欧姆社学习漫画

漫画密码

〔日〕三谷政昭 佐藤伸一/著
〔日〕Hinoki Iderou/漫画绘制
〔日〕VERTE/漫画制作
李光东 李 纯 刘敏亮/译

科学出版社
www.sciencep.com

❀前　言❀

当今是以因特网为核心的网络化信息社会。活用网页发布信息，使用电子邮件进行交流，以及网上购物与电子银行的普及，都极大地方便了我们的日常生活。

但是，在享受网络时代给我们带来的便利的同时，“网络安全”、“信息安全”、“个人信息保护”、“密码”这些我们听起来有些厌烦的词汇，不断冲击着我们的生活。那么，到底为什么会出现这种情况呢？

我们使用网络的时候，会进行各种各样的信息交流。在这些信息当中，就包含着不能被人所知的私密信息。比如，信用卡账号、银行账号、病例、贷款的金额、电子邮箱地址等，这些都是不能轻易泄露给他人、必须要进行“信息保护”的私密信息。由于私密信息被恶意利用而产生不良后果的事件时有发生，所以，如何保护信息的安全，就毫无疑问地成为网络信息时代最重要的研究课题。在这充斥着不安因素的网络信息社会，为了识别信息的真伪，防止冒名诈骗，信息的伪造、篡改、窃密等犯罪的发生，安心、安全地使用各种网络化信息服务，“密码”成为了保障信息安全的基本技术。

近年来，密码技术得到了突飞猛进的发展。它不仅仅只局限于信息情报安全专家们所研究的领域，对于使用网络服务的大众来说，密码技术也理所当然地成为了不可缺少的知识。

那么，密码技术是如何构成的呢？如何才能实现信息安全化，保护个人信息不被窃取呢？

本书以漫画故事为基础，详细讲解了密码技术的构成和作用。同时，为了理解密码技术而必须掌握的不可缺少的高等数学知识，我们也将进行简明扼要的讲解。亲爱的读者朋友，不管你是谁，都可以一边开心地看故事，一边轻松地学习密码知识。当然，在故事中会出现一些密码迷题，大家可以一边踏踏实实地学习，一边享受破解这些迷题的乐趣。

在读完本书的同时，希望大家都能迅速掌握基本的密码技术与信息情报安全的基础知识。

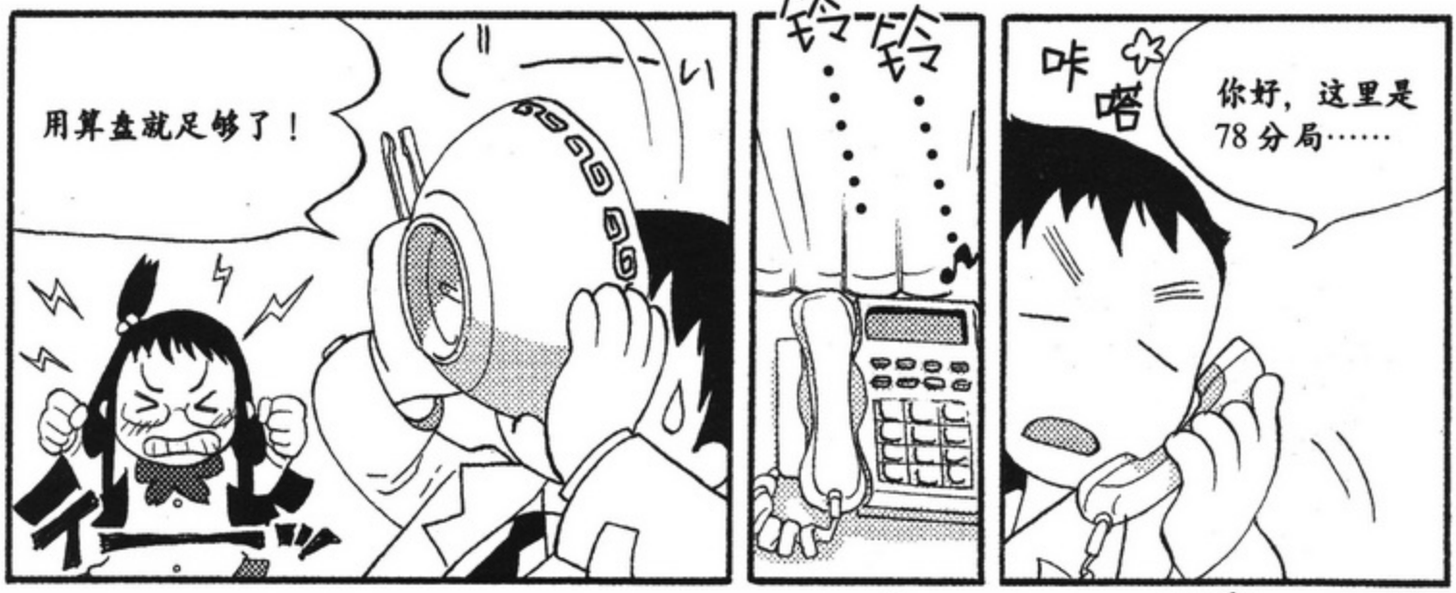
最后，向出版本书的欧姆社开发局的诸位，以及担任绘画的 Hinoki Iderou 先生，表示衷心的感谢。

著　者

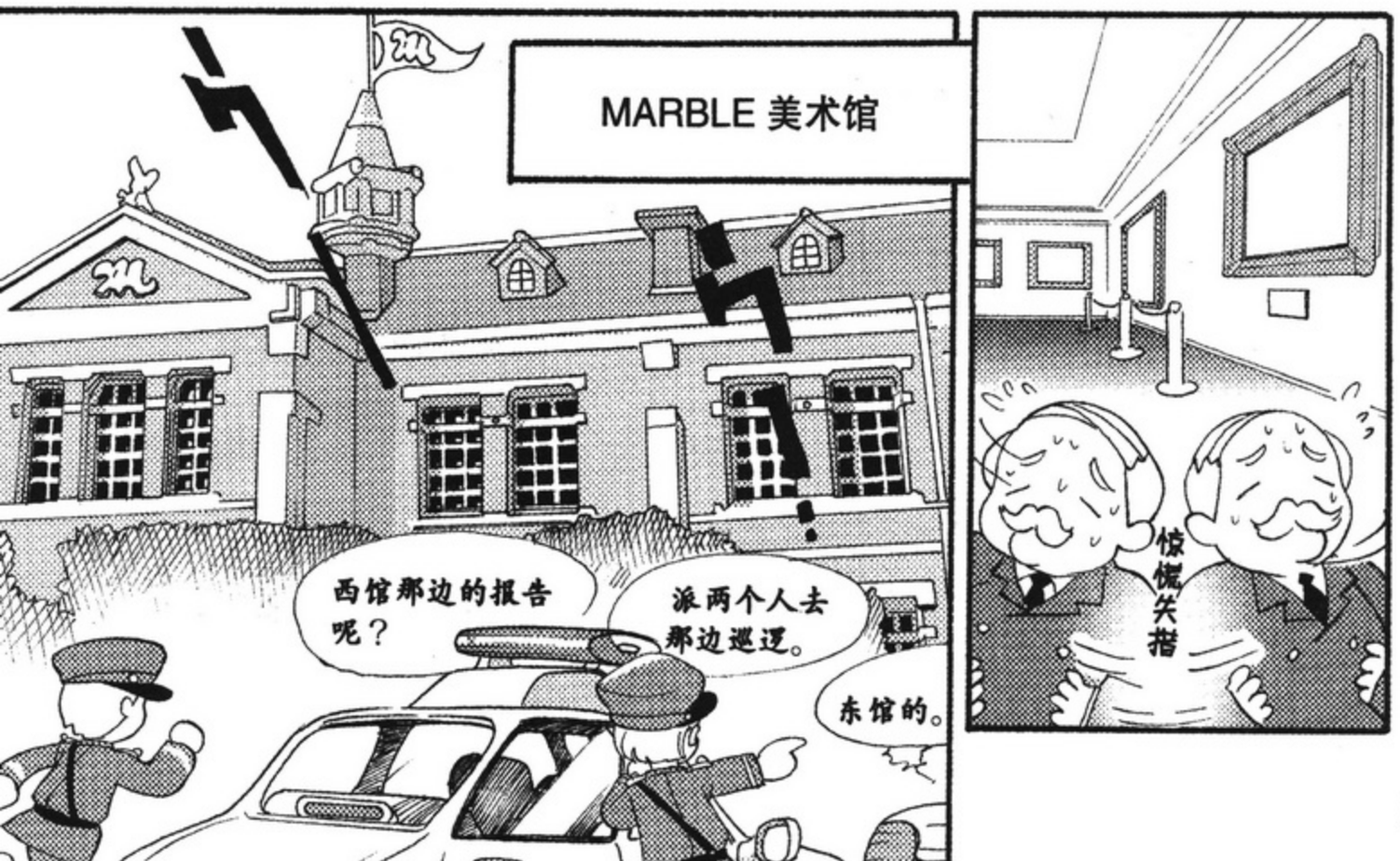
❀目　录❀

序　章


某省某市78分局
交通安
事科
咨询处
目黑留香
哥哥，给我买嘛！
目黑顺警官
不行，不行，电脑对高中生来说太奢侈了！
嘶嘶
但是对学习数学很有帮助啊！
$ax+by=gcd(a,b)$
$a^{p-1}=1(\mathrm{mod}\,p)$
$\varphi(p)=p-1$
x
y
……
我很想
要啊！


用算盘就足够了！
铃铃……
咔嗒
你好，这里是78分局……


什么——
被盗了？
哇！好胜啊！


MARBLE 美术馆
西馆那边的报告呢？
派两个人去那边逛逛。
东馆的。
惊慌失措


那幅《微笑的麦当娜》
价值3亿多日元啊！
是啊！
我也吓了一跳……
你们警察到底有没有好好看管啊？
嗯……
应该是被慎重保管起来的画，不知不觉就不见了，是吧？
确定警戒没有什么漏洞？
当然！
巡逻换班的时候，
也都确认过接头暗号！
山！
川！
正！
确！
嗯，嗯，很好！
呜……
小学生水平……

画的保管地点都是用密码写的，其他人应该不会知道的。

baomiguanchangmisuoshi midiwumicangku

好啊！
太完美啦！

太过分了……

那样的防范，根本起不到任何作用的！

咔嚓

你是谁啊？

咔嚓

我是每晚报社的记者米田理绪！


连报社都知道这件事了？
那你说说，我们的防范措施哪里做得不好？
口令和密码都太简单了！
你……你说什么？
那幅画的保管场所是第五仓库吧?
baomiguanchangmisuoshi
midiwumicangku
咻 咻
这是谜字游戏，应把所有的"mi"去掉来猜。
你怎么会知道？
原来你就是罪犯！
才，才不是呢！
那么简单的密码谁都能破解的吧！


啊？
真的吗？
目黑警官！
还是尽快把罪犯
抓住吧！
哇哇
罪犯……罪犯是……
罪犯就是
那家伙！
啊？
是吗？
你被逮捕啦！
ガチョン！
天呀！


不是馆长，是馆长身后那面墙！
咦？
Ms.Cipher♡
这里本来不应该有画的……

怪盗希芙到此，
画我拿走了！
下次再来取 VDVIRCU。
晚安♥


这个是密码啊！
瞄示了下次他要偷的东西！
画我拿走了！
下次再来取 VDVIRCU。
VDVIRCU
到底是什么？
因为我的英语不太好……
这个和删字密码不同，好像删掉哪个字都不能变成完整词汇。
VDVIRCU
又不是侦探小说，学习密码这玩意真的有用吗？
要是这样的话，先学习密码知识再来破解怪盗希芙的留言吧！
燃起斗志

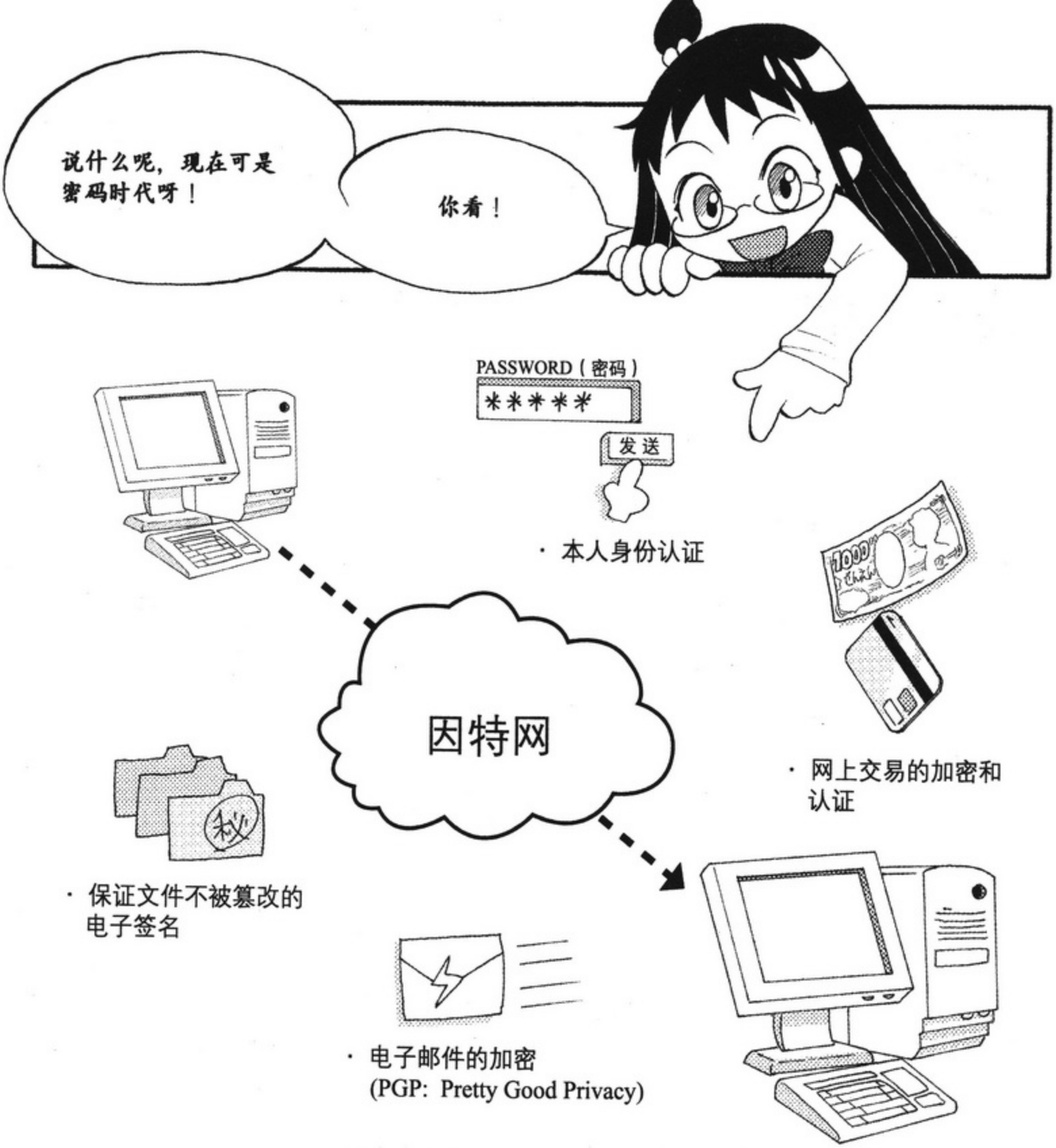
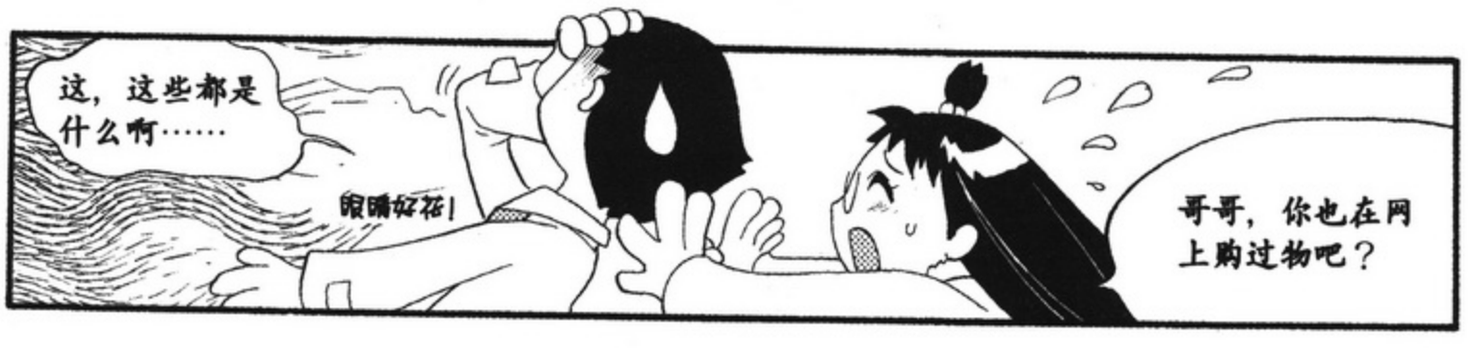


图 0.1　现代密码在社会中的广泛应用

如图 0.1 所示，在计算机和通信发达的现代社会，密码已经成为防止信息被篡改、盗取等方面不可或缺的技术。



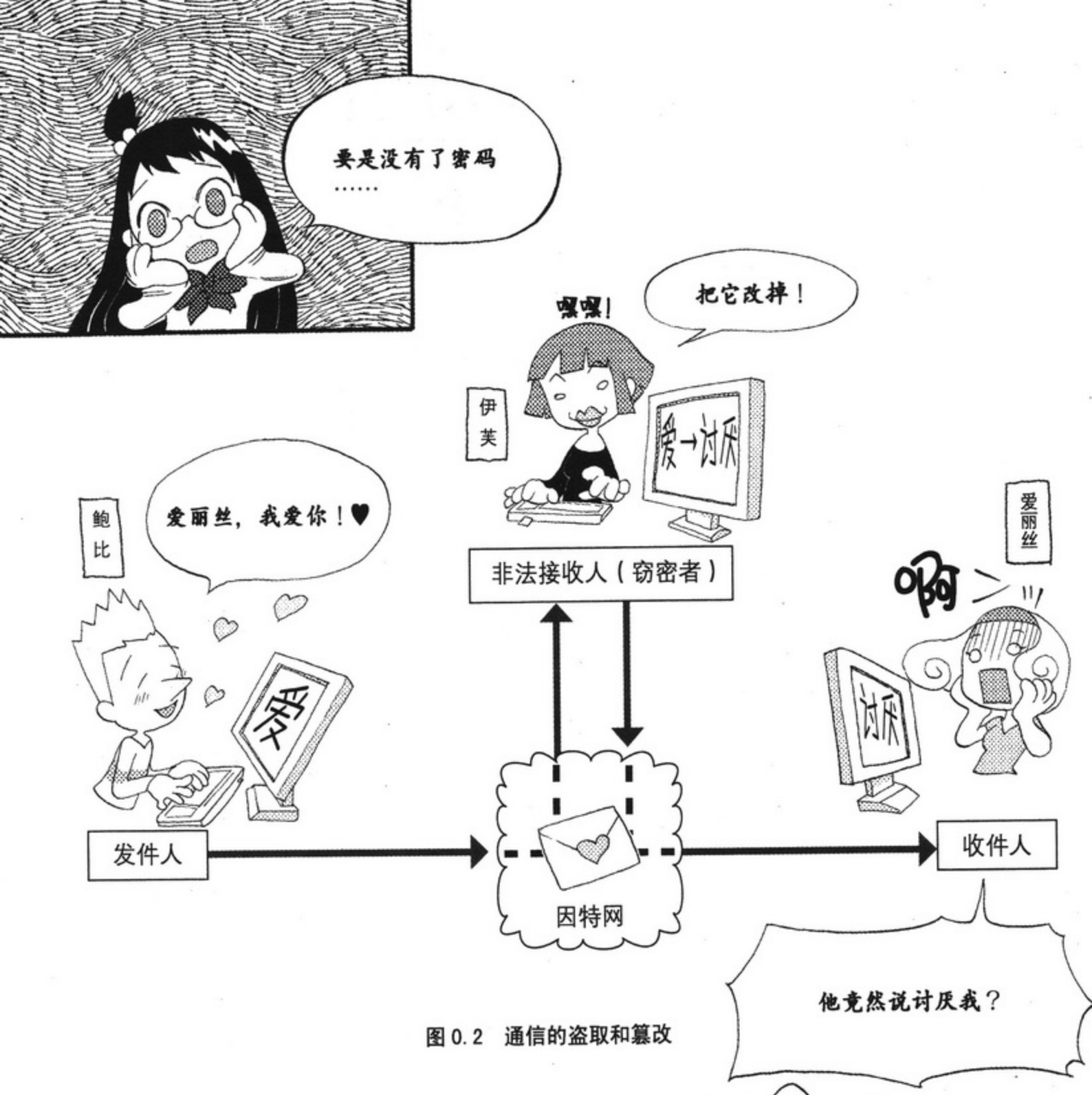


图 0.2　通信的盗取和篡改

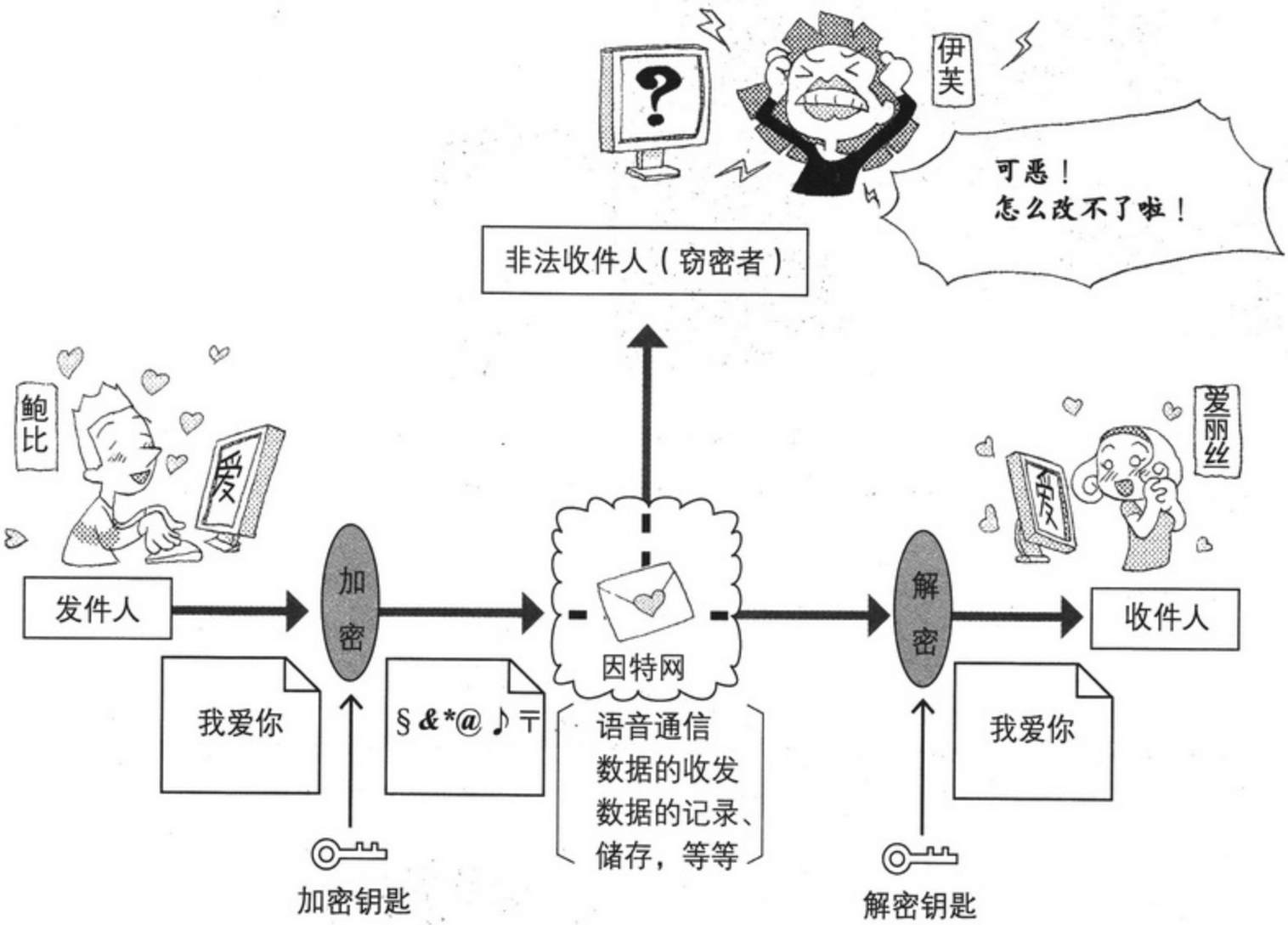


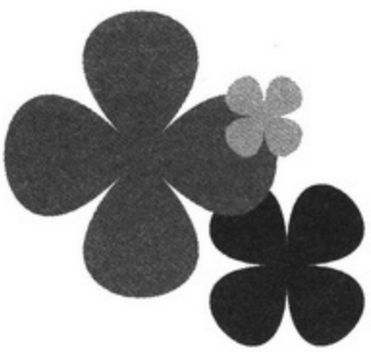
图 0.3　密码的模型

* 密码技术同时也应用于硬盘和存储卡等多种存储设备上。


我来教你！
让我们一起学习密码知识吧！
好，为了能抓住希芙，我一定会努力的！
警……警官……
馆长，这件案子会很快解决的，交给我吧！
啊……
快把手铐给我打开呀！

◆第1章◆

密码学基础



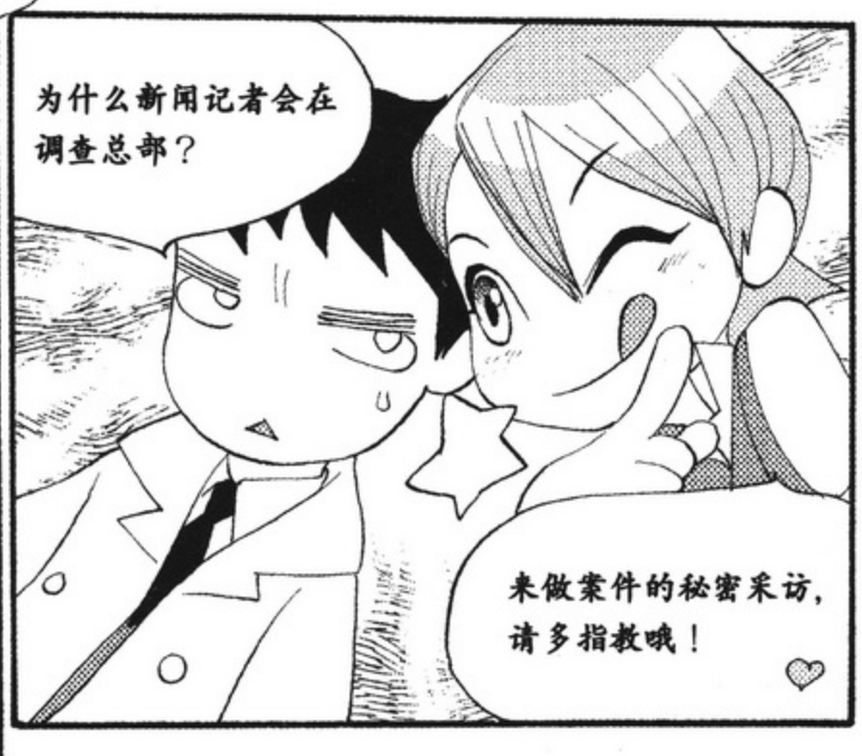
1-1 密码学的相关词汇


交通安

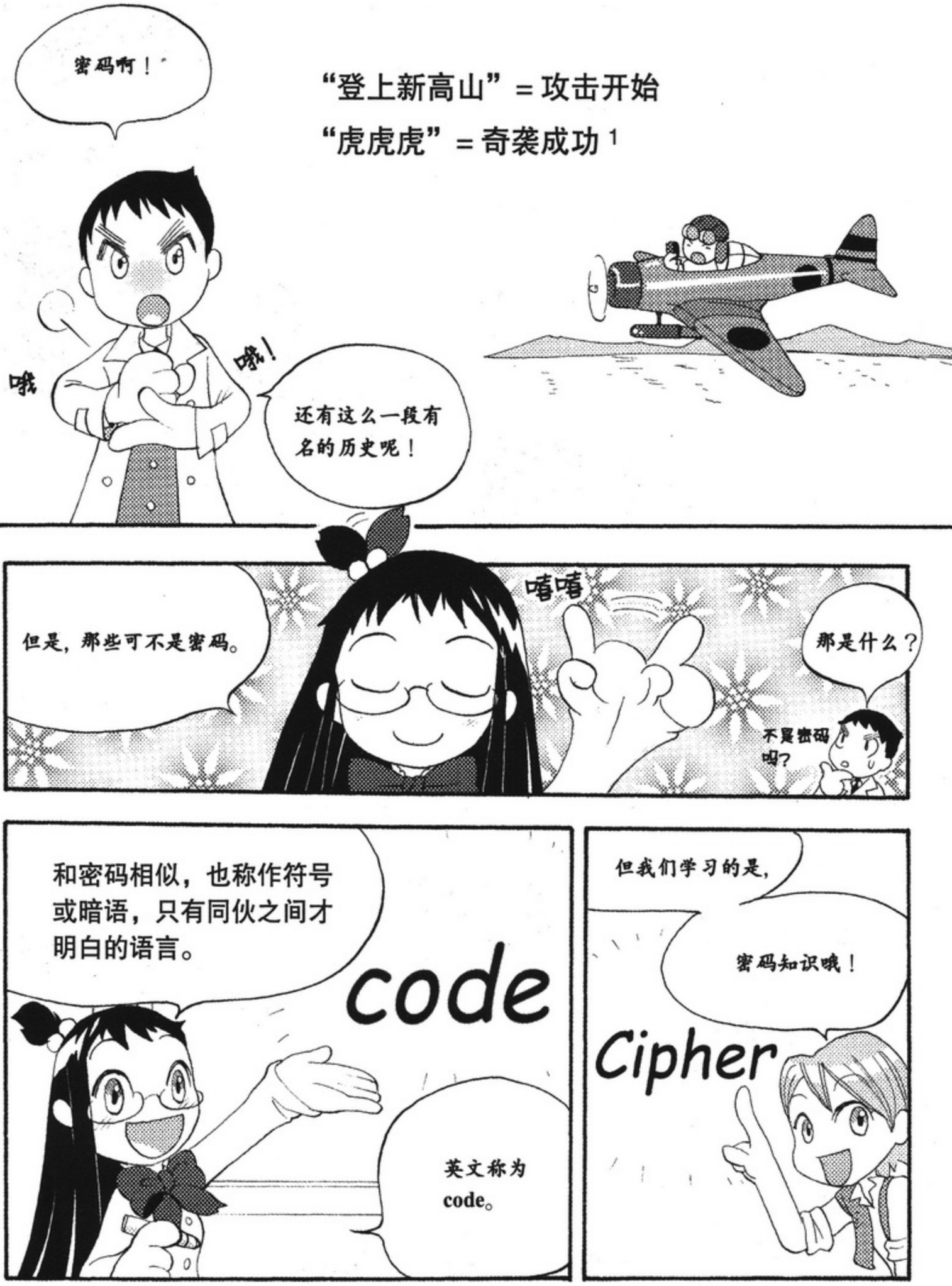

怪盗希芙案件调查总部


哥哥，你要好好学习哦！


耶！
话说回来……


为什么新闻记者会在调查总部？
来做案件的秘密采访，请多指教哦！

1. 日文中“钱包”的发音与“希芙”相似——译者注。



1. 这是太平洋战争中日本海军的电文。

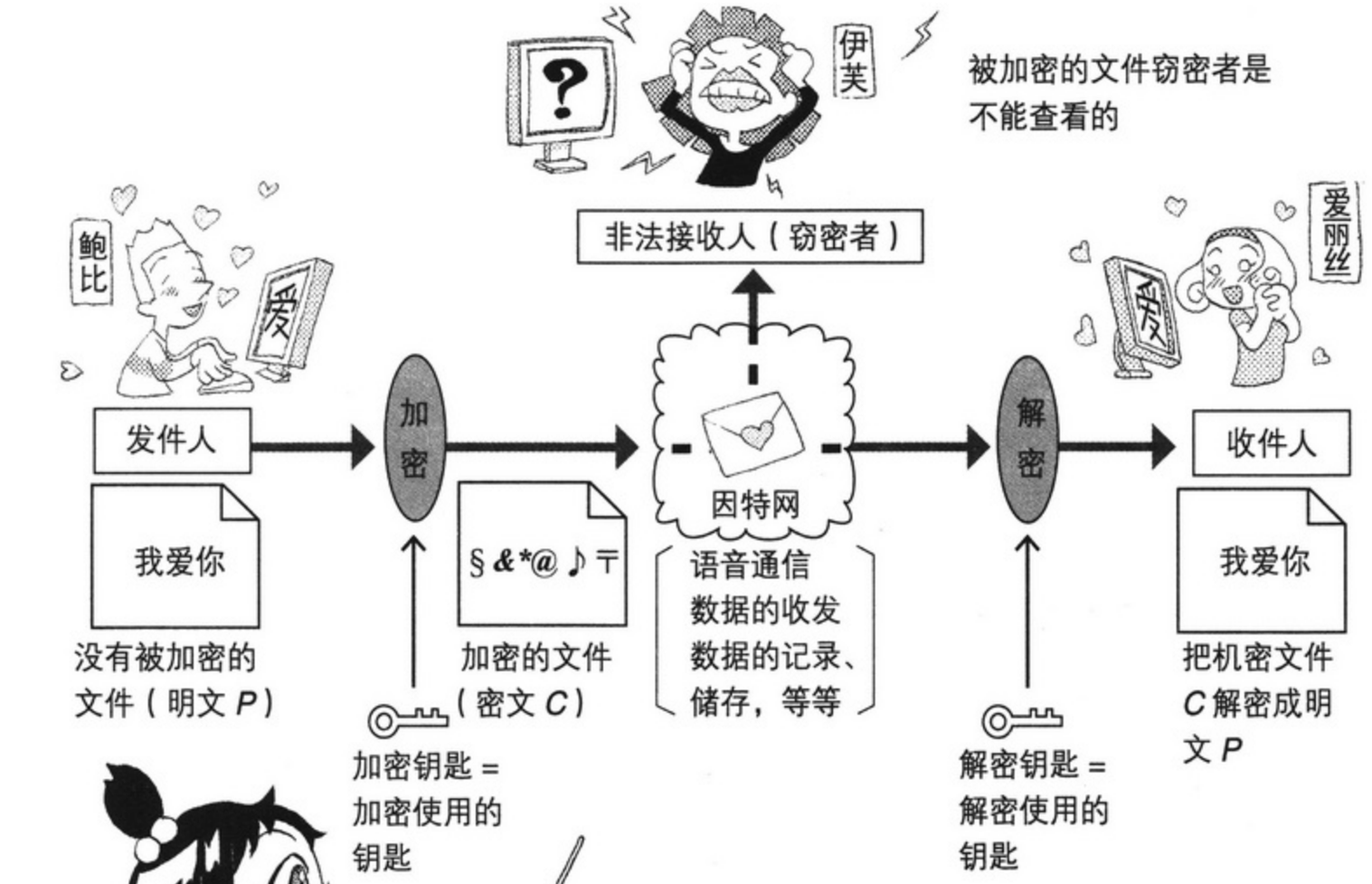
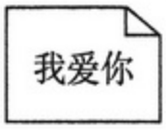

伊芙
被加密的文件窃密者是不能查看的
非法接收人（窃密者）
鲍比
爱
爱丽丝
爱
发件人
加密
因特网
解密
收件人
我爱你
§ &*@ ♪ ∓
我爱你
没有被加密的文件（明文 P）
加密的文件（密文 C）
语音通信
数据的收发
数据的记录、储存，等等
把机密文件 C 解密成明文 P
加密钥匙 = 加密使用的钥匙
解密钥匙 = 解密使用的钥匙

图 1.1　密码的模型

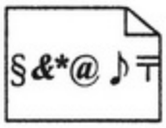

这个图在前面已经出现过了，是香农创造的密码模型哦！
让我们先来学习第 20 页的密码学的基本词汇吧！
克劳德 · 香农（1916—2001）。
20 世纪英国数学家。
1948 年，发表了论文《通信的数学理论》，
被誉为信息理论之父。

