而這個數字序列的產生必須滿足標準常態分配，即浮水印數列 $X \sim N(0,1)$，平均值為 0，變異數為 1，以確保每個浮水印數列彼此之間都為獨立且無關，避免 Collusion attack。

擷取流程：

在擷取浮水印的過程中，我們反向推導上面公式，得到解浮水印公式為：

$$X_i = \left(\frac{DCT(V_i')}{DCT(V_i)} - 1 \right) \frac{1}{\alpha}$$

V_i' ：含有浮水印的影像

V_i ：原始影像

圖 3-2 為浮水印擷取與鑑別流程圖。在浮水印的偵測方面，我們先將待測的圖片經由 DCT 轉換至頻率域，接著利用 $X_i = \left(\frac{DCT(V_i')}{DCT(V_i)} - 1 \right) \frac{1}{\alpha}$ 解出浮水印 X^* 。

由於解出來的浮水印我們並不知道它是否與原始訊息相符，因此經常利用 Similarity 或 Normalized correlation 來做為浮水印真偽的判斷，公式如下：

$$Sim(w', w) = \frac{\sum_{m=1}^{N} w_m' w_m}{\sqrt{\sum_{m=1}^{N} w_m' w_m'}}$$

$$Correlation(w', w) = \frac{\sum_{m=1}^{N} w_m w_m'}{\sqrt{\sum_{m=1}^{N} w_m^2 \sum_{m=1}^{N} w_m'^2}}$$

w：原始浮水印訊號

w’：擷取出來的浮水印訊號

我們將解出來的浮水印 X*以及原始浮水印 X 代入 Sim 中，如果得到的結果大於某一個特定的門檻值，則我們便可以宣稱這兩個浮水印是相同的。反之，如果小於某一特定的門檻值，則這兩個浮水印便不相同。

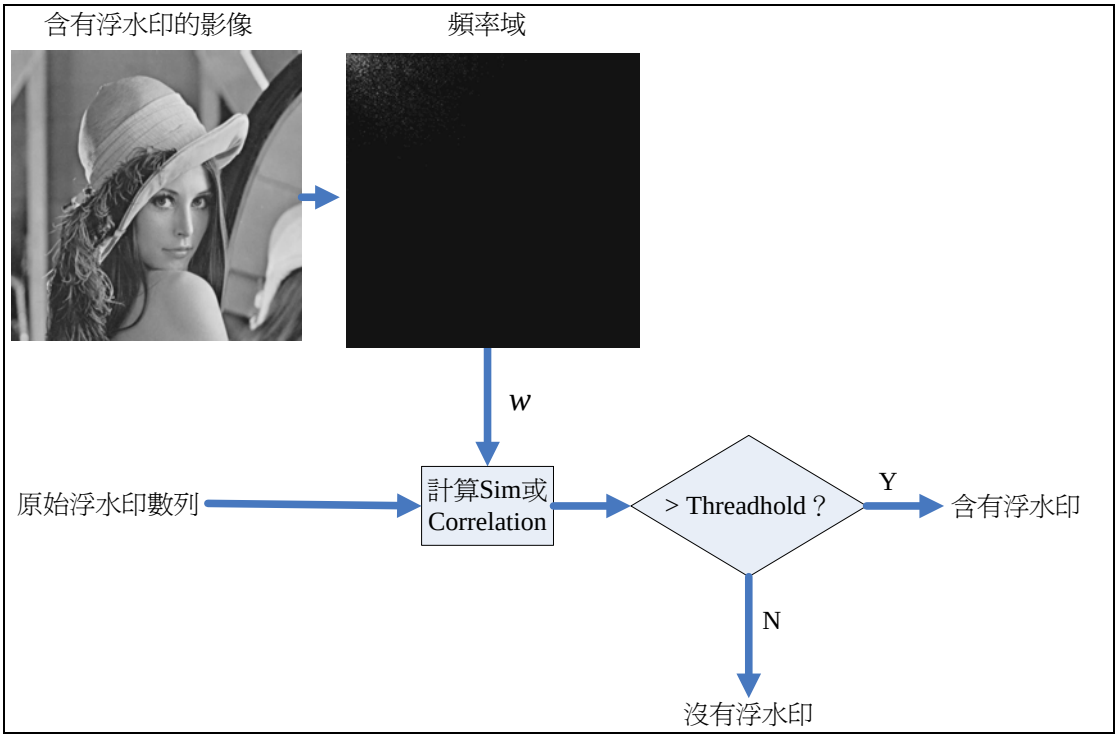


圖 3- 2 CKLS 擷取流程圖

3.2 CHU 浮水印演算法

2003 年，Wai C. Chu 提出了使用 Subsampling 的浮水印演算法[14]，給予一張圖 v，$v[n_1, n_2]$, $n_1 = 0, \ldots, N_1 - 1$, $n_2 = 0, \ldots, N_2 - 1$。首先先將圖 v 以下列方式切割成四張 Subimage：

$$v_1[n_1, n_2] = v[2n_1, 2n_2]$$

$$v_2[n_1, n_2] = v[2n_1 + 1, 2n_2]$$

$$v_3[n_1, n_2] = v[2n_1, 2n_2 + 1]$$

$$v_4[n_1, n_2] = v[2n_1 + 1, 2n_2 + 1]$$

$$n_1 = 0, \ldots, N_1/2 - 1, \; n_2 = 0, \ldots, N_2/2 - 1$$

即如圖 3-3 所示，將原圖以 2x2 個 pixel 為一個單位，每單位中的每個 pixel 分別分配到各 Subimage 當中。

v_1	v_2	v_1	v_2	v_1	v_2	v_1	v_2
v_3	v_4	v_3	v_4	v_3	v_4	v_3	v_4
v_1	v_2	v_1	v_2	v_1	v_2	v_1	v_2
v_3	v_4	v_3	v_4	v_3	v_4	v_3	v_4
v_1	v_2	v_1	v_2	v_1	v_2	v_1	v_2
v_3	v_4	v_3	v_4	v_3	v_4	v_3	v_4
v_1	v_2	v_1	v_2	v_1	v_2	v_1	v_2
v_3	v_4	v_3	v_4	v_3	v_4	v_3	v_4

圖 3- 3 Subimage 分配圖

圖 3- 4 Subsampling 後的結果

嵌入流程：

- Step1：將 Subimage 進行 DCT 轉換。.
- Step2：隨機挑出二張 Subimage，例如（1, 4）或（2, 4），進行相同位置的 DCT 係數比較，比較的位置是依 zigzag 順序決定（DC 值不參與比較）。

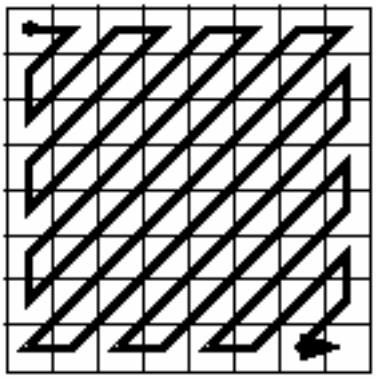


圖 3- 5 zigzag 順序圖

- Step3：令挑出來的二個係數一個為 V_i，另一個為 V_j，$V=\frac{V_i+V_j}{2}$，若 $\left|\frac{V_i-V_j}{V}\right|\geq 6\alpha$，則不對 V_i 與 V_j 嵌入浮水印。反之，將浮水印嵌入到 V_i 與 V_j 當中。嵌入方式如下：

$$V_i^{'}=V(1+\alpha X)$$
$$V_j^{'}=V(1-\alpha X)$$

浮水印數列 X 為標準常態分配，即 $\overline{X}=0$，$\delta_X=1$。

Step4：將 Subimage 組合回原圖

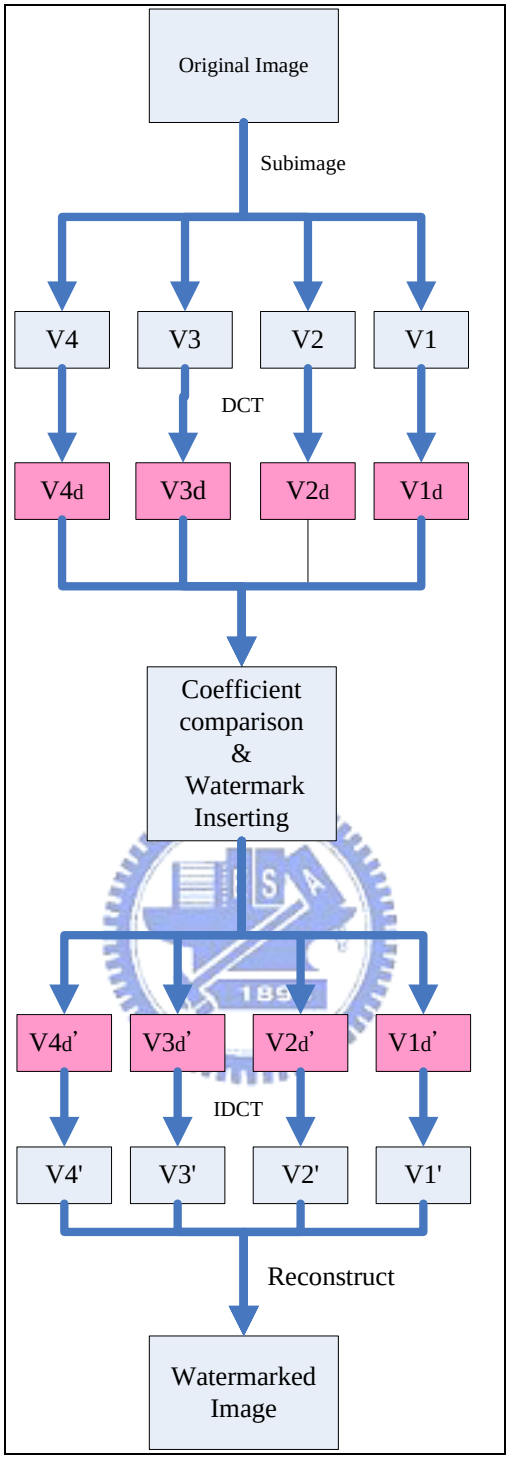


圖 3- 6 CHU 嵌入示意圖

由於每個浮水印數值嵌入的 Subimage 並是不固定的，因此 Subimage 的嵌入組合，例如（1, 4）、(2, 3)、(3, 1)、…等，可以成為一把金鑰，縱然浮水印的嵌入與取出演算法是公開的，但不具有金鑰者也無法順利地將浮水印取出。

取出流程：

- Step1：Decode 端依同樣的方式將嵌有浮水印的圖片分成四張 Subimage。
- Step2：依據嵌入時的位置，將嵌有 Watermark 的係數取出。
- Step3：令挑出來的二個係數一個為U_i，另一個為U_j，依下述方式取出浮水印：

$$X' = \frac{1}{\alpha}\left(\frac{U_i - U_j}{U_i + U_j}\right)$$

表 3- 1 Subimage 間的相似程度

Subimage i	Subimage j	PSNR
1	2	29.550705
1	3	32.281391
1	4	28.079046
2	3	29.261147
2	4	32.422520
3	4	29.541481

由表 3-1 可知，因為 Subimage 間彼此的相似度很高，$V_i \approx V_j$，$i \neq j$，透過下列算式便可知，我們不需要原始圖片就能擷取出浮水印。

$$V_1'[Kn] = V_1[Kn](1 + \alpha X[n])$$
$$V_2'[Kn] = V_2[Kn](1 - \alpha X[n])$$
$$V_1'[Kn] - V_2'[Kn] = V_1[Kn] - V_2[Kn] + \alpha X[n](V_1[Kn] + V_2[Kn])$$
$$\approx 2V_1[Kn]\alpha X[n]$$
$$\because V_1'[Kn] - V_2'[Kn] = V_1(1 + \alpha X[n]) + V_2[Kn](1 - \alpha X[n])$$
$$\approx V_1[Kn](1 + \alpha X[n] + 1 - \alpha X[n])$$
$$\approx 2V_1[Kn]$$
$$\therefore V_1'[Kn] - V_2'[Kn] \approx (V_1'[Kn] + V_2'[Kn])\alpha X[n]$$
$$\frac{V_1'[Kn] - V_2'[Kn]}{V_1'[Kn] + V_2'[Kn]}\frac{1}{\alpha} \approx X[n]$$

3.3 CKLS 密碼分析攻擊

根據[1]，CKLS 演算法存在著很大的安全缺口，攻擊者可以利用此一弱點，輕易將浮水印資訊移除。Das 和 Maitra 所提出的攻擊方法屬於 2.2.3 節所述之「只知密文攻擊」（Cipher text-only attack），即攻擊者擁有的資訊只有嵌入與取出演算法，以及一張嵌有浮水印的圖片。攻擊流程如下：

- Step1：將 Watermarked image 進行 DCT 轉換並排序，並找出頻率域中絕對值最大的前 N 個 AC 係數值
- Step2：保留最大的 50 個係數不變，其餘$(N-50)$個係數遞增排序後進行迴歸（Regression）分析，找出一 3 階迴歸方程式。由於排序後過的數值為一單調遞增，因此可以找出相當高相關係數的迴歸方程式。

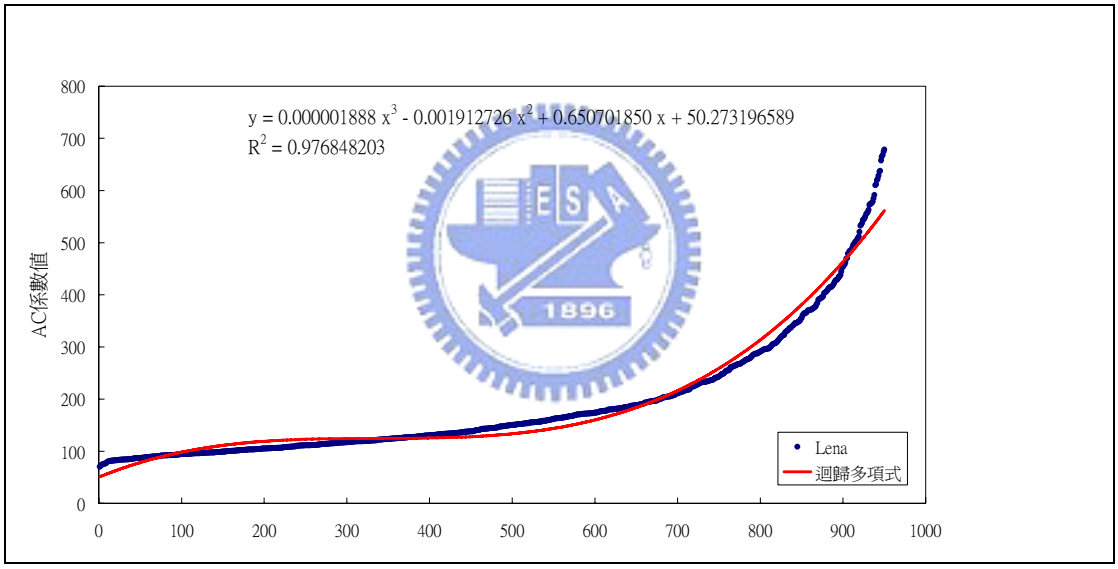


圖 3-7 不含有最大的５０個係數時的曲線圖

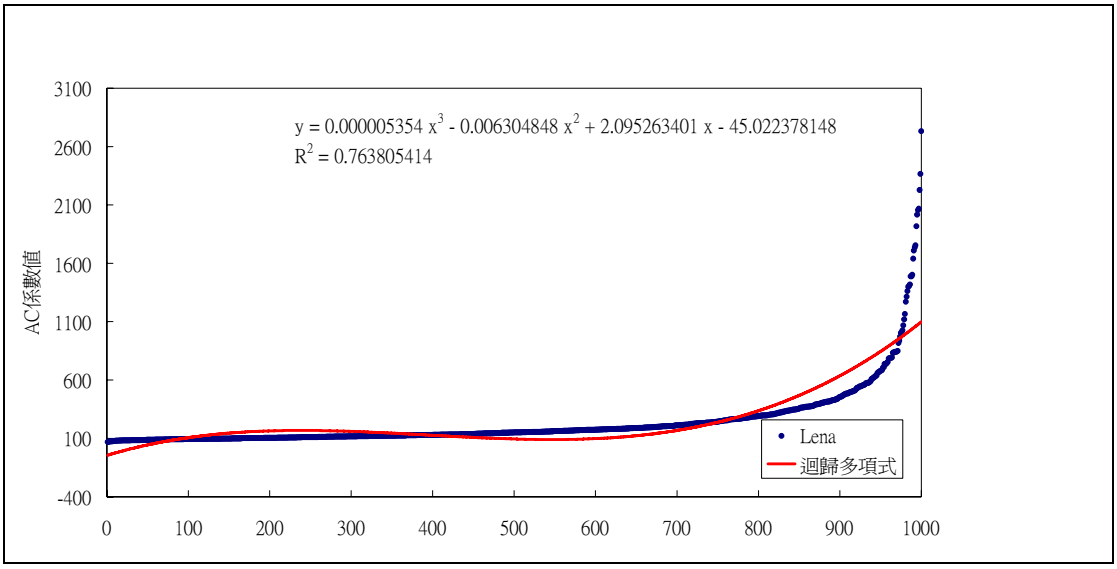


圖 3- 8 含有最大的５０個係數時的曲線圖

由圖 3-7 及圖 3-8 可看出，若不先將最大的 50 個係數先捨去，會降低迴歸曲線與係數分配曲線的相關係數。

- Step3：透過稍微修改迴歸方程式中的係數值，產生 M 張 Polynomial image。保留其中相較於 Watermarked image 之 PSNR 值大於 30db 的 Polynomial image。
- Step4：將各張 Polynomial image I_d^j 與 watermarked image I_d^i 進行統計分析，找出一組平均值為 α ，變異數為 2 的數字序列 $s_d^{(i,j)}$ 。

$$s_d^{(i,j)}[k] = \left(I_d^{(i)}[k] - I_d^{(j)}[k]\right) / \alpha I_d^{(j)}[k]$$

- Step5：將 $s_d^{(i,j)}$ 視為浮水印，嵌入至 watermarked image 中，產生 Attacked image。

3.4 密碼分析定理：

令 $x_1,...,x_n$ 為原始圖片訊號，現有一組浮水印數列 $v_1,...,v_n$ ，其平均值為 $\bar{v}$ ，標準差為 δ_v ，我們將它嵌入到圖片當中，因此一嵌有浮水印的圖片可視為：

$$x_1(1+\alpha v_1),...,x_t(1+\alpha v_t)$$

考慮另一組浮水印數列 $m_1,...,m_n$ ，其平均值為 $\bar{m}$ ，標準差為 δ_m ，同樣的嵌入至圖片後可得：

$$x_1(1+\alpha m_1),...,x_t(1+\alpha m_t)$$

由於二組浮水印間的關係常被假設為無關（Uncorrelatedness），因此$m_1,...,m_t$與$v_1,...,v_t$，同樣地，我們可以令它們之間的關係為無關。為了了解$x_1(1+\alpha v_1),...,x_t(1+\alpha v_t)$及$x_1(1+\alpha m_1),...,x_t(1+\alpha m_t)$二者間的關係，我們以下列的方式進行分析：

$$\frac{(x_k(1+\alpha m_k)-x_k(1+\alpha v_k))}{x_k(1+\alpha v_k)} \quad ,k=1,...,t$$

其平均值為$\overline{m}-\overline{v}+\frac{\alpha}{t}\sum_{k=1}v_k^2$，變異數為$\frac{\sum_{k=1}^{t}(m_k-v_k)^2-(\overline{m}-\overline{v})^2}{t}$

以下為平均值的推演流程：

1. $$\frac{(x_k(1+\alpha m_k)-x_k(1+\alpha v_k))}{x_k(1+\alpha v_k)}=\frac{m_k-v_k}{1+\alpha v_k}$$

2. 若α很小，則$\frac{1}{1+\alpha_K}\approx 1-\alpha v_k$

$$\frac{m_k-v_K}{1+\alpha_K}=\frac{1}{1+\alpha_K}(m_k-v_K)\approx(m_k-v_K)(1-\alpha v_k)$$

3. $\frac{m_k-v_K}{1+\alpha_K}$的平均數近似於$\frac{\sum_{k=1}^{t}(m_k-v_K)(1-\alpha v_k)}{t}$

$$\frac{\sum_{k=1}^{t}(m_k-v_K)(1-\alpha v_k)}{t}=\frac{1}{t}\sum_{k=1}^{t}(m_k-v_k-\alpha m_k v_k+\alpha v_k^2)$$

$$=\overline{m}-\overline{v}+\frac{\alpha}{t}\sum_{k=1}v_k^2$$

以下為變異數的推演流程：

1. $$\frac{(x_k(1+\alpha m_k)-x_k(1+\alpha v_k))}{x_k(1+\alpha v_k)}=\frac{m_k-v_k}{1+\alpha v_k}$$

2. 若α很小，則$\frac{m_k-v_k}{1+\alpha v_k}\approx m_k-v_k$

3. $\frac{m_k - v_k}{1+\alpha v_k}$ 的變異數便會近似於 $\frac{\sum_{k=1}^{t}(m_k - v_k)^2 - (\bar{m} - \bar{v})^2}{t}$，標準差為

$$\sqrt{\frac{\sum_{k=1}^{t}(m_k - v_k)^2 - (\bar{m} - \bar{v})^2}{t}}$$

推論：

考慮二組浮水印數列 $m_1,...,m_t$ 與 $v_1,...,v_t$，皆由一符合標準常態分配（平均值為 0，標準差為 1）的亂數產生器所產生，計算 $\frac{(x_k(1+\alpha m_k) - x_k(1+\alpha v_k))}{x_k(1+\alpha v_k)}$ 的平均值與標準差。

- 平均值：

$$\bar{m} - \bar{v} + \frac{\alpha}{t}\sum_{k=1} v_k^2$$

$$= 0 - 0 + \alpha\frac{1}{t}\sum_{k=1}^{t} 1$$

$$= \alpha$$

- 標準差：

$$\sqrt{\frac{\sum_{k=1}^{t}(m_k - v_k)^2 - (\bar{m} - \bar{v})^2}{t}}$$

$$= \sqrt{\frac{\sum_{k=1}^{t}(m_k - v_k)^2 - (0-0)^2}{t}}$$

$$= \sqrt{\frac{\sum_{k=1}^{t}(m_k - v_k)^2}{t}}$$

$$= \sqrt{\frac{\sum_{k=1}^{t}(m_k^2 - m_k v_k + v_k^2)}{t}}$$

$$= \sqrt{\frac{1}{t}\sum_{k=1}^{t} m_k^2 + \frac{1}{t}\sum_{k=1}^{t} v_k^2}$$

$$= \sqrt{1+1}$$

$$= \sqrt{2}$$

由以上的證明可得知，若我們首先取得一張嵌有浮水印的圖片 Watermarked image，並以修改迴歸方程式的方式產生許多張與 Watermarked image 相似的 Polynomial image，將 Watermarked image 與 Polynomial image 嵌有浮水印的數值取出，並以 $\frac{(x_k(1+\alpha m_k) - x_k(1+\alpha v_k))}{x_k(1+\alpha v_k)}$ 的方式進行統計分析後，若能得到一組平均值為 α，標準差為 $\sqrt{2}$ 的數列 s_k，我們就能夠將 Polynomial image 視為一張嵌有不同浮水印數列的 Watermarked image。

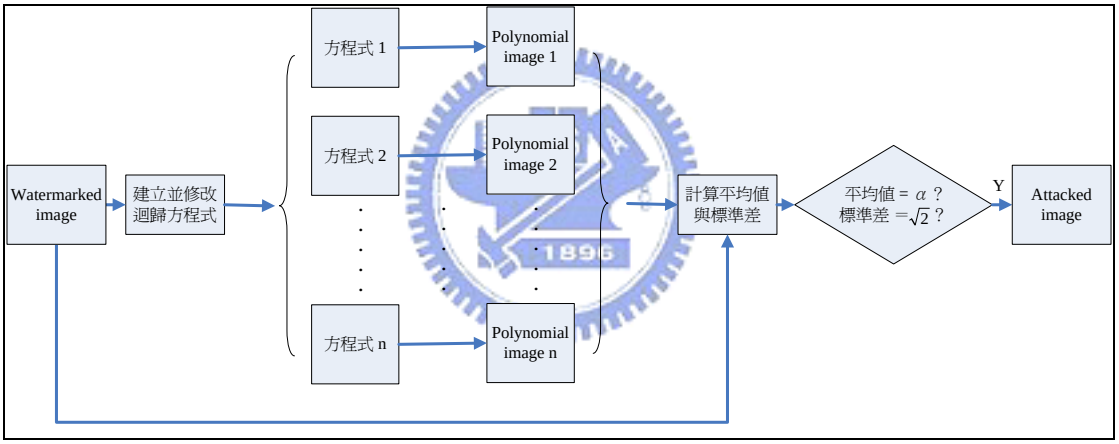


圖 3- 9 密碼分析攻擊流程圖

3.5 攻擊實驗

以下我們將根據上述的方法進行攻擊實驗，使用與[3]相同的設定值，即浮水印嵌入長度為 1000，強度設為 0.1。實驗所使用的圖片如圖 3-10。

			
Bridge	Couple	Elaine	F16
			
Fishboat	Goldhill	Lena	Peppers

圖 3- 10 實驗所採用的圖片

表 3- 2CKLS 密碼分析攻擊結果表

圖片名稱	張數	平均 PSNR（w, a）	平均 correlation	correlation[3]	PSNR[3]
Bridge	110	30.546389	0.171251	0.210	32.30
Couple	96	31.779536	0.116413	0.187	33.12
Elaine	99	30.695026	0.149967	0.232	32.07
F16	253	30.497470	0.091007	0.174	32.93
Fishboat	97	31.220406	0.118499	0.180	32.43
Goldhill	425	32.665145	0.095310	0.171	34.83
Lena	164	31.152823	0.090569	0.178	33.88
Peppers	114	30.126955	0.139663	0.181	31.93

