RSA

公钥、密钥生成

一般情况下：公钥和私钥均由两部分组成。

取两个大素数 p q 。

设 p*q = n ， n为公钥、私钥的第一部分。

获取公钥第二部分

取 e ， 使得

```
 1 < e < (p-1) * (q-1) && gcd( e,(p-1) * (q-1) ) = 1;

// gcd(a,b)=1 表明  a与b互质
```

公钥即为：PK = (n,e)

获取私钥第二部分

取 d ， 使得

```
(d * e) mod ((p-1)*(q-1)) = 1  //将选定的 e p q 带入即可算出 d
```

私钥即为：SK = (n,d)

加密解密

设 明文为 M 密文为 C ， 现在有公钥 PK = (n,e) SK = (n,d)

加密

$$C = M^e mod n$$

解密

$$M = C^d mod n$$

商用情况（以 2048 bit加密为例）

加密填充

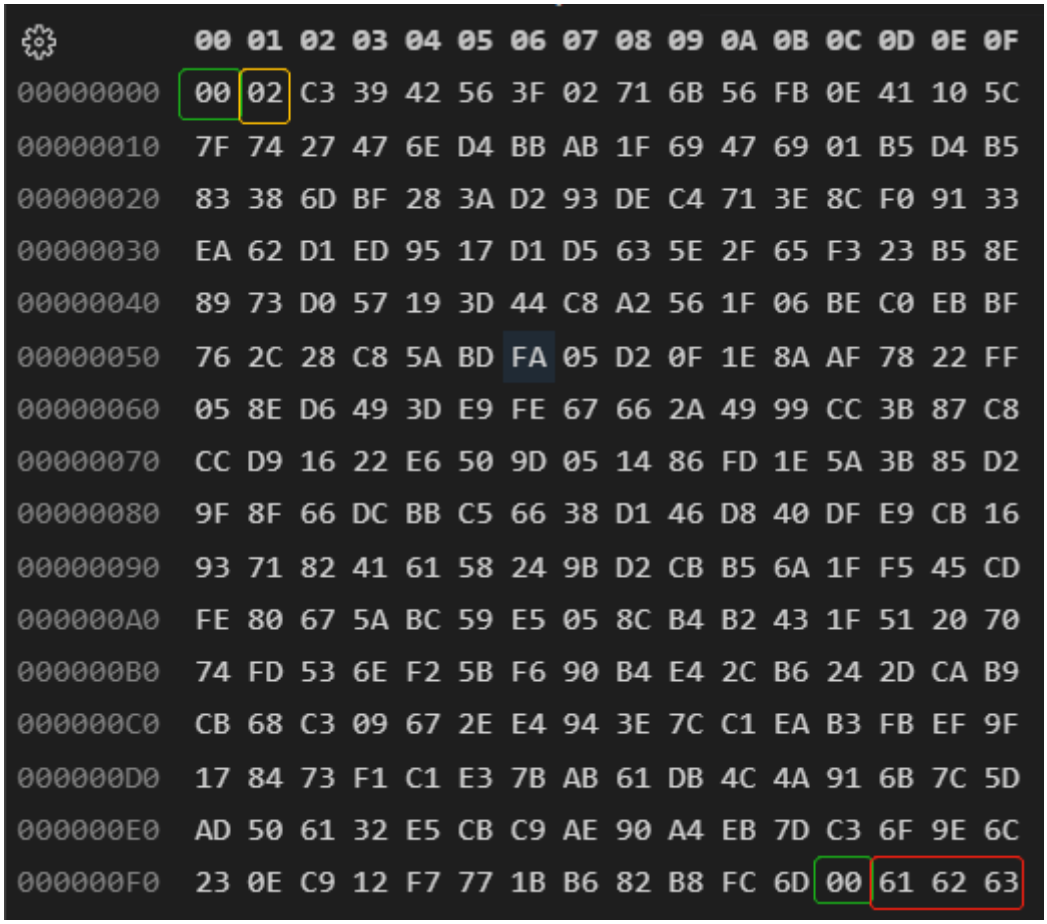
PKCS #1 明文填充，公式如下：EB长度为2048bit（256 byte）

