資料加密與解密-應用於網路安全

王鑫聖　翁鈺華　顏淑惠

淡江大學資訊工程系　淡江大學資訊工程系　淡江大學資訊工程系

u7510084@tknet.tku.edu.tw　u7510019@tknet.tku.edu.tw　shyen@cs.tku.edu.tw

摘 要

網路安全著重於資訊不被攔截或竄改，若特意將重要資料或檔案以亂碼形式傳送的話，只怕適得其反。收送雙方使用某種方式，以一般文章、圖片、或是多媒體檔案，在不改變原文或檔案形式下交換機密資料，即能降低被他人取得或毀壞資料的風險。在我們提出的方法中，融合了[3]所提出的方式與[4]所提出的觀念，以表面檔案（稱為 mask file）來傳遞機密檔案（稱為 original file，原來資料）。original file 與 mask file 可為各種檔案格式且不須為同一種格式。在我們的方法中，可以將 single byte 任意分成 3 bits, 4bits, 5 bits...為加密單位。只要收送雙方事先約定好加密單位的格式，就能增加資料的安全性。

關鍵字：混淆文件加密法(Confused Document Encrypting Scheme, CDES)，國際加密標準(IDEA)，資料加密(Information Encrypting)，原始機密檔案指標(Original File Index, OFI)，表面檔案位置表(Mask's Position Table, MPT).

一、前 言

由於電腦與通訊網路的普及，各種資訊得以透過通訊網絡無遠弗屆地交換。在資訊時代，除了享受科技帶來的方便之外，電腦病毒、駭客入侵等破壞系統、危害網路安全的情況日益嚴重，因此更需時時加以防範。

在網路上傳送資料時最擔心的就是資料被非原先指定的接收人取得，一般文件如此，具有機密性的文件、檔案更是如此。因此，加密成了保護文件的必要手段。加密的主要目的在於提高資訊內容的隱私性，資訊在公眾網路傳輸過程中，會經過許多的節點，重要檔案如果沒有經過加密處理，都有可能會經由這些節點被有心人取得資訊內容。如同平常生活中寄送普通信件一樣，在信件傳遞系統中會經過許多人，若沒有利用信封做為文件內容的隱蔽物，那麼這些經手的人都能讀到信件內容。若特意將重要文件資料或檔案以亂碼形式傳送的話，則有此地無銀三百兩之意，會引起非相關人士的覬覦或更蓄意的攔截或破壞。倘若傳送與接收的雙方可用某種方式，利用有意義的文章、各式圖片、或是各種多媒體檔案，在不改變原文或檔案形式下達成傳送或交換機密資料，那就可以大大降低被有心人士取得資訊或毀壞資料的風險。

二、CDES 方法的介紹與限制

在[3]所提出的 CDES(Confused Document Encrypting Scheme)方法，是利用字面上看來有意義的任何文章(cheating text)來取代以往所採用的加密原文(encrypted meaningless message)的方法傳送真正的機密資料(plaintext)給接收者。此所謂的 cheating text 是將資料以 ASCII code 表示的文字檔。

Cheating text 所使用的 ASCII code 至少需

包含 plaintext 中所出現過的所有 ASCII code ，然後利用這些出現在 plaintext 中的 cheating text 的 ASCII code 的位置紀錄整理成字元位置對應表(character's position table，CPT)。利用 CPT 中所紀錄每個 ASCII code 出現的位置來替代 plaintext ASCII code characters 的編碼，如果一個 ASCII code 有一個以上的位置，則以隨機的方式取出任意一個位置來取代，這一個用來紀錄 cheating text ASCII code characters 的檔案稱之為 PIF (plaintext index file)。接著，壓縮 PIF，再使用 IDEA [1,2]來加密 PIF。當接收者收到 cheating text 及 PIF 時，先為 PIF 解密，並利用 cheating text 產生 CPT，當 PIF 解壓縮後，利用 CPT 與 PIF 就可以得到 plaintext。