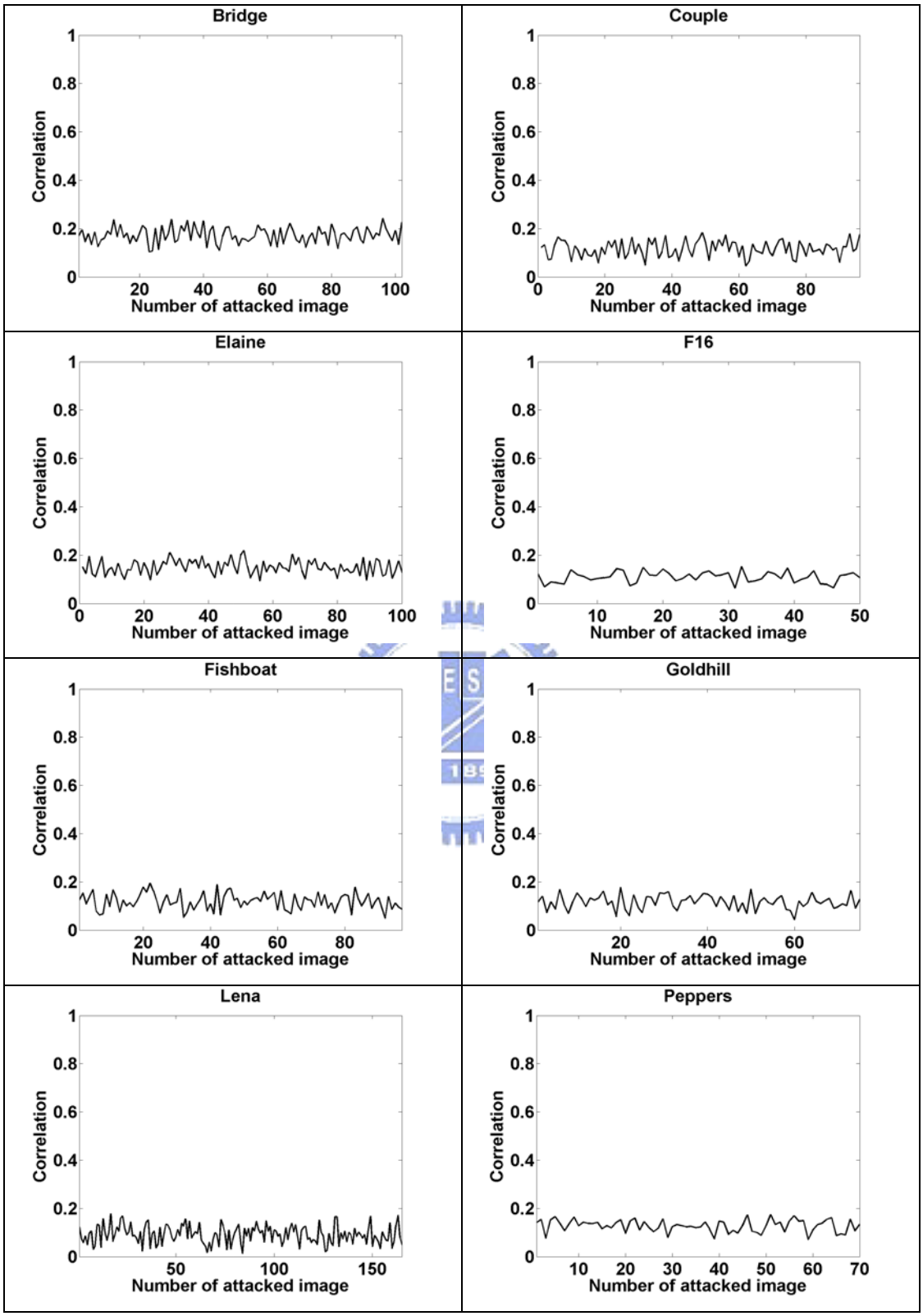


圖 3- 11 CKLS 密碼分析攻擊結果圖

表 3- 3 CHU 密碼分析攻擊結果表

圖片名稱	Subimage1 的張數	Subimage2 的張數	Subimage3 的張數	Subimage4 的張數	平均 correlation	平均 PSNR
Bridge	127	24	259	117	0.129702	35.16832
Couple	384	1107	147	64	0.105081	34.75419
Elaine	335	252	427	341	0.153919	34.86407
F16	142	360	262	320	0.085799	34.60532
Fishboat	198	344	77	32	0.044144	35.56868
Goldhill	60	140	310	149	0.024452	37.10776
Lena	15	157	65	33	0.107589	35.09609
Peppers	523	526	428	380	0.01631	35.15919

由表 3-2 及表 3-3 的實驗結果可以明顯看出，遭受攻擊的圖片，其浮水印的 correlation 都會降到 0.2 以下。

3.6 JPEG 壓縮攻擊

本節中我們採用最常見的 JPEG 壓縮對 CKLS 浮水印演算法進行攻擊，攻擊圖片依序為 Lena、Peppers 及 Fishboat，結果分別如表 3-4、表 3-5 及表 3-6：

表 3- 4 Lena 經 JPEG 壓縮攻擊後的結果

JPEG 品質	correlation	Sim	PSNR（Original image, Attacked image）
100	0.999522	30.998795	34.831482
90	0.998261	30.959661	33.865082
80	0.993566	30.814070	32.920021
70	0.983610	30.505302	32.264618
60	0.973384	30.188158	31.681620
50	0.964672	29.917946	31.227938
40	0.943976	29.276083	30.740879
30	0.897801	27.844059	30.180431
20	0.805601	24.984600	29.293552
10	0.528603	16.393888	27.426920
1	0.035235	1.092753	17.201916

表 3- 5 Peppers 經 JPEG 壓縮攻擊後的結果

JPEG 品質	correlation	Sim	PSNR（Original image, Attacked image）
100	0.999586	31.000757	33.648506
90	0.998419	30.964563	32.901035
80	0.994815	30.852787	32.236938
70	0.987977	30.640739	31.788435
60	0.980032	30.394323	31.333050
50	0.969595	30.070648	31.003012
40	0.956847	29.675280	30.613396
30	0.919209	28.507986	30.133951
20	0.850471	26.376165	29.283171
10	0.599955	18.606775	27.488588
1	0.085544	2.653031	16.513758

表 3- 6Fishboat 經 JPEG 壓縮攻擊後的結果

JPEG 品質	correlation	Sim	PSNR（Original image, Attacked image）
100	0.999530	30.999044	32.855469
90	0.997592	30.938931	32.010094
80	0.992827	30.791147	31.110771
70	0.982719	30.477663	30.451683
60	0.971138	30.118490	29.907721
50	0.956985	29.679567	29.449350
40	0.936007	29.028942	28.918043
30	0.885984	27.477552	28.271402
20	0.784435	24.328144	27.360588
10	0.525643	16.302094	25.613794
1	-0.012127	-0.376113	17.742254

由表 3-4、表 3-5 及表 3-6 的結果可以發現，JPEG 欲將 correlation 攻擊至 0.2 以下時，PSNR 卻需要降到 17db 以下。由以上實驗結果可知，Das 和 Maitra 的提出的密碼分析攻擊是一個相當有效的攻擊策略，當 correlation 下降至 0.2 以下時， PSNR 相較於原圖仍能大於 29db。

四、利用圖片攪亂提升安全性

4.1 CKSL演算法的改進

4.1.1 圖片攪亂

浮水印演算法的安全性通常可透過使用金鑰、紀錄浮水印嵌入過程的資訊（浮水印藏入的位置、浮水印產生的方式…，等）來達成。

除了增加額外的金鑰之外，有些浮水印演算法利用 Chaos 的觀念[11]，先對圖片進行攪亂的前置處理，再將浮水印嵌入至攪亂的圖片中，浮水印資訊嵌入後，再將圖片還原。

由於圖片攪亂類似於亂數重排，我們只需要使用記錄打亂時的亂數種子即可，需要增加額外的空間儲存空間也僅是一個整數的大小，便能提高安全性，同時，由於 Das 和 Maitra 的密碼分析攻擊藉由是重建嵌有浮水印的係數，因此，若我們能夠打亂嵌入位置，就能夠增加分析的困難。

4.1.2 浮水印演算法的改進

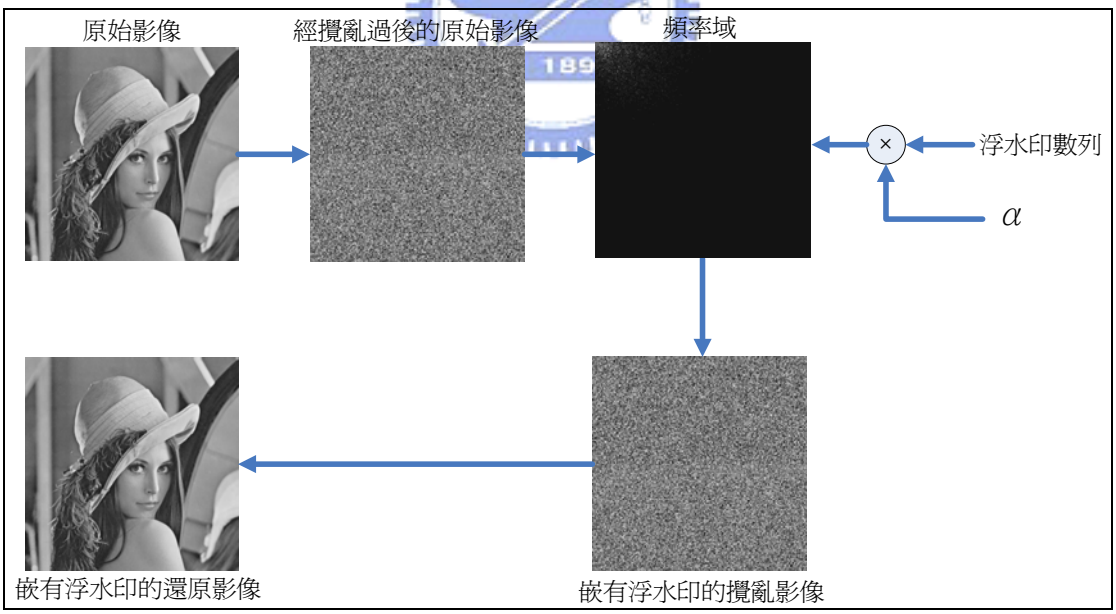


圖 4- 1 浮水印演算法改進圖

由圖 4-1 可看出，嵌入流程大致上仍如同原本的 CKLS 演算法，但是在嵌入浮水印之前，會加入「圖片攪亂」的前置處理動作，嵌入後，則加上攪亂還原的動作，如此便完成整個嵌入的動作。由於本文所提出的演算法會在原本 CKLS

及 CHU 演算法嵌入浮水印前多一步攪亂（Scrambling）的動作。

浮水印演算法會因為不同的圖片結構，產生不一樣的實驗結果，雖然我們並沒有大幅更動 CKLS 演算法，但當我們加上攪亂前處理之後，攪亂圖片和原圖會有很大的差異，幾乎等於是針對不同的圖片嵌入浮水印。因此，接下來進行將 JPEG 壓縮攻擊實驗，比較改良過後的演算法與原始演算法間的差異。

4.2 攪亂演算法與密碼分析攻擊

由第三章的密碼分析攻擊結果可知，密碼分析攻擊具有相當顯著的攻擊效果，因此以下將以密碼分析攻擊本文所提出的攪亂演算法，看看密碼分析攻擊是否仍然具有相同地效果。

我們將攪亂單位分成五種，分別為 1×1、2×2、4、4、8×8 與 16×16 個 Pixel 為一個攪亂單位進行攪亂，試圖了解不同的攪亂單位是否會對密碼分析攻擊帶來不同結果。浮水印嵌入長度同樣為 1000，強度為 0.1。首先先針對 CKLS 浮水印演算法進行測試。

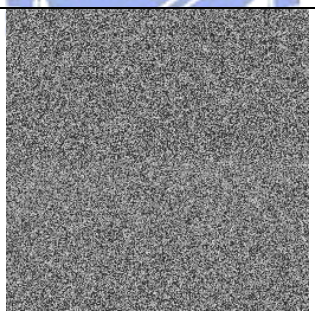
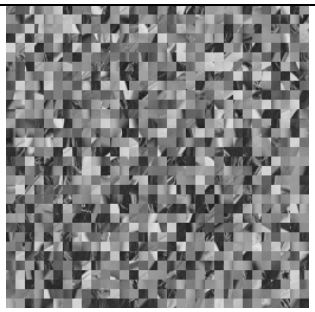
		
原圖	攪亂單位：1×1 個 Pixel	攪亂單位：8×8 個 Pixel

圖 4- 2 攪亂單位差異圖

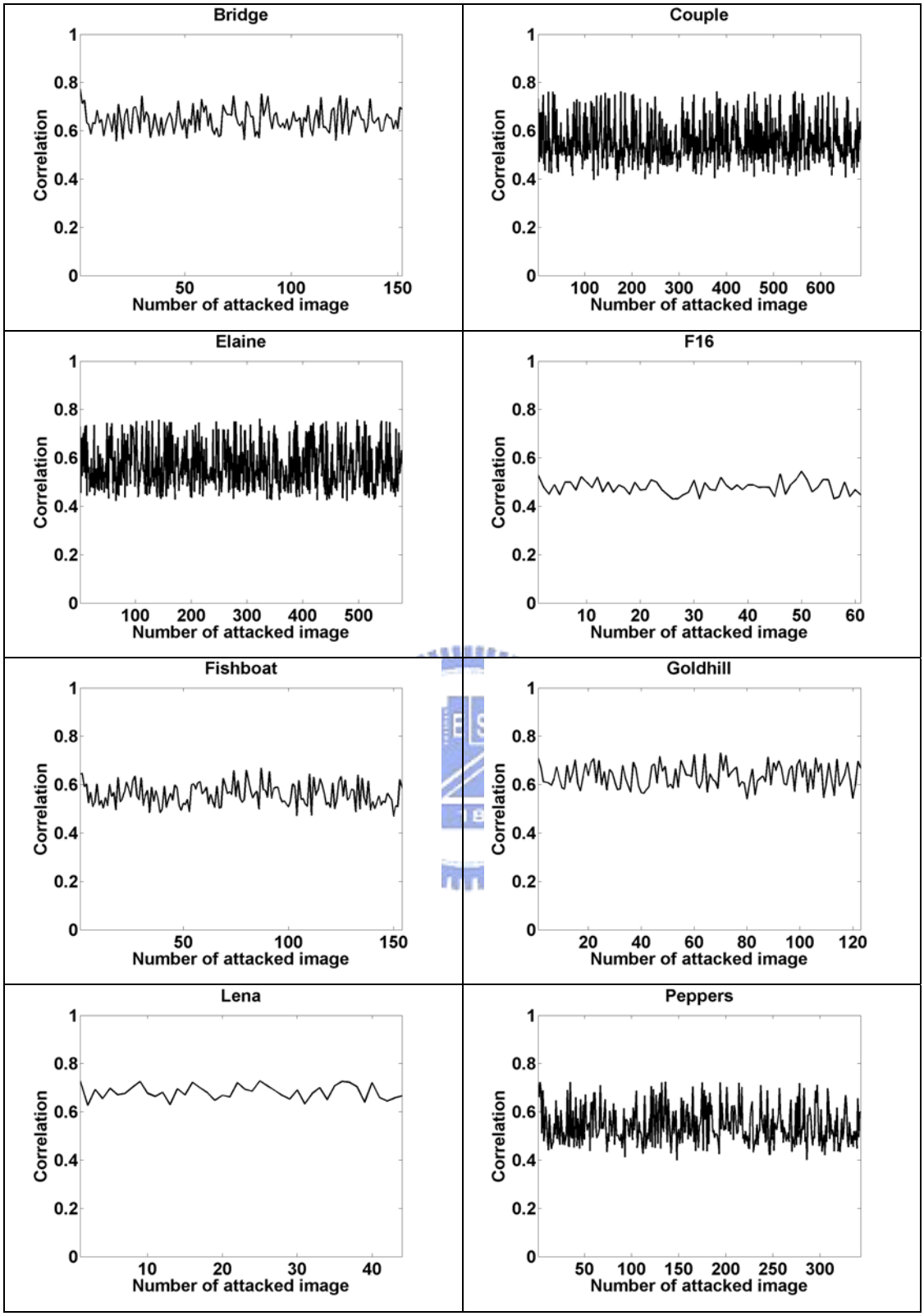


圖 4- 3 8x8 攪亂下的 CKLS 密碼分析攻結果圖

表 4- 1 CKLS 經過攪亂前處理後的密碼分析攻擊結果

Bridge	張數	最佳 PSNR（w, a）	最佳 PSNR 的 correlation	平均 PSNR（w, a）	Max correlation	Min correlation	平均 correlation
1×1	552	34.145588	0.937212	31.280602	0.940223	0.761991	0.852710
2×2	552	34.215813	0.912995	31.359640	0.913115	0.714795	0.817349
4×4	509	34.228294	0.883220	31.237510	0.883280	0.649020	0.759674
8×8	595	34.026249	0.786150	31.243336	0.791380	0.525104	0.661918
16×16	614	31.183960	0.702279	29.569861	0.705269	0.469818	0.577908

Couple	張數	最佳 PSNR（w, a）	最佳 PSNR 的 correlation	平均 PSNR（w, a）	Max correlation	Min correlation	平均 correlation
1×1	635	35.464481	0.914130	31.789358	0.914765	0.632608	0.752678
2×2	642	35.443802	0.885933	31.783535	0.888847	0.578019	0.710867
4×4	630	35.499359	0.821756	31.723779	0.821756	0.515294	0.636303
8×8	685	35.694969	0.762033	31.879893	0.763137	0.396047	0.550855
16×16	686	35.371483	0.659755	31.900590	0.671121	0.293990	0.463629

Elaine	張數	最佳 PSNR（w, a）	最佳 PSNR 的 correlation	平均 PSNR（w, a）	Max correlation	Min correlation	平均 correlation
1×1	554	35.124775	0.934062	31.457246	0.935901	0.666149	0.785768
2×2	542	35.050591	0.904979	31.419616	0.904979	0.616439	0.740832
4×4	550	35.053596	0.858486	31.315387	0.858486	0.576453	0.693809
8×8	578	34.956112	0.761160	31.408446	0.761160	0.422219	0.563086
16×16	567	34.767490	0.694164	31.364756	0.694164	0.391039	0.511814

F16	張數	最佳 PSNR（w, a）	最佳 PSNR 的 correlation	平均 PSNR（w, a）	Max correlation	Min correlation	平均 correlation
1×1	322	33.575188	0.919866	30.517954	0.921416	0.704926	0.786810
2×2	337	33.561066	0.900100	30.559173	0.900866	0.692031	0.769909
4×4	319	33.561455	0.832953	30.523150	0.832953	0.595592	0.675811
8×8	228	33.394653	0.664818	30.318494	0.664818	0.421408	0.498009
16×16	329	33.280254	0.670310	30.519166	0.672171	0.424565	0.510355

Fishboat	張數	最佳 PSNR（w, a）	最佳 PSNR 的 correlation	平均 PSNR（w, a）	Max correlation	Min correlation	平均 correlation
1×1	687	36.439678	0.953250	31.285648	0.953250	0.699997	0.795415
2×2	726	36.143764	0.917480	31.606330	0.917480	0.636104	0.753133
4×4	508	36.458817	0.870987	31.065589	0.870987	0.544927	0.653449
8×8	606	36.136391	0.812164	31.397683	0.812845	0.457839	0.587363
16×16	467	35.496532	0.619761	30.859346	0.619761	0.308409	0.420359

Goldhill	張數	最佳 PSNR（w, a）	最佳 PSNR 的 correlation	平均 PSNR（w, a）	Max correlation	Min correlation	平均 correlation
1×1	722	37.066360	0.961619	32.073837	0.961619	0.762147	0.857335
2×2	730	36.992828	0.947547	32.075126	0.947547	0.687322	0.814691
4×4	776	36.968658	0.908356	32.073660	0.908356	0.661017	0.764905
8×8	695	36.763206	0.848060	31.947667	0.848060	0.529557	0.639704
16×16	789	36.993179	0.785413	31.980110	0.785413	0.385622	0.525471

Lena	張數	最佳 PSNR（w, a）	最佳 PSNR 的 correlation	平均 PSNR（w, a）	Max correlation	Min correlation	平均 correlation
1×1	1764	34.082161	0.925558	31.289007	0.927666	0.707798	0.813541
2×2	773	34.118923	0.758118	31.322051	0.895364	0.646708	0.775724
4×4	796	33.945366	0.844593	31.256176	0.844593	0.575173	0.713563
8×8	763	33.989929	0.736145	31.289621	0.736145	0.481921	0.606391
16×16	799	33.875088	0.661373	31.262027	0.667992	0.411200	0.540423

Peppers	張數	最佳 PSNR（w, a）	最佳 PSNR 的 correlation	平均 PSNR（w, a）	Max correlation	Min correlation	平均 correlation
1×1	564	32.602261	0.917212	30.793187	0.917212	0.738898	0.840147
2×2	534	32.598080	0.883985	30.808755	0.883985	0.693635	0.799053
4×4	495	32.633114	0.816085	30.818460	0.816085	0.609667	0.724437
8×8	550	32.789223	0.722350	30.865739	0.724143	0.503296	0.624202
16×16	547	32.487698	0.628156	30.738578	0.628156	0.441466	0.544262

由以上實驗可發現，圖片攪亂對於浮水印安全性確有提升作用，即使經過密碼分析攻擊後，correlation 依舊能維持在 0.5 以上。

接下來再針對 CHU 浮水印演算法進行攪亂：

表 4- 2 以 Random 方式對 CHU 進行攪亂

Peppers	攪亂單位	Subimage1 的張數	Subimage2 的張數	Subimage3 的張數	Subimage4 的張數	平均 correlation	平均 PSNR
	1×1	515	490	437	490	0.077909	35.13524
	8×8	531	468	439	489	0.069349	34.48498

F16	攪亂單位	Subimage1 的張數	Subimage2 的張數	Subimage3 的張數	Subimage4 的張數	平均 correlation	平均 PSNR
	1×1	118	258	243	304	0.019064	35.72741
	8×8	132	247	262	320	0.034773	35.63779

Fishboat	攪亂單位	Subimage1 的張數	Subimage2 的張數	Subimage3 的張數	Subimage4 的張數	平均 correlation	平均 PSNR
	1×1	176	138	4	10	0.105876	35.80961
	8×8	47	180	5	11	0.132118	35.9219

由於攪亂後的結果並不理想，因此我們改採以 Fibonacci 方式攪亂，並採用三種不同的攪亂策略，分別為 1、使用原圖攪亂，2、對 Subimage 攪亂，3、對 Subimage 攪亂（使用不同的攪亂種子）。

表 4- 3 原圖（256x256）以 Fibonacci 方式攪亂：

Peppers	攪亂單位	Subimage1 的張數	Subimage2 的張數	Subimage3 的張數	Subimage4 的張數	平均 correlation	平均 PSNR
	1×1	9	30	4	53	0.103831	35.18429
	8×8	431	30	5	70	0.155301	35.11256

F16	攪亂單位	Subimage1 的張數	Subimage2 的張數	Subimage3 的張數	Subimage4 的張數	平均 correlation	平均 PSNR
	1×1	117	269	290	307	0.015564	35.8177
	8×8	516	259	721	320	0.069237	35.727867

表 4- 4 每張 Subimage 以 Fibonacci 方式攪亂：

Peppers	攪亂單位	Subimage1 的張數	Subimage2 的張數	Subimage3 的張數	Subimage4 的張數	平均 correlation	平均 PSNR
	1×1	7	27	5	44	0.495835	35.07117
	8×8	463	59	5	44	0.073507	35.01398

F16	攪亂單位	Subimage1 的張數	Subimage2 的張數	Subimage3 的張數	Subimage4 的張數	平均 correlation	平均 PSNR
	1×1	104	285	262	320	0.203985	35.73533
	8×8	329	79	262	320	0.018448	35.60674

表 4- 5 每張 Subimage 以 Fibonacci 方式攪亂（使用不同的攪亂種子）：

Peppers	攪亂單位	Subimage1 的張數	Subimage2 的張數	Subimage3 的張數	Subimage4 的張數	平均 correlation	平均 PSNR
	1×1	8	36	4	44	0.091258	35.15490
	8×8	23	18	6	44	0.143178	35.10386

F16	攪亂單位	Subimage1 的張數	Subimage2 的張數	Subimage3 的張數	Subimage4 的張數	平均 correlation	平均 PSNR
	1×1	145	307	262	320	0.099374	35.63076
	8×8	72	373	262	320	0.096409	35.802299

由以上實驗結果得知，以先分成 4 張 Subimage 後，每張 Subimage 再以 1×1 及相同攪亂種子攪亂，所能獲得的安全性為最高，因此以下將以此種方式對於其它圖片進行測式，並採用不同的攪亂方法。

表 4- 6 CHU 經過 Fibonacci 後的密碼分析攻擊結果

攪亂單位（1×1）	correlation	PSNR	correlation[3]	PSNR[3]
Bridge	0.202520	36.193825	0.191	34.39
Couple	0.311170	35.828552	0.158	34.87
Elaine	0.310561	34.875278	0.168	34.53
F16	0.395835	35.07117	0.114	35.14
Fishboat	0.173430	36.097137	0.137	34.02
Goldhill	0.310729	37.793503	0.124	36.11
Lena	0.307535	35.628736	0.113	34.78
Peppers	0.395835	35.07117	0.092	35.32

表 4- 7 CHU 經過 Random 後的密碼分析攻擊結果

攪亂單位（1×1）	correlation	PSNR	correlation[3]	PSNR[3]
Bridge	0.324466	36.136440	0.191	34.39
Couple	0.315988	34.532841	0.158	34.87
Elaine	0.307253	35.047394	0.168	34.53
F16	0.330752	35.631170	0.114	35.14
Fishboat	0.302009	35.960284	0.137	34.02
Goldhill	0.361170	37.505314	0.124	36.11
Lena	0.315030	35.894715	0.113	34.78
Peppers	0.437862	35.109879	0.092	35.32

經由以上實驗並與[3]比較後可看出，圖片攪亂即使對於 CKSL 及 CHU 浮水印演算法仍具有提升安全性的作用，即使經過密碼分析攻擊後，correlation 依舊能維持在 0.3 以上。

