

# 網路攻防 hw3

電機四 B03901153 陳楷訓

## 1. RSA 1 :

目的：

先觀察題目所給之資訊 `public-key.pem` 和 `flag.txt`，

目標為解開密文 `c`（也就是 `flag.txt`）獲得明文 `m`

Tool：

`gmpy` 套件、`Crypto` 套件、`factorDB`、`rsatool`

分析：

想解開密文（`c`）得到明文（`m`），利用

$$c^d \equiv m \pmod{N}$$

又因為  $m < N$  所以  $m \bmod N = m$ 。觀察上式，要解

得明文 `m` 須先獲得  $c, d, N$

(1)  $c$  : `flag.txt`

(2)  $N$ : 讀取 `public - key.pem` 獲得

(3)  $d$ :

(3-1) 閱讀 `public - pem` 獲得  $N, e$

(3-2) 將  $N$  放入 *factorDB* 獲得  $p, q$

(3-3)  $r = (p - 1)(q - 1)$

(3-4) 用 `gmpy.invert` 解  $d$

$$d = \text{int}(\text{gmpy.invert}(\text{keyPub.e}, r))$$

(4) 已有  $c, N, d$  :

利用 *rsatool* 製作 *private key*

(5) 用 *openssl* 解開密文  $c$  得明文  $m$

截圖：

Code

```
rsa_1.py
import gmpy
from Crypto.PublicKey import RSA
from base64 import b64decode

public_key = b'MGwDQYJKoZIhvcNAQEBBQADWwAwIAJRAK5btPJmADJZz5pvUhw8A0EBds8W310VNHbq47Ie3mw8ew09yiCzHABn/6eX50kQWKhz7vETpg/szZXetbk/EAZr4

# 讀取 public-key 得 n,e
keyDER = b64decode(public_key)
keyPub = RSA.importKey(keyDER)

print(keyPub.n)
print(keyPub.e)
# 把 n 丟去factorDB 得 p,q

p = 1634733645809253848443133883865090859841783670033092312181110852389333100104508151212118167511579
q = 1900871281664822113126851573935413975471896789968515493666638539088027103802104498957191261465571

# 求r
r = (p-1)*(q-1)
# 求d
d = int(gmpy.invert(keyPub.e, r))
print(d)

# 用 rsatool 製作private key
# python3 rsatool-master/rsatool.py -f PEM -o key.pem -n 3107418240490043721350750035888567930037346022842727545720161948823206440518081

# 用 openssl 搭配private key 解密文 (flag.txt) 得明文
# openssl rsautl -decrypt -inkey key.pem -in flag.txt -out flag
```

$n$  輸入 *factorDB* 得  $p, q$

Result:		
status (2)	digits	number
FF	193 (show)	<a href="#">3107418240...09</a> <sub>&lt;193&gt;</sub> = <a href="#">1634733645...79</a> <sub>&lt;97&gt;</sub> · <a href="#">1900871281...71</a> <sub>&lt;97&gt;</sub>

$n, e, d$

```

chenkaixundeMacBook-Pro:Decrypt_RSA kevin85421$ python3 rsa_1.py
n =
3107418240490043721350750035888567930037346022842727545720161948823206440518081504556346829671723286782437916272838033415471873108501919548529007337724822783525742386454014691736602
477652346609
e =
65537
d =
2464047959500535454951489660269116654533786992311074127530789876809197740220694563815619147113537879811295339427915163213618715392333280036495568759211274464407731279798672759722961
973941926913
chenkaixundeMacBook-Pro:Decrypt_RSA kevin85421$ █