這個方法會有下列的限制：1. cheating text 中必需包含在 plaintext 中出現的所有字元； 2. CDES 只適用於以 single byte 為基礎的文件，當 plaintext 不是 single byte 的檔案型式時(如：中文檔案)就無法使用。

三、所提出的改良方法

在我們的方法中，並不受限於 single byte 的檔案形式，可以將 single byte 分成 double bytes(加密單位為 4 bits)、4 bytes(加密單位為 2 bits)等等。original file、mask file 也可以任意使用 3 bits, 5 bits..為加密單位。如此一來，只需收送雙方事先約定好加密單位和加密選擇碼的格式，即使有心人破解了 IDEA 得到了 mask file、OFI (original file index)，光要猜出 MPT(mask's position table)就需要花不少時間，更能增加資料的安全性。在我們的方法裡，mask file (表面檔案) 可以不必包括所有出現在 original file (機密檔案) 的字元，也可以拿各種格式(中文、日文、圖形、media files…)來當 original file，並不限於英文。而且 original file 與 mask file 不須為同一種格式的檔案。而且我們同時使用兩把 IDEA 密鑰為資料加密(一把加密 mask ID，一把加密 OFI)，以增加安全性。

(一)、基本觀念與名詞

1. original file(OF) & mask file(MF)：所要傳送的重要文件稱為 original file, 用來作為混淆的文件稱為 mask file。

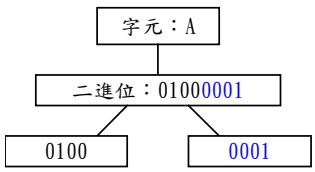
2. 加密單位和加密選擇碼：加密單位將決定字元分成 n 個部份。加密選擇碼是決定以哪幾個 bit 為一組。以字元 A 為例：(一個字元大小為 8bits)

   a).加密單位為 4bits。

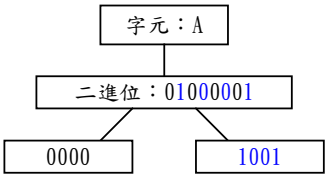
   i).加密選擇碼分別為：第 0~3 bit、第 4~7 bit。字元分成 2 個部份，如圖一(a).

   ii).加密單位為 4bits，加密選擇碼分別為： 第 1,3,5,7 bit、第 0,2,4,6 bit。字元分成 2 個部份，如圖一(b)

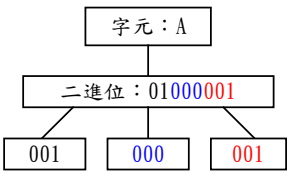
   b).加密單位為 3bits，加密選擇碼分別為： 第 0~2 bit、第 3~5 bit、第 6,7 bit。字元分成 3 個部份，如圖二。



圖一(a)：加密單位為 4bits

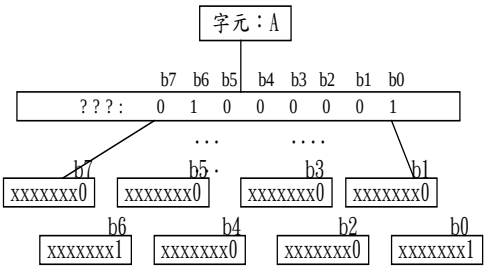


圖一(b)：加密單位為 4bits



圖二：加密單位為 3bits

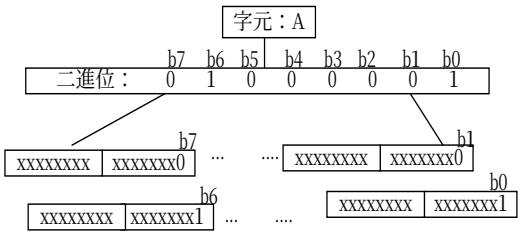
C). 以上加密單位均只對應 1byte，如圖三。進一步分析加密單位與加密選擇碼的變化如表一，此外，更可以讓一加密單位對應 2, 3, 4…. bytes，如圖四、圖五。再加上加密選擇碼的變化，因此，可說有無限多種變化。



