

单表代换加密

通用特点

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——对应。所以说，一般有以下两种方式进行破解

- 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，<http://quipqiup.com/>

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）加密时会把明文中的**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](#)
- 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。**

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。

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

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
- <http://planetcalc.com/1434/>
- <http://www.qqxiuzi.cn/bianma/ROT5-13-18-47.php>

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
```

下面给出一个例子：

```
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。

工具

- <http://www.practicalcryptography.com/ciphers/classical-era/atbash-cipher/>

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 : phqgiumeaylnofdxjkrvcstzwb
```

a 对应 p, d 对应 h, 以此类推。

```
明文: the quick brown fox jumps over the lazy dog  
密文: cei jvaql hkdtef 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 $26!26!$ ，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

- <http://quipqiup.com/>

仿射密码

原理

仿射密码的加密函数是 $E(x)=(ax+b)(\text{mod}m)$ ，其中

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

解密函数是 $D(x)=a^{-1}(x-b)(\text{mod}m)$ ，其中 a^{-1} 是 aa 在 Z_m 群的乘法逆元。

下面我们以 $E(x)=(5x+8)\text{mod}26$ 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
$y\text{mod}26$	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 **IHHWCSWFRCP**。

对于解密过程，正常解密者具有 a 与 b ，可以计算得到 a^{-1} 为 21，所以其解密函数是 $D(x)=21(x-8)(\text{mod}26)$ ，解密如下

密文	I	H	H	W	V	C	S	W	F	R	C	P
yy	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
$x\text{mod}26$	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

$$\phi(26)=\phi(2)\times\phi(13)=12\quad\phi(26)=\phi(2)\times\phi(13)=12$$

算上 b 的偏移可能，一共有可能的密钥空间大小也就是

$$12\times 26=312\quad 12\times 26=312$$

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 y_1, y_2 即可进行解密。那么我们还可以知道

$$y_1=(ax_1+b)(\text{mod } 26)\quad y_2=(ax_2+b)(\text{mod } 26)\quad y_1=(ax_1+b)(\text{mod } 26)\quad y_2=(ax_2+b)(\text{mod } 26)$$

两式相减，可得

$$y_1-y_2=a(x_1-x_2)(\text{mod } 26)\quad y_1-y_2=a(x_1-x_2)(\text{mod } 26)$$

这里 y_1, y_2 已知，如果我们知道密文对应的两个不一样的字符 x_1 与 x_2 ，那么我们就可以很容易得到 a，进而就可以得到 b 了。

例子

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

$$c_1=a_1c+b_1\quad c_2=a_2c_1+b_2=a_1a_2c+a_2b_1+b_2=kc+dc_1=a_1c+b_1\quad c_2=a_2c_1+b_2=a_1a_2c+a_2b_1+b_2=kc+d$$

根据第二行的推导，我们可以得到其实 c_n 也是这样的形式，可以看成 $c_n=xc+yc_n=xc+y$ ，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以及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

```
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)]
plainedelta = ord(flag[1]) - ord(flag[0])
cipherdalte = encrypted[1] - encrypted[0]
a = gmpy.invert(plainedelta, 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

```
→ TWCTF2016-super_express git:(master) X python exploit.py
TWCTF{Faster_Than_Shinkansen!}
```

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。

Playfair

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。
 - 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。
 - 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）。
 - 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

Polybius

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
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。

ADFGX 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

例子

这里以安恒杯 9 月 Crypto 赛题 Go 为例，题目为：

密文：ilnllliikkninlekile

压缩包给了一行十六进制：

546865206c656e677468206f66207468697320706c61696e746578743a203130

请对密文解密

首先对十六进制进行 hex 解码，得到字符串："The length of this plaintext: 10"

密文长度为 20，而明文长度为 10，密文只有 "l","i","n","k","e" 这五个字符，联想到棋盘密码。

首先试一下五个字符按字母表顺序排列：

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

根据密文解密得：iytghpkqmq。

这应该不是我们想要的 flag 答案。

看来这五个字符排列不是这么排列的，一共有 5! 种情况，写脚本爆破：

