第 3 章資料隱藏方法

本章主要是分析 SVG 檔案內隱藏資料的方法，資訊隱藏（steganography or information hiding）是一種加密的方法，把重要資訊隱藏到一般資訊中。不同於純粹對重要資訊直接加密，資訊隱藏是藉由一個不會令人起疑的媒介，將機密資料隱藏其中。相對於直接加密，資訊隱藏的方式更具有保密的效果，如果再搭配既有的一些加密方法，將欲保密的資料先作一層加密的動作，再隱藏至一般資料中，則保密效果就更好了，等於有了雙重的保密。

本研究提出兩大類資料隱藏法：

1. Information Mixture（資訊融入法）：將想要隱藏的資訊，以數值方式編碼混入圖檔程式碼內，和 SVG 元素的屬性數值融合在一起，外觀上無法察覺，需經由特定程式，才能萃取出有意義的資訊，再經由解碼程式判別出訊息。以此方法製作的 SVG 文件，內含 Invisible Watermark，適合隱藏圖檔版權資訊，具有實用價值。
2. Information Add-On（資訊附加法）：將想要隱藏的資訊以原始面貌呈現在文件中，加以包裝、偽裝後，並不影響圖片的呈現結果，但隱藏方法是建立在一般人對 SVG 文件的陌生，因而不引起懷疑，適合製作 Visible Watermark。

3.1 資訊融入法

本系統隱藏資訊使用字元範圍 ASCII 碼從 33～126，此為可見字元的一部份,目的為方便由鍵盤輸入,先製作一個 94 個數字的函數 f(x)如表 3-1 亂數表一，固定在程式中，數值範圍 1～94。每次執行程式先灑亂數種子，此亂數種子記錄在 SVG 程式中，以利反解碼。假設產生的亂數種子為 n(0~93 表位移)，則 ASCII 碼 x(33～126)對應到亂數表：

編碼 → f (（x-32+n）mod 94) = y

反解 → $(f^{-1}(y)+32-n + 94) \bmod 94 = x$

例如：亂數表如下，假設亂數種子 n=70，字元 A 的 ASCII 碼為 x = 65

表 3 - 1 亂數表

51	7	87	90	85	65	47	71	35	61
81	57	70	67	45	3	37	48	13	60
17	56	69	92	23	5	43	49	38	40
84	36	63	12	59	91	4	72	29	44
19	25	46	21	34	62	80	74	83	26
58	73	79	9	75	18	31	88	53	15
76	94	30	68	32	11	28	41	42	22
14	20	24	77	16	27	54	78	89	52
64	6	10	93	1	39	2	33	8	55
50	66	82	86						

（65-32+70)mod 94 = 9 → f(9)=35 → y=35 SVG 明碼程式中藏入 35 的相關數值，

反解亂數表如下：

表 3 - 2 反解亂數表

85	87	16	37	26	82	2	89	54	83
66	34	19	71	60	75	21	56	41	72
44	70	25	73	42	50	76	67	39	63
57	65	88	45	9	32	17	29	86	30
68	69	27	40	15	43	7	18	28	91
1	80	59	77	90	22	12	51	35	20
10	46	33	81	6	92	14	64	23	13
8	38	52	48	55	61	74	78	53	47
11	93	49	31	5	94	3	58	79	4
36	24	84	62						

相關數值由 SVG 程式內抓出 35 ➔ $f^{-1}(y) = x$ ➔ $f^{-1}(35)= 9$ ➔（9+32-70 +94）mod 94 = 65 還原成可見字元『A』。配合公開金鑰和自訂私鑰，將產生 94×94×94=79524 種函數對應組合，應用於 SVG 常用元素有下列六種更換 Attribute 值的方式：

- 轉置隱藏（Hide Transform）
- 位移隱藏（Hide Move）
- 動畫隱藏 (Hide Motion)
- 顏色隱藏 (Hide Color)
- 外形隱藏 (Hide Shape)
- 矩陣隱藏 (Hide Matrix)

3.1.1 轉置隱藏（Hide Transform）

一般 SVG 圖檔在瀏覽時按下右鍵，可縮放四次，考慮圖形放大四次(16 倍)的情況下而不失真，更換 Attribute 的值，將字元編碼放在小數點下第四位和第

五位，一些特定 SVG 元素對於 Attribute 的值採四捨五入取整數值，不影響原始數值資料在圖形上的表現。

假設座標 A(x,y)在轉置隱藏下藏入數值δ，新座標 A'(x+δ,y-δ)，若放大 16 倍得座標 A(16x,16y)和新座標 A'(16x+16δ,16y-16δ)，將 16δ控制小於 0.1(亦即δ小於 0.00625)，則 16x+16δ和 16y-16δ的值，將與 16x 和 16y 相當接近，在瀏覽器上 A 和 A'呈現位置就難以分辨，例如座標 A(20.786,30.129)在轉置隱藏下，藏入資料 68 而得到新座標 A'(20.786+0.00068,30.129-0.00068) = (20.78668, 30.12832)，若同時放大 16 倍得座標 A(332.576, 482.064)和 A'(332.58688, 482.05312)，在瀏覽器上 A 和 A'呈現位置都和座標(332.6,482.1)相同，原始 SVG 程式片段如下：

<path d="M-40-30l 80 0l 0 60l-80 0l 0-60Z" style="fill:rgb(255,255,255);
stroke:rgb(0,0,255);" transform="translate(20.786,30.129)" > </path>

隱藏資料的過程依照 DOM Tree 逐項遍歷 element → attribute → methods → 改變 Value，其中 element 可以是 path、use、text 或 g 等，attribute 則是 transform，methods 可以是 scale、translate、opacity 等，attribute value 可以兩位座標或單獨一個數值，隱藏資訊後程式片段如下：

<path d="M-40-30l 80 0l 0 60l-80 0l 0-60Z" style="fill:rgb(255,255,255);
stroke:rgb(0,0,255);" transform="translate(20.78668, 30.12832)" > </path>

