CTF竞赛技术 - CRYPTO从入门到放弃

转载

JacobTsang 于 2018-10-20 15:17:38 发布 1960 收藏 11

分类专栏： Information Security 文章标签： CTF

Information Security 专栏收录该内容

49 篇文章 2 订阅

订阅专栏

CTF竞赛技术 - CRYPTO从入门到放弃

CRYPTO是CTF中常见的一种题型，一般包括各种类型的密码学、编码、编程语言加密等知识点，有时候也会包含一些考验选手脑洞的题目。

不常见的密码学知识

文本加密方法

- 栅栏密码(Rail-fence Cipher)
  - 栅栏密码(Rail-fence Cipher)就是把要加密的明文分成N个一组，然后把每组的第1个字符组合，每组第2个字符组合…每组的第N(最后一个分组可能不足N个)个字符组合，最后把他们全部连接起来就是密文，这里以2栏栅栏加密为例。
  - hello justech => hellojustech => hloutc eljseh => hloutceljseh
- 曲路密码(Curve Cipher)
  - 曲路密码(Curve Cipher)是一种换位密码，需要事先双方约定密钥(也就是曲路路径)。
  - 明文： The quick brown fox jumps over the lazy dog 填入5行7列表(事先约定填充的行列数)
  - 加密的回路线(事先约定填充的行列数)
  - 密文： gesfc inpho dtmwu qoury zejre hbxva lookT
- 列移位密码(Columnar Transposition Cipher)
  - 列移位密码(Columnar Transposition Cipher)是一种比较简单，易于实现的换位密码，通过一个简单的规则将明文打乱混合成密文。以明文 The quick brown fox jumps over the lazy dog，密钥 how are u为例：
  - 填入5行7列表(事先约定填充的行列数，如果明文不能填充完表格可以约定使用某个字母进行填充)
  - 密钥： how are u
  - 按how are u在字母表中的出现的先后顺序进行编号，我们就有a为1, e为2，h为3，o为4，r为5，u为6，w为7，所以先写出a列，其次e列，以此类推写出的结果便是密文：
  - 密文： qoury inpho Tkool hbxva uwmtd cfseg erjez
- 埃特巴什码(Atbash Cipher)
  - 埃特巴什码(Atbash Cipher)是一种以字母倒序排列作为特殊密钥的替换加密，也就是下面的对应关系：
  - ABCDEFGHIJKLMNOPQRSTUVWXYZ => ZYXWVUTSRQPONMLKJIHGFEDCBA
  - 明文： the quick brown fox jumps over the lazy dog
  - 密文： gsv jfrxp yildm ulc qfnkh levi gsv ozab wlt
- 凯撒密码(Caesar Cipher)

    - 凯撒密码(Caesar Cipher或称恺撒加密、恺撒变换、变换加密、位移加密)是一种替换加密，明文中的所有字母都在字母表上向后（或向前）按照一个固定数目进行偏移后被替换成密文。例，当偏移量是3的时候，所有的字母A将被替换成D，B变成E，以此类推
    - 明文： The quick brown fox jumps over the lazy dog
    - 偏移量：1
    - 密文： Uif rvjdl cspxo gpy kvnqt pwfs uif mbaz eph
- ROT5/13/18/47
    - ROT5/13/18/47是一种简单的码元位置顺序替换暗码。此类编码具有可逆性，可以自我解密，主要用于应对快速浏览，或者是机器的读取。
    - ROT5 是 rotate by 5 places 的简写，意思是旋转5个位置，其它皆同。
    - ROT5：只对数字进行编码，用当前数字往前数的第5个数字替换当前数字，例如当前为0，编码后变成5，当前为1，编码后变成6，以此类推顺序循环。
    - ROT13：只对字母进行编码，用当前字母往前数的第13个字母替换当前字母，例如当前为A，编码后变成N，当前为B，编码后变成O，以此类推顺序循环。
    - ROT18：这是一个异类，本来没有，它是将ROT5和ROT13组合在一起，为了好称呼，将其命名为ROT18。
    - ROT47：对数字、字母、常用符号进行编码，按照它们的ASCII值进行位置替换，用当前字符ASCII值往前数的第47位对应字符替换当前字符，例如当前为小写字母z，编码后变成大写字母K，当前为数字0，编码后变成符号_。用于ROT47编码的字符其ASCII值范围是33－126，具体可参考ASCII编码。
    - 以rot13以例：
        - 明文： the quick brown fox jumps over the lazy dog
        - 密文： gur dhvpx oebja sbk whzcf bire gur ynml qbt
- 简单换位密码(Simple Substitution Cipher)
    - 简单换位密码(Simple Substitution Cipher)加密方式是以每个明文字母被与之唯一对应且不同的字母替换的方式实现的，它不同于恺撒密码，因为密码字母表的字母不是简单的移位，而是完全是混乱的。 比如：
        - 明文字母 : abcdefghijklmnopqrstuvwxyz
        - 密文字母 : phqgiumeaylnofdxjkrcvstzwb
        - 明文： the quick brown fox jumps over the lazy dog
        - 密文： cei jvaql hkdtf udz yvoxr dsik cei npbw gdm
- 希尔密码(Hill Cipher)
    - 希尔密码(Hill Cipher)是基于线性代数多重代换密码，由Lester S. Hill在1929年发明。每个字母转换成26进制数字：A=0, B=1, C=2…Z=25一串字母当成n维向量，跟一个n×n的矩阵相乘，再将得出的结果MOD26。
    - 加密
        - 明文 ACT
        - 明文矩阵
        - 加密密钥： GYBNQKURP
        - 密钥矩阵
        - 加密过程
        - 得到密文 FIN
    - 解密
        - 密文 FIN
        - 计算密钥矩阵的逆矩阵