```
import itertools

key = []
cipher = "ilnllliikkninlekile"

for i in itertools.permutations('ilnke', 5):
    key.append(''.join(i))

for now_key in key:
    solve_c = ""
    res = ""
    for now_c in cipher:
        solve_c += str(now_key.index(now_c))
    for i in range(0, len(solve_c), 2):
        now_ascii = int(solve_c[i])*5+int(solve_c[i+1])+97
        if now_ascii>ord('i'):
            now_ascii+=1
        res += chr(now_ascii)
    if "flag" in res:
```

```
print now_key,res
```

脚本其实就是实现棋盘密码这个算法，只是这五个字符的顺序不定。

跑出下面两个结果：

```
linke flagishere  
linek flagkxhdwd
```

显然第一个是我们想要的答案。

附上正确的密码表：

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

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

	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	明文
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	

密钥

明文: come greatwall

密钥: crypto

密文: efkt zferltzn

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，我们可以使用卡斯基试验和弗里德曼试验来获取。

卡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: DYDUXRMHTVDVNQDQNWYDUXRMHARTJGWNQD

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。可以参考

<http://www.practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-vigenere-cipher/>。

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。

Nihalist

原理

Nihalist 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 $n \times n$ 的矩阵相乘，再将得出的结果模 26。注意用作加密的矩阵（即密钥）在 Z_{26} 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。

$\begin{bmatrix} 0 & 2 & 19 \end{bmatrix}$

假设密钥为：

$\begin{bmatrix} 6 & 2 & 4 & 1 & 1 & 3 & 1 & 6 & 1 & 0 & 2 & 0 & 1 & 7 & 1 & 5 \end{bmatrix}$

加密过程为：

$\begin{bmatrix} 6 & 2 & 4 & 1 & 1 & 3 & 1 & 6 & 1 & 0 & 2 & 0 & 1 & 7 & 1 & 5 \end{bmatrix} \begin{bmatrix} 0 & 2 & 19 \end{bmatrix} = \begin{bmatrix} 6 & 7 & 2 & 2 & 2 & 3 & 1 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 5 & 1 & 4 & 7 \end{bmatrix} \pmod{26}$
 $\begin{bmatrix} 0 & 2 & 19 \end{bmatrix} = \begin{bmatrix} 6 & 7 & 2 & 2 & 2 & 3 & 1 & 9 \end{bmatrix} \pmod{26}$

密文即为

密文：POH

工具

- <http://www.practicalcryptography.com/ciphers/hill-cipher/>
- 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
- 未知关键词
 - <http://www.practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-autokey-cipher/>
 - `tools` 文件夹下 `break_autokey.py`，待完成。

其它类型加密

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

- <http://rumkin.com/tools/cipher/baconian.php>

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

工具

- <https://www.qqxiuzi.cn/bianma/zhalanmima.php>

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)

