



# 密码学

Cryptography

极夜酱

# 目录

<b>1</b>	<b>古典密码学</b>	<b>1</b>
1.1	密码学 . . . . .	1
1.2	凯撒密码 . . . . .	3
1.3	单表密码 . . . . .	5
1.4	维吉尼亚密码 . . . . .	8
1.5	一次性密码本 . . . . .	11

# Chapter 1 古典密码学

## 1.1 密码学

### 1.1.1 密码学 (Cryptography)

密码学是一种用来混淆的技术，它希望将正常的、可识别的信息转变为无法识别的信息。

密码学算法可以将明文 (plaintext) 加密 (encrypt) 为密文 (ciphertext)，也可将密文解密 (decrypt) 为明文。因此密码编码者 (cryptographer) 需要设计更加安全的加密算法，而破译者 (cryptanalyst) 的目的就是破解这些加密算法。

算法的安全性取决于破解的难度。例如一个旋转密码锁有 40 个数字，如果密码是一个 3 元组 (如  $5R, 12L, 7R$ )，那么可能的密码组合有  $40^3 = 64000$  种。假设尝试一种组合需要花费 10 秒，一共需要大约 178 小时才能尝试完。

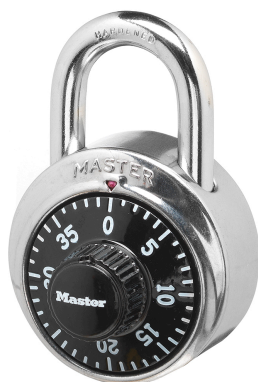


图 1.1: 旋转密码锁

如果密码的组合是一个 4 元组，那么可能的密码组合有  $40^4 = 2560000$  种。假设尝试一种组合需要花费 13 秒，一共需要大约 9244 小时才能尝试完。

大部分商业的加密算法是公开的，而军用的加密算法是保密的。

### 1.1.2 安全性

加密算法的安全性分为：

1. 无条件安全性 (unconditionally secure)：即使密码分析者拥有无限的计算资源和密文，都没有足够的信息恢复出明文。事实上，只有一次一密乱码本，才是不可破的，对于实际应用的密码算法都是可破的。在实际中，无条件安全的系统是不存在的。
2. 计算安全性 (Computationally secure)：当破解所花费的代价远超于被加密信息的生命周期和本身的价值时，破解就失去了意义。

## 1.2 凯撒密码

### 1.2.1 凯撒密码 (Caesar Cipher)

凯撒密码是一种最简单的加密算法，它是由罗马帝国的凯撒 (Julius Caesar) 发明的。

凯撒加密将明文中的字母用另一个字母来替换，替换的规则是将字母向后移动 3 位。例如，明文中的 A 将被替换为 D，明文中的 B 将被替换为 E，依此类推。

- 明文: meet me after the party
- 密文: phhw ph diwhu wkh sduwb

破解的方法也很容易，只需要将密文的字母向前移动 3 位即可。

改进后的凯撒加密不再采用固定移动 3 位的策略，而是可以移动任意位数。对于英文字母的加密而言，一个字母可能被替换为另外的 25 个字母之一。通过暴力枚举每个移位，还是可以破解凯撒加密。

#### 凯撒密码 (加密)

```
1 def encrypt(plaintext, shift=3):
2     shift %= 26
3     ciphertext = ""
4
5     for c in plaintext:
6         if not c.isalpha():
7             ciphertext += c
8             continue
9         if c.isupper():
10            ciphertext += chr((ord(c) + shift - 65) % 26 + 65)
11        else:
12            ciphertext += chr((ord(c) + shift - 97) % 26 + 97)
13
14    return ciphertext
```

## 凯撒密码（解密）

```
1 def decrypt(ciphertext, shift=3):
2     plaintext = ""
3
4     for c in ciphertext:
5         if not c.isalpha():
6             plaintext += c
7             continue
8         if c.isupper():
9             plaintext += chr((ord(c) - shift - 65) % 26 + 65)
10        else:
11            plaintext += chr((ord(c) - shift - 97) % 26 + 97)
12
13    return plaintext
```