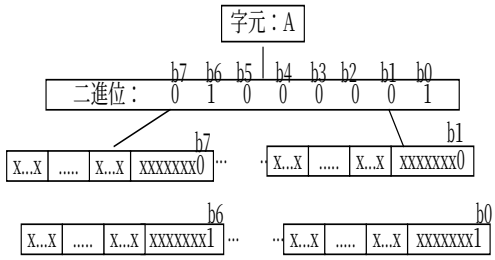
圖三　以一加密單位對應 1 byte

表一

加密選擇碼 / 加密單位	加密選擇碼變化量
1 bit	$[P(8,1)]^8 = 8^8$
2 bits	$[P(8,2)]^4$
3 bits	$[P(8,3)]^2[P(8,2)]$
4 bits	$[P(8,4)]^2$
5 bits	P(8,5)P(8,3)
6 bits	P(8,6)P(8,2)
7 bits	P(8,7)P(8,1)
8 bits	P(8,8)



圖四 以一加密單位對應 2 bytes



圖五 以一加密單位對應 n bytes

3. 數值和子數值：將 original file 中所有出現的字元或原檔案格式轉成所需的加密單位得到的值稱為數值。以圖一(a)來說，字元 A 所得到的數值為 41，子數值為 x4 和 x1。以圖一(b)來說，字元 A 所得到的數值就變成 09，子數值為 x0 和 x9 以圖二來說，字元 A 所得到的數值就成了 101，子數值為 x1,x0 和 x1。

4. 數值和子數值：將 original file 中所有出現的字元或原檔案格式轉成所需的加密單位得到的值稱為數值。以圖一(a)來說，字元 A 所得到的數值為 41，子數值為 x4 和 x1。以圖一(b)來說，字元 A 所得到的數值就變成 09，子數值為 x0 和 x9 以圖二來說，字元 A 所得到的數值就成了 101，子數值為 x1,x0 和 x1。

5. mask's position table (MPT)：紀錄 mask file 中的數值所出現的位置。

6. original file index (OFI)：比對 original file 和 MPT，利用隨機的方式取出同一 index 位置出現在 mask file 的位置做為 original file 的 index，以製成 original file index (OFI)。

7. Mask ID：為了混淆覬覦者應同時傳送多個 mask file(檔案格式可以相同也可以不同)，以隨機的方式來產生 mask ID，用來做為識別 mask file 的編號。

(二)、實作步驟

1. 準備 original file 及數種 mask files。

2. 決定加密單位與加密選擇碼：在我們的實作中以 4bits 為加密單位，加密選擇碼為第 0~第 3bit、第 4~第 7bit。

3. 字元轉成數值：先分析 original file 與 mask file 中出現那些字元，字元轉成數值。

4. 準備 original file 及數種 mask files。

5. 決定加密單位與加密選擇碼：在我們的實作

中以 4bits 為加密單位，加密選擇碼為第 0~第 3bit、第 4~第 7bit。

6. 字元轉成數值：先分析 original file 與 mask file 中出現那些字元，字元轉成數值。數值分割：由於加密單位是 4bits，因此將 original file 所得數值分成兩個部份(left part and right part)，每個部份各自成為一個部份數值各 4bits，並分別置於兩個子數值的 right part。如此 original file 中的數值從 single byte 變成 double bytes，而這兩個 bytes 的 left part 所要填入的數值則採隨機方式產生。

7. 製作 MPT：將 mask file 的數值也分成兩部份，但不分割成兩個子數值。然後紀錄 mask file 中的數值(以 right part 為 index)所出現的位置來建立 mask's position table (MPT)。