$$EB = 00 + BT + PS + 00 + D$$

一般加密时认为 BT 为 0x02 ， PS为随机填充的字节 D即为需要加密的明文，例子：

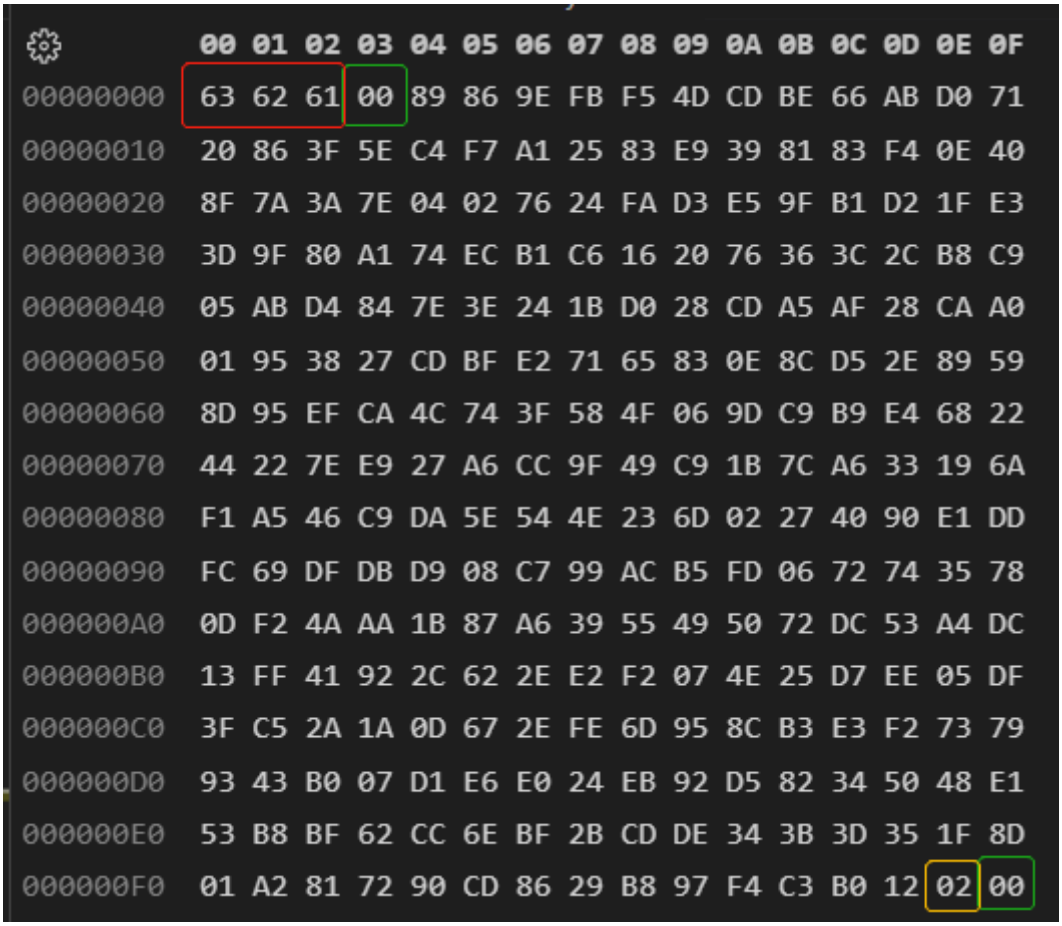
例如需要 加密原明文： “abc” 三个字节，填充后： （绿色为填充的 00 黄色为填充的 BT 红色为D）