```

## 用 rsatool 製作 private key

```

chenkaixundeMacBook-Pro:Decrypt_RSA kevin85421$ python3 rsatool-master/rsatool.py -f PEM -o key.pem -n 310741824049004372135075003588856793003734602284272754572016194882320644051808
1504556346829671723286782437916272838033415471873108501919548529007337724822783525742386454014691736602477652346609 -d 24640479595005354549514896602691166545337869923110741275307898
76809197740220694563815619147113537879811295339427915163213618715392333280036495568759211274464407731279798672759722961973941926913
Using (n, d) to initialise RSA instance

n =
ae5bb4f26003259cf9a6f521c3c03410176cf16df53953476eae3b21ede6c3c7b03bdca20b31c00
67ffa797e4e910597873eef113a60feccd95deb5b2bf10066be2224ace29d532dc0b5a74d2d006f1

e = 65537 (0x10001)

d =
8a422e3a08a81f45185a5debbe77d81cb40c822aa0eca663f3e84ea5efd46ffff858c71f2d5fb3137
d13b93532570f36d772356c23fea51d39a1e7eeb0bb7e208a614526edcb094b9cf6e260ade687c01

p =
e3d237a8d58eb41328c65fca337affe16835804c1b7d136fb920a2bfe1f26670bb51d47b0242be3

q =
c3eca069bc6aa1ccb8e54b2ef6048320eee72e71bc49a4a3db5cdebeba174431f969b29548be21b

Saving PEM as key.pem
chenkaixundeMacBook-Pro:Decrypt_RSA kevin85421$ █

```

## 用 openssl 搭配 private key 解開密文 flag.txt

```
openssl rsautl -decrypt -inkey key.pem -in flag.txt -out flag
```

明文 (m)

*FLAG\_IS\_WeAK\_rSA*

## 2. RSA 2:

目的：

題目已給  $p, q, e, c$  求  $m$

分析：

同 RSA\_1 先求  $N, c, d$  在求  $m$

(1)  $N = p * q$

(2)  $c$  已知

(3)  $d$  之定義： $ed \equiv 1(mod r)$

(3-1) 已知  $e = 65537$

(3-2)  $r = (p - 1)(q - 1)$

(3-3) 將  $e, r$  代入定義，得：

$$65537d \equiv 1 \pmod{r},$$

白話文：65537d 除以 r 餘 1，又 d

$< r$  故 d 為唯一解

$$d = \text{int}(\text{gmpy.invert}(e, r))$$

(4) 已有  $c, N, d$ ，代入  $c^d \equiv m \pmod{N}$  求解

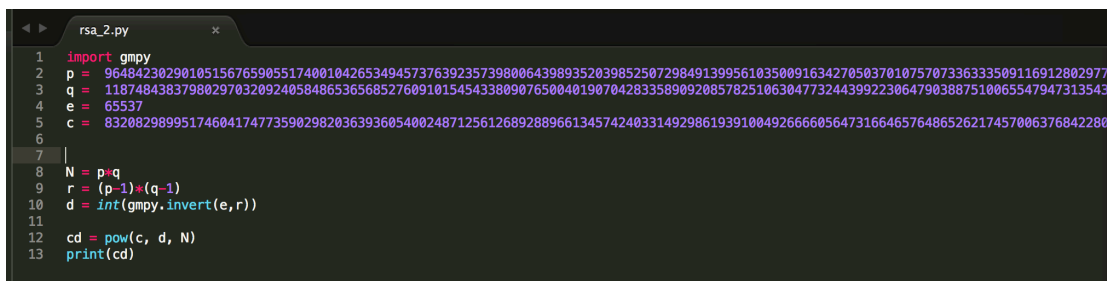
又因為  $m < N$  所以  $c^d \equiv m \pmod{N} = m$ ，

操作型定義為： $c^d \bmod N = m$ ，c, d, N 都已

有所以輕鬆解。

截圖：

Code



```
1 import gmpy
2 p = 96484230290105156765905517400104265349457376392357398006439893520398525072984913995610350091634270503701075707336333509116912802977
3 q = 11874843837980297032092405848653656852760910154543380907650040190704283358909208578251063047732443992230647903887510065547947313543
4 e = 65537
5 c = 83208298995174604174773590298203639360540024871256126892889661345742403314929861939100492666605647316646576486526217457006376842280
6
7
8 N = p*q
9 r = (p-1)*(q-1)
10 d = int(gmpy.invert(e,r))
11
12 cd = pow(c, d, N)
13 print(cd)
```

明文

```
[chenkaixundeMacBook-Pro:crypto kevin85421$ python3 rsa_2.py  
5577446633554466577768879988
```