4.3 攪亂演算法與密碼分析攻擊小結

由 4.2 節的實驗可知，攪亂演算法對於密碼分析攻擊具有良好的防護效果，不論攪亂單位是 1×1 或 8×8，correlation 都還能維持在 0.3 以上。

在 CKLS 不同攪亂單位的實驗可以發現，以 1×1 為攪亂單位時，correlation 能維持在 0.7 至 0.9 之間；以 8×8 為攪亂單位時，correlation 則維持在 0.5 至 0.7

之間，圖片攪亂能夠成功提升浮水印安全性，嘗試分析其原因如下：

圖 4-4 為 Lena 在不同攪亂單位下，空間域圖形與頻率域浮水印嵌入位置圖（白點）。由以上空間域與頻率域浮水印嵌入位置可以發現，當我們先將圖片攪亂再嵌入浮水印時，若攪亂的單位愈小，浮水印分佈的位置會愈分散於各頻區。

由於我們將浮水印嵌入後，會將攪亂的圖片還原，所以當攻擊者取得嵌有浮水印的圖片並且想要透過迴歸分析進行 DCT 係數重建時，重建的位置會趨於左上角，也就是與 Original 的嵌入位置相同，但其實我們將圖片攪亂後，嵌入的位置是相當分散的，攻擊者無法找出正確的位置並加以分析與重建，因此縱然試圖攻擊後，我們還是能夠取出正確的浮水印。

但對於 CHU 浮水印演算法而言，浮水印嵌入位置依照 zigzag 順序嵌入，但 CKLS 則是選擇前 N 大嵌入，CKSL 在經過攪亂後，前一千大的位置會有所改變，但 CHU 的方法並不會，攻擊者比較能夠準確地針對嵌有浮水印的位置進行係數重建，因此攪亂前處理在 CHU 的方法上，略遜於 CKLS。相關的多項式係數表列表文末的附錄中。

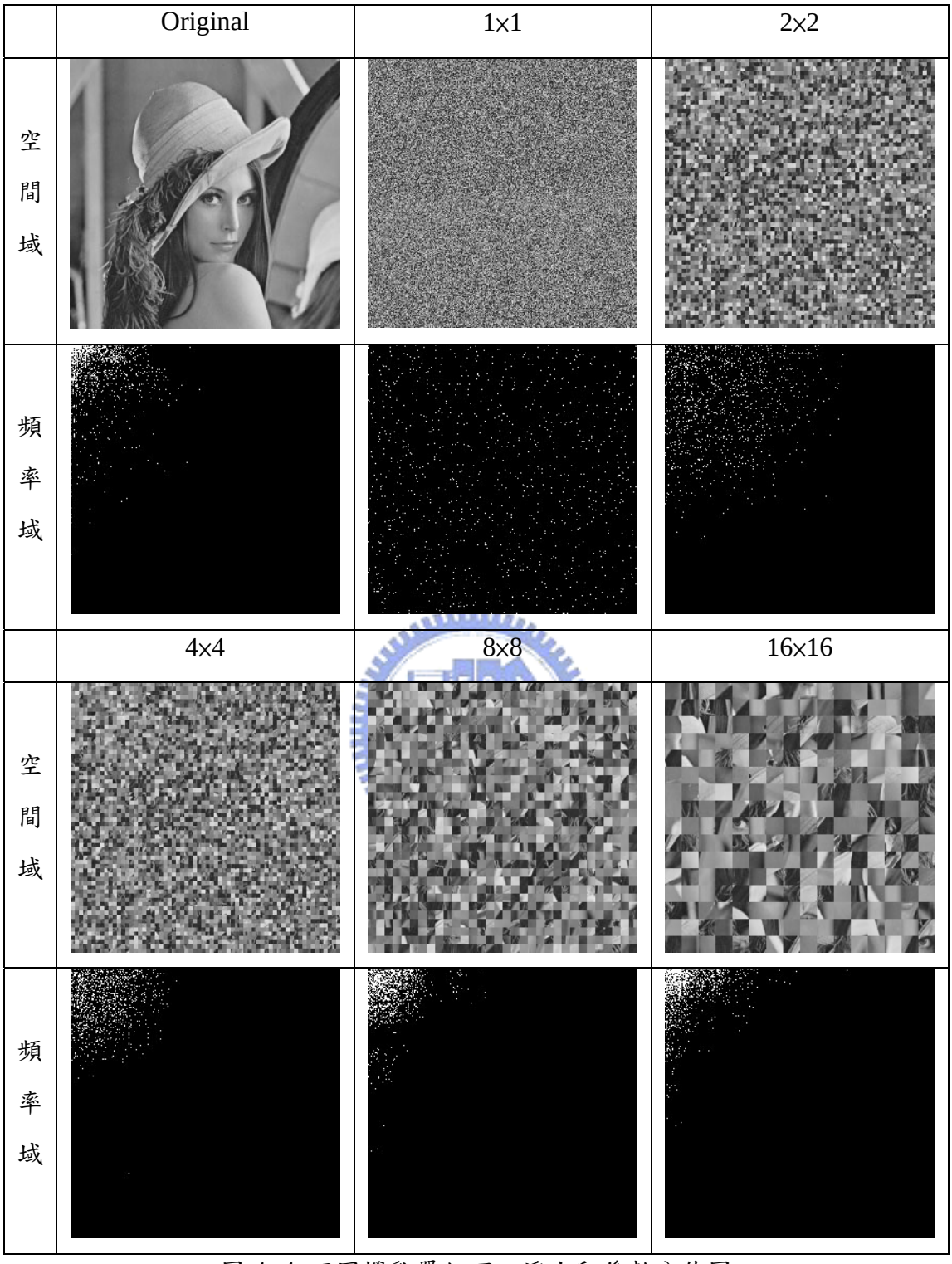


圖 4- 4 不同攪亂單位下之浮水印係數分佈圖

4.4 圖片強韌性

浮水印演算法會因為不同的圖片結構，產生不一樣的實驗結果，雖然我們並沒有大幅更動 CKLS 演算法，但當我們加上攪亂前處理之後，攪亂圖片和原圖會有很大的差異，幾乎等於是針對不同的圖片嵌入浮水印。因此，接下來將進行影像處理攻擊實驗，比較改良過後的演算法與原始演算法的差異。影像處理攻擊的方式如下：

- JPEG Compression

JPEG 壓縮攻擊是一種頻率域攻擊：將圖片分成 8x8 的區塊大小，對每一個區塊進行 DCT 轉換，移除高頻的部份，然後進行反向 DCT 轉換回空間域的圖片。

- Median Filter

Median Filter 通常用於移除影像中多餘的雜訊(Noise)，不過也會因此造成影像的模糊化。

- Pixel Shift

讓影像中的每一列像素都向右位移 n 個像素，並將最右方超出影像邊界的 n 列像素搬移至影像的最左方，形成一個向右位移的循環。

- Rotation and Scaling

此種攻擊方法是以圖片的中心點為圓心，進行順時針與逆時針的旋轉，然後再對旋轉之後的圖片進行裁切修正的處理，使其維持與原圖一樣的大小。

影像處理攻擊實驗裡，我們將攪亂的方式分為五種，分別為 1x1、2x2、4x4、8x8 及 16x16 個 Pixel 構成一個攪亂單位進行攪亂，觀察不同的攪亂單位的圖片與無攪亂圖片之間的強韌性差異。首先先針對 CKLS 浮水印演算法進行實驗。