反解時 DOM Tree 抓出 20.78668 和 30.12832，依照特徵值取出 68(以 32 比對驗證)，還原成對照的可見字元，每一個元素的 Attribute 值可供轉置隱藏一個字元，第二個以後的字元隱藏在下一個元素，直到所有資訊隱藏完畢。

3.1.2 位移隱藏（Hide Move）

SVG 的圖檔移動路徑，由 A 點直線移到到 C 點，可以在直線路徑 A、C 之直線上增加 B 點，那麼由 A 點經 B 點到 C 點，或由 A 點超越 C 點先到 B 點再回頭到 C 點，在瀏覽器畫面上產生的圖形路徑不變，都只看到結果是由 A 點移動到 B 點。

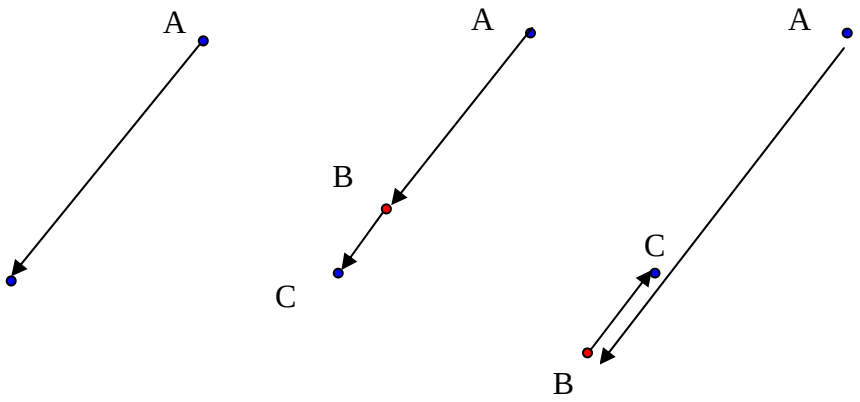

圖 3 - 1 位移隱藏示意圖

如果一個 SVG 圖檔是由很多路徑組合而成，可經由亂數取足夠位置，在眾多路徑中各自做位移隱藏，原始 SVG 程式片段如下：

<path style="fill:none; stroke:blue;" ="M512.5234375 428.349365 L525. 46875 406.698730" >

原始路徑要移動到絕對座標C(512.5234375,428.349365)然後畫出一條直線到D(525. 46875 406.698730)，現在藏入藏入資料67資料而得到新座標B(579.234375 428.349365)，由B點到C點則是向量(-67,0)，因為向量AC＝向量AB+向量BC，後程式片段如下：

<path style="fill:none; stroke:blue;" d="M579.5234375 428.349365 m-67 0 L525. 46875 406.698730">

3.1.3 動畫隱藏 (Hide Montion)

動畫時間以秒計時，增減一點點延遲時間是察覺不出來的，起始、結束的時間值同時加上延遲時間，則動畫經歷時間長度相同，或是增加或減少動畫經歷時間長度，只要保留起始、結束其中一個時間值。將隱藏字元的編碼放在小數點下第三位和第四位，些微差距不影響原始數值資料在動畫上的表現，原始SVG程式片段如下：

<animateMotion calcMode="paced" dur="8.0S" repeatCount="indefinite" begin="6.0S" rotate="auto">

動畫起始時間是在第6秒，歷經8秒動畫後於14秒時停止，藏入數值資料67後，調整起始時間是在第6.0067秒，歷經7.9933秒動畫後，依然於14秒時停止，隱藏資訊後程式片段如下：

<animateMotion calcMode="paced" dur="7.9933S" repeatCount="indefinite" begin="6.0067S" rotate="auto">

3.1.4 顏色隱藏 (Hide Color)

RGB 三顏色數值為整數，對於小數點部分採取無條件捨去，將字元編碼放在小數點下第二位以後，繪圖填色時會自動忽略，不影響原始數值資料在顏色上的表現，原始 SVG 程式片段如下：

<line x1="20" y1="40" x2="220" y2="40" style="stroke:rgb(255 , 127 , 63); stroke-width:4"/>

一次藏入 3 個數值資料 76、68、9 後，隱藏資訊後程式片段如下：

<line x1="20" y1="40" x2="220" y2="40" style="stroke:rgb(255.076 , 127.068 , 63.009); stroke-width:4"/>

3.1.5 外形隱藏 (Hide Shape)

SVG 圖形的外形物件元素都有固定的屬性，其屬性值一般是配對座標如中心 x 和 y、長和寬、橢圓長軸半徑和短軸半徑、圓角矩形 Roundness-x 和 Roundness-y 等，依照 DOM Tree 逐項遊走遍歷 element → attribute → 改變 Value，其中 element 可以是 Line、Rect、Circle、Ellipse 或 Text 等，attribute 則是固定配對格式如(x1，y1)、(rx，ry)、(cx，cy)、(width，height)、(x，y)等，原始 SVG 程式片段如下：

<rect x="50" y="100" width="150" height="20" style="stroke:#FF0000; fill:#CCCCCC"/>

每次藏入 1 個數值資料，本例將字元編碼 76 藏於小數點第二、三位數，不影響原始數值資料在 SVG 圖形外形上的表現，隱藏資訊後程式片段如下：

<rect x="50.076" y="100.076" width="150"

height="20" style="stroke:#FF0000; fill:#CCCCCC"/>

3.1.6 矩陣隱藏 (Hide Matrix)

元素 matrix 在 SVG 圖形中包含剛性運動和非剛性運動，包含平移、旋轉和鏡射三種基本模式，非剛性運動指的是伸縮。

SVG 中所有的矩陣轉換都可以用一個 3×3 矩陣 $\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix}$ 來表達，矩陣中真正有用的數據只有六個，所以在 SVG 中用 matrix[a b c d e f]來表示，而新舊座標之間的關係是 $\begin{bmatrix} x_{\text{舊座標}} \\ y_{\text{舊座標}} \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x_{\text{新座標}} \\ y_{\text{新座標}} \\ 1 \end{bmatrix}$，舉例來說：