8. 製作 OFI，壓縮並且加密：從 MPT 利用隨機的方式取出 mask file 的同一 index 的任一位置做為 original file 的 index，以製成 original file index (OFI)。並將所選定的位置的 left part 數值填入 original file 的 left part 中，壓縮 OFI 後，用 IDEA key1 加密 OFI。

9. 加密 mask ID：使用 IDEA key2 為 mask ID 加密。

10. 放置 mask ID：將加密後的 mask ID 置於壓縮且加密完成的 OFI 之前，接收者將利用此 ID 來做為辨識所收到的 OFI 是屬於那一份 mask file 所產生。

11. 傳送：傳送方傳送 mask file、OFI。收送方式是以網路程式來模擬在網路上實際傳送接收的雙方，使得能在同一電腦中完成傳送與接收的動作模擬。

12. 接收：接收方在收到資料時，先用 IDEA key2 為位於 OFI 前端的 mask ID 解密，取出 mask ID。

13. 找出 MPT 並解密：找到與 mask ID 相對應的 mask file，由 mask file 產生相對應的 MPT，再以用 IDEA key1 為 OFI 解密。。

14. original file。根據 OFI 的值(index)，到 MPT 去找到對應的值，由於加密單位是 4bits，需用兩個 index 才能找到一個 original file 的數值。最後依加密單位還原數值成字元或原檔案格式。

(三)、實驗結果

我們舉出三種實驗的結果。第一部份先突破 CDES 限制的部份，mask file 中可以不必包括所有出現在 original file 的字元。第二部份拿非 single byte 的格式(以中文為例)來當 original file。第三部份實作 original file 與 mask file 為不同格式的檔案(以圖形和音樂檔為例)。限於篇幅，於此只列出第一部分，完整結果請見：http://pria.cs.tku.edu.tw/

實驗： mask file 中不必包括所有出現在 original file 的字元(以 4bits 為加密單位)。

- **original file：**

This page has nothing to do with C programming. The rest of this document does, read on and good luck.

- **mask file：**

This documentary provides an exceptional educational opportunity to enlighten and inform the general public about this little-known tradition of pottery. The virtual field trip offers educators a tool to open the eyes of students to the mystery, wonder and spontaneity of this art form and its importance in Japanese culture.

步驟分析：

1. 將字元轉成數值：以 4bits 為加密單位。

original file：

字元	**T**	**h**	**i**	**s**		**p**	**a**	**g**	**e**		**h**	**a**	**s**
數值	54	68	69	73	20	70	61	67	65	20	68	61	73
字元		**n**	**o**	**t**	**h**	**i**	**n**	**g**		**t**	**o**		**d**
數值	20	6E	6F	74	68	69	6E	67	20	74	6F	20	64

字元	o		w	i	t	h		C		p	r	o	g
數值	6F	20	77	69	74	68	20	43	20	70	72	6F	67
字元	r	o	g	r	a	m	m	i	n	g	.		T
數值	72	6F	67	72	61	6D	6D	69	6E	67	2E	20	54
字元	h	e		r	e	s	t		o	f		t	h
數值	68	65	20	72	65	73	74	20	6F	66	20	74	68
字元	i	s		d	o	c	u	m	e	n	t		d
數值	69	73	20	64	67	63	75	6D	65	6E	74	20	64
字元	o	e	s	,		r	e	a	d		o	n	
數值	6F	65	73	2C	20	72	65	61	64	20	6F	6E	20
字元	a	n	d		g	o	o	d		l	u	c	k
數值	61	6E	64	20	67	6F	6F	64	20	6C	75	63	6B
字元	.												
數值	2E												