- 计算出明文矩阵的逆矩阵
    - 解密过程
    - 得到明文 ACT
- 猪圈密码(Pigpen Cipher)
    - 猪圈密码(Pigpen Cipher或称九宫格密码、朱高密码、共济会密码或共济会员密码)，是一种以格子为基础的简单替代式密码。
    - 明文： THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
    - 密文：
    - 一些变种
        - 圣堂武士密码(Templar Cipher)是共济会的“猪圈密码”的一个变种，一直被共济会圣殿骑士用。
- 波利比奥斯方阵密码（Polybius Square Cipher)
    - 波利比奥斯方阵密码（Polybius Square Cipher或称波利比奥斯棋盘）是棋盘密码的一种，是利用波利比奥斯方阵进行加密的密码方式，简单的来说就是把字母排列好，用坐标(行列)的形式表现出来。字母是密文，明文便是字母的坐标。
    - 常见的排布方式
    - 加密实例：
        - 明文： THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
        - 密文： 442315 4145241325 1242345233 213453 2445323543 442315 31115554 143422
- 夏多密码(曲折加密)
    - 夏多密码是作者麦克斯韦·格兰特在中篇小说《死亡之链》塑造夏多这一英雄人物中所自创的密码
    - 在以上所示的字母表密钥的底部，列有四个附加符号1，2，3，4.他们可以放在密文中的任何地方。每个附加符号指示，如何转动写有密文的纸张，再进行后续的加密或解密操作，直到出现另一个附加符号。可以把每个附加符号中的那根线看作是指示针，它指示了纸张的上端朝上，朝右，朝下，朝左。比如说：如果出现符号3，那么纸张就应该转动180度，使其上端朝下； 符号2表示纸张上端朝右，依次类推。
    - 源文本： I AM IN DANGER SEND HELP(我有危险，速来增援)
- 普莱菲尔密码(Playfair Cipher)
    - 普莱菲尔密码(Playfair Cipher)是第一种用于实际的双字替换密码，用双字加密取代了简单代换密码的单字加密，很明显这样使得密文更难破译，因为使用简单替换密码的频率分析基本没有什么作用，虽然频率分析，通常仍然可以进行，但是有25×25=625种可能而不是25种可能，可以分为三个步骤，即编制密码表、整理明文、编写译文。
    - 以明文：THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 和密钥 CULTURE 为例：
        - 编制密码表
            - 1.整理密钥字母 C U L T U R E，去掉后面重复的字母得到： C U L T R E
            - 2.用上一步得到的字母自上而下来填补5乘5方表的纵列（也可横排），之后的空白按照相同的顺序用字母表中剩余的字母依次填补完整，得到如下的方格:
            - 整理密钥字母时，如果出现"Z"，则需要去除，因为在英文里"Z"的使用频率最低，相应的如果是德文，则需将"I"与"J"当作一个字母来看待，而法语则去掉"W"或"K"。
        - 整理明文
            - 要遵循的原则是“两个一组”，得到是若干个两两成对的字母段，用到的是明文 THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 与字母" X "：
            - 1.将明文两两一组按顺序排开，得到： TH EQ UI CK BR OW NF OX JU MP SO VE RT HE LA ZY DO G
            - 2.对于末尾的单个字母要加上一个" X "使之成对： TH EQ UI CK BR OW NF OX JU MP SO VE RT HE LA ZY DO GX
            - 需要注意的要点：对于相连字母相同者，每个后面都需要加" X "，例如 TOMORROW ，需要写成：TO MO

RX RX OW

- 编写密文
  - 对于每个字母对，要严格遵循如下的原则
  - 1.如果两个字母在同一行则要用它右邻的字母替换，如果已在最右边，则用该行最左边的替换，如明文为" CE “，依据上表，应替换为” EG "
  - 2.如果两个字母在同一列则要用它下边的字母替换，如果已在最下边，则用该行最上边的替换，如明文为" OQ “，依据上表，应替换为” PS "
  - 3.如果两个字母在不同的行或列，则应在密码表中找两个字母使四个字母组成一个矩形，明文占据两个顶点，需用另外两个顶点的字母替换，如明文为" HX “，可以替换为” WIJ “或” IJW "（下面的例子将按照横向替换原则即同行优先）。
  - 按照上述原则，将明文 TH EQ UI CK BR OW NF OX JU MP SO VE RT HE LA ZY DO GX 加以转换得到 KU ND LH GT LF WU ES PW LH SIJ NP CG CR AG BU VZ QA IJV （/表示或者，不过一般用I不用J，所以分析密文时你看25个字母都有而只差一个字母没有用到可以考虑一下这种加密方式）将得到的字母改为大写并五个一组列好，得到密文 KUNDL HGTLF WUESP WLHSI NPCGC RAGBU VZQAI V
- Python库

```
>>>from pycipher import Playfair
>>>Playfair('CULTREABDFGHIKMNOPQSVWXYZ').encipher('THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG')
'UKDNLHTGFLWUSEPWHLISNPCGCRGAUBVZAQIV'
>>>Playfair('CULTREABDFGHIKMNOPQSVWXYZ').decipher('UKDNLHTGFLWUSEPWHLISNPCGCRGAUBVZAQIV')
'THEQUICKBROWNFOXIUMPSOVERTHELAZYDOGX'
```

- 维吉尼亚密码(Vigenère Cipher)
  - 维吉尼亚密码(Vigenère Cipher)是在单一恺撒密码的基础上扩展出多表代换密码，根据密钥(当密钥长度小于明文长度时可以循环使用)来决定用哪一行的密表来进行替换，以此来对抗字频统计
  - 明文： THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
  - 密钥(循环使用，密钥越长相对破解难度越大)： CULTURE
  - 加密过程：如果第一行为明文字母，第一列为密钥字母，那么明文字母’T’列和密钥字母’C’行的交点就是密文字母’V’，以此类推。
  - 密文： VBP JOZGM VCHQE JQR UNGGW QPPK NYI NUKR XFK
  - Python库

```
>>>from pycipher import Vigenere
>>>Vigenere('CULTURE').encipher('THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG')
'VBPJOZGMVCHQEJQRUNGGWQPPKNYINUKRXFK'
>>>Vigenere('CULTURE').decipher('VBPJOZGMVCHQEJQRUNGGWQPPKNYINUKRXFK')
'THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG'
```

- 自动密钥密码(Autokey Cipher)
  - 自动密钥密码(Autokey Cipher)是多表替换密码，与维吉尼亚密码密切相关，但使用不同的方法生成密钥，通常来说要比维吉尼亚密码更安全。自动密钥密码主要有两种，关键词自动密钥密码和原文自动密钥密码
  - 明文： THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
  - 关键词： CULTURE
  - 自动生成密钥： CULTURE THE QUICK BROWN FOX JUMPS OVER THE
  - 接下来的加密过程和维吉尼亚密码类似，从密表可得：
  - 密文： VBP JOZGD IVEQV HYY AIICX CSNL FWW ZVDP WVK

- Python库

```
>>>from pycipher import Autokey
>>>Autokey('CULTURE').encipher('THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG')
'VBPJOZGDIVEQVHYYAIICXCSNLFWWZVDPWVK'
>>>Autokey('CULTURE').decipher('VBPJOZGDIVEQVHYYAIICXCSNLFWWZVDPWVK')
'THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG'
```