- $\begin{bmatrix} 1 & 0 & m \\ 0 & 1 & n \\ 0 & 0 & 1 \end{bmatrix}$ 表 matrix [1 0 0 1 m n]或 tanslate(m,n)，將座標(x,y)平移到(x+m,y+n)

- $\begin{bmatrix} c & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$ 表 matrix [c 0 0 d 0 0]或 scale(c,d)，將 x 座標伸縮 c 倍，y 座標伸縮 d 倍, 座標(x , y)變成(cx , dy)

- $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ 表 matrix [$\cos\theta$ $\sin\theta$ $-\sin\theta$ $\cos\theta$ 0 0]或 rotate(θ)，將 x 軸旋轉θ度，座標(x , y)變成[$\cos\theta - \sin\theta$, $\sin\theta + \cos\theta$]。

- $\begin{bmatrix} 1 & \tan\theta & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ 表 matrix [1 0 $\tan\theta$ 1 0 0]，或 skewX(θ)，沿 x 軸水平頃斜θ度，座標(x , y)變成($x + y\tan\theta$, y)

- $\begin{bmatrix} 1 & 0 & 0 \\ \tan\theta & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ 表 matrix [1 $\tan\theta$ 0 1 0 0]，或 skewY(θ)，沿 y 軸垂直頃斜θ度，座標(x , y)變成(x , $x\tan\theta + y$)

兩個 3×3 矩陣經過運算合併後，依然得到一個 3×3 矩陣，新舊座標之間的關係是 $\begin{bmatrix} x_{舊座標} \\ y_{舊座標} \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & c_1 & e_1 \\ b_1 & d_1 & f_1 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} a_2 & c_2 & e_2 \\ b_2 & d_2 & f_2 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x_{新座標} \\ y_{新座標} \\ 1 \end{bmatrix}$，如果多個巢狀矩陣經過運算合併最後的結果還是一個矩陣，稱為 CTM(the current transformation matrix)

$$\text{CTM}= \begin{bmatrix} a_1 & c_1 & e_1 \\ b_1 & d_1 & f_1 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} a_2 & c_2 & e_2 \\ b_2 & d_2 & f_2 \\ 0 & 0 & 1 \end{bmatrix} \bullet \ldots\ldots \bullet \begin{bmatrix} a_n & c_n & e_n \\ b_n & d_n & f_n \\ 0 & 0 & 1 \end{bmatrix}$$

$\begin{bmatrix} x_{舊座標} \\ y_{舊座標} \\ 1 \end{bmatrix}$ =CTM $\bullet \begin{bmatrix} x_{新座標} \\ y_{新座標} \\ 1 \end{bmatrix}$，例如經過三個轉置運算，先平移 translate(50,90)，接著旋轉 –45°，最後平移 translate(130,160)，運算如下：

CTM＝translate(50,90),rotate(-45),translate(130,160)

$$= \begin{bmatrix} 1 & 0 & 50 \\ 0 & 1 & 90 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 0.707 & 0.707 & 0 \\ -0.707 & 0.707 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & 130 \\ 0 & 1 & 160 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.707 & 0.707 & 255.03 \\ -0.707 & 0.707 & 111.21 \\ 0 & 0 & 1 \end{bmatrix}$$

本系統利用 matrix 來隱藏資訊，先假設 SVG 文件原來的 matrix 中有六個數值分別是 a,b,c,d,e,f，記成 M（a,b,c,d,e,f），而要藏的資料假設 6 個字元為 m,n,o,p,q,r，放在 matrixA（m,n,o,p,q,r）之中

令 matrix B＝(am+cn,bm+dn,ao+cp,bo+dp,aq+cf+e,bq+dr+f)

令 matrix C=$\left(\frac{\text{p}}{\text{mp-no}}, \frac{\text{-n}}{\text{mp-no}}, \frac{\text{-o}}{\text{mp-no}}, \frac{\text{m}}{\text{mp-no}}, \frac{\text{or-pq}}{\text{mp-no}}, \frac{\text{-mr+nq}}{\text{mp-no}}\right)$

則 $\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} am+cn & ao+cp & aq+cf+e \\ bm+dn & bo+dp & bq+dr+f \\ 0 & 0 & 1 \end{bmatrix} \bullet \frac{1}{mp-no} \begin{bmatrix} p & -o & or-pq \\ -n & m & -mr+nq \\ 0 & 0 & mp-no \end{bmatrix}$

即 matrix A=matrix(B • C)，可以用兩個矩陣 B 和 C 來替代原來矩陣 A，這樣得到相同結果，原始 SVG 程式片段如下：

```
<rect transform="matrix(1,1,0,1,50,100)" x="50" y="80" width="150"
height="75"    style="fill: #FFC; stroke: #F00; stroke-width: 1.5px"/>
```

原本矩陣內資料（a,b,c,d,e,f）＝（1,1,0,1,50,100），假設要藏入的資訊是（m,n,o,p,q,r）＝（1,2,3,4,5,6），則 matrix B＝（1,3,3,7,55,111），matrix C＝（-2,1,1.5,-0.5,1,-2），隱藏資訊後程式片段如下：

```
<rect transform="matrix(1,3,3,7,55,111) matrix(-2,1,1.5,-0.5,1,-2)" x="50"
y="80" width="150" height="75"    style="fill: #FFC; stroke: #F00;
stroke-width: 1.5px"/>
```

因為 $\begin{bmatrix}1 & 0 & 50\\ 1 & 1 & 100\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 3 & 55\\ 3 & 7 & 111\\ 0 & 0 & 1\end{bmatrix}\bullet\begin{bmatrix}-2 & 1.5 & 1\\ 1 & -0.5 & -2\\ 0 & 0 & 1\end{bmatrix}$ 代表 matrix(1,1,0,1,50,100)

運算等於 matrix(1,3,3,7,55,111) matrix(-2,1,1.5,-0.5,1,-2)，兩個矩陣合併運和前面單一矩陣有一樣的結果，所以原 SVG 圖形在瀏覽器上顯示結果相同，完全不會改變。