mask file：限於篇幅的關係，只有第一段話顯現於此。

字元	T	h	i	s		d	o	c	u	m	e	n	t
數值	54	68	69	73	20	64	6F	63	75	6D	65	6E	74
字元	a	r	y		p	r	o	v	i	d	e	s	
數值	61	72	79	20	70	72	6F	76	69	64	65	73	20
字元	a	n		e	x	c	e	p	t	i	o	n	a
數值	61	6E	20	65	78	63	65	70	74	69	6F	6E	61
字元	l		e	d	u	c	a	t	i	o	n	a	l
數值	6C	20	65	64	75	63	61	74	69	6F	6E	61	6C
字元		o	p	p	o	r	t	u	n	i	t	y	
數值	20	6F	70	70	6F	72	74	75	6E	69	74	79	20
字元	t	o		e	n	l	i	g	h	t	e	n	
數值	74	6F	20	65	6E	6C	69	67	68	74	65	6E	20
字元	a	n	d		i	n	f	o	r	m		t	h
數值	61	6E	64	20	69	6E	66	6F	72	6D	20	74	68
字元	e		g	e	n	e	r	a	l		p	u	b
數值	65	20	67	65	6E	65	72	61	6C	20	70	75	62
字元	l	i	c		a	b	o	u	t		t	h	i
數值	6C	69	63	20	61	62	6F	75	74	20	74	68	69
字元	s		l	i	t	t	l	e	-	k	n	o	w
數值	73	20	6C	69	74	74	6C	65	2D	6B	6E	6F	77
字元	n		t	r	a	d	i	t	i	o	n		o
數值	6E	20	74	72	61	64	69	74	69	6F	6E	20	6F
字元	f		p	o	t	t	e	r	y	.		T	h
數值	66	20	70	6F	74	74	65	72	79	2E	10	54	68
字元	e		v	i	r	t	u	a	l		f	i	
數值	65	20	76	69	72	74	75	61	6C	20	66	69	

2. original file 數值分割：

將 original file 數值分成兩個部份，每個部份各自成為一個部份數值(各 4bits)，並分別置於兩個子數值的 right part。因此 original file 中的數值從 single byte 變成 double bytes。下表以 original file 前幾個值為例：(以?表示此值以 random 方式產生，因而不填數值。)

OF 字元	T		h		i		s			
數值	54		68		69		73		20	
2 parts	L	R	L	R	L	R	L	R	L	R
Each part	5	4	6	8	6	9	7	3	2	0
Size(byte)	1		1		1		1		1	
	L	R	L	R	L	R	L	R	L	R	L	R	L	R	L	R	L	R	L	R
子數值	?5	?4	?6	?8	?6	?9	?7	?3	?2	?0
Size(byte)	2		2		2		2		2	
OF 字元	p		a		g		e			
數值	70		61		67		65			
2 parts	L	R	L	R	L	R	L	R		
Each part	7	0	6	1	6	7	6	5		
Size(byte)	1		1		1		1			
	L	R	L	R	L	R	L	R	L	R	L	R	L	R				
子數值	?7	?0	?6	?1	?6	?7	?6	?5		
Size(byte)	2		2		2		2			

從上表可看出，original file 中每個字元的單位已由 single byte 變成 double bytes。

3. mask file 數值分割：

同樣將 mask file 數值分成兩個部份，將 left part 的數值視為 garbage，於製作 MPT 時將捨棄不使用。

以 mask file 前幾個值為例：

位置	1		2		3		4		5	
MF 字元	T		h		i		s			
數值	54		68		69		73		20	
2 parts	L	R	L	R	L	R	L	R	L	R
Each Part	5	4	6	8	6	9	7	3	2	0
位置	6		7		8		9		10	
MF 字元	d		o		c		u		m	
數值	64		6F		63		75		6D	
2 parts	L	R	L	R	L	R	L	R	L	R
Each Part	6	4	6	F	6	3	7	5	6	D
位置	11		12		13					
MF 字元	e		n		t					
數值	65		6E		74					
2 parts	L	R	L	R	L	R				
Each Part	6	5	6	E	7	4				

4. MPT 製作：

只紀錄 mask file 中 right part 的數值所出現的位置來建立 mask's position table (MPT) ，因此，在製作 MPT 時 mask file 字元位置與 index 如下：