- 博福特密码(Beaufort Cipher)
  - 博福特密码(Beaufort Cipher)，是一种类似于维吉尼亚密码的代换密码，由弗朗西斯·蒲福(Francis Beaufort)发明。它最知名的应用是Hagelin M-209密码机。博福特密码属于对等加密，即加密演算法与解密演算法相同。
  - 明文： THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
  - 密钥(循环使用，密钥越长相对破解难度越大)： CULTURE
  - 加密过程：如果第一行为明文字母，第一列为密文字母，那么沿明文字母'T'列出现密钥字母'C'的行号就是密文字母'J'，以此类推。
  - 密文： JNH DAJCS TUFYE ZOX CZICM OZHC BKA RUMV RDY
  - Python库

```
>>>from pycipher import Beaufort
>>>Beaufort('CULTURE').encipher('THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG')
'JNHDAJCSTUFYEZOXCZICMOZHCBKARUMVRDY'
>>>Beaufort('CULTURE').decipher('JNHDAJCSTUFYEZOXCZICMOZHCBKARUMVRDY')
'THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG'
```

- 滚动密钥密码(Running Key Cipher)
  - 滚动密钥密码(Running Key Cipher)和维吉尼亚密码有着相同的加密机制，区别是密钥的选取，维吉尼亚使用的密钥简短，而且重复循环使用，与之相反，滚动密钥密码使用很长的密钥，比如引用一本书作为密钥。这样做的目的是不重复循环使用密钥，使密文更难破译，尽管如此，滚动密钥密码还是可以被攻破，因为有关于密钥和明文的统计分析模式可供利用，如果滚动密钥密码使用统计上的随机密钥来源，那么理论上是不可破译的，因为任何可能都可以成为密钥，并且所有的可能性都是相等的。
  - 明文： THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
  - 密钥：选取C语言编程(1978版)第63页第1行"errors can occur in several places. A label has…"，去掉非字母部分作为密钥(实际选取的密钥很长，长度至少不小于明文长度)。
  - 加密过程：加密过程和维吉尼亚密码加密过程相同
  - 密文: XYV ELAEK OFQYH WWK BYHTJ OGTC TJI DAK YESR
- Porta密码(Porta Cipher)
  - Porta密码(Porta Cipher)是一个由意大利那不勒斯的医生Giovanni Battista della Porta发明的多表代换密码，Porta密码具有加密解密过程的是相同的特点。
  - 密码表
  -
  - 明文： THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
  - 密钥(循环使用，密钥越长相对破解难度越大)： CULTURE
  - 加密过程：明文字母'T'列与密钥字母'C'行交点就是密文字母'F',以此类推。
  - 密文： FRW HKQRY YMFMF UAA OLWHD ALWI JPT ZXHC NGV
- 同音替换密码(Homophonic Substitution Cipher)

- 同音替换密码(Homophonic Substitution Cipher)
  - 同音替换密码(Homophonic Substitution Cipher)是单字母可以被其他几种密文字母同时替换的密码，通常要比标准替换密码破解更加困难，破解标准替换密码最简单的方法就是分析字母出现频率，通常在英语中字母'E'(或'T')出现的频率是最高的，如果我们允许字母'E'可以同时被3种不同字符代替，那么就不能还是以普通字母的频率来分析破解，如果允许可代替字符越多，那么密文就会更难破译。
  - 常见代换规则表：
  - 明文: THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
  - 密文(其中一种)： 6CZ KOVST XJ0MA EQY lOGL4 0W1J UC7 P9NB F0H
- 仿射密码(Affine Cipher)
  - 仿射密码(Affine Cipher)是一种单表代换密码，字母表中的每个字母相应的值使用一个简单的数学函数映射到对应的数值，再把对应数值转换成字母。这个公式意味着每个字母加密都会返回一个相同的字母，意义着这种加密方式本质上是一种标准替代密码。因此，它具有所有替代密码的弱点。每一个字母都是通过函数（ax + b）mod m加密，其中B是位移量，为了保证仿射密码的可逆性，a和m需要满足gcd(a , m)=1，一般m为设置为26。
  - 常见字母对应关系
  - 以E(x) = (5x + 8) mod 26函数为例
  - 解密过程
  - 解密结果
  - 用Python实现
- 培根密码(Baconian Cipher)
  - 培根密码(Baconian Cipher)是一种替换密码，每个明文字母被一个由5字符组成的序列替换，最初的加密方式就是由'A'和'B'组成序列替换明文(所以你当然也可以用别的字母)，比如字母'D'替换成"aaabb"，以下是全部的对应关系(另一种对于关系是每个字母都有唯一对应序列，I和J与U/V各自都有不同对应序列)：

```
A = aaaaa  I/J = abaaa  R = baaaa
B = aaaab  K = abaab    S = baaab
C = aaaba  L = ababa    T = baaba
D = aaabb  M = ababb    U/V = baabb
E = aabaa  N = abbaa    W = babaa
F = aabab  O = abbab    X = babab
G = aabba  P = abbba    Y = babba
H = aabbb  Q = abbbb    Z = babbb
```

  - 明文： T H E F O X
  - 密文： baaba aabbb aabaa aabab abbab babab
- ADFGX和ADFGVX密码(ADFG/VX Cipher)
  - ADFGX密码(ADFGX Cipher)是结合了改良过的Polybius方格替代密码与单行换位密码的矩阵加密密码，使用了5个合理的密文字母：A，D，F，G，X，这些字母之所以这样选择是因为当转译成摩尔斯电码(ADFGX密码是德国军队在一战发明使用的密码)不易混淆，目的是尽可能减少转译过程的操作错误。
    - 加密矩阵示例
    -
    - 明文：THE QUICK BROWN FOX
    - 矩阵加密 => XF AD DA AF XD XG GA FG XA FX DX GX DG FA DX FF
    - 列移位密钥： how are u

-
- 密文： DXADF AGXF XFFXD FXGGX DGFG AADA ADXXF
- Python库

```
>>>from pycipher import ADFGX
>>>a = ADFGX('phqgmeaynofdxkrcvszwbutil','HOWAREU')
>>>a.encipher('THE QUICK BROWN FOX')
'DXADFAGXFXFFXDFXGGXDGFGAADAADXXF'
>>>a.decipher('DXADFAGXFXFFXDFXGGXDGFGAADAADXXF')
'THEQUICKBROWNFOX'
```

- ADFGVX密码实际上就是ADFGX密码的扩充升级版，一样具有ADFGX密码相同的特点，加密过程也类似，不同的是密文字母增加了V，使得可以再使用10数字来替换明文。
  - 加密矩阵
  - Python库

```
>>>from pycipher import ADFGVX
>>>a = ADFGVX('ph0qg64mea1yl2nofdxkr3cvs5zw7bj9uti8','HOWAREU')
>>>a.encipher('THE QUICK BROWN FOX')
'DXXFAFGFFXGGGFGXDVGDVGFAVFVAFVGG'
>>>a.decipher('DXXFAFGFFXGGGFGXDVGDVGFAVFVAFVGG')
'THEQUICKBROWNFOX'
```