✖ 密码学的基本词汇

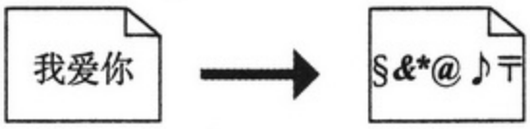
明文 P（Plain text）= 没有经过加密的普通文本。



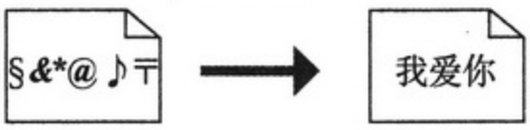
密文 C（Cipher text）= 加密之后的文本。



加密（Encryption/Encipherment）= 将明文转变为密文。



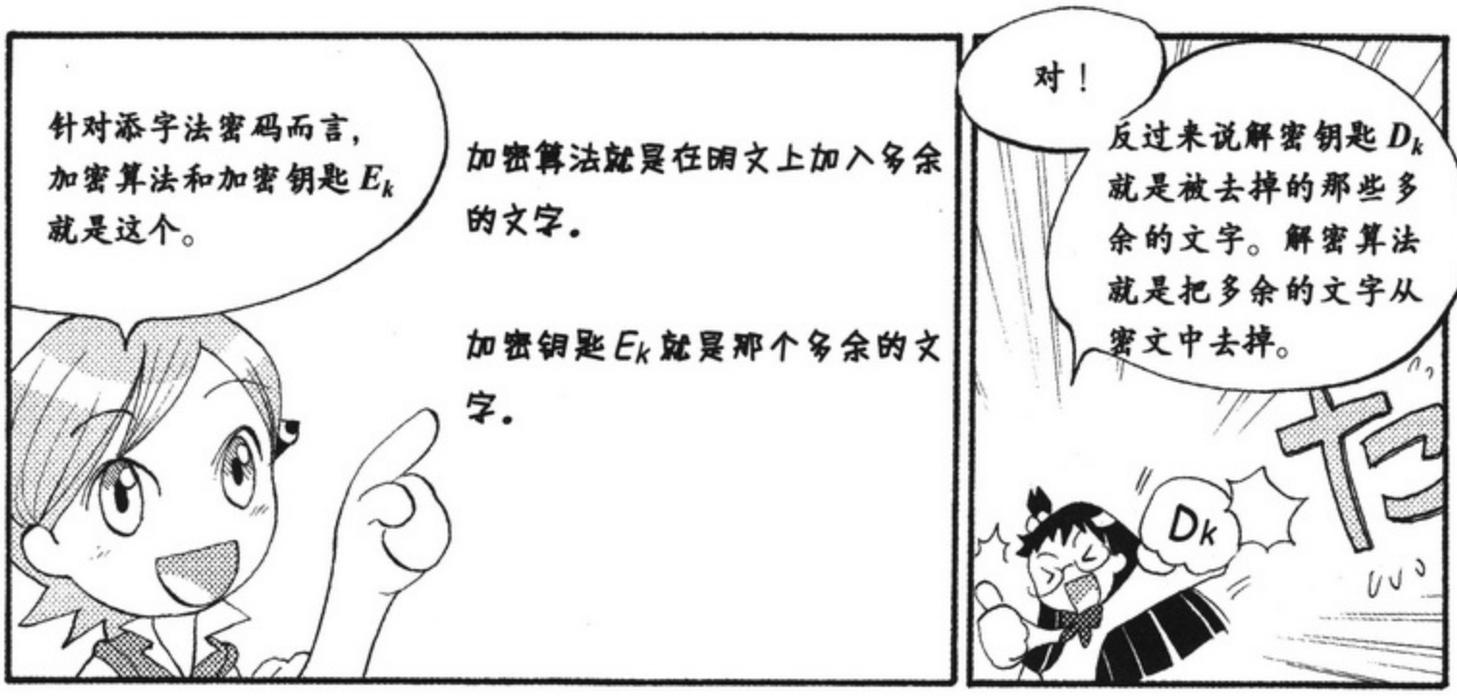
解密（Decryption/Decipherment）= 将密文还原为明文。



加密钥匙 E_k（Encryption Key）= 加密时使用的钥匙。

解密钥匙 D_k（Decryption Key）= 解密时使用的钥匙

1. 算法就是为达成目的和解决问题的一系列清晰指令。



✤ 加密钥匙 E_k 与解密钥匙 D_k 的关系

发送方将明文进行加密。即利用明文 P 和加密钥匙 E_k（加密函数），生成密文 C。

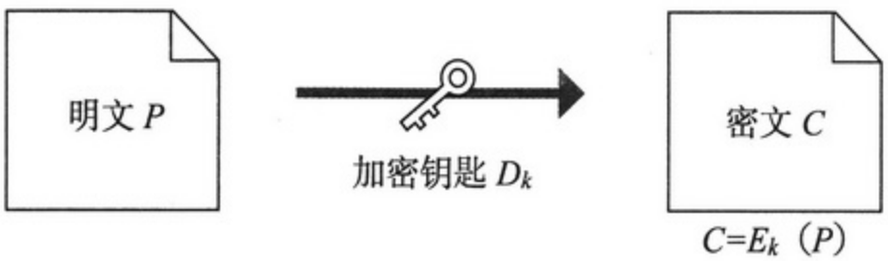


图 1.2　使用加密钥匙 E_k 进行加密

收件方将密文进行解密。此时利用密文 C 和解密钥匙 D_k(解密函数)，将密文解密成为明文 P。

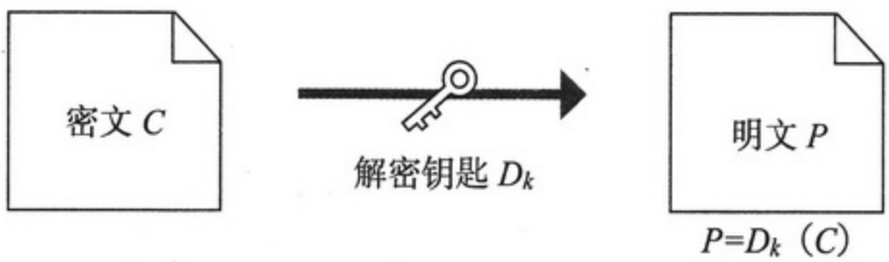


图 1.3　使用解密钥匙 D_k 进行解密


好了，我来出道题！
mjvyjbohifonfjmj
嗜嗜


加密钥匙如果是按照英文字母表的顺序每个字母向后移动一个字符位置的话，
A B C D E
F G H I J
K M N
O
S
明文会是什么呢？


mjvyjbohifonfjmj
l i uxianghenmei l i
是“留香很美丽”吧？
呵呵


哇哈哈哈哈哈
哈哈哈……
“留香很美丽”这话可是大错特错啦……


咚


完全正确！
解密钥匙就是按照英文字母表的顺序每个字母向前移动一个字符位置。
留香老师好可怕……
哇……


但是这样的密码不是很容易就被破解了吗？
信息安全手册
OHM
这本书很重呢！

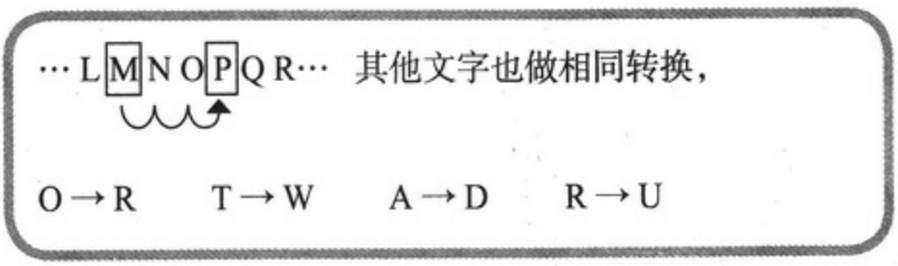

密码是在与窃密者斗智斗勇的过程中逐渐发展起来的。
下一页我来介绍一下几种古典密码吧！

1-2 古典密码技术

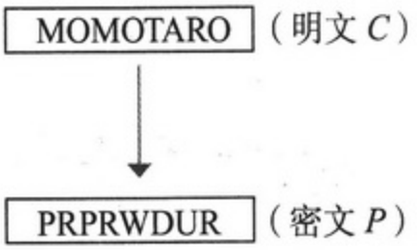
✤ 凯撒密码

将明文中的各个字符，按顺序进行 n 个字符错位转换的加密方法称为凯撒密码。试着将“MOMOTARO”进行加密看看。

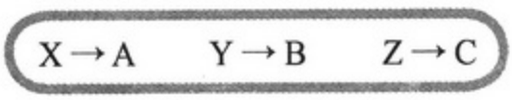
例如：假设 n =3，向后错位 3 个字母。



这样可以生成如下的密文。



此外，字母序列的最后 3 个字母，则与前面的字母进行循环。



凯撒（Gaius Julius Caesar，公元前 100 年—公元前 44 年），古罗马时期的军事家和政治家。高卢战争的时候，他发明了这个密码。这样可以在敌人无法知晓通信内容的情况下，与联军进行联络。

1. 日语中“像掷骰子的人一样”形容人独断专行，而“骰子”的发音与“犀牛”相同。理绪想说的是掷“骰子”，而顺想到的是“犀牛”——译者注。

✤ 换字式密码

将凯撒密码进行复杂变换，即将每个字符及后移字符数都进行变化的方式称为**换字式密码**。

其中，将明文的每个字符通过不同字符与密文一一对应的方式称为“单一换字密码”。凯撒密码也是单一换字密码的一种。

例如：将英文 26 个字母，按如下规则进行变换。

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

↓

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

= 变换规则 $\overset{\text{sigema}}{\sigma}$

从而生成如下的密码。

M O M O T A R O （明文 C）

根据变换规则 σ 进行变换

↓

D G D G Z Q K G （密文 P）

在上述密码中，把换字作为算法，“每个文字的转换方式”即变换规则 σ 被认为是加密钥匙 E_k。

✤ 多表替代密码

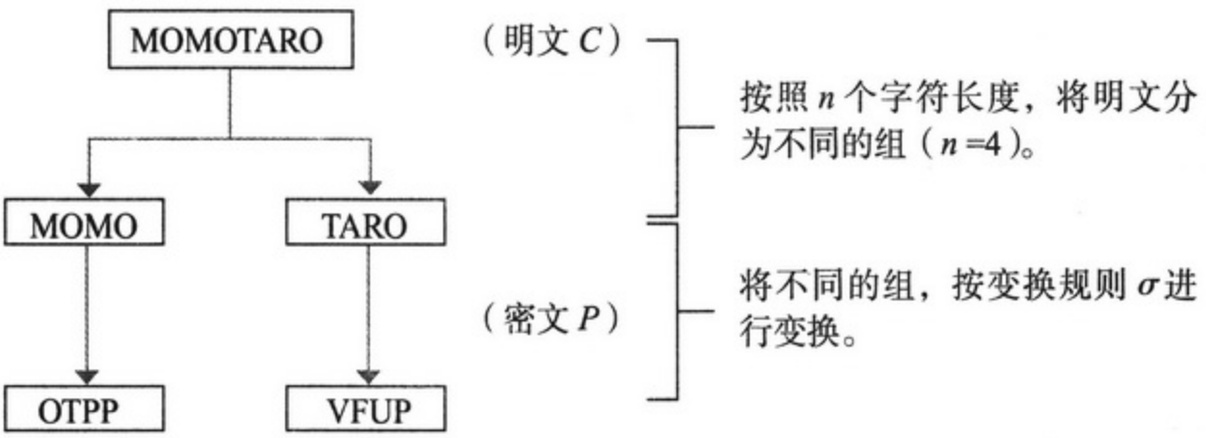
将明文内容按照 n 个字符长度分成不同的分组，并在各组分别使用不同的变换规则，这种方式称为多表替代密码。也可以称为凯撒密码的扩展版本。

例如：假设 n =4，变换规则为 σ，并以下面的方式进行变化。

第一组文字 → 移动 2 个字母
第二组文字 → 移动 5 个字母
第三组文字 → 移动 3 个字母
第四组文字 → 移动 1 个字母

= 变换规则 σ

这样的话，可以生成如下的密文。



在这种密码中，分组内的字符数和字母的变换规则，称为加密钥匙。

✤ 转制式密码

将明文内容按照 n 个字符长度进行分组，并将各组内文字的顺序进行替换的密码，称为转制式密码。

例如：假设 $n=4$，并将 τ 作为替换规则，规定如下：

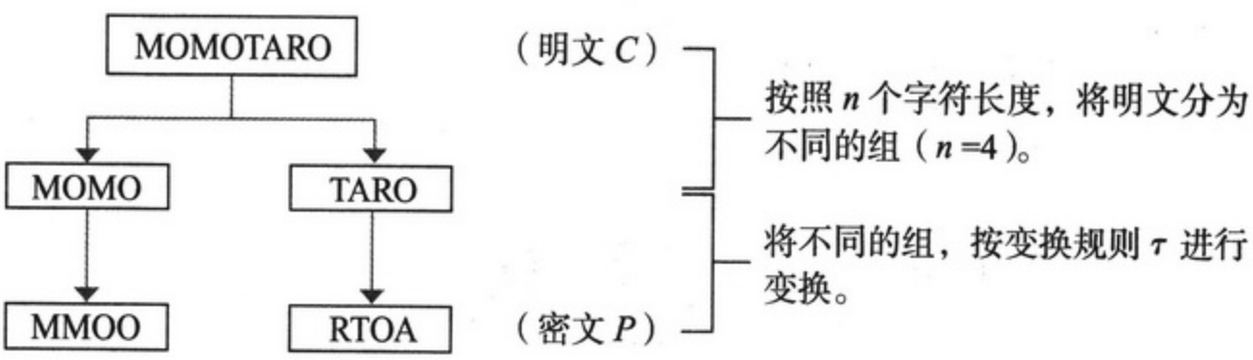
$$\tau = \begin{pmatrix} 1234 \\ 2413 \end{pmatrix}$$

上面的算式，表示为如下的替换规则。

第一个文字 → 转变为第二个
第二个文字 → 转变为第四个
第三个文字 → 转变为第一个
第四个文字 → 转变为第三个

= 替换规则 τ

这样的话，可以生成如下的密文。

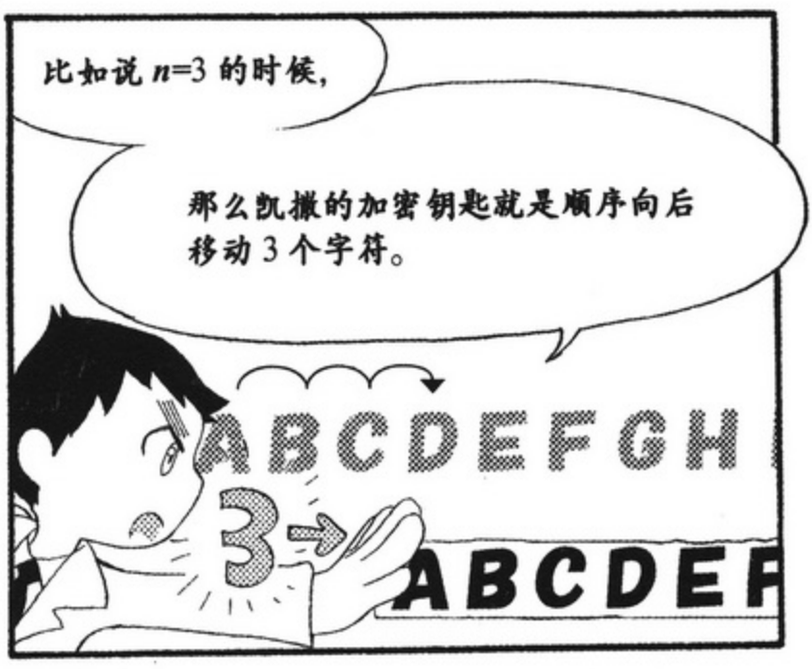


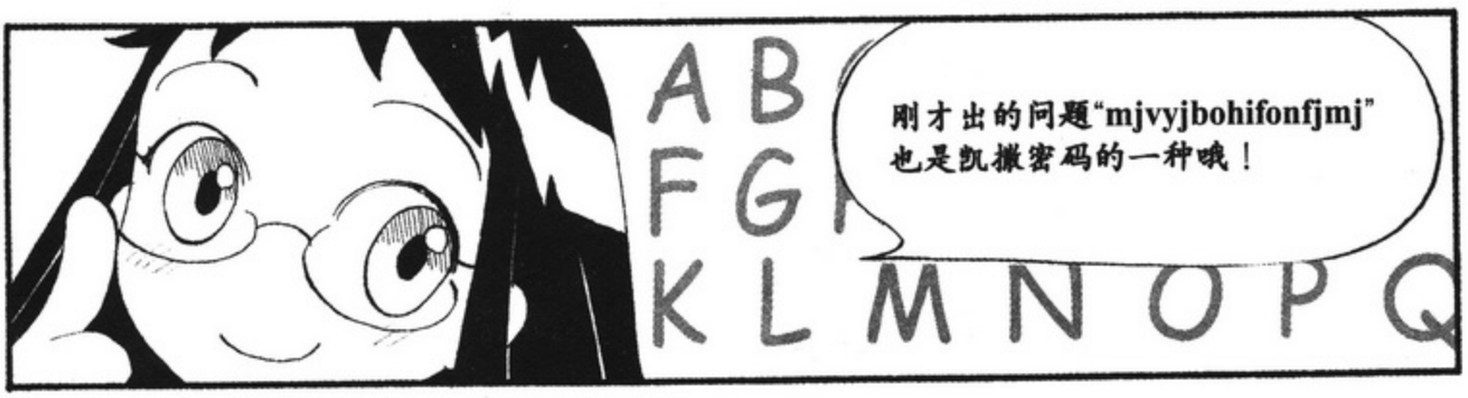
在这种密码中，将文字进行变换的规则，称为加密算法，分组内的字符数和变换规则，就是加密钥匙。

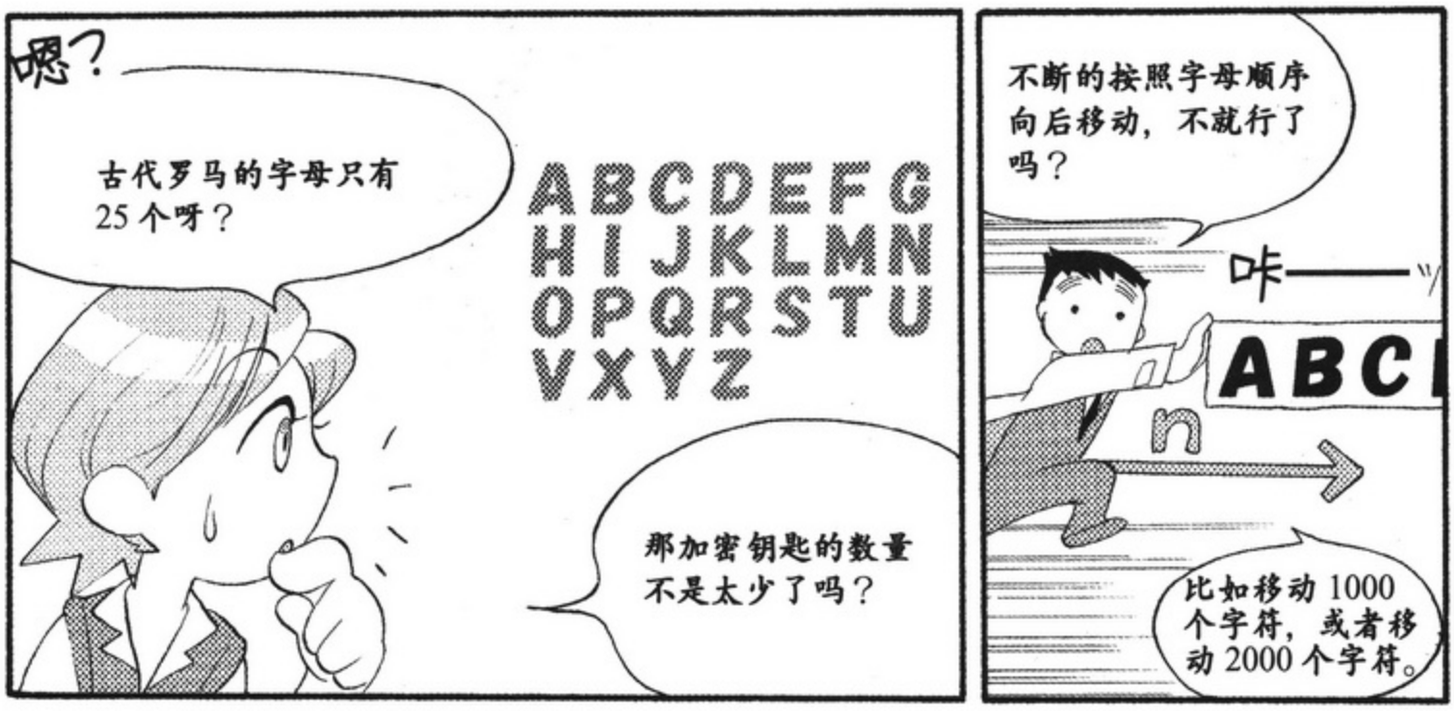
1-3 密码的安全强度


凯撒密码虽然出现于2000多年以前，
但由于其中使用了算法和钥匙的概念，
所以与现代密码理论也是有紧密联系的。


凯撒密码的加密算法，
就是把明文的字母按字母序列顺序移动 n 个字符吧！
n →


比如说 n=3 的时候，
那么凯撒的加密钥匙就是顺序向后移动3个字符。
3 →


刚才出的问题“mjvyjbohifonfjmj”也是凯撒密码的一种哦！


嗯？
古代罗马的字母只有25个呀？
A B C D E F G
H I J K L M N
O P Q R S T U
V X Y Z
那加密钥匙的数量不是太少了吗？
不断的按照字母顺序向后移动，不就行了吗？
咔——
A B C
n
比如移动1000个字符，或者移动2000个字符。


就和手表的表盘一样，只能按照固定好的数字来循环哦！
呀——
A B C D E F G H I J K L M N O P Q R S T U V X Y Z
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
咕噜咕噜
嗯，就算移动1000个字符，或者2000个，都可以喽！


也就是说，不管移动多少个字符，加密钥匙充其量也只有24个而已嘛！


所以说，古代的窃密者只要注意凯撒密码的规律，
最多试验24次就能够找到钥匙，密码就被破解了。
8 9 10 11 12
25


不用字母序列，用日语的五十音图不就可以了吗？
哈哈
真是好主意
平假名，片假名，再加上汉字，可以组成1万多个钥匙呢！


为了不被窃密者攻击，
钥匙的数量还是越多越好啊！


在别的密码类型中，钥匙的数量到底有多少呢？一起来看看吧！

✤ 换字式密码的钥匙数量

英语中有 26 个字母。现在采用英文字母来对密码进行说明。因此，钥匙的总数是，对不同的 26 个字符进行不同顺序的排列所得出的总数，可以按如下方式进行计算。

$$P_{26}^{26} = 26! = 26 \times 25 \times 24 \times \cdots \times 3 \times 2 \times 1 \approx 4.03291461 \times 10^{26}$$

这里所讲的排列是指将字母按照不同的方式进行排列，可用 Permutation 的 P 来表示。这是一个非常庞大的数值。用计算机以每秒 1 亿次的速度来计算，以逐个排查的方法来寻找钥匙的话，最长可能需要 1280 亿年这样一个惊人的计算时间（计算量）。

从理论上讲，可以通过寻找钥匙来破解密码。但以实际的计算量来看，换字式密码被公认为安全密码，尽管人们已经知道利用频率分析法（针对明文中出现字符的频率与密文中出现字符的频率相同的弱点）的方式来对其进行破解。同时，在换字式密码中，还有一种从计算量来看比较安全的，只限使用一次的钥匙，称为一次性密码钥匙。

在这里先复习一下相关的数学知识。大家听说过排列和组合的算法吗？排列是指从 n 个字符中取出 r 个，并将其依次排成一列的方法总和，计算公式如下所示。

$$P_n^r = n \times (n-1) \times (n-2) \times \cdots \times (n-r+1) = \frac{n!}{(n-r)!}$$

组合是指从 n 个字符中取 r 个的方法总和，用 Combination 的 C 来表示。

$$C_n^r = \frac{P_n^r}{r!} = \frac{n!}{(n-r)!\ r!}$$

在排列和组合中，在排列里因为顺序非常重要，所以 AB 和 BA 被认为是不同的，但组合仅仅是指取出的方法，而与顺序无关。因此，AB 和 BA 被认为是相同。此外，感叹号“！”则表示阶乘的意思。阶乘是以 n 为范围，从 1 到 n 之间的所有整数的乘积。

$$n! = n \times (n-1) \times \cdots \times 3 \times 2 \times 1$$

✤ 多表替代密码的钥匙数量

假设 1 组中有 n 个字符。其中第 1 个字符因为不知道其转换规则，因此需要测试 26 次。与此相同，第 2 个字符一直到第 n 个字符，各自需要测试 26 次。因此，钥匙总数为如下所示：

$$\underbrace{26 \times 26 \times \cdots \times 26 \times 26}_{n\text{个}} = 26^n$$

假设 n =4，结果如下：

$$\underbrace{26 \times 26 \times 26 \times 26}_{4\text{个}} = 26^4$$

$$26^4 = 456\,976$$

随着 n 的数值不断加大，钥匙数将急剧增加。当 n =10 时，钥匙数将会超过 140 兆。

✤ 转置式密码的钥匙数量

设定 1 组为 n 个字符长度，钥匙的总数为如下所示：

$$P_n^n = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1 = n!$$

假设 1 组中有 4 个字符（n =4）钥匙 E_k 的总数为如下所示：

$$4! = 4 \times 3 \times 2 \times 1 = 24$$

n 的数值越大，钥匙总数也越多，从而提高了密码的安全强度。特别是当 n =26 时，钥匙总数达到了最大。

如果说钥匙的数量越多，密码就越安全的话，虽然换字式密码的安全强度很高，

嗯

换字式

多表式

但多表式的加密方法看起来更复杂啊！

对于很长的单一换字式密码，还是有破解线索的。

黄金虫……
是昆虫图鉴吧。

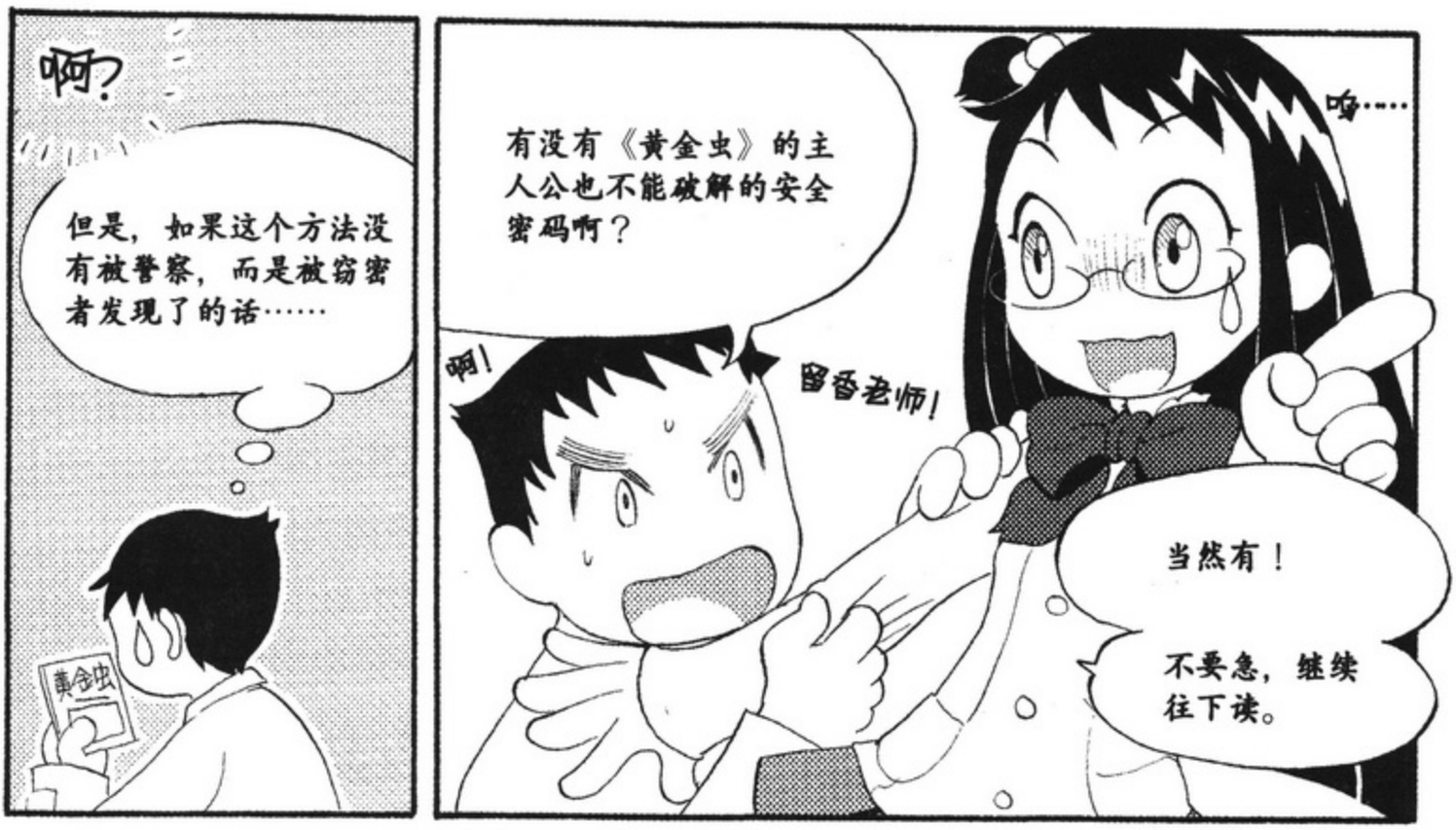
我是富翁。

是叫《黄金虫》的很有名的短篇推理小说，专门讲述如何破解密码的。

哦，
是小说啊！

《黄金虫》中的密码节选：
53‡‡†305))6*;4826)
4‡.)4‡);806*;48†
8¶60))85;1‡(;:‡*8


统计一下英文中经常用到的单词和字母就会明白。
《黄金虫》就是把那个当成启示来破解密码的！
e
the
英文中最常用的字母是“e”，单词是“the”，
那么，“8”是“e”，“；48”就一定是“the”！
没错！
太聪明啦！
我都变成神探啦！
密文越长线索就越多，就会更容易破解。
较短的密文就很难破解了！



✤ 解密需要的条件

一般情况下，可以认为解密（窃密）需要以下条件：

1. 知道了加密的算法。
2. 某些字符出现频率过高或过低，针对明文具有统计性质的数据。
3. 持有大量加密算法的例文。

✤ 绝对安全的密码

基于仅使用一次的随机数，采用一次性密码钥匙，可以生成不可破解的密码。该密文不可重复显示。

具体来说，就是在明文 P 上附加同等字符长度的随机数列，而生成的密文 C。这个称为费纳姆密码（费纳姆于 1917 年设计出来，并取得了专利），利用单次使用的密码（一次性密码钥匙），其不可被破译性，在 1949 年被克劳德 · 香农（参看第 19 页）通过数学方式进行了论证。

下面是费纳姆密码的简单例子。

首先，将每个字母对应一个不同的文字编码（数值）。

表 1.1　文字编码

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

将数值相加之后得出的和，再被 26 整除，得出余数。

① 将字母转换为文字编码。

明文	M	O	M	O	T	A	R	O
	↓	↓	↓	↓	↓	↓	↓	↓
	12	14	12	14	19	0	17	14

② 与仅使用一次的随机数相加。

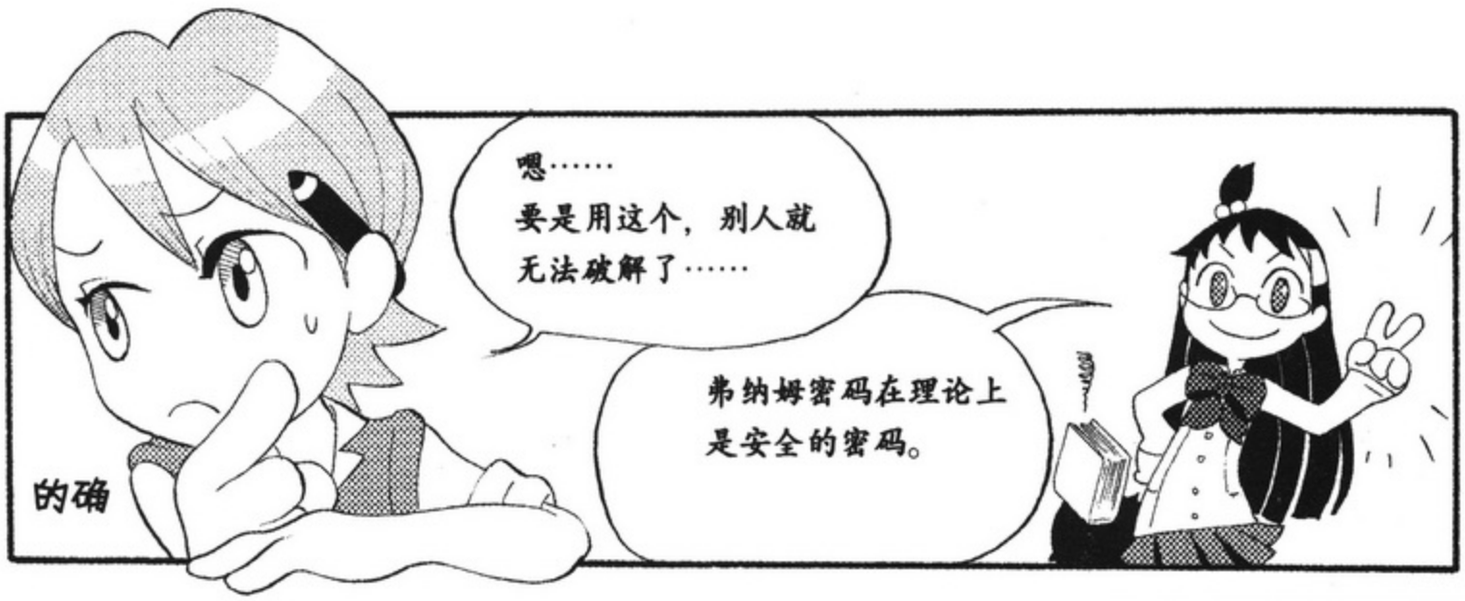
	12	14	12	14	19	0	17	14
	+	+	+	+	+	+	+	+
随机数列	9	20	15	23	27	2	15	8
（加密钥匙）	=	=	=	=	=	=	=	=
	21	34	27	37	46	2	32	22

③ 计算被 26 整除得出的余数。

	21	34	27	37	46	2	32	22
	↓	↓	↓	↓	↓	↓	↓	↓
	21	8	1	11	20	2	6	22

④ 将得出余数文字数码转换为字母。

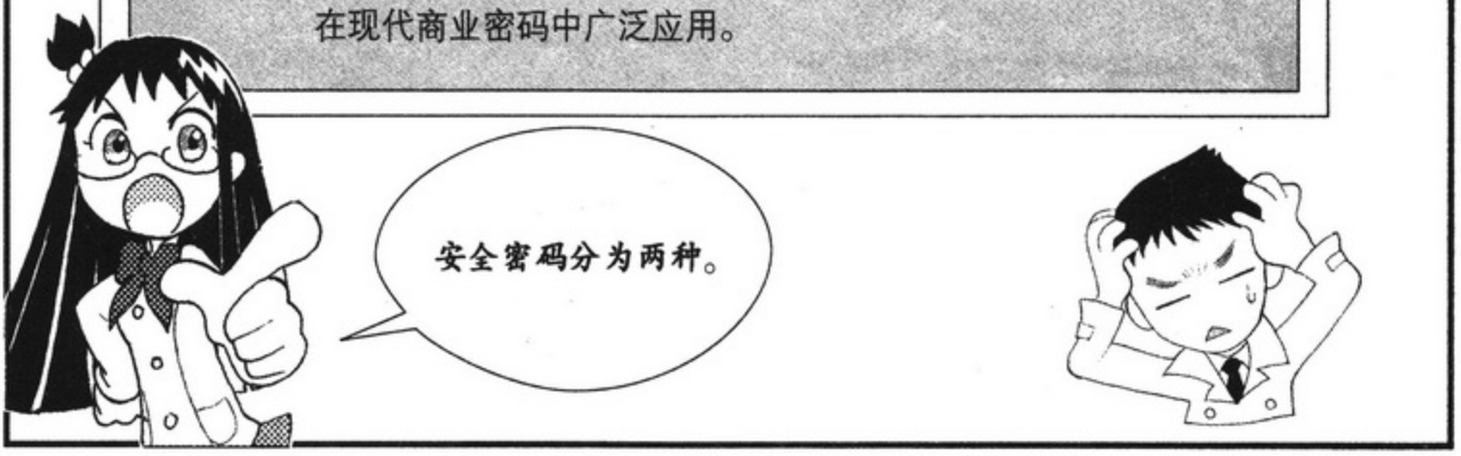
	21	8	1	11	20	2	6	22
	↓	↓	↓	↓	↓	↓	↓	↓
密文	V	I	B	L	U	C	G	W