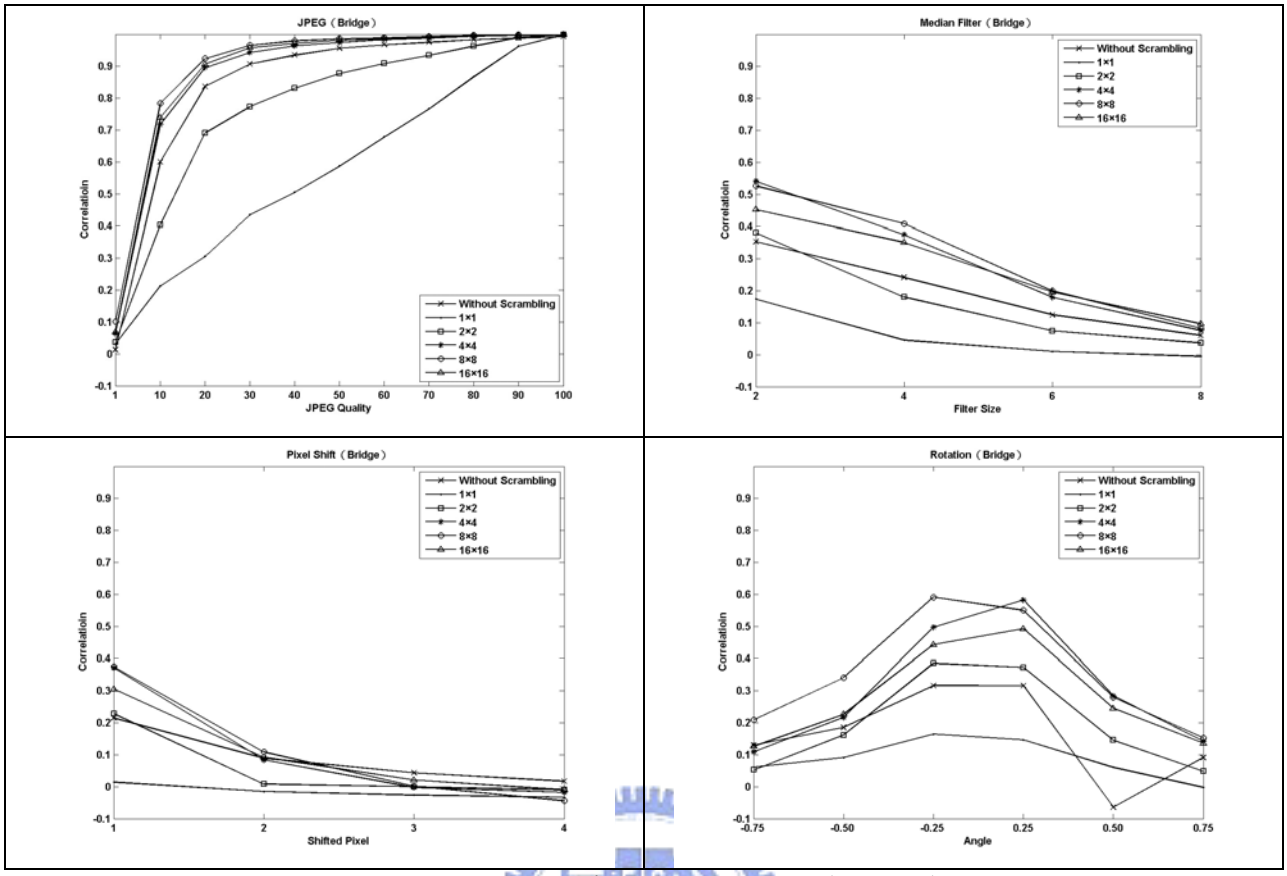


圖 4- 5 CKLS 影像處理攻擊結果圖（Bridge）

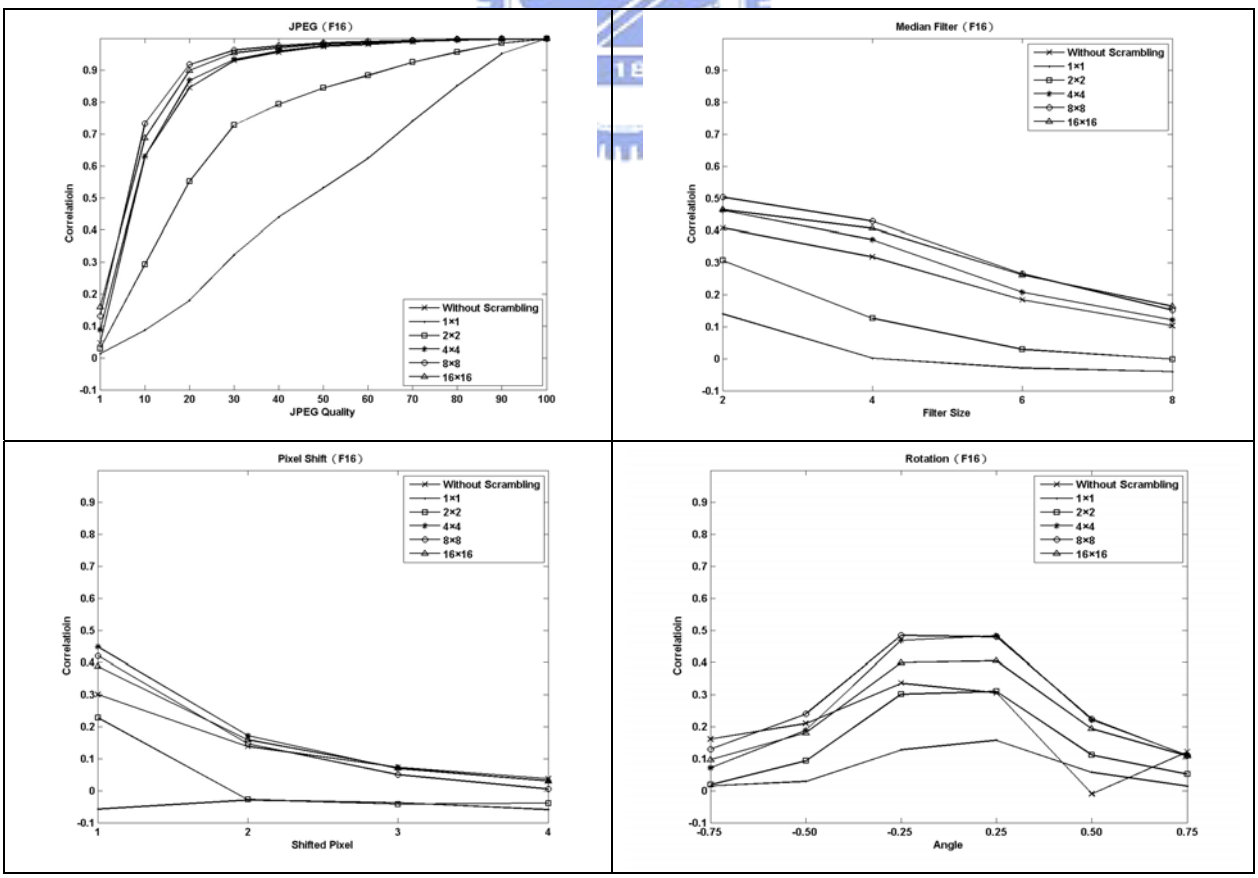


圖 4- 6 CKLS 影像處理攻擊結果圖（F16）

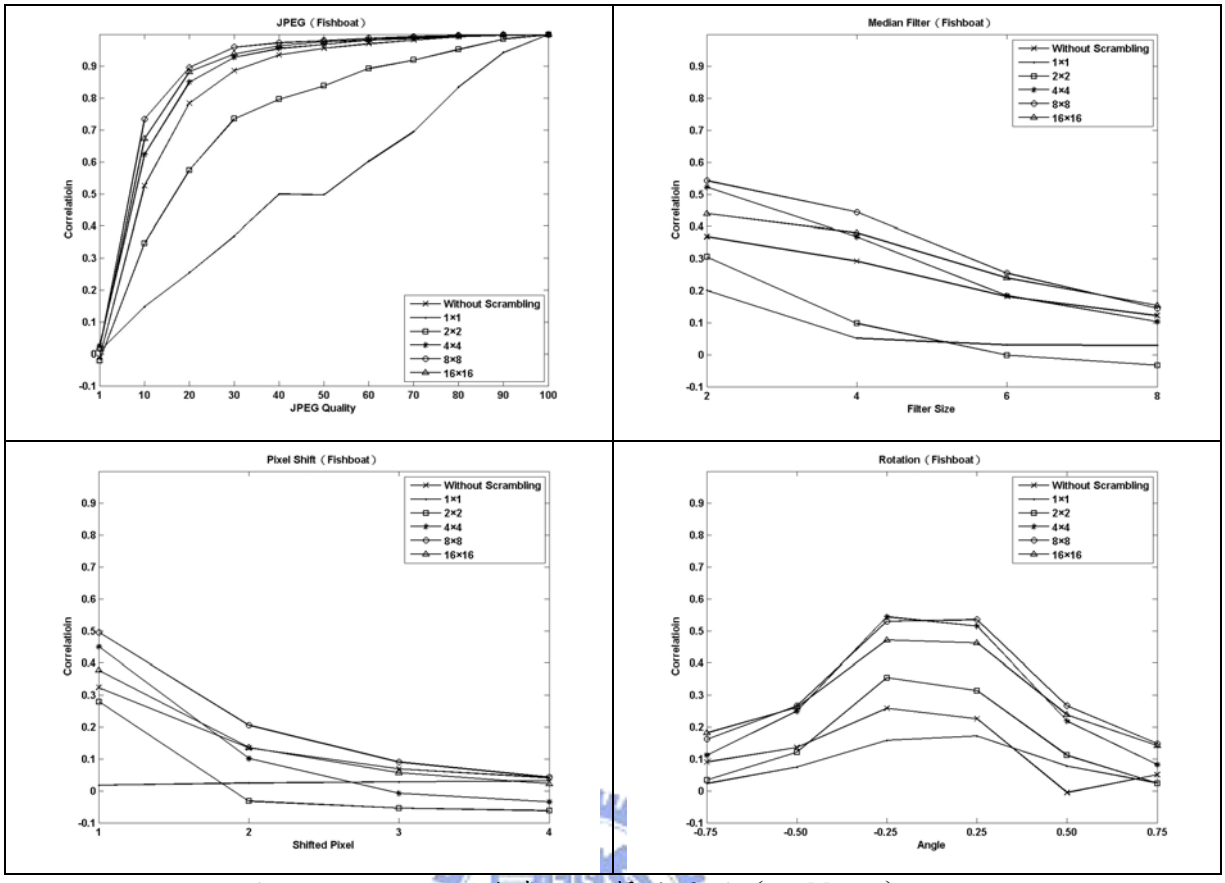


圖 4- 7　CKLS 影像處理攻擊結果圖（Fishboat）

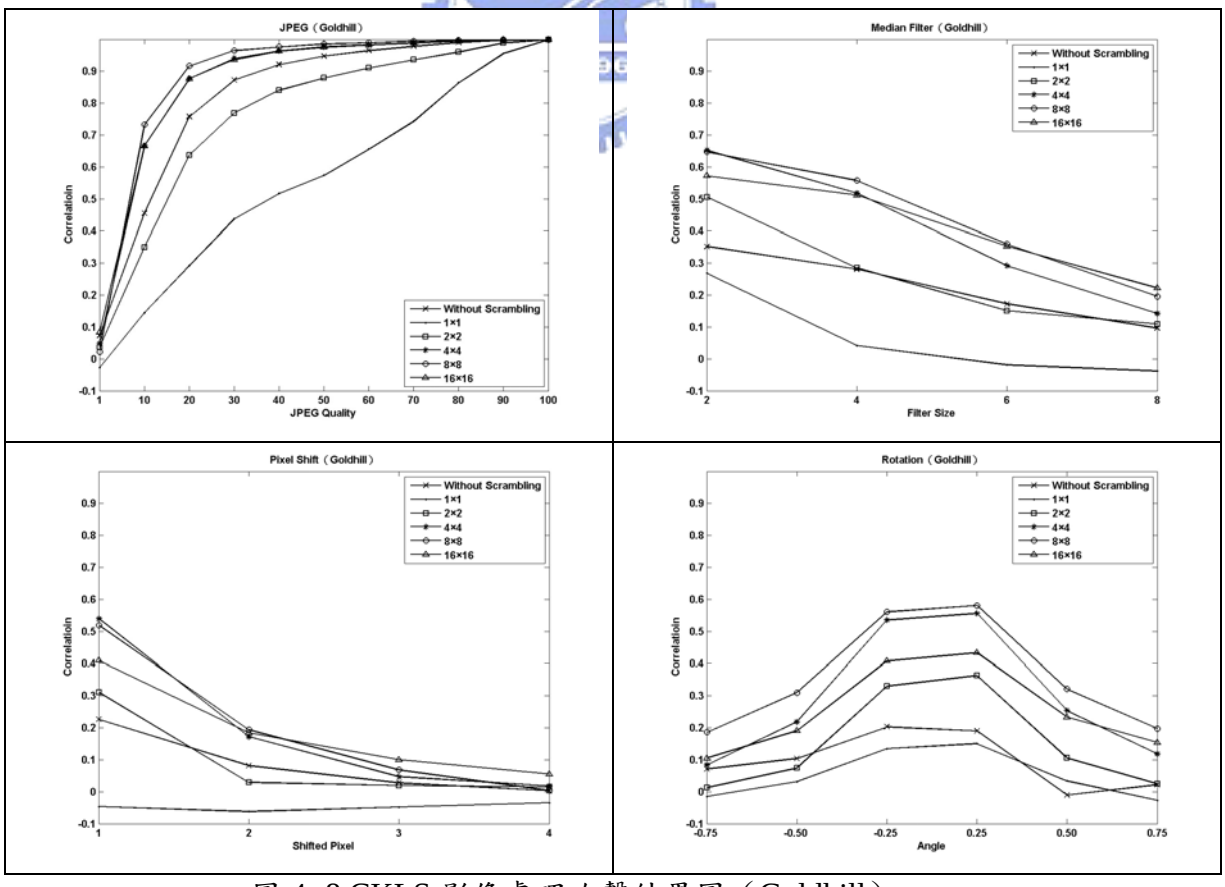


圖 4- 8 CKLS 影像處理攻擊結果圖（Goldhill）

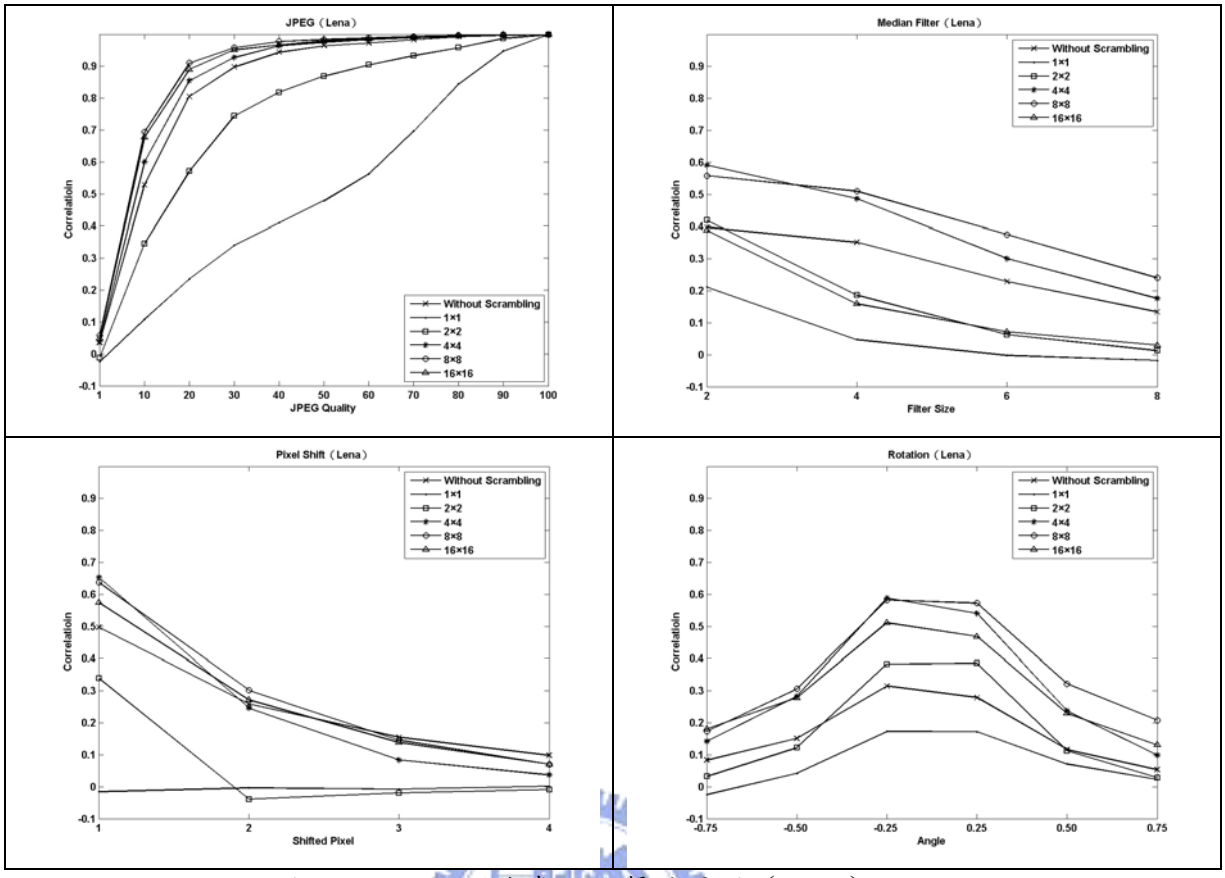


圖 4- 9 CKLS 影像處理攻擊結果圖（Lena）

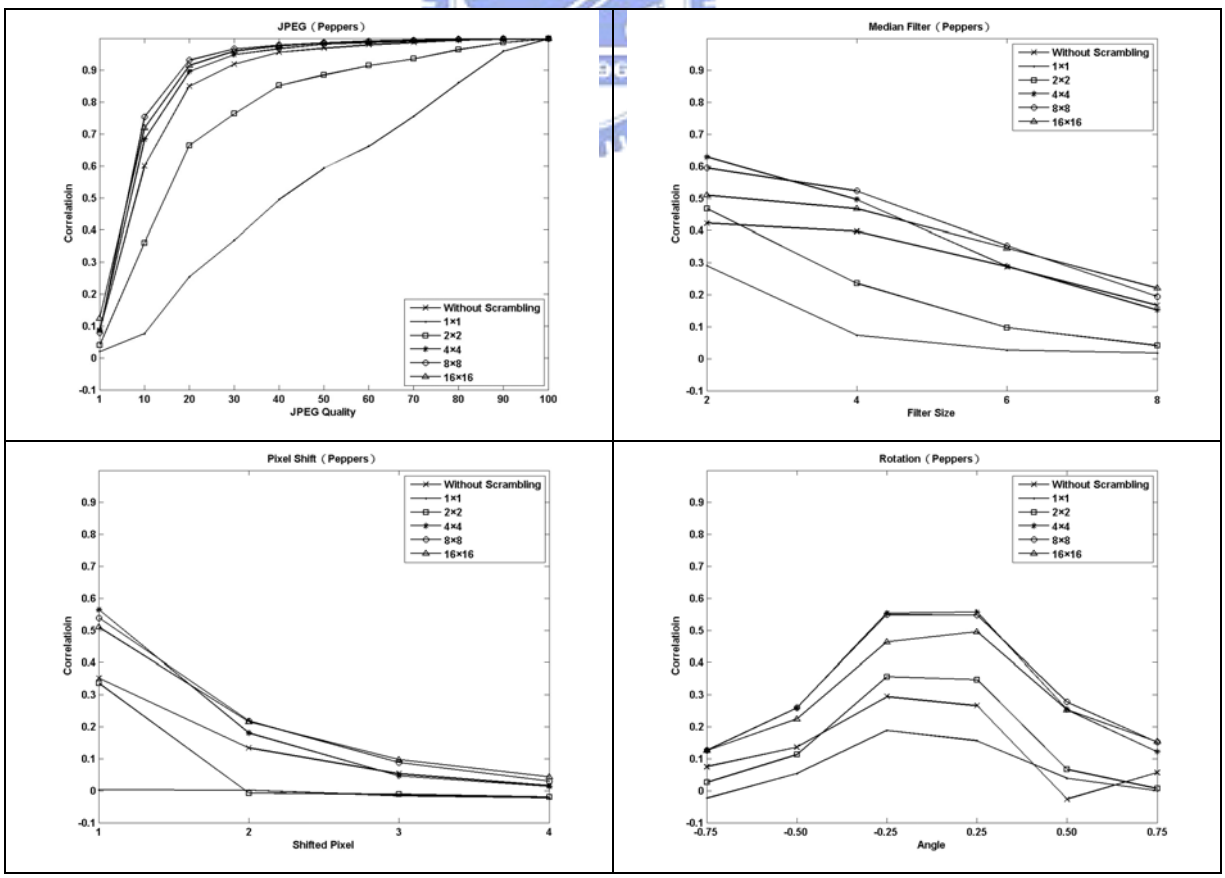


圖 4- 10 CKLS 影像處理攻擊結果圖（Peppers）

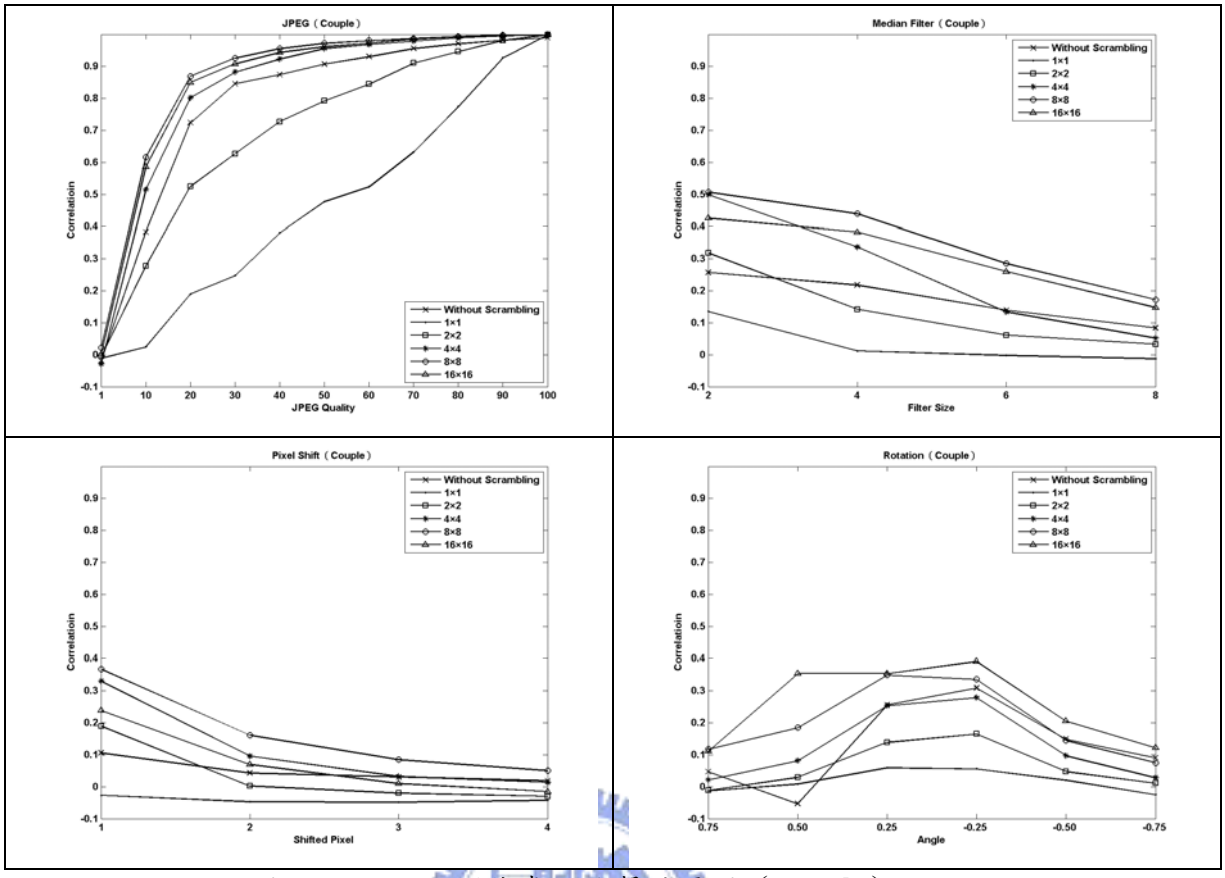


圖 4- 11 CKLS 影像處理攻擊結果圖（Couple）

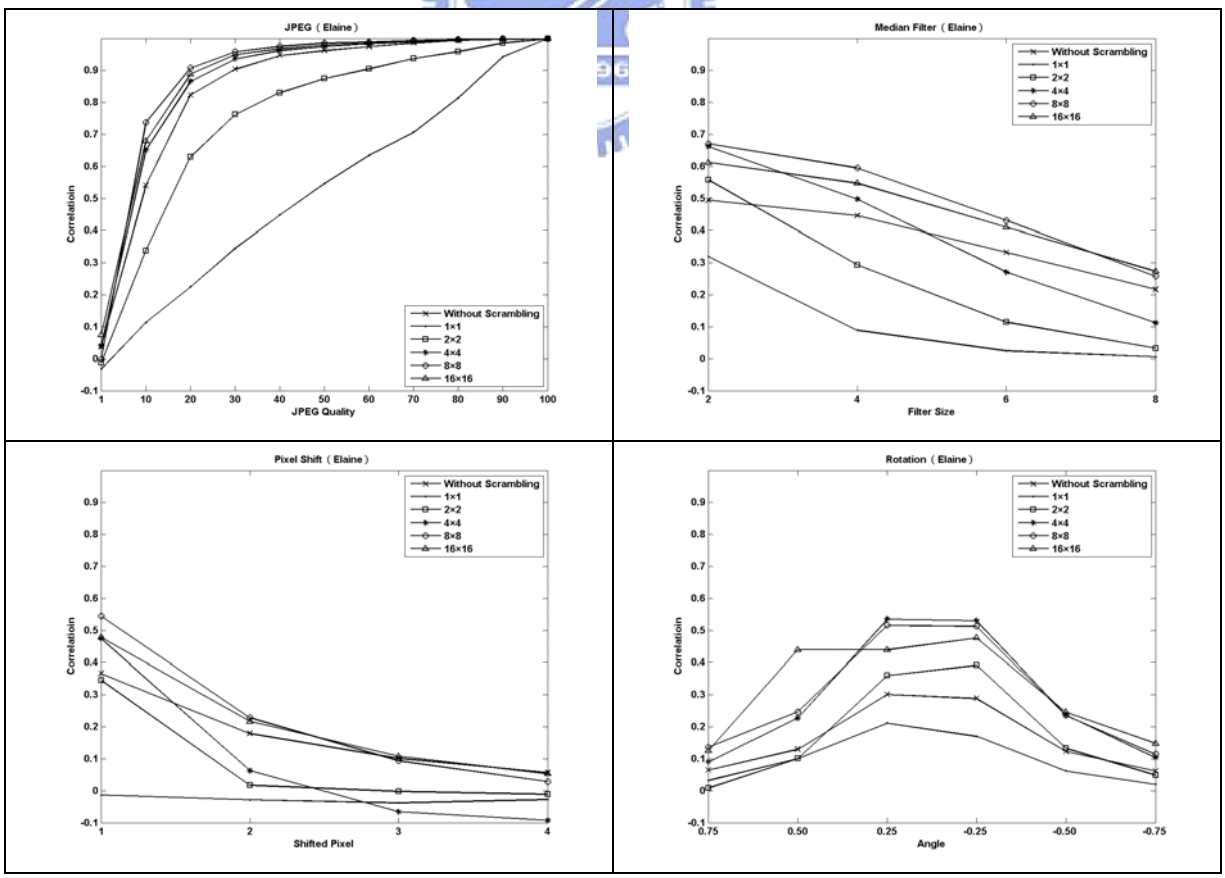


圖 4- 12 CKLS 影像處理攻擊結果圖（Elaine）

由以上實驗結果可看出，不論是在何種攻擊類型上，8x8 擾亂能提供最佳的強韌性。

接下來再針對 CHU 浮水印演算法進行實驗：

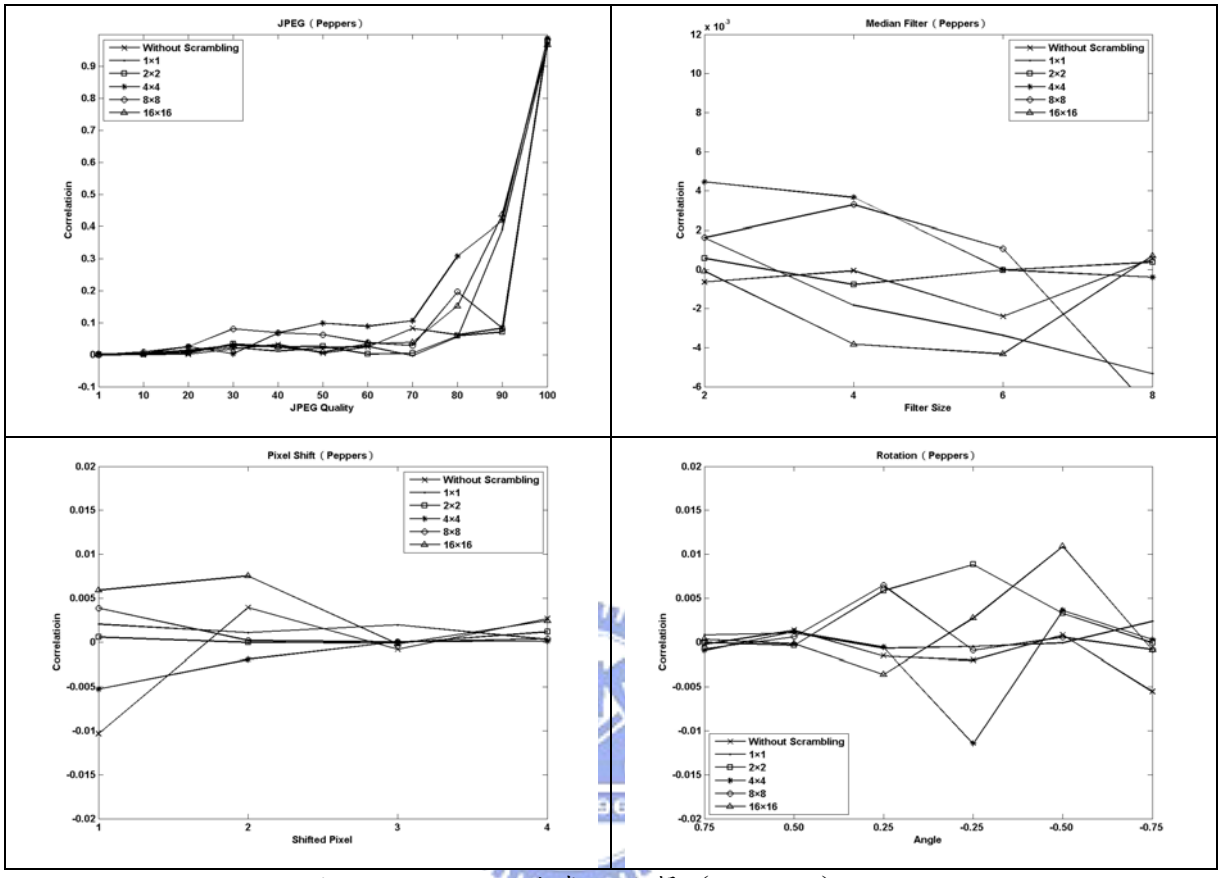


圖 4- 13 CHU 影像處理攻擊（Peppers）

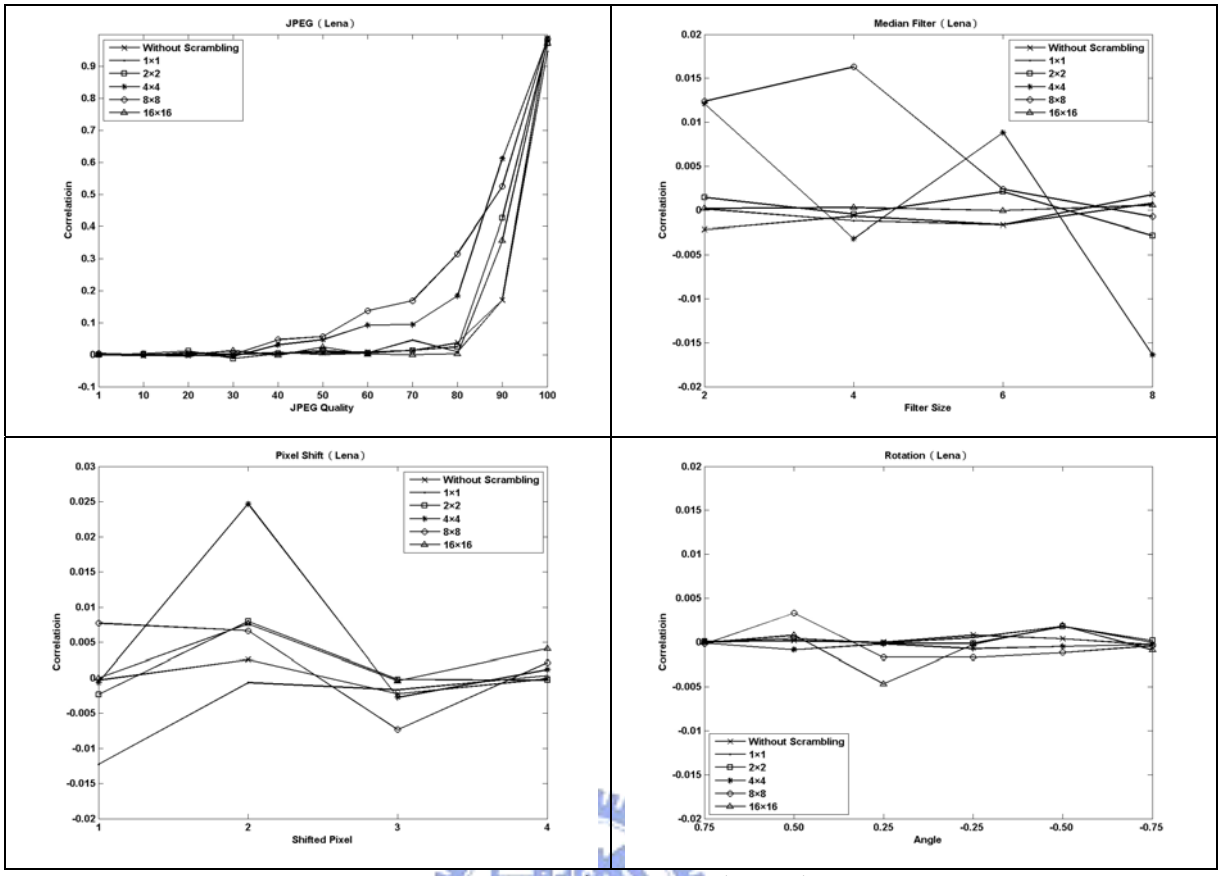


圖 4- 14 CHU 影像處理攻擊（Lena）

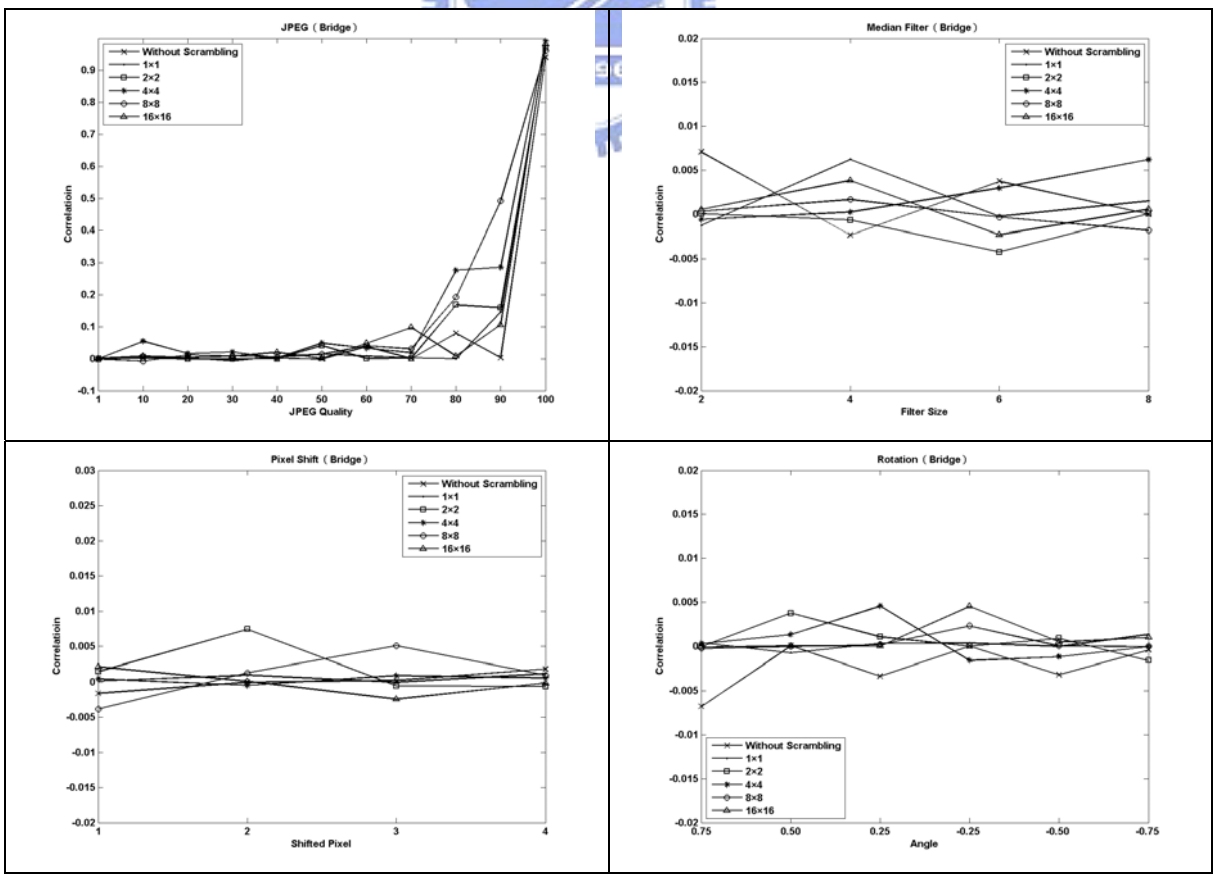


圖 4- 15 CHU 影像處理攻擊（Bridge）

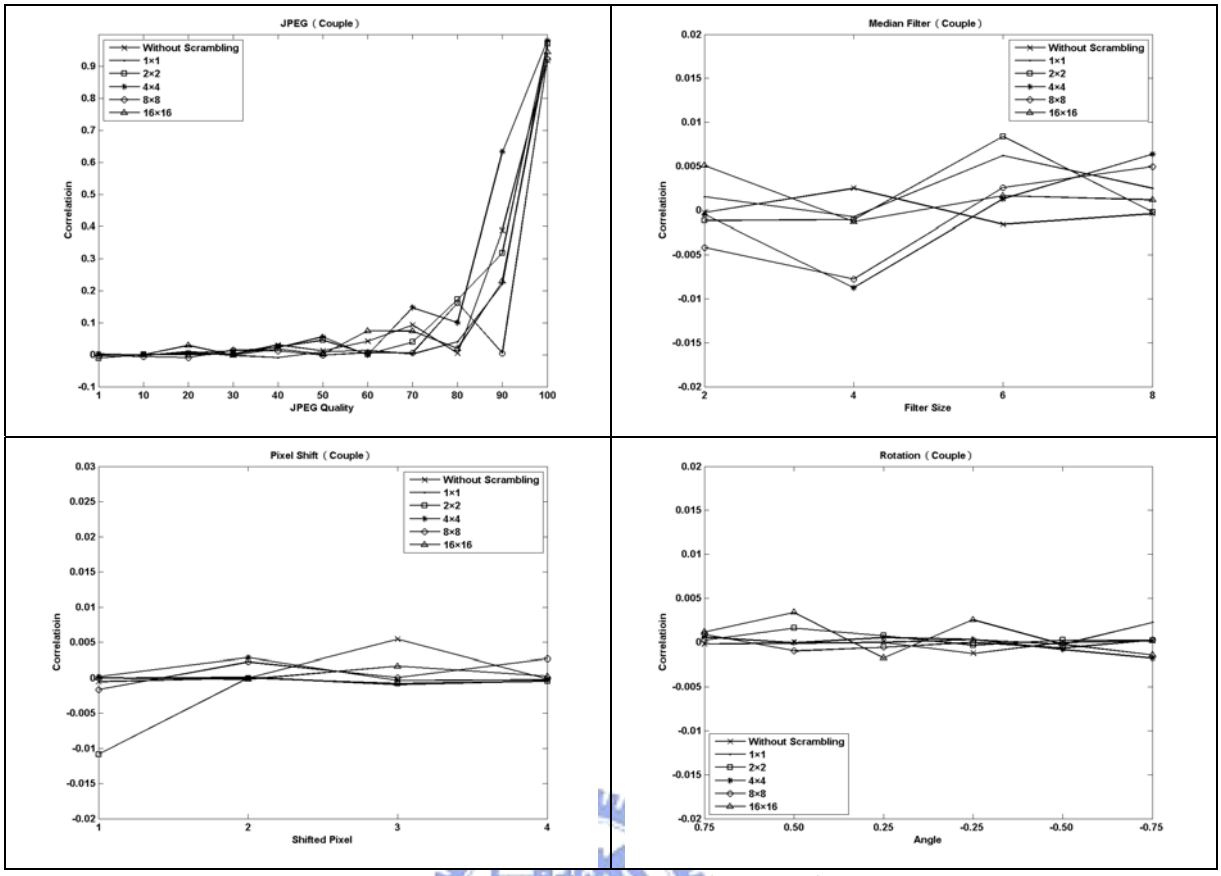


圖 4- 16 CHU 影像處理攻擊（Couple）

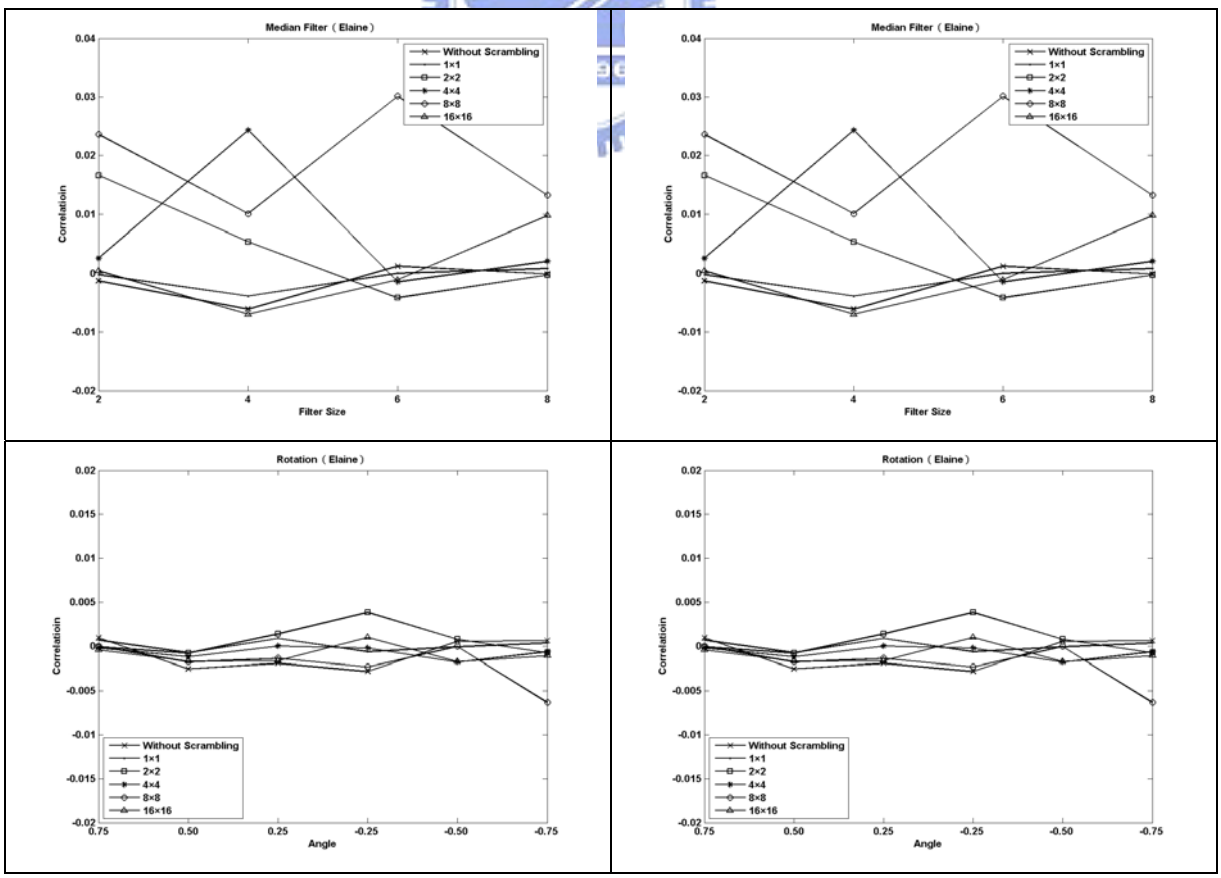


圖 4- 17 CHU 影像處理攻擊（Elaine）

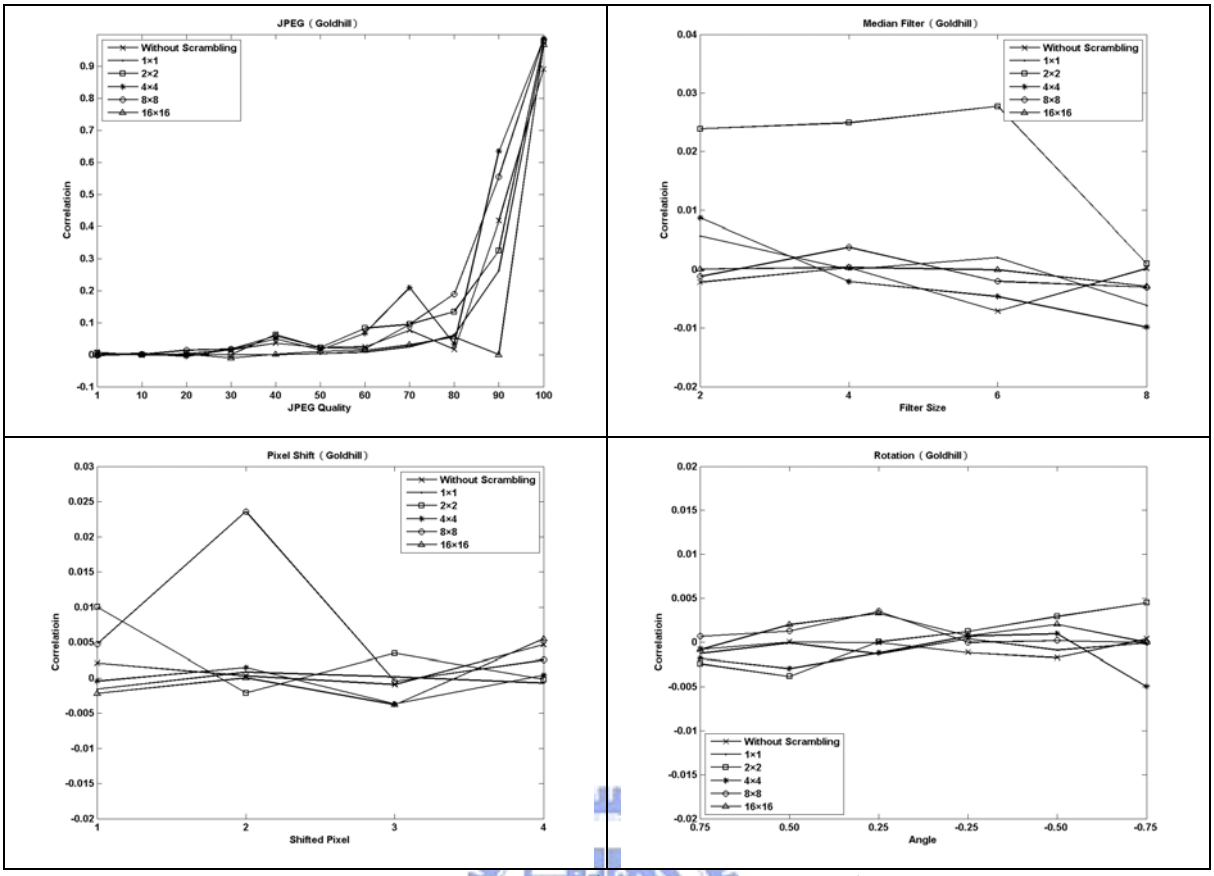


圖 4- 18 CHU 影像處理攻擊（Goldhill）

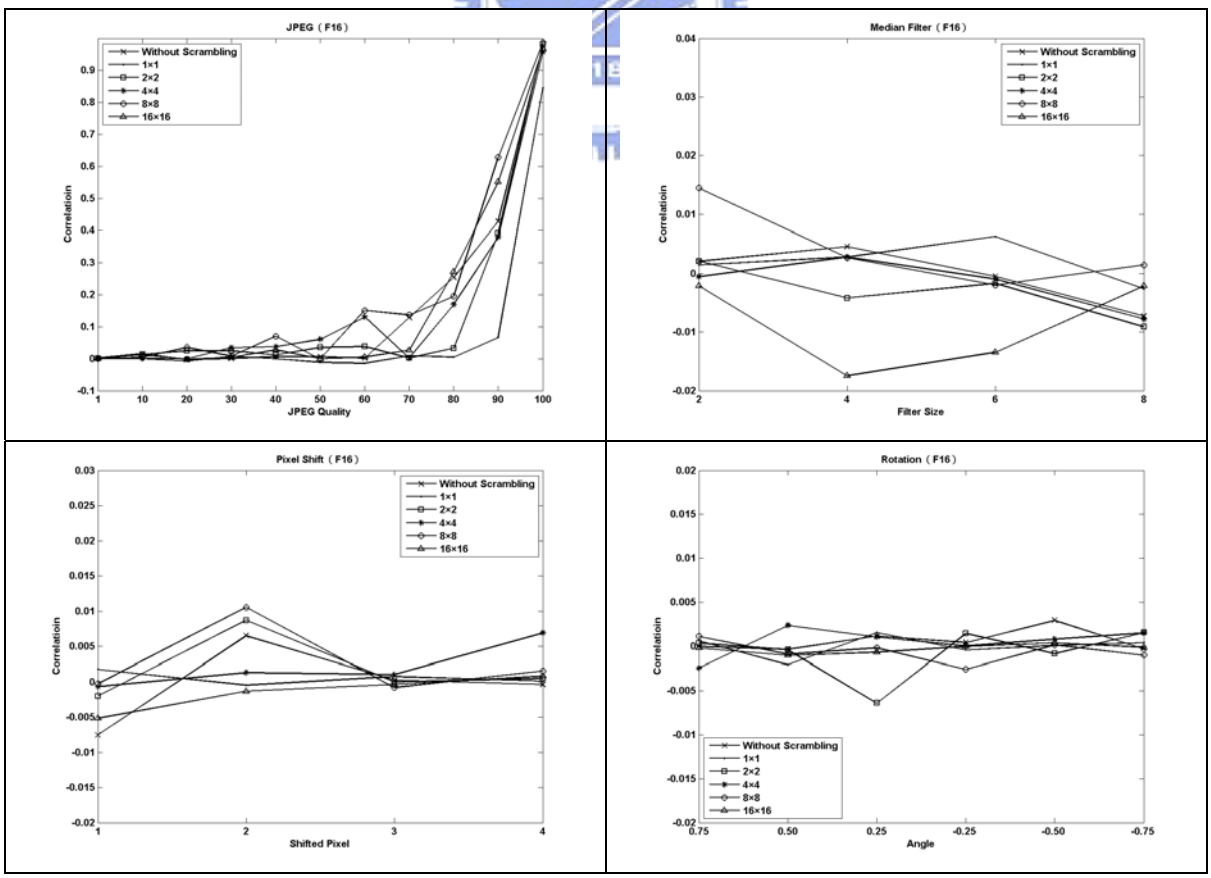


圖 4- 19 CHU 影像處理攻擊（F16）

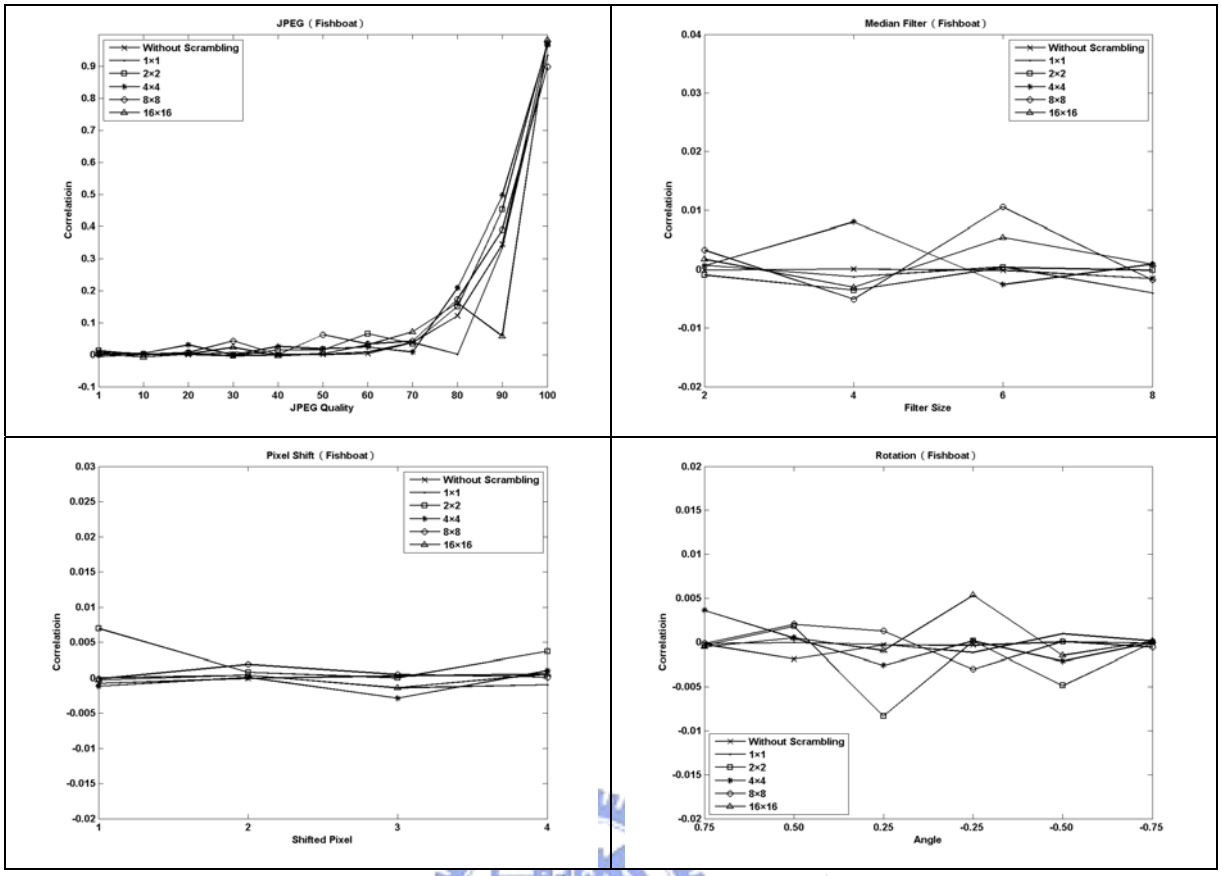


圖 4- 20 CHU 影像處理攻擊（Fishboat）

為了解不同擾亂單位下強韌性的差異，本研究進一步對於不同攪亂單位下嵌有浮水印的前一千大係數進行平均值計算：

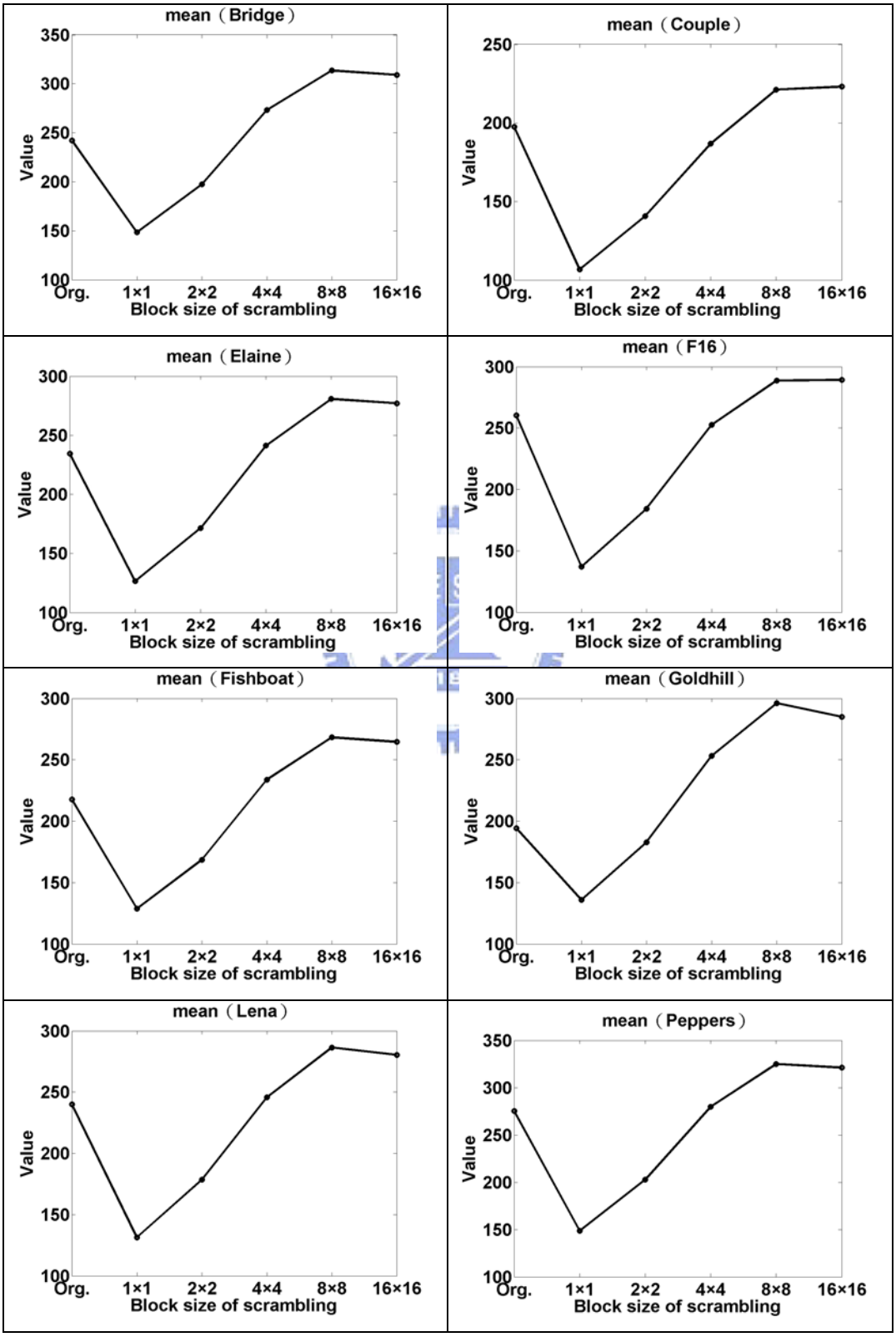


圖 4- 21 不同攪亂單位下浮水印係數平均值的差異圖

實驗結果發現，以 8x8 擾亂過後的圖片經 DCT 轉換後，能提供較高的 DCT 係數值藏入較多的資訊量，致使提升了整體浮水印的強韌性。

由影像處理攻擊的實驗可知，1x1 攪亂會降低浮水印的強韌性，8x8 攪亂卻有提升浮水印強韌性的效果，雖然在安全性上，8x8 攪亂的保護性不如 1x1 攪亂，但仍能將 correlation 維護在 0.5 以上，因此就強韌性與安全性兼顧的設計要點來看，8x8 攪亂應為一較好的攪亂單位。另外，由於 CHU 浮水印演算法本身具有盲擷取的特性，因此在強韌性上較 CKSL 為差，同時不論是以何種攪亂方式攪亂，都不會像 CKLS 浮水印演算法般出現一種趨勢性。

