密码学--CTF Crypto 总结

原创

m0_64910183 于 2021-12-15 16:42:50 发布 1288 收藏 2
分类专栏： CTF 文章标签： 安全 哈希算法

密码学简介

密码学（Cryptography）一般可分为古典密码学和现代密码学。

其中，古典密码学，作为一种实用性艺术存在，其编码和破译通常依赖于设计者和敌手的创造力与技巧，并没有对密码学原件进行清晰的定义。其主要包含以下几个方面

- 单表替换加密
- 多表替换加密
- 奇奇怪怪的加密方式

而现代密码学则起源于 20 世纪末出现的大量相关理论，这些理论使得现代密码学成为了一种可以系统而严格地学习的科学。现代密码学又可以简单的分为以下几个方面

- 对称密码，以 DES，AES，RC4 为代表。
- 非对称密码，以 RSA，椭圆曲线加密为代表。
- Hash，以 MD5，SHA1，SHA512 等为代表。
- 数字签名，以 RSA 签名，ElGamal 签名，DSA 签名为代表。

其加密方式主要有两种方式

- 块加密
- 流加密

一般来说，密码设计者的基本想法是确保密码框架的

- 保密性
- 完整性
- 可用性
- 不可否认性

其中，前三者又称为 CIA 三元组。

而对于密码破解者来说，一般都是要想办法识别密码算法，然后利用暴力破解方法或者密码框架的漏洞进行破解。当然，也有可能是处于希望构造虚假的哈希值或者签名来绕过相应的检测。

一般来说，我们都会假设攻击者知道要攻破的密码体制，一般来说会有如下几种攻击类型，

攻击类型	说明	适用场景

攻击类型	说明	适用场景
唯密文攻击	攻击者只拥有密文	古典密码
已知明文攻击	知道明文和对应的密文	古典密码，对称密码，非对称密码
选择明文攻击	能够对选择一些明文加密后获得其相应的密文	对称密码，非对称密码
选择密文攻击	能够对选择一些密文解密后获得明文	非对称密码

这里推荐一些资料

- 可汗学院公开课
- 深入浅出密码学——常用加密技术原理与应用

参考

- 维基百科-密码学

古典密码简介

shannon的保密系统的通信理论发表前的密码都归为古典密码，古典密码编码方法归根结底主要有两种，即置换和代换。

置换密码

把明文中的字母重新排列，字母本身不变，但其位置改变了，这样编成的密码称为置换密码。最简单的置换密码是把明文中的字母顺序倒过来，然后截成固定长度的字母组作为密文。

- 列置换 加密：将明文按固定长m分组，即每行m个字母，在密钥控制下按某一顺序交换列，最后按列优先的顺序依次读出，即产生了密文。 解密：逆过程。

- 周期置换 很大程度上同列置换，只不过加、解密时，在列交换后是按行优先的顺序向下进行。

代换

代换密码则是将明文中的字符替代成其他字符。 - 单表代换密码 ①加法密码 A和B是有 n个字母的字母表。 定义一个由A到B的映射：$f:A\rightarrow B$ $f(a_i)= b_i=a_j\ j=i+k \mod n$ 加法密码是用明文字母在字母表中后面第 k个字母来代替。 K=3 时是著名的凯撒密码，是古罗马恺撒大帝在营救西塞罗战役时用来保护重要军情的加密系统。是世界历史上第一个著名密码应用。

②乘法密码 A和B是有n个字母的字母表。?定义一个由A到B的映射：$f:A\rightarrow B$ $f(a_i)= b_i= a_j\ j=ik \mod n$ 其中: (n,k)=1。 注意：只有(n,k)=1，才能正确解密。

③密钥词组代替密码 随机选一个词语，去掉其中的重复字母，写到矩阵的第一行，从明文字母表中去掉这第一行的字母，其余字母顺序写入矩阵。然后按列取出字母构成密文字母表。

- 多表代换密码 单表代替密码的安全性不高，一个原因是一个明文字母只由一个密文字母代替。可以利用频率分析来破译。故产生了更为安全的多表代换密码，即构造多个密文字母表，在密钥的控制下用以一系列代换表依次对明文消息的字母序列进行代换。

在古典密码学中，我们主要介绍单表替代密码，多表替代密码，以及一些其它比较有意思的密码。

值得一提的是，在古典密码学中，设计者主要考虑消息的保密性，使得只有相关密钥的人才可以解密密文获得消息的内容，对于消息的完整性和不可否认性则并没有进行太多的考虑。

单表代换加密

[目的]

- 学习常见的几种加密方式

[环境]

- Ubuntu

[工具]

- JPK

[原理]

在单表替换加密中，所有的加密方式几乎都有一个共性，那就是明密文一一对应。所以说，一般有以下两种方式来进行破解

- 在密钥空间较小的情况下，采用暴力破解方式
- 在密文长度足够长的时候，使用词频分析

当密钥空间足够大，而密文长度足够短的情况下，破解较为困难。

凯撒密码

原理

凯撒密码（Caesar）加密时会将明文中的 每个字母 都按照其在字母表中的顺序向后（或向前）移动固定数目（循环移动）作为密文。例如，当偏移量是左移 3 的时候（解密时的密钥就是 3）：

```
明文字母表：ABCDEFGHIJKLMNOPQRSTUVWXYZ
密文字母表：DEFGHIJKLMNOPQRSTUVWXYZABC
```

使用时，加密者查找明文字母表中需要加密的消息中的每一个字母所在位置，并且写下密文字母表中对应的字母。需要解密的人则根据事先已知的密钥反过来操作，得到原来的明文。例如：

```
明文：THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
密文：WKH TXLFN EURZQ IRA MXPSV RYHU WKH ODCB GRJ
```

根据偏移量的不同，还存在若干特定的恺撒密码名称：

- 偏移量为 10：Avocat（A→K）
- 偏移量为 13： ROT13，ROT13是它自己本身的逆反，也就是说，要还原ROT13，套用加密同样的算法即可得，故同样的操作可用再加密与解密。
- 偏移量为 -5：Cassis （K 6）
- 偏移量为 -6：Cassette （K 7）

此外，还有一种基于密钥的凯撒密码 Keyed Caesar。其基本原理是 利用一个密钥，将密钥的每一位转换为数字（一般转化为字母表对应顺序的数字），分别以这一数字为密钥加密明文的每一位字母。

移位密码

与凯撒密码类似，区别在于移位密码不仅会处理字母，还会处理数字和特殊字符，常用 ASCII 码表进行移位。其破解方法也是遍历所有的可能性来得到可能的结果。

Atbash Cipher

原理

埃特巴什码（Atbash Cipher）其实可以视为下面要介绍的简单替换密码的特例，它使用字母表中的最后一个字母代表第一个字母，倒数第二个字母代表第二个字母。在罗马字母表中，它是这样出现的：