没被圈起来的就是填充的随机PS（PS不能填充 0x00 ）



	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	00	02	C3	39	42	56	3F	02	71	6B	56	FB	0E	41	10	5C
00000010	7F	74	27	47	6E	D4	BB	AB	1F	69	47	69	01	B5	D4	B5
00000020	83	38	6D	BF	28	3A	D2	93	DE	C4	71	3E	8C	F0	91	33
00000030	EA	62	D1	ED	95	17	D1	D5	63	5E	2F	65	F3	23	B5	8E
00000040	89	73	D0	57	19	3D	44	C8	A2	56	1F	06	BE	C0	EB	BF
00000050	76	2C	28	C8	5A	BD	FA	05	D2	0F	1E	8A	AF	78	22	FF
00000060	05	8E	D6	49	3D	E9	FE	67	66	2A	49	99	CC	3B	87	C8
00000070	CC	D9	16	22	E6	50	9D	05	14	86	FD	1E	5A	3B	85	D2
00000080	9F	8F	66	DC	BB	C5	66	38	D1	46	D8	40	DF	E9	CB	16
00000090	93	71	82	41	61	58	24	9B	D2	CB	B5	6A	1F	F5	45	CD
000000A0	FE	80	67	5A	BC	59	E5	05	8C	B4	B2	43	1F	51	20	70
000000B0	74	FD	53	6E	F2	5B	F6	90	B4	E4	2C	B6	24	2D	CA	B9
000000C0	CB	68	C3	09	67	2E	E4	94	3E	7C	C1	EA	B3	FB	EF	9F
000000D0	17	84	73	F1	C1	E3	7B	AB	61	DB	4C	4A	91	6B	7C	5D
000000E0	AD	50	61	32	E5	CB	C9	AE	90	A4	EB	7D	C3	6F	9E	6C
000000F0	23	0E	C9	12	F7	77	1B	B6	82	B8	FC	6D	00	61	62	63

注意：因为BT位为高位，D为低位，所以需要将上述的字节序列转换端序：转换后：



	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	63	62	61	00	89	86	9E	FB	F5	4D	CD	BE	66	AB	D0	71
00000010	20	86	3F	5E	C4	F7	A1	25	83	E9	39	81	83	F4	0E	40
00000020	8F	7A	3A	7E	04	02	76	24	FA	D3	E5	9F	B1	D2	1F	E3
00000030	3D	9F	80	A1	74	EC	B1	C6	16	20	76	36	3C	2C	B8	C9
00000040	05	AB	D4	84	7E	3E	24	1B	D0	28	CD	A5	AF	28	CA	A0
00000050	01	95	38	27	CD	BF	E2	71	65	83	0E	8C	D5	2E	89	59
00000060	8D	95	EF	CA	4C	74	3F	58	4F	06	9D	C9	B9	E4	68	22
00000070	44	22	7E	E9	27	A6	CC	9F	49	C9	1B	7C	A6	33	19	6A
00000080	F1	A5	46	C9	DA	5E	54	4E	23	6D	02	27	40	90	E1	DD
00000090	FC	69	DF	DB	D9	08	C7	99	AC	B5	FD	06	72	74	35	78
000000A0	0D	F2	4A	AA	1B	87	A6	39	55	49	50	72	DC	53	A4	DC
000000B0	13	FF	41	92	2C	62	2E	E2	F2	07	4E	25	D7	EE	05	DF
000000C0	3F	C5	2A	1A	0D	67	2E	FE	6D	95	8C	B3	E3	F2	73	79
000000D0	93	43	B0	07	D1	E6	E0	24	EB	92	D5	82	34	50	48	E1
000000E0	53	B8	BF	62	CC	6E	BF	2B	CD	DE	34	3B	3D	35	1F	8D
000000F0	01	A2	81	72	90	CD	86	29	B8	97	F4	C3	B0	12	02	00

填充后的一整个256byte的字节序列就作为M（没错，填充后M为一个 2048bit 长的大整数）

密钥获取

因为是2048bit 的RSA，要求n为2048bit长，从网上生成的公钥密钥通常为Base64编码的，例如：

我们到http://www.metools.info/code/c80.html 选择 PKCS#1 2048bit 生成密钥对：

公钥：

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEArFpzwIkohy9RQuA+poeS
R9pRu0oFuZQ7TVPxNAOyqyaSWxCg5yjAQbfcheeOBR3acGdKFaGeq9Ctk7yFlICu
fCjMERO95OOT4sw8P1vwO5t6x+Dn12MmacckjzsO29mKYX+yEf9ZeQQ2yv2PI+f2
jy4DCIkd3WZIH4O6MK3ZOHiss25PBpC3W6Rf1cKxlBbS/YgarNeyBMbTxFAujRR3
p1YhTGwLA3nNgfauYPTcOtvOONE9Ju8vVfaXSfpZrEy14siL82hDAd7q0LxwTRxW
1Puk1TZ2KLRJV32sl1vXTtaHDRgC2LVRVvjFOJUjtBlxBbc+OgSt5uYGi3jWRaw9
7QIDAQAB
-----END PUBLIC KEY-----
```

私钥:

```
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCsWnPAiSiHL1FC
4D6mh5JH2lG7SgW5lDtNU/EOA7KrJpJbEKDnKMBBt9yF57QFHdpwZOoVoZ6rOK2T
vIWUgK58KMwRE73k7RPizDw/W/DTm3rH4OfXYyZpxySPOzTb2Yphf7IR/1l5BDbK
/Y8j5/aPLgMIiR3dZkgfjTowrdk4eKyzbk8GkLdbpF/VwrGUFtL9iBqs17IExtPE
UC6NFHenViFMbAsDec2B9q5g9NzS2/Q4OTOm7y9V9pdJ+lmsTLXiyIvzaEMB3urQ
vHBNHFbU+6TVNnYotElXfayXW9dO1ocNGALYtVFW+MXQlSOOGXEFtz7SBK3m5gaL
eNZFrD3tAgMBAAECggEAKltIWybaCO79/3twqihjMLlcoLR+V68wK77XZjsKa7lc
+nhJssV3Ci+PSHE68o71oe4gbNR9VChCoC4Sff5p4yV7kS7S2jHRUOlx/PgxZS+u
```

```
qvSAxZQeWpyQRYr9YJJDydZCAt5INM9uT9CknuRg2asrH6vi5Qg/7VEd3dy7p37J
VeJf2NgJKUnJ8rfMxh83oF+LOefsGoOjJYpOFf+REXSW3jWcECgOnanoUPR9eD1u
qLROunxlaIeuPqNFFtqNvHZkgQ2Wkv871pg8pKnFJgU/xvmTWeo5Kg+T7qwp+AL4
4Z46MmzXdD5XinM56xhwWEAtyAyBwMXX/elLpy71GQKBgQDZAXNdevDsQrTEGyr1
EhN6vVGw1Xth//PROrIGTthOVPwrpxRzwRDrT5YgPxu2P+SGWHJmWIwsb4loXGXg
jC6e2yd98ZOrSIukie7/EtxMxZOOQ4utpLbTXkJ7C45QmIldyOY5fxxKt2Jvc7Y4
+6HFvZDTqO8ytxDSASa5GlRrPwKBgQDLUu8/OsVOBFJDUIP4r7nyb5eJUk3AvLK/
wtjxFWwgCwV9b/3wwBGkJV3y5BGR8nKRKRbYWeyhGehzRSrfhGMQRjhror8OTybZ
+QthUYuS6q14q/ec8L+U3ks3KzmhuvWheafkGzXgbVTayOklr05ydpYH2MeBRSNc
+vY/4C7nOwKBgGIkzdBRcfTolmi7AMqlyjmQtgmMA6lm4RaXvTwtHKhAZ2w2vIE9
qDFEZVlgMWBTtT8tcU/obDOMlNCmOUO9GTivVUUYpPiqbr2TxIuINCpklZy1j86Q
2D7wOpD7wPmigz3wWWpXmqwQFGzjoF5VL+OGtyGDHeuJ8AmKi8Dbr6JxAoGAFPm8
VuA+nJrThcBDrR8r1rfucSs9fDm7Uw8d5Fkqa2/XpCYuk2ytJTr5ozomGeFiMlmD
OtQMFl6e3W1wS9jTg3VLEseaZCIaRvhrVSHdfU2akc1j4RvxF8GLDuj1rBqmhn8P
c/feqgV9cIizjMC2HhbfyXVyt4JTcJ8vUk2laSMCgYB+zQXngc/Azphc5iH9+vTE
n+grs3TdQBufDpz3mgH4MgRf82/oTMnY19ZXBvfu7n6TvAeX2fk8C0f9wSfPj54t
9MocrEJUMUmWWPk8D/wwtI3KzOtlvsTtmVq4CoiOYw/OiD63ELqUOX91tqIRf8rZ
V8Rnj5k7fkFBMAIWxv9uOQ==
-----END PRIVATE KEY-----
```

解析

#PKCS1的 公钥的格式为 PK = (n,e)，私钥格式为：SK = (n,e,d,p,q,dmp,dmq,crt)，后面几个是用来加快计算的（看下文）。

我们使用ASN .1 解码解析上述的公私钥，在线解码地址：

以公钥为例，解析结果：

