密码学

Cryptography

极夜酱

目录

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 加密算法

一种最简单的加密算法就是换位密码/转置密码（Transposition Cipher），它将明文中的字符以某种规则移动位置，形成密文。

例如栅栏密码（Rail Fence Cipher），将明文的字符在行与行之间交替，形成密文。

比如有一段明文“meet me after the party”，将其分成两行交错排列，得到：

```
m e m a t r h p r y
  e t e f e t e a t
```

这种加密方式仅仅只是打乱了顺序，并没有改变明文中的字符和出现频率，因此很容易破解。

1.1.3 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）

```python
def encrypt(plaintext, shift=3):
    shift %= 26
    ciphertext = ""

    for c in plaintext:
        if not c.isalpha():
            ciphertext += c
            continue
        if c.isupper():
            ciphertext += chr((ord(c) + shift - 65) % 26 + 65)
        else:
            ciphertext += chr((ord(c) + shift - 97) % 26 + 97)

    return ciphertext
```

凯撒密码（解密）

```
def decrypt(ciphertext, shift=3):
    plaintext = ""

    for c in ciphertext:
        if not c.isalpha():
            plaintext += c
            continue
        if c.isupper():
            plaintext += chr((ord(c) - shift - 65) % 26 + 65)
        else:
            plaintext += chr((ord(c) - shift - 97) % 26 + 97)

    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只是将字母后移，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。

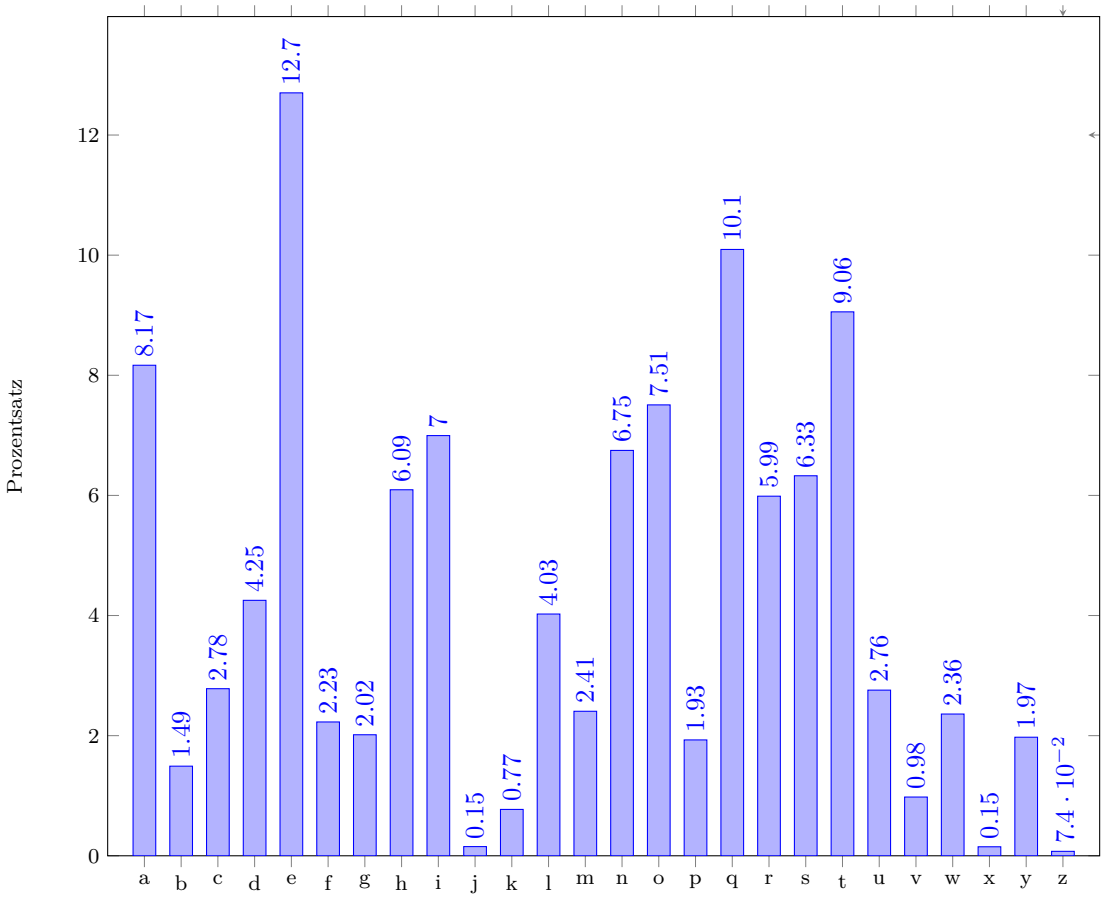

图 1.2: 字母使用频率

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