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

简单替换密码

原理

简单替换密码（Simple Substitution Cipher）加密时，将每个明文字母替换为与之唯一对应且不同的字母。它与恺撒密码之间的区别是其密码字母表的字母不是简单的移位，而是完全是混乱的，这也使得其破解难度要高于凯撒密码。 比如：

```
明文字母 : abcdefghijklmnopqrstuvwxyz
密钥字母 : phqgiumeaylnofdxjkrcvstzwb
```

a 对应 p，d 对应 h，以此类推。

仿射密码

原理

仿射密码的加密函数是 E(x)=(ax+b)(modm)，其中

- x 表示明文按照某种编码得到的数字
- a 和 m 互质
- m 是编码系统中字母的数目。

解密函数是 D(x)=a−1(x−b)(modm)，其中 a−1 是 a 在 Zm 群的乘法逆元。

[步骤]

凯撒密码

这里以 XMan 一期夏令营分享赛宫保鸡丁队 Crypto 100 为例进行介绍。

```
明文：s0a6u3u1s0bv1a
密钥：guangtou
偏移：6,20,0,13,6,19,14,20
密文：y0u6u3h1y0uj1u
```

破解

对于不带密钥的凯撒密码来说，其基本的破解方法有两种方式

1. 遍历 26 个偏移量，适用于普遍情况
2. 利用词频分析，适用于密文较长的情况。

其中，第一种方式肯定可以得到明文，而第二种方式则不一定可以得到正确的明文。

而对于基于密钥的凯撒密码来说，一般来说必须知道对应的密钥。

具体以本题为例：guangtou首先转换为偏移，a转换为0，其它字母进行对应偏移，则最终偏移为6,20,0,13,6,19,14,20，而明文为s0a6u3u1s0bv1a，只对其中的字母进行加密，其中对第一个字母s，进行密钥第一个字母s对应的偏移，所以加密后为y，继续对明文第二个字母a进行密钥u对应的偏移，加密后字母为u，全部加密完成后可从明文得到密文，反向操作可从密文得到明文。

工具

一般我们有如下的工具，其中JPK比较通用。

- JPK，可解带密钥与不带密钥
- 凯撒密码在线解密工具

Atbash Cipher

下面给出一个例子，该例根据在罗马字母表中对应规则加密而成，理解即可

```
明文：the quick brown fox jumps over the lazy dog
密文：gsv jfrxp yildm ulc qfnkh levi gsv ozab wlt
```

破解

可以看出其密钥空间足够短，同时当密文足够长时，仍然可以采用词频分析的方法解决。

简单替换密码

示例，下方示例使用了原理部分，简单替换部分的对应规则加密而成

```
明文：the quick brown fox jumps over the lazy dog
密文：cei jvaql hkdtf udz yvoxr dsik cei npbw gdm
```

而解密时，我们一般是知道了每一个字母的对应规则，才可以正常解密。

破解

由于这种加密方式导致其所有的密钥个数是26!，所以几乎上不可能使用暴力的解决方式。所以我们 一般采用词频分析。

工具

- 词频分析破解工具

仿射密码

下面我们以 E(x)=(5x+8)mod26 函数为例子进行介绍，加密字符串为 AFFINE CIPHER，这里我们直接采用字母表26个字母作为编码系统

明文	A	F	F	I	N	E	C	I	P	H	E	R
x	0	5	5	8	13	4	2	8	15	7	4	17
y=5x+8	8	33	33	48	73	28	18	48	83	43	28	93
ymod26	8	7	7	22	21	2	18	22	5	17	2	15
密文	I	H	H	W	V	C	S	W	F	R	C	P

其对应的加密结果是 IHHWVCSWFRCP。

对于解密过程，正常解密者具有a与b，根据原理部分的对应说明可以计算得到 a−1 为 21，所以其解密函数是 D(x)=21(x−8)(mod26) ，解密如下

密文	I	H	H	W	V	C	S	W	F	R	C	P
y	8	7	7	22	21	2	18	22	5	17	2	15
x=21(y−8)	0	-21	-21	294	273	-126	210	294	-63	189	-126	147
xmod26	0	5	5	8	13	4	2	8	15	7	4	17
明文	A	F	F	I	N	E	C	I	P	H	E	R

可以看出其特点在于只有 26 个英文字母。

破解

首先，我们可以看到的是，仿射密码对于任意两个不同的字母，其最后得到的密文必然不一样，所以其也具有最通用的特点。当密文长度足够长时，我们可以使用频率分析的方法来解决。

其次，我们可以考虑如何攻击该密码。可以看出当a=1 时，仿射加密是凯撒加密。而一般来说，我们利用仿射密码时，其字符集都用的是字母表，一般只有26个字母，而不大于26的与26互素的个数一共有

$$\phi(26)=\phi(2)\times\phi(13)=12$$
算上b的偏移可能，一共有可能的密钥空间大小也就是

$$12\times26=312$$
一般来说，对于该种密码，我们至少得是在已知部分明文的情况下才可以攻击。下面进行简单的分析。

这种密码由两种参数来控制，如果我们知道其中任意一个参数，那我们便可以很容易地快速枚举另外一个参数得到答案。

但是，假设我们已经知道采用的字母集，这里假设为26个字母，我们还有另外一种解密方式，我们只需要知道两个加密后的字母 y1,y2 即可进行解密。那么我们还可以知道

$$y1=(ax1+b)(mod26)y2=(ax2+b)(mod26)$$
两式相减，可得

$$y1-y2=a(x1-x2)(mod26)$$
这里 y1,y2 已知，如果我们知道密文对应的两个不一样的字符 x1 与 x2 ，那么我们就可以很容易得到 a ，进而就可以得到 b 了。

例子

下方程序保存在操作机桌面/tools/TWCTF2016-super_express/目录中

这里我们以TWCTF 2016 的 super_express为例进行介绍。简单看一下给的源码

```
import sys
key = '****CENSORED***************'
flag = 'TWCTF{*******CENSORED********}'

if len(key) % 2 == 1:
    print("Key Length Error")
    sys.exit(1)

n = len(key) / 2
encrypted = ''
for c in flag:
    c = ord(c)
    for a, b in zip(key[0:n], key[n:2*n]):
        c = (ord(a) * c + ord(b)) % 251
    encrypted += '%02x' % c

print encrypted
```

可以发现，虽然对于 flag 中的每个字母都加密了 n 次，如果我们仔细分析的话，我们可以发现

c1=a1c+b1c2=a2c1+b2=a1a2c+a2b1c+b2=kc+d
根据第二行的推导，我们可以得到其实 cn 也是这样的形式，可以看成 cn=xc+y ，并且，我们可以知道的是，key 是始终不变化的，所以说，其实这个就是仿射密码。