- 双密码(Bifid Cipher)
  - 双密码(Bifid Cipher)结合了波利比奥斯方阵换位密码，并采用分级实现扩散，这里的“双”是指用2个密钥进行加密。双密码是由法国Felix Delastelle发明，除此之外Felix Delastelle还发明了三分密码(Trifid Cipher)，四方密码(Four-Square Cipher)。还有一个 两方密码 (Two-Square)与四方密码类似， 共轭矩阵双密码 (Conjugated Matrix Bifid Cipher)也是双密码的变种。
  - 示例密阵:

```
1 2 3 4 5
1| p h q g m
2| e a y l n
3| o f d x k
4| r c v s z
5| w b u t i/j
```

  - 明文: THE QUICK BROWN FOX
  - 经过密阵转换：
  - 行: 512 15543 54352 333
  - 列: 421 33525 21115 214
  - 分组:
    - 51215 54354 35233 3
    - 42133 52521 11521 4
  - 合并：
    - 5121542133 5435452521 3523311521 34
  - 经过密阵转换后密文: WETED TKZNE KYOME X
  - Python库

```
>>>from pycipher import
```

```
>>>from pycipher import
>>>Bifid('phqgmeaylnofdxkrcvszwbuti',5).encipher('THE QUICK BROWN FOX')
'WETEDTKZNEKYOMEX'
>>>Bifid('phqgmeaylnofdxkrcvszwbuti',5).decipher('WETEDTKZNEKYOMEX')
'THEQUICKBROWNFOX'
```

- 三分密码(Trifid Cipher)
  - 三分密码(Trifid Cipher)结合换位和替换，三分密码与双密码非常相似，差别之处就是用除了3×3×3的密阵代替5×5密阵。
  - 示例密阵:
  - 密阵顺序 = EPSDUCVWYM.ZLKXNBTFGORIJHAQ

```
方阵 1      方阵 2      方阵 3
1 2 3      1 2 3      1 2 3
1 E P S    1 M . Z    1 F G O
2 D U C    2 L K X    2 R I J
3 V W Y    3 N B T    3 H A Q
```

  - 明文: THE QUICK BROWN FOX.
  - 经过密阵转换：

```
T H E Q U I C K B R O W N F O X .
2 3 1 3 1 3 1 2 2 3 3 1 2 3 3 2 2
3 3 1 3 2 2 2 2 3 2 1 3 3 1 1 2 1
3 1 1 3 2 2 3 2 2 1 3 2 1 1 3 3 2
```

  - T(233)表示T在第一个方阵第三行第三列的位置
  - 分组(分组密钥以5为例):

```
THEQU ICKBR OWNFO X.
23131 31223 31233 22
33132 22232 13311 21
31132 23221 32113 32
```

  - 合并 => 23131 33132 31132 31223 22232 23221 31233 13311 32113 22 21 32
  - 在经过密阵转换后密文
  - 2313133132311323122322322322132131233133113211322213 2
  - N O O N W G B X X L G H H W S K W
- 四方密码(Four-Square Cipher)
  - 四方密码(Four-Square Cipher)是类似普莱菲尔密码双字母加密密码，这样使加密效果强于其他替换密码，因为频率分析变得更加困难了。四方密码使用4个预先设置的5×5字母矩阵，每个矩阵包括25个字母，通常字母'j'被融入到'i'中(维基百科上说'q'被忽略，不过这不重要，因为'q'和'j'都是很少出现的字母)，通常左上和右下矩阵式是标准字母排序明文矩阵，右上和左下矩阵是打乱顺序的密钥矩阵。
  - 示例矩阵
  - 明文： THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
  - 整理明文(分组不够时用'X'填充)： TH EQ UI CK BR OW NF OX JU MP SO VE RT HE LA ZY DO GX
  - 加密过程：分别在明文矩阵中找到'TH'，分别找到他们在右上矩阵有左下矩阵的交点字母'ES'就是密文，以此类推。
  - 密文： ESZWQAFHGTDKWHRKUENYQOLMQTUNWMBPTGHQ
  - Python库

```
```

```
>>>from pycipher import Foursquare
>>>fs = Foursquare('zgptfoihmuwdrcnykeqaxvsbl','mfnbdcrhsaxyogvituewlqzkp')
>>>fs.encipher('THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG')
'ESZWQAFHGTDKWHRKUENYQOLMQTUNWMBPTGHQ'
>>>fs.decipher('ESZWQAFHGTDKWHRKUENYQOLMQTUNWMBPTGHQ')
'THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG'
```

- 棋盘密码（Checkerboard Cipher)
  - 棋盘密码（Checkerboard Cipher)是使用一个波利比奥斯方阵和两个密钥作为密阵的替换密码，通常在波利比奥斯方阵中J字母往往被包含在I字母中。
  - 示例密阵：

```
Q  U  I  C  K
-------------
B |K  N I/J G  H
R |P  Q  R  S  T
O |O  Y  Z  U  A
W |M  X  W  V  B
N |L  F  E  D  C
```

  - 经过密阵替换:
    - 明文:T H E Q U I C K B R O W N F O X
    - 密文:RK BK RU OC OC BI NK BQ WK RI OQ WI BU NU OQ WU
- 跨棋盘密码(Straddle Checkerboard Cipher)
  - 跨棋盘密码(Straddle Checkerboard Cipher)是一种替换密码，当这种密码在结合其他加密方式，加密效果会更好。
  - 棋盘示例(选择3和7作为变换):

```
0 1 2 3 4 5 6 7 8 9
f k m   c p d   y e
3: h b i g q r o s a z
7: l u t j n w v x
```

  - 明文: T H E Q U I C K B R O W N F O X
  - 经过加密棋盘替换得到密文: 72 30 9 34 71 32 4 1 31 35 36 75 74 0 36 77
  - 还可以继续用其他的加密方式在对跨棋盘密码加密出的结果再进行加密:
  - 示例变换密钥:83729

```
837298372983729837298372983729837
+723093471324131353675740367 7
-----------------------------
55028174320785018086301224 04
```

  - 在经过棋盘转换后:

```
55028174320785018086301224 04
ppfmyk n if  pfkyfyd hkmmcfc
```

  - 最终得到密文: ppfmyk n if pfkyfyd hkmmcfc
- 分组摩尔斯替换密码(Fractionated Morse Cipher)
  - 分组摩尔斯替换密码(Fractionated Morse Cipher)首先把明文转换为莫尔斯电码，不过每个字母之间用 x 分开，每个单词用 xx 分开。然后使用密钥生成一个替换密表，这个密表包含所有 . - x 组合的情况(因为不会出现 xxx 的情况，所以一共26种组合)。

- 密钥: MORSECODE
- 密表:

```
MORSECDABFGHIJKLNPQTUVWXYZ
.....----------XXXXXXXX
...---XXX...---XXX...---XX
.-X.-X.-X.-X.-X.-X.-X.-X.-
```