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: 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 3	o 4	w 7	a 1	r 5	e 2	u 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 uwmted cfseg erjez

工具

- <http://www.practicalcryptography.com/ciphers/classical-era/columnar-transposition/> 行列数相等

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

8842101220480224404014224202480122

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
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。

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

[illegible]

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 ¶

- <https://www.splitbrain.org/services/ook>

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

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

我们举一个例子，如明文为 x marks the spot，那么密文如下

➤ ◻ ◻ ◻ ◻ ◻ ➤ ◻ ◻ ◻ ◻ ◻ ◻

X M A R K S T H E S P O T

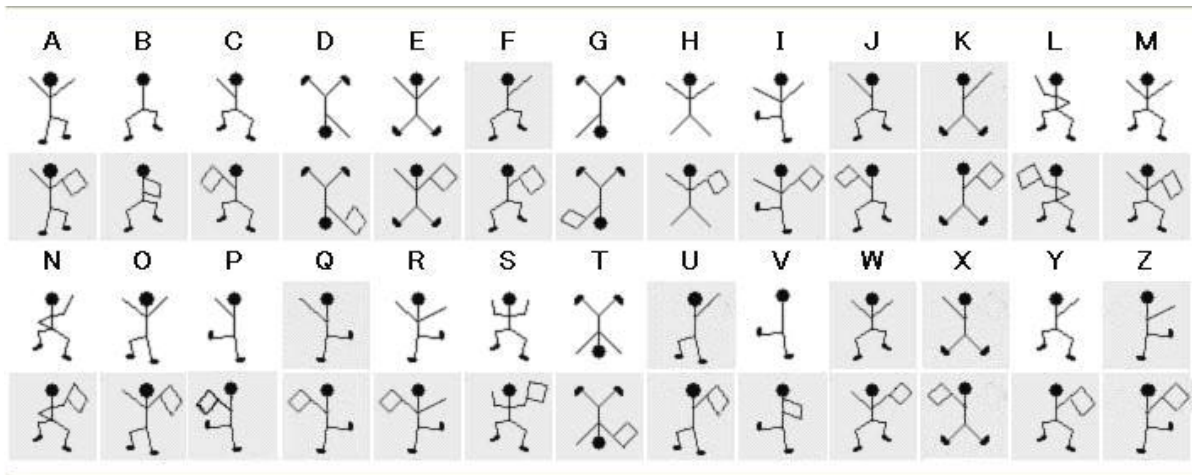
工具 ¶

- <http://www.simonsingh.net/The Black Chamber/pigpen.html>

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

手机键盘密码



简单的替换密码。

采用坐标方法加密。

例：

21 = A; 22 = B; 94 = Z.

特点：第一项数字为2-9, 第二项为1-4.

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。

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	~	!	@	#	\$	%	^	&	*	()	_	+	
2	`	1	2	3	4	5	6	7	8	9	0	-	=	\
3	Q	W	E	R	T	Y	U	I	O	P	{	}		
4	q	w	e	r	t	y	u	i	o	p	[]		
5	A	S	D	F	G	H	J	K	L	:	"			
6	a	s	d	f	g	h	j	k	l	;	'			
7	Z	X	C	V	B	N	M	<	>	?				
8	z	x	c	v	b	n	m	,	.	/				

键盘密码加密的原理同棋盘密码，只是利用了键盘作为方阵。

例：
密文：
87 34 112
55 47 87 410

明文：
mR_Gump

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



- 一. 电脑键盘密码(坐标法)
- 法1. (蓝色, 黄色) 即蓝色框内数字为横坐标, 黄色框内数字为纵坐标. (1, 1) = Q; (1, 2) = W; (2, 1) = A.
- 法2. (黄色, 蓝色) 即黄色框内数字为横坐标, 蓝色框内数字为纵坐标. (1, 1) = Q; (1, 2) = A; (2, 1) = W.
- 区分 如果所有的数字纵坐标为X>3, 则为法1. 反之则为法2. (特殊情况特殊处理)
- 如果所有的数字横坐标为X>3, 则为法2. 反之则为法1. (特殊情况特殊处理)

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。

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

- 二. QWE=ABC
- 即把键盘上的字母按顺序对应ABC.

注意: 红色的为明码 (即你手中的密码)

黑色的就是对应的密码了.

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。

OCTF 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。

题目

- 实验吧 奇怪的短信

总结

古典密码分析思路

CTF 中有关古典密码的题目，通常是根据密文求出明文，因此采用**唯密文攻击**居多，基本分析思路总结如下：

1. 确定密码类型：根据题目提示、加密方式、密文字符集、密文展现形式等信息。
2. 确定攻击方法：包括直接分析、蛮力攻击、统计分析等方法。对于无法确定类型的特殊密码，应根据其密码特性选用合适的攻击方法。
3. 确定分析工具：以在线密码分析工具与 Python 脚本工具包为主，以离线密码分析工具与手工分析为辅。

以上唯密文攻击方法的适用场景与举例如下：

攻击方法	适用场景	举例
直接分析法	由密码类型可确定映射关系的代换密码	凯撒密码、猪圈密码、键盘密码等
蛮力攻击法	密钥空间较小的代换密码或置换密码	移位密码、栅栏密码等
统计分析法	密钥空间较大的代换密码	简单替换密码、仿射密码、维吉尼亚密码等

实验吧 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，题目中还有还有栅栏与 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{*****} *****
POR4dnyTLHBfbwXAAZhe}ocZR3Cxcftw9
```