此外，题目中还给出了密文，密文内容如下

805eed80cbbccb94c36413275780ec94a857dfec8da8ca94a8c313a8ccf9

以及部分部分密文对应的明文，那么我们就很容易利用已知明文攻击的方法来攻击了，利用代码如下，将密文保存为文本文件并重命名为encrypted，在encrypted目录运行以下代码

```
import gmpy

key = '****CENSORED***************'
flag = 'TWCTF{*******CENSORED********}'

f = open('encrypted', 'r')
data = f.read().strip('\n')
encrypted = [int(data[i:i + 2], 16) for i in range(0, len(data), 2)]
plaindelta = ord(flag[1]) - ord(flag[0])
cipherdalte = encrypted[1] - encrypted[0]
a = gmpy.invert(plaindelta, 251) * cipherdalte % 251
b = (encrypted[0] - a * ord(flag[0])) % 251
a_inv = gmpy.invert(a, 251)
result = ""
for c in encrypted:
    result += chr((c - b) * a_inv % 251)
print result
```

结果如下(在文件所在目录右键，选择open Terminal here可在当前目录打开终端，如下运行解密程序即可)

```
shell
➜  TWCTF2016-super_express git:(master) ✗ python exploit.py
TWCTF{Faster_Than_Shinkansen!}
```

[总结]

通过本节的学习，我们可以学到几种常见的加密方式以及破解方法

多表代换加密

对于多表替换加密来说，加密后的字母几乎不再保持原来的频率，所以我们一般只能通过寻找算法实现对应的弱点进行破解。

原理

Playfair 密码（Playfair cipher or Playfair square）是一种替换密码，1854 年由英国人查尔斯·惠斯通（Charles Wheatstone）发明，基本算法如下：

1. 选取一串英文字母，除去重复出现的字母，将剩下的字母逐个逐个加入 5 × 5 的矩阵内，剩下的空间由未加入的英文字母依 a-z 的顺序加入。注意，将 q 去除，或将 i 和 j 视作同一字。
2. 将要加密的明文分成两个一组。若组内的字母相同，将 X（或 Q）加到该组的第一个字母后，重新分组。若剩下一个字，也加入 X 。
3. 在每组中，找出两个字母在矩阵中的地方。
    1. 若两个字母不同行也不同列，在矩阵中找出另外两个字母（第一个字母对应行优先），使这四个字母成为一个长方形的四个角。
    2. 若两个字母同行，取这两个字母右方的字母（若字母在最右方则取最左方的字母）。
    3. 若两个字母同列，取这两个字母下方的字母（若字母在最下方则取最上方的字母）。

新找到的两个字母就是原本的两个字母加密的结果。

以 playfair example 为密匙，得

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

要加密的讯息为 Hide the gold in the tree stump

```
HI DE TH EG OL DI NT HE TR EX ES TU MP
```

就会得到

```
BM OD ZB XD NA BE KU DM UI XM MO UV IF
```

工具

- CAP4

原理

Polybius密码又称为棋盘密码，其一般是将给定的明文加密为两两组合的数字，其常用密码表

	1	**2**	**3**	**4**	**5**
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P

	1	2	3	4	5
4	Q	R	S	T	U
5	V	W	X	Y	Z

举个例子，明文 HELLO，加密后就是 23 15 31 31 34。

另一种密码表

	A	D	F	G	X
A	b	t	a	l	p
D	d	h	o	z	k
F	q	f	v	s	n
G	g	j	c	u	x
X	m	r	e	w	y

注意，这里字母的顺序被打乱了。

A D F G X 的由来：

> 1918 年，第一次世界大战将要结束时，法军截获了一份德军电报，电文中的所有单词都由 A、D、F、G、X 五个字母拼成，因此被称为 ADFGX 密码。ADFGX 密码是 1918 年 3 月由德军上校 Fritz Nebel 发明的，是结合了 Polybius 密码和置换密码的双重加密方案。

举个例子，HELLO，使用这个表格加密，就是 DD XF AG AG DF。

工具

- CrypTool

Vigenere 维吉尼亚密码

原理

维吉尼亚密码（Vigenere）是使用一系列凯撒密码组成密码字母表的加密算法，属于多表密码的一种简单形式。

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

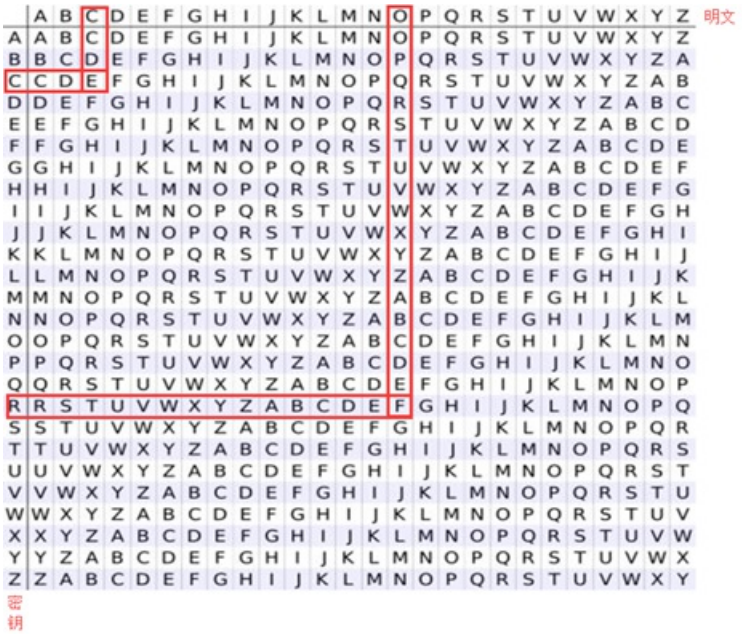
下面给出一个例子

明文：come greatwall
密钥：crypto

首先，对密钥进行填充使其长度与明文长度一样。

明文	**c**	**o**	**m**	**e**	**g**	**r**	**e**	**a**	**t**	**w**	**a**	**l**	**l**
密钥	c	r	y	p	t	o	c	r	y	p	t	o	c

其次，查表得密文



明文：come greatwall
密钥：crypto
密文：efkt zferrltzn

破解

对包括维吉尼亚密码在内的所有多表密码的破译都是以字母频率为基础的，但直接的频率分析却并不适用，这是因为在维吉尼亚密码中，一个字母可以被加密成不同的密文，因而简单的频率分析在这里并没有用。

破译维吉尼亚密码的关键在于它的密钥是循环重复的。 如果我们知道了密钥的长度，那密文就可以被看作是交织在一起的凯撒密码，而其中每一个都可以单独破解。关于密码的长度，我们可以 使用卡西斯基试验和弗里德曼试验来获取。