4.5 圖片特徵值

由前一節的實驗結果得知，攪亂演算法對於密碼分析攻擊具有良好的抵抗能力，密碼分析者是否能在攻擊前發覺此含有浮水印的圖片，在嵌入程序時有將圖片攪亂呢？圖片特徵值是否有所改變？以下我們採用[9]所提出的影像特徵值作分析。

4.5.1 圖片特徵值分析

由 2.1.3 節可知，一張圖片可以分成空間域與頻率域。透過轉換處理，將圖片由空間域轉換成頻率域後，可以將圖片不同頻率的個別濾出，產生高低頻帶。由於一張嵌有浮水印的圖片以空間域呈現時，我們很難用肉眼察覺出是否經過攪亂前處理，因此在分析圖片特徵值時，我們首先將以 CKLS 浮水印演算法嵌入浮水印的圖片進行 DCT 轉換為頻率域，分別對低頻、中頻、高頻區進行分析。實驗使用的特徵值包括下列 8 項：

1. Energy：$\frac{1}{N}\sum_{i} I(i)^2$

用以計算目標資料的能量大小。簡而言之，即是目標資料的平均絕對值。當數值越大，代表目標資料的能量越大。若用在計算小波係數值，當數值越大，代表可以藏入的浮水印空間較大。

2. Normal entropy：$-\frac{1}{N}\log\left(I(i)^2\right)$

Entropy 又可以稱為資訊來源的不確定性(uncertainty)，「它定義了由觀察單

一個資訊來源輸出所得的平均資訊量。隨著數值大小的增加，代表更多的不確定性，因而有更多的資訊將伴隨資料來源。」因此，當 Entropy 數值越大，目標資料越雜亂且不確定。

3. Kurtosis：$\frac{1}{N}\sum_i\left(\frac{I(i)-mean}{\delta}\right)^4-3$

此特徵值稱為峰態，可以計算目標資料的分布情況。當數值接近零，代表數值分佈接近常態分布。若是數值大於零，代表統計的 Histogram 形狀較陡；若是數值小於零，則形狀較平緩。

4. Log Energy entropy：$\sum_i I(i)\log(I(i))$

如同特徵值的名字，此種 Entropy 是在 Energy 的公式中多加上一個 log 運算元。如同 Entropy，可以用來計算資訊量。

5. Range：$\max-\min$

如公式所呈現，此為最大值與最小值的差距。

6. Shannon entropy：$-\frac{1}{N}I(i)^2\log\left(I(i)^2\right)$

依據提出此一量測值的數學家 Claude Shannon 而命名，同樣是計算能量不確定性的特徵值。雖然已有標準的 Entropy 特徵值可以使用，但因為 Shannon Entropy 也是一項常用的量測值，所以一併作為統計觀察的特徵值使用。

7. Skewness：$\frac{1}{N}\sum_i\left(\frac{I(i)-mean}{\delta}\right)^3$

此特徵值稱為偏態，可以計算目標資料的分布情況。當數值接近零，代表資料多集中在中間數值。數值越大，資料多集中在較大的數值區；數值越小，資料則多集中在較小的數值區。

8. Variance：$\frac{1}{N-1}\sum_i\left(I(i)-mean\right)^2$

變異數是用來衡量資料的變異情形，變異數越大，表示資料越分散。

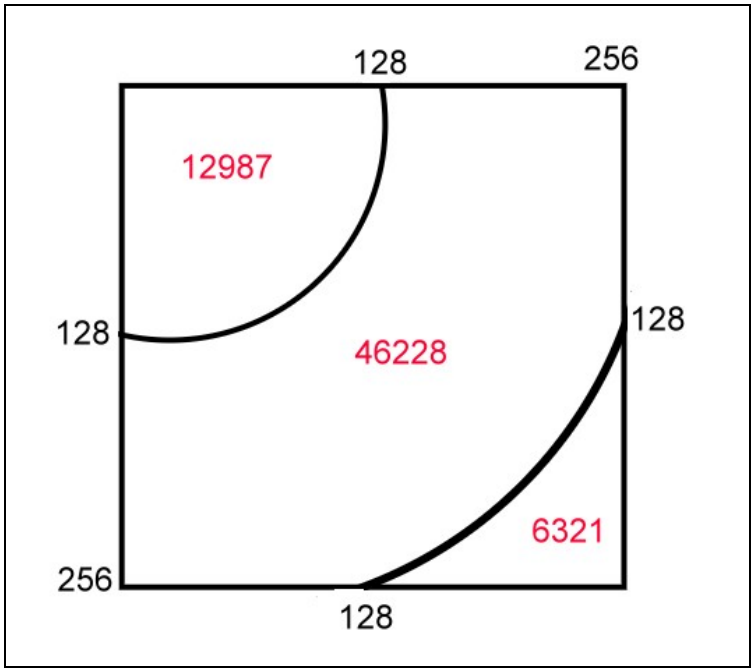


圖 4- 22 一張 256x256 圖片各頻區的係數個數分佈圖

我們實驗所採用的圖片大小為 256x256，此種尺寸的圖片，其經過 DCT 轉換後，各頻區的係數個數如圖 4-20 所示，低頻區含有 12987 個數值，中頻區含有 46228 個數值，高頻區含有 6321 個數值。