- 说明:密表下半部分是固定的，密表的安全性以及加密效果主要取决于使用的密钥。
- 明文： THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
- (类似)摩尔斯电码:

```
-x....x.xx--.-x..-x..x-.-x-.-xx-...x.-.x---x.--x-.xx..-.x---x-..-xx.---x..- --x.--.x...xx---x...-x.x.-.
xx-x....x.xx.-..x.-x--..x-.--xx-..x---x--.
```

- 说明:明文在转换为(类似)摩尔斯电码后进行每3个字符分组，再进行密表的查表。
- 密文(经过密表替换): LMUWC OQVHG ZMTAK EVYSW NOYJQ NLIQB JQCDH XMDYF TWRGP FWNH

- Bazeries密码(Bazeries Cipher)
  - Bazeries密码(Bazeries Cipher)是换位密码和替换密码的组合，使用两个波利比奥斯方阵，一个明文字母方阵，使用一个随机的数字(一般小于1000000)的生成一个密钥矩阵同时作为第一轮明文划分分组，比如2333这个数字翻译为英文便是TWO THOUSAND THREE HUNDRED THIRTY THREE,从第一个字母T开始选取不重复的字母，之后再从字母表中按序选取没有出现的字母组成密钥矩阵。
  - 明文: THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
  - 随机数字: 2333
  - 明文矩阵:

```
A  F  L  Q  V
B  G  M  R  W
C  H  N  S  X
D I/J O  T  Y
E  K  P  U  Z
```

  - 示例密钥矩阵:

```
T  W  O  H  U
S  A  N  D  R
E I/J Y  B  C
F  G  K  L  M
P  Q  V  X  Z
```

  - 明文分组:

```
2   3   3   3   2   3   3   3  2   3   3  3
TH EQU ICK BRO WN FOX JUM PSO VE RTH ELA ZYD OG
```

  - 分组明文反序:

```
HT UQE KCI ORB WN XOF MUJ OSP EV EHT ALE DYZ GO
```

  - 使用密钥矩阵替换:

```
IL XHP QEG KDS YR CKW NXG KBV PU ILD TOP FMZ AK
```

  - (此加以在明文矩阵对应到密钥矩阵的位置就是此)

- (比如 H 在明文矩阵对应到密钥矩阵的位置就是 1)

- Digrafid密码(Digrafid Cipher)
  - Digrafid密码(Digrafid Cipher)使用两个密钥生成分别生成类似波利比奥斯方阵的3x9方格的密表。，主要有3分组和4分组两类。
  - 第一个方阵密钥: digrafid
  - 第二个方阵密钥: cipher
  - 密表：

```
1 2 3 4 5 6 7 8 9
D I G R A F D B C 1 2 3
E H J L M N O P Q 4 5 6
S T U V W X Y Z # 7 8 9
              c f s 1
              i g t 2
              p j u 3
              h k v 4
              e l w 5
              r m x 6
              a n y 7
              b o z 8
              d q # 9
```

  - 明文: THE QUICK BROWN FOX
  - 密表转换(以4分组为例):

```
Th Eq Ui Ck   Br Ow Nf Ox
2  1  3  9    8  7  6  7
7  5  7  2    1  6  5  6
4  9  2  4    6  5  1  6
```

  - 说明:T在第一矩阵第2列，h在第二矩阵第4行，T所在的行与h所在的列相交的位置数字为7，所以Th表示为274。
  - 转换密文:

```
213 975 724 924   876 716 566 516
Ip  #e  Dk  Ck    Zr  Dr  Mx  Ar
```

- 格朗普雷密码(Grandpré Cipher)
  - 格朗普雷密码(Grandpré Cipher)是替换密码的一种，一般使用8个8字母的单词横向填充8x8方阵，且第一列为一个单词，并且在方阵中26个字母都必须出现一次以上。
  - 示例密阵:
  -

```
明文:T  H  E  Q  U  I  C  K  B  R  O  W  N  F  O
密文:84 27 82 41 51 66 31 36 15 71 67 73 52 34 67

说明：明文中的字母在密阵位置可能不止一个，所以加密结果可能有多种，但是不影响解密。密阵还有6x6，7x7，9x9,10x10几种。显然密阵越大每个字母被替换的情况就可能越多，那么加密效果就更好。
```

- 比尔密码(Beale ciphers)
  - 比尔密码(Beale ciphers)有三份密码，当然这里说的是已被破解第二份，是一种类似书密码的替换密码。
  - 以第二密码为例，每一个数字代表美国《独立宣言》的文本中的第几个词的首字母，如1代表第1个词的首字母“w”，2代

表第2个词首字母“i”。解密后的文字如下：

    - I have deposited in the county of Bedford…

- 键盘密码(Keyboard Cipher)
    - 手机九宫格密码
    - 电脑五笔、全拼打字法密码等

常见的密码算法

常见的编码知识

- ASCII编码
- Base64/32/16编码
    - base64、base32、base16可以分别编码转化8位字节为6位、5位、4位。16,32,64分别表示用多少个字符来编码，这里我注重介绍base64。Base64常用于在通常处理文本数据的场合，表示、传输、存储一些二进制数据。包括MIME的email，email via MIME,在XML中存储复杂数据。
    - 编码原理：Base64编码要求把3个8位字节转化为4个6位的字节，之后在6位的前面补两个0，形成8位一个字节的形式，6位2进制能表示的最大数是2的6次方是64，这也是为什么是64个字符(A-Z,a-z，0-9，+，/这64个编码字符，=号不属于编码字符，而是填充字符)的原因，这样就需要一张映射表，如下：

    - 举个例子(base64)：
    - 源文本：T h e
    - 对应ascii码:84 104 101
    - 8位binary：01010100 01101000 01100101
    - 6位binary：010101 000110 100001 100101
    - 高位补0：000010101 00000110 00100001 00100101
    - 对应ascii码：21 6 33 37
    - 查表：V G h l
- shellcode编码
    - 十六进制

```
\x54\x68\x65\x7f\x71\x75\x69\x63\x6b\x7f\x62\x72\x6f\x77\x6e\x7f\x66\x6f\x78\x7f\x6a\x75\x6d\x70\x73\x7f
\x6f\x76\x65\x72\x7f\x74\x68\x65\x7f\x6c\x61\x7a\x79\x7f\x64\x6f\x67
```

- Quoted-printable编码
    - 多用途互联网邮件扩展（MIME）一种实现方式。有时候我们可以邮件头里面能够看到这样的编码

```
=E6=95=8F=E6=8D=B7=E7=9A=84=E6=A3=95=E8=89=B2=E7=8B=90=E7=8B=B8=E8=B7=B3=E8
=BF=87=E4=BA=86=E6=87=92=E6=83=B0=E7=9A=84=E7=8B=97
```