卡西斯基试验是基于类似 the 这样的常用单词有可能被同样的密钥字母进行加密，从而在密文中重复出现。例如，明文中不同的 CRYPTO 可能被密钥 ABCDEF 加密成不同的密文：

```
密钥：ABCDEF AB CDEFA BCD EFABCDEFABCD
明文：CRYPTO IS SHORT FOR CRYPTOGRAPHY
密文：CSASXT IT UKSWT GQU GWYQVRKWAQJB
```

此时明文中重复的元素在密文中并不重复。然而，如果密钥相同的话，结果可能便为（使用密钥 ABCD）：

```
密钥：ABCDAB CD ABCDA BCD ABCDABCDABCD
明文：CRYPTO IS SHORT FOR CRYPTOGRAPHY
密文：CSASTP KV SIQUT GQU CSASTPIUAQJB
```

此时卡西斯基试验就能产生效果。对于更长的段落此方法更为有效，因为通常密文中重复的片段会更多。如通过下面的密文就能破译出密钥的长度：

```
密文：DYDUXRMHTVDVNQDQNWDYDUXRMHARTJGWNQD
```

其中，两个 DYDUXRMH 的出现相隔了 18 个字母。因此，可以假定密钥的长度是 18 的约数，即长度为 18、9、6、3 或 2。而两个 NQD 则相距 20 个字母，意味着密钥长度应为 20、10、5、4 或 2。取两者的交集，则可以基本确定密钥长度为 2。接下来就是进行进一步的操作了。

关于更加详细的破解原理，这里暂时不做过多的介绍。

工具

- 已知密钥
    - Python 的 pycipher 库
    - 在线解密 Vigenère cipher
    - CAP4
- 未知密钥
    - Vigenère Cipher Codebreaker
    - Vigenere Solver, 不够完善。

Nihilist

原理

Nihilist密码又称关键字密码：明文 + 关键字 = 密文。以关键字 helloworld 为例。

首先利用密钥构造棋盘矩阵（类似 Polybius 密码） - 新建一个 5 × 5 矩阵 - 将字符不重复地依次填入矩阵 - 剩下部分按字母顺序填入 - 字母 i 和 j 等价

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

对于加密过程参照矩阵 M 进行加密：

```
a -> M[2,3] -> 23
t -> M[4,5] -> 45
```

对于解密过程

参照矩阵 M 进行解密：

```
23 -> M[2,3] -> a
45 -> M[4,5] -> t
```

可以看出，密文的特征有如下几点

- 纯数字
- 只包含 1 到 5
- 密文长度偶数。

Hill

原理

希尔密码（Hill）使用每个字母在字母表中的顺序作为其对应的数字，即A=0，B=1，C=2 等，然后将明文转化为 n 维向量，跟一个 n × n 的矩阵相乘，再将得出的结果模 26。注意用作加密的矩阵（即密匙）
在 Zn26Z26n 必须是可逆的，否则就不可能解码。只有矩阵的行列式和 26 互质，才是可逆的。下面举一个例子

```
明文：ACT
```

将明文化为矩阵。

□□□0219□□□[0219]
假设密钥为：

□□□6241131610201715□□□[6241131610201715]
加密过程为：

□□□6241131610201715□□□□□□0219□□□≡□□□67222319□□□≡□□□15147□□□mod26[6241131610201715][0219]≡[67222319]≡[15147]mod26
密文即为

```
密文：POH
```

工具

- CAP4
- Cryptool

例子

这里我们以ISCC 2015 base decrypt 150为例进行介绍，题目为

> 密文： 22,09,00,12,03,01,10,03,04,08,01,17 （wjamdbkdeibr）
>
> 使用的矩阵是 1 2 3 4 5 6 7 8 10
>
> 请对密文解密.

首先，矩阵是 3 × 3 的。说明每次加密3个字符。我们直接使用 Cryptool，需要注意的是，这个矩阵是按照列来排布的。即如下

```
1 4 7
2 5 8
3 6 10
```

最后的结果为 overthehillx。

AutokeyCipher

原理

自动密钥密码（Autokey Cipher）也是多表替换密码，与维吉尼亚密码密码类似，但使用不同的方法生成密钥。通常来说它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。下面我们以关键词自动密钥为例：

```
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，从相应的表格可得：

密文

```
VBP JOZGD IVEQV HYY AIICX CSNL FWW ZVDP WVK
```

工具

- 已知关键词
- Python 的 pycipher 库

其他类型加密

培根密码

原理

培根密码使用两种不同的字体，代表 A 和 B，结合加密表进行加解密。

a	**AAAAA**	**g**	**AABBA**	**n**	**ABBAA**	**t**	**BAABA**
b	AAAAB	h	AABBB	o	ABBAB	u-v	BAABB
c	AAABA	i-j	ABAAA	p	ABBBA	w	BABAA
d	AAABB	k	ABAAB	q	ABBBB	x	BABAB
e	AABAA	l	ABABA	r	BAAAA	y	BABBA
f	AABAB	m	ABABB	s	BAAAB	z	BABBB

上面的是常用的加密表。还有另外的一种加密表，可认为是将 26 个字母从 0 到 25 排序，以二进制表示，A 代表 0，B 代表 1。

下面这一段内容就是明文 steganography 加密后的内容，正常字体是 A，粗体是 B：

To encode a message each letter of the plaintext is replaced by a group of five of the letters 'A' or 'B'.

可以看到，培根密码主要有以下特点

- 只有两种字符
- 每一段的长度为 5
- 加密内容会有特殊的字体之分，亦或者大小写之分。

工具

- 培根密码在线解密工具

栅栏密码

原理

栅栏密码把要加密的明文分成 N 个一组，然后把每组的第 1 个字连起来，形成一段无规律的话。这里给出一个例子

```
明文：THERE IS A CIPHER
```

去掉空格后变为

```
THEREISACIPHER
```

分成两栏，两个一组得到

```
TH ER EI SA CI PH ER
```

先取出第一个字母，再取出第二个字母

```
TEESCPE
HRIAIHR
```

连在一起就是

```
TEESCPEHRIAIHR
```

上述明文也可以分为2栏。

```
THEREIS ACIPHER
```

组合得到密文

```
TAHCEIRPEHIESR
```

曲路密码

原理

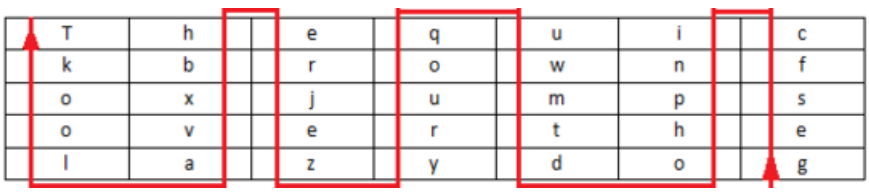
曲路密码（Curve Cipher）是一种换位密码，需要事先双方约定密钥（也就是曲路路径）。下面给出一个例子：