分析結果如下：

Range（低頻區）

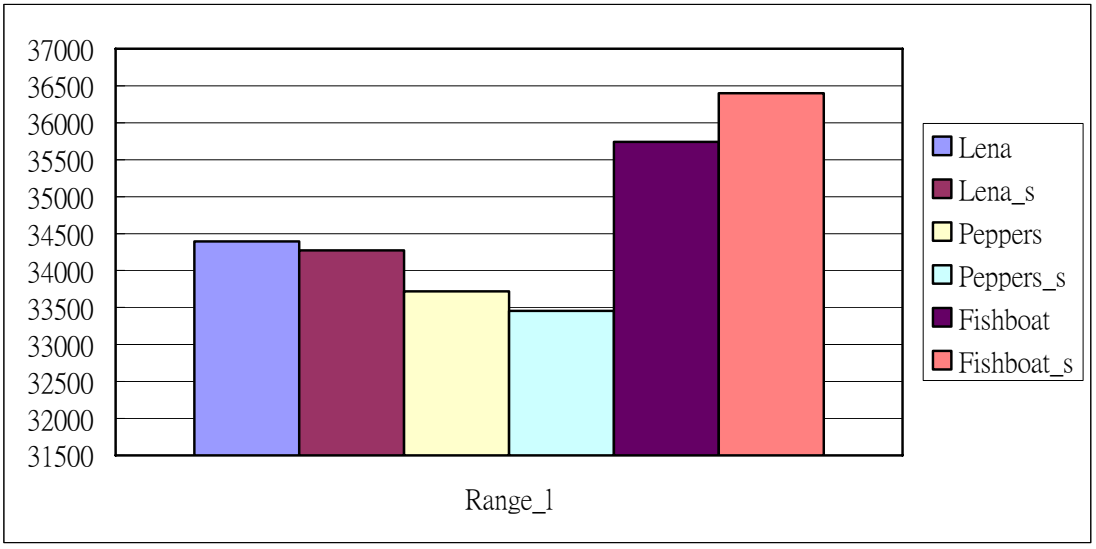


圖 4- 23 CKLS_Range（低頻區特徵值）

Range（中、高頻區）

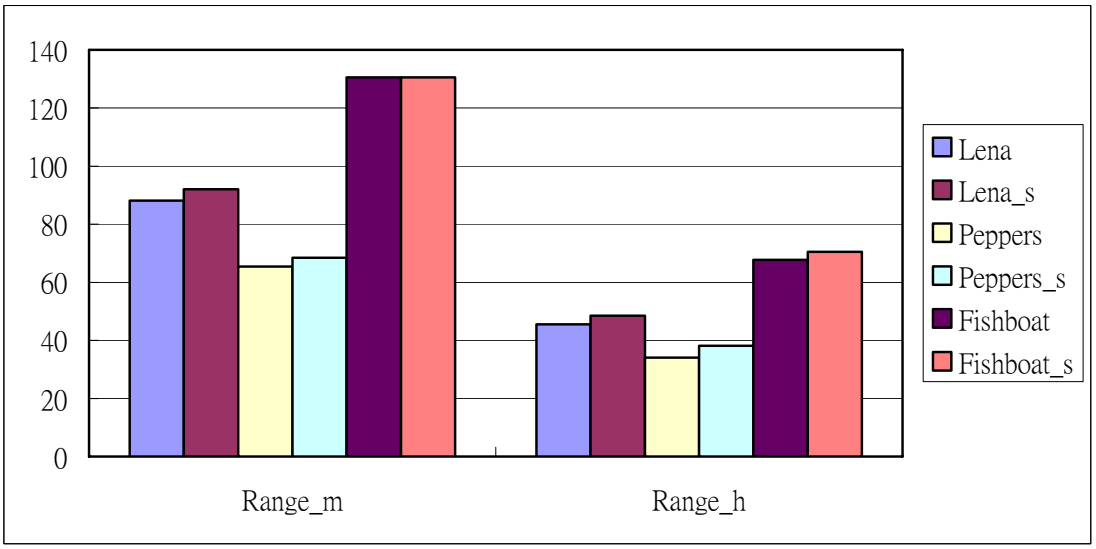


圖 4- 24 CKLS_Range（中高頻區特徵值）

由圖 4-21 及 4-22 可以看出，不論是在低頻區或中高頻區，有無攪亂的圖片，其係數的 Range 都幾乎為一致。

Variance（低頻區）

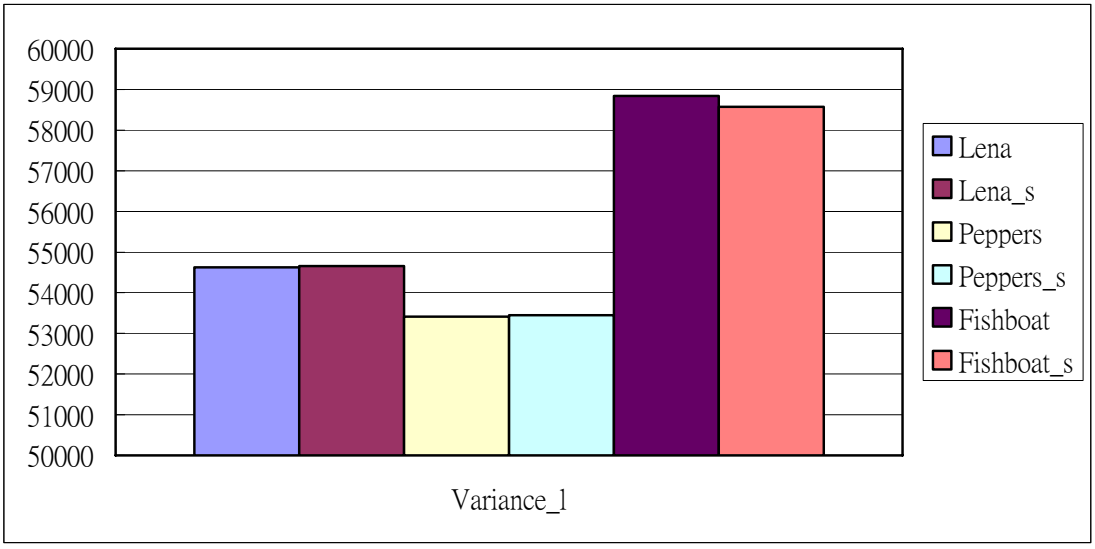


圖 4- 25 CKLS_Variance（低頻區特徵值）

Variance（中、高頻區）

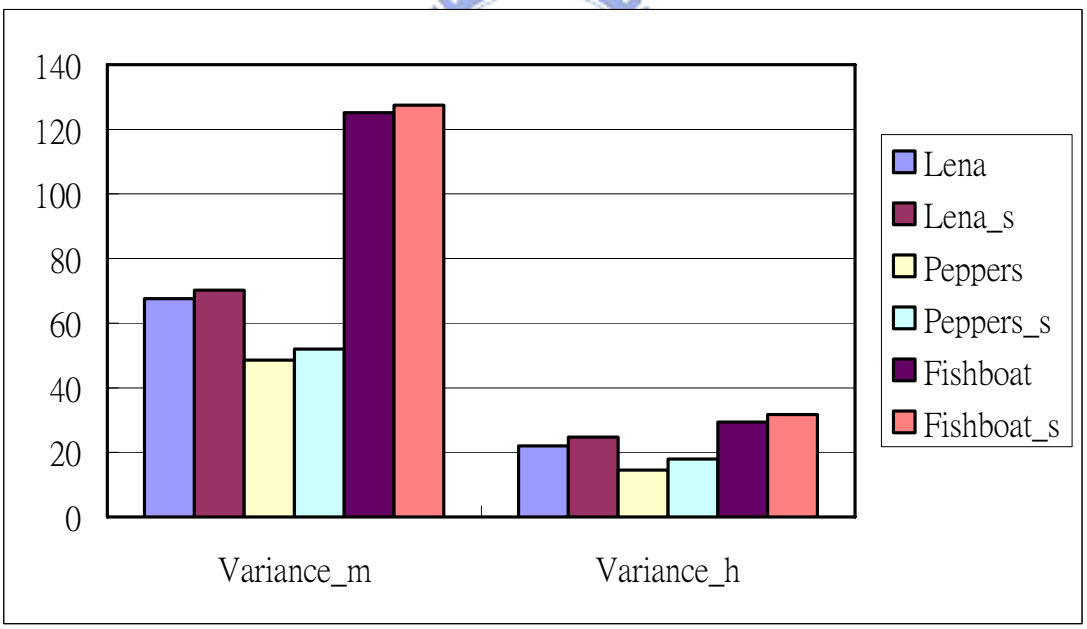


圖 4- 26 CKLS_Variance（中高頻區特徵值）

由圖 4-23 及 4-24 可以看出，不論是在低頻區或中高頻區，有無攪亂的圖片，其係數間的變異程度都沒有產生太大的差異。

Energy（低頻區）

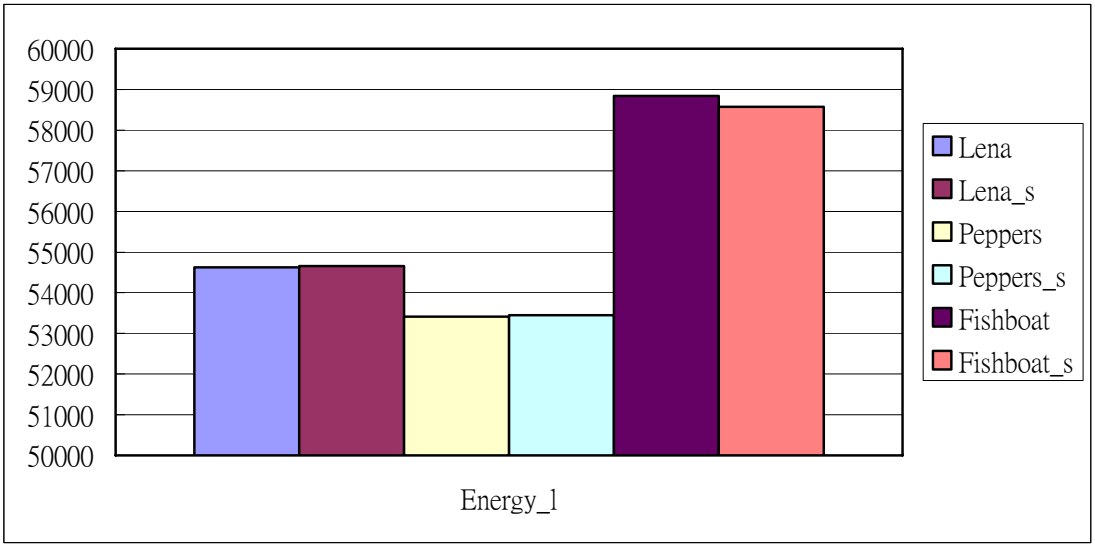


圖 4- 27 CKLS_Energy（低頻區特徵值）

Energy（中、高頻區）

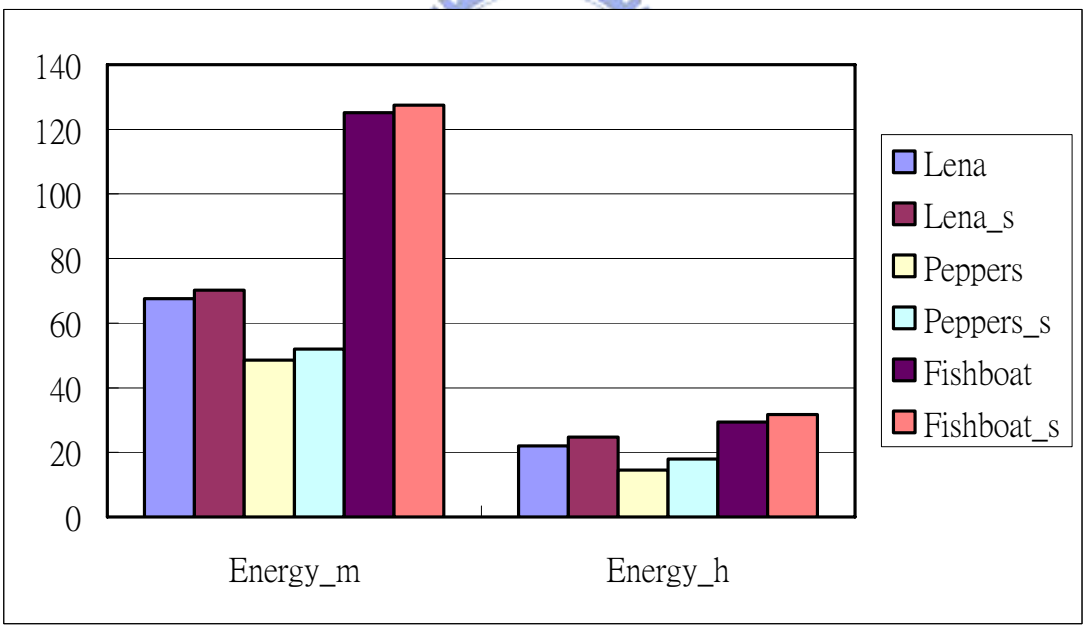


圖 4- 28 CKLS_Energy（中高頻區特徵值）

由圖 4-25 及 4-26 可以看出，不論是在低頻區或中高頻區，有無攪亂的圖片，其頻區的能量都幾乎為一致。

Log Energy（低、中、高頻區）

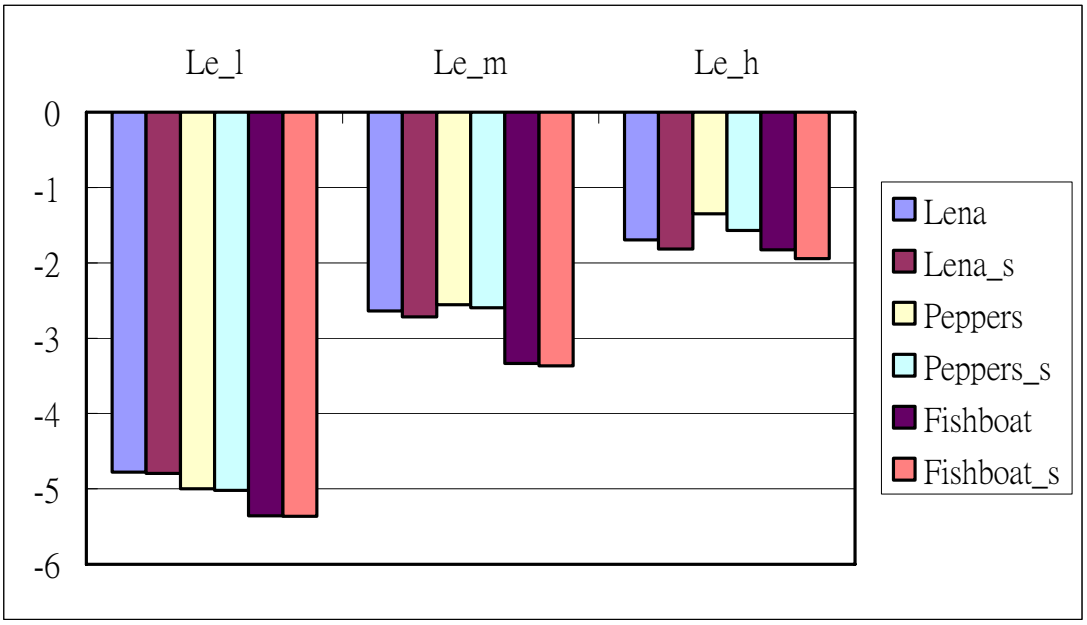


圖 4- 29 CKLS_Log Energy（低中高頻區特徵值）

由圖 4-27 可以看出，不論是在低頻區或中高頻區，有無攪亂的圖片，其 Log Energy 皆近乎相同。

Shannon（低頻區）

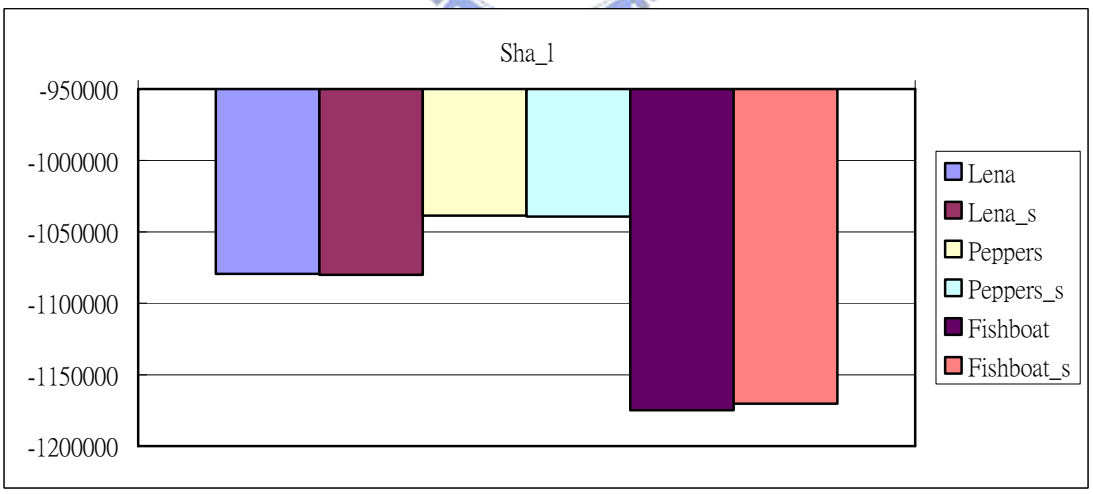


圖 4- 30 CKLS_Shannon（低頻區特徵值）

Shannon（中、高頻區）

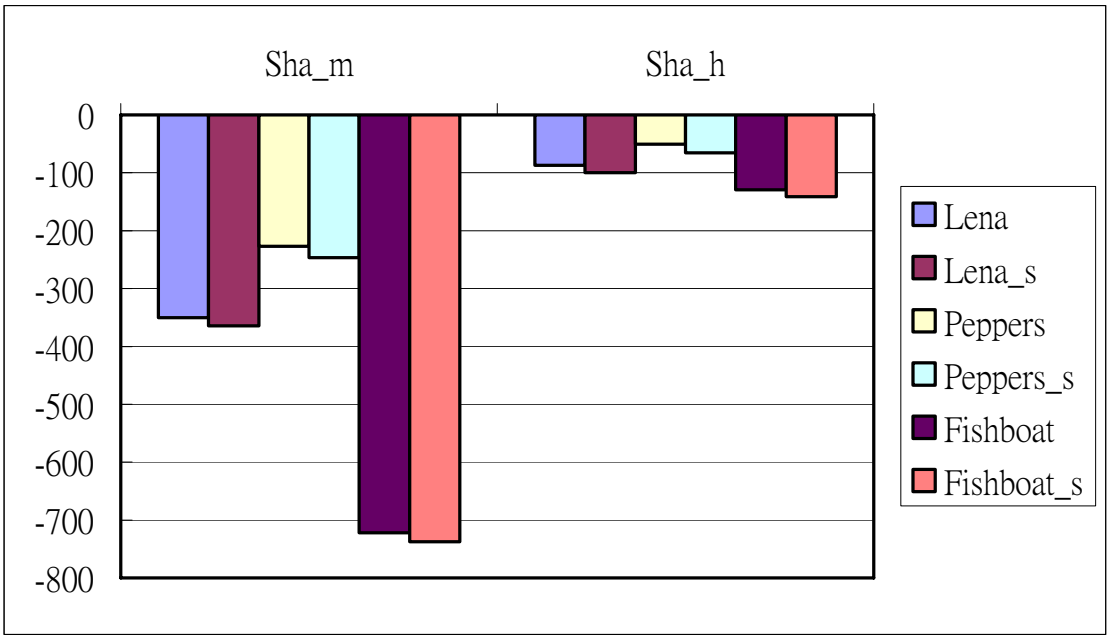


圖 4- 31 CKLS_Shannon（中高頻區特徵值）

由圖 4-28 及 4-29 可以看出，不論是在低頻區或中高頻區，有無攪亂的圖片，其 Shannon Entropy 幾乎無異。

Entropy（低頻區）

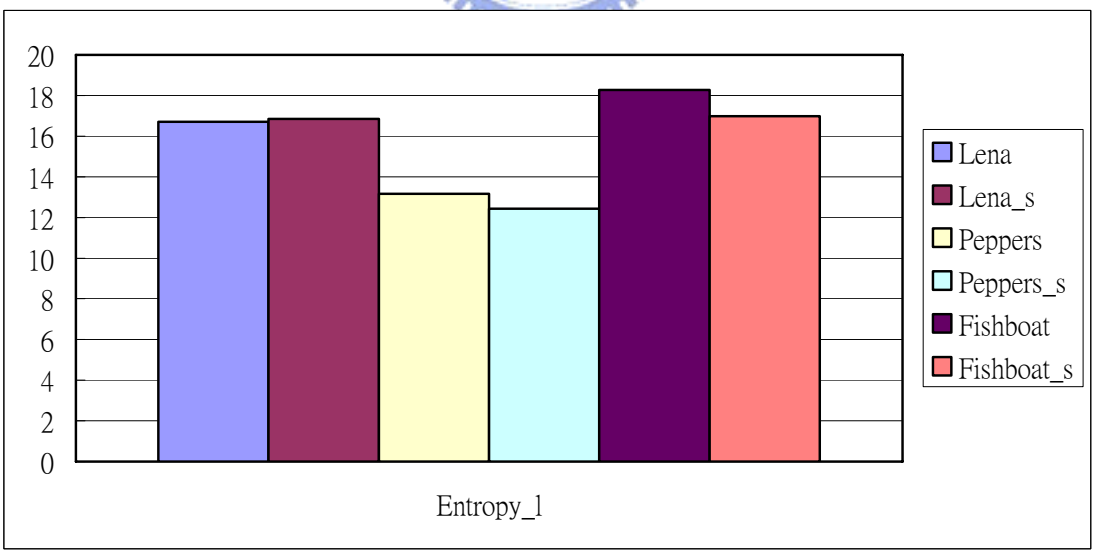


圖 4- 32 CKLS_Entropy（低頻區特徵值）

Entropy（中、高頻區）

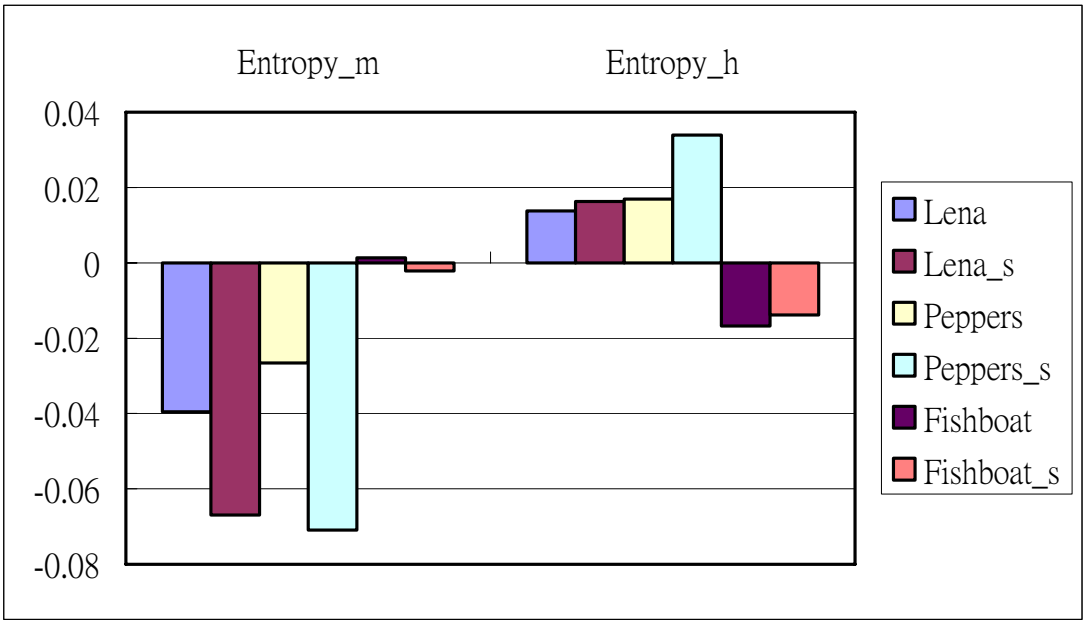


圖 4- 33 CKLS_Entropy（中高頻區特徵值）

由圖 4-30 及 4-31 可以看出，Lena 及 Peppers 這二張圖，其有無攪亂的圖片，在中頻區產生較大的差異，但是 Fishboat 卻未出現相同的現象。

Skewness（低頻區）

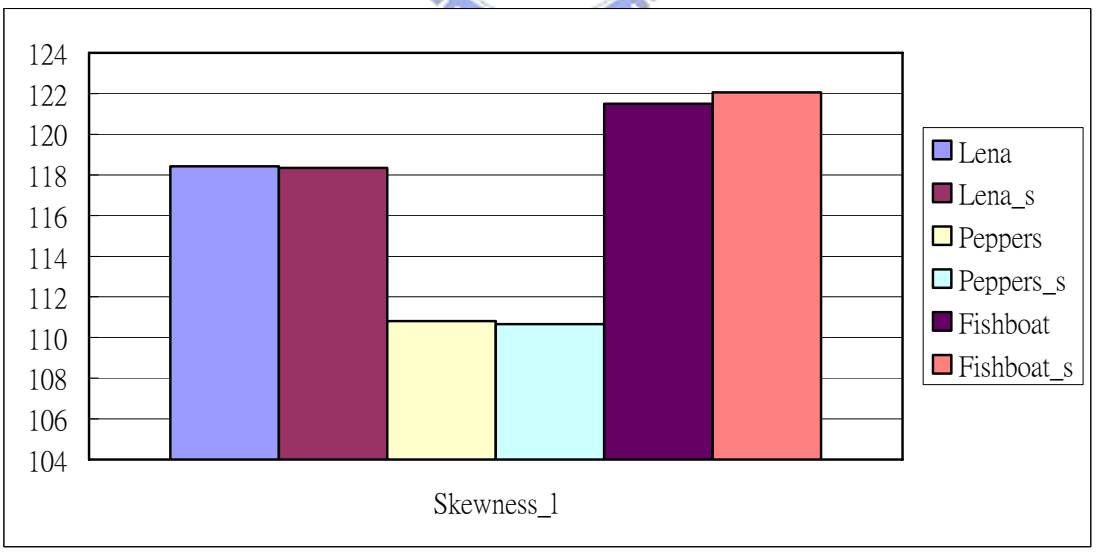


圖 4- 34 CKLS_Skewness（低頻區特徵值）

Skewness（中、高頻區）

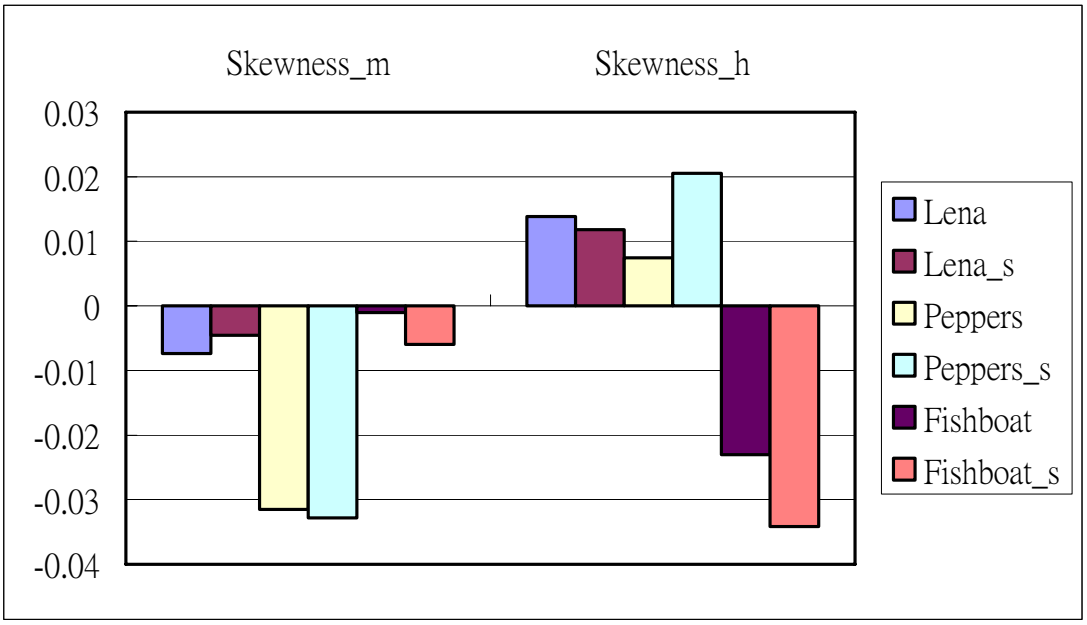


圖 4- 35 CKLS_Skewness（中高頻區特徵值）

由圖 4-32 及 4-33 可以看出，不論是在低頻區或中高頻區，有無攪亂的圖片，其偏態都呈現一致。

Kurtosis（低頻區）

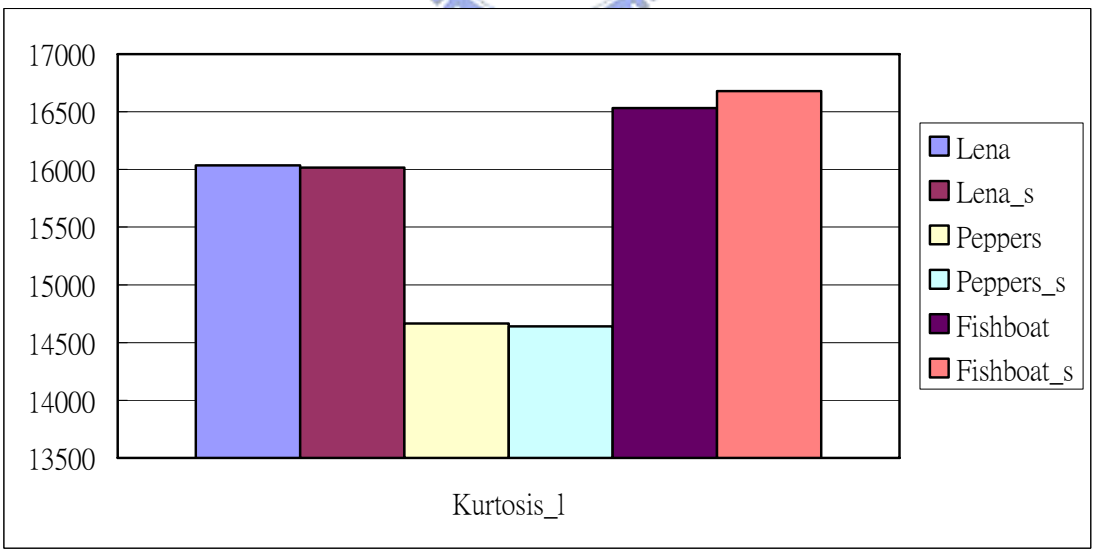


圖 4- 36 CKLS_Kurtosis（低頻區特徵值）

Kurtosis（中、高頻區）

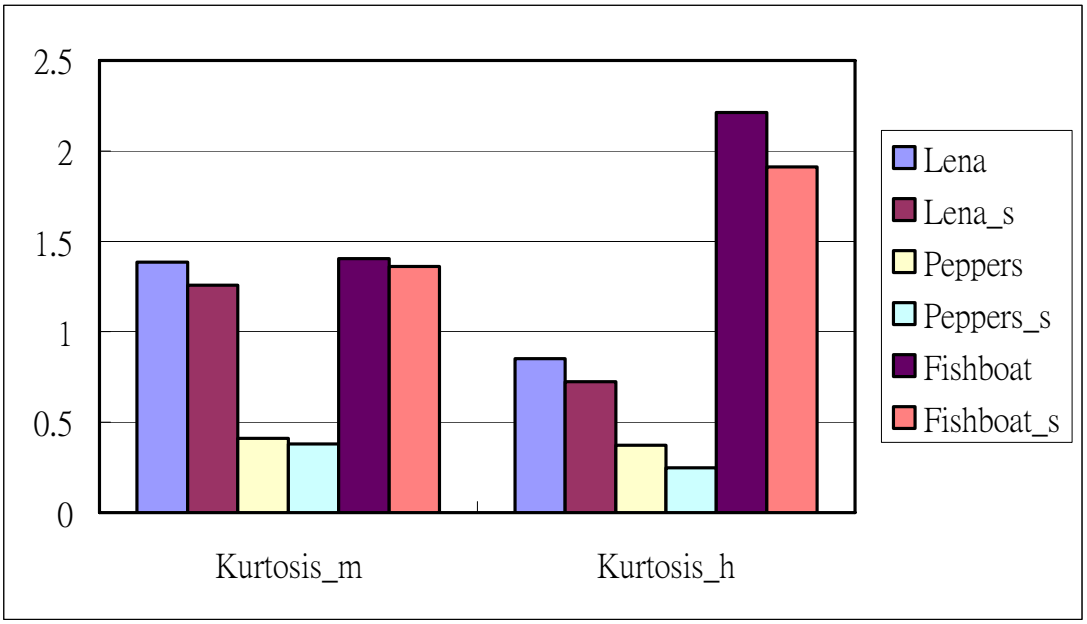


圖 4- 37 CKLS_Kurtosis（中高頻區特徵值）

由圖 4-34 及 4-35 可以看出，不論是在低頻區或中高頻區，有無攪亂的圖片，其峰態都呈現一致。

接下來以 CHU 浮水印演算法嵌入的實驗中，我們所採用的圖片為第一張 Subimage（大小為 128x128），其經過 DCT 轉換後，各頻區的係數個數如圖 4-36 所示，低頻區的數值共有 3278 個，中頻區有 11556 個，高頻區有 1550 個。