- XXencode编码
    - XXencode将输入文本以每三个字节为单位进行编码。如果最后剩下的资料少于三个字节，不够的部份用零补齐。这三个字节共有24个Bit，以6bit为单位分为4个组，每个组以十进制来表示所出现的数值只会落在0到63之间。以所对应值的位置字符代替。它所选择的可打印字符是：
    ±0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz，一共64个字符。跟base64打印字

符相比，就是UUencode多一个“-” 字符，少一个”/” 字符。

-
- 源文本： The quick brown fox jumps over the lazy dog
- 编码后： hJ4VZ653pOKBf647mPrRi64NjS0-eRKpkQm-jRaJm65FcNG-gMLdt64FjNkc+

- UUencode编码
  - UUencode是一种二进制到文字的编码，最早在unix 邮件系统中使用，全称：Unix-to-Unix encoding，UUencode将输入文本以每三个字节为单位进行编码，如果最后剩下的资料少于三个字节，不够的部份用零补齐。三个字节共有24个Bit，以6-bit为单位分为4个组，每个组以十进制来表示所出现的字节的数值。这个数值只会落在0到63之间。然后将每个数加上32，所产生的结果刚好落在ASCII字符集中可打印字符（32-空白…95-底线）的范围之中。
  - 源文本： The quick brown fox jumps over the lazy dog
  - 编码后： M5&AE('%U:6-K(&)R;W=N(&9O>"!J=6UP<R!O=F5R('1H92!L87IY(&1O9PH*
- URL编码
  - url编码又叫百分号编码，是统一资源定位(URL)编码方式。URL地址（常说网址）规定了常用地数字，字母可以直接使用，另外一批作为特殊用户字符也可以直接用（/,:@等），剩下的其它所有字符必须通过%xx编码处理。 现在已经成为一种规范了，基本所有程序语言都有这种编码，如js：有encodeURI、encodeURIComponent，PHP有 urlencode、urldecode等。编码方法很简单，在该字节ascii码的的16进制字符前面加%. 如 空格字符，ascii码是32，对应16进制是’20’，那么urlencode编码结果是:%20。
  - 源文本： The quick brown fox jumps over the lazy dog
  - 编码后：

```
%54%68%65%20%71%75%69%63%6b%20%62%72%6f%77%6e%20%66%6f%78%20%6a%75%6d%70%73%20%6f%76%65%72%20%74%68%65%20%6c%61%7a%79%20%64%6f%67
```

- Unicode编码
  - Unicode编码有以下四种编码方式：
  - 原文本： The
  - &#x [Hex]： The
  - &# [Decimal]： The
  - \U [Hex]： \U0054\U0068\U0065
  - \U+ [Hex]： \U+0054\U+0068\U+0065
- Escape/Unescape编码
  - Escape/Unescape加密解码/编码解码,又叫%u编码，采用UTF-16BE模式， Escape编码/加密,就是字符对应UTF-16 16进制表示方式前面加%u。Unescape解码/解密，就是去掉"%u"后，将16进制字符还原后，由utf-16转码到自己目标字符。如：字符“中”，UTF-16BE是：“6d93”，因此Escape是“%u6d93”。
  - 原文本： The
  - 编码后： %u0054%u0068%u0065
- HTML实体编码

-
- 完整手册
- http://www.w3school.com.cn/tags/html_ref_entities.html

- 敲击码(Tap code)
  - 敲击码(Tap code)是一种以非常简单的方式对文本信息进行编码的方法。因该编码对信息通过使用一系列的点击声音来编码而命名，敲击码是基于5×5方格波利比奥斯方阵来实现的，不同点是是用K字母被整合到C中。
  - 敲击码表:

```
1  2  3  4  5
1 A  B C/K D  E
2 F  G  H  I  J
3 L  M  N  O  P
4 Q  R  S  T  U
5 V  W  X  Y  Z
```

- 莫尔斯电码(Morse Code)
  - 摩尔斯电码(Morse Code)是由美国人萨缪尔·摩尔斯在1836年发明的一种时通时断的且通过不同的排列顺序来表达不同英文字母、数字和标点符号的信号代码，摩尔斯电码主要由以下5种它的代码组成：
    - 点（.）
    - 划（-）
    - 每个字符间短的停顿（通常用空格表示停顿）
    - 每个词之间中等的停顿（通常用 / 划分）
    - 以及句子之间长的停顿
  - 摩尔斯电码字母和数字对应表：

```
A  .-    N  -.    .  .-.-.-  +  .-.-.    1  .----
B  -...  O  ---   ,  --..--  _  ..--.-   2  ..---
C  -.-.  P  .--.  :  ---...  $  ...-..-  3  ...--
D  -..   Q  --.-  "  .-..-.  &  .-...    4  ....-
E  .     R  .-.   '  .----.  /  -..-.    5  .....
F  ..-.  S  ...   !  -.-.--             6  -....
G  --.   T  -     ?  ..--..             7  --...
H  ....  U  ..-   @  .--.-.             8  ---..
I  ..    V  ...-  -  -....-             9  ----.
J  .---  W  .--   ;  -.-.-.             0  -----
K  -.-   X  -..-  (  -.--.
L  .-..  Y  -.--  )  -.--.-
M  --    Z  --..  =  -...-
```

  - 原文本: THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
  - 编码后:

```
- .... . / --.- ..- .. -.-. -.- / -... .-. --- .-- -. / ..-. --- -..- / .--- ..- -- .--. ... / --- ...-
. .-. / - .... . / .-.. .- --.. -.-- / -.. --- --.
```