```
<SEQUENCE>
 <SEQUENCE>
  <OBJECT_IDENTIFIER Comment="PKCS #1"
Description="rsaEncryption">1.2.840.113549.1.1.1</OBJECT_IDENTIFIER>
  <NULL/>
 </SEQUENCE>
 <BIT_STRING>
  <SEQUENCE>

<INTEGER>0x00AC5A73C08928872F5142E03EA6879247DA51BB4A05B9943B4D53F13403B2AB26925
B10A0E728C041B7DC85E7B4051DDA70674A15A19EABD0AD93BC859480AE7C28CC1113BDE4ED13E2C
C3C3F5BF0D39B7AC7E0E7D7632669C7248F3B34DBD98A617FB211FF59790436CAFD8F23E7F68F2E0
308891DDD66481F8D3A30ADD93878ACB36E4F0690B75BA45FD5C2B19416D2FD881AACD7B204C6D3C
4502E8D1477A756214C6C0B0379CD81F6AE60F4DCD2DBF438D13D26EF2F55F69749FA59AC4CB5E2C
88BF368430 1DEEAD0BC704D1C56D4FBA4D5367628B449577DAC975BD74ED6870D1802D8B55156F8C
5D09523B4197105B73ED204ADE6E6068B78D645AC3DED</INTEGER>
   <INTEGER>65537</INTEGER>
  </SEQUENCE>
 </BIT_STRING>
</SEQUENCE>
```

就可以得到PK：

```
n =
0x00AC5A73C08928872F5142E03EA6879247DA51BB4A05B9943B4D53F13403B2AB26925B10A0E728
C041B7DC85E7B4051DDA70674A15A19EABD0AD93BC859480AE7C28CC1113BDE4ED13E2CC3C3F5BF0
D39B7AC7E0E7D7632669C7248F3B34DBD98A617FB211FF59790436CAFD8F23E7F68F2E0308891DDD
66481F8D3A30ADD93878ACB36E4F0690B75BA45FD5C2B19416D2FD881AACD7B204C6D3C4502E8D14
77A756214C6C0B0379CD81F6AE60F4DCD2DBF438D13D26EF2F55F69749FA59AC4CB5E2C88BF36843
01DEEAD0BC704D1C56D4FBA4D5367628B449577DAC975BD74ED6870D1802D8B55156F8C5D09523B4
197105B73ED204ADE6E6068B78D645AC3DED

e = 65537
```

同理解析SK:

```
n =
0xAC5A73C08928872F5142E03EA6879247DA51BB4A05B9943B4D53F13403B2AB26925B10A0E728C0
41B7DC85E7B4051DDA70674A15A19EABD0AD93BC859480AE7C28CC1113BDE4ED13E2CC3C3F5BF0D3
9B7AC7E0E7D7632669C7248F3B34DBD98A617FB211FF59790436CAFD8F23E7F68F2E0308891DDD66
481F8D3A30ADD93878ACB36E4F0690B75BA45FD5C2B19416D2FD881AACD7B204C6D3C4502E8D1477
A756214C6C0B0379CD81F6AE60F4DCD2DBF438D13D26EF2F55F69749FA59AC4CB5E2C88BF3684301
DEEAD0BC704D1C56D4FBA4D5367628B449577DAC975BD74ED6870D1802D8B55156F8C5D09523B419
7105B73ED204ADE6E6068B78D645AC3DED