還原方法從第二個矩陣 matrix(-2,1,1.5,-0.5,1,-2)求逆矩陣，假設

（a,b,c,d,e,f）的逆矩陣是（m,n,o,p,q,r）

則 $\begin{bmatrix}a & c & e\\ b & d & f\\ 0 & 0 & 1\end{bmatrix}^{-1}=\frac{1}{ad-bc}\begin{bmatrix}d & -c & cf-de\\ -b & a & -af+be\\ 0 & 0 & ad-bc\end{bmatrix}=\begin{bmatrix}m & o & q\\ n & p & r\\ 0 & 0 & 1\end{bmatrix}$，

$m=\frac{d}{ad-bc}$、$n=\frac{-b}{ad-bc}$、$o=\frac{-c}{ad-bc}$、$p=\frac{a}{ad-bc}$、$q=\frac{cf-de}{ad-bc}$、$r=\frac{-af+be}{ad-bc}$

可還原出隱藏的資料為： m=1、n=2、o=3、p=4、q=5、r=6

3.1.7 隱藏雙位元文字

世界性的電腦字元編碼標準 ISO646（7-bit coded character set for information interchange）和 ASCII（American Standard Codefor Information Interchange），雖然是電腦與網路世界裡的標準，因編碼空間太小，不足以因應各種應用程式的需求。不同的國家各自使用不同的編碼方式，亞洲國家語如台灣、中國、日本、南北韓、越南、新加坡和港澳地區所使用的漢字中文字、排版系統的標誌符號、非英語拼音字母和圖形符號等的編碼，需要使用 2 或多個位元組來編碼。跨國傳輸交換檔案時，因為編碼方式不同而容易造成錯誤和遺失。

Unicode(Universal Multiple Octet Coded Character Set 中文翻譯成統一碼或標準萬國碼）提供了解決的方法，不論是什麼平臺，不論是什麼程式，不論什麼語言。Unicode 標準已經被這些工業界的領導們所採用，例如：Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Sybase, Unisys 和其他許多公司。最新的標準都需要 Unicode，例如 XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML 等等，許多作業系統、最新的瀏覽器和其他產品都支援 Unicode，它是近來全球軟體技術最重要的發展趨勢[3]。

Unicode 跟 Big-5 不同的地方，在於字表容納的字數。由於 Unicode 的字數遠遠大於 Big-5，所以除了 CJK（中日韓）文字外，也把世界上大部分可以被書寫的各種符號列入了。所以要在 SVG 圖檔中秀出亞洲雙語系文字，就要靠 Unicode 的編碼方式，使用 JBuilder 編輯器撰寫 JAVA 程式當然支援 Unicode，也能準確判斷雙位元字母長度，在讀取文字時必須採用『字元（char）陣列』方式讀取，從瀏覽器畫面輸入雙位元文字並把 Unicode 顯示出來的方法在下列程式碼中：

```
char    ctemp []= this.j8Text_chin.getText().toCharArray();
int len=ctemp.length;
int temp[] = new int [len];
code = new int[len];
String TempString ="";
for (int i=0;i<temp.length;i++)
{
```

```
            code[i] = (int) ctemp[i];
            TempString=TempString+" "+code[i];
        }
        this.j8Text_unicode.setText(TempString.trim());
```

此段程式在 j8Text_chin 的 Text 欄位中輸入『師大資訊教育』以 CharArray 方式，依字串長度逐字讀入到 ctemp[]暫存，再以整數 int 方式將文字內碼換到變數 TempString，此時 TempString 內容已經是 24107 22823 36039 35338 25945 32946，化成十六進位的 Unicode 是 5E2B 5927 8CC7 8A0A 6559 80B2。

使用 matrix 來隱藏雙位元文字，藏入『師大資訊教育ぁそべえぢま生き残り戦略』共 18 個字，十六進位 Unicode 是『5E2B 5927 8CC7 8A0A 6559 80B2 3041 305D 3079 3048 3062 307E 751F 304D 6B8B 308A 6226 7565』，化成十進位『24107 22823 36039 35338 25945 32946 12353 12381 12409 12360 12386 12414 29983 12365 27531 12426 25126 30053』，將 18 個代碼分裝在 3 個矩陣中：

A＝（24107,22823,36039,35338,25945,32946）或 $A=\begin{bmatrix}24107 & 36039 & 25945\\ 22823 & 35338 & 32946\\ 0 & 0 & 1\end{bmatrix}$

B＝（12353,12381,12409,12360,12386,12414）或 $B=\begin{bmatrix}12353 & 12409 & 12386\\ 12381 & 12360 & 12414\\ 0 & 0 & 1\end{bmatrix}$

C＝（29983,12365,27531,12426,25126,30053）或 $C=\begin{bmatrix}29983 & 27531 & 25126\\ 12365 & 12426 & 30053\\ 0 & 0 & 1\end{bmatrix}$

假設 SVG 程式中原始矩陣為 M=(1,2,3,4,5,6)，則

MA＝（92576,139506,142053,213430,124788,183680）

$A^{-1}B$＝（-0.32914,0.562935,-0.23601,0.502192,8.878441,-6.31515）

$B^{-1}C$＝（-227.922,229.3098,-195.328,196.6551,64.46184,-63.1443）

C^{-1}＝

（.000386526,-.00038463,-.000856384,.000932657,16.0250834,-18.3649731）

亦即 $MA=\begin{bmatrix} 92576 & 142053 & 124788 \\ 139506 & 213430 & 183680 \\ 0 & 0 & 1 \end{bmatrix}$、

$A^{-1}B=\begin{bmatrix} -0.32914 & -0.23601 & 8.878441 \\ 0.562935 & 0.502192 & -6.31515 \\ 0 & 0 & 1 \end{bmatrix}$、

$B^{-1}C=\begin{bmatrix} -227.922 & -195.328 & 64.46184 \\ 229.3098 & 196.6551 & -63.1443 \\ 0 & 0 & 1 \end{bmatrix}$、

$C^{-1}=\begin{bmatrix} 0.000386526 & -0.000856384 & 16.0250834 \\ -0.00038463 & 0.000932657 & -18.3649731 \\ 0 & 0 & 1 \end{bmatrix}$