FLAG = 5577446633554466577768879988

### 3. stego1.png

這題經過很多嘗試：

- (a) 用 *stegsolve* 看不同狀態下的圖片，並未有收穫 --> 非 LSB
- (b) 用 *binwalk* 看是否有隱藏檔案，只看到一個 *png* 檔案和 *zlib*  
→ 代表並未藏東西
- (c) 用 *hex friend* 看 16 進位中是否藏有 *flag* --> 無收穫
- (d) 比對 *stego1.png* 和其他正常之 *png data format* 比對
- (e) 修改 *hex* 中的圖片 *height, length*  
(改變高度會只顯示一部分圖片，可以隱藏資訊)

後來詢問助教獲得提示

- (a) <http://www.libpng.org/pub/png/book/chapter09.html>
- (b) <https://www.w3.org/TR/PNG-Filters.html>

於是 google 關鍵字：png filter stego



### 「png filter stego」的圖片搜尋結果



→ 更多符合「png filter stego」的圖片

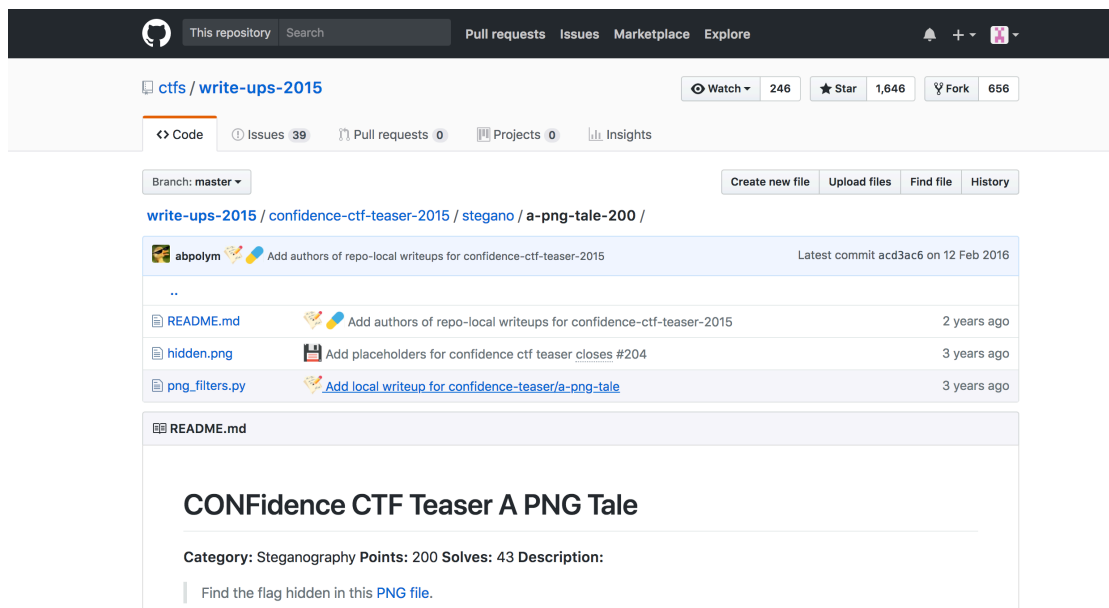
檢舉圖片

### a-png-tale-200 - GitHub

<https://github.com/ctfs/write-ups-2015/tree/master/.../a-png-tale-200> ▼ 翻譯這個網頁

Write-up. by polym. Note: This writeup is based on This Writeup. We are given a 800x800 PNG picture that seemingly contains no hidden information on first sight: Examining the picture with common steganotools such as stegsolve or outguess and hexeditors, we also don't see anything obvious. However, PNG applies ...