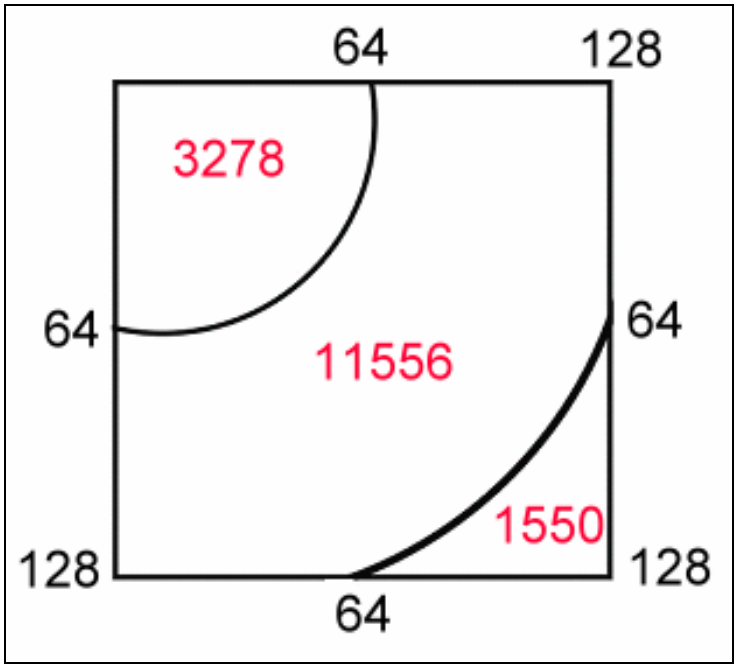


圖 4- 38 一張 128x128 圖片各頻區的係數個數分佈圖

Range（低頻區）

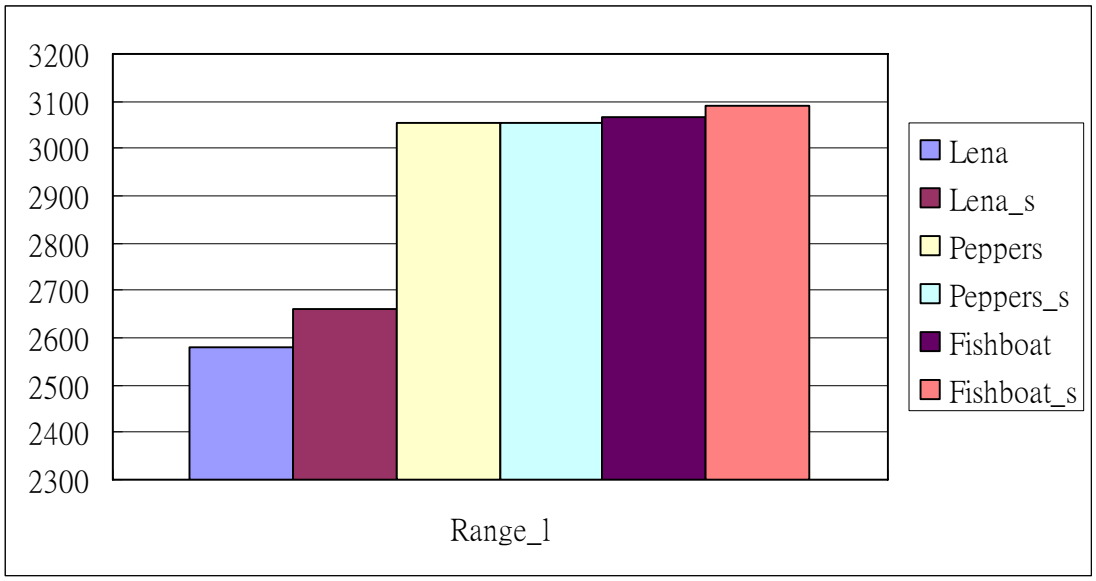


圖 4- 39 CHU_Range（低頻區特徵值）

Range（中、高頻區）

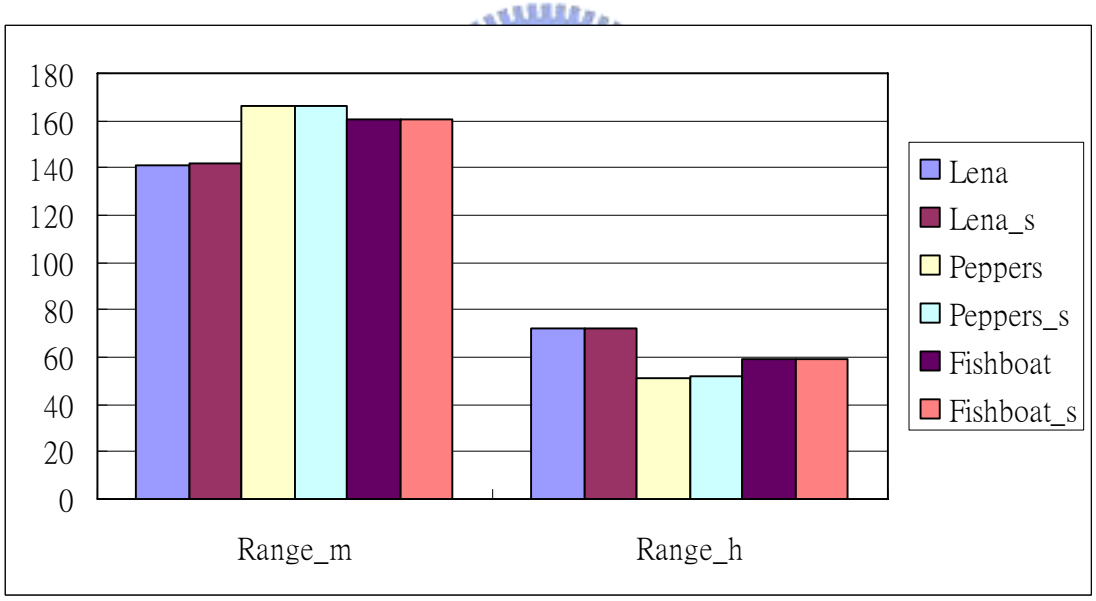


圖 4- 40 CHU_Range（中高頻區特徵值）

Variance（低頻區）

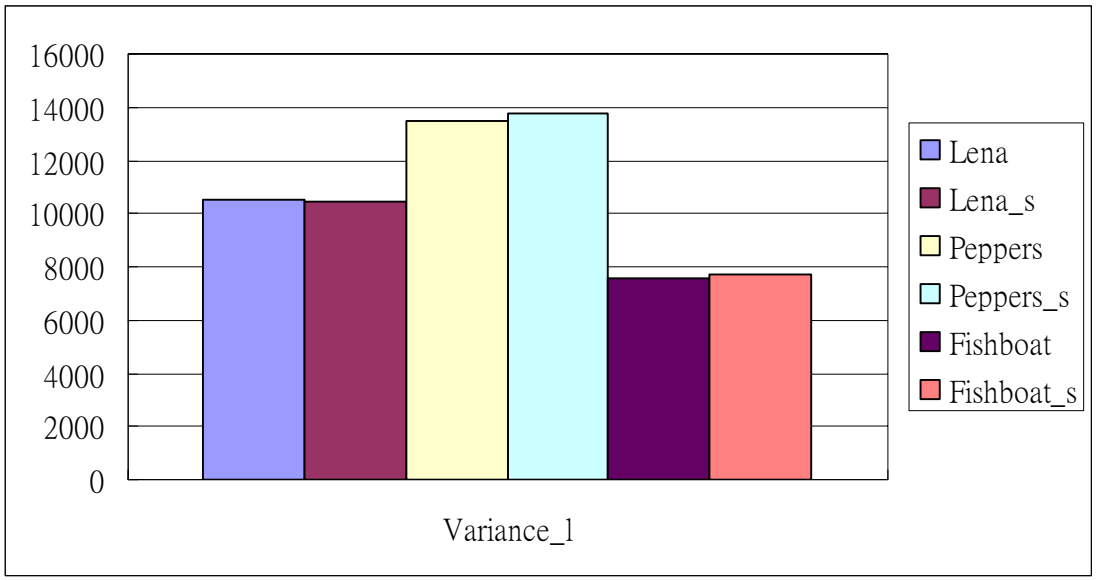


圖 4- 41 CHU_Variance（低頻區特徵值）

Variance（中、高頻區）

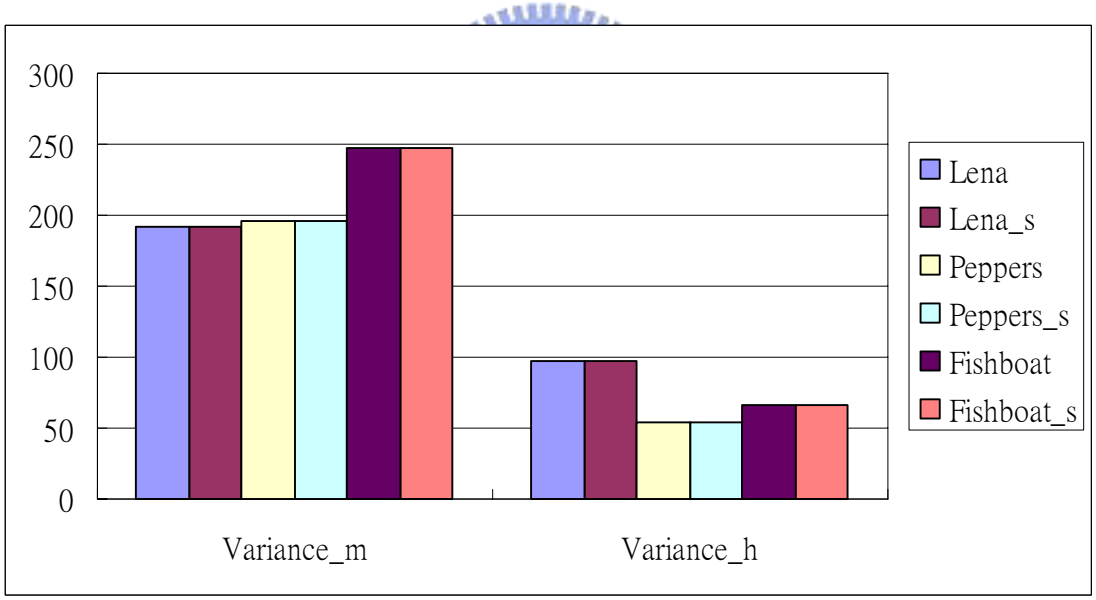


圖 4- 42 CHU_Variance（中高頻區特徵值）

Energy（低頻區）

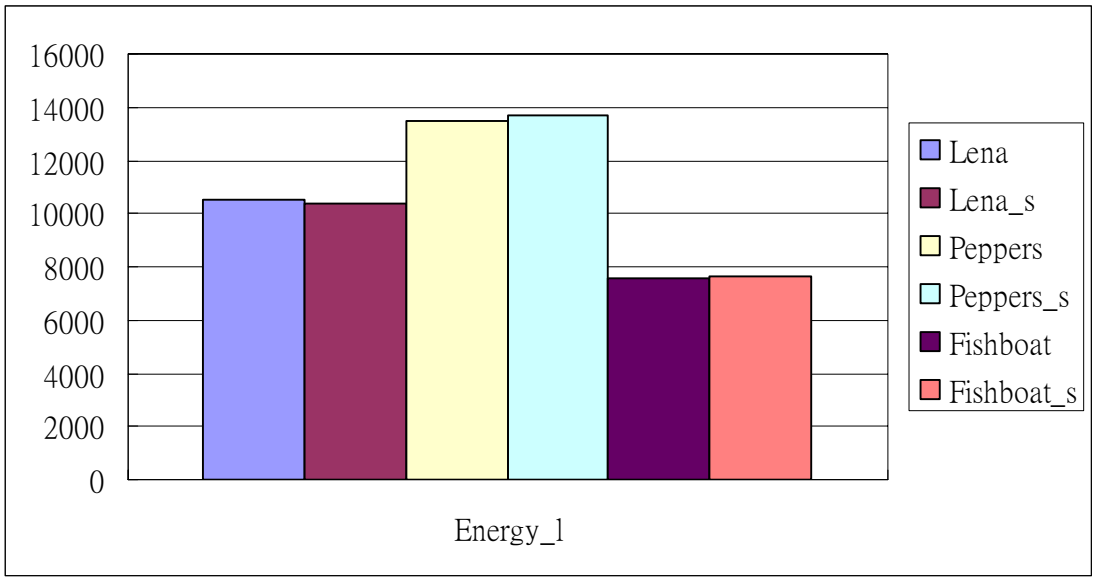


圖 4- 43 CHU_Energy（低頻區特徵值）

Energy（中、高頻區）

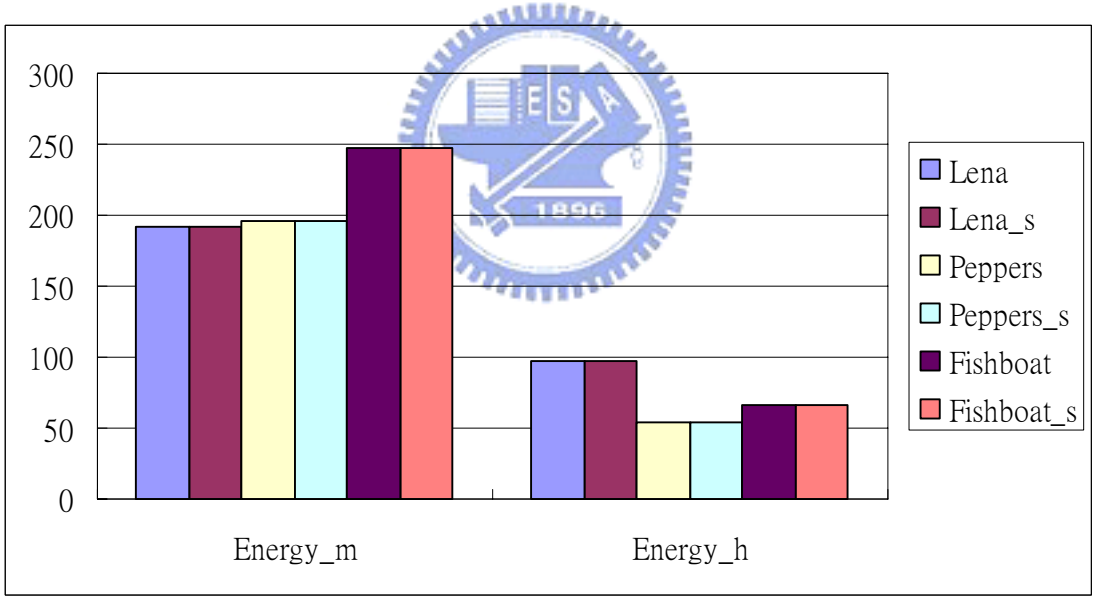


圖 4- 44 CHU_Energy（中高頻區特徵值）

Log Energy（低、中、高頻區）

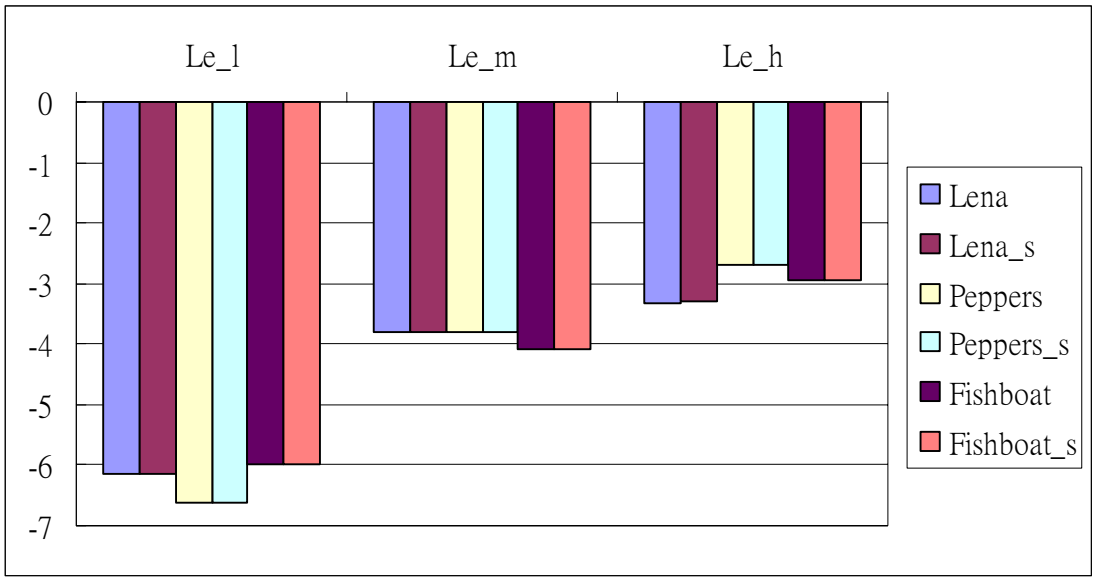


圖 4- 45 CHU_Log Energy（低中高頻區特徵值）

Shannon（低頻區）

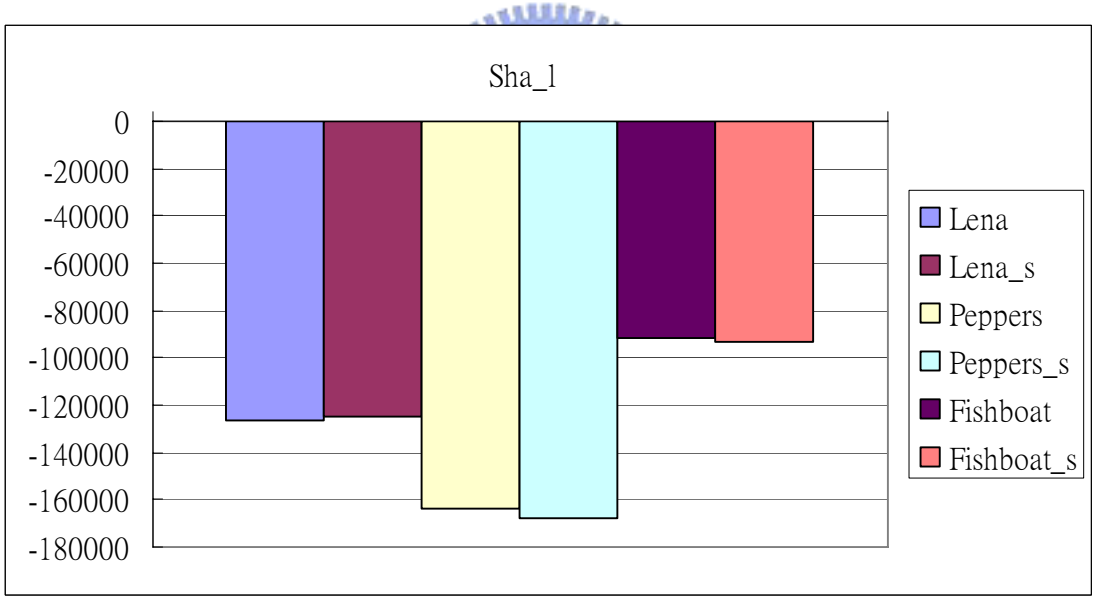


圖 4- 46 CHU_Shannon（低頻區特徵值）

Shannon（中、高頻區）

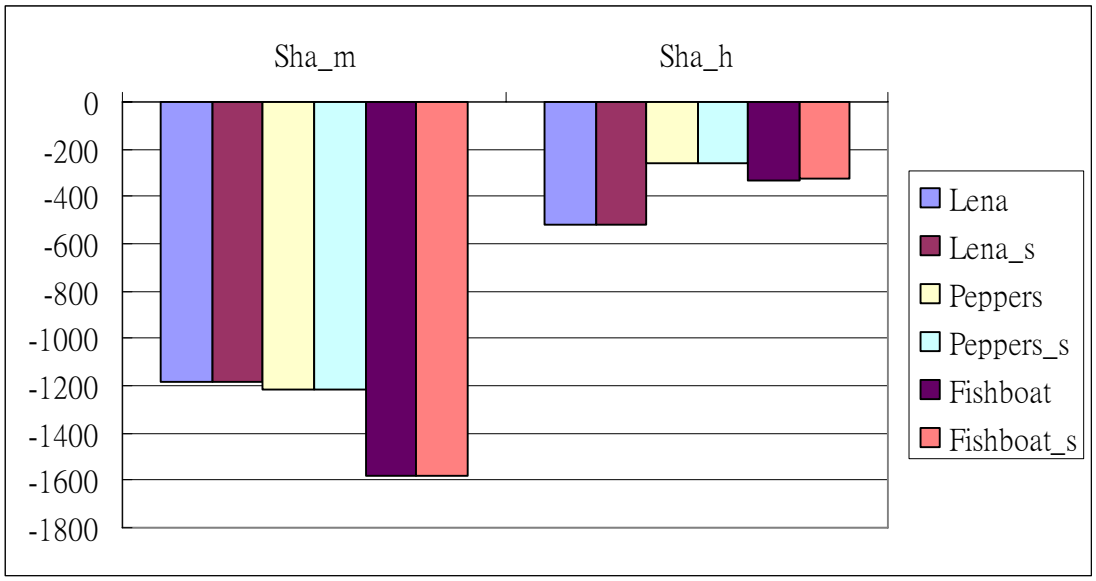


圖 4- 47 CHU_ Shannon（中高頻區特徵值）

Entropy（低頻區）

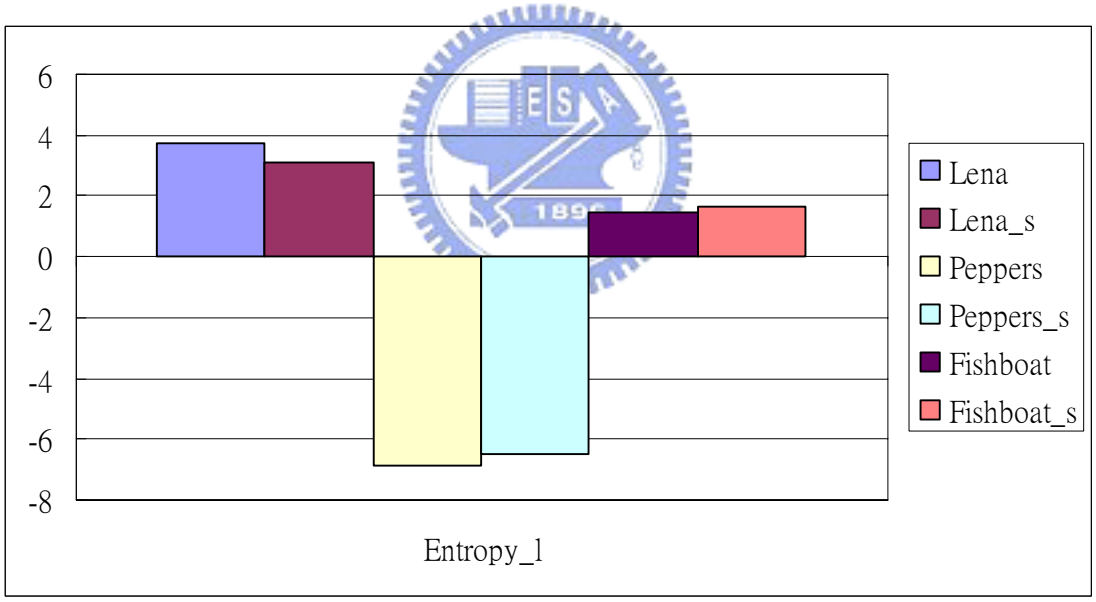


圖 4- 48 CHU_Entropy（低頻區特徵值）

Entropy（中、高頻區）

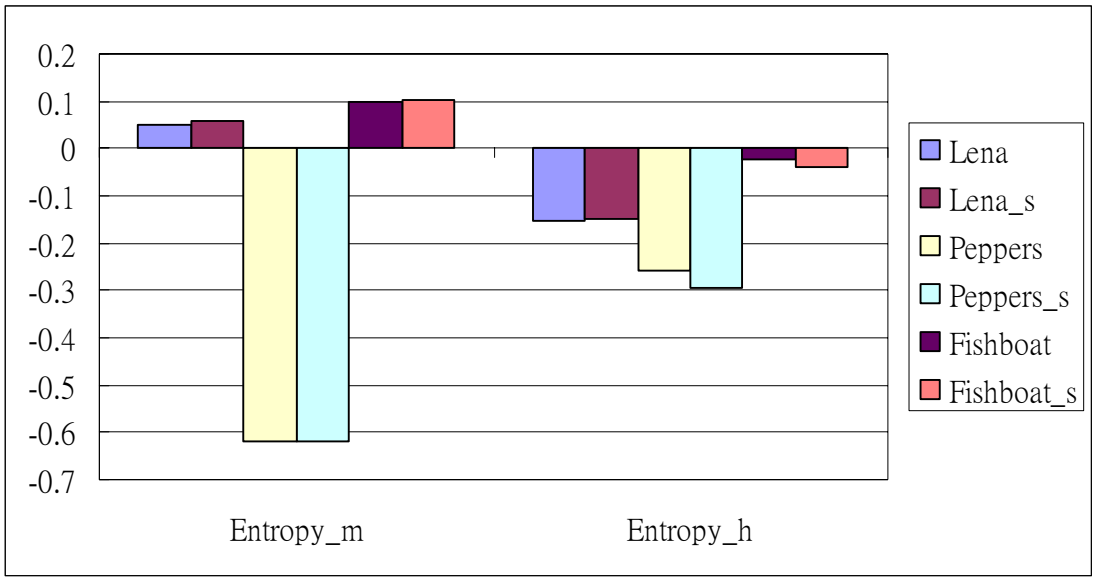


圖 4- 49 CHU_Entropy（中高頻區特徵值）

Skewness（低頻區）

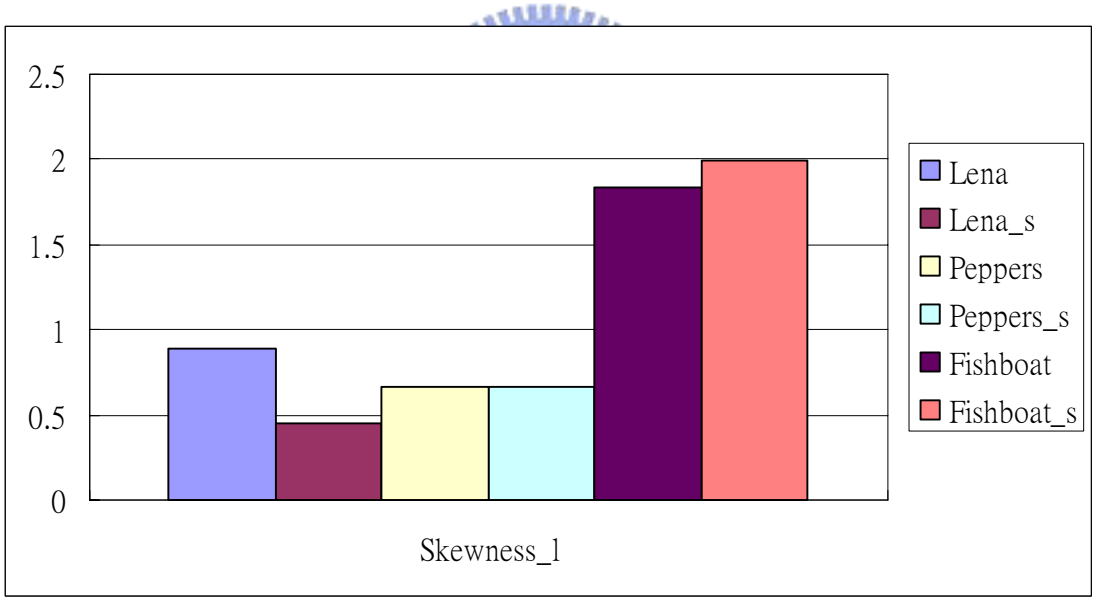


圖 4- 50 CHU_Skewness（低頻區特徵值）

Skewness（中、高頻區）

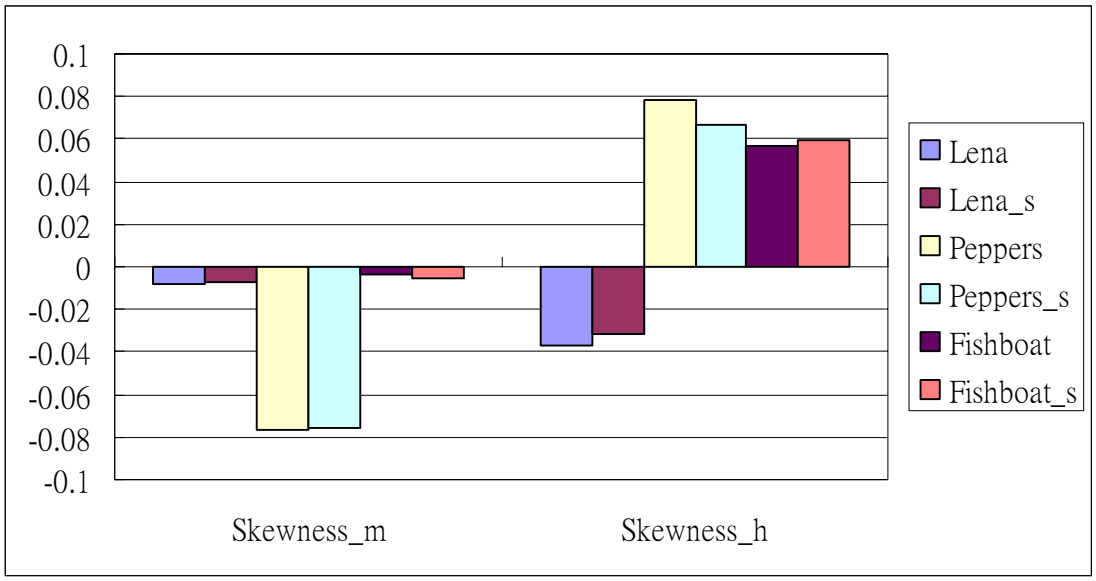


圖 4- 51 CHU_Skewness（中高頻區特徵值）

Kurtosis（低頻區）

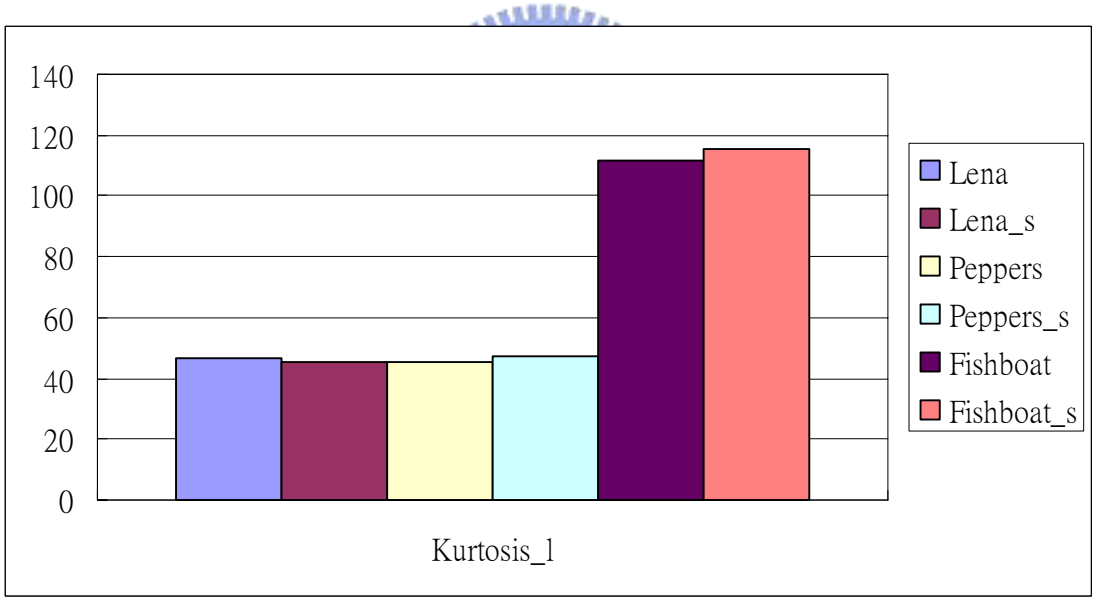


圖 4- 52 CHU_Kurtosis（低頻區特徵值）

Kurtosis（中、高頻區）

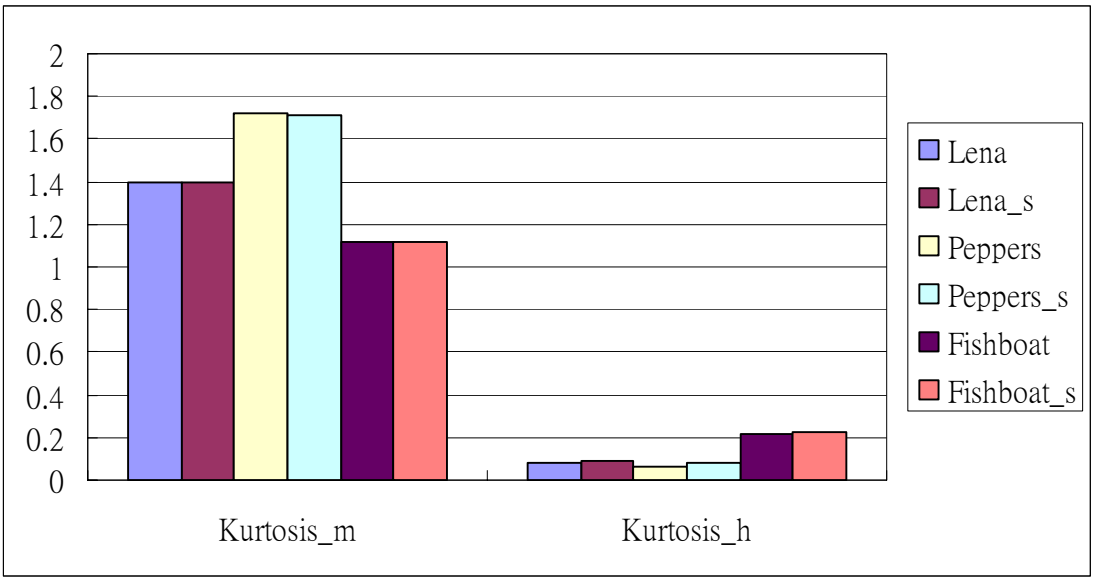


圖 4- 53 CHU_Kurtosis（中高頻區特徵值）

4.5.2 特徵值分析小結

經由上面特徵值分析的結果可以看出，原始的 CKLS 及 CHU 浮水印演算法與本文所提出來的亂攪亂演算法，在圖片各頻區的特徵值上並不會產生相當大的差異，這一點可使得密碼分析者無從得知嵌入時是否經過攪亂前處理，縱然試圖攻擊後，我們還是能夠擷取出正確的浮水印。

五、結論與未來展望

5.1 結論

密碼分析攻擊對於 CHU 及 CKLS 演算法具有相當顯著的攻擊成效，當 correlation 降到 0.2 以下時，仍能將攻擊圖片的 PSNR 維持在 30db 以上。圖片攪亂的觀念多用於資訊隱藏，其目的是將藏有秘密訊息的圖片打亂，增加攻擊者發現私密訊息的困難度。因此本文引用此一概念，針對 CHU 及 CKLS 演算法進行安全性上的補強，在原本的浮水印演算法上加入攪亂前處理，即是在嵌入浮水印之前，先將原圖以亂數打亂，再嵌入浮水印。浮水印嵌入完成後，再以相同的亂數種子（seed）將圖片還原。

由實驗結果得知，本文所提出的攪亂前處理，僅需增加少量的計算與儲存成本，就能夠有效地防禦密碼分析，將攻擊過後的 correlation 維持在 0.3 以上，更進一步地提升 CKLS 演算法對於影像處理攻擊的強韌性。

有鑑於我們等於是將圖片結構作了改變後再嵌入浮水印，因此進一步地比較了圖片特徵值，試圖發現相同演算法對於有加上攪亂前處理與沒有加上攪亂前處理的圖片嵌入浮水印時，是否會產生差異，實驗結果顯示，二者間並不存在著明顯差異的特徵值，如此一來，更增加攻擊者分析的困難度。

5.2 未來展望

本文最後提出三點未來可繼續研究的方向：

1. 將分析範圍拉大：

   由於密碼分析是針對疑似嵌有浮水印的係數進行重建，當我們加上攪亂前處理後，浮水印嵌入的位置就會變的無法預知，若採取拉大重建範圍，也許就能將攻擊效果提升，但相對地，重建的係數愈多，對於圖片品質破壞的愈嚴重。

2. 將圖片攪亂延伸至其它浮水印演算法

   由於圖片攪亂只是對圖片進行前置處理，不更動原有的演算法，卻能帶來更好的安全性，因此可嘗試套用至其它演算法上。

3. 使用不同的攪亂種子

   本研究中只有使用亂數及 Fibonacci，可嘗試尋找其它攪亂方法加以實驗。

參考文獻

英文資料：

1. Tanmoy Kanti Das and Subhamoy Maitra, “Cryptanalysis of correlation-based Watermarking Schemes Using Single Watermarked Copy”, IEEE Signal Processing Letters, VOL. 11, NO. 4, pp. 446-449, Apr. 2004
2. Ingemar J. Cox, Joe Kilian, F. Thomson Leighton, and Talal Shamoon, “Secure Spread Spectrum Watermarking for Multimedia”, IEEE Transactions on Image Processing, Vol. 6, No. 12, pp. 1673-1687, Dec. 1997
3. Tanmoy Kanti Das, Subhamoy Maitra, and Jianying Zhou, “Cryptanalysis of Chu’s DCT Based Watermarking Scheme”, IEEE Transactions on Multimedia, Vol. 8, No. 3, pp. 629-632, Jun. 2006
4. Fabien A. P. Petitcolas, Ross J. Anderson and Markus G. Kuhn, “Information Hiding-A Survey”, Proceedings of the IEEE, special issue on protection of multimedia content, 87(7), pp. 1062-1078, Jul. 1999.
5. J. Cox, M. L. Miller, and A. L. McKellips, “Watermarking as Communications with Side Information”, Proceedings. of the IEEE, vol. 87, no. 7, pp. 1127-1141, Jul 1999.
6. F. Hartung and B. Girod, “Digital Watermarking of Uncompressed and Compressed Video”, Signal Processing Special Issue on Watermarking 66, pp. 283-302, May 1998.
7. Stefano Bossi, Francesco Mapelli and Rosa Lancini, “Semi-Fragile Watermarking for Video Quality Evaluation in Broadcast Scenario”, ICIP 2005, Vol. 1, pp. 161-4, SEPTERMBER 2005.
8. T. Kalker, G. Depovere, J. Haitsma, and M. Maes, “A Video Watermarking System for Broadcast Monitoring”, Proc. SPIE 3657, pp. 103–112, Jan 1999.
9. S. Saha and Rao Vemuri, “Analysis-based Adaptive Wavelet Filter Selection in Lossy Image Coding Schemes”, Proc. IEEE International Symposium on Circuits and Systems, vol. 3, pp. 29-32, May 2000.
10. Sin-Joo Lee and Sung-Hwan Jung, “A Survey of Watermarking Techniques applied to multimedia”, Proc. of ISIE Vol. 1, pp. 272-276, Jun. 2001.
11. Wei Lu, Hong-Tao Lu, and Fu-Lai Chung, “Chaos-Based Spread Spectrum Robust Watermarking in DWT Domain”, Proceedings of the Fourth International

Conference on Machine Learning and Cybernetics, pp. 18-21 Aug. 2005.

12. Langelaar, GC, Lagendijk RL, Biemond J, “Real-Time Labeling of MPEG-2 Compressed Video”, Journal of Visual Communication and Image Representation, Volume 9, pp. 256-270, December 1998.
13. Stallings, William(2003), “Cryptography and network security :principles and practices (3rd edition) ”, Upper Saddle River, N.J. :Prentice Hall,c2003.
14. Wai C. Chu, "DCT-Based ImageWatermarking Using Subsampling", IEEE Transations on Multimedia, Vol. 5, No. 1, pp 34-38, March 2003.

中文資料：

15. 張真誠（2001），數位影像處理技術，台北市：松崗出版社
16. 王旭正（2006），資訊與網路安全，台北市：博碩出版社。

附錄 I

CKLS－密碼分析攻擊多項式係數

Lena	x^3	$-x^2$	$+x$	$+c$
1×1	0.0000018403652	0.0018349064459	0.6086238929602	56.3887662087709
2×2	0.0000018589447	0.0018622852332	0.6201270720530	55.3705619645752
4×4	0.0000018350159	0.0018240635911	0.6021206552494	58.0772611292869
8×8	0.0000017904215	0.0017750062117	0.5922145092945	58.6101949296808
16×16	0.0000018262668	0.0018109353212	0.6004080433092	58.0353621068753

Peppers	x^3	$-x^2$	$+x$	$+c$
1×1	0.0000021286876	0.0021266485214	0.7112691300790	64.0810787887185
2×2	0.0000021383883	0.0021428286518	0.7195381654425	63.0522715186689
4×4	0.0000021715662	0.0021917901687	0.7406876842417	60.9346759779473
8×8	0.0000021170146	0.0021011589738	0.6990401946464	66.9718506321533
16×16	0.0000021433145	0.0021526190012	0.7192701026864	66.1423342357211

Goldhill	x^3	$-x^2$	$+x$	$+c$
1×1	0.0000011001707	0.0010846380786	0.3733678659716	60.5973250337511
2×2	0.0000010945865	0.0010763528443	0.3697374690458	61.1402963092532
4×4	0.0000010603815	0.0010326922073	0.3581288028726	61.6639638154702
8×8	0.0000011202986	0.0011090194310	0.3819372604481	62.4012644900158
16×16	0.0000010491407	0.0010036349246	0.3444379161794	63.2934688281566

Elaine	x^3	$-x^2$	$+x$	$+c$
1×1	0.0000014044225	0.0013651660344	0.4943582293620	63.1523428502006
2×2	0.0000014024126	0.0013599250324	0.4912024410051	63.7660827632390
4×4	0.0000013904675	0.0013435615318	0.4841270197544	65.1449295705606
8×8	0.0000013929798	0.0013402120981	0.4821592285757	66.1632150311852
16×16	0.0000014210432	0.0013762120629	0.4910630081855	66.1106309689658

Fishboat	x^3	$-x^2$	$+x$	$+c$
1×1	0.0000014213177	0.0013723139905	0.4458292084504	69.0722769228469
2×2	0.0000014275564	0.0013867412668	0.4541484447197	68.3064947729818
4×4	0.0000014795126	0.0014632428081	0.4871818627813	64.4398021160669
8×8	0.0000014428100	0.0014071436667	0.4607365985003	69.1099205483897