```python
import re

def get_frequency(text):
    frequency = {}
    for c in text:
        if c in frequency:
            frequency[c] += 1
        else:
            frequency[c] = 1
    return frequency


def main():
    with open("ciphertext.txt") as file:
        text = file.readlines()
        text = "".join(text)
        # remove all non-alphabetic characters
        text = re.sub("[ \n\r]", '', text)

        print(get_frequency(text))


if __name__ == '__main__':
    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
EeYEeOeDtStUFeO 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
EeYEeOeDtatUFeO 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

```python
import re

def generate_key(plaintext, key):
    n = len(plaintext) - len(key)
    for i in range(n):
        key += key[i % len(key)]
    return key


def encrypt(plaintext, key):
    # remove all non-alphabetic characters
    plaintext = re.sub("[ \n\r]", '', plaintext).upper()
    key = generate_key(plaintext, key)

    ciphertext = ""
    for i in range(len(plaintext)):
        ciphertext += chr((ord(plaintext[i]) + ord(key[i])) % 26 + 65)
    return ciphertext


def decrypt(ciphertext, key):
    # remove all non-alphabetic characters
    ciphertext = re.sub("[ \n\r]", '', ciphertext).upper()
    key = generate_key(ciphertext, key)

    plaintext = ""
    for i in range(len(ciphertext)):
        plaintext += chr((ord(ciphertext[i]) - ord(key[i])) % 26 + 65)
    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 XOR 0 = 0
- 0 XOR 1 = 1
- 1 XOR 0 = 1
- 1 XOR 1 = 0

同时另一个 XOR 的特性就是可逆性，即 A XOR B = C，则 C 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，但是你并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。

1.6 Playfair 密码

1.6.1 Playfair 密码（Playfair Cipher）

Playfair 密码是一种使用一个关键词方格来加密字符对的加密法，在 1854 年由 Charles Wheatstone 发明。曾在一战时期被英军所使用，二战时期澳大利亚所使用。

首先选取一个英文单词作为密钥，将这个单词（去除重复字母）填入一个 5×5 的矩阵中，剩余的位置按照字母表顺序填入方格中，其中 I 和 J 视作同一个字母。

例如选取单词 PLAYFAIR 作为密钥，生成的密钥矩阵如下：

$$\begin{bmatrix} P & L & A & Y & F \\ I & R & B & C & D \\ E & G & H & K & M \\ N & O & Q & S & T \\ U & V & W & X & Z \end{bmatrix}$$

假设需要加密的明文为 HELLO THERE，将明文中的字符两两配对。当一对中出现两个相同字母时，用 x 分隔。当最后一对只剩一个字母时，也是用 x 配对。

- 明文：HELLO THERE
- 配对：HE LX LO TH ER EX

对于每一对字母，如果它们在密钥矩阵中位于同一行，将它们替换为各自右边的字母，如 HE 加密为 KG。如果它们位于同一列，将它们替换为各自下面的字母，如 LO 加密为 RV。其它情况，将它们的列下标互换，如 LX 加密为 YV。

- 密文：KG YV RV QM GI KU

1.6.2 解密

假设目前截获了一份使用 Playfair 加密的密文和对应的原文，以及一个不完整的密钥矩阵，现在需要对另一份密文进行解密。

- 明文：THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
- 密文：RMAPLHBMUAWRVDPWHLISNPXGEGIRBEVZHNEV
- 待解密密文：RMTGNTEATG
- 不完整的密钥矩阵：

$$\begin{bmatrix} & & & A & T \\ & U & & B & C \\ & & I & & \\ N & & P & & S \\ & W & & & Z \end{bmatrix}$$

解密的方法就是根据已有的一组明文和密文，推算出完整的密钥矩阵。再根据密钥矩阵，反向利用 Playfair 的加密方法进行解密。

首先将已有的明文和密文进行两两匹配：

- 明文：TH EQ UI CK BR OW NF OX JU MP SO VE RT HE LA ZY DO GX
- 密文：RM AP LH BM UA WR VD PW HL IS NP XG EG IR BE VZ HN EV

可以发现在这些配对中，有些配对在明文和密文中出现了相同的字母：

- 明文：TH EQ UI CK BR OW NF OX JU MP SO VE RT HE LA ZY DO GX
- 密文：RM AP LH BM UA WR VD PW HL IS NP XG EG IR BE VZ HN EV

这种情况只有可能发现在被加密的两个字母出现在相邻的位置上，这样被加密时，使用它们各自右边或下面的字母，就会出现重复。

根据这个信息，可以推算出 V、Y、R、O 的位置。

$$\begin{bmatrix} & R & & A & T \\ & U & & B & C \\ & & I & & \\ N & O & P & & S \\ V & W & & Y & Z \end{bmatrix}$$

由于秘钥矩阵后，后面的字母都是按照字母表的顺序排列的，所以推算出一部分缺失的字母。

$$\begin{bmatrix} & R & & A & T \\ & U & & B & C \\ & & I & & \\ N & O & P & Q & S \\ V & W & X & Y & Z \end{bmatrix}$$

再次根据已有的明文和密文，可以推算出其它字母的位置。

- 明文：TH EQ UI CK BR OW NF OX JU MP SO VE RT HE LA ZY DO GX
- 密文：RM AP LH BM UA WR VD PW HL IS NP XG EG IR BE VZ HN EV

$$\begin{bmatrix} G & R & E & A & T \\ F & U & L & B & C \\ D & H & I & K & M \\ N & O & P & Q & S \\ V & W & X & Y & Z \end{bmatrix}$$

在破解出完整的密钥矩阵后，就可以利用 Playfair 的解密方法对密文 RMTGNTEATG 解密了。

- 密文：RM TG NT EA TG
- 明文：TH AT SG RE AT

Chapter 2 数论

2.1 最大公约数 / 最小公倍数

2.1.1 最大公约数（GCD, Greatest Common Divisor）

两个整数 a 和 b 的最大公约数 $gcd(a,b)$ 为能够同时整除 a 和 b 的最大整数。

例如：

- $gcd(24,36) = 12$
- $gcd(17,22) = 1$
- $gcd(500,128) = 4$

计算最大公约数的算法包括递归方法和欧几里得（Euclidean）算法/辗转相除法。

最大公约数