爽居然一樣



用 python2.7 png\_filter.py stego1.png

```
[chenkaixundeMacBook-Pro:stego kevin85421$ python2.7 png_filter.py stego1.png
PNG Signature: ('\x89', 'P', 'N', 'G', '\r', '\n', '\x1a', '\n')
Pos : 8
Type: IHDR
Size: 13
CRC : 5412913F

Pos : 33
Type: IDAT
Size: 10980
CRC : 98F96EEB

Pos : 11025
Type: IEND
Size: 0
CRC : AE426082

Data length in PNG file : 10980
Decompressed data length: 1920800
Flag: DrgnS{WhenYouGazeIntoThePNGThePNGAlsoGazezIntoYou}
```

**Flag : DrgnS{WhenYouGazeIntoThePNGThePNGAlsoGazezIntoYou}**

#### **4. stego2.jpg**

目的：

破解圖片的隱碼術，找出 flag

分析：

這題其實很簡單，但是由於我當初沒有先安裝

binwalk 所以瞎子摸象了很久，我的解法為：

- (1) 觀看許多 jpg 檔的 16 進位編碼，加上網路資料，發現 jpg 檔案都為 FFD8 開頭，FFD9 結尾。
- (2) 用 hex friend 打開 stego2.jpg 觀察 16 進位編碼，發現 FFD9 後居然有其他編碼。推測在

jpg 檔之後仍藏有一個檔案。

(3) 將 FFD9 後接的編碼 504B0304 丟去

google，發現是 zip 檔的 header。

(4) 將 FFD9 後的所有編碼複製下來，貼到另一個檔案，並將副檔名改為.zip。

(5) 解壓縮得 flag

(後來發現用 binwalk 秒解 QQ)

截圖：

正常圖片編碼

(a) 開頭 FFD8

```
0 FFD8FFE1 28274578 69660000 49492A00 08000000 08000F01 02000600 00006E00 00001001 02001500 00007400 00001A01 05000100 00008A00
56 00001B01 05000100 00009200 00002801 03000100 00000200 00003101 02002B00 00009A00 00003201 02001400 0000C600 00006987 04000100
```

(b) 結尾 FFD9

```
376824 6224F2F8 D2831711 B9041655 6B8C0302 EA6E49C3 1247C1B0 3AC8CA09 27BA47CA AA0EA9E3 FAEF49F8 8D8FC865 60549517 566406C0 F92073F
376880 D41858EC 88047D40 A92B2B39 8C913F23 58AE488A C1B89467 601463C9 18DEDEE0 2DC702DA C2D7080A 40D48F5C A86D9330 FCCFADEB FFD9
```

stego2.jpg 編碼

(a) 開頭 FFD8

```
0 FFD8FFE0 00104A46 49460001 0101012C 012C0000 FFD80043 00050304 04040305 04040405 05050607 0C080707 07070F0B 0B090C11 0F12121
56 0F111113 161C1713 141A1511 11182118 1A1D1D1F 1F1F1317 2224221E 241C1E1F 1EFFDB00 43010505 05070607 0E08080E 1E141114 1E1E1E1
```

(b) FFD9 之後仍有編碼

```
40768 E34DCA49 60154126 8A287B14 96A3EDD2 23322B9D C08E71DA 8A28ACE7 7B9A53B5 B63FFD9 504B0304 14000000 08003B84 7544CD63 38D3E18
40824 0000F783 00000800 1C00676F 74322E6A 70675554 0900039A 1C2C539F 1C2C5375 780B0001 04E80300 0004E803 00009DFD 6378655D D42D8AA
```