16×16	0.0000014995308	0.0014985605905	0.5126561822881	57.0316111492539

F16	x^3	$-x^2$	$+x$	$+c$
1×1	0.0000016657449	0.0016737653051	0.5746513957054	87.9519494389624
2×2	0.0000016692841	0.0016763677773	0.5734838974693	88.8484648773210
4×4	0.0000016801425	0.0016883392706	0.5760735744385	89.1346750915236
8×8	0.0000017994728	0.0018836191768	0.6740395188739	76.0160746364545
16×16	0.0000016060602	0.0015934792021	0.5526556963860	89.6652999725466

Bridge	x^3	$-x^2$	$+x$	$+c$
1×1	0.0000017703005	0.0018047751258	0.5953459402303	76.0307533884352
2×2	0.0000017740165	0.0018111739252	0.5987878821898	75.7167558204128
4×4	0.0000017800041	0.0018150033055	0.5961166440878	77.3189634197952
8×8	0.0000018004049	0.0018635153838	0.6252275376520	75.6104950068038
16×16	0.0000017994863	0.0018517835452	0.6178463891594	74.4903633276754

Couple	x^3	$-x^2$	$+x$	$+c$
1×1	0.0000012823733	0.0012631285645	0.4276527688332	64.5847107788668
2×2	0.0000012903517	0.0012730548695	0.4304490216891	64.7729622585357
4×4	0.0000012825471	0.0012623105717	0.4269360824747	64.7464923564512
8×8	0.0000012641854	0.0012357228917	0.4185826814527	65.2221043095760
16×16	0.0000013243640	0.0013218502456	0.4527063562813	63.2947191153489

附錄Ⅱ

CHU－密碼分析攻擊多項式係數（Random）

Bridge	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000008495934	0.0008615911825	0.3452409870263	14.8801511801712
Subimag2	0.0000008206507	0.0008099734478	0.3205136471542	12.2724899856594
Subimag3	0.0000008872967	0.0009055276746	0.3570666111016	15.4844793440179
Subimag4	0.0000008451377	0.0008407182337	0.3298401513596	13.2085107237481

Couple	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000007861407	0.0008321240190	0.3236817948779	14.6222707003571
Subimag2	0.0000007664370	0.0007971652419	0.3064221762753	13.2473499741113
Subimag3	0.0000005942068	0.0005807121056	0.2387311245636	8.6059380848351
Subimag4	0.0000005636860	0.0005347706352	0.2222325379296	8.8112120189007

Elaine	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000006336527	0.0006233236360	0.2945134458229	10.6342774627092
Subimag2	0.0000006899658	0.0006961659413	0.3179452553346	12.2076105980395
Subimag3	0.0000006420654	0.0006339521596	0.3001801281725	12.4827059242025
Subimag4	0.0000006790824	0.0006723216754	0.3067327659953	12.4944592104248

F16	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000007353249	0.0007306182588	0.3316906439065	10.8352335263953
Subimag2	0.0000007893963	0.0008090203774	0.3614926242388	13.6921543384169
Subimag3	0.0000007802188	0.0007956245126	0.3585965242693	12.9661880986132
Subimag4	0.0000008322533	0.0008660414791	0.3806979209367	13.7275567062925

Fishboat	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000007039653	0.0007014668274	0.2950501430001	10.9353158527601
Subimag2	0.0000006991733	0.0007034247458	0.2986263106046	10.7809574695302
Subimag3	0.0000006114617	0.0005809078861	0.2572273967446	9.9709837838036
Subimag4	0.0000006137098	0.0005889968989	0.2597223709270	8.8230813631035

Goldhill	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000005272957	0.0005241743454	0.2307576607192	8.6596217265810
Subimag2	0.0000005385200	0.0005387410652	0.2347723028905	8.3949463814952
Subimag3	0.0000005476656	0.0005550475683	0.2440594721995	10.0094184327909
Subimag4	0.0000005455457	0.0005610355226	0.2507476845635	10.5550574738850

Lena	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000008618935	0.0008153552040	0.3073683174453	13.4316701503731
Subimag2	0.0000008697933	0.0008330639117	0.3170130620486	14.4639489216970
Subimag3	0.0000008647444	0.0008176867100	0.3054452571452	12.2663206780298
Subimag4	0.0000008908068	0.0008568357565	0.3220275329525	13.7113156379482

Peppers	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000009638366	0.0009698438698	0.4047239815156	16.2357607590466
Subimag2	0.0000010250420	0.0010356483781	0.4200366936500	17.0507597663418
Subimag3	0.0000009465454	0.0009240427603	0.3782212875470	15.2747466113860
Subimag4	0.0000010440150	0.0010406364726	0.4114429464573	17.1738324960525

CHU－密碼分析攻擊多項式係數（Fibonacci）

Bridge	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000008424434	0.0008499419368	0.3402678986704	14.3385118840979
Subimag2	0.0000008215857	0.0008125895029	0.3217122758811	12.4056425673807
Subimag3	0.0000008872967	0.0009055276746	0.3570666111016	15.4844793440179
Subimag4	0.0000008451377	0.0008407182337	0.3298401513596	13.2085107237481

Couple	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000008741938	0.0009197361910	0.3584397584948	15.8067724240227
Subimag2	0.0000008526798	0.0008855766008	0.3438631819395	14.8555811946271
Subimag3	0.0000006859999	0.0006750283218	0.2775039473860	10.4086150848161
Subimag4	0.0000006441143	0.0006135973050	0.2551663230184	10.1036054134647

Elaine	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000006264922	0.0006135586494	0.2903658733382	10.1025460111932
Subimag2	0.0000006923659	0.0006973672522	0.3179425051864	12.2633062296945
Subimag3	0.0000006420654	0.0006339521596	0.3001801281725	12.4827059242025
Subimag4	0.0000006790824	0.0006723216754	0.3067327659953	12.4944592104248

F16	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000007270790	0.0007158562396	0.3272140268542	11.6874929622900
Subimag2	0.0000007619649	0.0007626729474	0.3407857715480	12.2596991152323
Subimag3	0.0000007653752	0.0007787023626	0.3536596954037	13.0998243915456
Subimag4	0.0000008092373	0.0008291185930	0.3633350484984	12.4068227580337

Fishboat	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000006359264	0.0006349111070	0.2674310079456	10.0098211163331
Subimag2	0.0000006323430	0.0006375379487	0.2714620743486	10.2396686428137
Subimag3	0.0000005515030	0.0005240174381	0.2321047271530	9.0341445876293
Subimag4	0.0000005534861	0.0005313536469	0.2344418058621	8.0141583870291

Goldhill	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000005246967	0.0005197532262	0.2286359353040	8.4277043373577
Subimag2	0.0000005422357	0.0005447880779	0.2376018159977	8.7407955834228
Subimag3	0.0000005476656	0.0005550475683	0.2440594721995	10.0094184327909
Subimag4	0.0000005455457	0.0005610355226	0.2507476845635	10.5550574738850

Lena	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000008627199	0.0008171514502	0.3085514321547	13.4940546421212
Subimag2	0.0000008732944	0.0008392597839	0.3202114109926	14.9946373926105
Subimag3	0.0000008647444	0.0008176867100	0.3054452571452	12.2663206780298
Subimag4	0.0000008908068	0.0008568357565	0.3220275329525	13.7113156379482

Peppers	x^3	$-x^2$	$+x$	$-c$
Subimag1	0.0000009672007	0.0009767469402	0.4082494247502	16.7239473050586
Subimag2	0.0000010233929	0.0010319869051	0.4180237211957	16.5808779545205
Subimag3	0.0000009465454	0.0009240427603	0.3782212875470	15.2747466113860
Subimag4	0.0000010440150	0.0010406364726	0.4114429464573	17.1738324960525