```
def gcd(a, b):
    if b == 0:
        return a
    return gcd(b, a % b)


def euclid_gcd(a, b):
    while b != 0:
        remainder = a % b
        a = b
        b = remainder
    return a
```

2.1.2 最小公倍数（LCD, Least Common Multiple）

两个整数 a 和 b 的最小公倍数 $lcm(a,b)$ 为能够同时被 a 和 b 整除的最小整数。

例如：

- $lcm(24,36) = 72$
- $lcm(17,22) = 374$
- $lcm(500,128) = 16000$

最小公倍数

```
1 def lcm(a, b):
2     return a * b // gcd(a, b)
```

2.2 同余定理

2.2.1 模算数（Modular Arithmetic）

当 $a \in \mathbb{Z}$、$M \in \mathbb{Z}^+$，那么将 a 除以 m 的余数记为 $a \bmod m$。

例如：

- $17 \bmod 5 = 2$
- $2001 \bmod 101 = 82$
- $-10 \bmod 3 = -1$

2.2.2 同余定理（Congruence Theorem）

当 $a \in \mathbb{Z}$、$b \in \mathbb{Z}$、$M \in \mathbb{Z}^+$，如果 m 能够整除 $a - b$，那么就称 a 和 b 对模 m 同余，记作 $a \equiv b \pmod{m}$。

因此，

$$a \equiv b \pmod{m} \leftrightarrow a \bmod m \equiv b \bmod m \tag{2.1}$$

例如：

- $17 \equiv 5 \pmod{6}$
- $17 \equiv 12 \pmod{5}$
- $24 \equiv 3 \pmod{7}$

当 $a \equiv b \pmod{m}$、$c \equiv d \pmod{m}$，同余定理满足以下性质：

- $a + c \equiv b + d \pmod{m}$
- $ac \equiv bd \pmod{m}$

Exercise

因为 $7 \equiv 2 \pmod 5$、$11 \equiv 1 \pmod 5$。

(a) $7 + 11 \pmod 5 = 2 + 1 \pmod 5 = 3$

(b) $7 \cdot 11 \pmod 5 = 2 \cdot 1 \pmod 5 = 2$

Exercise

(a) $7^{10} \bmod 5 = 2^{10} \bmod 5 = 4$

(b) $7^{100} \bmod 3 = 1^{100} \bmod 5 = 1$