原始 SVG 程式碼如下：

```
<rect transform="matrix(1,2,3,4,5,6)" x="50" y="80" width="150"
height="75"    style="fill: #FFC; stroke: #F00; stroke-width: 1.5px"/>
```

隱藏資訊後程式碼如下：

```
<rect transform="matrix(92576,139506,142053,213430,124788,183680)
matrix(-0.32914,0.562935,-0.23601,0.502192,8.878441,-6.31515)
matrix(-227.922,229.3098,-195.328,196.6551,64.46184,-63.1443)
```

matrix(.000386526,-.00038463,-.000856384,.000932657,16.0250834,-18.3649731)" x="50" y="80" width="150" height="75" style="fill: #FFC; stroke: #F00; stroke-width: 1.5px"/>

一個 SVG 文件中，令 M 為原始矩陣，M＝$\begin{bmatrix} a_0 & c_0 & e_0 \\ b_0 & d_0 & f_0 \\ 0 & 0 & 1 \end{bmatrix}$，在此文件中要插入雙位元文字 $6n$ 個，分成 n 組每組藏 6 個編碼後的文字，$[a_i, b_i, c_i, d_i, e_i, f_i]$ ，亦即 $A_i = \begin{bmatrix} a_i & c_i & e_i \\ b_i & d_i & f_i \\ 0 & 0 & 1 \end{bmatrix}, i \in [1..n]$ ，則原始矩陣 M 改成 $MA_1 \bullet CTM \bullet A_n^{-1}$ ，其中

$$MA_1 = \begin{bmatrix} a_0a_1 + c_0b_1 & a_0c_1 + c_0d_1 & a_0q + c_0f_1 + e_0 \\ b_0a_1 + d_0b_1 & b_0c_1 + d_0d_1 & b_0q + d_0f_1 + f_0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$CTM = \prod_{i=1}^{n-1} \frac{1}{a_id_i - b_ic_i} \begin{bmatrix} d_i & -c_i & c_if_i - d_ie_i \\ -b_i & a_i & -a_if_i + b_ie_i \\ 0 & 0 & a_id_i - b_ic_i \end{bmatrix} \bullet \begin{bmatrix} a_{i+1} & c_{i+1} & e_{i+1} \\ b_{i+1} & d_{i+1} & f_{i+1} \\ 0 & 0 & 1 \end{bmatrix},$$

$$A_n^{-1} = \frac{1}{a_nd_n - b_nc_n} \begin{bmatrix} d_n & -c_n & c_nf_n - d_ne_n \\ -b_n & a_n & -a_nf_n + b_ne_n \\ 0 & 0 & a_nd_n - b_nc_n \end{bmatrix}$$

還原隱藏資料 A_k 的演算法如下：

$Temp \leftarrow A_n^{-1}$

If k = n

then Return $Temp^{-1}$

$i \leftarrow$ n

while $i \neq$ k

do

$i \leftarrow i-1$

$Temp \leftarrow (A_i^{-1} \bullet A_{i+1}) \bullet Temp$

If $i = k$

then Return $Temp^{-1}$

3.2 資訊附加法

標記（Markup）是在稿件或文章上加上的記號，以記錄各種不同的資訊，主要可分為兩大類，一類是關於「排版或顯示格式」的標記，另一類則是關於「資料結構或內容」的標記。XML 最大的一個優點就是它可以自訂標籤（Tag），在 SVG 圖檔中，合乎 SVG 規範的標籤會影響輸出的圖形，不合 SVG 規範但合 XML 規範的標籤，會被 parser 過濾掉而不顯示在螢幕上，對於原本的 SVG 圖形毫無影響，所以這是一個藏資訊的好地方。

3.2.1 增加標籤 (ADD Tag)

一個簡單的原始 SVG 程式如下：

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
<rect style="fill:rgb(255,255, 0); stroke:rgb(0,0,255);" stroke-width="5"
x="20" y="20" width="200" height="130" />
</svg>
```

圖形只有顯示出一個長 200 寬 130 的矩形，如果在標籤</svg>之前插入一組自訂標籤<xpplab></xpplab>，只要是符合有良好格式（Well-formed），在這組標

籤之間便可以填入適當的 SVG 元素當作隱藏資料，下列 SVG 程式藏入『NTNU』四個字元，以 20%透明度、旋轉 45 度出現在矩形左上角，但只有在移除此自訂標籤<xpplab></xpplab>後，才會顯示隱藏的資料。

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
<rect style="fill:rgb(255,255,0); stroke:rgb(0,0,255);" stroke-width="5"
x="20" y="20" width="200" height="130" />
<xpplab><text font-size="36" transform="rotate(30,-80,80)"
style="fill-opacity:0.2">NTNU</text></xpplab>
</svg>
```

下圖 3-2 左方為去除標籤<xpplab></xpplab>後，才會顯示隱藏資訊『NTNU』，而右方為沒有去除標籤的情形下，不會顯示隱藏的資訊。

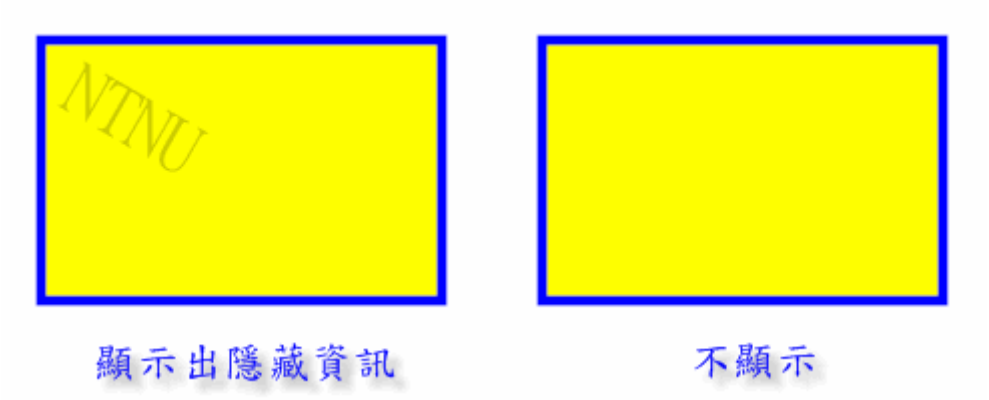

圖 3 - 2 隱藏資訊顯示與不顯示

3.2.2 附加動畫 (Add Animate)