位置	1	2	3	4	5	6	7	8
MF 字元	T	h	i	s		d	o	c
數值	54	68	69	73	20	64	6F	63
Index	4	8	9	3	0	4	F	3

位置	9	10	11	12	13	14	15	16
MF 字元	**u**	**m**	**e**	**n**	**t**	**a**	**r**	**y**
數值	75	6D	65	6E	74	61	72	79
Index	5	D	5	E	4	1	2	9
位置	17	18	19	20	21	22	23	24
MF 字元		**p**	**r**	**o**	**v**	**i**	**d**	**e**
數值	20	70	72	6F	76	69	64	65
Index	0	0	2	F	6	9	4	5
位置	25	26	27	28	29	30	31	32
MF 字元	**s**		**a**	**n**		**e**	**x**	**c**
數值	73	20	61	6E	20	65	78	63
Index	3	0	1	E	0	5	8	3

由上表可知，以加密單位為 4bits 的情況下，mask file 的 right part 形成的 index 的範圍將為 0～2^4-1。

此例所實作做出的 MPT：(index：位置)

000: 5 17 18 26 29 34 41 53 55 56 65 68 78 82 89 93 101 102 108 114 119 132 142 145 146 154 158 166 172 176 177 184 194 196 201 204 206 209 213 218 221 230 233 237 246 253 257 259 269 272 277 281 286 290 294 297 305 308 311 317 326

001: 14 27 39 46 51 79 99 109 135 164 189 195 254 263 278 287 301 310 312

002: 15 19 58 87 98 104 110 134 151 161 174 182 192 243 252 279 284 299 323

003: 4 8 25 32 45 107 118 183 188 193 217 222 229 240 258 276 293 303 315 318

004: 1 6 13 23 35 43 47 59 63 66 75 81 90 113 115 122 123 133 136 138 148 149 155 162 171 173 186 190 197 202 210 223 225 228 231 234 241 250 256 262 267 273 280 289 292 300 321

005: 9 11 24 30 33 42 44 60 69 76 92 95 97 103 112 125 150 157 163 169 181 185 187 207 212 214 216 224 226 236 242 251 265 304 314 316 319 322 324

006: 21 85 144 159 167 179 180 220 271 282

007: 73 94 130 247

008: 2 31 74 91 116 156 211 235 274

009: 3 16 22 36 48 62 64 72 83 106 117 121 137 139 152 160 168 175 215 239 244 266 268 275 291 295 306

010: 309 328

011: 127

012: 40 52 71 100 105 120 124 165 170 200 245 320

013: 10 88 126 238 285 296 327

014: 12 28 38 50 61 70 77 80 84 96 128 131 141 153 208 227 249 255 261 264 288 302 307 313 325

015: 7 20 37 49 54 57 67 86 111 129 140 143 147 178 191 198 199 203 205 219 232 248 260 270 283 298

5. OFI 製作：

(1). 經過數值分割後，original file 的每一字元均變成 2 bytes(於程式實作中預設先處理 right part)，其 index 與數值的變化如下表：

index	1		2		3		4	
OF 字元	**T**		**h**		**i**		**s**	
數值	54		68		69		73	
Each part	5	4	6	8	6	9	7	3
index	1	2	3	4	5	6	7	8
子數值	?4	?5	?8	?6	?9	?6	?3	?7
index	5		6		7		8	
OF 字元			**p**		**a**		**g**	
數值	20		70		61		67	
Each part	2	0	7	0	2	0	7	0
index	9	10	11	12	9	10	11	12
子數值	?0	?2	?0	?7	?0	?2	?0	?7

(2). 由 original file 的 index 1 開始比對 MPT 的 index，並以隨機方式取出該 index 所在位置取代 original file 原有的 index 位置，來製作 OFI。例如：