```
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）

T	h	e	q	u	i	c
k	b	r	o	w	n	f
o	x	j	u	m	p	s
o	v	e	r	t	h	e
l	a	z	y	d	o	g

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
```

列移位加密

原理

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。下面给出一个例子。

我们以明文 The quick brown fox jumps over the lazy dog，密钥 how are u 为例：

将明文填入 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）

T	h	e	q	u	i	c
k	b	r	o	w	n	f
o	x	j	u	m	p	s
o	v	e	r	t	h	e
l	a	z	y	d	o	g

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：

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

密文： qoury inpho Tkool hbxva uwmtd cfseg erjez

01248 密码

原理

该密码又称为云影密码，使用 0，1，2，4，8 四个数字，其中 0 用来表示间隔，其他数字以加法可以表示出如：28=10，124=7，18=9，再用 1->26 表示 A->Z。

可以看出该密码有以下特点

- 只有 0，1，2，4，8

例子

这里我们以CFF 2016 影之密码为例进行介绍，题目

> 884210122048022440401422420248012 2

我们按照0来进行分割，如下

内容	数字	字符
88421	8+8+4+2+1=23	W
122	1+2+2=5	E
48	4+8=12	L
2244	2+2+4+4=12	L

内容	数字	字符
4	4	D
142242	1+4+2+2+4+2=15	O
248	2+4+8=14	N
122	1+2+2=5	E

所以最后的 flag 为 WELLDONE。

JSFuck

原理

JSFuck 可以只用 6 个字符 []()!+ 来编写 JavaScript 程序。比如我们想用 JSFuck 来实现 alert(1) 代码如下

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

其他一些基本的表达：

```
false       =>  ![]
true        =>  !![]
undefined   =>  [][[]]
NaN         =>  +[![]]
0           =>  +[]
1           =>  +!+[]
2           =>  !+[]+!+[]
10          =>  [+!+[]]+[+[]]
Array       =>  []
Number      =>  +[]
String      =>  []+[]
Boolean     =>  ![]
Function    =>  []["filter"]
eval        =>  []["filter"]["constructor"]( CODE )()
window      =>  []["filter"]["constructor"]("return this")()
```

工具

- JSFuck 在线加密网站

BrainFuck

原理

Brainfuck，是一种极小化的计算机语言，它是由 Urban Müller 在 1993 年创建的。我们举一个例子，如果我们想要一个在屏幕上打印Hello World！，那么对应的程序如下。对于其中的原理，感兴趣的可以自行网上搜索。

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

与其对应的还有 ook。

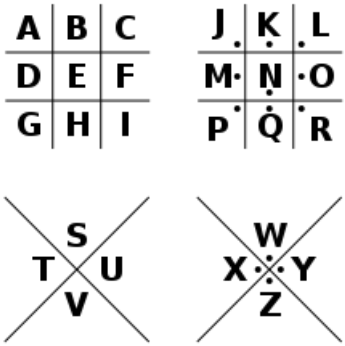
工具

- Brainfuck在线解密工具

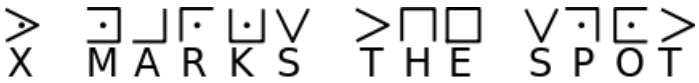
猪圈密码

原理

猪圈密码是一种以格子为基础的简单替代式密码，格子如下



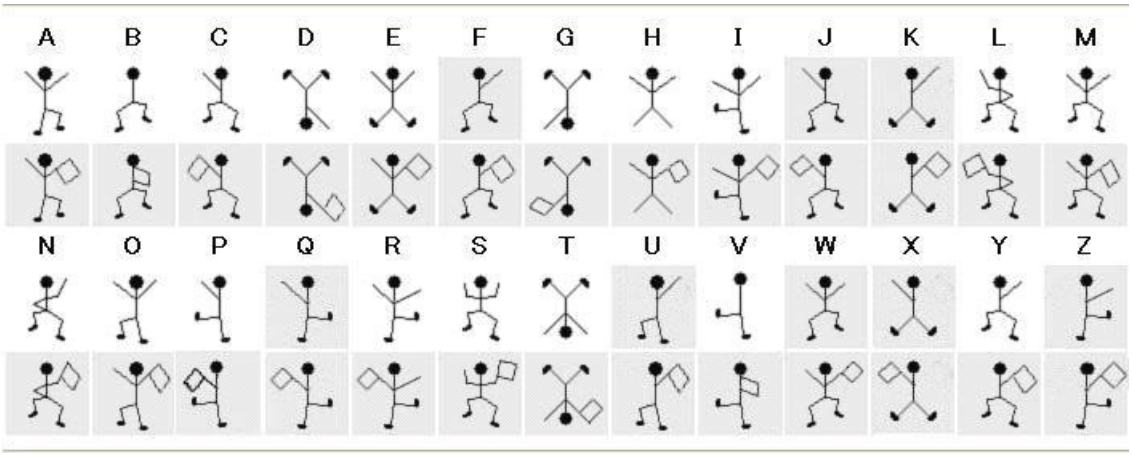
我们举一个例子，如明文为 X marks the spot ，那么密文如下



舞动的小人密码

原理

这种密码出自于福尔摩斯探案集。每一个跳舞的小人实际上对应的是英文二十六个字母中的一个，而小人手中的旗子则表明该字母是单词的最后一个字母，如果仅仅是一个单词而不是句子，或者是句子中最后的一个单词，则单词中最后一个字母不必举旗。

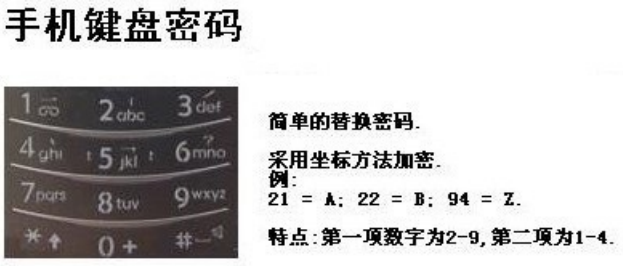


键盘密码

所谓键盘密码，就是采用手机键盘或者电脑键盘进行加密。

手机键盘密码

手机键盘加密方式，是每个数字键上有 3-4 个字母，用两位数字来表示字母，例如：ru 用手机键盘表示就是：7382，那么这里就可以知道了，手机键盘加密方式不可能用 1 开头，第二位数字不可能超过 4，解密的时候参考此



关于手机键盘加密还有另一种方式，就是「音的」式（这一点可能根据手机的不同会有所不同），具体参照手机键盘来打，例如：「数字」表示出来就是：748 94。在手机键盘上面按下这几个数，就会出：「数字」的拼音。

电脑键盘棋盘¶

电脑键盘棋盘加密，利用了电脑的棋盘方阵。

电脑键盘坐标

电脑键盘坐标加密，利用键盘上面的字母行和数字行来加密，例：bye 用电脑键盘 XY 表示就是：351613

电脑键盘 QWE

电脑键盘 QWE 加密法，就是用字母表替换键盘上面的排列顺序。

键盘布局加密

简单地说就是根据给定的字符在键盘上的样子来进行加密。

0CTF 2014 classic

> 小丁丁发现自己置身于一个诡异的房间，面前只有一扇刻着奇怪字符的门。 他发现门边上还有一道密码锁，似乎要输入密码才能开门。。4esxcft5 rdcvgt 6tfc78uhg 098ukmnb

发现这么乱，还同时包括数字和字母猜想可能是键盘密码，试着在键盘上按照字母顺序描绘一下，可得到0ops字样，猜测就是flag了。

2017年xman选拔赛——一二三，木头人

> 我数123木头人，再不行动就要被扣分。
>
> 23731263111628163518122316391715262121
>
> 密码格式xman{flag}

题目中有很明显的提示123，那么就自然需要联想到键盘密码中电脑键盘坐标密码，可以发现前几个数字第二个数字都是1-3范围内的，也验证了我们的猜测。于是

> 23-x
>
> 73-m
>
> 12-a
>
> 63-n
>
> 11-q

不对呀，密码格式是 xman{，第四个字符是 {，于是看了看 { 的位置，其并没有对应的横坐标，但是如果我们手动把它视为 11 的话，那么111就是 {。然后依次往后推，发现确实可行，，最后再把 121 视为 } 即可得到 flag。