利用增加標籤的方法將合法的 SVG 元素加入原始檔案中，在</svg>之前加入下面的虛擬程式碼：

< element attribute-value >NTNU</ element >

<shape attribute-value >

<animate attributeName="vector" begin=" begin-time "

from="start-position" to=" end-position " dur=" during-time "

repeatCount="times" />

</ shape >

其中『NTNU』是附加的主要資訊，將在螢幕左上角產生一個方形色塊，在文字上無限次左右巡迴移動，但是將標籤<text>偽裝後，所有所有動畫、文字均被隱藏，與未隱藏資訊前的圖檔顯示結果相同，直到解除偽裝後才顯示出來。

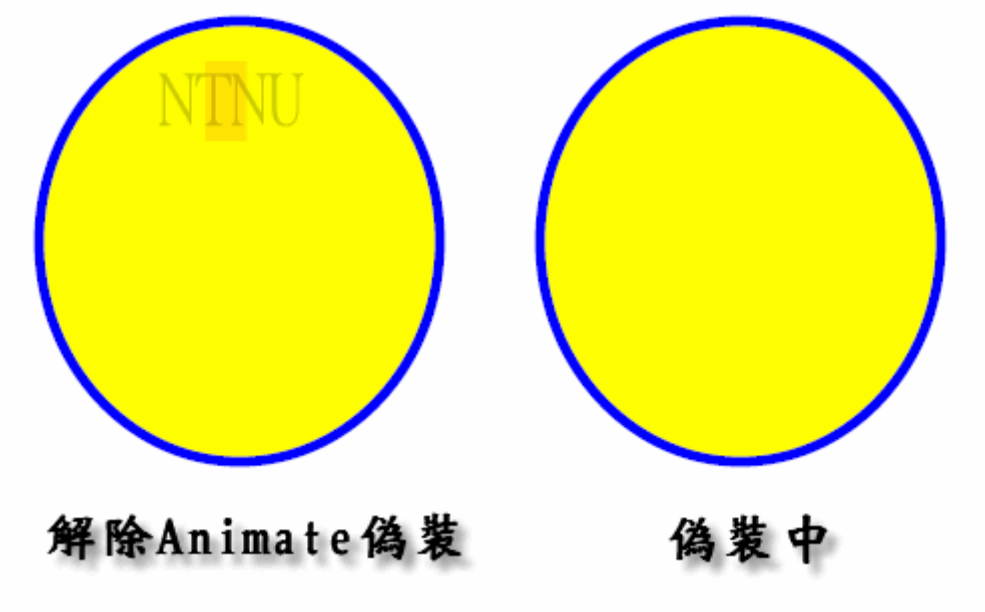

圖 3 - 3 Add Animate 比對圖

動畫元素 Animate 會依附在上一層元素標籤<element></ememt>之內，更改此<element>標籤即可達到停止動畫，偽裝的方法通常以竄改合法 SVG 元素，使 SVG Viewer 無法正常解讀，而達到隱藏效果。常見下列幾種方式，竄改大小寫任意字母如<Element>，刪減字母如<elemnt>、增加字母如<elements>，變更字

母順序如<elenemt>，增加額外前置 Tag 如<xpplab-element>，……等。下面 SVG 程式是經過偽裝過的：

```
<?xml version="1.0" ?>
<svg width="100%" height="100%">
<ellipse style="fill:rgb(255,255,0);stroke:rgb(0,0,255);" stroke-width="5"
    cx="120" cy="120" rx="100" ry="110" />
<ellipse style="fill:rgb(255,255,0);stroke:rgb(0,0,255);" stroke-width="5"
    cx="370" cy="120" rx="100" ry="110" />
<Text dx="80" dy="60" font-size="36" style="fill-opacity:0.2">NTNU</text>
<rcet x="20x" y="30px" width="20px" height="40px" style="fill-opacity:0.2;
        fill:#ff7711; ">
        <animate attributeName="x" begin="0s" from="65px" to="150px"
dur="2s" repeatCount="indefinite" />
</rcet>
</svg>
```

3.2.3 附加遮罩 (Add Mask)

SVG 圖形的遮罩可以借助內部或外部 CSS 語法格式，在圖檔中增加了 CSS 語法不影響圖形的顯示，直到 CSS 被引用才會改變圖形。偽裝 musk 隱藏步驟主要分成三步：

1. 在<defs>之後增加 CSS 宣告：

```
<style type="text/css">
<![CDATA[
shape{stroke: color;fill:color;}
```

```
element.masked{mask:url(#myMask1);fill: color;font-size:size;}
]]>
</style>
```

2. 在</defs>之前增加動畫遮罩 id 及屬性宣告：

```
<mask id="myMask1" maskUnits="userSpaceOnUse">
< musk-shape attribute-value>
<animate attributeName="vector" from="start-position" to="
end-position " begin="begin-time" dur=" during-time "
repeatCount="times"/>
</ musk-shape >
</mask>
```

3. 在</svg>前插入遮罩目的元素：

```
< musk-element attribute-value class="masked">NTNU</
musk-element >
```

偽裝 mask 時只針對第三步驟，只有第三步驟真正執行步驟一、二，偽裝的方法可使用 3.3.1 新增標籤的方法或 3.3.2 竄改合法 SVG 元素，使 SVG Viewer 無法正常解讀，而達到隱藏效果。常見下列幾種方式，竄改大小寫任意字母如<Element>，刪減字母如<elemnt>、增加字母如<elements>，變更字母順序如<elenemt>，增加額外前置 Tag 如<xpplab-element>等。