解法一：

首先，我们先来分析一下 t 的构成 $t[i][j]=s[i+j]+s[:i+j] \setminus t[i][k]=s[i+k]+s[:i+k]$

$t[i][j][k]t[i][j][k]$ 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 $\text{len}(s)$ 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{We1c0me_to_SECCON_CTF_2017}
```

解法二

关于此题的分析：

1. 考虑到在程序正常运行下，数组访问不会越界，我们在讨论时做以下约定：
 $arr[index] \Leftrightarrow arr[index \% len(arr)]$

2. 关于 python 程序中定义的 `l(offset,arr)` 函数，发现以下等价关系： $l(offset,arr)[index] \Leftrightarrow arr[index+offset] \Leftrightarrow l(offset,arr)[index] \Leftrightarrow arr[index+offset]$
3. 关于 python 的 main 函数中三维矩阵 t 的定义，发现以下等价关系： $t[a][b][c] \Leftrightarrow l(a+b,s)[c] \Leftrightarrow t[a][b][c] \Leftrightarrow l(a+b,s)[c]$
4. 综合第 2 第 3 点的观察，有如下等价关系： $t[a][b][c] \Leftrightarrow s[a+b+c] \Leftrightarrow t[a][b][c] \Leftrightarrow s[a+b+c]$
5. 我们将 s 视为一种编码格式，即：编码过程 `s.find(x)`，解码过程 `s[x]`。并直接使用其编码结果的数字替代其所代指的字符串，那么加密过程可以用以下公式表示：
6. $e = f + k_1 + k_2$
7. 其中，e 是密文，f 是明文，k1 与 k2 是通过复制方法得到、与 f 长度一样的密钥，**加法是向量加**。

所以我们只需要通过计算 `k1+k2`，模拟密钥，即可解密。关于此题的解密 python 脚本：

```
# exp2.py
enc_str = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz_{}'
dec_dic = {k:v for v,k in enumerate(enc_str)}
encrypt = 'POR4dnyTLHBfwbxAAZhe}}ocZR3Cxcftw9'
flag_bg = 'SECCON{*****}'

sim_key = [dec_dic[encrypt[i]]-dec_dic[flag_bg[i]] for i in range(7)] # 破解模拟密钥
sim_key = sim_key + sim_key[::-1]

flag_ed = [dec_dic[v]-sim_key[k%14] for k,v in enumerate(encrypt)] # 模拟密钥解密
flag_ed = ''.join([enc_str[i%len(enc_str)] for i in flag_ed]) # 解码
print(flag_ed)
```

得到明文如下：

```
$ python exp2.py
SECCON{we1c0me_to_SECCON_CTF_2017}
```

消失的三重密码

密文

```
of zit kggd zitkt qkt ygxx ortfzoeqs wqlatzwqssl qfr zvg ortfzoeqs yggzwqssl. fgv
oy ngx vqfz zg hxz zitd of gft soft.piv dgfn lgsxzogfl qkt zitkt? zohl:hstqlt
eiqfut zit ygkd gy zit fxdwtk ngx utz.zit hkgukqddtkl!
```

使用 quipquip 直接解密。