## 1.3 单表密码

### 1.3.1 单表密码 (Monoalphabetic Cipher)

单表密码并不像凯撒密码一样仅仅是将字母后移，而是随机地将一个字母替换为另一个字母。例如字母的替换规则如下：

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

- 明文：if we wish to replace letters
- 密文：WI RF RWAJ UH YFTSDVF SFUUFYA

单表密码一共有  $25! = 1.55 \times 10^{25}$  种可能的替换规则，但是它并不安全。由于人类语言的特性，不同的字母的使用频率是不同的。

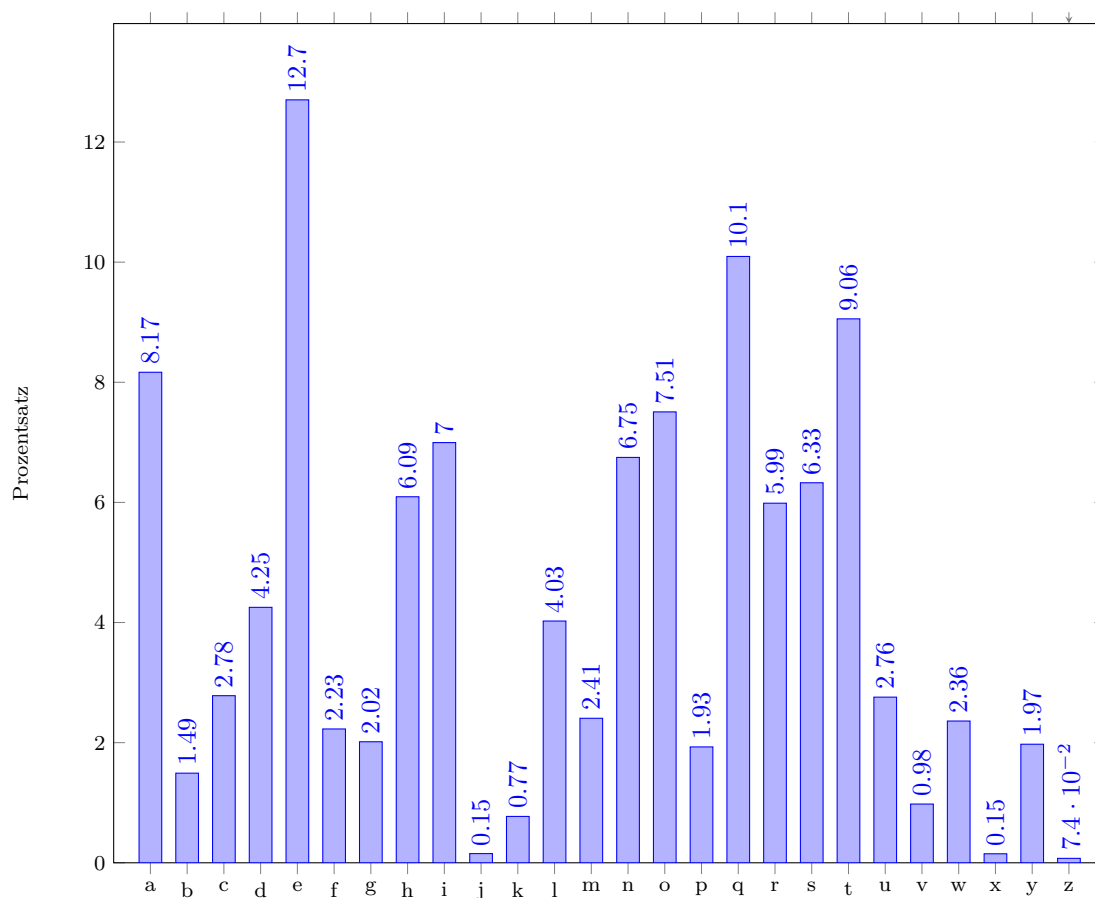


图 1.2: 字母使用频率

例如有这么一段密文：

UZ QSO VUOHXMOPV GPOZPEVSG ZWSZ OPFPESX UDBMETSX  
AIZ VUEPHZ HMDZSHZO WSFP APPD TSVP QUZW YMXUZUHSX  
EPYEPOPDZSZUFPO MB ZWP FUPZ HMDJ UD TMOHMQ

### 字母出现频率

```
1 import re
2
3 def get_frequency(text):
4     frequency = {}
5     for c in text:
6         if c in frequency:
7             frequency[c] += 1
8         else:
9             frequency[c] = 1
10    return frequency
11
12
13 def main():
14     with open("ciphertext.txt") as file:
15         text = file.readlines()
16         text = "".join(text)
17         # remove all non-alphabetic characters
18         text = re.sub("[\n\r]", '', text)
19
20         print(get_frequency(text))
21
22
23 if __name__ == '__main__':
24     main()
```

这段密文的字母使用频率如下：

{'U': 10, 'Z': 14, 'Q': 3, 'S': 10, 'O': 9, 'V': 5, 'H': 7, 'X': 5, 'M': 8, 'P': 16, 'G': 2,



'E': 6, 'W': 4, 'F': 4, 'D': 6, 'B': 2, 'T': 3, 'A': 2, 'I': 1, 'Y': 2, 'J': 1}

其中 P 和 Z 的出现次数最多，分别为 16 和 14 次。可以猜测 P 和 Z 是对应明文中最常用的字母，因此 P 和 Z 非常有可能是 e 和 t。

Ut QSO VUOHXMOeV GeOteEVSG tWSt OeFeESX UDBMETSX  
AIt VUEeHt HMDtSHtO WSFe Aeed TSVe QUtW YMXUtUHSX  
EeYEEeOeDtStUFeO MB tWe FUet HMDJ UD TMOHMQ

通过英语中单词的组合，可以猜测 S 对应 a、W 对应 h，因此 ZWSZ 对应 that、ZWP 对应 the。

Ut QaO VUOHXMOeV GeOteEVaG that OeFeEaX UDBMETaX  
AIt VUEeHt HMDtaHtO haFe Aeed TaVe QUth YMXUtUHax  
EeYEEeOeDtatatUFeO MB the FUet HMDJ UD TMOHMQ