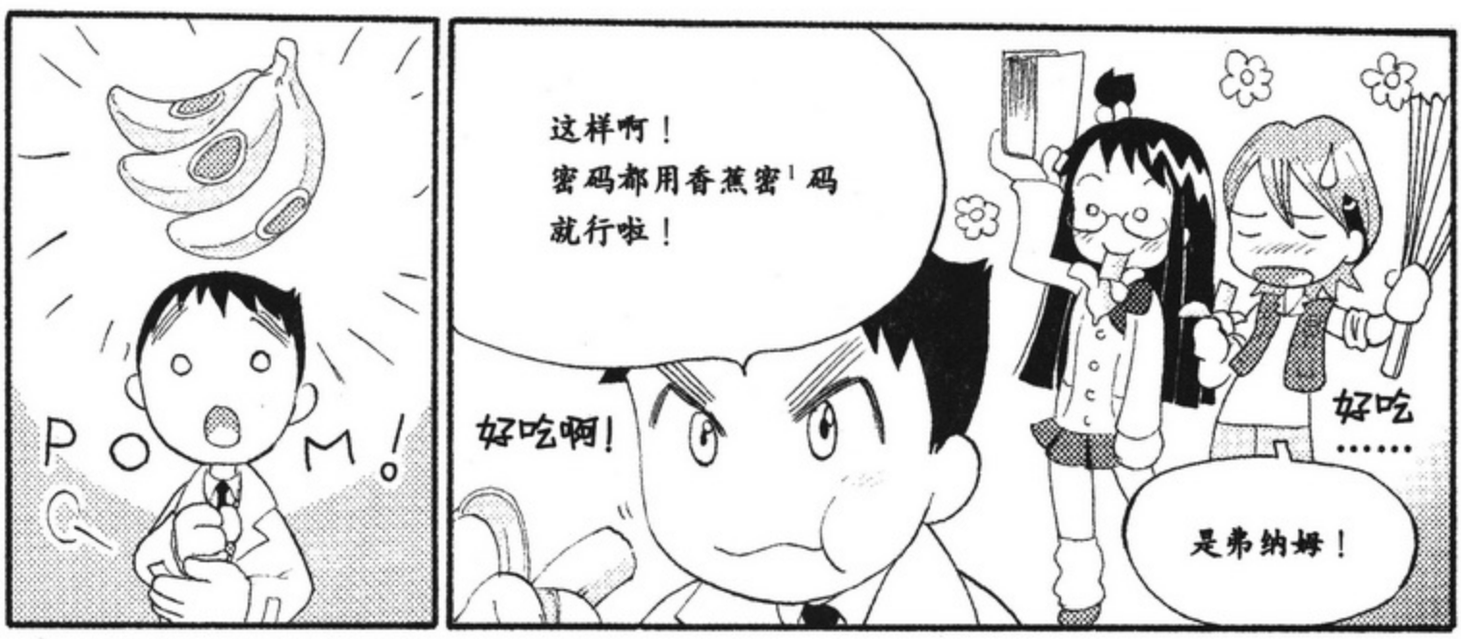


✤ 安全的密码

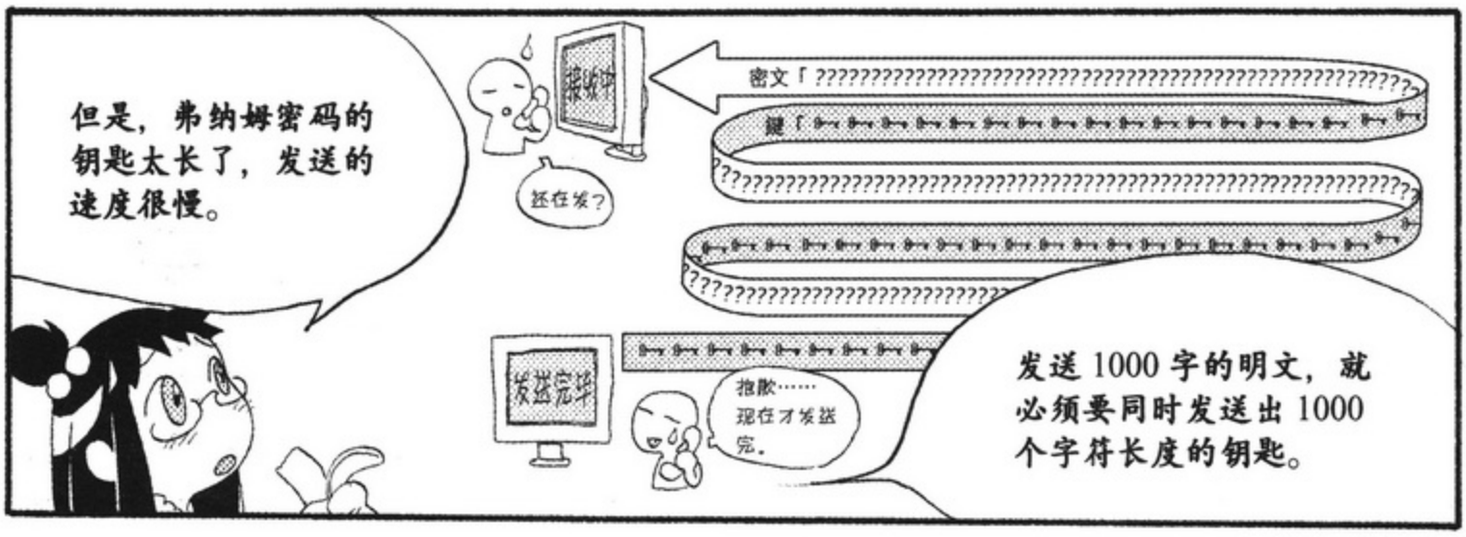
① **绝对安全的密码**
类似于费纳姆密码，理论上不可破解。

② **具有庞大运算量的安全密码**
破解时需要花费过多的成本和时间。
在现代商业密码中广泛应用。



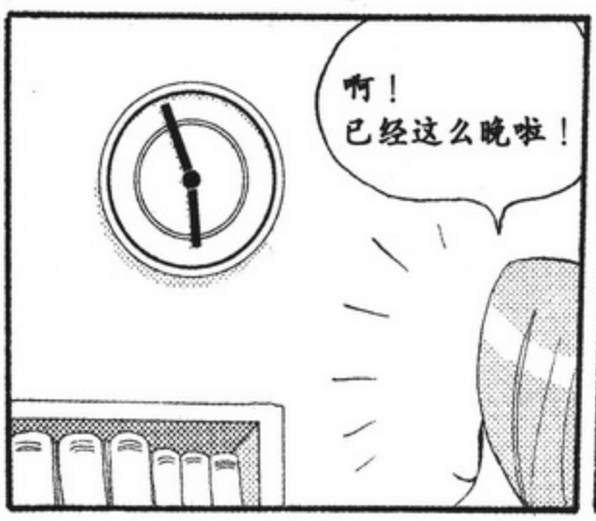


1. “香蕉”的日文发音与“弗纳姆”的日文发音相似——译者注。


但是，弗纳姆密码的钥匙太长了，发送的速度很慢。
接收中
还在发？
密文「 ??
钥匙「
发送完毕
抱歉……现在才发送完。
发送1000字的明文，就必须要同时发送出1000个字符长度的钥匙。


希芙的VDVIRCU密码难道就是弗纳姆密码吗？
因为是故意留下的，所以一般不是那种复杂的密码。
天啊


一定是应该有什么暗示吧……


啊！
已经这么晚啦！


我回报社了。
呼

天完全黑了

承蒙关照，小兔餐厅来送餐了。

啊，来啦！来啦！

让您久等了—

感谢惠顾——

小兔餐厅

拉面，小兔，希芙，

晚安，密码……

好吃，好吃！

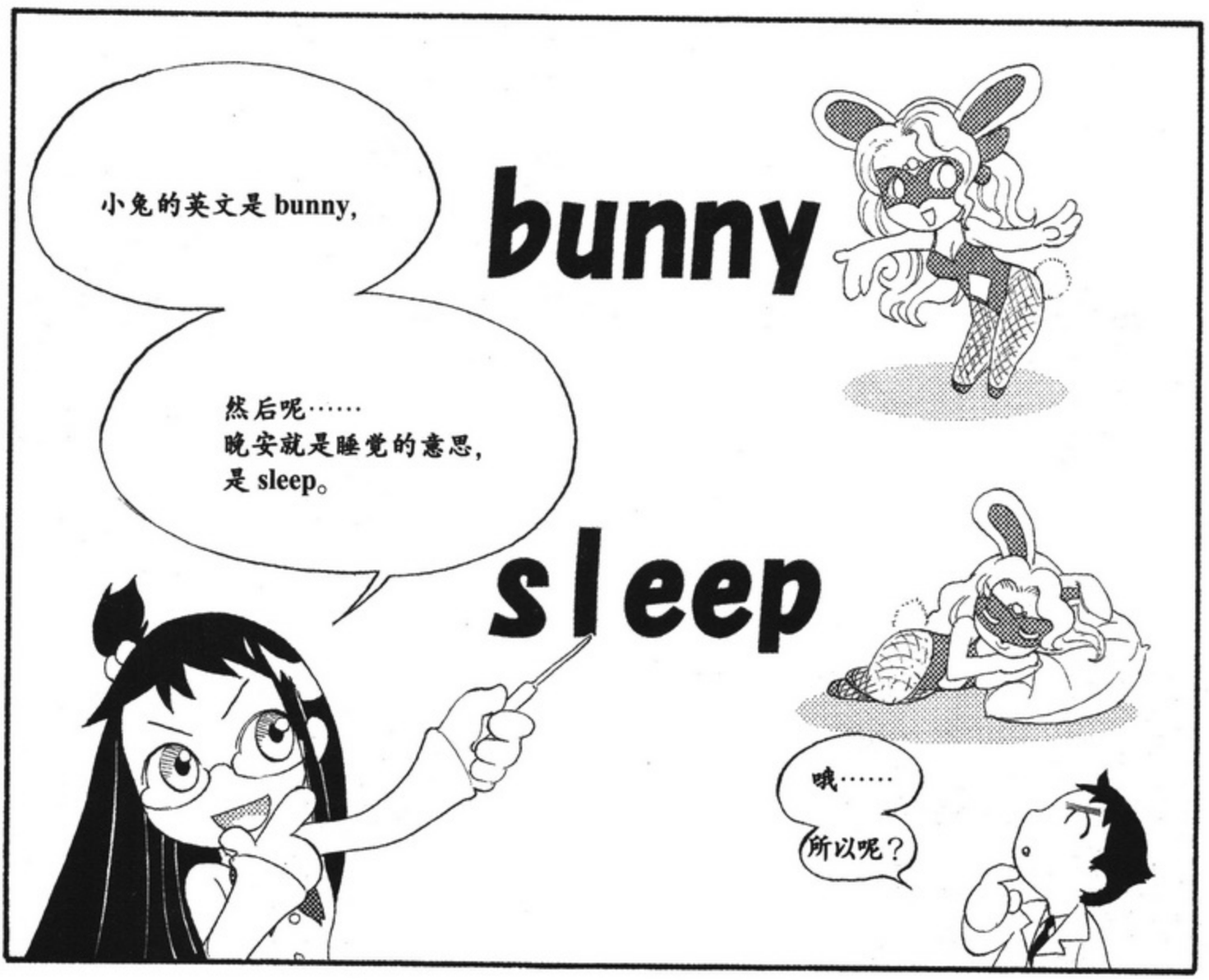
……

嘶

哈哈哈


喂，拉面都凉了！
啊……
我知道啦！
那就快吃吧！
我说的不是拉面！
我解开密码了！
我知道希芙想要偷什么啦！
什，什么？
ぶにょ——
哇——
好恶心啊。


告诉你答案的话，
就给我买电脑吗？
唉……没办法，
只能这样啦。
小兔和晚安就是
线索！
怪盗希芙到此，
画我拿走了，
下次再来取
VDVIRCU。
晚安
那是
什么？


小兔的英文是 bunny，
bunny
然后呢……
晚安就是睡觉的意思，
是 sleep。
sleep
哦……
所以呢？

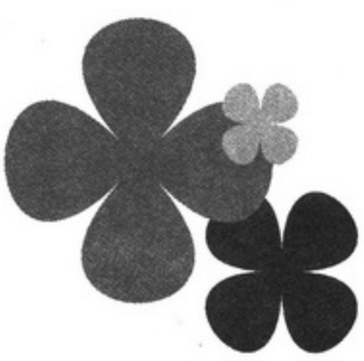

bunny 每个字母向后移动
17 个字，
bunny
↓↓↓↓↓17个字
sleep
就是 sleep ！
是啊！
移动 17 个字就是
钥匙啦！
对！
嘿
嘿
那么把 VDVIRCU 向前
移动 17 个字……
嘿
嘿
嘿
答案就是……
紧张


绿……
绿宝石！
耶！
正……正是……
EMERALD（绿宝石）……
怎么？
国际宝石展的绿宝石被盗啦！
希芙又作案啦，马上去现场！
等
等一下
……


转身
嘿嘿嘿嘿
说出答案的是巡查，电脑的事以后再说吧……
嘿嘿！
好！
快走——
是！

◆第2章◆

通用钥匙加密技术



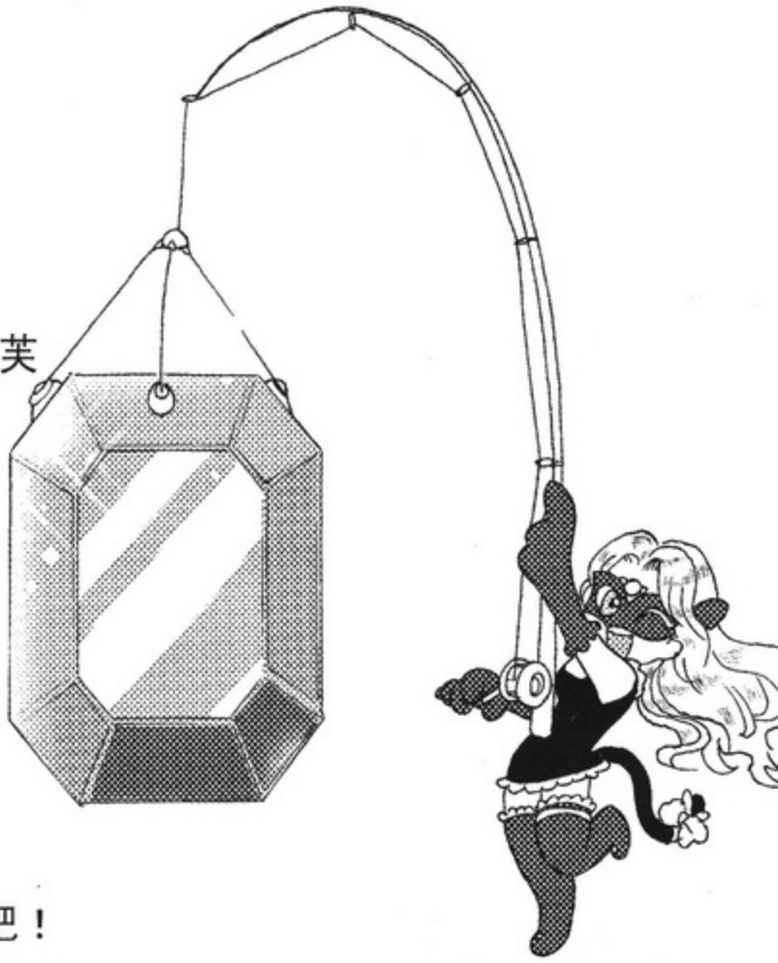
2-1　二进制运算和不可兼析取


价值3亿日元的绿宝石啊！
哇啊啊啊
展示柜是锁着的吗？
不知什么时候被盗走的……
一定是在从棚上巧妙地把这贵重的宝石钓上去的。
嗨！
你怎么会知道盗窃的手法？
果然是你吗……
有人把这个寄到了报社！

今天也大有收获！

绿宝石被我钓走啦！

怪盗希芙



补充

为了下次再见，

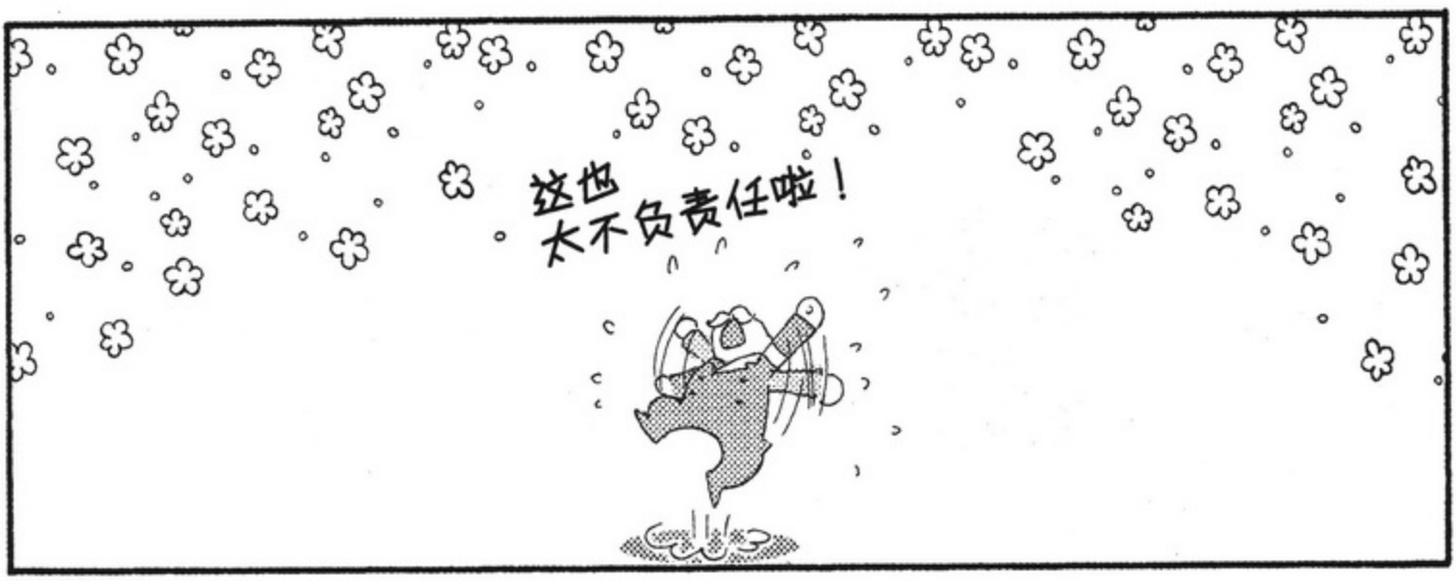
大家先研究一下这个吧！

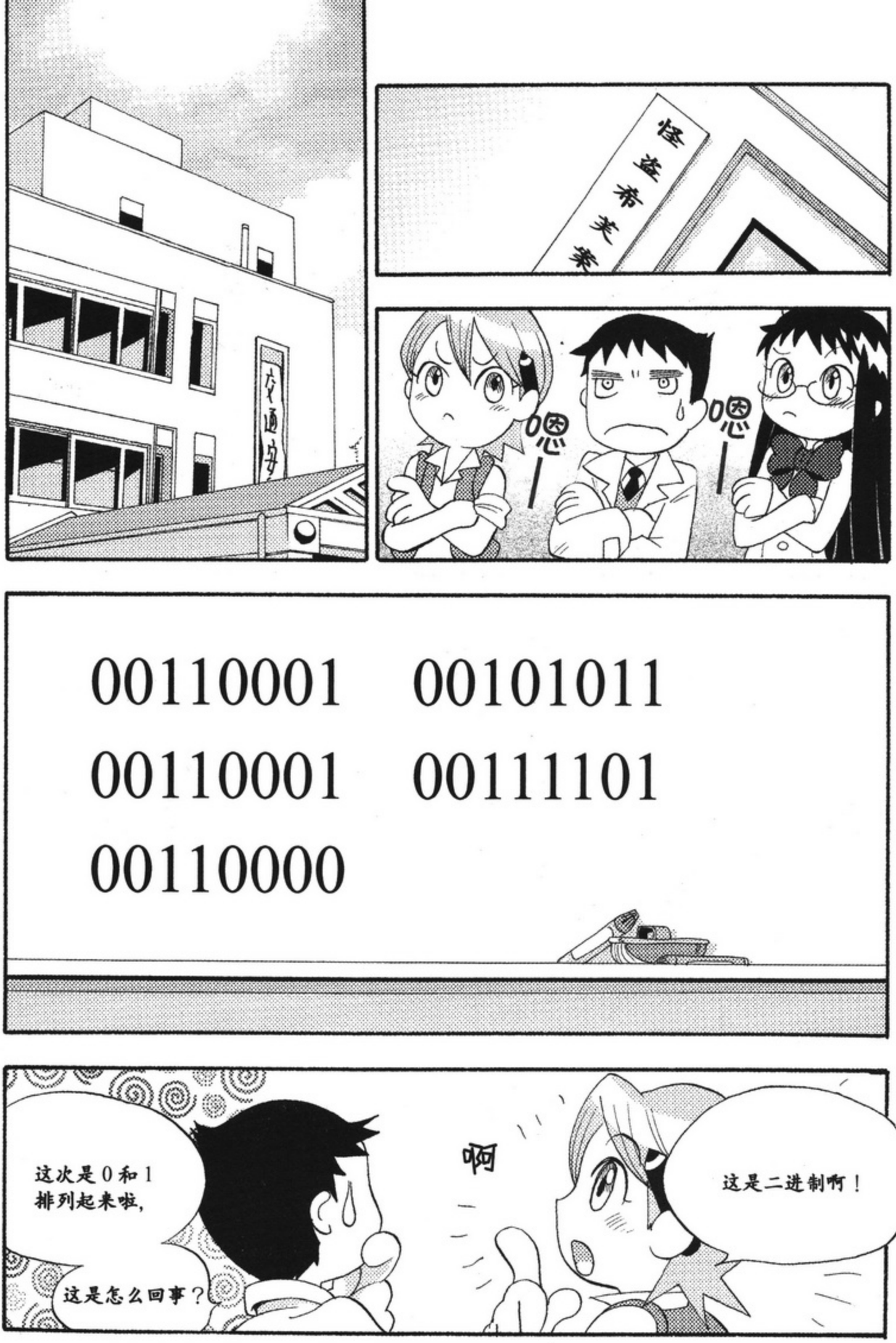
00110001　00101011　00110001　00111101　00110000


我知道罪犯是谁啦！
不妨直说，
我就是罪犯哦。


那就快去抓啊……


等调查出他的真实身份和藏身之处再说吧！
啊
呔——
等等我！


这也太不负责任啦！


交通安
怪盗希美参
嗯——
嗯——
00110001 00101011
00110001 00111101
00110000
这次是0和1
排列起来啦，
这是怎么回事？
啊
这是二进制啊！

用 0 或 1 作为表示数据的最小单位称作比特（bit）。

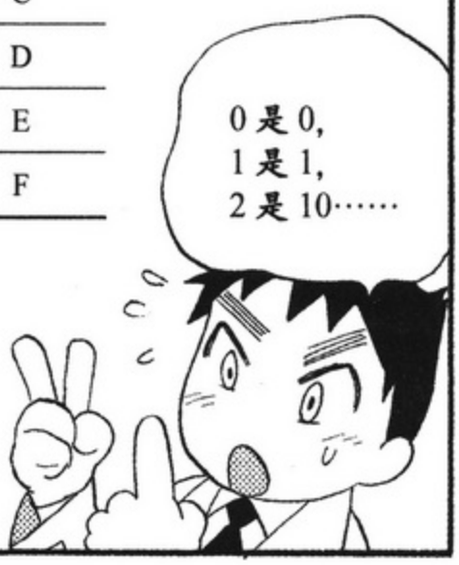
8 比特（由 0 或 1 组成的 8 个数字即 8 位二进制）成为 1 字节（byte）。

1 字节是 2 的 8 次方，也就是说可以用公式 $2^8=256$ 来表示。

表 2.1　二进制与十进制与十六进制的对照

二进制	十进制	十六进制	二进制	十进制	十六进制
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	10	A
0011	3	3	1011	11	B
0100	4	4	1100	12	C
0101	5	5	1101	13	D
0110	6	6	1110	14	E
0111	7	7	1111	15	F

为了对应二进制的数字变大，列数会跟着增加，经常会用十六进制方式来表示。
为了区别十六进制，有时会出现在前面加入“0x”记号的情况。十进制的 10 就用十六进制的“0xA”来表示。



※ JIS X 0201 代码是把世界标准的 ASCII（7 比特）代码变成能在日本国内广泛使用的 1 字节（8 比特），用英文和符号，还有半角片假名等等来表示文字代码。

※表中的十六进制用上边的上 4 位比特和左边的下 4 位比特来表示。

表 2.2　JIS X 0201 代码

下4位比特 \ 上4位比特	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00		DE		0	@	P		p				―	タ	ミ		
01	SH	D1	!	1	A	Q	a	q			。	ア	チ	ム		
02	SX	D2	"	2	B	R	b	r			「	イ	ツ	メ		
03	EX	D3	#	3	C	S	c	s			」	ウ	テ	モ		
04	ET	D4	$	4	D	T	d	t			、	エ	ト	ヤ		
05	EQ	NK	%	5	E	U	e	u			・	オ	ナ	ユ		
06	AK	SN	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
07	BL	EB	'	7	G	W	g	w			ァ	キ	ヌ	ラ		
08	BS	CN	(	8	H	X	h	x			ィ	ク	ネ	リ		
09	HT	EM	)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
0A	LF	SB	*	:	J	Z	j	z			ェ	コ	ハ	レ		
0B	HM	EC	+	;	K	[	k	{			ォ	サ	ヒ	ロ		
0C	CL	→	,	<	L	¥	l	|			ャ	シ	フ	ワ		
0D	CR	←	-	=	M	]	m	}			ュ	ス	ヘ	ン		
0E	SO	↑	.	>	N	^	n	‾			ョ	セ	ホ	"		
0F	SI	↓	/	?	O	_	o				ッ	ソ	マ	°		

那么希芙留下的数字要这样解释啦！

表 2.3　二进制与 JIS X 0201 代码

二进制	十六进制	JIS X 0201 代码
00110001	31	1
00101011	2B	+
00110001	31	1
00111101	3D	=
00110000	30	0