將 FFD9 後之編碼複製貼上到一個空白檔案



0	504B0304	14000000	08003B84	7544CD63	38D3E180	0000F783	00000800	1C00676F	74322E6A	70675554	0900039A	1C2C539F	1C2C5375	780B0001
56	04E80300	0004E803	00009DFD	637865D0	D42D8AAE	D8498515	DBB69D8A	60A762DB	66C5B66D	73C5A9D8	762AB66D	55EAA6DE	F7D3D967	EF7BEF39
112	334F5F68	3E59738C	D1D17A1B	63FE6A7F	56FE6C03	BE488949	8A014040	010090CF	3FC09F35	80080012	1C021202	1C121202	120A0A12	1AF60B2C
168	2C0C0C2C	1A2212FC	97AF6898	985FD130	30B0F1C8	08B07148	70313008	690949C8	29A8A8A9	B008E818	E92819C9	28A928FF	4E020205	05050B0D
224	8B0A0E88	4A89D8D1	40F9FFF8	FAD30340	86064907	35030321	06802283	802183FC	1900107C	FA0901F2	CF05F88F	0B0414EC	D35F2868	1858B3CF
280	07805F00	A0206060	A0E06010	10E0E09F	BFFA7CFE	0E004786	40216212	82445534	80227640	630E88CF	872611AE	EF45579A	BD266531	740C8481
336	C5F88A89	854D464E	414945CD	CAC6CEC1	C9C52DF2	4D544C5C	42524A59	45554D5D	4353CBC8	D8C4D4CC	DDC2D2C9	D9C5D5DC	DDC333E8	47704868
392	58784442	6252724A	6A5A7A46	41615171	49695979	45436313	B0B9A5B5	ADBDAF7F	60706878	64746C6E	7E617169	79E5D7EA	CEEEDEFE	C1E1D1F1
448	C9E9CDED	DDFDC3E3	D3F3CBEB	DFB84000	6020FF79	FD6FE342	FE8C0B14	1C1C0C1C	EA6F5C20	A06E7F1F	40068720	62824411	52843270	4025660E
504	8046138E	CFAEF858	2161518A	4637749C	85C52065	DD21BBF9	1BDA3F91	FDDF1758	E0FFABCB	FE2BB0FF	8E6B1500	0F06F259	3C306480	00E0452B
560	4094320F	34E23F8D	EA86C8E9	4210BEA0	E9ABBC5C	579AA5C2	5CB91C74	84A6BA47	B473086E	4AC2F27E	50A4E234	7FCA82A0	397F58AA	1BF86AC6
616	EB01E566	7F4AAC4A	0591CE2F	DCAA820E	5A19E69B	88D405E4	66D304A9	447A8B78	56A70AF4	7D2BBD8E	71546C06	79951996	D92F172A	67CAB2FE
672	C96A6A57	89D8E432	12CDF41F	06152807	B2175DB7	C7A07BC1	9D77CB26	99A26104	DCEFF3AB	25711EF5	E57D81C5	C60D42AC	32859472	0826ABEB
728	EB1887BC	A54EBAB6	A157840A	8686F2F6	D449B62C	7EA2ED2D	1CD405CC	84B6EC6D	BD9356A7	DCCE4F4C	040D2804	CDF34B8D	F7D5E463	D20EFC18
784	16A951A8	51D8D000	3EB6F3BF	030FF7CE	D561990B	B98A16FD	5385BE8A	FD8AAF3E	4C700637	08058FD8	3E5A4658	6A71B12B	13C8B222	4E5BA81D
840	F1B15A56	1E58E266	CB625AE7	6A6B1B62	C2605E14	19D012CB	DE8DC690	55160746	78DD3D29	86A941FD	F419A489	198E2BCE	72C893D4	386AF48C
896	D50A5202	AE211AE9	F72FA84E	7AF28948	756BD389	83C3A314	394215DB	5A2C61B4	687DCF2D	BD0F0F44	B3D32A1B	DAE53D06	A70521F5	9C7C7D4F
952	0596D6DD	02DBE86A	F7CCE1F0	D38660CA	D974E997	FA4435E9	A599B040	35BE58C4	A4A82752	DCA0738C	BE8D85C6	E7AD0574	F9153A2	14374749