依次类推，可以破解整个密文：

it was disclosed yesterday that several informal  
but direct contacts have been made with political  
representatives of the viet cong in moscow

## 1.4 维吉尼亚密码

### 1.4.1 维吉尼亚密码 (Vigenere Cipher)

维吉尼亚密码是在凯撒密码的基础上产生的一种加密方法，它将凯撒密码的全部 25 种位移排序为一张表，与原字母序列共同组成 26 行 26 列的密码表。

	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

表 1.1: 维吉尼亚密码表

除了密码表，还必须有一个密钥 (key)。密钥由字母组成，最少一个字母，最多可与明文字母数相等。如果密钥只有 1 个字母，相当于凯撒密码。

### 1.4.2 加密

例如：

- 明文：I Love You
- 密钥：OK

首先，如果密钥长度小于明文长度（非字母忽略），就重复拼接密钥，使其与明文长度相等。

- 明文：I Love You
- 密钥：OKOKOKOK

接着根据密码表进行加密。明文第一个字母是 I，密钥的第一个字母是 O，因此在查找密码表的 I 行 O 列，得到加密后的字母 W。同理，第二个字母为位于 L 行 K 列的字母 V。

依次类推，得到密文：W VCFS ICE。

### 1.4.3 解密

例如：

- 密文：PWZRNZBZEANQKBUHNLNB
- 密钥：WIND

首先把密钥重复拼接，直到和密文长度相同。

- 密文：PWZRNZBZEANQKBUHNLNB
- 密钥：WINDWINDWINDWINDWIND

根据密文的第一个字母 P，在密码表中找到 P 行，在 P 行找出值为 W 的列，沿着该列向上找到该列为 T，因此 P 解密后可以得到 T。

根据相同的方法，可以得到明文：TOMORROWISANOTHERDAY。

### 维吉尼亚密码

```
1 import re
2
3 def generate_key(plaintext, key):
4     n = len(plaintext) - len(key)
5     for i in range(n):
6         key += key[i % len(key)]
7     return key
8
9
10 def encrypt(plaintext, key):
11     # remove all non-alphabetic characters
12     plaintext = re.sub("[ \n\r]", '', plaintext).upper()
13     key = generate_key(plaintext, key)
14
15     ciphertext = ""
16     for i in range(len(plaintext)):
17         ciphertext += chr((ord(plaintext[i]) + ord(key[i])) % 26 + 65)
18     return ciphertext
19
20
21 def decrypt(ciphertext, key):
22     # remove all non-alphabetic characters
23     ciphertext = re.sub("[ \n\r]", '', ciphertext).upper()
24     key = generate_key(ciphertext, key)
25
26     plaintext = ""
27     for i in range(len(ciphertext)):
28         plaintext += chr((ord(ciphertext[i]) - ord(key[i])) % 26 + 65)
29     return plaintext
```

## 1.5 一次性密码本

### 1.5.1 一次性密码本 (One-Time Pad)

一次性密码本算法所用的密钥是一次性的，因此不会出现因为密钥泄露导致之前的加密内容被破解的情况。即使密钥被泄露了，也只会影响一次通信过程。

计算机在传输数据时，会将文字转换为对应的二进制编码。通过随机生成一个与明文的二进制编码长度相同的密钥，一次性密码本将明文和密钥进行异或 (XOR) 运算，得到密文。

由于 XOR 运算的特性，即比特位相同为 0，不同为 1：

- $0 \text{ XOR } 0 = 0$
- $0 \text{ XOR } 1 = 1$
- $1 \text{ XOR } 0 = 1$
- $1 \text{ XOR } 1 = 0$

同时另一个 XOR 的特性就是可逆性，即  $A \text{ XOR } B = C$ ，则  $C \text{ XOR } B = A$ 。那么通过将密文和密钥再次异或操作就可以得到原文。

### 1.5.2 优势

虽然一次性密码本非常简单，但是一次性密码本是无法破译的，这个破译并不是指现有的计算能力不够，而是指即使拥有无穷大的计算能力也无法破译。

假如你拿到了一段 128 bits 的密文，通过遍历等长的密钥进行暴力破解，将会生成  $2^{128}$  个原文，即原文的所有排列组合。

在这些排列组合中，可能会出现一些有意义的文字，但是你不能确定这些文字是否就是原文，因为在所有的排列组合中会产生大量有意义的问题。

因此破解一次性密码本算法是无意义的，就像是知道了原文的长度，然后自己构造了一个这个长度的原文。

### 1.5.3 缺陷

既然一次性密码本这么好，那么为什么我们在实际中很少用到呢？

- 密钥太长：密钥的长度与原文相同，如果原文很大，那么对应的密钥也很大。
- 密钥无法重用：每个密钥只用一次，既是优点也是缺点。这意味着每次都要不停地更换密钥，增加了复杂性。
- 密钥的配送：目标端如果想解密就必须拿到密钥，如果能够机密地传输密钥给目标端，那为什么不直接将原文机密地传送给目标端呢？
- 密钥的保存：每一个明文都需要保存一个同样长度的密钥。