e = 0x10001

d =
0x2A5B485B26DA08EEFDFF7B70AA286330B95CA0B47E57AF302BBED7663B0A6BB95CFA7849B2C577
0A2F8F48713AF28EF5A1EE206CD47D542842A02E127DFE69E3257B912ED2DA31D1534971FCF83165
2FAEAAF480C5941E5A9C90458AFD609243C9D64202DE4834CF6E4FD0A49EE460D9AB2B1FABE2E508
3FED511DDDDCBBA77EC955E25FD8D8092949C9F2B7CCC61F37A05F8B39E7EC1A83A3258A4E15FF91
117496DE359C1028349DA9E850F47D783D6EA8B44EBA7C656887AE3EA34516DA8DBC7664810D9692
FF3BD6983CA4A9C526053FC6F99359EA392A0F93EEAC29F802F8E19E3A326CD7743E578A7339EB18
7058402DC80C81C0C5D7FDE94BA72EF519

p =
0x00D901735D7AF0EC42B4C41B2AF512137ABD51B0D57B61FFF3D13AB2064ED87454FC2BA71473C1
10EB4F96203F1BB63FE486587266588C2C6F89685C65E08C2E9EDB277DF193AB488BA489EEFF12DC
4CC59D34438BADA4B6D35E427B0B8E5098895DCB46397F1C4AB7626F73B638FBA1C5BD90D3A8EF32
B710D20126B91A546B3F

q =
0x00CB52EF3FD2C54E0452435083F8AFB9F26F9789524DC0BCB2BFC2D8F1156C200B057D6FFDF0C0
11A4255DF2E41191F272912916D859ECA119E873452ADF84631046386BA2BF344F26D9F90B61518B
92EAAD78ABF79CF0BF94DE4B372B39A1BAF5A179A7E41B35E06D54DAC8E925AF4E72769607D8C781
45235CFAF63FE02EE7D3

dmp =
0x6224CDD05171F4E89668BB00CAA5CA3990B6098C03A966E11697BD3C2D1CA840676C36BC813DA8
3144655960316053B53F2D714FE86C3D0C94D0A6394D3D1938AF554518A4F8AA6EBD93C48B88342A
64959CB58FCE90D83EF03A90FBC0F9A2833DF0596A579AAC10146CE3A05E552FED06B721831DEB89
F0098A8BC0DBAFA271
```

```
dmq =
0x14F9BC56E03E9C9AD385C043AD1F2BD6B7EE712B3D7C39BB530F1DE4592A6B6FD7A4262E936CAD
253AF9A33A2619E162325983D2D40C165E9EDD6D704BD8D383754B12C79A64221A46F86B5521DD7D
4D9A91CD63E11BF117C18B0EE8F5AC1AA6867F0F73F7DEAA057D7088B38CC0B61E16DFC97572B782
53709F2F524DA56923

crt =
0x7ECD05E781CFC0CE985CE621FDFAF4C49FE82BB374DD401B9F0E9CF79A01F832045FF36FE84CC9
D8D7D65706F7EEEE7E93BC0797D9F93C0B47FDC127CF8F9E2DF4CA1CAC425431499658F93C0FFC30
B48DCACF4B65BEC4ED995AB80A888E630FCE883EB710BA94D17F75B6A2117FCAD957C4678F993B7E
4141300216C6FF6ED1
```

快速加解密算法

可以看出，要只通过基本公式计算 2048bit 长的数的 2048bit长的 次方，再取模。很明显，这个计算量，是连计算机都无法承受的。

我们采用 中国剩余定理（CRT） 和 快速幂 的方法来优化计算：

CRT：

按步骤执行：

$$mp = C^{dmp} \ mod \ p$$

$$mq = C^{dmq} \ mod \ q$$

$$h = (mp - mq) * crt \ mod \ p$$

$$M = h * q + mq$$

得到M

参考代码（python代码简单，C++代码原理相同）：