下圖是一個以『NTNU』為主要附加資訊，而藏在遮罩之後，當揭開隱藏 musk 時，在呈現交織的 31 個基本圖形上，左上角出現圓形探照燈形狀，左右來回掃瞄著『NTNU』四個字。

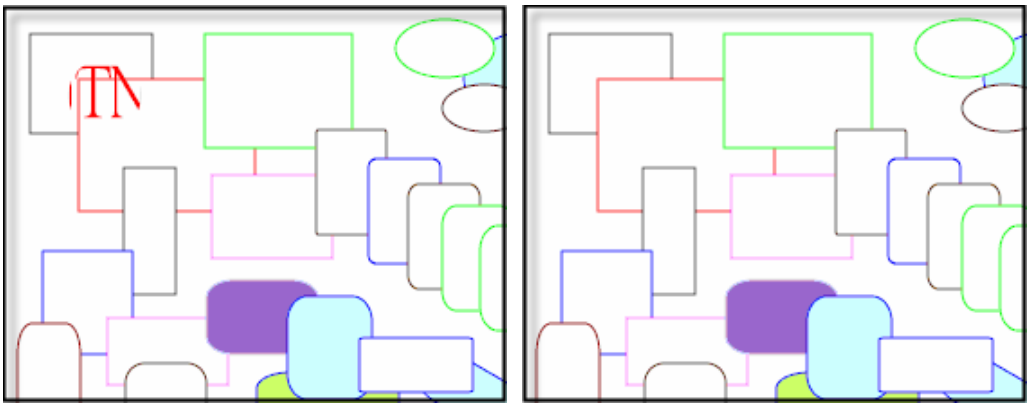

圖 3 - 4 Add musk 比對圖

3.2.4 附加曲線 (Add Curve)

在 SVG 圖形中藏入圖形，並不是藏入點陣圖，其實還是藏入文字格式，而這個文字經過 SVG Viewer 解釋後，卻是一個真正的圖形。以元素 path 指定路徑和屬性值後繪製成 curve，還可以增加文字繞曲線，隱藏 Curve 的步驟主要分成 2 步：

1. 在<defs>之後增加 Curve 的指定路徑：

```
<path id="myCurve" style=" value; stroke-width: value ; stroke:color;
   fill:color;opacity:value" d="curve-path"/>
```

2. 在</svg>前插入隨 Curve 隱藏的 TextPath：

```
<element xlink:href="#myCurve" />
<text font-size=" size " style=" style-value">
 <textPath xlink:href="#myCurve" startOffset="value" >
   hided Text with Curve here
 <textPath />
</text>
```

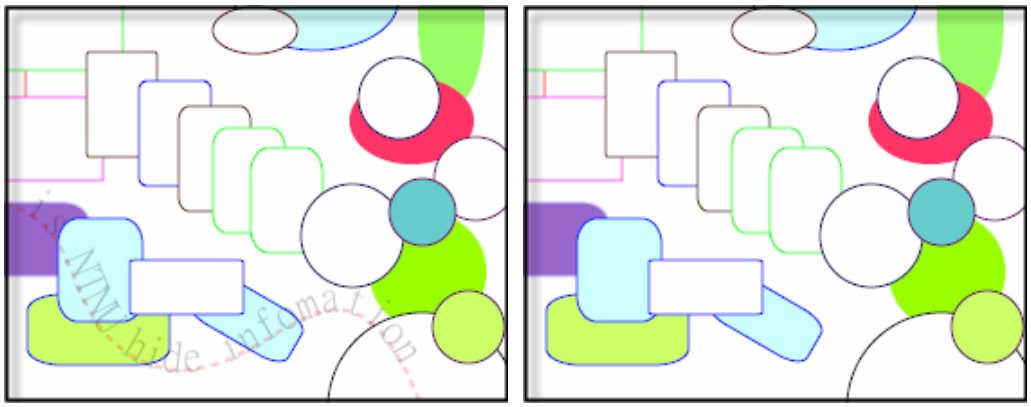

圖 3 - 5 Add curve 比對圖

3.3 資料隱藏之比較

一個 SVG 檔案能夠藏多少資訊，要看這個 SVG 文件的內容而定，就 Information Mixture 方法能隱藏資訊的數量而言，可分為無限字元隱藏和有限字元隱藏：

表 3 - 3 Information Mixture 隱藏字元數量比較表

方法 數量	矩陣 隱藏	轉置 隱藏	位移 隱藏	動畫 隱藏	顏色 隱藏	外形 隱藏
有限		V		V	V	V
無限	V		V			

並不是每一個 SVG 檔案都能作矩陣隱藏或位移隱藏，必須依據文件內部是否有 matrix 和 move 元素而定，至少要有一個，那麼資訊便可無窮盡藏入，甚至超越本身檔案大小的資料都可以。本研究以每次增加 6 個隱藏字元，分別比較矩陣隱藏和位移隱藏檔案大小與隱藏資料量的關係，如下圖 3-6、3-7：