1008	3297EB77	3E95752C	4A5CF3F5	5FC4E284	F2662940	E56BE89C	30695FB8	EF7E0D4A	0D2129FF	03C01E87	9A0EAE85	0D00729F	E2EDBC67	1D472D07
1064	15970F75	0C3A8D34	E232D0E2	A60A126B	49A19324	67EA706C	20F79372	251F26AB	5A17E1ED	D336FF6F	393E5255	3057FE1D	63106EA4	855D101A
1120	D3661592	49679228	7F414951	70B02021	02A045C5	222DF8D5	F907921F	A66D6087	467EF002	E4AAA8FD	AB917587	B0F56D13	7254AB00	5735CA8C
1176	4CA3033E	530B8878	B5BC1525	72322D12	5EB4D906	122FB880	7D5616B6	4A8886C4	6D0081B2	8CA8AC1A	DF307163	0243B5AD	3C726259	59CA7AC0
1232	939F14A4	998B8E5E	B03E5907	BCBA5CCE	389497F1	6626668C	78B1CAAC	35E838BC	3AEF72BE	F92391A1	D2F82479	0D7D0B61	F2857EFA	D89D6ACE
1288	AC3093C1	A48B5B3C	F1FEB669	1D83068A	3A6641AF	8F4C7083	26A992A8	4A5B497F	63BCD623	7BE608E7	6ECD772D	CA481C27	1191B6A3	96A54844
1344	57A454F4	B1536AAE	9E63AB19	36FFBCF8	164CFCE4	414B0714	6D0CC37D	9ECC119E	AF2D2532	D7318B3A	FFC4C133	FAACF92B	8F8E3562	87D37A84
1400	5AABCE41	41638EDA	6A996435	8F2CB93B	16690A4B	534E6BAD	D21C2F15	3865C3C8	EC760C17	EE62FA77	89081E3D	2990A8A0	0C6CACC9	F5938999
1456	4B51CE2A	574876EA	0DABCE1F	C23E0D40	F1DB0E4F	C14E5A07	09B6CC62	370F47E0	2A51BD09	CC9215EA	D034C282	E0E13490	8DFAE8BC	B811AE8C
1512	BA21544D	40DBE86A	C5051100	2929CA3C	30D6CF7A	201617A1	35011445	D13E8D9F	74D34110	BACC6952	B8E57B9A	A7D20B61	F2857EFA	250C8B9F
1568	74A4C7B6	83369231	3E83B54B	950FB952	AD3F94D4	493581B3	FB6E7688	63464444	0D1C85D6	618DBE39	29F556B6	E78F1B18	906DD883	876CDE86
1624	2CCA8713	D11F8EAC	2AB570E5	BCF8D4C4	D0A7D4D6	7D7BF1CD	33629284	0A151ADB	6A25D986	D3C457CA	416BE805	4BC4D623	176A55A5	4C122C33
1680	4B4B5E3B	2F4661D3	D8B82E3D	354EE01C	F6A2B219	7897B632	BC077B71	56E4B748	E432AE6C	8E7D47B4	E8EDCEE5	0F40A2E1	B081920F	95C0995A
1736	1B78E50B	6D2D0594	6F5F7280	D15A5A2A	633FC0CE	14211C5A	8CCC07F7	FB54F0F9	CEE85D94	1CF738CB	6B6B5A06	C606A2F0	99340E3D	7091334A
1792	F3226BBD	676B567A	48F2ABA3	99773F24	1A820B00	81D078A2	4EEC17E1	EAF61B41	731DD544	B56C8A78	C569AFA4	4CF3C5D0	A621F16E	86ED55C8
1848	E287E19A	F514BCBE	A5EE73EE	10E7B033	86423DF6	A5243765	6CDD70EE	3583F16B	F2030E8C	2F05A79B	0FF392EE	751A0556	18CCF0B5	9B708992
1904	7665111F	88D70F45	2D71BACF	81E7E1F2	5F88AF31	1F4997B6	0611D8E9	C04475C5	252D0C3B	E8A113C8	87F826ED	B51ED2ED	13354E4B	584FC349
1960	BE28E537	DD00B779	144E17EA	48D85EEC	BF7CFB61	6B70CF6C	A21573E0	E378F080	DE92706A	079C6E71	1A1343B0	34473E79	6C973137	8E374B7E
