影像處理作業

A watermarking-based image ownership and tampering authentication scheme  
基於浮水印技術圖像認證方法

實作

系級:資工碩一

學號:107522099

姓名:詹振宗

**摘要**

本作業依照論文

A watermarking-based image ownership and tampering authentication scheme 內的提出的影像浮水印技術時做出一個可執行程式，此方法不同於以往的浮水印技術無法定位修改處的缺點，此方法可定位出影像被竄改之處，為了使影像更容易實作，目前只時作出灰階浮水印

使用python實作

**相關工作**

影像浮水印技術分為

* fragile authentication

任何變動都是不允許的

* robust authentication

可認證經過濾，壓縮，裁剪，銳化，模糊的多媒體

影像浮水印依據實作方式分為

* labeling-based schemes

需額外檔案作為認證

屬於robust authentication

缺點增加記憶體與硬碟額外的負擔

* watermarking-based

直接嵌入在多媒體內

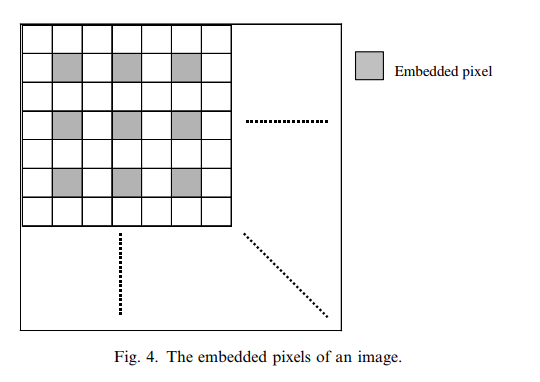
分為浮現式、隱藏式

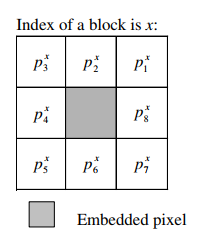
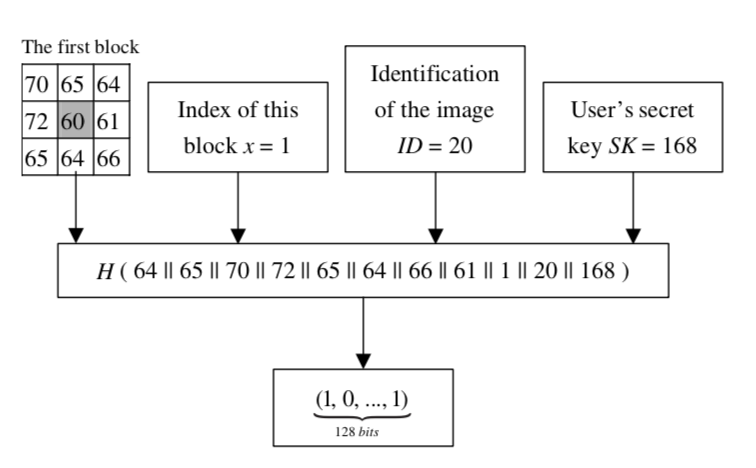
通常屬於fragile authentication

**實驗方法**

A.實作步驟

1. 經圖片切為多個Blocks(9X9)
2. 利用HASH得到每個Block 的特徵 (P1~P8)
3. 塞入中間的Pixel



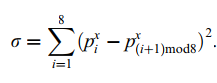


**嵌入認證**

要藏到LSB不能太多個Bits不然會失真=>把128Bits縮短，怎麼縮?

首先要決定要縮成多短(論文規定2~4個)

怎麼決定呢?



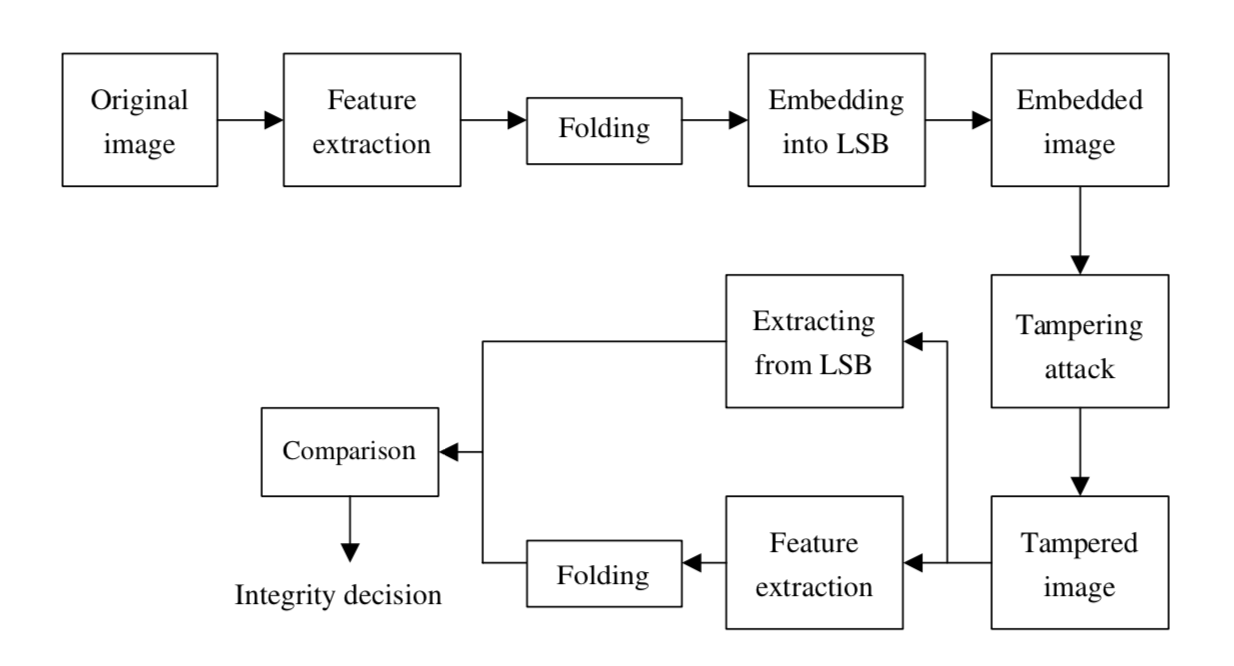
0 ≤ σ < 8 2個

8 ≤ σ <16 3個

16 ≤ σ ≤ 255 4個

**認證影像**

首先得到一張已作浮水印處理之影像

1. 照上述方法每個block的8個Pixel作運算得到認證值
2. 塞入值與中間的Pixel的LSB相比
3. 相同=>驗證成功
4. 不相同=>驗證失敗
5. 

**‘**

**實驗**

**實驗環境**

* 程式語言 python
* 實驗平台 windows 10
* 相依套件
  + OpenCV-python
  + Hashlib
  + numpy

本實作使用灰階圖片，因爲本方法使用fragile authentication

在此浮水印壓縮技術，任何變動都是不允許的，必須使用不壓縮或壓縮後不失真的影像格式，如bmp,png等影像格式



利用程式

Main.py

本程式讀取gray\_path變數內的未做浮水印灰階影像檔案路徑

本程式提供兩個函式，分別為

* add\_waterMarking()
  + 傳入值為灰階影像像素的二維陣列
  + 回傳已做浮水印處理後的灰階像素的二維陣列
* add\_waterMarking()
  + 傳入值為灰階影像像素的二維陣列
  + 回傳1)影像是否有被竄改2)被竄改的像素所在的九宮格位置