1+1= 0

1 加 1 不是等于 2 吗？

希芙傻了吗？

不是那样的！

这是 XOR 运算！

也就是说这是用不可兼析取来表示的！

对于密码来说是很重要的逻辑运算！

XO 酱？

XO酱

牙疼？

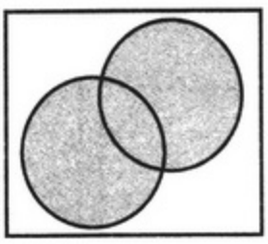
鸟？

突然想起来还有事，我先走了……

好啦！好啦！

OR 逻辑和 A+B

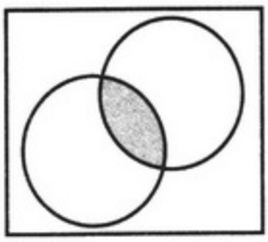
A	B	A+B
0	0	0
1	0	1
0	1	1
1	1	1



哪里有 1 的时候结果就是 1。

AND 逻辑积 A · B

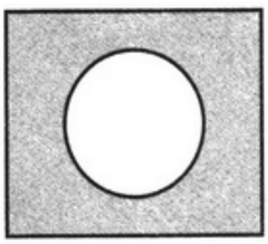
A	B	A+B
0	0	0
1	0	0
0	1	0
1	1	1



两边都是 1 的时候结果就是 1。

NOT 否定 $\overline{A}$

A	$\overline{A}$
1	0
0	1

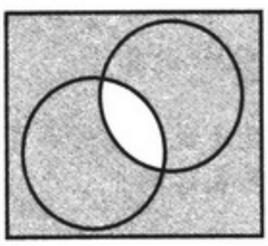


1 变为 0，0 变为 1。

NAND（否定逻辑积）

$\overline{A \cdot B}$

A	B	AB
0	0	1
1	0	1
0	1	1
1	1	0

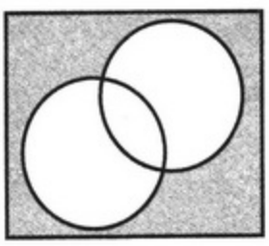


哪里有 0 的时候结果就是 1。

NOR（否定逻辑和）

$\overline{A + B}$

A	B	$\overline{A+B}$
0	0	1
1	0	0
0	1	0
1	1	0

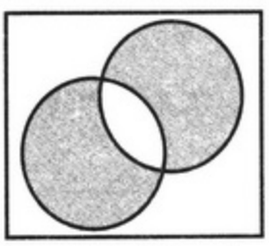


两边都是 0 的时候结果就是 1。

XOR（不可兼析取）

$\overline{A} \cdot B + A \cdot \overline{B} = (A \oplus B)$

A	B	A⊕B
0	0	0
1	0	1
0	1	1
1	1	0



两边不同的时候就是 1（也称作一致、不一致论理）。

图 2.1 逻辑运算

假设（1101）是明文，（1001）是加密钥匙，来进行 XOR 运算。

（1101）⊕（1001）=（0100）
明文　加密钥匙　密文

把运算结果的（0100）作为密文，接下来将密文（0100）和解密钥匙（1001）进行 XOR 运算。

（0100）⊕（1001）=（1101）
密文　解密钥匙　明文

那么，就得到了解密之后的明文。然后，将密文（0100）和明文（1101）进行 XOR 运算，就得到了下面的钥匙。

（0100）⊕（1101）=（1001）
密文　明文　解密钥匙 = 加密钥匙

也就是说，通过明文、加密钥匙、解密钥匙，密文之中的任意 2 个数据，就能得出剩下的那个唯一的数据。

那，那是为什么……

原来如此！
使用 XOR 运算就能进行加密和解密的处理啊！

加密

明文 ⊕ 加密钥匙 = 密文

解密

密文 ⊕ 解密钥匙 = 明文

（加密钥匙 = 解密钥匙）

正是如此！

先用第 1 章出现过的代换和置换，然后再使用 XOR 运算的话……

通用钥匙密码 = 换字处理 + 转置处理 +XOR 运算

现代密码的“通用钥匙密码”就完成了。

通用钥匙？

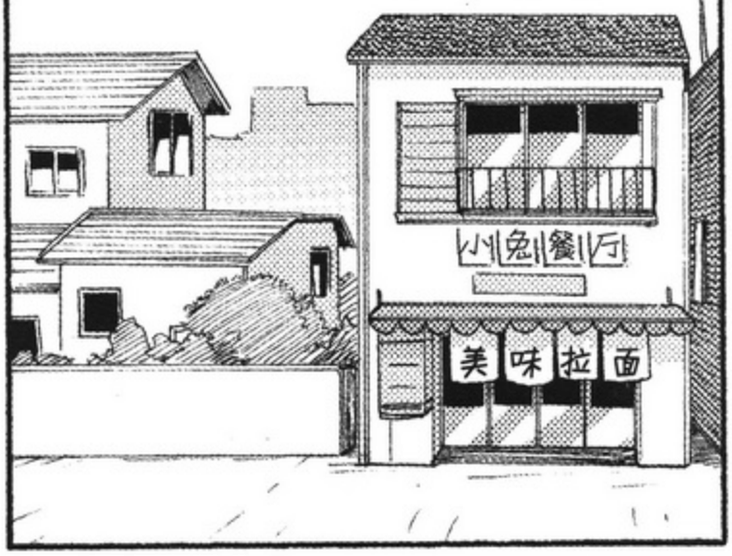
就是加密和解密用同一个钥匙的密码哦！

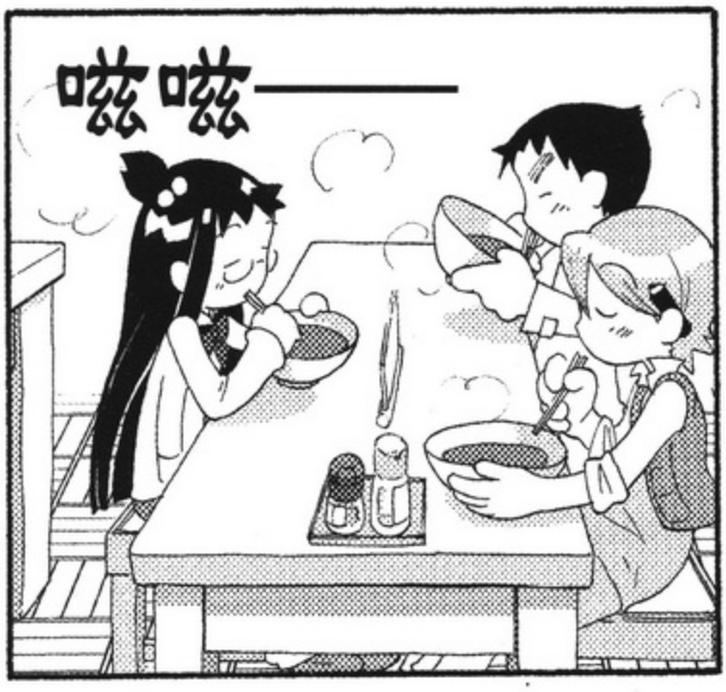
没事吧？

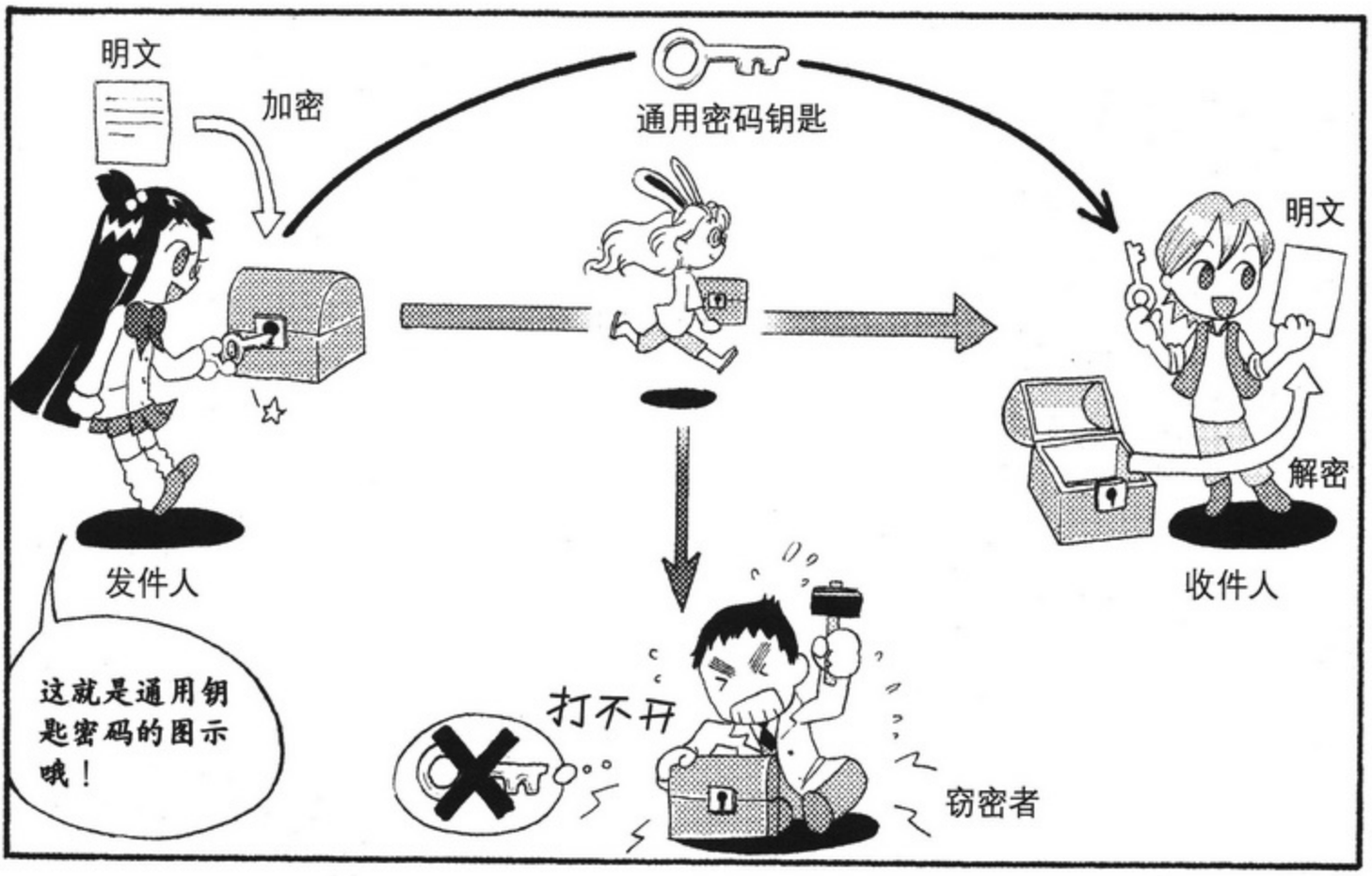
下面就来学习一下关于通用钥匙密码的知识吧！

不知道还能不能学下去……

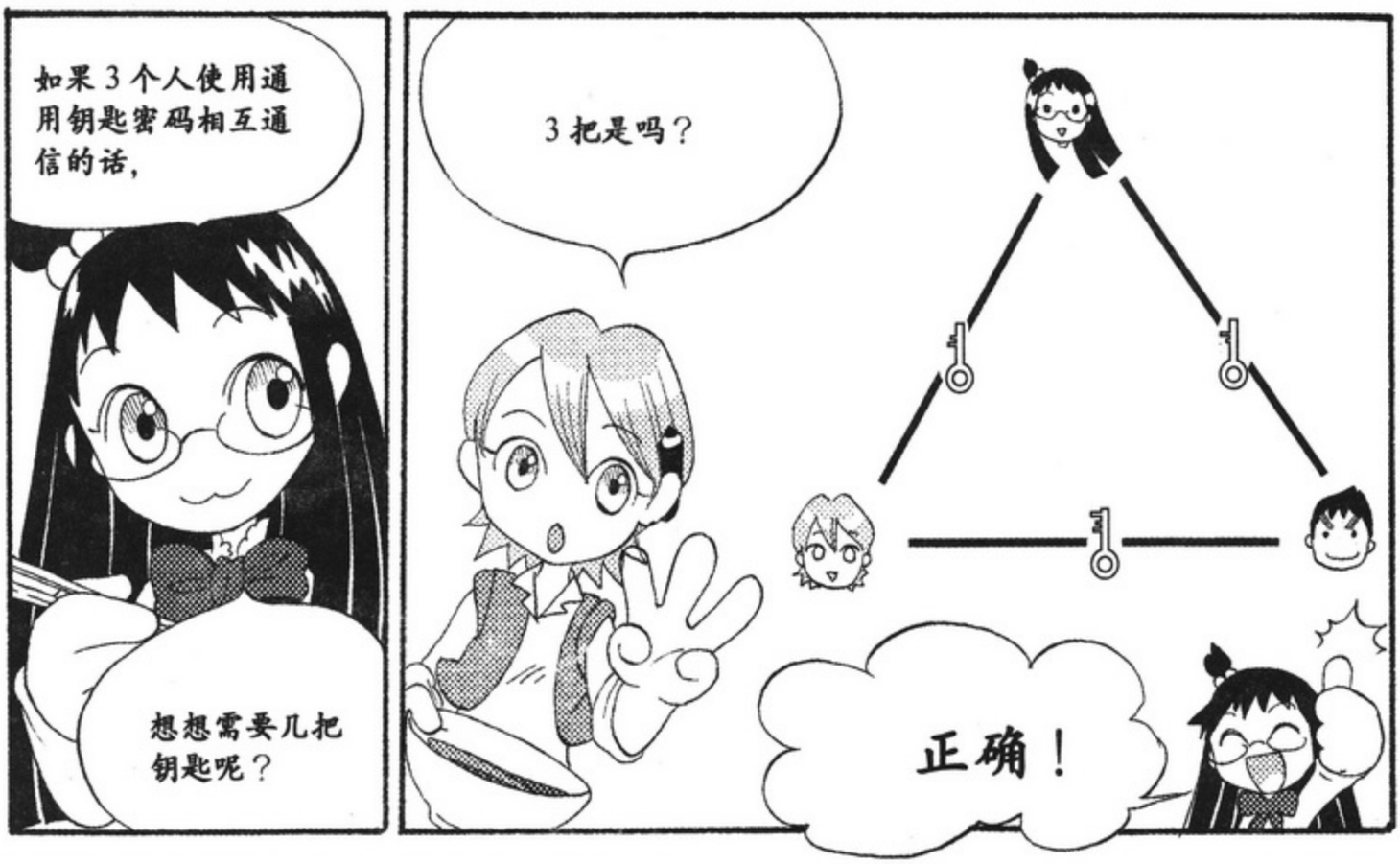
2-2 通用钥匙密码的定义

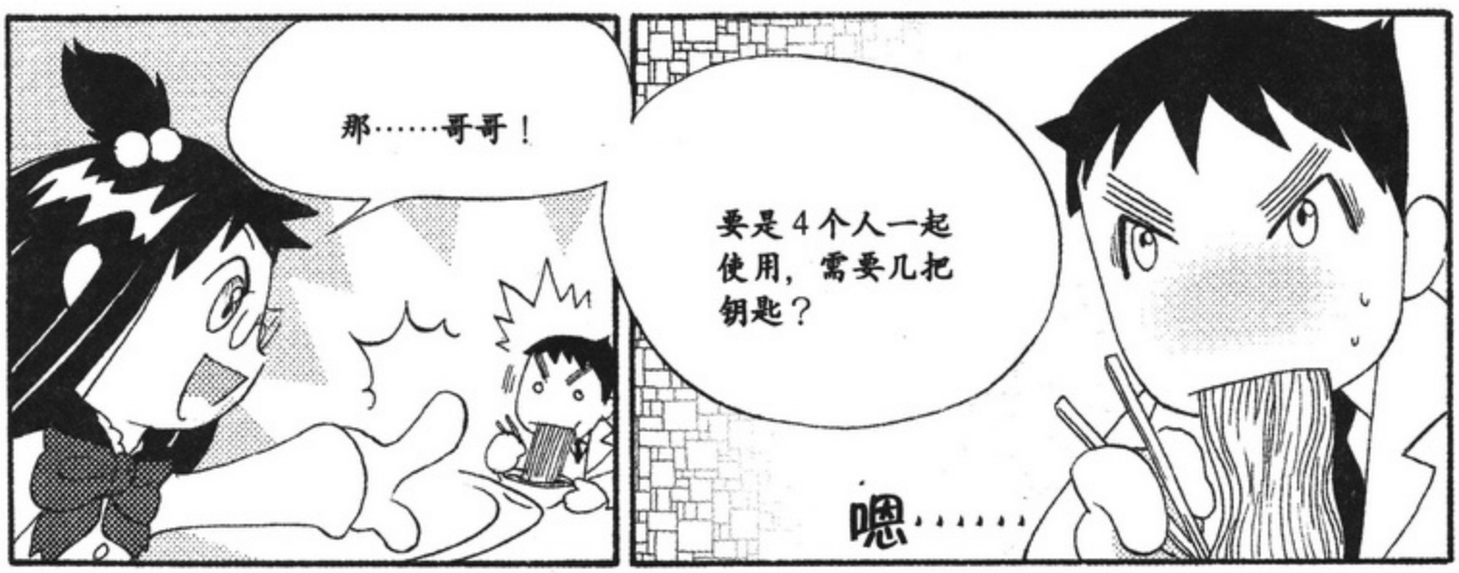




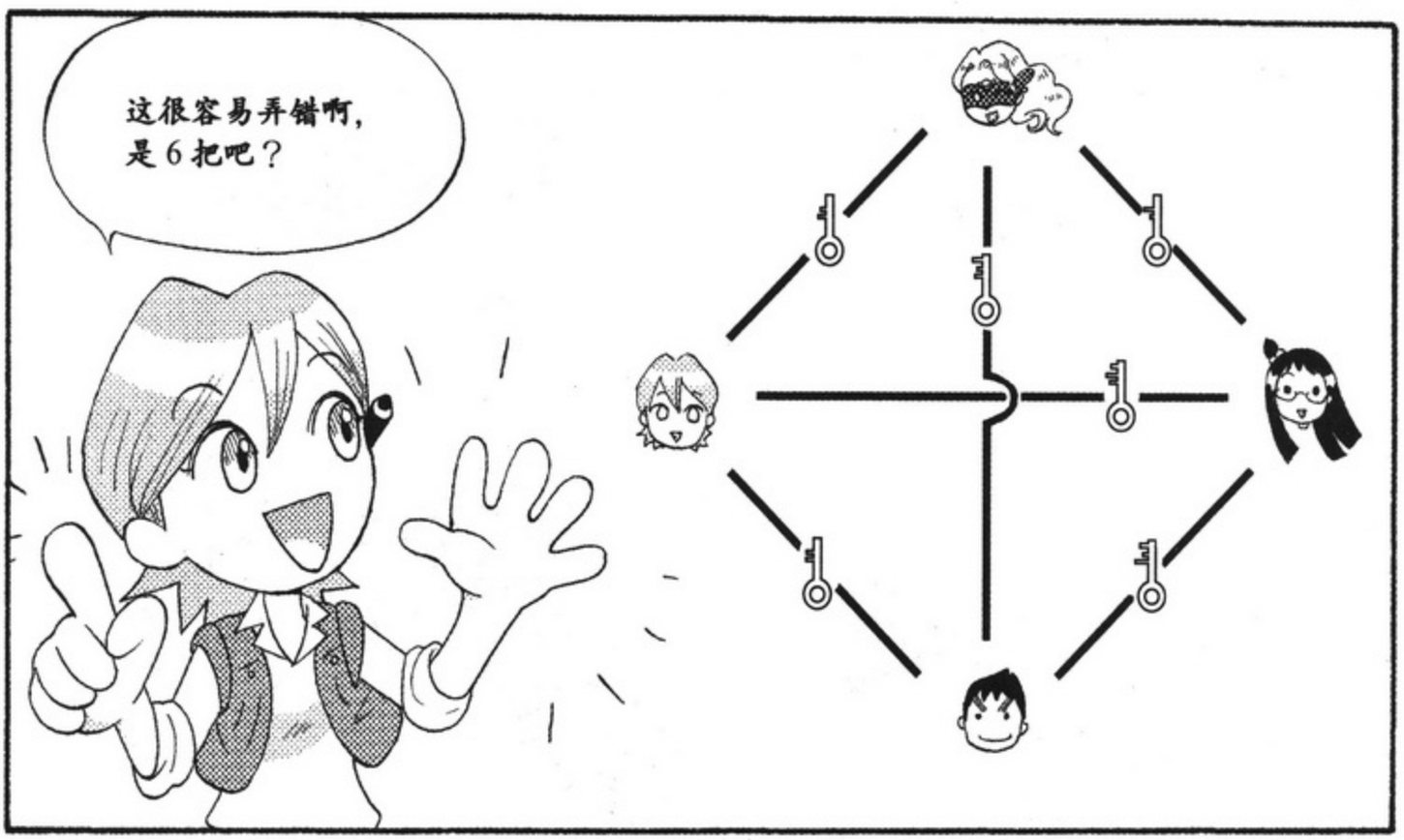


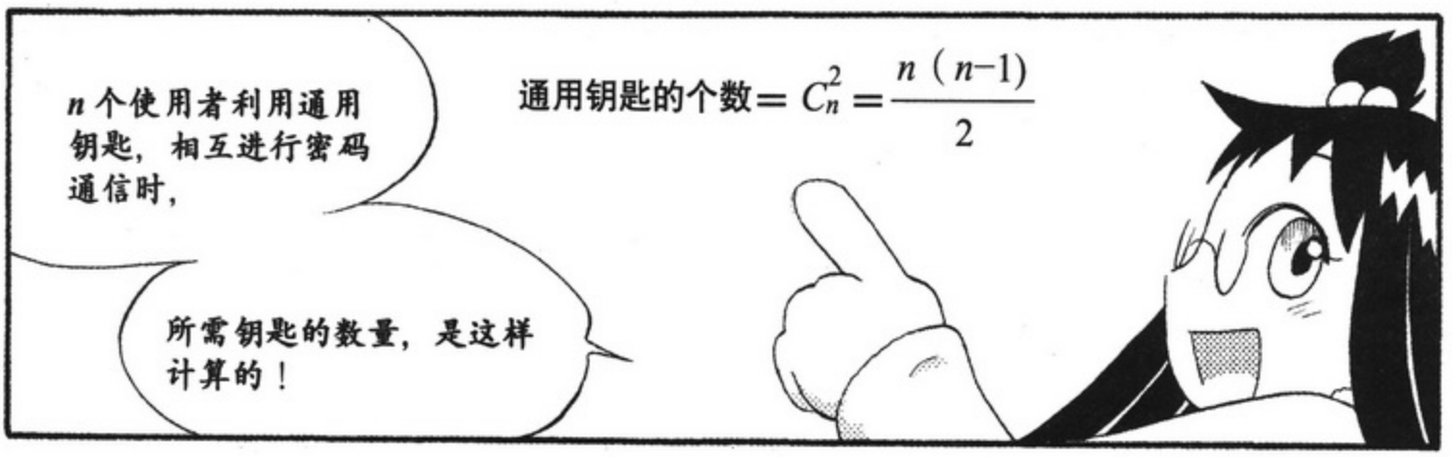
从特征来说，通用钥匙密码（Common Key Cryptography）也被称作对称密码（Symmetric Key Cryptography），或秘密钥匙密码（Secret Key Cryptography）。古典密码（惯用密码）全部属于通用钥匙密码。

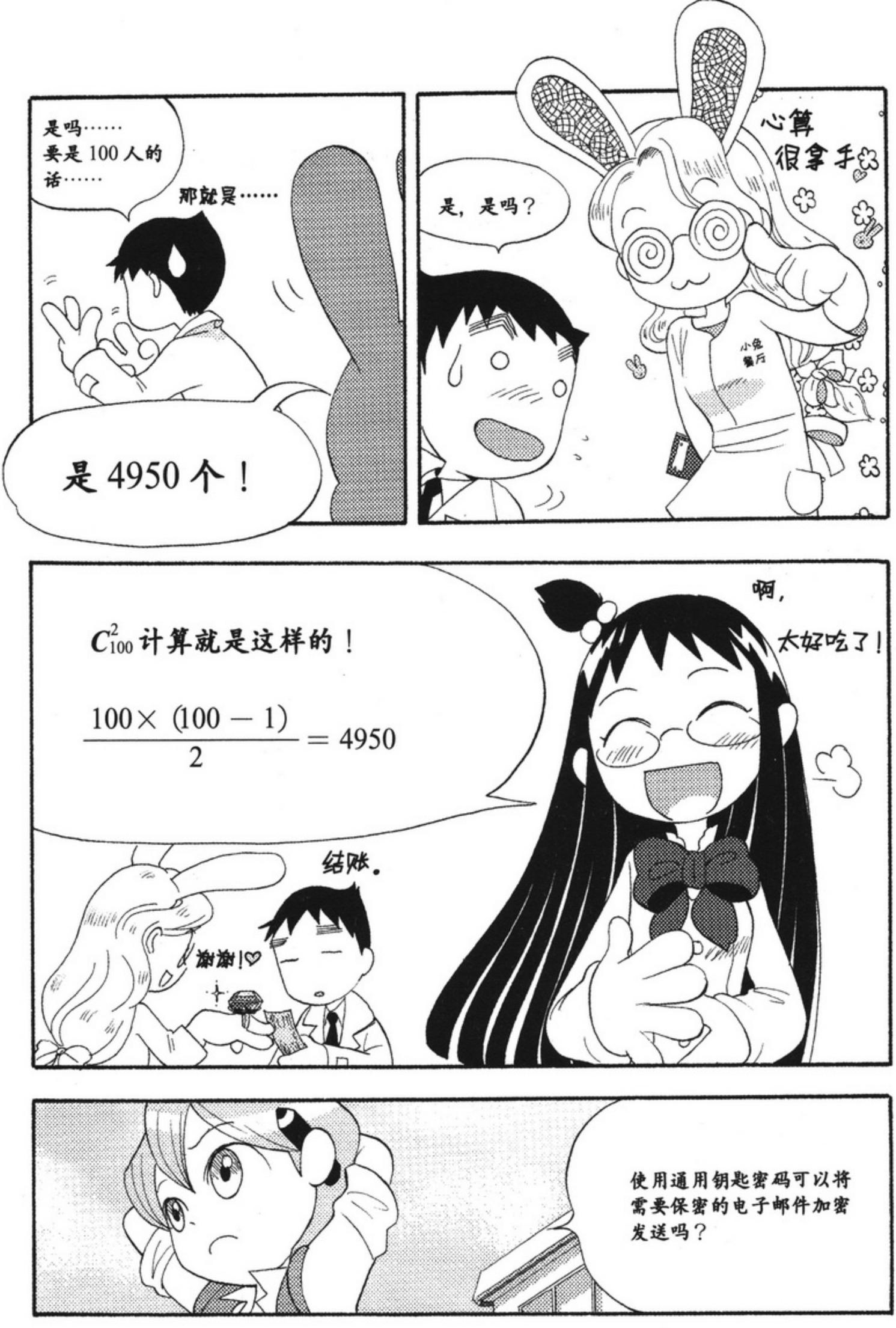





3个人用3把的话，
4个人就用4把？
大错特错！


这很容易弄错啊，
是6把吧？


n个使用者利用通用钥匙，相互进行密码通信时，
所需钥匙的数量，是这样计算的！
通用钥匙的个数＝ $C_n^2 = \frac{n(n-1)}{2}$


是吗……
要是100人的话……
那就是……
是4950个！
心算很拿手
是，是吗？
小兔餐厅
啊，太好吃了！
C_{100}^{2} 计算就是这样的！
$\frac{100\times(100-1)}{2}=4950$
结账。
谢谢！♡
使用通用钥匙密码可以将需要保密的电子邮件加密发送吗？

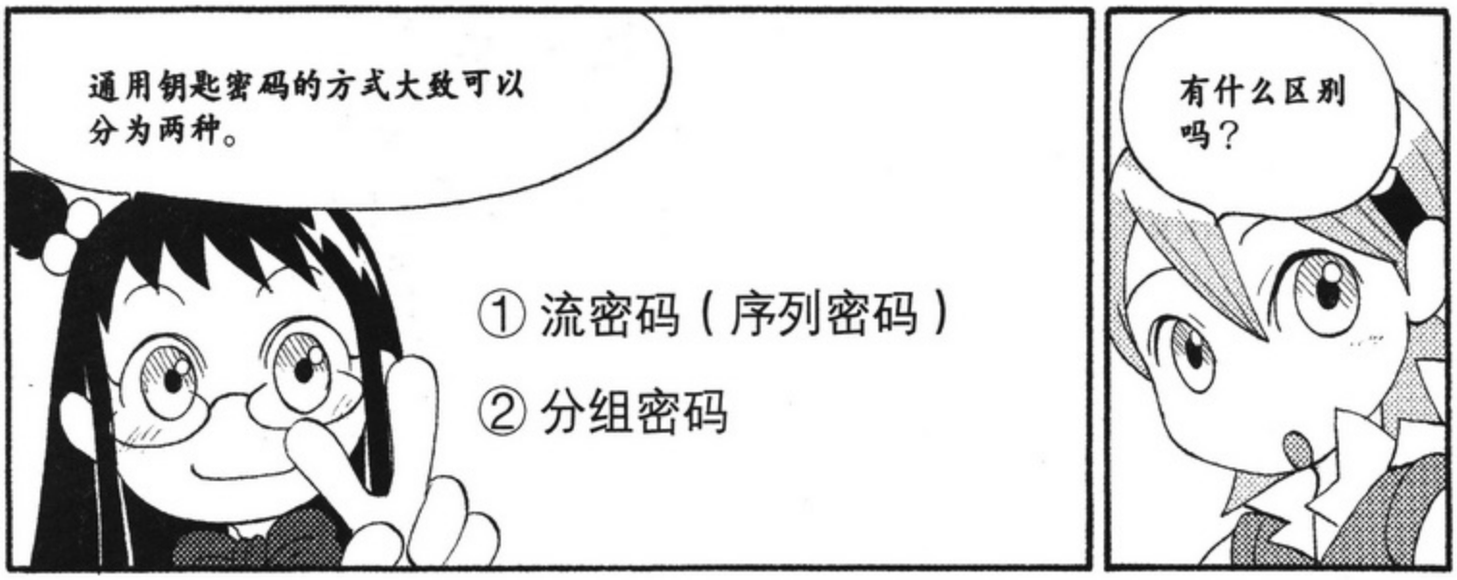

是呀！
但是怎么才能将通用钥匙交给收件人也是个问题啊！
钥匙也用电子邮件发送不行吗？
如果钥匙被盗的话，不就没有意义了吗？
那么，那个钥匙用别的通用钥匙加密不就行了？
那别的通用钥匙怎么发送呢？
用电子邮件……
咦？
别的钥匙用别的钥匙加密……
然后别的钥匙再用别的去加密……
这个……
这个……
这个嘛……

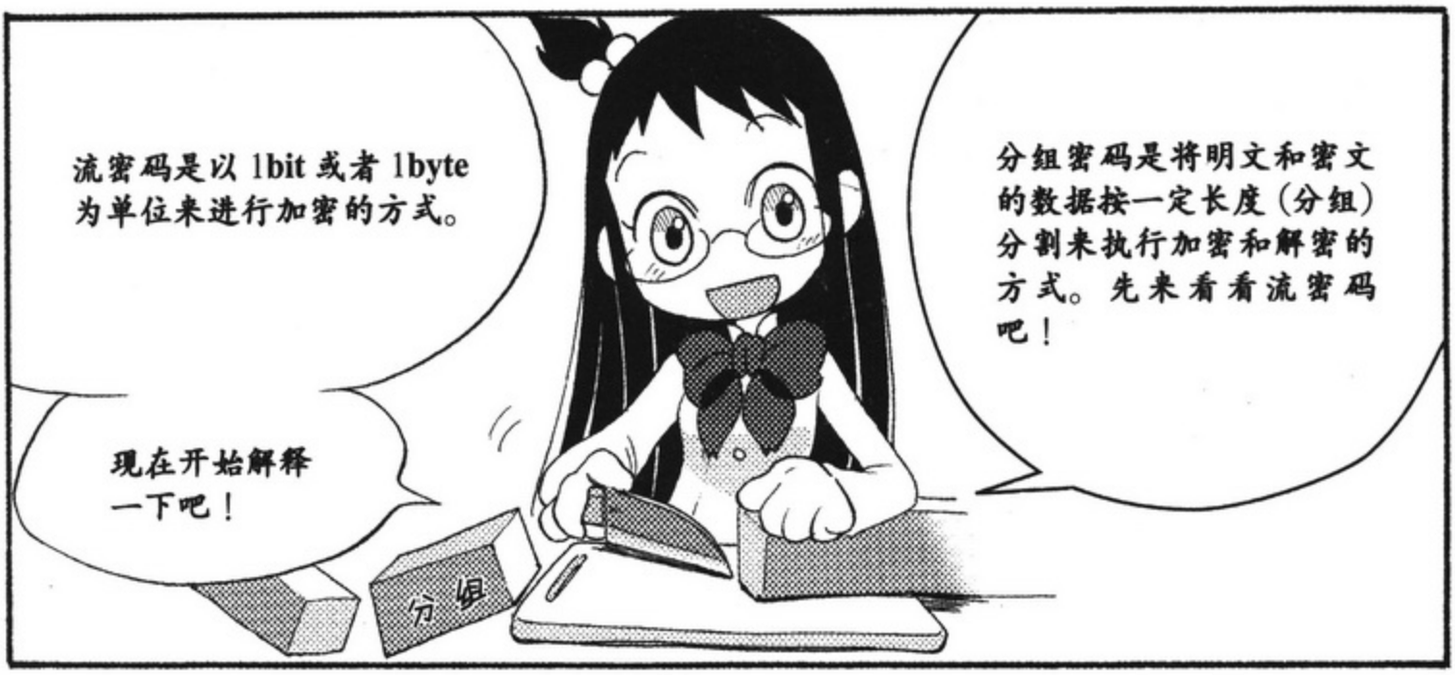
✤ 通用钥匙密码的特征

• 在发送和保管时，要注意钥匙不能被他人所知。

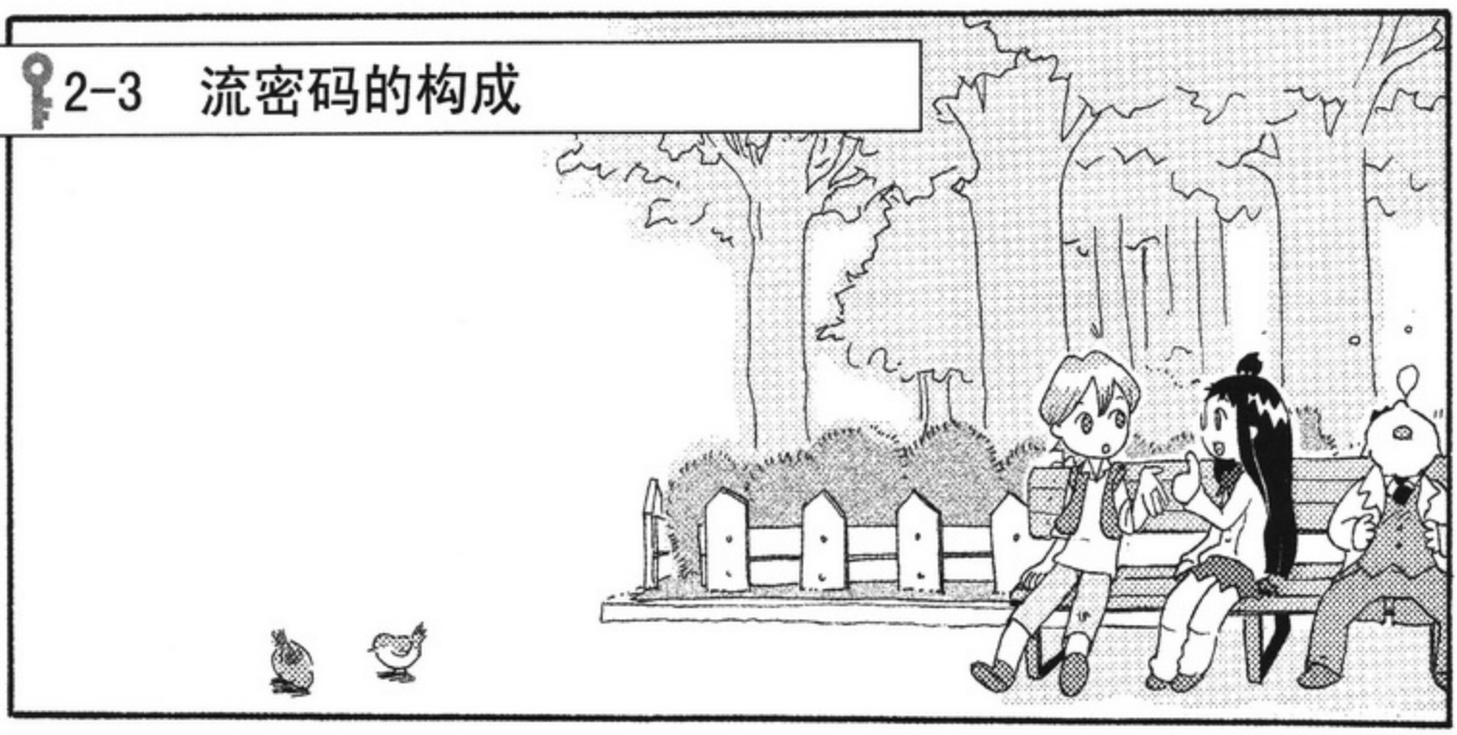
• 运算数量较少，可以高速加密和解密，适合于庞大的数据传输中使用。

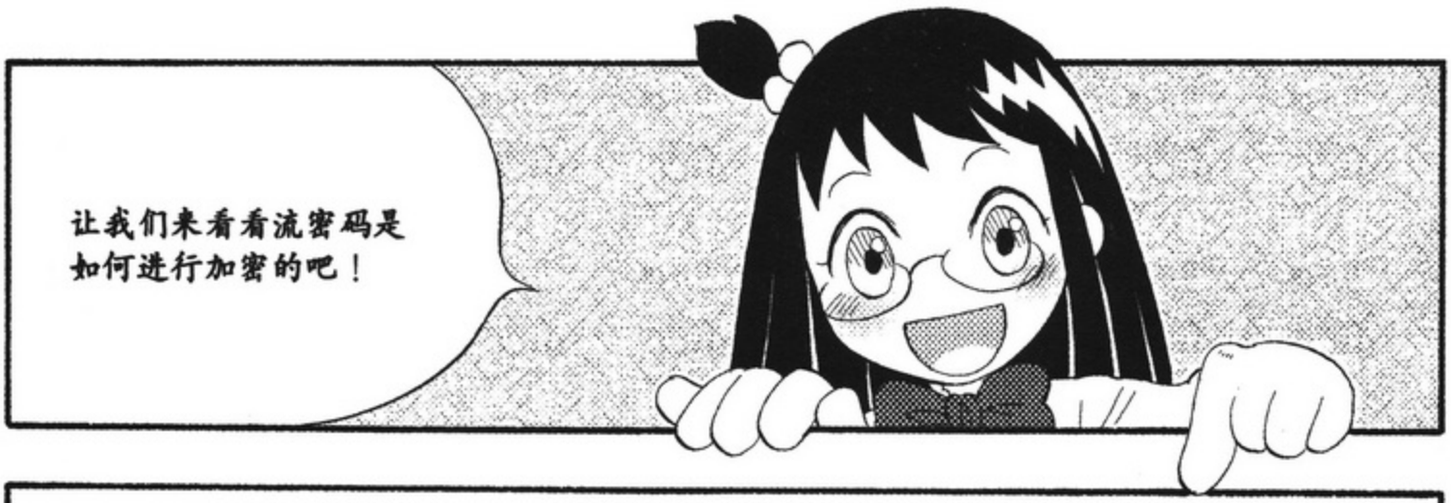
• 需要使用多个钥匙及管理，因此不适合对非特定多数的传输。


通用钥匙密码的方式大致可以分为两种。
① 流密码（序列密码）
② 分组密码
有什么区别吗？


流密码是以 1bit 或者 1byte 为单位来进行加密的方式。
现在开始解释一下吧！
分组
分组密码是将明文和密文的数据按一定长度（分组）分割来执行加密和解密的方式。先来看看流密码吧！

2-3 流密码的构成





Stream 就是流动的意思。由于是逐次进行加密和解密，所以在移动电话通信等领域中都有所应用。

钥匙是计算机生成较长的模拟随机数列（不规则的数字排列）。明文的数据和钥匙只是用按照一定顺序使用 XOR 运算来进行加密的。

与分组密码相比，操作更加简易，可进行高速处理。

“RC4” 和 “SEAL” 等是具有代表性的流密码。

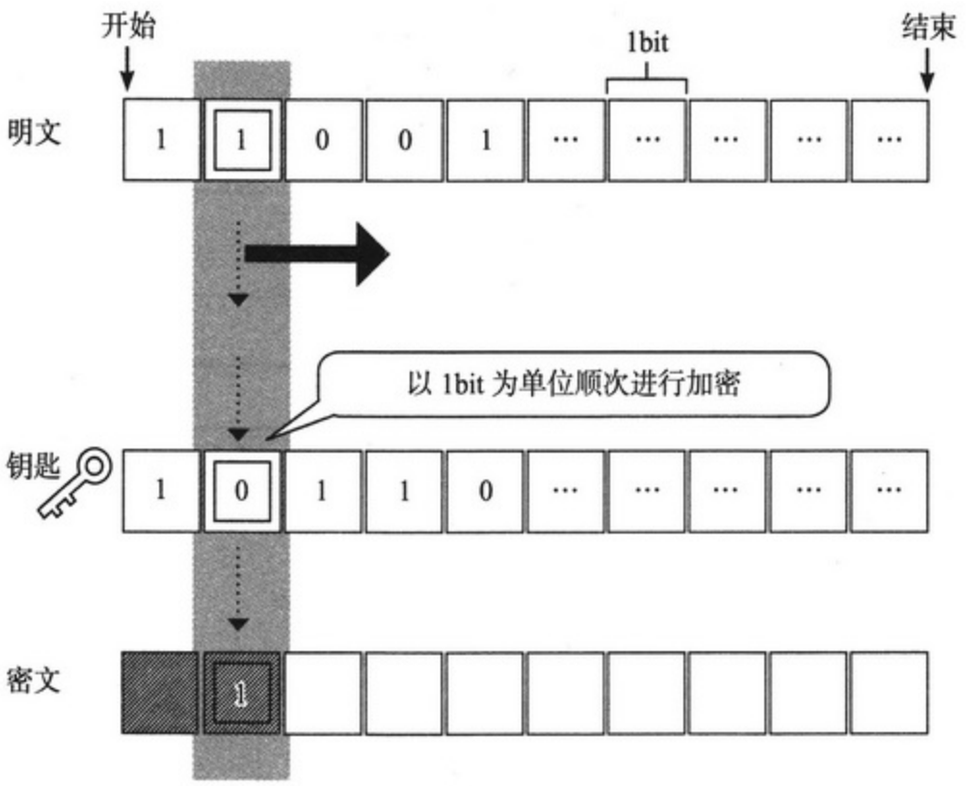


图 2.2　流密码的构成


“模拟随机数列”是什么呢？
模拟随机数列的意思是看起来类似于随机数列的数列。
（参照第 225 页）


顺便说一句，不管是流密码也好，还是分组密码也好，如果一一尝试的话，总会找到钥匙的！
（参照第 79 页）
只能在计算量上保证安全强度啊！
嗯


不能生成真正的随机数列吗？
要是能生成比明文还长的真随机数列，把这个数列作为钥匙，不就成了绝对安全的弗纳姆密码了吗？
用计算机是很难做到的！


咦？
哥哥呢？
？？？
是啊，人呢？


哈哈！
哈哈！

流密码是以 1bit 为单位进行加密，分组密码则是按照一定的分组进行加密（图 2.3）。1 个组的长度将随着密码而改变，其中有 64bit 和 128bit 两种类型。此外，1byte 的文字等同于 8bit，假设 64bit 为 1 组的话，那么 8 个文字称为 1 组。

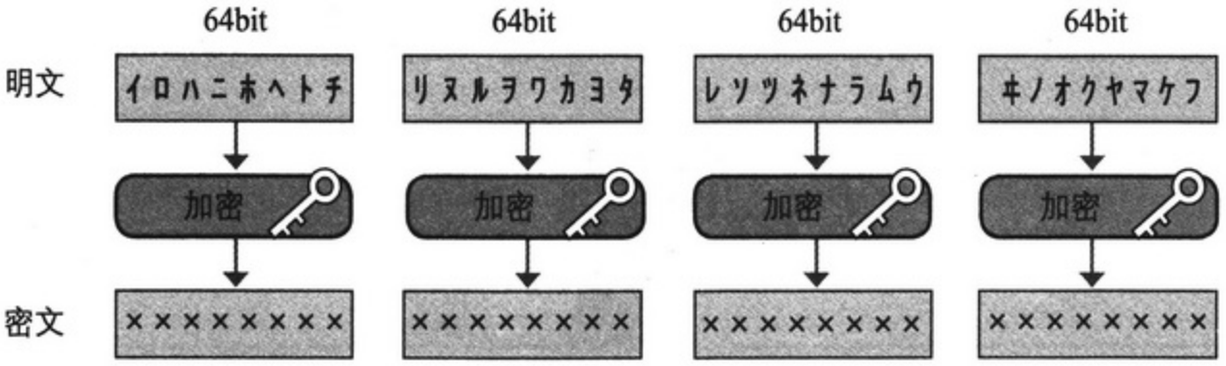
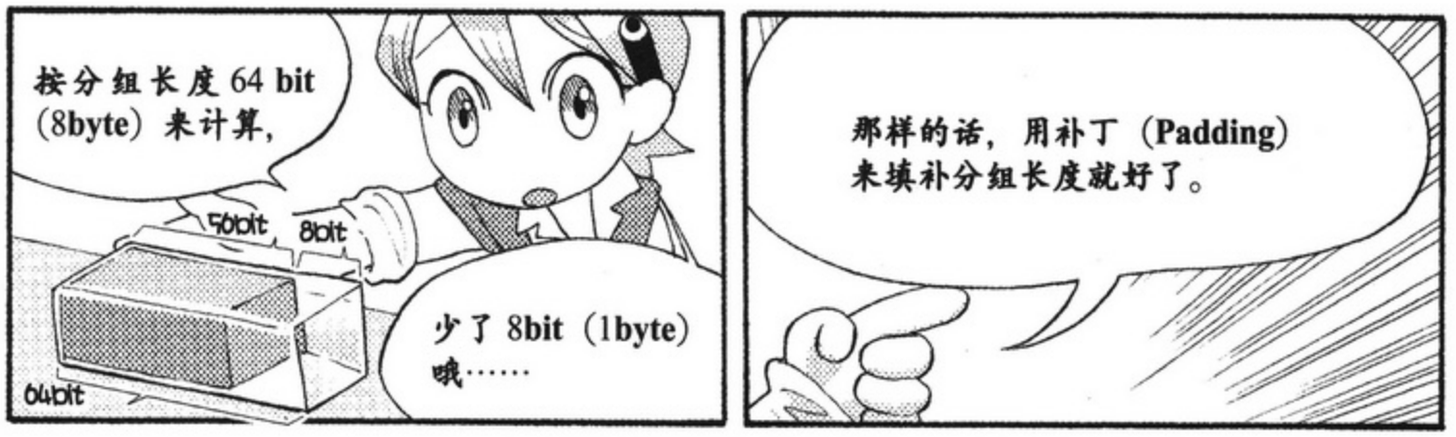


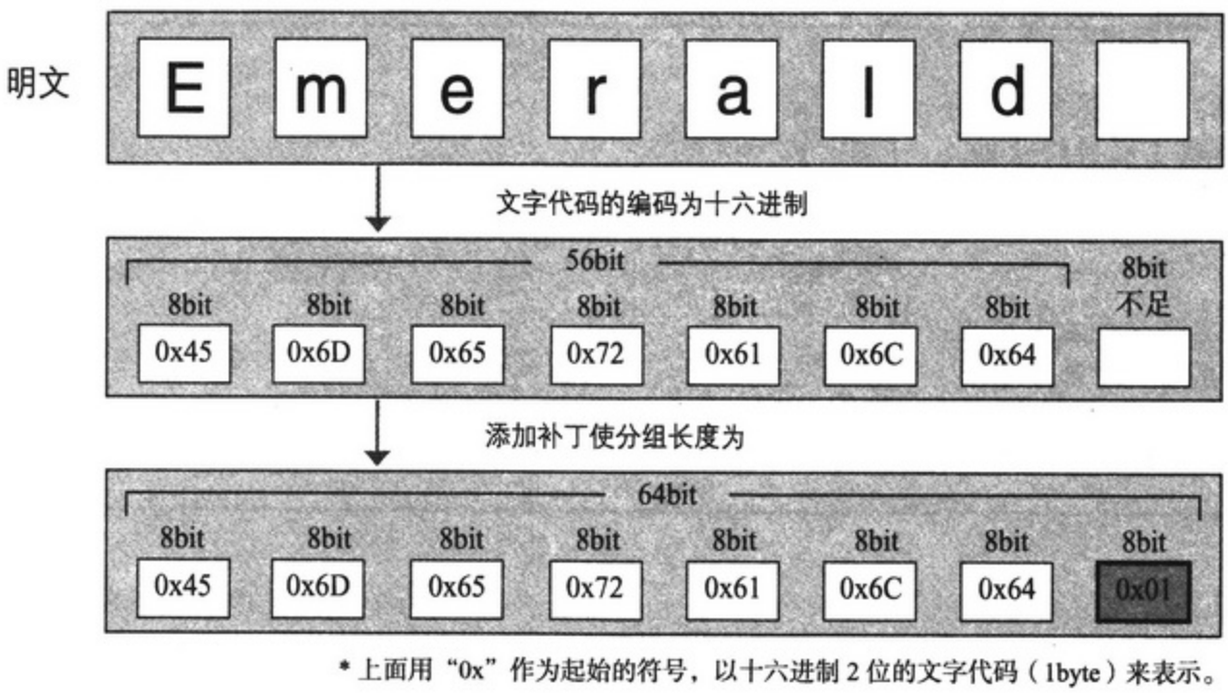
图 2.3 分组密码的构成

将 1 组设定为 64bit 的原因是，随着计算机处理性能的提高，可以用很少的计算次数来完成运算。此外，如果过度减少 bit 数的话，将会出现安全强度方面的问题。