常见的编程语言加密

asp混淆加密

- asp混天绫

php混淆加密

- http://www.zhaoyuanma.com/phpencode.html

css/js混淆加密

- jjencode将JS代码转换成只有符号的字符串，类似于rrencode，介绍的 PPT，aaencode可以将JS代码转换成常用的网络表情，也就是我们说的颜文字js加密。
- alert(“Hello, world”)

```
ﾟωﾟ/= / ｀m´) ﾉ ~┻━┻   //*´▽｀*/ ['_']; o=(ﾟｰﾟ)  =_=3; c=(ﾟΘﾟ) =(ﾟｰﾟ)-(ﾟｰﾟ); (ﾟДﾟ) =(ﾟΘﾟ)= (o^_^o)/ (o^_
^o);(ﾟДﾟ)={ﾟΘﾟ: '_' ,ﾟωﾟﾉ : ((ﾟωﾟ/==3) +'_') [ﾟΘﾟ] ,ﾟｰﾟﾉ :(ﾟωﾟ/+ '_')[o^_^o -(ﾟΘﾟ)] ,ﾟДﾟ/:((ﾟｰﾟ==3) +'_')[ﾟ
ｰﾟ] }; (ﾟДﾟ) [ﾟΘﾟ] =((ﾟωﾟ/==3) +'_') [c^_^o];(ﾟДﾟ) ['c'] = ((ﾟДﾟ)+'_') [ (ﾟｰﾟ)+(ﾟｰﾟ)-(ﾟΘﾟ) ];(ﾟДﾟ) ['o'] =
 ((ﾟДﾟ)+'_') [ﾟΘﾟ];(ﾟoﾟ)=(ﾟДﾟ) ['c']+(ﾟДﾟ) ['o']+(ﾟωﾟﾉ +'_')[ﾟΘﾟ]+ ((ﾟωﾟ/==3) +'_') [ﾟｰﾟ] + ((ﾟДﾟ) +'_') [
(ﾟｰﾟ)+(ﾟｰﾟ)]+ ((ﾟｰﾟ==3) +'_') [ﾟΘﾟ]+((ﾟｰﾟ==3) +'_') [(ﾟｰﾟ) - (ﾟΘﾟ)]+(ﾟДﾟ) ['c']+((ﾟДﾟ)+'_') [(ﾟｰﾟ)+(ﾟｰﾟ)]+
(ﾟДﾟ) ['o']+((ﾟｰﾟ==3) +'_') [ﾟΘﾟ];(ﾟДﾟ) ['_'] =(o^_^o) [ﾟoﾟ] [ﾟoﾟ];(ﾟεﾟ)=((ﾟｰﾟ==3) +'_') [ﾟΘﾟ]+ (ﾟДﾟ) .ﾟДﾟﾉ
+((ﾟДﾟ)+'_') [(ﾟｰﾟ) + (ﾟｰﾟ)]+((ﾟｰﾟ==3) +'_') [o^_^o -ﾟΘﾟ]+((ﾟｰﾟ==3) +'_') [ﾟΘﾟ]+ (ﾟωﾟﾉ +'_') [ﾟΘﾟ]; (ﾟｰﾟ)+=
(ﾟΘﾟ); (ﾟДﾟ)[ﾟεﾟ]='\\'; (ﾟДﾟ).ﾟΘﾟ/=(ﾟДﾟ+ ﾟｰﾟ)[o^_^o -(ﾟΘﾟ)];(oﾟｰﾟo)=(ﾟωﾟﾉ +'_')[c^_^o];(ﾟДﾟ) [ﾟoﾟ]='\"';(ﾟД
ﾟ) ['_'] ( (ﾟДﾟ) ['_'] (ﾟεﾟ+(ﾟДﾟ)[ﾟoﾟ]+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ (ﾟΘﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+
(ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ ((o^_^o) - (ﾟΘﾟ))+ (
ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+((ﾟｰﾟ) + (ﾟΘﾟ))+ (c^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟｰﾟ)+ ((o^_^o)
 - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟΘﾟ)+ (c^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((
ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ ((ﾟｰﾟ) +
(o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟｰﾟ)+ (c^_^o)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o^_^o)
)+ ((ﾟｰﾟ) + (o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ ((ﾟｰﾟ) + (o^_^o))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((o^_^o) +(o
^_^o))+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ ((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟｰﾟ)+ (ﾟДﾟ)[ﾟεﾟ]+(ﾟΘﾟ)+ (ﾟｰﾟ)+ (ﾟｰﾟ)+ (ﾟДﾟ)[
ﾟεﾟ]+(ﾟｰﾟ)+ ((o^_^o) - (ﾟΘﾟ))+ (ﾟДﾟ)[ﾟεﾟ]+((ﾟｰﾟ) + (ﾟΘﾟ))+ (ﾟΘﾟ)+ (ﾟДﾟ)[ﾟoﾟ]) (ﾟΘﾟ)) ('_');
```

VBScript.Encode混淆加密

ppencode

rrencode

jjencode/aaencode

JSfuck

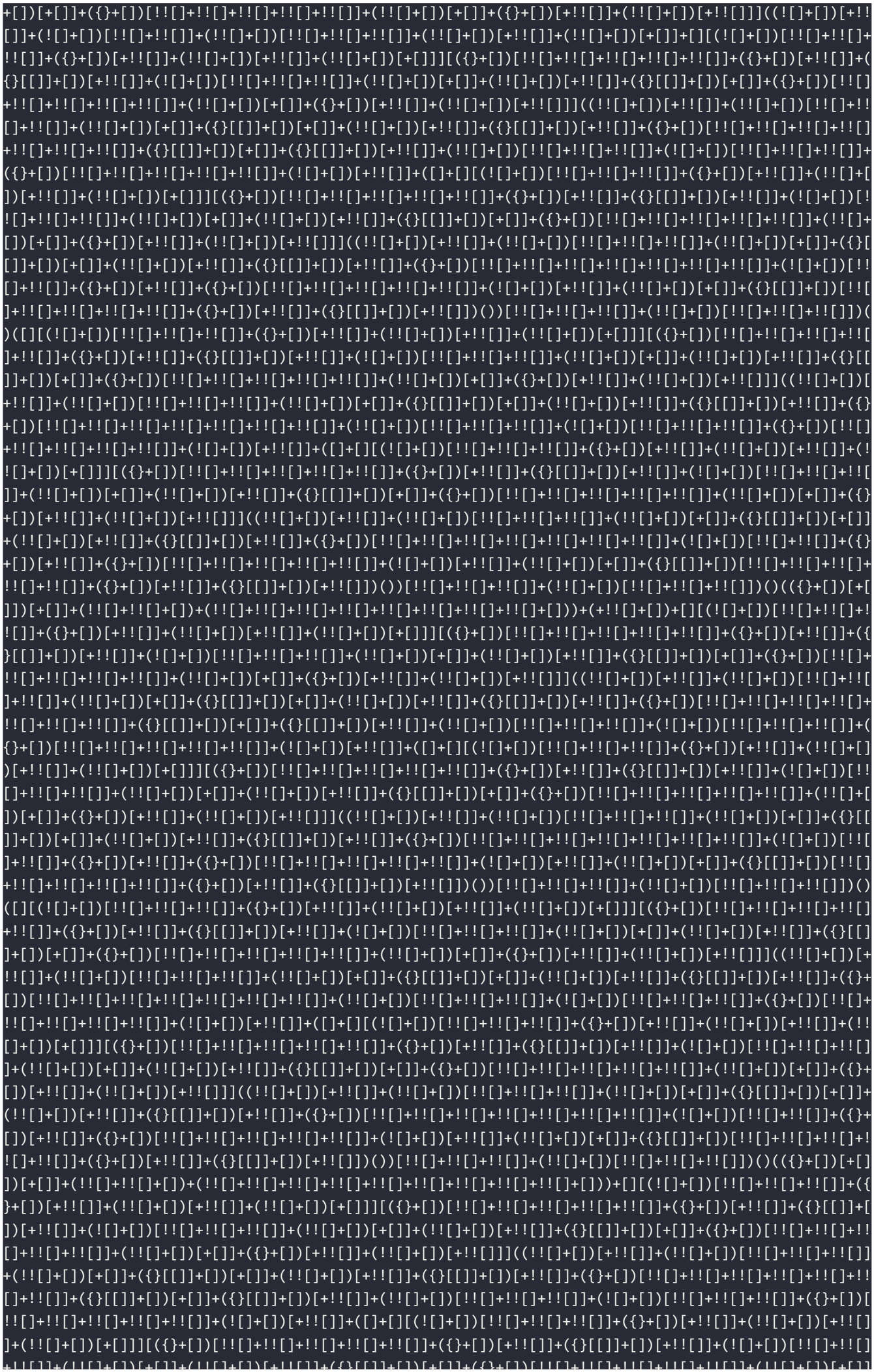
- alert(1)

```
[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]
]+(!![]+[])[+!+[]]][([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!!
[]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!!
+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]+(!![]+[])[+[]]+([]
[[]]+[])[+!+[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]+([][(![]+[])[+
[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+
!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+
[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[+!+[]+[+[]]]+(!![]+[])[+!+[]]](![]
+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]+(!![]+[])[!+[]+!+[]]+(!![]+[
])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]]+(!![]+[])[+
])[+!+[]]])[!+[]+!+[]+[+[]]]+[+!+[]]+(!![]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+
[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])()
```