```
xman{hintisenough}
```

从这里我们可以看出，我们还是要注意迁移性，不能单纯地照搬一些已有的知识。

总结

古典密码的基本解题思路可以总结如下

- **已知密码，识别密码**
- **未知密码，分析密码特性，利用暴力破解或者相应思路求解**

有以下几种方法可以用来识别密码

- **加密方式判别**
- **字符集判别**
- **加密结果样子判别**

围在栅栏里的爱

题目描述

> 最近一直在好奇一个问题，QWE到底等不等于ABC？
>
> -.- .. --.- .-.. .-- - ..-. -.-. --.- --. -. ... --- ---
>
> flag格式：CTF{xxx}

首先，根据密码样式判断是摩斯电码，解密后得到 KIQLWTFCQGNSOO，看着也不像 flag，题目中还提示有栅栏与 QWE到底等不等于ABC，两个都试了试之后，发现是先 QWE 然后栅栏可得到结果。

首先键盘 QWE 解密，试着解密得到 IILYOAVNEBSAHR。继而栅栏解密得到 ILOVESHIYANBAR。

2017 SECCON Vigenere3d

程序如下

```
# Vigenere3d.py
import sys
def _l(idx, s):
    return s[idx:] + s[:idx]
def main(p, k1, k2):
    s = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz_{}"
    t = [[_l((i+j) % len(s), s) for j in range(len(s))] for i in range(len(s))]
    i1 = 0
    i2 = 0
    c = ""
    for a in p:
        c += t[s.find(a)][s.find(k1[i1])][s.find(k2[i2])]
        i1 = (i1 + 1) % len(k1)
        i2 = (i2 + 1) % len(k2)
    return c
print main(sys.argv[1], sys.argv[2], sys.argv[2][::-1])

$ python Vigenere3d.py SECCON{************************} **************
POR4dnyTLHBfwbxAAZhe}}ocZR3Cxcftw9
```

首先，我们先来分析一下 t 的构成

t[i][j]=s[i+j:]+s[:i+j]t[i][k]=s[i+k:]+s[:i+k]t[i][j]=s[i+j:]+s[:i+j]t[i][k]=s[i+k:]+s[:i+k]

t[i][j][k]t[i][j][k] 为 t[i][j]t[i][j] 中的第 k 个字符，t[i][k][j]t[i][k][j] 为 t[i][k]t[i][k] 中的第 j 个字符。无论是 i+j+ki+j+k 是否超过 len(s) 两者都始终保持一致，即 t[i][j][k]=t[i][k][j]t[i][j][k]=t[i][k][j] 。

故而，其实对于相同的明文来说，可能有多个密钥使其生成相同的密文。

然而上面分析就是单纯地分析而已，，下面开始正题。

不难看出，密文的每一位只与明文的相应位相关，而且，密钥的每一位的空间最大也就是 s 的大小，所以我们可以使用爆破来获取密钥。这里根据上述命令行提示，可以知道密钥长度为 14，恰好明文前面7个字节已知。恢复密钥的 exp 如下

```
def get_key(plain, cipher):
    s = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz_{}"
    t = [[_l((i + j) % len(s), s) for j in range(len(s))]
         for i in range(len(s))]
    i1 = 0
    i2 = 0
    key = ['*'] * 14
    for i in range(len(plain)):
        for i1 in range(len(s)):
            for i2 in range(len(s)):
                if t[s.find(plain[i])][s.find(s[i1])][s.find(s[i2])] == cipher[
                        i]:
                    key[i] = s[i1]
                    key[13 - i] = s[i2]
    return ''.join(key)
```

恢复明文的脚本如下

```
def decrypt(cipher, k1, k2):
    s = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz_{}"
    t = [[_l((i + j) % len(s), s) for j in range(len(s))]
         for i in range(len(s))]
    i1 = 0
    i2 = 0
    plain = ""
    for a in cipher:
        for i in range(len(s)):
            if t[i][s.find(k1[i1])][s.find(k2[i2])] == a:
                plain += s[i]
                break
        i1 = (i1 + 1) % len(k1)
        i2 = (i2 + 1) % len(k2)
    return plain
```

得到明文如下