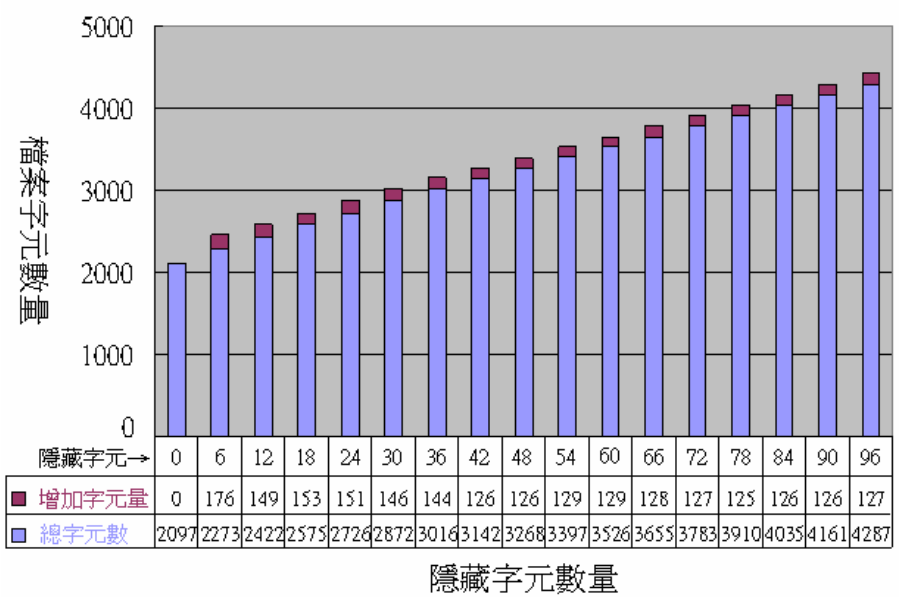

圖 3 - 6 Hide Matrix 隱藏字元增加數量與檔案大小

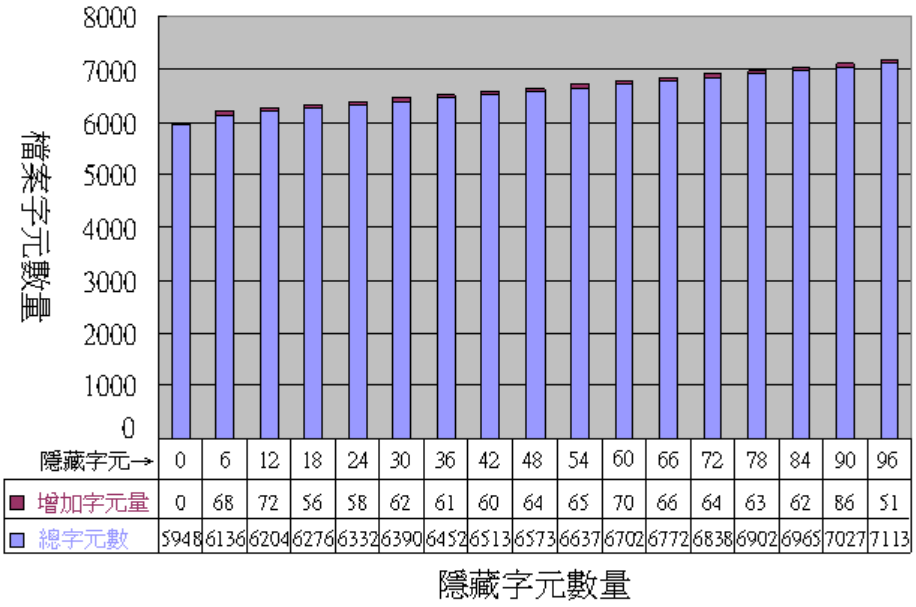

圖 3 - 7 Hide Move 隱藏字元增加數量與檔案大小

計算檔案大小不以所佔磁碟空間為依據，SVG 文件中空格如同 HTML 不會影響檔案執行結果，但會使檔案變大，因此不計檔案內空白字元、空白行距，純

粹計算文件內有效字元。為了方便計算每隱藏一個字元，文件中需增加多少掩護的字元，本研究提出一個隱藏字元成本（Cost of Hiding Character）公式：

$$CHC = \frac{C_{\text{after}} - C_{\text{before}}}{H_{\text{number}}}$$

其中 C_{after} 表示文件隱藏資訊後字的元數，C_{before} 表示文件隱藏資訊前的字元數，H_{number} 表示隱藏的字元數。例如隱藏 NTNU 四個字元到一個原有 4928 字元的 SVG 檔案內，藏入後檔案共有 5002 個字元，即多出 74 的字元來隱藏此『NTNU』，則隱藏字元成本 $\text{CHC} = \frac{5002 - 4928}{4} = 18.5$。

不同的檔案，適用不同的方法，轉置隱藏、動畫隱藏、顏色隱藏、外形隱藏，只能藏入有限的資訊，其隱藏的容納量與該檔案相關的 element 數量成正比，而位移隱藏和矩陣隱藏可以融入非常大量的資訊。矩陣隱藏的 CHC 值雖然高於位移隱藏，但藏入資訊後的隱蔽狀態也優於位移隱藏，下表為 Information Mixture 的各種隱藏方法在不同檔案下，隱藏相同數目字元的 CHC 值：

表 3 - 4 Information Mixture 隱藏字元 CHC 成本比較表

方法 數量	hide transform	hide Move	hide Montion	hide Color	hide Shape	hide matrix
6	10	11.33	10	5	8	29.33
12	10	11.33	10	5	8	27.08
18		11.56				26.56
24		11.00				26.21
30		10.73				25.83
36		10.67				25.53
42		10.60				24.88
48		10.52				24.40
54		10.54				24.07
60		10.57				23.82
66		10.67				23.61
72		10.69				23.42
78		10.69				23.24
84		10.68				23.07
90		10.66				22.93

雖然 Add Tag、Add Animate、Add Mask、Add Curve 方法可以附加藏入很大量的資訊，但藏得多就容易被發現或被破壞，因為它們隱藏的資訊本身沒有經過編碼處理，隨著藏入資訊量大而降低的 CHC 值，並沒有太大的意義，所以適合製作 Visible Watermark。下表為 Infomation Add-On 的各種隱藏方法在不同檔案下，隱藏相同數目字元的 CHC 值：

表 3 - 5 Infomation Add-On 隱藏字元 CHC 成本比較表

方法 數量	Add Tag	Add Animate	Add Mask	Add Curve
6	16.83	46.33	95.33	81.67
12	8.92	23.67	48.17	41.33
18	6.28	16.11	32.44	27.89
24	4.96	12.33	24.58	21.17
30	4.17	10.07	19.87	17.13
36	3.64	8.56	16.72	14.44
42	3.26	7.48	14.48	12.52
48	2.98	6.67	12.79	11.08
54	2.76	6.04	11.48	9.96
60	2.58	5.53	10.43	9.07
66	2.44	5.12	9.58	8.33
72	2.32	4.78	8.86	7.72
78	2.22	4.49	8.26	7.21
84	2.13	4.24	7.74	6.76
90	2.06	4.02	7.29	6.38