表 2.4 分组密码的种类

密码名称	分组长度 / bit	钥匙长度 / bit
DES	64	64
AES	128	128　192　256

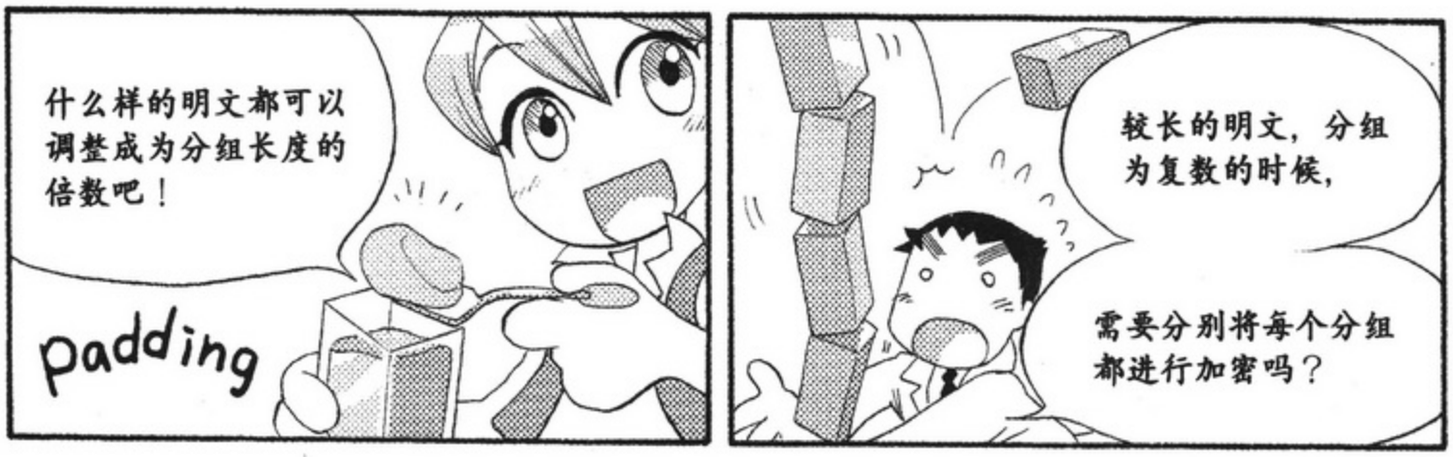


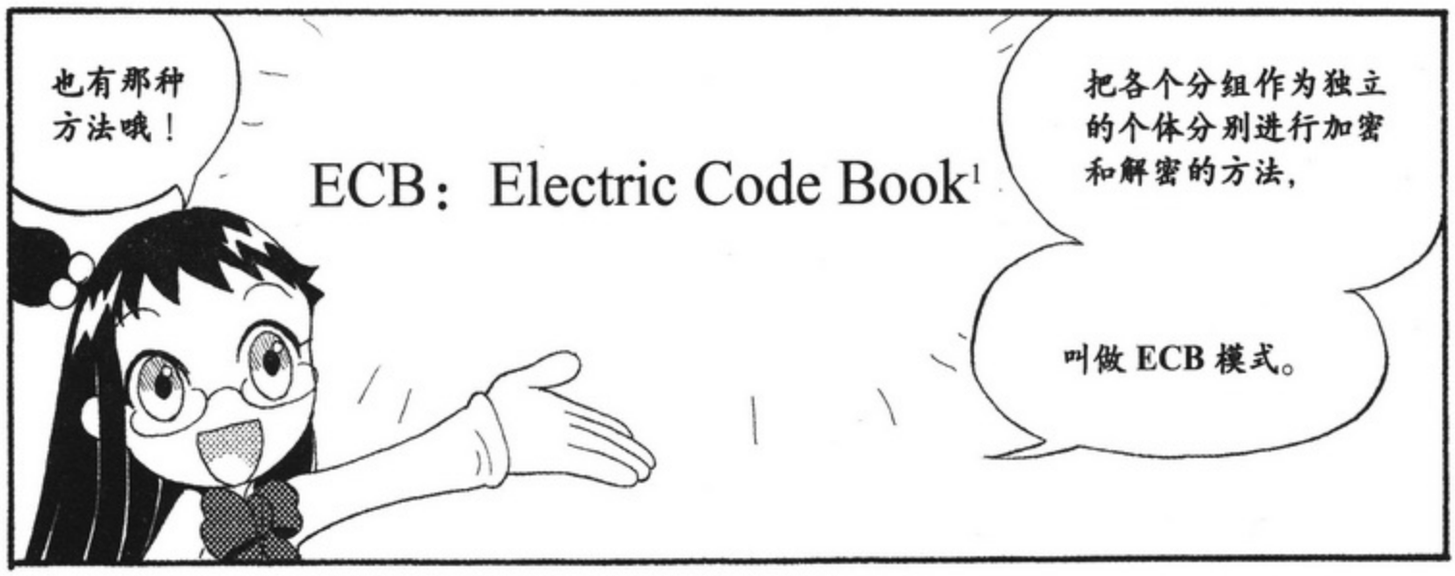


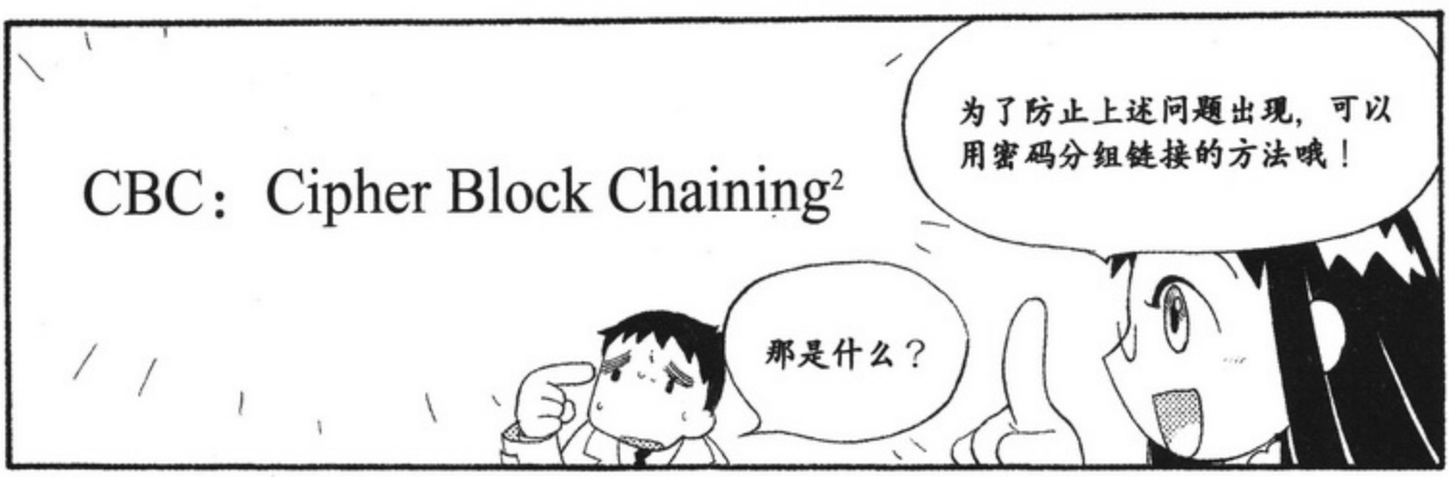
* 上面用“0x”作为起始的符号，以十六进制 2 位的文字代码（1byte）来表示。

图 2.4　使用补丁的例子

上图的例子就是加入 0x01，使分组长度等于 64bit（8byte）。0x01 为 1，即表示解密的时候去掉作为补丁的 byte 数。补丁的用法除此例之外还有其他若干种方法。







1. 电子编码本。　2. 密码分组链接。

✤ CBC 模式

CBC 模式是，可以将内容相同的明文设法生成不同的密文的方法。

将明文进行分组并组合时，将输入数据或者输出数据的一部分，返回到分组单位中，进一步将明文中的分组之间关联情报进行分散，从而增强密码的安全强度。

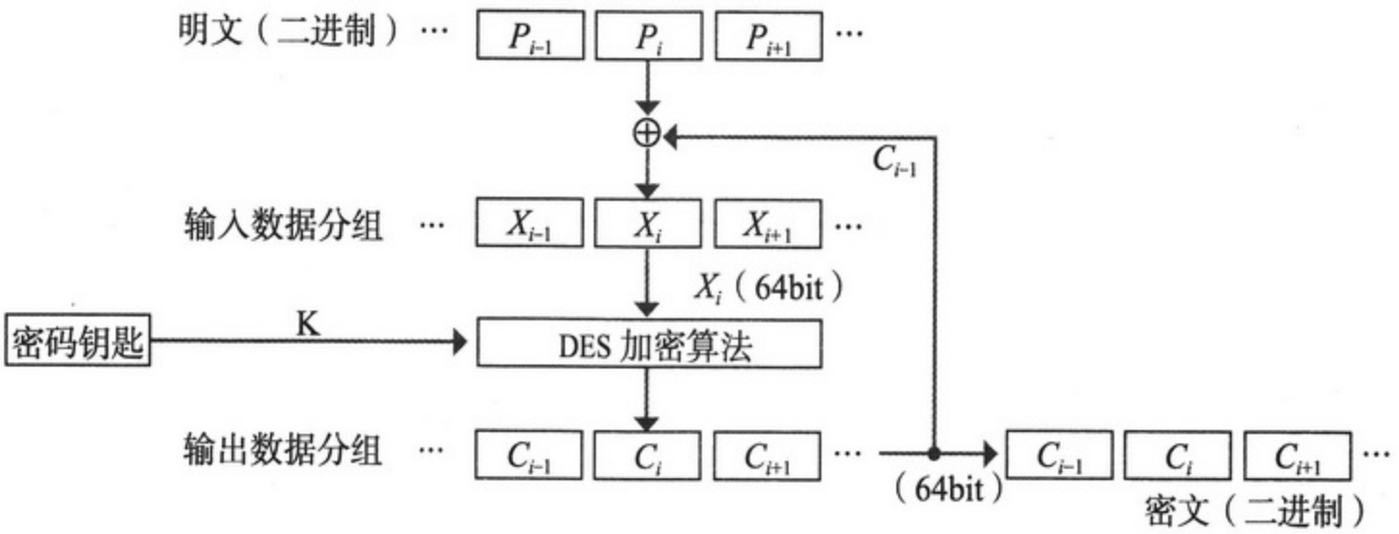


图 2.5　CBC 的结构

参照图2.5的举例，如果将前一个密文分组C_{i-1}和下一个明文分组P_i进行异或（XOP）运算，即

$$X_i = C_{i-1} \oplus P_i$$

将运算结果X_i，作为将要进行加密的输入数据。这样的话，即使有相同数据的明文分组，也不会生成相同数据的密文。但是，最初的分组，因为不具有相对应的返回数据C_0，所以需要另行设定开始前的密码分组即初始值（Initial Vector）。

这样将复数的分组进行加密的规则称为模式化。在分组密码模式中，包含上述的 CBC 模式，除前页提到的 ECB 模式之外，还有 OFB（Output Feedback, 输出反馈模式）模式和 CFB 模式（Clipher Feedback, 密码反馈）等。

2-5 DES密码的构成

为了更好的理解 DES 密码的构成，请阅读第 87 页的“简易版 DES 加密和解密详解”。

虽然世界上最先标准化的商务用密码就是 DES 密码，

但这个密码是有前身的！

最初的民间标准规格密码：
DES（Data Encryption Standard）

加密系统的前身：
Lucifer Cipher（金星密码）

研发人：IBM 公司的 Horst Feistel

哦！

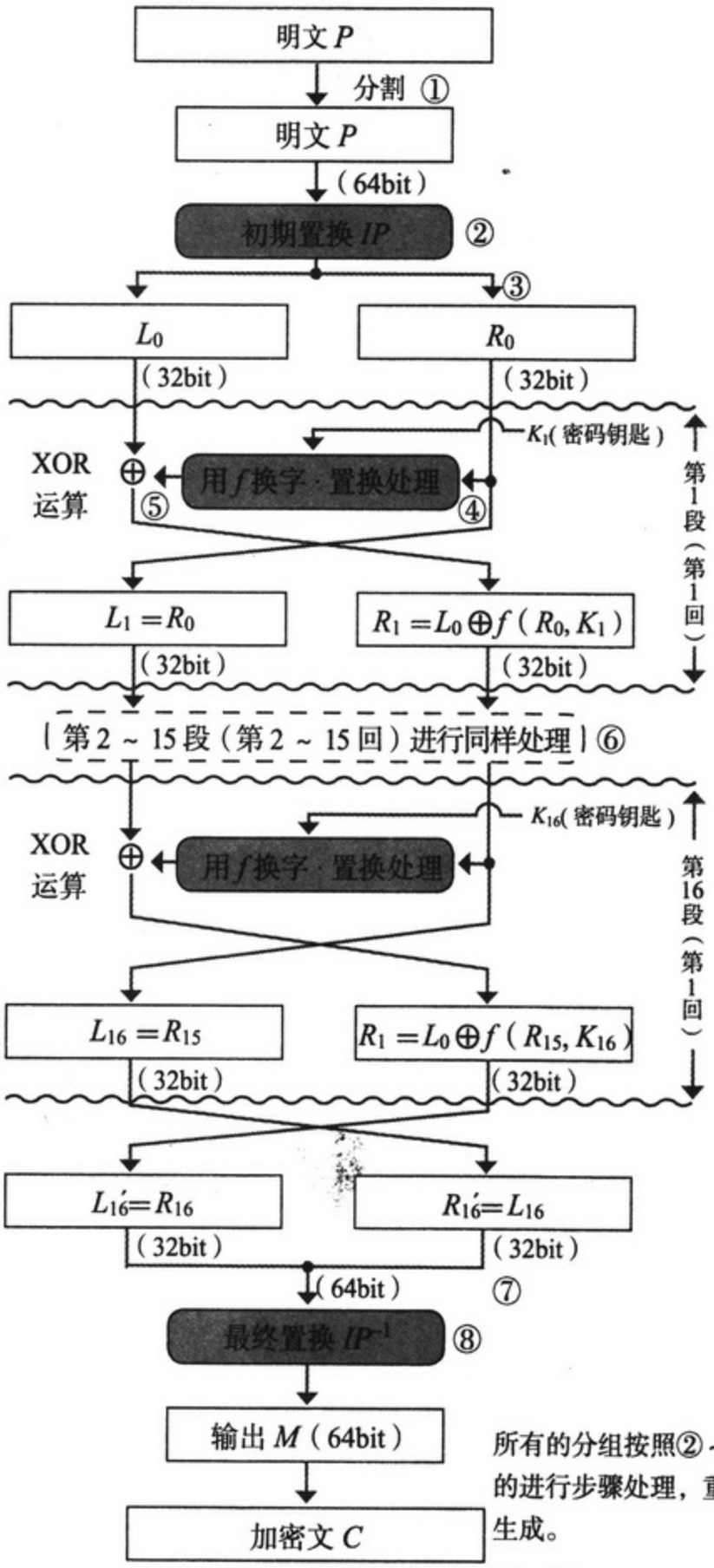
实际的构成是怎样的呢？
用实例来说说吧！

好的！

那就阅读一下第 87 页的“简易版 DES 加密和解密详解”吧！

❀ Feistel 类型密码的基本结构

费斯特鲁（Feistel）提出的加密方法，如下所示。



所有的分组按照②~⑧的进行步骤处理，重新生成。

图 2.6 DES 加密顺序（加密文的生成）

①将明文以 64bit 为单位，进行分组。

②64bit 的分组进行初期置换 IP[1] 后，将其 bit 替换，再重新零散编组。

③将 64bit 的分组，分割为左侧 32bit（L_0）和右侧 32bit（R_0）两个组。

④将 R_0 组，使用密码钥匙（K_1）的非线性函数 f，进行复杂的换字 · 置换处理。

⑤④处理之后，将 R_0 和 L_0 再进行 XOR 运算，从而生成新的右侧 32bit（R_1）。原来的 R_0 则生成新的左侧 32bit（L_1）。

⑥将④和⑤的处理过程称为是第 1 段（第 1 回），对第 2 段（第 2 回）到第 15 段（第 15 回）进行同样处理。

⑦将左侧 32 bit（L_{16}）和右侧 32 bit（R_{16}）的 2 个组，重新构成一个 64bit 组。

⑧最后，进行与初期置换相反的 IP^{-1} 处理，即完成 1 组的加密。

1. IP：Initial Permutation

✤ Involution

Involution 是，经过 2 次变换而返回到原来状态的方式。

例如： 我们假设变换方式为 1 变为 4，2 变为 3，3 变为 2，4 变为 1。可以看出该变换是，变换前与变换后的值一一对应。再次进行同样变换可看出如下结果，

$1 \rightarrow 4 \rightarrow 1$

$2 \rightarrow 3 \rightarrow 2$

$3 \rightarrow 2 \rightarrow 3$

$4 \rightarrow 1 \rightarrow 4$

全部返回到原来的数值。这样的变换称为 Involution。Involution 与 DES 密码的解密方法有着很深的关系。

DES 和 Lucifer 是相同的吗？

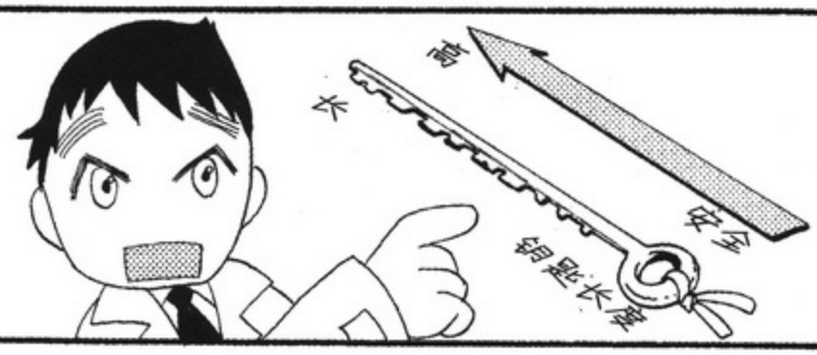
钥匙长度是完全不同的哦！

DES 的钥匙长度比 Lucifer 要短得多，只有 64bit 哦！

DES 整体的钥匙长度为 64bit，但是在实际应用中是 56bit，剩下的 8bit 则用于奇偶校验（检查错误）。

所谓奇偶校验，就是用来检查由于乱码的混入和读取失败等状况而产生的数据错误。

钥匙长度越长，安全强度就越高吗？



制定规格的时候，美国的 NSA（国家安全局）对钥匙的候补数量进行了限制。

钥匙的候补数只可在 10 京[1] 个以下。

也就是 56bit (2^{56}) ……

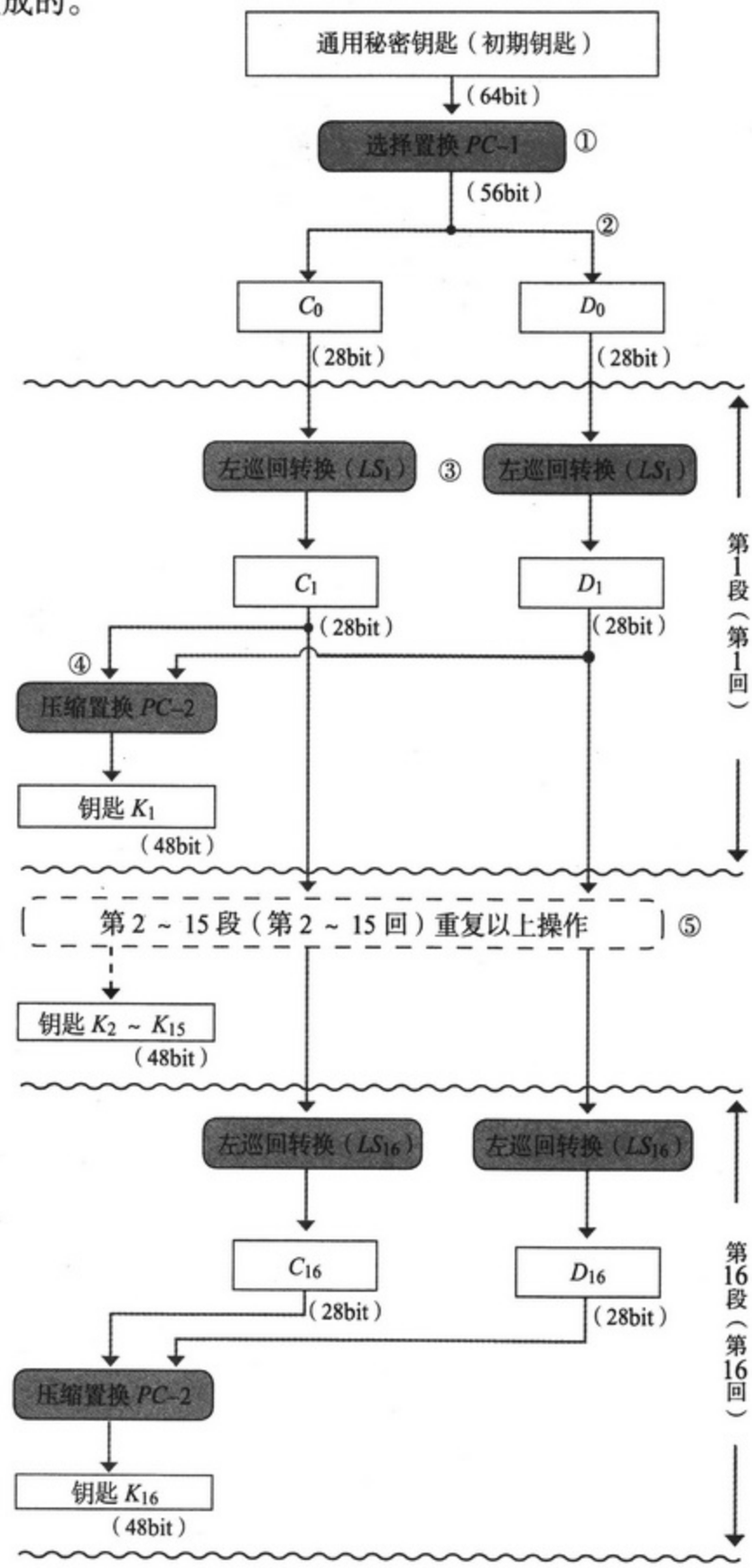
为什么？

1. 亿 $=10^8$ 兆 $=10^{12}$ 京 $=10^{16}$ 垓 $=10^{20}$。


如果钥匙的候补数过多的话，
NSA就无法对密码进行盗取了，是吧？
大概是这样。
じ——っ
NSA可是个世界性的谍报活动组织哦！
会经常做破解密码和探寻重要机密的工作！
哎哟
好可怕啊……
DES把钥匙按同样的道理经过不同的加工，
作为各段（回）的非线性函数f的常数来使用。

❀ DES 密码钥匙的生成

将 DES 加密时，使用的钥匙 K_1，K_2，K_3，…，K_{16} 是按每 1 段（1 回）构成的不同结果组成的。



①在 64bit 的初始钥匙中，除去检查错误使用的 8bit 之外，进行选择置换 PC^1–1。

②将 56bit 分为 2 组，左侧 28bit（C_0）和右侧 28bit（D_0）。

③将 C_0 和 D_0 中所有的 bit，进行左巡回转换（LS1），生成 C_1 和 D_1。

④将 C_1 和 D_1 合并，除去 8bit 并进行压缩置换 PC–2，生成 48bit 钥匙 K_1。

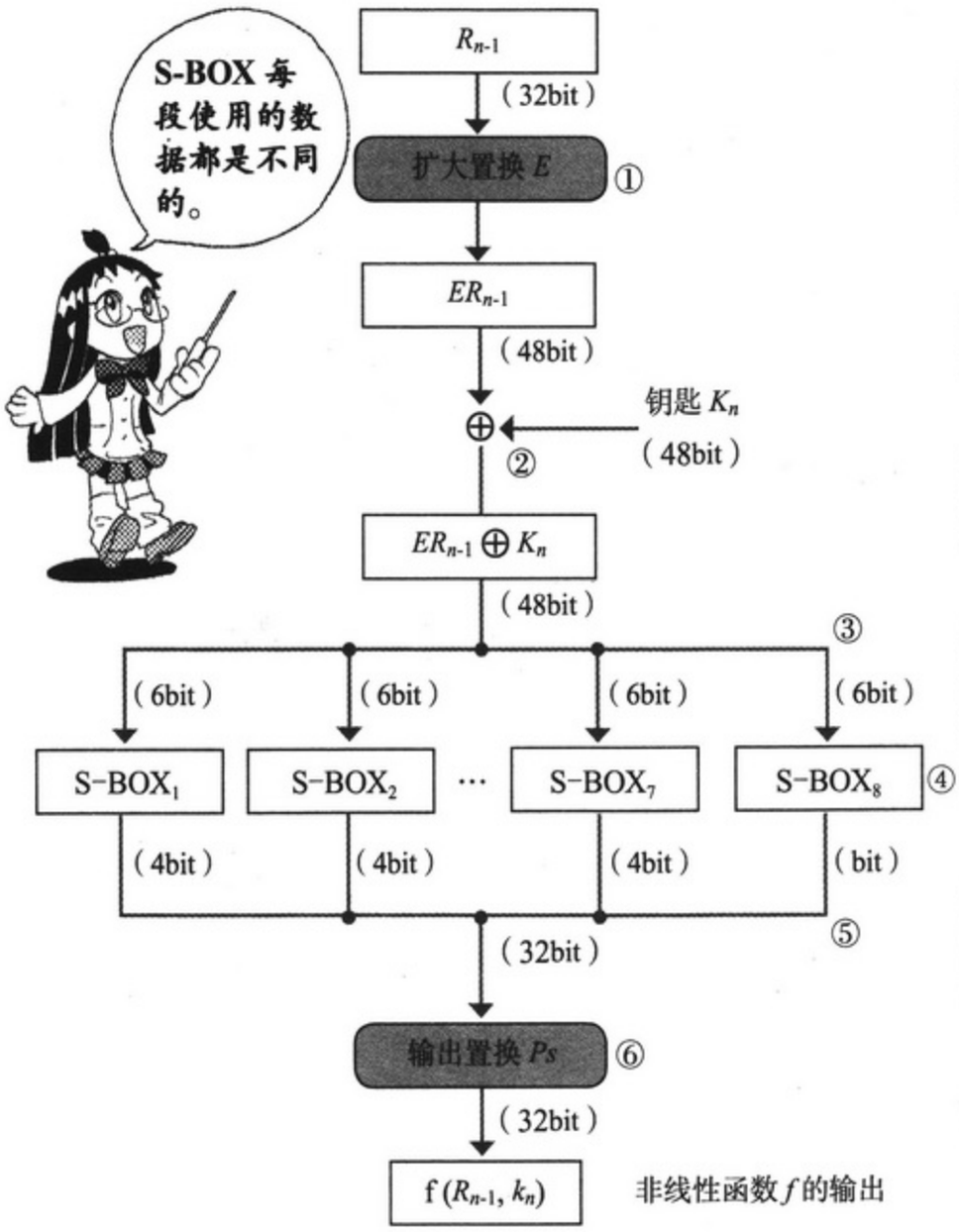
⑤重复③和④的处理，生成出在各段（回）加密中使用的钥匙 K_n。

解密钥匙生成时，由左巡回转换变为右巡回转换。解密钥匙生成的顺序与加密钥匙生成的顺序相反。按照 K_{16} ~ K_1 的顺序生成。

图 2.7　DES 密码的加密钥匙和解密钥匙的生成顺序

1. PC：Permuted Choice

✤ DES 非线性函数 f 的构成



①输入的左 32bit 数据，与钥匙合并为 48bit 的扩大置换 E。

②将该数据，与钥匙进行 XOR 运算。

③运算的结果，分割为 8 个 6bit 数据。

④将 6bit 数据，通过各个 S-BOX（变换表），按照从第一个 box 至第八个 box 的顺序，全部变换为 4bit。

⑤把从 S-BOX 输出的数据，按照顺序进行合并，重新构造成 32bit。

⑥最后置换为 PS 的数据，成为函数 f 的输出数据。

图 2.8　DES 的非线性函数 f 的构成

✤ DES 加密和解密的基本构成

DES 加密和解密方法，如下所示，明文的加密变换处理和密文的解密处理是相反的作业流程。

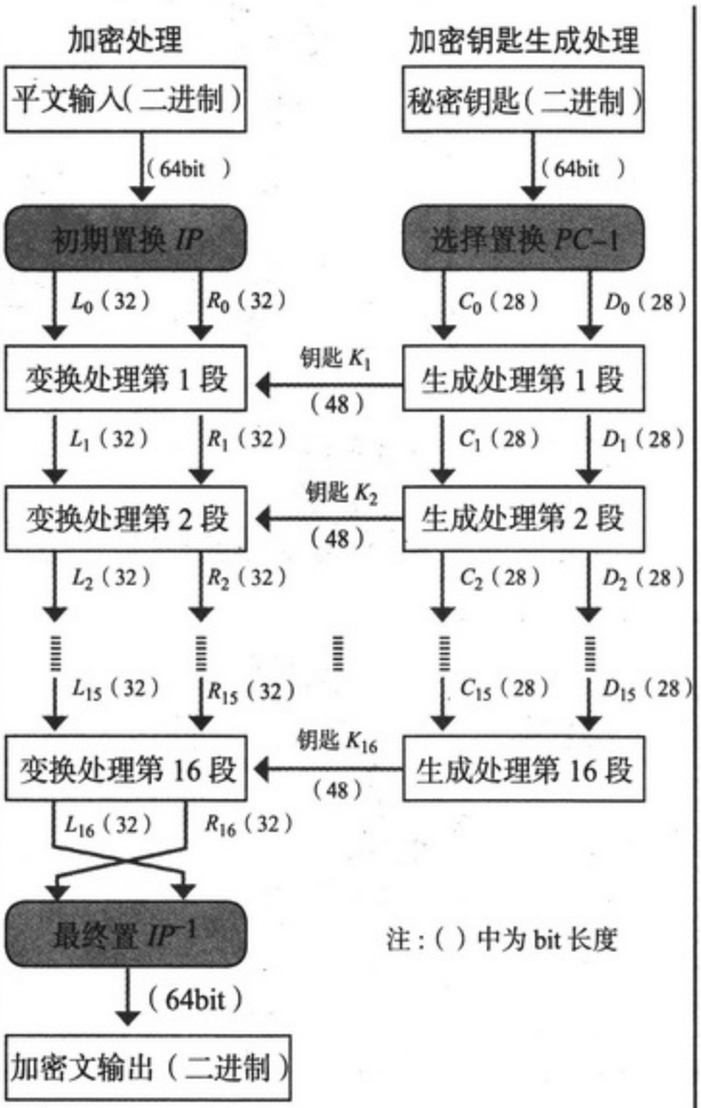


图 2.9　DES 加密的基本构成

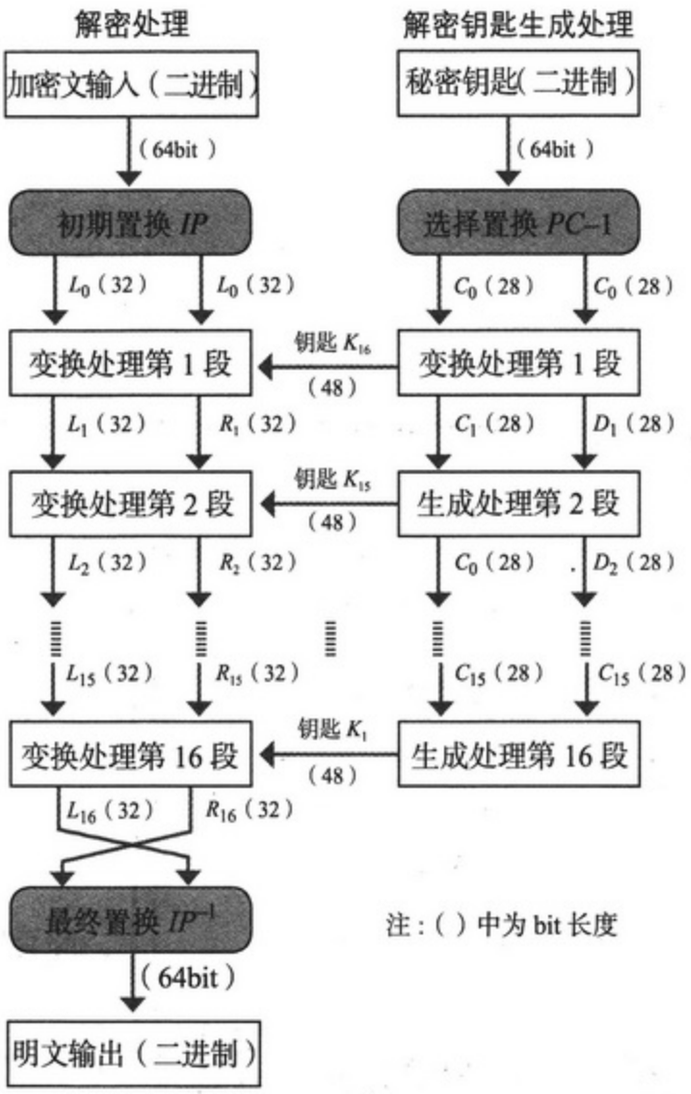
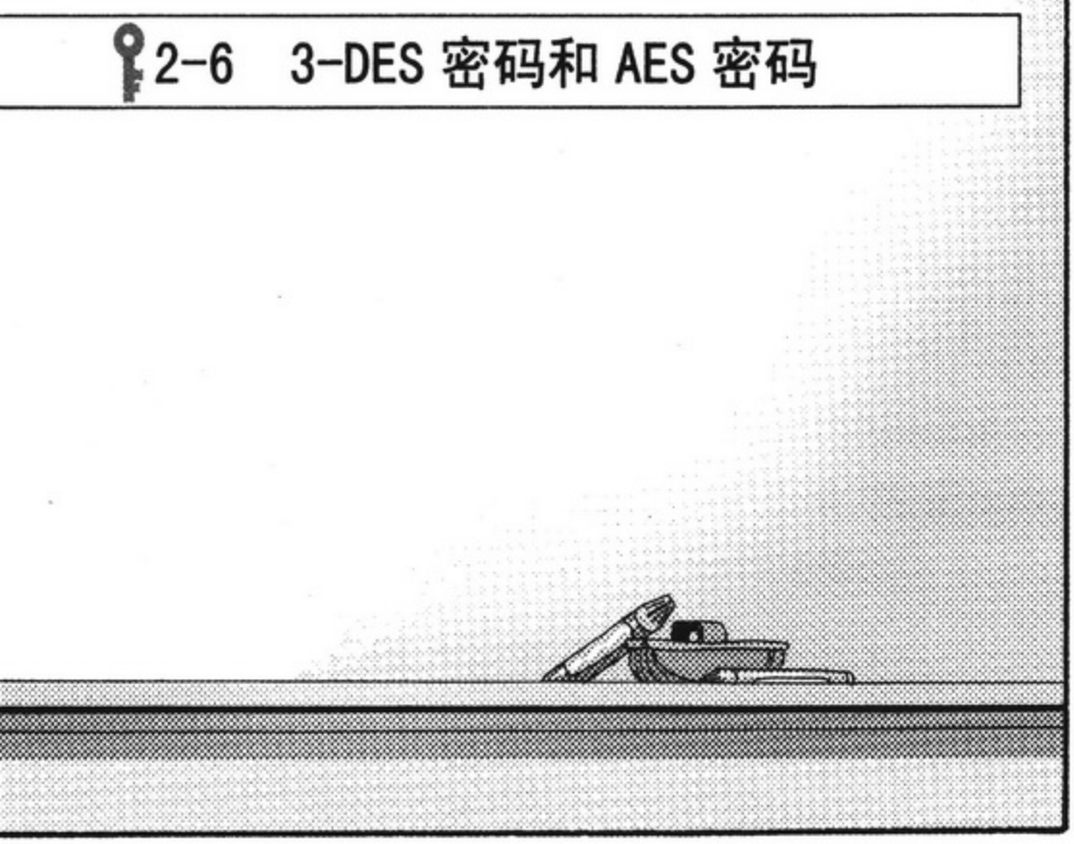