```
➜  2017_seccon_vigenere3d git:(master) python exp.py
SECCON{Welc0me_to_SECCON_CTF_2017}
```

对称加密简介

定义

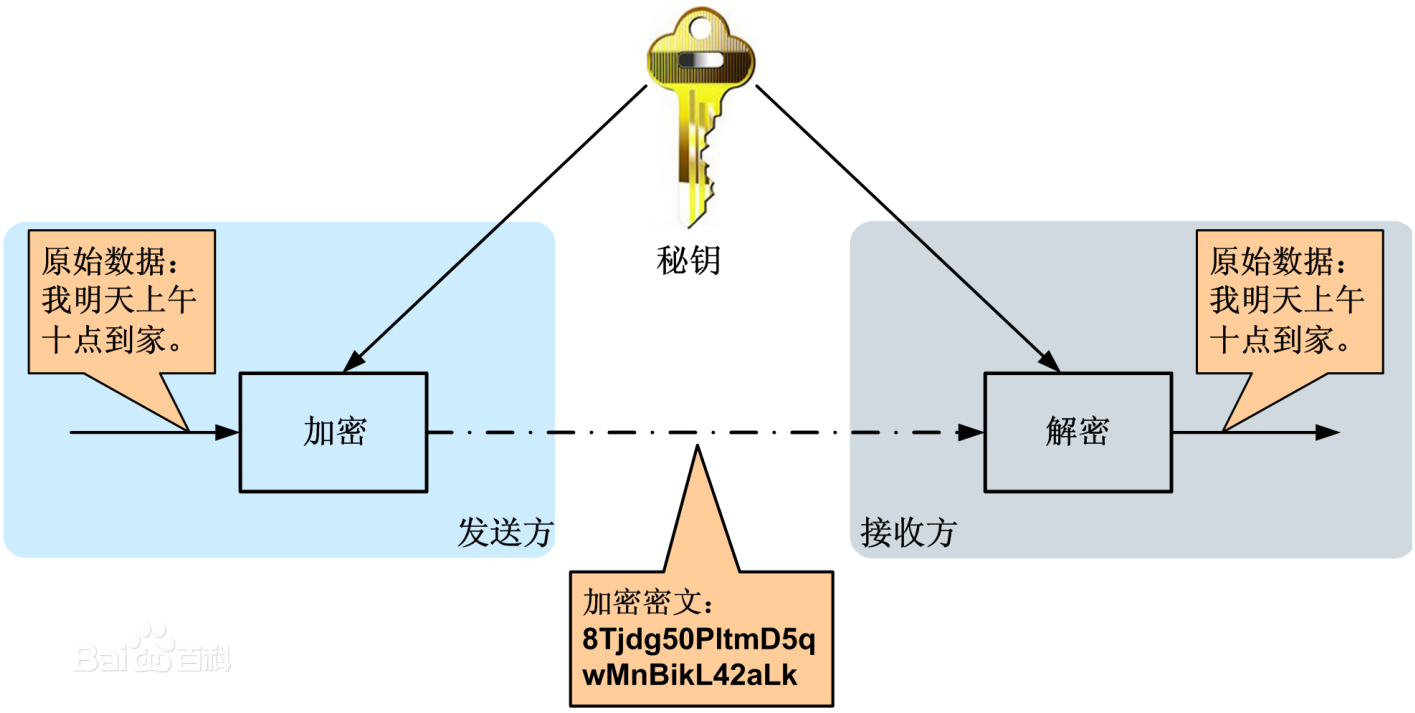
所谓对称加密就是采用单钥密码系统的加密方法，同一个密钥同时用于信息的加密和解密，也称为单密钥加密。

在对称加密中，密钥是控制加密及解密过程的指令；算法是一组规则，规定如何进行加密和解密。加密的安全性不仅取决于加密算法本身，密钥管理的安全性也同样重要。

对称算法的安全性依赖于密钥，泄漏密钥就意味着任何人都可以对他们发送或接收的消息解密，所以密钥的保密性对通信的安全性至关重要。

工作过程

在对称加密中，数据发送方将明文(原始数据)和加密密钥一起经过加密算法处理后，使其变成复杂的加密密文发送出去；接收方收到密文后，若想解读原文，则需要使用加密密钥及相同算法的逆算法对密文进行解密，才能使其恢复成可读明文。在对称加密算法中，使用的密钥只有一个，发收信双方都使用这个密钥对数据进行加密和解密。



特点

- 优点

算法公开 、加密计算量小、速度块，适合对大量数据进行加密的场景。

- 不足之处
- 密钥传输问题：，由于对称加密的加密和解密使用的是同一个密钥，所以对称加密的安全性就不仅仅取决于加密算法本身的强度，更取决于密钥是否被安全的保管，因此加密者如何把密钥安全的传递到解密者手里，就成了对称加密面临的关键问题。（比如，客户端不能直接存储对称加密的密钥，因为被反编译之后，密钥就泄露了，数据安全性就得不到保障，所以实际中一般都是客户端向服务端请求对称加密的密钥，而且密钥还得通过非对称加密加密后再传输。）
- 密钥管理问题：随着密钥数量的增多，密钥的管理问题会逐渐显现出来。比如在加密用户的信息时，不能所有用户都用同一个密钥加密解密，否则，一旦一个用户密钥泄漏，就相当于泄露了所有用户的信息，因此需要为每一个用户单独的生成一个密钥并且管理，这样密钥管理的代价也会非常大。