jother

- alert(1);

```
[][(![]+[])[!!]+!!]+!!]]+({}+[])[+!!]]+(!!]+[])[+!!]]+(!!]+[])[+!!]]+(!!]+[])[+]]][({}+[])[!!]+!!]+!!]+!!]+
!!]]+({}+[])[+!!]]+({}[]]+[])[+!!]]+(![]+[])[!!]+!!]+!!]]+(!!]+[])[+]]+(!!]+[])[+!!]]+({}[]]
```



```
+({}+[])[+!![]]+(!![]+[])[+!![]]((!![]+[])[+!![]]+(!![]+[])[!![]+!![]+!![]]+(!![]+[])[+[]]+({}[[]]+[])[
+[]]+(!![]+[])[+!![]]+({}[[]]+[])[+!![]]+({}+[])[!![]+!![]+!![]+!![]+!![]+!![]]+(![]+[])[!![]+!![]]
+({}+[])[+!![]]+({}+[])[!![]+!![]+!![]+!![]+!![]]+(![]+[])[+!![]]+(!![]+[])[+[]]+({}[[]]+[])[!![]+!![]+!
![]+!![]+!![]]+({}+[])[+!![]]+({}[[]]+[])[+!![]])())[!![]+!![]+!![]]+(!![]+[])[!![]+!![]+!![]])()([][(![
]+[])[!![]+!![]+!![]]+({}+[])[+!![]]+(!![]+[])[+!![]]+(!![]+[])[+[]][({}+[])[!![]+!![]+!![]+!![]+!![]]+
({}+[])[+!![]]+({}[[]]+[])[+!![]]+(![]+[])[!![]+!![]+!![]]+(!![]+[])[+[]]+(!![]+[])[+!![]]+({}[[]]+[])[+
[]]+({}+[])[!![]+!![]+!![]+!![]+!![]]+(!![]+[])[+[]]+({}+[])[+!![]]+(!![]+[])[+!![]]](!![]+[])[+!![]]+(
!![]+[])[!![]+!![]+!![]]+(!![]+[])[+[]]+({}[[]]+[])[+[]]+(!![]+[])[+!![]]+({}[[]]+[])[+!![]]+({}+[])[!![
]+!![]+!![]+!![]+!![]+!![]]+(!![]+[])[!![]+!![]+!![]]+(![]+[])[!![]+!![]+!![]]+({}+[])[!![]+!![]+!!
[]+!![]+!![]]+(![]+[])[+!![]]+([]+[][(![]+[])[!![]+!![]+!![]]+({}+[])[+!![]]+(!![]+[])[+!![]]+(!![]+[])[
+[]]][({}+[])[!![]+!![]+!![]+!![]+!![]]+({}+[])[+!![]]+({}[[]]+[])[+!![]]+(![]+[])[!![]+!![]+!![]]+(!![]
+[])[+[]]+(!![]+[])[+!![]]+({}[[]]+[])[+[]]+({}+[])[!![]+!![]+!![]+!![]+!![]]+(!![]+[])[+[]]+({}+[])[+!!
[]]+(!![]+[])[+!![]]](!![]+[])[+!![]]+(!![]+[])[!![]+!![]+!![]+!![]+!![]]+(!![]+[])[+[]]+({}[[]]+[])[+[]]+(!![]+[
])[+!![]]+({}[[]]+[])[+!![]]+({}+[])[!![]+!![]+!![]+!![]+!![]+!![]+!![]]+(![]+[])[!![]+!![]]+({}+[])[+!!
[]]+({}+[])[!![]+!![]+!![]+!![]+!![]]+(![]+[])[+!![]]+(!![]+[])[+[]]+({}[[]]+[])[!![]+!![]+!![]+!![]+!![
]]+({}+[])[+!![]]+({}[[]]+[])[+!![]])())[!![]+!![]+!![]]+(!![]+[])[!![]+!![]+!![]])()(({}+[])[+[]])[+[]]
+(!![]+!![]+!![]+[])+({}+[])[!![]+!![]]))()
```

brainfuck编程语言

- 明文：hello!

```
+++++ +++++ [->++ +++++ +++<] >++++ .---. +++++ ++..+ ++.<+ +++++ +++++
[->++ +++++ ++++< ]>+++ ++++. <++++ +++[- >---- ---<] >--.< +++++ ++[->
----- --<]> ----- ----- .<
```

一些脑洞

被雷劈过的字符串 => gndk€rlqhmtkwwp}z

比较一下"gndk"与"flag"的ASCII码

gndk的10进制的ASCII码分别是：103 110 100 107

flag的10进制的ASCII码分别是 ：102 108 97 103

解密代码：

```c
#include<stdio.h>
#include<string.h>

int main(){
    int i;
    char a[100];
    printf("请输入被雷劈过的字符串:\n");
    scanf("%s",a);
    printf("解密:\n");
    for(i=0;i<strlen(a);i++){
        a[i]=a[i]-(i+1);
        printf("%c",a[i]);
    }
}
```

参考资料

https://www.cnblogs.com/mq0036/p/6544055.html
https://www.cnblogs.com/Yuuki-/p/7868581.html
http://news.mydrivers.com/1/190/190926.htm
RSA算法：
https://blog.csdn.net/qq_18661257/article/details/54563017