图 2.10　DES 解密的基本构成

2-6 3-DES 密码和 AES 密码



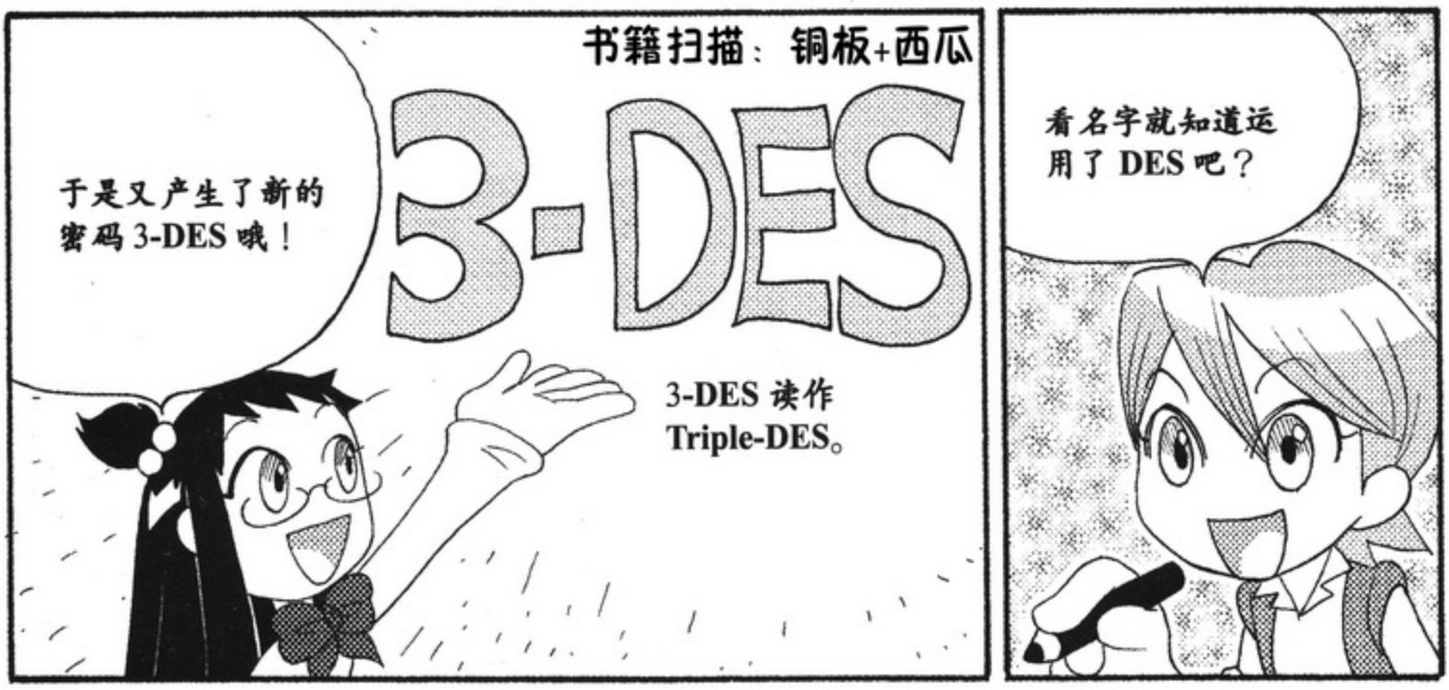
DES 的缺点

- 钥匙长度很短。钥匙长度短的话，处理速度会变慢，也容易被破解。
- 因为没有 S-BOX 的设计标准，实际应用时作用很薄弱。

表 2.3　分组密码破解法

全数检索分组法	检索全部数据寻找钥匙的方法
差分破解法	利用将输入差分保持原状变为输出差分的 XOR 运算性质来找出钥匙的方法
线性破解法	把 S-BOX 变为近似线性（近似一次函数的直线），以输出概率来推定的方法


在 1990 年，DES 类的密码，
用全数检索法和线性破解法就可以破解。
啊？
好不容易学会的。
这可怎么办啊！


书籍扫描：铜板+西瓜
3-DES
于是又产生了新的密码 3-DES 哦！
3-DES 读作 Triple-DES。
看名字就知道运用了 DES 吧？

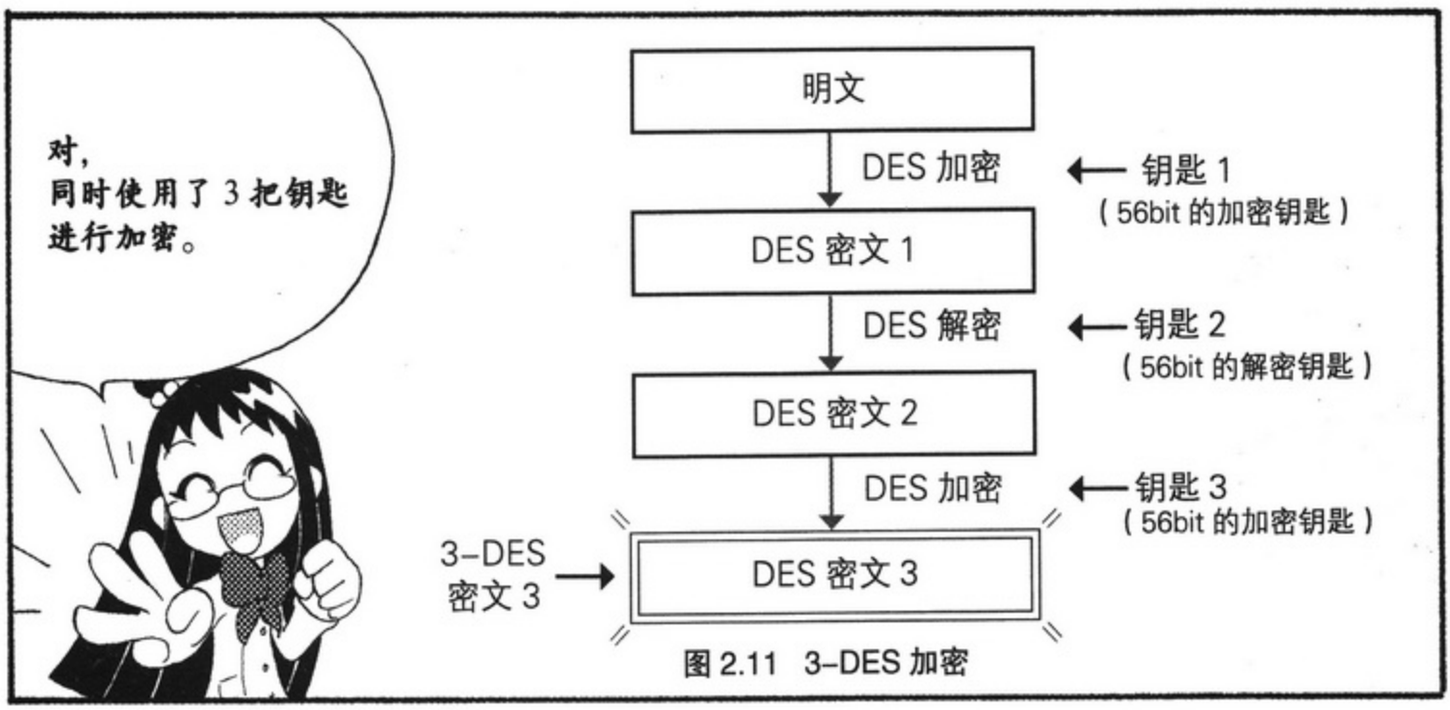
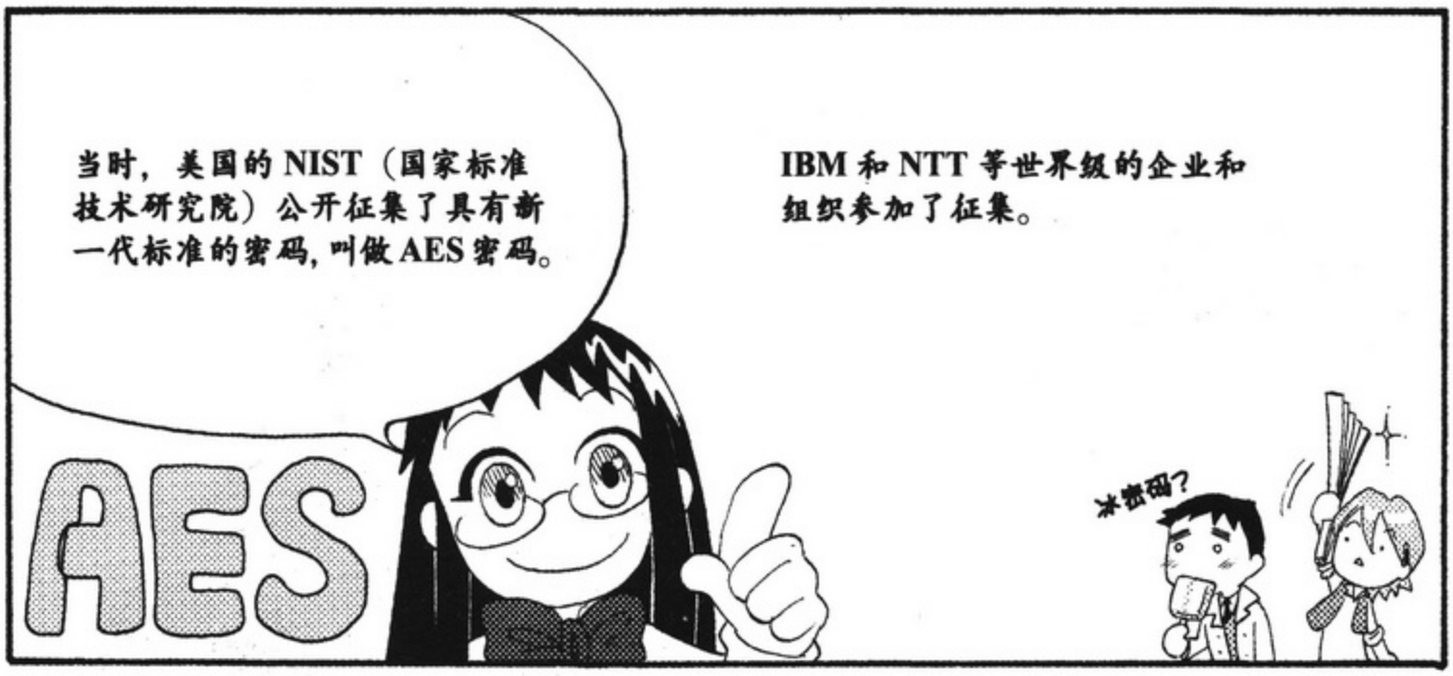

对，
同时使用了 3 把钥匙进行加密。
明文
DES 加密
钥匙 1
（56bit 的加密钥匙）
DES 密文 1
DES 解密
钥匙 2
（56bit 的解密钥匙）
DES 密文 2
DES 加密
钥匙 3
（56bit 的加密钥匙）
3-DES 密文 3
DES 密文 3

图 2.11　3-DES 加密


最初的加密和解密用同一把钥匙的话……
第 2 次加密时不管用什么样的钥匙，都和 DES 的结果是相同的。
这种情况下，实际的钥匙长度还是 56bit。
最初加密和第 2 次加密使用同一把钥匙……
解密时使用不同的钥匙的话，钥匙长度就变成了 56×2=112（bit）。
112bit
这种 3-DES 的方式叫做 EDE（EncryptDecrypt Encrypt）模式。


168 bit!
全部使用不同的钥匙，钥匙长度就变成 56×3=168（bit）了！


3-DES 只是钥匙长度变长而已吗？
好像是那样。
还没有解决根本问题。


当时，美国的 NIST（国家标准技术研究院）公开征集了具有新一代标准的密码，叫做 AES 密码。
IBM 和 NTT 等世界级的企业和组织参加了征集。
AES
密码？


最终被 AES 采用的就是，
Rijndael
Rijndael 加密算法！
现在仍然被美国政府应用于机密文件的加密工作。

AES 密码的概要

2000 年，Rijndael 作为 ASE 被 FIPS（Federal Information Processing Standard，联邦情报处理标准）采用。Rijndael 是以开发者的名字命名的。开发者为比利时的计算机科学家 Vincent Rijmen 和 Joan Daemen。

AES 根据钥匙长度，可分为 3 种类型（表 2.4）。

表 2.4 AES 密码的种类

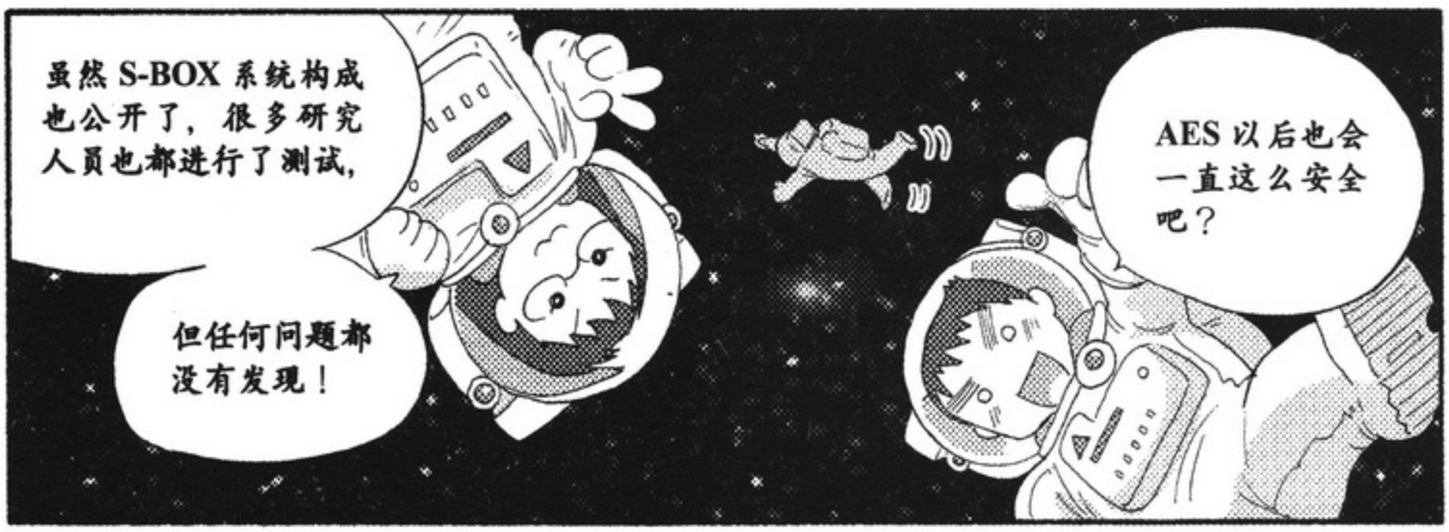
类　型	钥匙长度 / bit	分组长度 / bit	段　数
AES-128	128	128	10
AES-192	192	128	12
AES-256	256	128	14

密码的强度随着钥匙的长度增加和段数的增多而加大。

密码的结构，并非是 Feistel 类型，而是被称为 SPN（Substitution Permutation Network）类型，是将输入组和各段的钥匙进行 XOR 运算，通过同时进行换字和置换处理，并将各段累加。

今后的加密方式也许会逐渐从 DES 推移到安全性能更高的 AES 方式吧。


1年左右吧？
据说需要149兆年！
结婚！

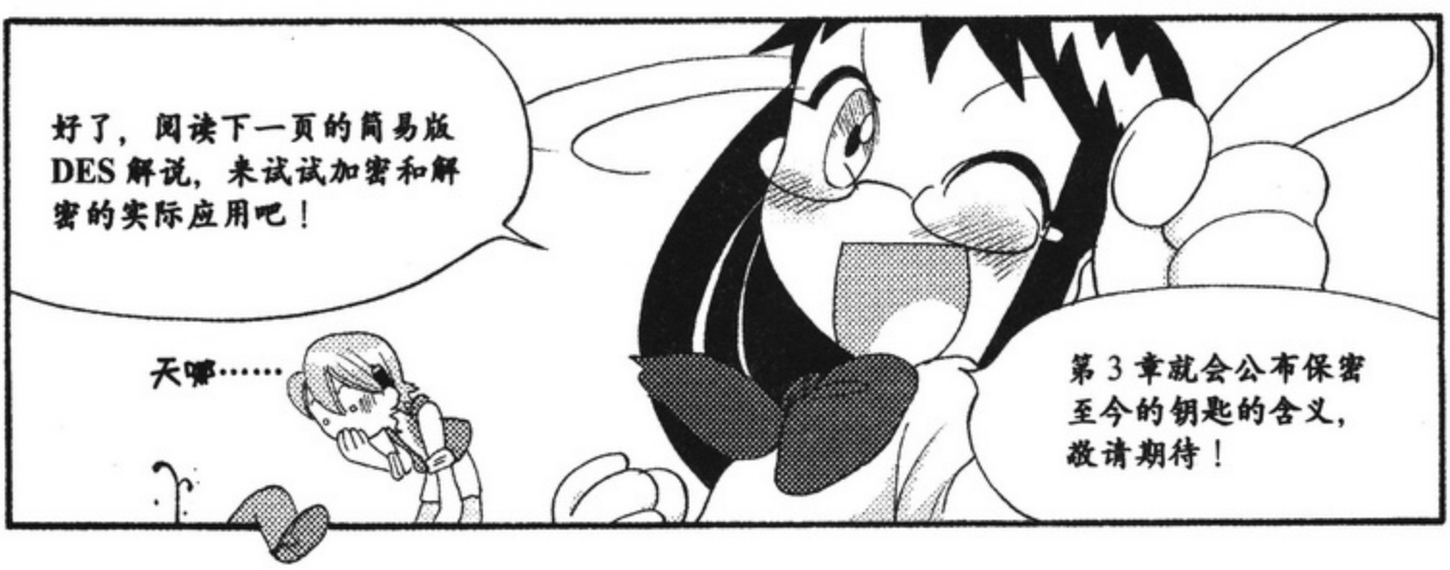

虽然S-BOX系统构成也公开了，很多研究人员也都进行了测试，
但任何问题都没有发现！
AES以后也会一直这么安全吧？


破解方面也在不断发展哦！
也有人说AES的安全强度只会维持10年左右。


什么嘛……
那么，哥哥！
教你这么多的密码知识，是不是该给我买电脑了！
啊？
就知道你会这么说，我早就准备好啦！
真的吗？
哈哈哈哈哈
看！
on
÷
×
−
+
=
.
0
1
2
3
4
5
6
7
8
9
MR
M−
M+
off
%
√
C
AC


这个电子计算器要比算盘方便多了哦……
不错吧!
啊
啊
啊


我要的不是计算器！
是电脑！
哥哥是个大白痴——


好了，阅读下一页的简易版DES解说，来试试加密和解密的实际应用吧！
天哪……
第3章就会公布保密至今的钥匙的含义，敬请期待！

简易版 DES 加密和解密详解

在 DES 密码里，是如何进行加密和解密的呢？这里采用 DES 的简易版来进行说明。

二进制数据的变换

由于不仅仅是 DES 密码，在其他的现代密码中也应用了二进制数据，所以无论是文章还是数字，都需要将明文变换为二进制数据。如表 2.7 所示，这里仅将使用的 16 字符 (其中含有一个没有意义的空字符)，将每个字符都对应不同的 4bit 的二进制编码进行变换，将明文表示成“0”和“1”的系列。

表 2.7 字符和二进制编码

文字	二进制编码	文字	二进制编码
A	0000	I	1000
B	0001	J	1001
C	0010	K	1010
D	0011	L	1011
E	0100	M	1100
F	0101	N	1101
G	0110	O	1110
H	0111	(空字符)	1111

DES 密文的生成

在 DES 密码中，虽然是将 64bit 分为 1 组的，但为了不失普遍性，这里还是进行一下通俗易懂的说明。这里将以 8bit 分为 1 组，对 2 段形式的简易 DES 密码进行分析。DES 密码生成是基于加密过程和密码钥匙生成的 2 个处理过程来进行的 (图 2.12)。

首先，如图 2.12 所示，将需要加密的明文按表 2.7 变换为“0”和“1”的排列。8bit 的二进制数据，先经过初期置换 IP 后被随机存取。随机存取的方式，如表 2.8 所示。

表 2.8 是对已经按 8bit 分组输入的明文，比如，输入的第 1 个 bit 进行过初期置换后，被置换为输出的第 5 个 bit。之后，按从左到右的顺序，输入的第 2 个 bit 就是对应输出的第 1 个 bit……依此类推进行置换。

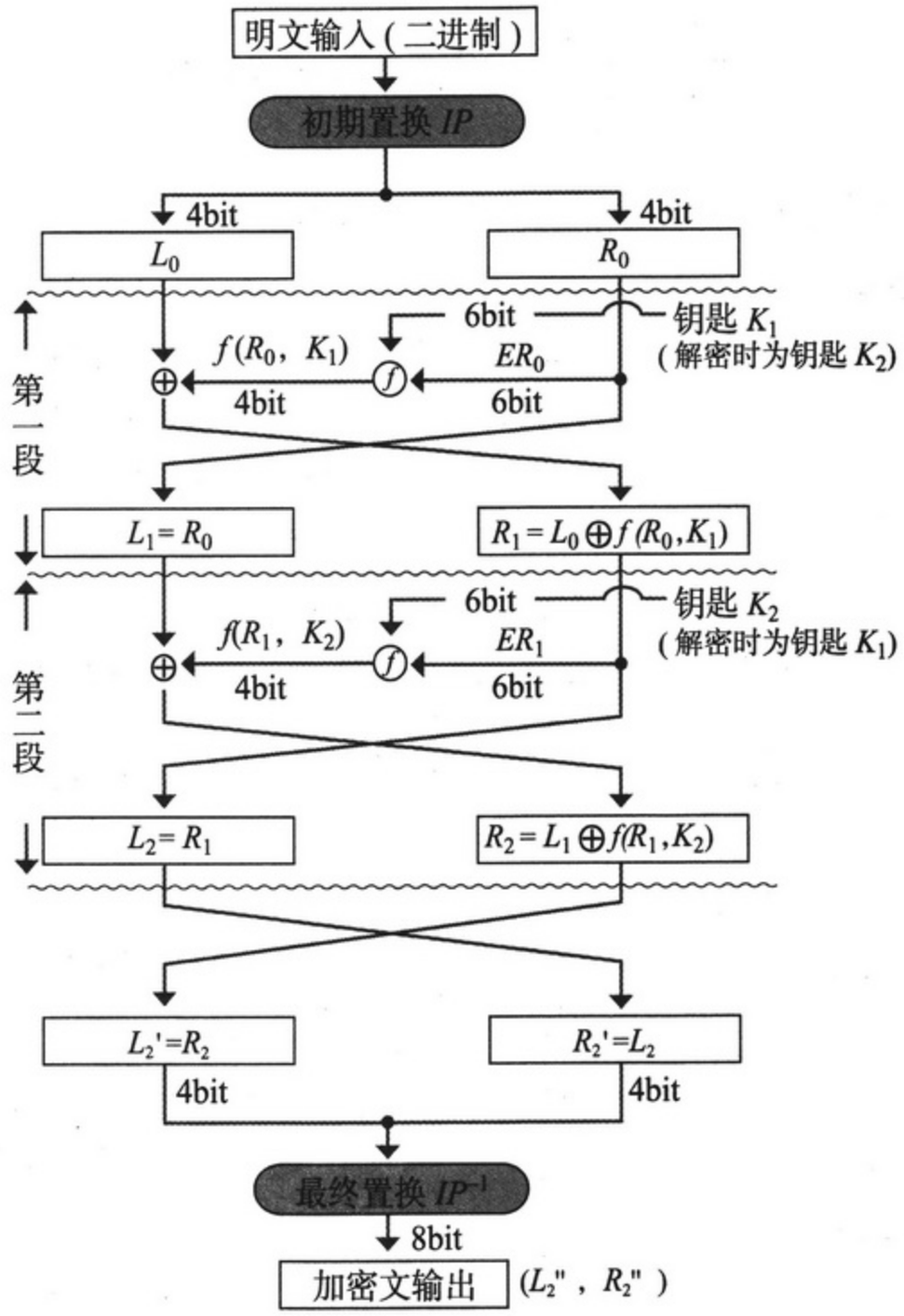


图 2.12 简易版 DES 加密文的生成顺序

表 2.8 初期置换 *IP*

输入 bit 位置 j	1	2	3	4	5	6	7	8
输出 bit 位置 k	5	1	6	2	7	3	8	4

表 2.9 初期置换(表 2.8 的另外一种表现)

输出 bit 位置 j	1	2	3	4	5	6	7	8
输入 bit 位置 k	2	4	6	8	1	3	5	7

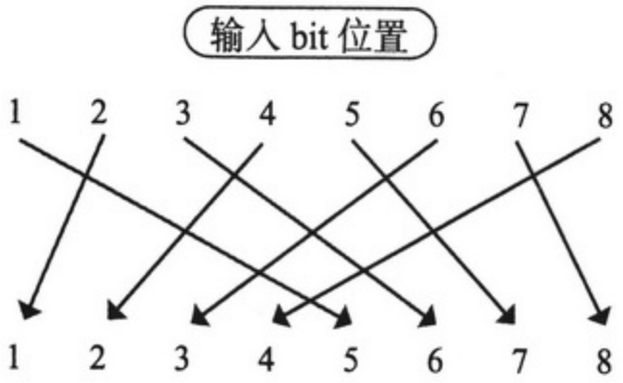


图 2.13　初期置换 *IP*

此外，将表 2.8 输出的 bit 按顺序排列，得出的结果如表 2.9 所示内容。在表 2.9 中，经过初期置换输出的第 1bit 与输入的第 2bit 对应，输出的第 2bit 与输入的第 4bit 对应……依此类推 (图 2.14)。

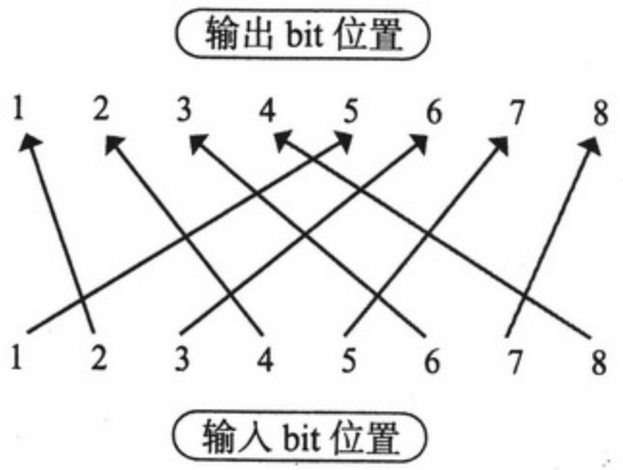


图 2.14　初期置换 *IP* 的另外一种表现

被初期置换的 bit 排列 (二进制数据)，经过如图 2.12 中 2 段的密码生成处理后，再经过如表 2.10 的最后置换 IP^{-1}，返回到输入的 bit 原来的位置。

表 2.10　最终置换 IP^{-1}

输入 bit 位置 k	1	2	3	4	5	6	7	8
输出 bit 位置 j	2	4	6	8	1	3	5	7

即，如果将表 2.8 和表 2.10 连接起来看的话，例如，在表 2.8 中输入的第 5 个 bit 将被输出为第 7 个 bit。再将该第 7 个 bit 经过表 2.10 变换为第 5 个 bit，可以得出回到了原来的第 5 个 bit 的位置 (图 2.15)。

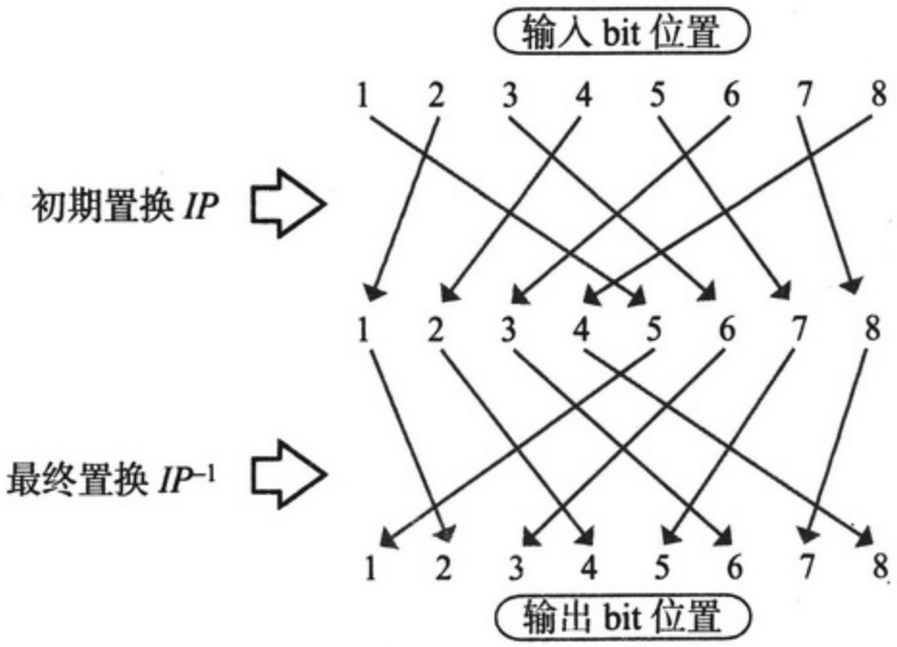


图 2.15　最初置换 *IP* 与最终置换 IP^{-1} 的组合

在这里，把将在今后具体说明过程中所必要的 DES 密码的 2 个钥匙设定为

$K_1 = (110001)$，$K_2 = (111000)$ ……(1)

(有关钥匙的生成，将在后面说明)。现在，把 1 个字符表示为 4bit 后，试着将 MC 字符列加密成简略版的 DES 加密文。根据表 2.7，将 MC 变换为二进制数据，MC 表示为 11000010。

下面，将 DES 密码生成的流程，用具体实例进行说明，希望大家也能认真地逐个计算，从而加深理解。

步骤 1

进行初期转换，依据表 2.8 的初期转换表，将明文 (11000010) = “MC” 转换为输出数据。(图 2.16)。

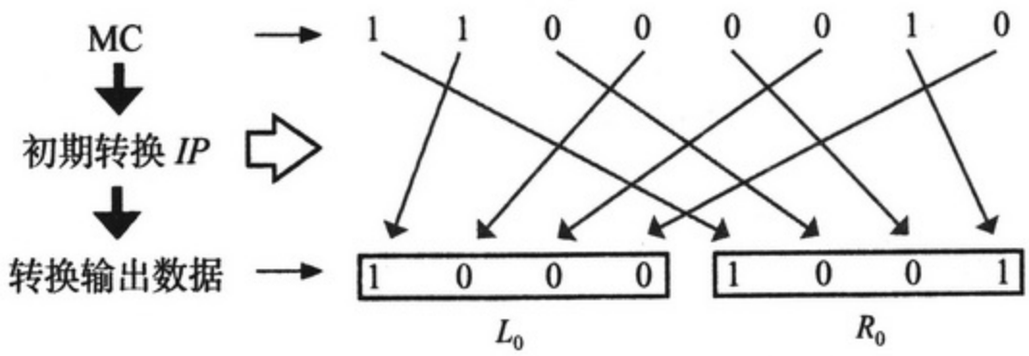


图 2.16　依据明文的初期转换 *IP* 输出数据

步骤 2

将步骤 1 中得出的转换输出数据，分割为上位 4bit（左侧）L_0 和下位 4bit（右侧）R_0。如图 2.16，得出以下等式。

$$L_0 = (1000) \cdots\cdots (2)$$

$$R_0 = (1001) \cdots\cdots (3)$$

步骤 3

依据表 2.11 的扩大置换 E（Expansion Permutation），将等式 (3) 的下划线部分的第 3 个 bit 和第 4 个 bit 复制，对 R_0 进行扩大置换（把 4bit 的 bit 数增加为 6bit，改变 bit 的位置）。

$$ER_0 = (011001) \cdots\cdots (4)$$

表 2.11 扩大置换 E

输出 bit 位置 k	1	2	3	4	5	6
输入 bit 位置 j	3	4	1	2	3	4

步骤 4

将等式 (4) 的扩大置换 ER_0 和钥匙 $K_1 = (110001)$ 进行异或运算（图 2.17）。

$$ER_0\,(K_1) = ER_0 \oplus K_1 \cdots\cdots (5)$$

$$= (011001) \oplus (110001)$$

$$= (101000) \cdots\cdots (6)$$

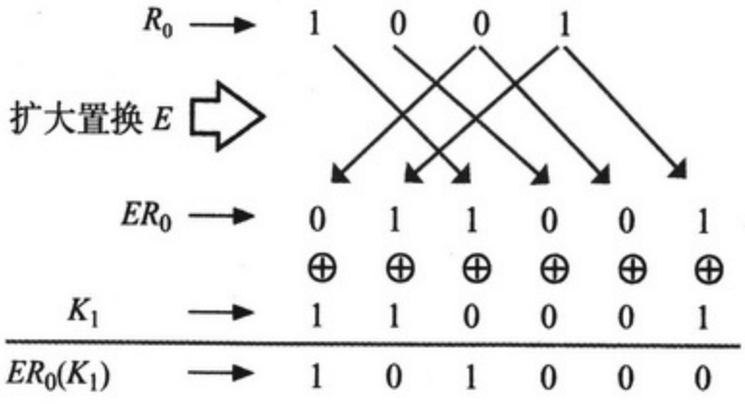


图 2.17 步骤 4 的计算流程

步骤 5

基于表 2.12 的压缩换字变换 S (Substitution)，将等式 (6) 进行压缩换字变换。(将 6bit 减少为 4bit，选择换字)

表 2.12 压缩换字变换 S

行序号 \ 列序号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	④	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

在表 2.12 中，准备了按行序号 0，1，2，3 表示的 4 种换字表。将等式 (6)6bit 之中最开始的 bit(最左侧的第 1 个 bit) 和最后的 bit(最右侧的第 6 个 bit) 所指定的数值，在换字表种类中选择行序号。并将剩余的 4bit 的指示数值，在列号码 (0~15) 中确定一个，进行换字选择。

例如，对于等式 (6) 的 (① 0100 ⓪)，首先选择第 (① ⓪)$_2$ = (2)$_{10}$ 行，其次再选择与第 (0100)$_2$ = (4)$_{10}$ 相列交叉的值 (13)$_{10}$(表 2.12 中□括上的位置)，进行二进制变换后得出 (1101)$_2$ 。再将所得结果 (1101)$_2$ 基于表 2.13 的输出置换 PS 的置换处理后，得出 (1101) → (0111)(图 2.18)。另外，( )$_2$ 及 ( )$_{10}$ 的下标字符则表示二进制和十进制。

表 2.13 压缩换字变换的输出置换 PS

输入 bit 位置 j	1	2	3	4
输出 bit 位置 k	3	4	1	2

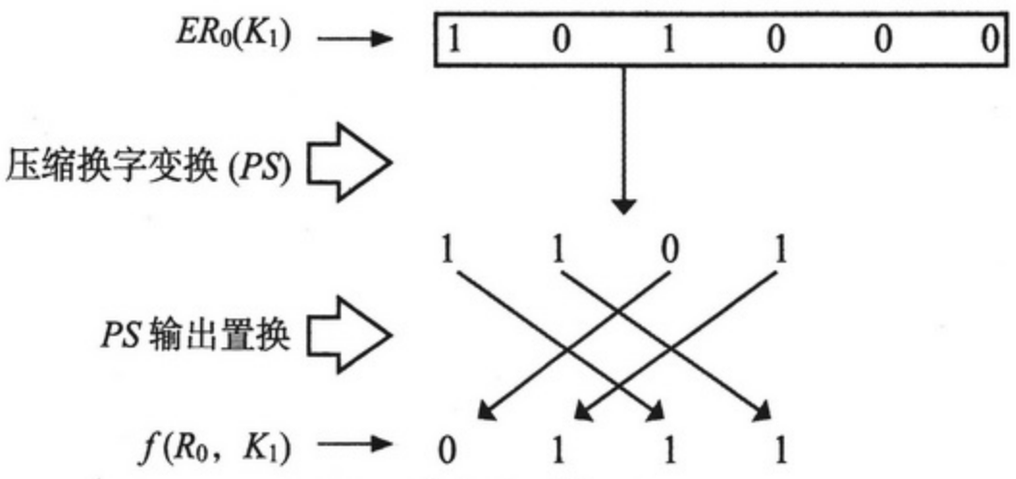


图 2.18 “步骤 5” 计算流程

上述这些处理运算称为压缩换字 · 置换变换。将该变换用非线性函数 f 表示为下列等式。

$$f(R_0,\ K_1)=(0111) \cdots\cdots (7)$$

这里的非线性函数是指，不满足 $f(ax+by)=af(x)+bf(y)$ 条件的函数。比如 $f(x)=2x$，这样通过原点的一次函数，满足这个条件则称为线性函数，但是比如 $f(x)=x^2$，这样的二次函数，因不满足条件而称为非线性函数。

步骤 6

如图 2.12 所示，将第 1 段的输出，上位 4bit(左侧)L_1 和下位 4bit(右侧)R_1，利用等式 (2)、等式 (3)、等式 (7)，得出下列等式。

$$L_1 = R_0 = (1001) \cdots\cdots (8)$$

$$R_1 = L_0 + f(R_0\ ,\ K_1) \cdots\cdots (9)$$

$$= (1000)\oplus(0111) = (1111) \cdots\cdots (10)$$

以下类同，经过从**步骤 3**到**步骤 6**的反复计算，可以生成 DES 密文。

步骤 7

基于表 2.11 的扩大置换 E，将 R_1 进行扩大置换。

$$ER_1 = (111111) \cdots\cdots (11)$$

步骤 8

将等式 (11) 的扩大置换 ER_1 和钥匙 =(111000) 进行不可兼析取计算。

$$ER_1(K_2) = ER_1 \oplus K_2 \cdots\cdots (12)$$

$$=(111111)+(111000)$$

$$=(000111) \cdots\cdots (13)$$

步骤 9

对于等式 (13)，首先选择第 $(⓪①)_2 = (1)_{10}$ 行，其次选择与第 $(\underline{0011})_2 = (3)_{10}$。

相列交叉的值 $(4)_{10}$（表 2.12 中○括上的位置）后，进行二进制变换得出 $(0100)_2$。再基于表 2.13 的输出置换 *PS* 的置换处理得出，

$$(0100) \rightarrow (0001) \cdots\cdots (14)$$

最后表示为

$$f(R_1, K_2) = (0001) \cdots\cdots (15)$$

步骤 10

如图 2.12 所示，将第 2 段的输出，上位 4bit（左侧）L_2 和下位 4bit（右侧）R_2，利用等式 (8)、等式 (10)、等式 (15) 进行运算，得出下列等式。

$$L_2 = R_1 = (1111) \cdots\cdots (16)$$

$$R_2 = L_1 \oplus f(R_2, K_2) \cdots\cdots (17)$$

$$= (1001) \oplus (0001) = (1000) \cdots\cdots (18)$$

步骤 11

如图 2.12，最终段里，将上位 bit L_2 和下位 bit R_2 进行调换（图 2.19）。

$$L_2' = R_2 = (1000) \cdots\cdots (19)$$

$$R_2' = L_2 = (1111) \cdots\cdots (20)$$

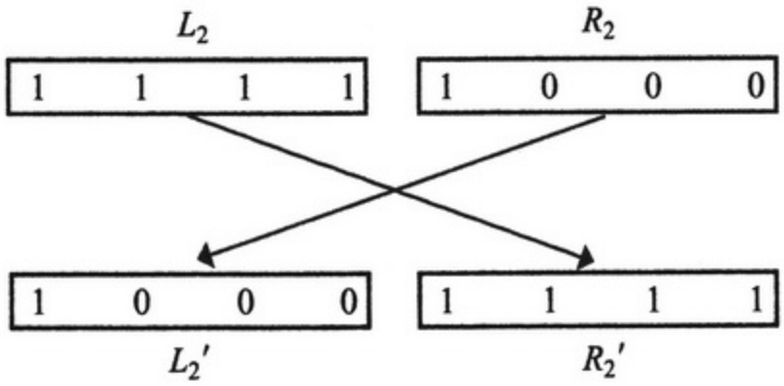


图 2.19　步骤 11 的计算流程

步骤 12

将图 2.18 的二进制数据，基于表 2.10 的最后置换 IP^{-1}，生成置换输出数据（图 2.20）。这样得出的 8bit 输出数据即为 DES 密文。

$L_2'' = (1110)$ ……(21)