2016	AA8AF050	3ED1A429	3C55D40E	9B20D1C7	647626F4	5A1C9E81	767EFE75	C962D53	FF1248F6	A142D835	FAAE5D83	AAD73323	6F023B02	39D0FEC1
2072	EC23EBF8	F8C4EE68	A35DE5C1	F43D89F1	28A2D2F9	CC1783B2	882C7C2C	8EBD82EA	9C8CDEB1	6A903E11	7C489C93	157EE386	96B8D2D2	47B84E68
2128	98A4DAB1	919D1E6E	2F68462D	ADAB03A3	8A8C7048	0C9AA458	9476E929	A2C2014B	2820E7DE	BF40EE81	4C70BA58	F28B3CA5	1D7C03D8	53570579
2184	F42E4F12	CACFA623	E6B15C87	8898E73C	C474824C	B99F2D5E	5CB0820A	1884D75B	5AD26FC9	6D90EAE0	3192500A	B286AB1B	3A2FE6F1	048C9F3B
2240	CD851265	CAE5AA6A	029B20A2	FA412FB9	3FDC5A1B	3DD08475	50C010CF	92AC661B	2A91C251	DE303A41	72AB3198	20B2A284	1822029A	BF161504
2296	1D004D51	14C04FBE	D21BAA88	FDD8023E	54EACA4C	8348BA97	31172E28	0A565713	F68B98CC	376482C5	61B4B29E	934FD670	ADDE16E6	B0CA439E
2352	68ADCDF0	018DB3B4	9C2A3302	04D749FE	DB78404C	49619332	5E7BA024	71B52612	F325A38F	A5B5EA54	8B48FDEF	BE966EB6	B9CA2AF6	F76DAF79
2408	E733FD14	B861EB25	D34AA7C6	0302A992	A619FA78	84C33C38	76AEAA67	C59AEFAE	CC30152C	D2E90B3C	FAA8BAEC	39D72BEB	6B5AE3A8	4D2D608C
2464	3BBE0FB2	6976F586	482BC285	3411A515	3A1557A2	E79C35B1	4A4DEE85	E344CA51	F131C9FD	A7531F16	9CA74CC2	EB12FB0F	2F97F88F	0C941F47
2520	506ED5AA	D24AA71C	F49267DA	7E4616E9	E15350D5	7A4DF328	C6D1926B	B8D9D0A7	BFEE2B0E	90C77F59	CC74F531	A4CCA3D8	BA7AE62A	5FE8E2BA
2576	D65A3186	0BA6EFF7	D39AB3DC	BEESCAAS	BFAAFD2E	494711C3	984387E0	57516084	39976FF9	3E5692A2	719F2959	14CF8F54	F1D3B841	8699240A
2632	17ECCE40E	3F688B8C	D1A2E052	BD49082F	E8BB650B	7DDC30F9	5C44EBC6	B4D039A3	D37D1503	9665C0DC	14A5785A	C2371D6B	4CA8437D	F777EA9C
2688	7CDE160C	49B7EA09	A396D6AC	11C2A8F7	89C8F860	FCA214AC	8BA00B0D	6CE0FD55	A9769B3C	B9106D3C	84E2587E	3FC46F59	D92115AF	3EDC4226
2744	254D399E	6DFC0758	E9C7B49A	25325332	EFF5FB13	C535E172	4A8F7030	2E520E3E	6A880C11	8E740CC5	AF45F9AF	AF73A030	5F58F8FB	C316DDEC
2800	D8D2796C	9D623B56	S85C9CDA	397B61A3	0469C35A	C16EAF34	E6FBC604	87105904	D1CB26C1	8DAAD142	B00A5993	1440A425	45017468	9F445910
2856	410928EC	4227AB43	600E6411	8443A323	3A76D072	55EB8CE1	56CDA471	26A9B658	5BF35FFF	623913C1	60913857	120BD9A3	0336E812	FFEB12D7
2912	49B58616	7A6318ED	1C923E4D	D27E8FD0	6B59C4C6	09D8B3AD	E6D863A4	03E24FFD	65A2BAB7	84D425EC	F4AC171D	61426A74	E442F3D5	BBF48056
2968	D6AAE132	EA26C3DC	66E590FE	5CA1BE6F	C97DAE73	6B9BD4B4	3D3DDFEF	F2B0F99D	AF5F8E52	6E483D57	AA81DEFE	3107BBD5	030428C8	E3BDD0B5

存成 flag.zip

Save As:

Tags:

Where: 

stego

Cancel

Save

解壓縮得 flag