OF index	1	2	3	4	5	6	7	8
子數值	?4	?5	?8	?6	?9	?6	?3	?7
OF index	9	10	11	12	13	14	15	16
子數值	?0	?2	?0	?7	?1	?6	?7	?6

OF index1 數值為”?4”，MPT 的 index 為 004

隨機得出位置：250

OF index2 數值為”?5”，MPT 的 index 為 005，隨機得出位置：44

OF index3 數值為”?8”，MPT 的 index 為 000，隨機得出位置：235

OF index4 數值為”?6”，MPT 的 index 為 006，隨機得出位置：180

OF index5 數值為”?9”，MPT 的 index 為 009，隨機得出位置：160

OF index6 數值為”?6”，MPT 的 index 為 006，隨機得出位置：282

OF index7 數值為”?3”，MPT 的 index 為 003，隨機得出位置：25

OF index8 數值為”?7”，MPT 的 index 為 007，隨機得出位置：73

OF index9 數值為”?0”，MPT 的 index 為 000，隨機得出位置：233

OF index10 數值為”?2”，MPT 的 index 為 002，隨機得出位置：104….餘類推。

得出 OFI：

250 44 235 180 160 282 25 73 233
104 18 73 135 159 94 85 212 220
142 299 156 167 189 21 8 73 233
87 50 271 219 159 292 73 235 282
16 180 84 85 247 85 201 151 267
130 232 167 308 161 155 167 67 271
196 134 73 247 295 21 289 94 211
167 281 15 107 13 253 58 230 247
252 247 20 179 247 282 161 73 195
85 88 85 88 179 48 282 255 282 73
85 325 15 209 110 1 236 156 21 314
167 317 192 134 130 33 159 222 94
171 130 18 104 178 179 180 159 257
182 43 130 116 144 106 159 222 247
290 58 273 85 147 271 315 220 265
247 285 85 242 220 61 85 113 73
172 104 173 271 57 271 30 271 25
73 170 151 311 323 182 73 216 179
99 85 136 180 101 252 54 167 80
220 82 182 135 220 325 271 300 180
93 134 247 179 260 21 270 282 300
167 78 104 40 271 42 94 183 220
127 144 96 134 294 284

6. 壓縮、加密、放置 mask ID、傳送。

7. 接收方收到資料後，先用 IDEA key2 為位於 OFI 前端的 mask ID 解密，取出 mask ID。找出與 mask ID 相對應的 mask file，根據雙方事先約定的加密單位由 mask file 產生相對應的 MPT。

8. 用 IDEA key1 為 OFI 解密，利用 MPT 與 OFI 來得到 original file。

(a).根據 OFI 的 index 位置，到 MPT 去找到相同 index 位置的 index。

OFI：

index	1	2	3	4	5
Index 位置	250	44	235	180	160
index	6	7	8	9	10
Index 位置	282	25	73	233	104

MPT index 位置 250，index：004

MPT index 位置 44，index：005

MPT index 位置 235，index：000

MPT index 位置 180，index：006….餘類推

(b).由於加密單位是 4bits，需用兩個 index 才能找到一個 original file 的數值。

MPT Index	004	005	000	006	009	006
子數值	?4	?5	?8	?6	?9	?6
數值	54		68		69	
字元	T		h		i	
MPT Index	003	007	000	002		
子數值	?3	?7	?0	?2		
數值	73		20			
字元	s		空白			

(c).得出 original file：

This page has nothing to do with C programming. The rest of this document does, read on and good luck.