$R_2'' = (1010)$ ……(22)

生成的密文 (11101010) ……(23)

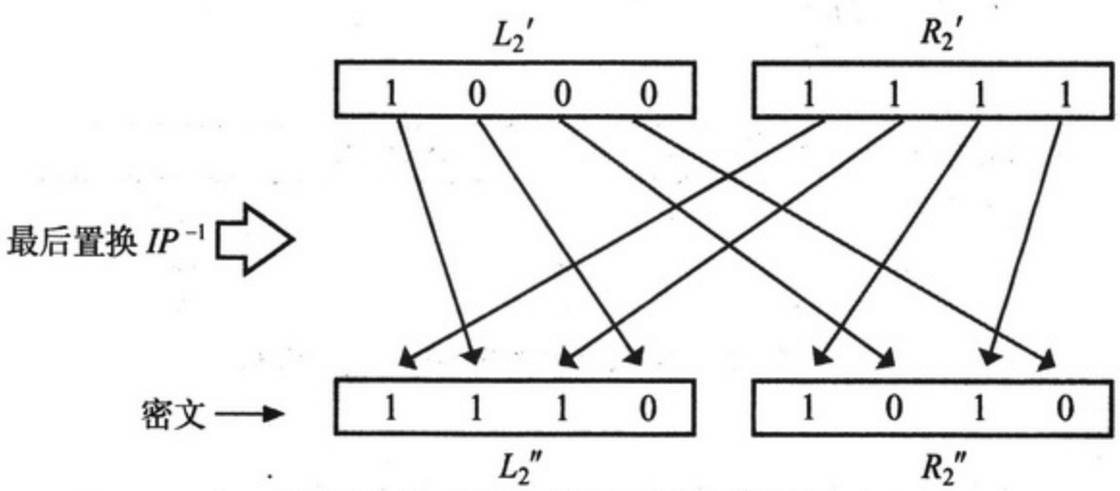


图 2.20 基于最后置换 IP^{-1} 输出的密文

✤ DES 密文的解密

这次，试着将 DES 密文还原到明文吧。

解密的过程，可以套用图 2.12 生成密文的过程。但是，在生成 DES 密文时，使用钥匙的顺序是 K_1，K_2，而在解密时需要将此顺序颠倒，在第 1 段里使用 K_2，第 2 段里使用 K_1。

首先，最开始先对等式 (23) 的密文，进行步骤 1 的初期置换。

步骤 1

初期置换就是，基于表 2.8 的初期置换表，将密文 (11101010) 做成置换输出数据 (图 2.21)。

步骤 2

将步骤 1 得出的置换输出数据，上位 4bit(左侧)L_0 和下位 4bit(右侧)R_0 进行分割。如图 2.21 所示，对照等式 (19) 和等式 (20)，生成以下等式。

$L_0 = (1000)(=L_2')$ ……(24)

$R_0 = (11\underline{11})(=R_2')$ ……(25)

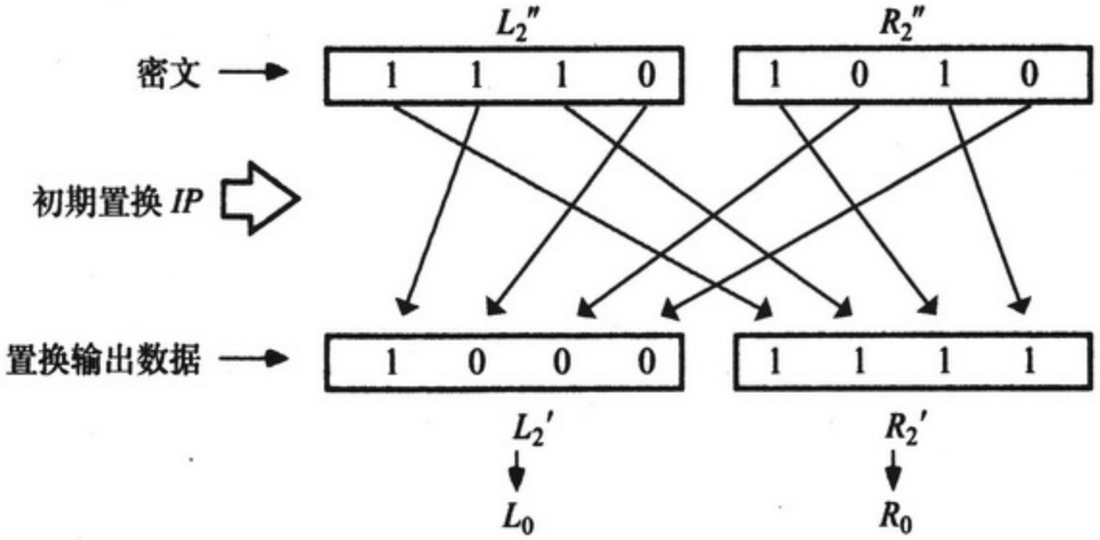


图 2.21　基于密文的初期置换 *IP* 输出数据

步骤 3

基于表 2.11 的扩大置换 E，将等式 (25) 的下划线部分的第 3bit 和第 4bit 复制，对 R_0 进行扩大置换 E。

$$ER_0 = (\underline{1}111\underline{1}1) \cdots\cdots (26)$$

步骤 4

将等式 (25) 的扩大置换 ER_0，和钥匙 $K_2 = (111000)$ 进行异或计算。

$$ER_0(K_2) = ER_0 \oplus K_2 \cdots\cdots (27)$$

$$= (111111) \oplus (111000)$$

$$= (000111) \cdots\cdots (28)$$

步骤 5

基于表 2.12 的压缩换字变换，将等式 (28) 进行压缩换字变换。对于等式 (28) 中的 $(⓪\underline{0011}①)_2$，首先选择第 $(⓪①)_2 = (1)_{10}$ 行，其次选择与第 $(0011)_2 = (3)_{10}$ 相列交叉的值 $(4)_{10}$（表 2.12 中○括上的位置）后，进行二进制变换得出 $(0100)_2$。再将所得结果 $(0100)_2$ 基于表 2.13 的输出置换 PS，得出 $(0100) \rightarrow (0001)$，最后表示为下列等式。

$$f(R_0, K_2) = (0001) \cdots\cdots (29)$$

步骤 6

如图 2.12 所示，将第 1 段的输出，上位 4bit(左侧)L_1 和下位 4bit(右侧)R_1，利用等式 (24)、等式 (25)、等式 (29)，得出下列等式。

$$L_1 = R_0 = (1111) \cdots\cdots (30)$$

$$R_1 = L_0 \oplus f(R_0,\ K_2) \cdots\cdots (31)$$

$$= (1000) \oplus (0001) = (10\underline{01}) \cdots\cdots (32)$$

以下类同，从步骤 3 到步骤 6，进行反复计算。

步骤 7

基于表 2.11 的扩大置换 E，将 R_1 进行扩大置换。

$$ER_1 = (\underline{01}10\underline{01}) \cdots\cdots (33)$$

步骤 8

将等式 (33) 的扩大置换 ER_1 和钥匙 $K_1 = (110001)$ 进行异或计算。

$$ER_1(K_1) = ER_1 \oplus K_1 \cdots\cdots (34)$$

$$= (011001) \oplus (110001)$$

$$= (101000) \cdots\cdots (35)$$

步骤 9

对于等式 (35) 的 $(①\underline{0100}⓪)_2$，首先选择第 $(①⓪)_2 = (2)_{10}$ 行，其次选择与第 $(\underline{0100})_2 = (4)_{10}$ 相列交叉的值 $(13)_{10}$(表 2.12 中□括上的位置) 后，进行二进制变换得出 $(1101)_2$。再基于表 2.13 的输出置换 PS 得出，

$$(1101) \rightarrow (0111) \cdots\cdots (36)$$

最后表示为

$$f(R_1,\ K_1) = (0111) \cdots\cdots (37)$$

步骤 10

如图 2.12 所示，将第 2 段的输出，上位 4bit（左侧）L_2 和下位 4bit（右侧）R_2，利用等式（30）、等式（31）、等式（37），得出下列等式。

$$L_2 = R_2 = (1001) \quad \cdots\cdots (38)$$

$$R_2 = L_1 \oplus f(R_1, K_1) \quad \cdots\cdots (39)$$

$$= (1111) \oplus (0111) = (1000) \quad \cdots\cdots (40)$$

步骤 11

如图 2.12，在最后段中，将上位 bit L_2 和下位 4bit R_2 进行调换（图 2.22）。

$$L_2' = R_2 = (1000) \quad \cdots\cdots (41)$$

$$R_2' = L_2 = (1001) \quad \cdots\cdots (42)$$

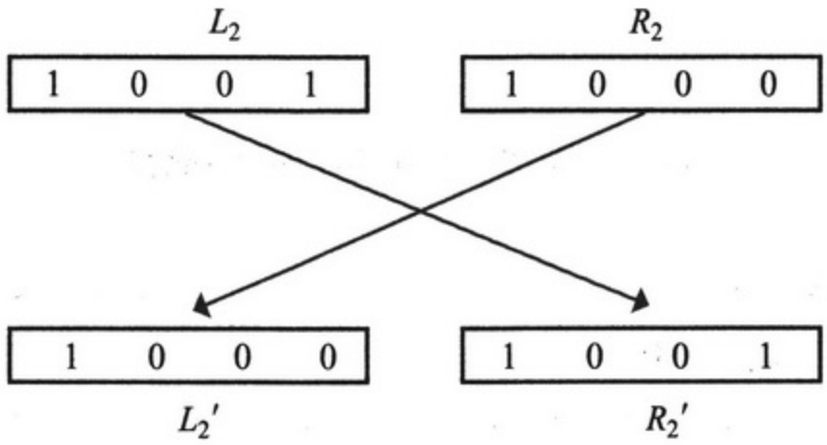


图 2.22　步骤 11 的计算流程

步骤 12

将图 2.22 的二进制数据，基于表 2.9 的最后置换 IP^{-1}，生成置换输出数据（图 2.23）。

$$L_2'' = (1100) \quad \cdots\cdots (43)$$

$$R_2'' = (0010) \quad \cdots\cdots (44)$$

得出明文 1100 0010

“M” “C”

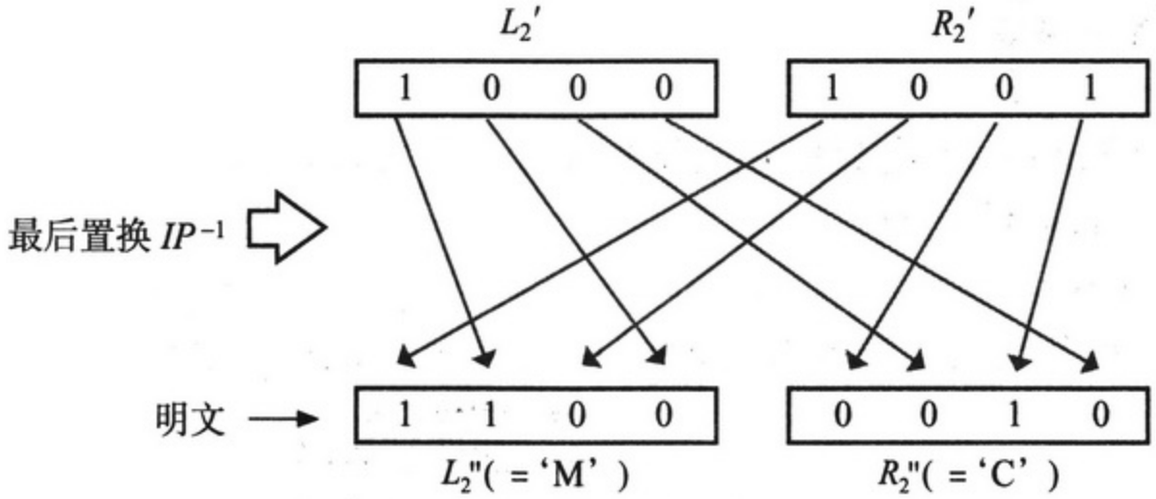


图 2.23　步骤 12 的计算流程

这样得出的 8bit 输出数据相当于明文，将等式（43）和等式（44）的二进制编码，分别根据表 2.7 得出“M”和“C”文字，因此，可以认为 DES 密文已被解密。

综上所述，将 DES 密码的加密处理和解密处理进行对比的话，可通过执行与加密处理完全相反的流程，确认解密处理的实行过程（图 2.24）。

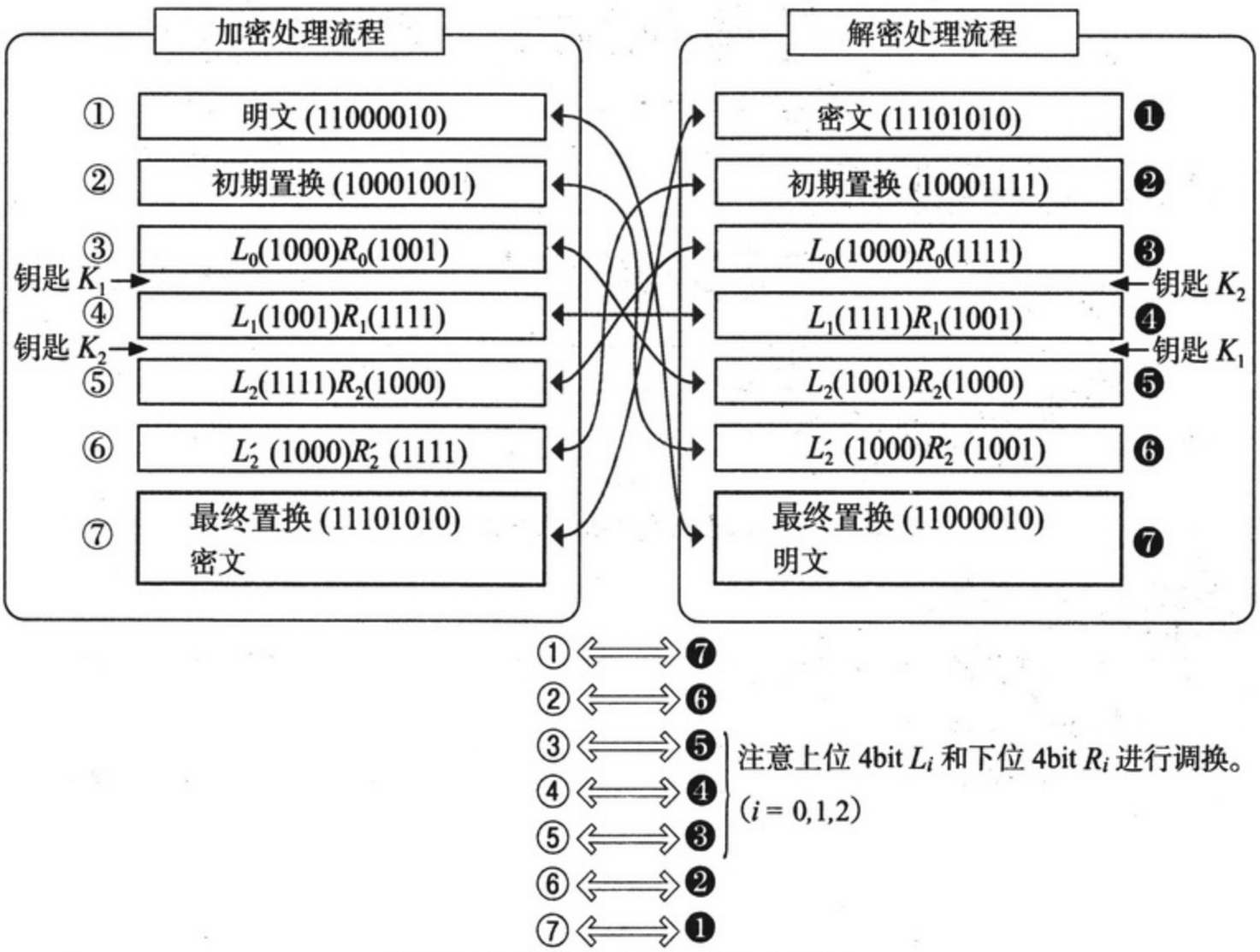


图 2.24　加密处理和解密处理的对应关系

✤ DES 加密钥匙的生成

接下来，对 DES 密码通用钥匙的加密钥匙及解密钥匙，生成过程进行说明。假设 8bit 的通用密钥（初始密码）K_0 为

K_0 =（10011001）……（45）

首先介绍图 2.12 中的第 1 段钥匙 K_1 和第 2 段钥匙 K_2 的生成过程（图 2.25）。

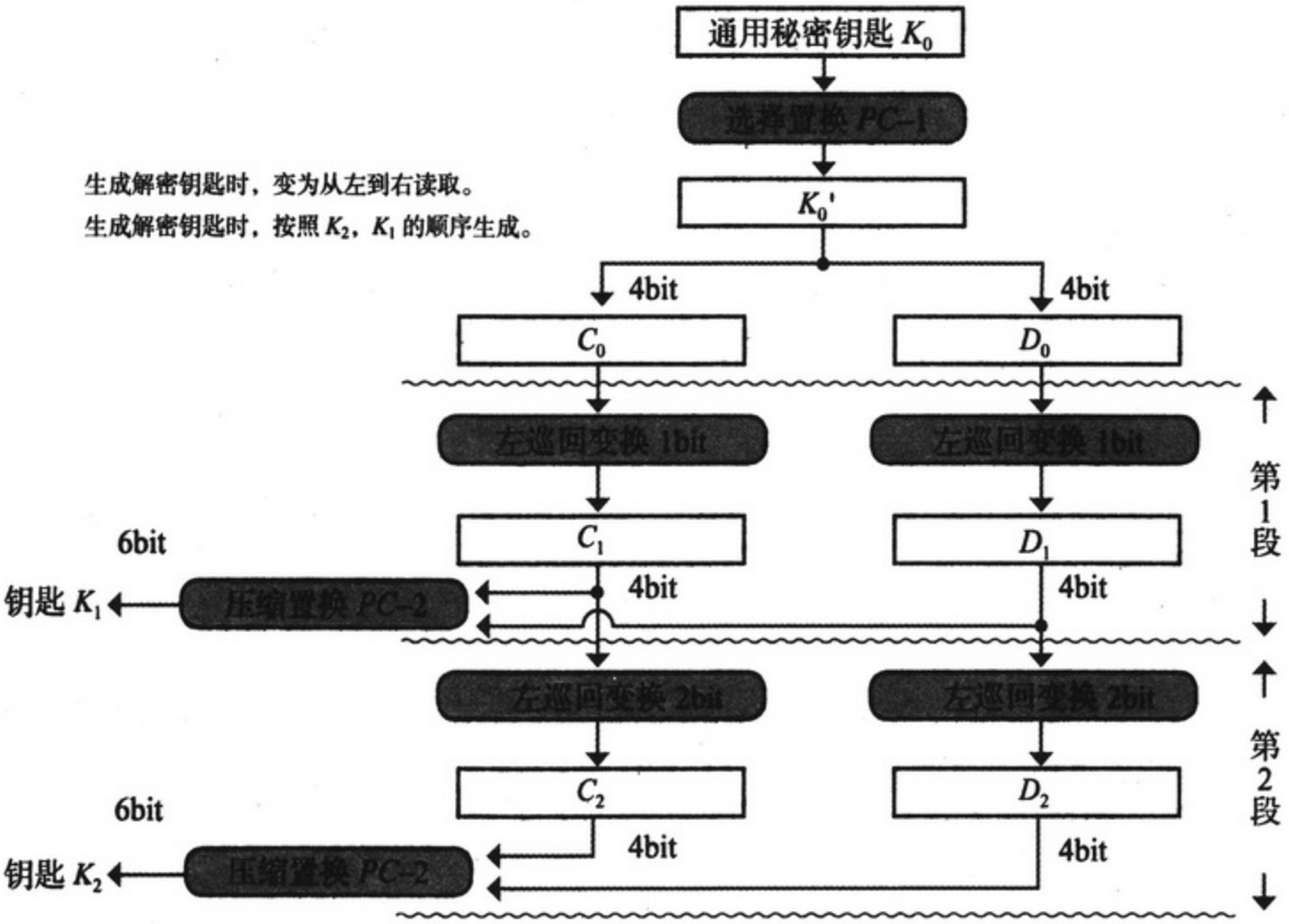


图 2.25　加密钥匙和解密钥匙的生成过程

步骤 1

将等式（45）的通用钥匙（秘密钥匙）K_0，基于表 2.14 的选择置换 PC–1，经随机存取后得出下列等式（图 2.26）。

K_0' =（00110101）……（46）

这里将等式（46）的钥匙 K_0'，分为上位 4bit C_0 和下位 4bit D_0，表示为以下等式。

C_0 =（0011）……（47）

D_0 =（0101）……（48）

表 2.14 选择置换 *PC*–1

	上位 4bit C_i				下位 4bit D_i			
输入 bit 位置 j	1	2	3	4	5	6	7	8
输出 bit 位置 k	8	7	1	3	6	2	5	5

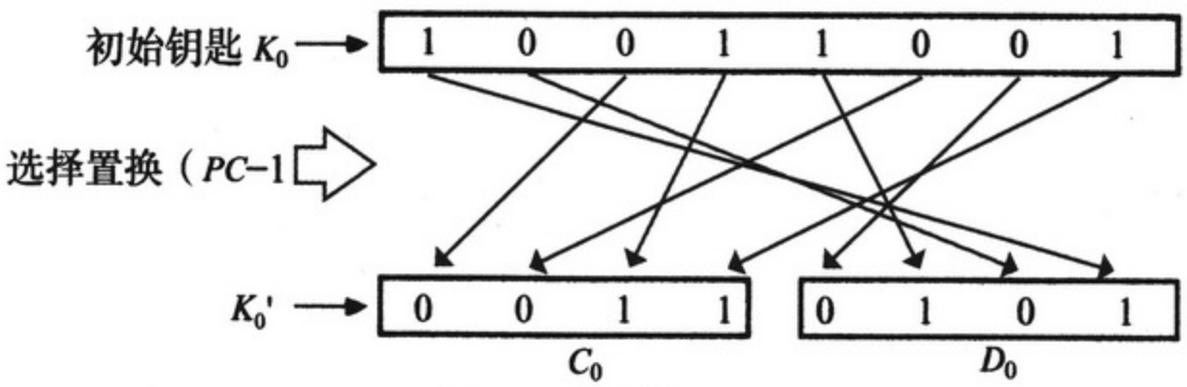


图 2.26 选择置换 *PC*–1

步骤 2

如表 2.15 左巡回变换的 bit 数，因为第 1 段的左巡回变换的 bit 数为 1bit，所以将 C_0 和 D_0 的各个 bit 进行 1bit 向左巡回变换，其结果表示用 C_1 和 D_1（图 2.27）。

C_1 =（0110）……（49）

D_1 =（1010）……（50）

表 2.15 左巡回变换的 bit 数

段　数	1	2
变换 bit 数	1	2

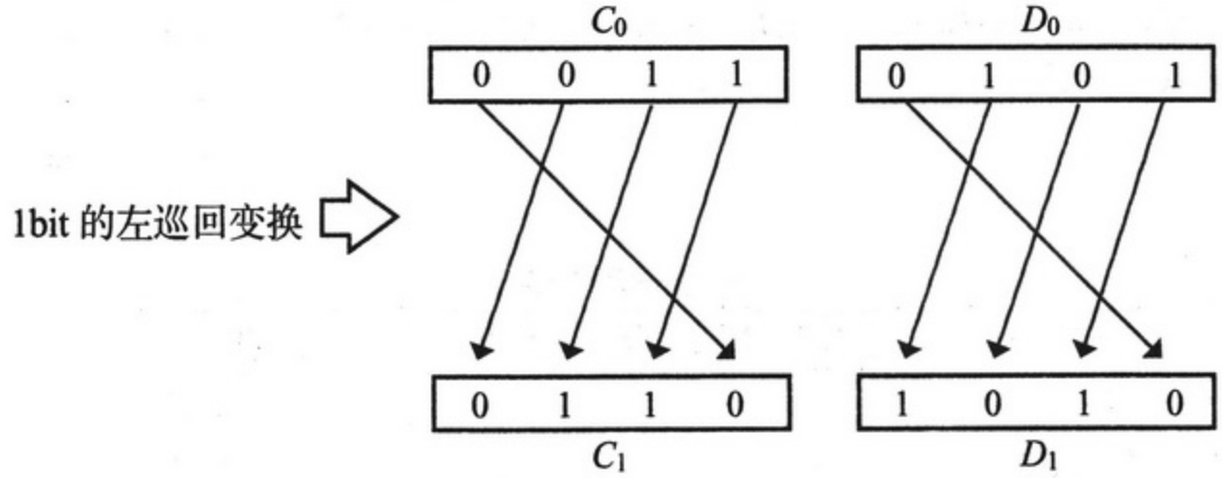


图 2.27　左巡回变换的处理（步骤 2）

步骤 3

基于表 2.16 的压缩置换 *PC*–2，将 C_1 和 D_1 的全部（等式（49）、等式（50）），从 8bit 压缩变换至 6bit，得出在第 1 段中使用的加密钥匙 K_1（图 2.28）。

K_1 =（110001）……（51）

表 2.16　压缩置换 *PC*–2

输出 bit 位置 k	1	2	3	4	5	6
输入 bit 位置 j	7	5	1	8	6	2

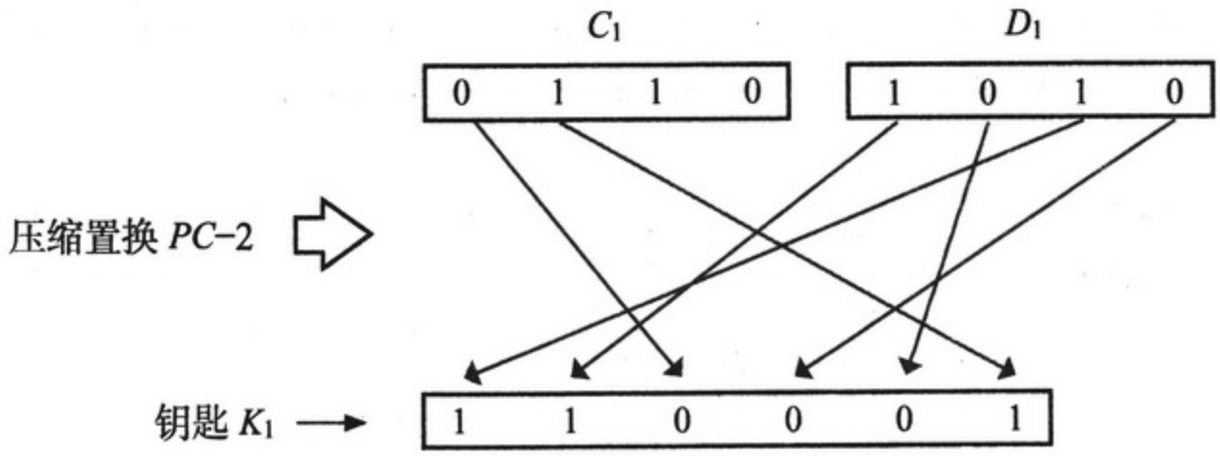


图 2.28　压缩置换 *PC*–2 的处理 “步骤 3”

以下类同，经过从步骤 2 到步骤 3 的反复计算，可以逐个得出加密中使用的钥匙。

步骤 4

如表 2.15 左巡回变换的 bit 数，因为第 2 段的左巡回变换的 bit 数为 2bit，所以将 C_1 和 D_1 的各个 bit 进行 2bit 向左巡回变换，其结果表示用 C_2 和 D_2（图 2.29）。

$C_2 =$（1001）……（52）

$D_2 =$（1010）……（53）

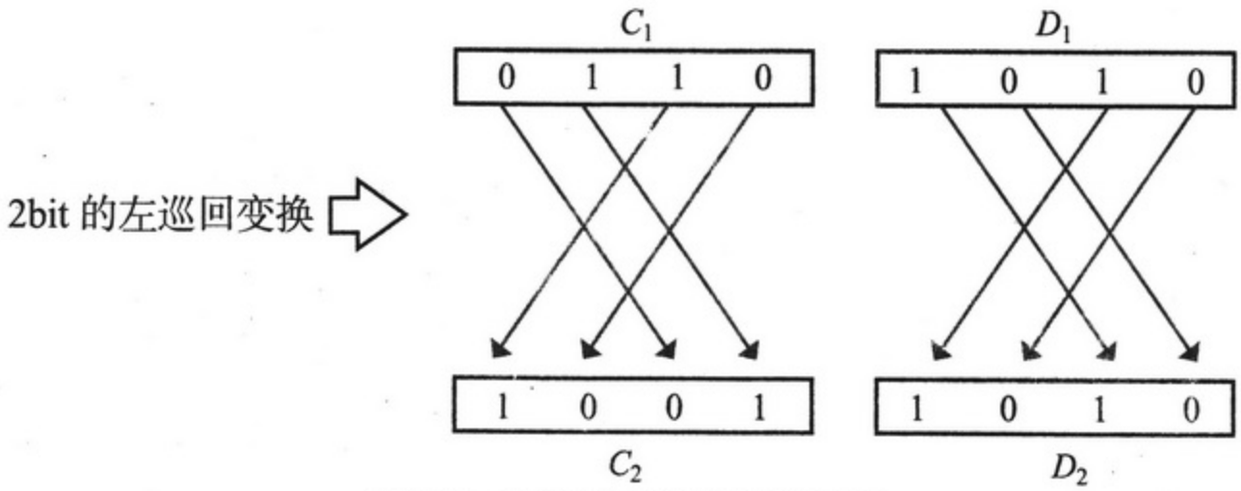


图 2.29　左巡回变换的处理（步骤 4）

步骤 5

基于表 2.16 的压缩置换 *PC*–2，将 C_2 和 D_2 的全部（等式（52）、等式（53）），从 8bit 压缩变换至 6bit，得出在第 2 段中使用的钥匙 K_2（图 2.30）。

$K_2 =$（111000）……（54）

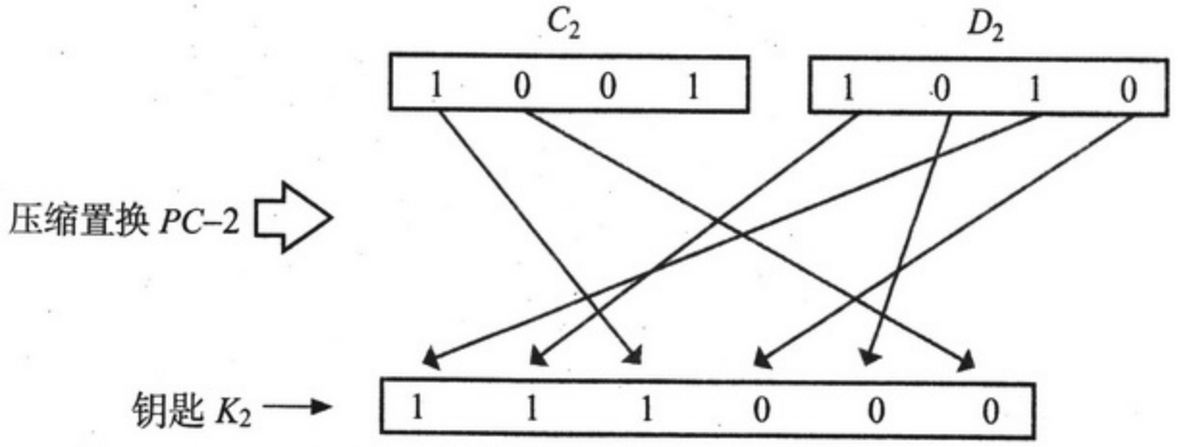


图 2.30　压缩置换 *PC*–2 的处理 “步骤 5”

❀ DES 解密钥匙的生成

加密钥匙是，在等式（45）的通用钥匙（初始钥匙）K_0=（10011001）的基础上，按 K_1，K_2 的顺序生成的。相反，将密文还原成明文的解密钥匙则是，在通用钥匙 K_0 的基础上，按 K_2，K_1 的顺序生成。此时，如果与图 2.25 的处理过程相同的话，生成加密钥匙采用向左巡回变换的处理方式，生成解密钥匙则变为向右巡回变换的处理方式。下面，在图 2.12 基础上，归纳生成解密钥匙的过程。

步骤 1

将等式（45）的通用钥匙（秘密钥匙）K_0，基于表 2.14 的选择置换 PC–1 的基础上，进行随机存取。

$$K_0' = (00110101) \quad \cdots\cdots (55)$$

$$C_0 = (0011) \quad \cdots\cdots (56)$$

$$D_0 = (0101) \quad \cdots\cdots (57)$$

步骤 2

如表 2.17 右巡回变换的 bit 数，因为第 1 段的左巡回变换的 bit 数为 1bit，所以将 C_0 和 D_0 的各个 bit 进行 1bit 向右巡回变换，其结果用 C_1 和 D_1 表示（图 2.31）。

$$C_1 = (1001) \quad \cdots\cdots (58)$$

$$D_1 = (1010) \quad \cdots\cdots (59)$$

表 2.17 右巡回变换的 bit 数

段数	1	2
变换 bit 数	1	2

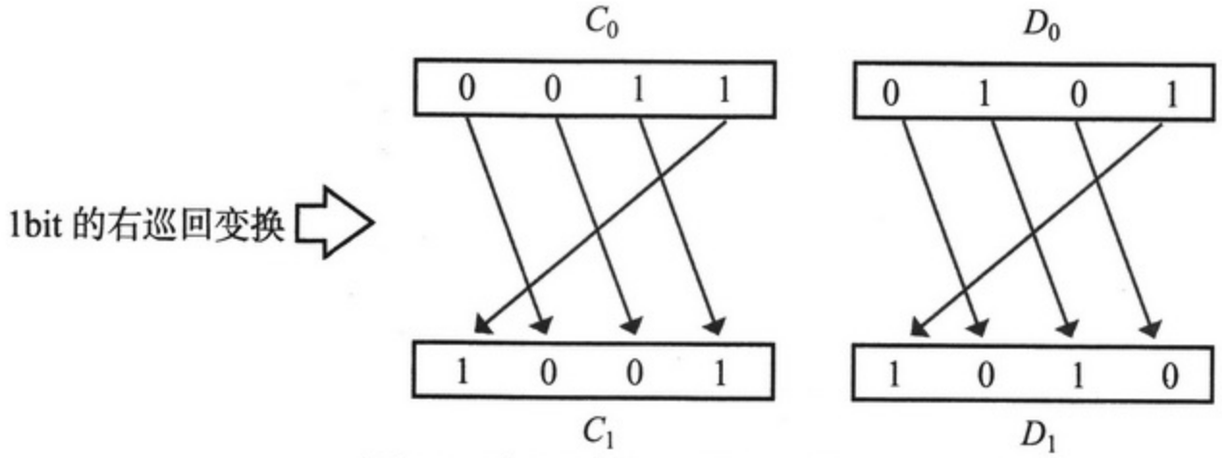


图 2.31　右巡回变换的处理（步骤 2）

步骤 3

基于表 2.16 的压缩置换 *PC*–2，将 C_1 和 D_1 的全部（等式（58）、等式（59）），从 8bit 压缩变换至 6bit，得出在第 1 段中使用的解密钥匙 K_2（图 2.32）。

$$K_1 = (111000) \quad \cdots\cdots (60)$$

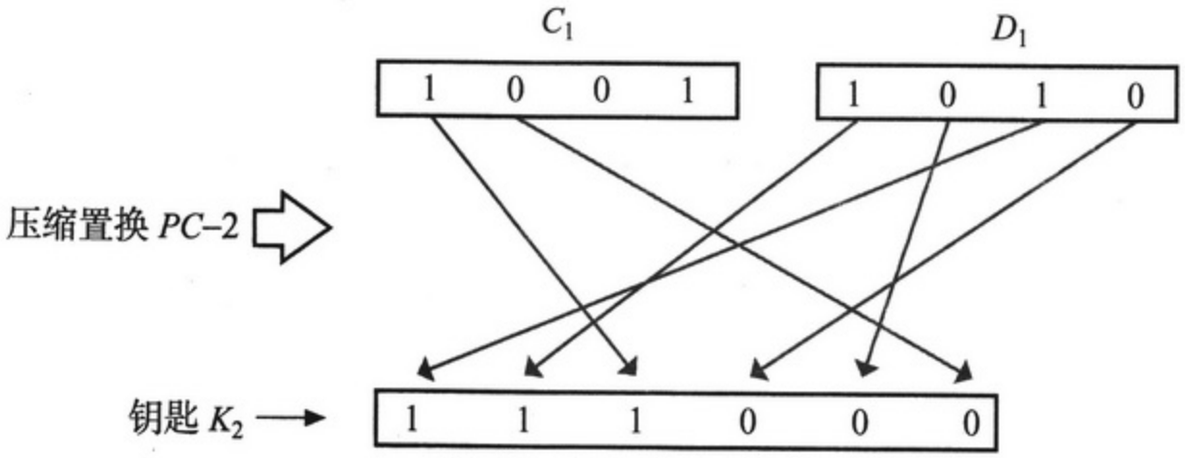


图 2.32　压缩置换 *PC*–2 的处理 “步骤 3”