常用算法

在对称加密算法中常用的算法有：

- RC4加密
- DES加密
- 3DES加密
- AES加密

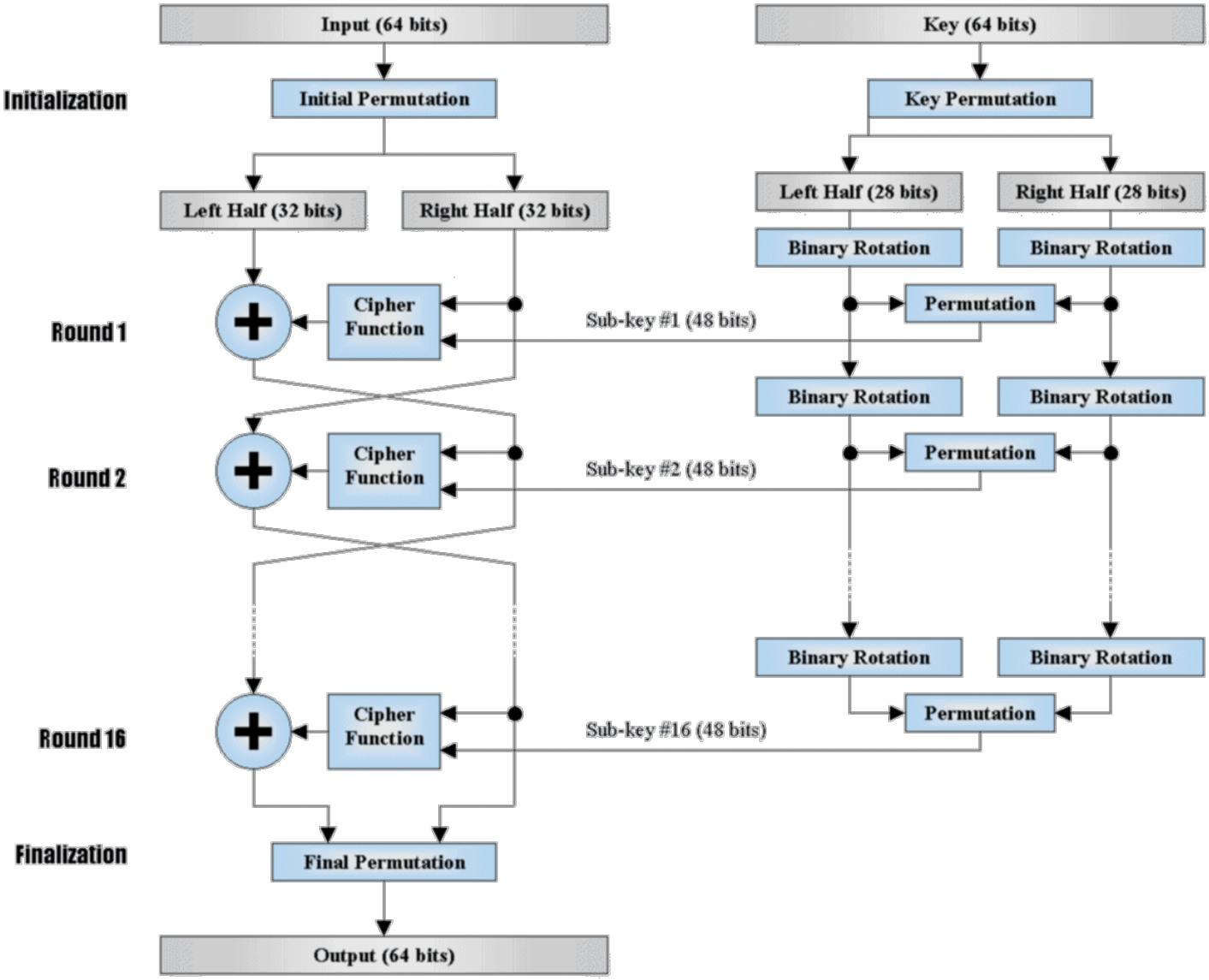
DES

¶基本介绍¶

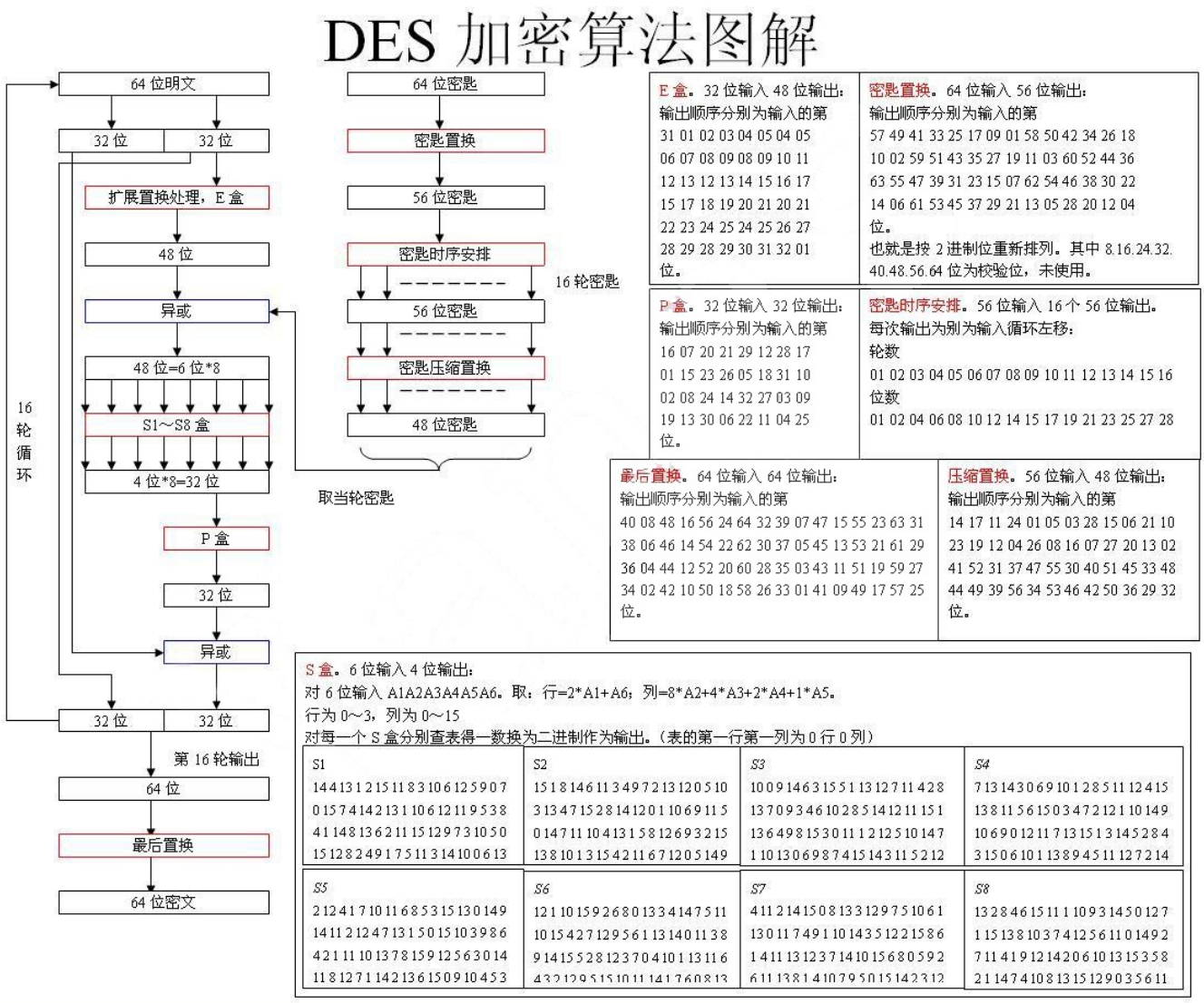
DES全称为Data Encryption Standard，是典型的块加密，其基本信息如下

- 使用64位密钥中的56位，剩余的8位要么丢弃，要么作为奇偶校验位
- 输入64位
- 输出64位。
- 明文经过16轮迭代得到密文。

给出一张简单的DES 流程图。


Input (64 bits)
Key (64 bits)
Initialization
Initial Permutation
Key Permutation
Left Half (32 bits)
Right Half (32 bits)
Left Half (28 bits)
Right Half (28 bits)
Binary Rotation
Binary Rotation
Round 1
Cipher Function
Permutation
Sub-key #1 (48 bits)
Binary Rotation
Binary Rotation
Round 2
Cipher Function
Permutation
Sub-key #2 (48 bits)
Binary Rotation
Binary Rotation
Round 16
Cipher Function
Permutation
Sub-key #16 (48 bits)
Finalization
Final Permutation
Output (64 bits)

再给一张比较详细的图。



其中

- S盒的设计标准并未给出。