四、討 論

IDEA 是一個很好的文件加密工具，但是當多人擁有同一把密鑰時，資料的安全性就會風

險產生。在多人有同一密鑰的前提下，若再加上我們所提出的方法，可以與各方約定不同的加密單位與加密選擇碼，分別給予各方不同的檔案，來進一步確保資料的安全性。在本研究中我們所實作的都是一對一的檔案加密方式，但事實上此一加密方式可變化成一對多或多對多。分別說明如下：

一、一對一：即一個 original file 用一個 mask file 來對應。檔案格式為：

(MF Id)+(MF data)+(MF Id)+(OFI data)

二、 一對多：一個 original file 使用多個 mask file 來對應。檔案格式為：

(MF Id*1*)+(MF data*1*)+ (MF Id*2*)+(MF data*2*)+……+ (MF Id*n*)+(MF data*n*)+ (MF Id*1*)+(OFI data)+(MF Id*2*)+(OFI data)+.. +(MF Id*n*)+(OFI data)

三、 多對多：多個 original file 使用多個 mask file 來對應。檔案格式為：

(MF Id*1*)+(MF data*1*)+ (MF Id*2*)+(MF data*2*)+……+ (MF Id*n*)+(MF data*n*)+ (MF Id*1*)+(OFI data*m1*)+(MF Id*2*)+(OFI data*2 m2*)+.. +(MF Id*n*)+(OFI data *mn*)

延伸上述的觀念及應用 [4] 中的 two-out-of two visual-threshold scheme，建立 secret –sharing scheme 的概念，將 original file 分成不同大小的 m 個區塊，但各個區塊彼此間均有部份重疊的資料，再利用我們所提出的加密方式傳送後，接收方可只選擇 n 份 (n≦m) 即可產生 original file，此一概念如圖六所示。

以 original file 為一篇文章來說，若這一篇文章有三個段落，我們選擇將之分成三份，A、B、C。假設 A 包括第一及第二段，B 包括第二及第三段，C 包括第一及第三段，加密傳送後，接收方可任選至少二份，例如，AB 或 AC 或 BC…即可得出 original file。在未來，經過一定設計後，也可以用資料中的每個位元（bit）為基準來實作此一架構。

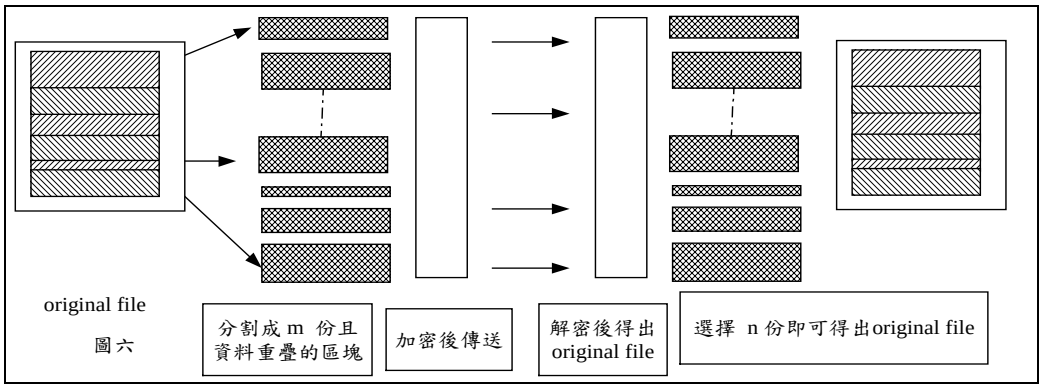
五、參考文獻

[1] A. Curiger, B. Stuber, “Specifications for the IDEA Chip”, *technical report*, No.92/03, ETH Zurich, Institute for Integrierte Systeme, 1992.

[2] X. Lai, J. Massey, “A Proposal for a New Block Encryption Standard”, *Proceedings of Eurocrypt ’90*, Spring-Verlag, Berlin, pp.389-404, 1991.

[3] C. Lin and T. Lee, “A Confused Document Encrypting Scheme and its Implementation”, *Computer & Security*, Vol. 17, No.6, pp.543-551, 1998.

[4] D. Stinson, “Visual cryptography & threshold scheme”, *IEEE Potentials*, Vol.18, Issue 1, pp.13-16, Feb/Mar 1999.



圖六