```python
def fast(c,n,e,d,p,q,dmp,dmq,crt):
    mp  = tool(c,dmp,p)
    mq = tool(c,dmq,q)
    if mp > mq:
        h = mp - mq
    else:
        h = mq - mp
        h = p-h
    h = h * crt
    h = h % p
    h = h * q
    mm = h + mq
    return mm
```

快速幂：

```
def tool(c,cf,n):#计算 c 的 cf 次方 mod n
    t = c
    a= 1
    while(cf!= 0):
        if(cf&1!=0):
            a = a * t
            a = a%n
        cf = cf >>1
        t = t*t
        t = t %n
    return a
```

C++测试：

```
plaintext:abc
En_time:1438 ms EnCode:
0x9984C858B6B53A0776AD7E55EC99720E3A9AC836B2857654749FBD683829C2D6151584853E6D77E7FD68CF75BE7A7CE662F67546A2788D067E6F
B0B1515F968F9A41A5CC727D6949C7BDA19C460987DAF7E61A8F1F27E22FECC829F7E5EB51CBA772023E30D6001861FF8561EB1ECDE5F191479FE6
462101808825E0DB8B49A1FFD6F7AFCC380E9ED43E23EE6284DB7F79D55E9983C8354B69A74898D2344E7CD0489CCD944319763099E614F217C8A4
29D19EB34C6A2C2A0647B3555A6C875936DBF292C675F239CBFB94FA1282EBE7F14555AEA6F1646E0B1D5F8BECD186F52EA2734B2C136094D47959
B676F5B6DCE6DA1A2BE59841C1EC4D71006E92B94C



De_time:64062 ms DeCode:
0x2DAC21430117DFF591B15553CFE09E879AD0CF072D1B5F4A1C878EE30EED5AAC31CCF6F996FA0E1D34BAB26139FB248496A49DACA7E8ADFA3840
BE72C400D7BE016F6BB6BCD02460594355AFF42943D9BE294DB216FD3B02FDC06E8515732C5D1CF2042D1B40BF696FE9B3A2770178E171ED6399F2
DF630F0DA5139C6BDAD50414F334AF1AB23B82F91C50EB4B0A3056425F53D6B3F5D278B85D422B4FAE4E4D69F191CF33AB25578569B1ECD8BA0682
BFFDB2FAE2D6E5DABE578EB4394414938C6B8E92096185406B19B88768013B56E578F809399F60323DB9A152B4610CCFA3D4041D540B893EC11462
F8D15C54E98FC37BF33A514E6BE3BE200616263

all:abc


De_time:403985 ms DeCode:
0x2DAC21430117DFF591B15553CFE09E879AD0CF072D1B5F4A1C878EE30EED5AAC31CCF6F996FA0E1D34BAB26139FB248496A49DACA7E8ADFA3840
BE72C400D7BE016F6BB6BCD02460594355AFF42943D9BE294DB216FD3B02FDC06E8515732C5D1CF2042D1B40BF696FE9B3A2770178E171ED6399F2
DF630F0DA5139C6BDAD50414F334AF1AB23B82F91C50EB4B0A3056425F53D6B3F5D278B85D422B4FAE4E4D69F191CF33AB25578569B1ECD8BA0682
BFFDB2FAE2D6E5DABE578EB4394414938C6B8E92096185406B19B88768013B56E578F809399F60323DB9A152B4610CCFA3D4041D540B893EC11462
F8D15C54E98FC37BF33A514E6BE3BE200616263

all:abc
```

第一段是 填充后 的 “abc”加密输出的数。

第二段是通过CRT 和 快速幂 解密得到的数（即为abc填充后的数 看开头的 00 02 和最后的00 61 62 63 即为 abc），和去除BT PS 和 00 得到的 实际解密结果 abc，所需时间64秒

第三段是仅使用 (n,d) 和 快速幂 解密得到的数（即为abc填充后的数 看开头的 00 02 和最后的00 61 62 63即为 abc），和去除BT PS 和 00 得到的 实际解密结果 abc，所需时间为404秒

补充：是因为我使用的 大数运算库 的效率太低。所以计算时间很长。若使用更优的大数结构，运算速度可以有质的提升，实际使用python相同算法相同数据量下，python 解密仅需要0.009秒左右。😓