步骤 4

如表 2.17 右巡回变换的 bit 数，因为第 2 段的 bit 数为 2bit，所以将 C_1 和 D_1 的各个 bit 进行 2bit 向右巡回变换，其结果用 C_2 和 D_2 表示（图 2.33）。

$$C_2 = (0110) \quad \cdots\cdots (61)$$

$$D_2 = (1010) \quad \cdots\cdots (62)$$

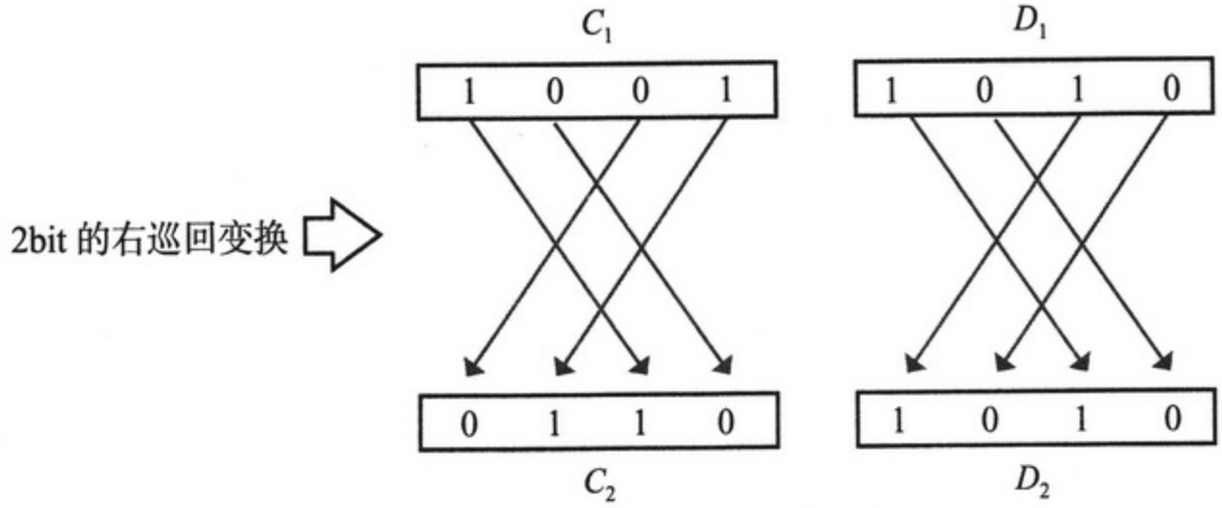


图 2.33　右巡回变换的处理（步骤 4）

步骤 5

基于表 2.16 的压缩置换 *PC*–2，将 C_2 和 D_2 的全部（等式（61），等式（62）），从 8bit 压缩变换至 6bit，得出在第 2 段中使用的钥匙 K_1（图 2.34）。

$$K_1 = (110001) \quad \cdots\cdots (63)$$

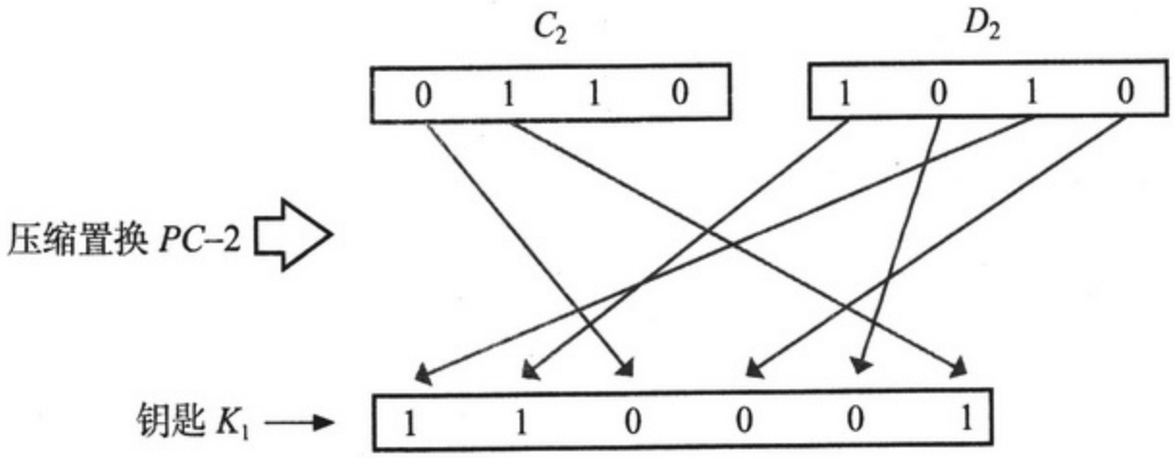
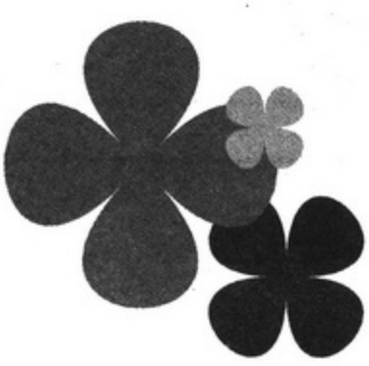


图 2.34　压缩置换 PC–2 的处理 “步骤 5”

通过上述结果，将解密时使用的钥匙（等式（60）、等式（63））与加密时使用的钥匙（等式（51）、等式（54））进行对比，可以得出加密时使用的钥匙顺序（K_1，K_2），解密时使用与之相反的（K_2，K_1）顺序。

◆第3章◆

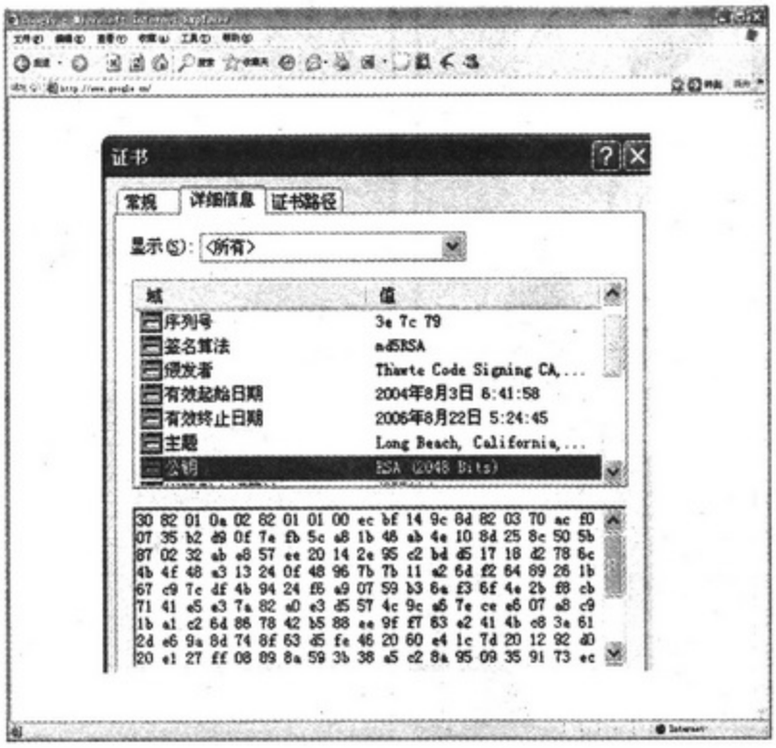
公开钥匙加密技术



3-1 公开钥匙密码的基础知识

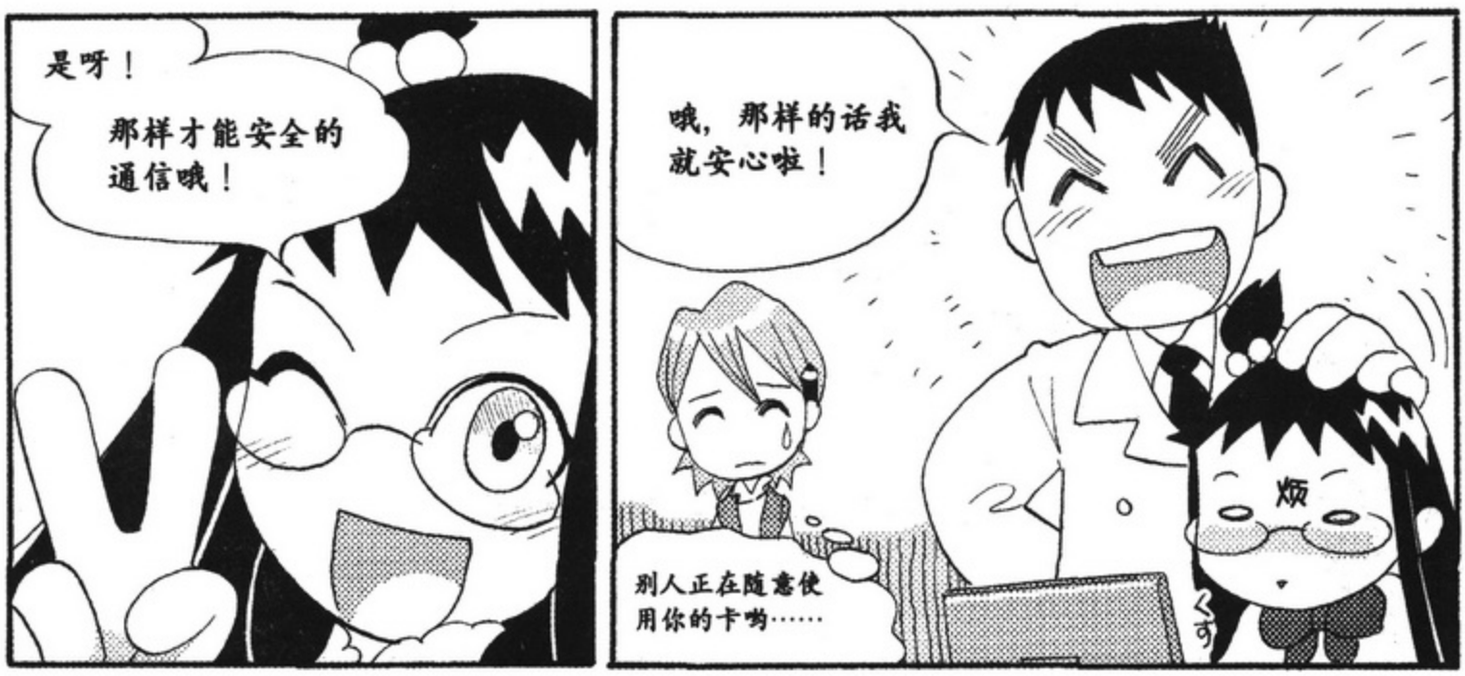

有了电脑就可以做很多事情哦……
呵呵
呵呵
电子邮件、网上购物、玩游戏……
喂喂，不是用来学习的吗？
呵呵！
我也经常在网上购物哦……
用信用卡结算很方便。
BOOK
CD
SHOES
不会有危险吗？信用卡的信息被盗的话……
为了防止被窃密，就要使用公开钥匙加密技术哦！


购物天堂网站
高级名牌皮包
￥300,000
购买
详细
这个很漂亮吧？
啊……
还行吧。
不是很清楚。
那，信用卡借
我用一下。
你想干什么？
输入卡号和
密码。
咔
咔
CARD
好了，点击！
点击
￥300,0
购买
详细
咔嚓
你，你在干什么？
看看这个！



公开钥匙是密码在通信进行状态下表示出来的数据。浏览器是 Explorer 的时候，菜单是按照文件—属性—证书—详细的顺序排列的。

浏览器是 Safari 或 Firefox 的时候，点击锁定图标，将详细信息设定权限。



……所以呢，虽然可能没注意到，

但是，实际上大家都在网上使用着公开钥匙，让我们一起来学习吧！

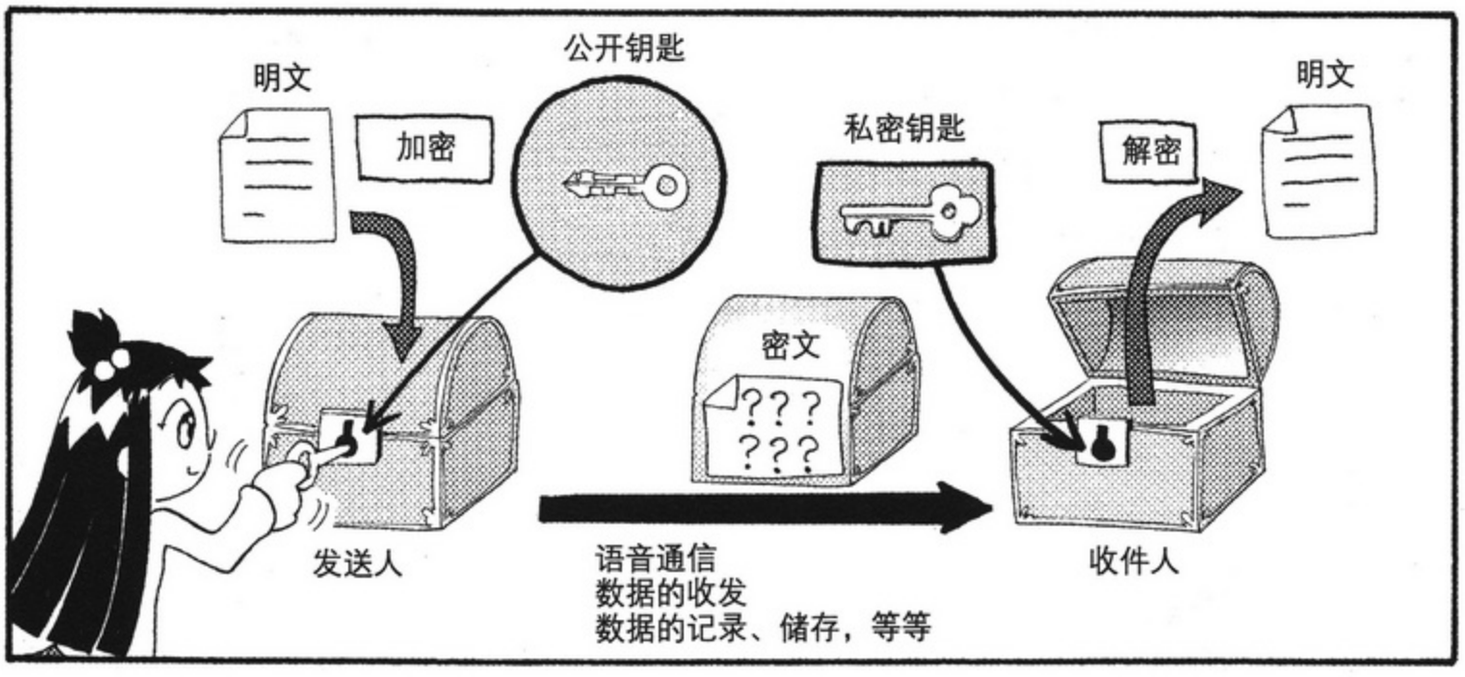
到现在为止的密码，所有钥匙都是绝对不会让别人知道的啊！

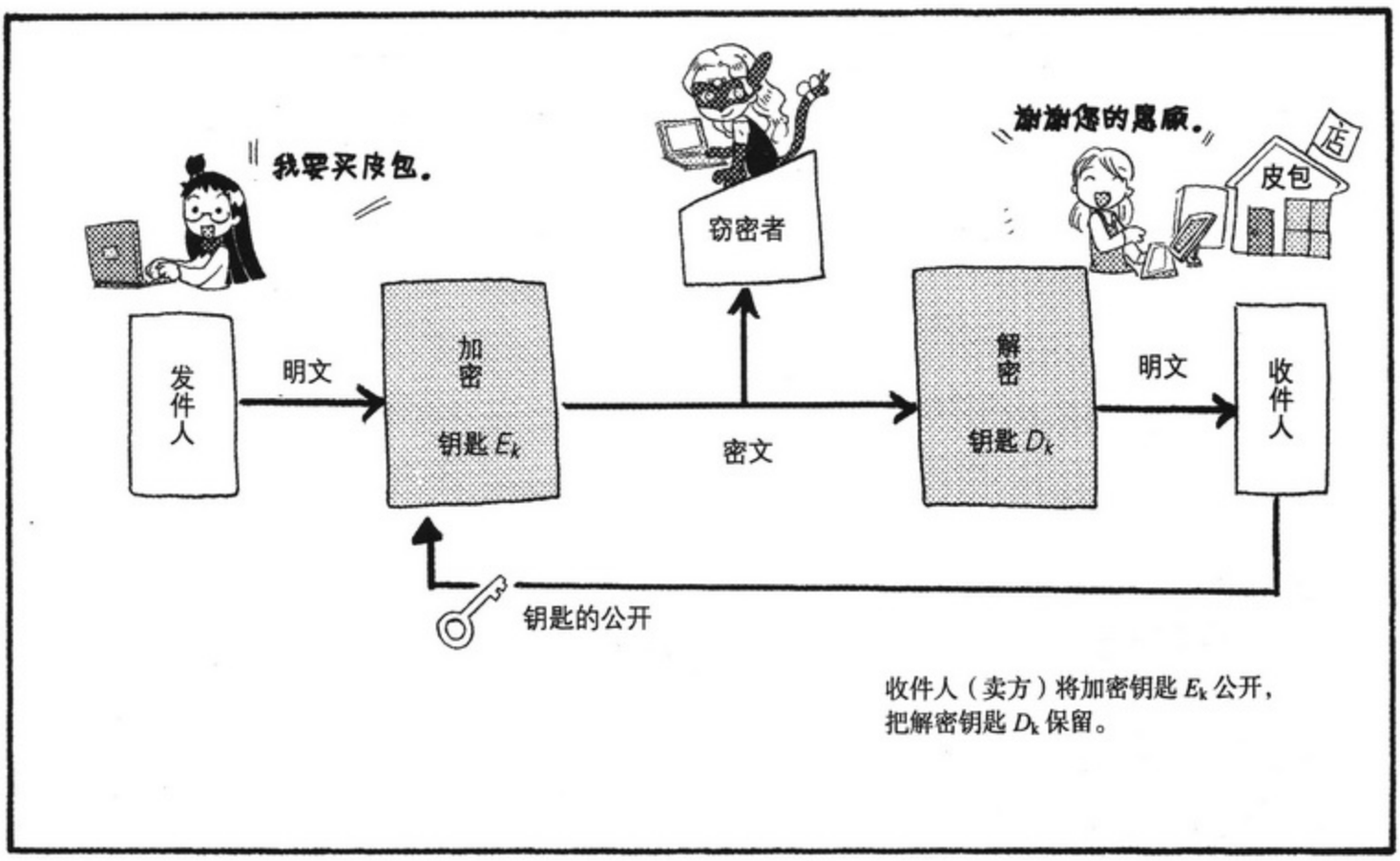
是啊，是啊！

为什么钥匙被别人知道了，也没关系呢？

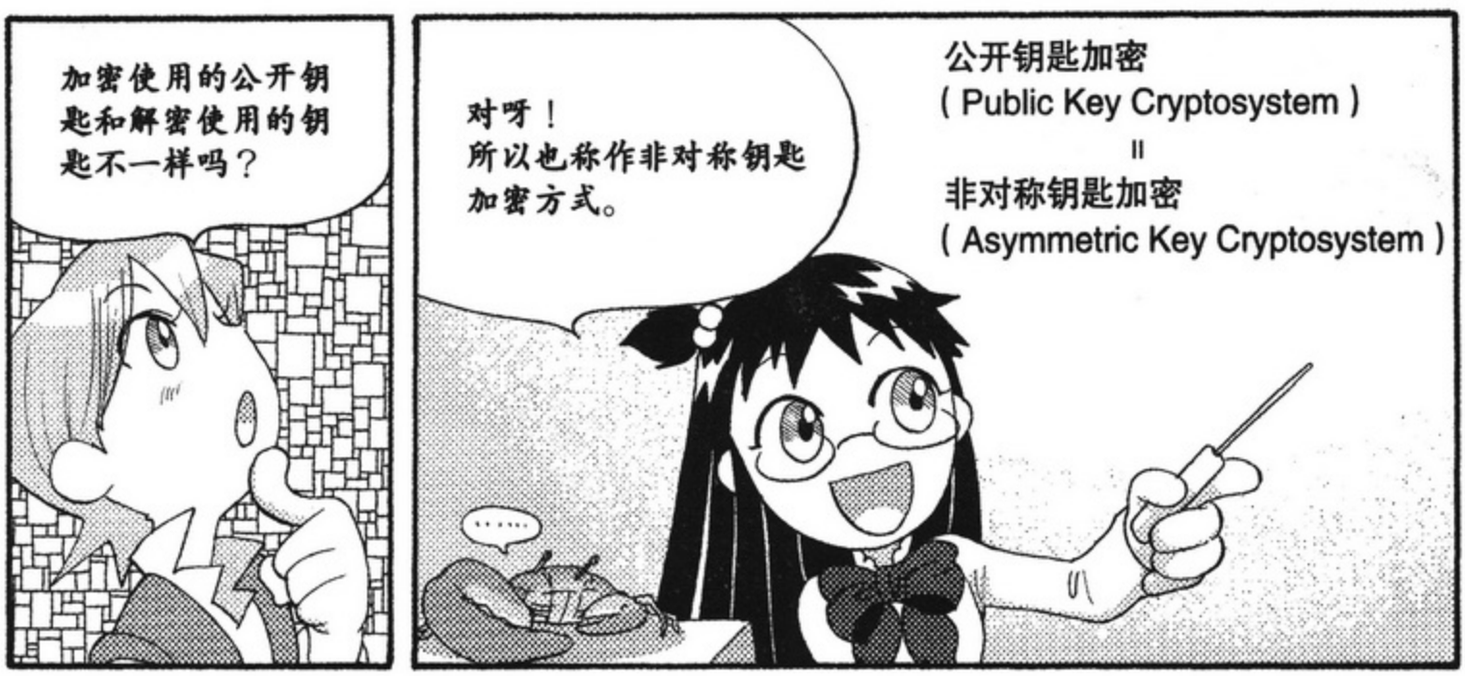
其实钥匙不止公开钥匙一个，

还有一个叫做私密钥匙哦！





收件人（卖方）将加密钥匙 E_k 公开，把解密钥匙 D_k 保留。

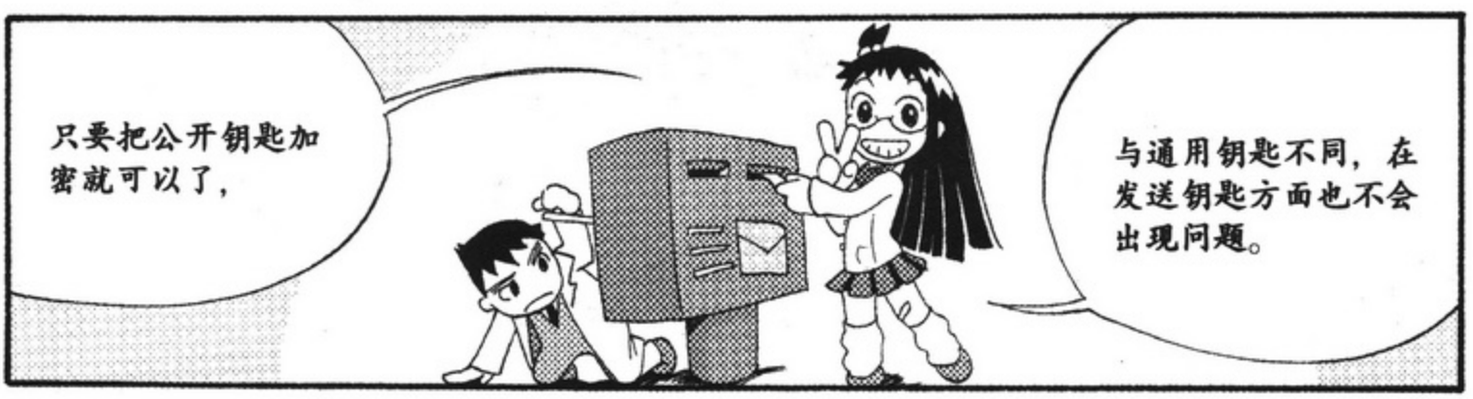


使用公开钥匙加密的时候，n 名使用人用密码互相通信，钥匙的总数有 $2n$ 个就可以了。

在 1000 人使用的情况下，用通用钥匙加密，

$$C_{1000}^{2} = \frac{1000 \times (1000-1)}{2}$$

需要 499500 个钥匙，但使用公开钥匙加密的话为 2×1000，钥匙数只需要 2000 个就可以。




这真是太好了！
不用通用钥匙加密，只使用公开钥匙加密行吗？
可是…… 使用公开钥匙，钥匙长度就变长啦？
通用钥匙的长度是 64，128，256bit，而公开钥匙有 1024bit 那么长。
其实公开钥匙加密需要很大的运算量，
处理的时间会很长。
那么，长篇文章加密的时候……
在第四章会学到的。
通用钥匙加密和公开钥匙加密组合使用。
嗯？
混合加密（参照第 188 页）等方法都在被广泛使用。
那么，使用通用钥匙进行密码通信的话，
虽然当事者们的互相通信是一目了然的……
嗯


但使用公开钥匙的话，是谁送来的数据，收件人根本无法确认啊！
篡改
冒名诈骗
网络钓鱼
诈骗
所以需要本人的身份认证嘛！
关于这个我们后面（参照第 198 页）会学到哦！
那么，先告诉我们种类和构成吧！
公开钥匙加密与通用钥匙加密相同，有很多种方法。

✖ 公开钥匙加密方式的主要种类

公开钥匙加密方式按照密码魔法的技巧种类，可分为两大类：

“密码魔法的技巧：
素因数分解问题”

· RAS 密码
Rabin 密码等

“密码魔法的技巧：
离散对数问题”

· ElGamal 密码
椭圆曲线密码
DSA 认证等

✤ 单向函数

能够通过单向的计算得出结论，但反方向的计算非常困难，具有这种特性的函数称为单向函数。

在这里，让我们来看一下单向函数的举例。

（1）素因数分解问题

我们很容易计算将 2 个较大素数相乘的结果。但是相反，如果想要从相乘后的数（合数），来计算原来的素数，则非常困难。

通过合数来计算原来的素数称为素因数分解问题（参看第 122 页）。

（2）离散对数问题

让我们看下面的线性方程式：

$$a^x = y \pmod{p}$$

如果知道 a 和 x 的值，算出 y 是比较容易的。但是，从 a 和 y 的值，计算出 y 的对数 x 则非常困难。这就是离散对数问题。离散是连续的反义词，表示分散而不连续的数值（参看第 175 页）。

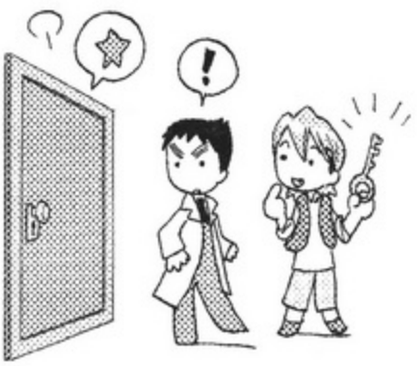

这里为什么是单向的呢？
不好辨驳的反对意见！
通用钥匙加密不是把一边加密的程序送到另一边再解密的吗？
使用公开钥匙加密的时候，如果不是单向性的话，会有私密钥匙被推算出来的危险啊！
注意
哎呀！
喂！喂！
但是完全使用单向函数，不就无法进行解密了吗？
解密使用私密钥匙不就行了吗？
啪
没错！
有那样的私密钥匙，
这样的单向函数叫做单向陷门函数。

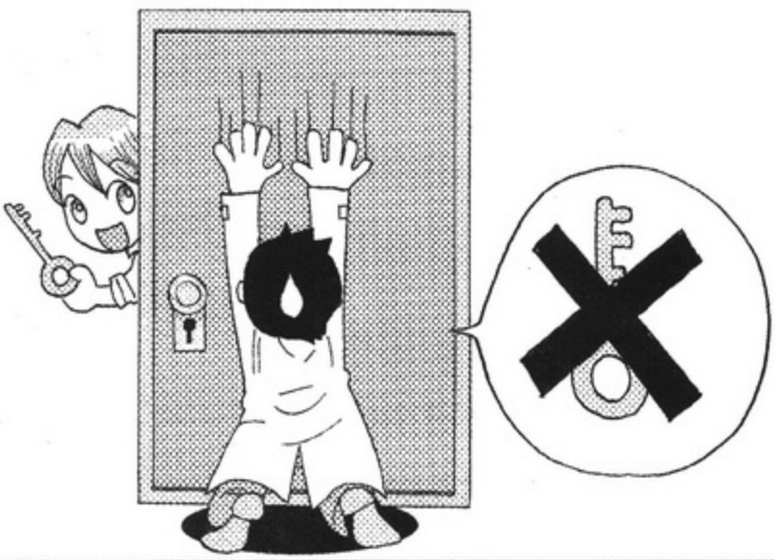
从有自动锁的门出来，如果没有钥匙的话是回不去的。类似于这样的函数，就叫做单向陷门函数。

即使没有钥匙也可以从房间里出来。

没有钥匙就进不去房间。

有钥匙才可以进房间。





RSA 密码的诞生

RSA 密码是在 1977 年被公布的世界上最早的公开钥匙密码。

RSA 的名字是由美国 3 名开发研究员，罗纳德 · 李维斯特（Ron Rivest），阿迪 · 萨莫尔（Adi Shamir）和伦纳德 · 阿德曼（Leonard Adleman）的姓氏开头字母拼在一起组成的。

利用素因数分解问题能够保证密码的安全强度。在当年的科学杂志上刊登了由他们 3 人提出的问题。该问题就是，将某些数分解为素因数后，并解读出该信息。

当时提出的就是下列的 129 个自然数。

114381625757888867669235779976146612010218296721242
362562561842935706935245733897830597123563958705058
989075147599290026879543541

这个素因数分解结果，在 17 年后的 1994 年，使用约 1600 台计算机进行计算，终于被破解出来了。17 年的时间，虽然一般人都会认为这是一个过于漫长的过程，但相对于其中一位 RSA 开发人员罗纳德 · 李维斯特预测要花费 1000 年的说法来看，还是要快很多了。破解的信息内容是“THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE”。

现在，在 RSA 密码中所使用的是十进制的 300 位以上的数字。如果将这些素因数进行分解的话，将需要花费相当于天文数字那么长的时间。

3-2 素数和素因数分解


1 2 3 4
21 22 23 24 25
41 42 43 44 45
61 62 63 64
83 84


首先，发一下资料吧！
数学
数学
这也是为了更好地理解RSA密码嘛！
很有趣哦♪


有趣的
素数


有趣的素数……
不像啊？
一点意思都没有！
我可是连分数的除法运算都没学过啊，还是算了吧！
抓狂

RSA 密码在数学上的运用只有非负整数，

要记住其中不会出现无理数和分数哦！

有理数

包括整数和分数

整 数

包括自然数，0 和负数（…，−2，−1，0，1，2，3，…）

自然数

包括 1 和 1 以上的整数（1，2，3，…）

非负整数

即为不是负数的整数（0，1，2，3，…）

分 数

表示 2 个整数之比，用小数来表示就是可变为有限小数和循环小数的数

$\left(-\frac{3}{2}=-1.5,\right.$

$\frac{1}{7}=0.142857142857142857142857\cdots$

$\left.=0.\dot{1}42857\dot{1}\text{ 等}\right)$

无理数

无法表示 2 个整数之比，即无限不循环小数。

（$\sqrt{2}$，π，e 等）

啊？真的吗？

那就学学看！

那我给你出个题！

这里有 30 个橘子，

橘子

橘子

平均分给孩子们，不能有剩余，应该怎么分？

这个简单！

表 3.1　人数与个数的关系

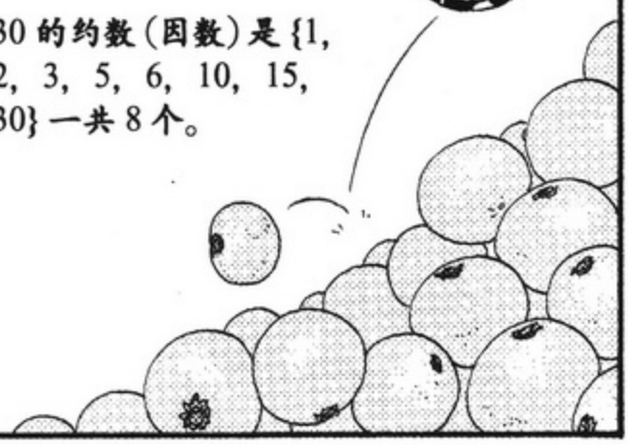
人　数	每人分配的个数
1 人	30 个
2 人	15 个
3 人	10 个
5 人	6 个
6 人	5 个
10 人	3 个
15 人	2 个
30 人	1 个

完全正确！

像这样没有剩余的分配方法相对人数或个数的数字，

称作“约数”或者是“因数”。

30 的约数（因数）是 {1, 2, 3, 5, 6, 10, 15, 30} 一共 8 个。



有的自然数的约数只有 1 和其本身，我们称之为“素数”或“质数”。

1 不是素数吗？

1

数学里规定素数中不包括 1。

1，对不起哦。

来看一下 20 以内的素数吧！

表 3.2　20 以内的素数判断

2	只能被 2 本身和 1 整除	是素数
3	只能被 3 本身和 1 整除	是素数
4	可以被 2 整除	不是素数
5	只能被 5 本身和 1 整除	是素数
6	可以被 2，3 整除	不是素数
7	只能被 7 本身和 1 整除	是素数
8	可以被 2，4 整除	不是素数
9	可以被 3 整除	不是素数
10	可以被 2，5 整除	不是素数
11	只能被 11 本身和 1 整除	是素数
12	可以被 2，3，4，6 整除	不是素数
13	只能被 13 本身和 1 整除	是素数
14	可以被 2，7 整除	不是素数
15	可以被 3，5 整除	不是素数
16	可以被 2，4，8 整除	不是素数
17	只能被 17 本身和 1 整除	是素数
18	可以被 2，3，6，9 整除	不是素数
19	只能被 19 本身和 1 整除	是素数
20	可以被 2，4，5，10 整除	不是素数

$4 = 2^2 = 2 \times 2$
$6 = 2 \times 3$
$8 = 2^3 = 2 \times 2 \times 2$
$9 = 3^2 = 3 \times 3$
$10 = 2 \times 5$
$12 = 2^2 \times 3 = 2 \times 2 \times 3$
$14 = 2 \times 7$
$15 = 3 \times 5$
$16 = 2^4 = 2 \times 2 \times 2 \times 2$
$18 = 2 \times 3^2 = 2 \times 3 \times 3$
$20 = 2^2 \times 5 = 2 \times 2 \times 5$

每个合数的素因数的分解方法只有一种，这叫做素因数分解的唯一性。

素数之所以不包括 1，也是为了保证素因数分解的唯一性哦！

啊——

为什么要一个一个地去确认是不是素数呢？

有一种叫做厄拉多塞筛法的方法。

这种方法运用了下面的性质哦！

自然数 N 不能被小于 N 开方的所有素数整除，这个自然数 N 就是素数。

为什么会那样呢？

从 $N=pq$ 来看，

$$N=pq$$

$$p \leqq \sqrt{N}$$
$$q \leqq \sqrt{N}$$

N 等于两个自然数 p, q 相乘，q, p 都必须小于 $\sqrt{N}$ 的开方。

原来如此！

p 和 q 要是大于 $\sqrt{N}$ 的开方的话，p 和 q 的乘积就会大于 N 了。

$$p > \sqrt{N} \qquad q > \sqrt{N}$$
$$\downarrow$$
$$pq > N$$

那……那个厄拉什么什么的是怎么回事啊？

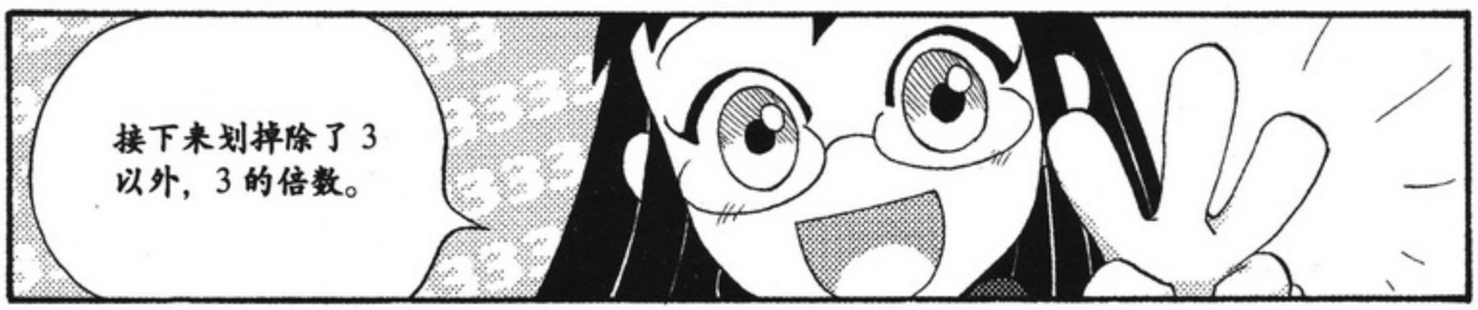
我是古希腊的学者啊！

厄拉多塞

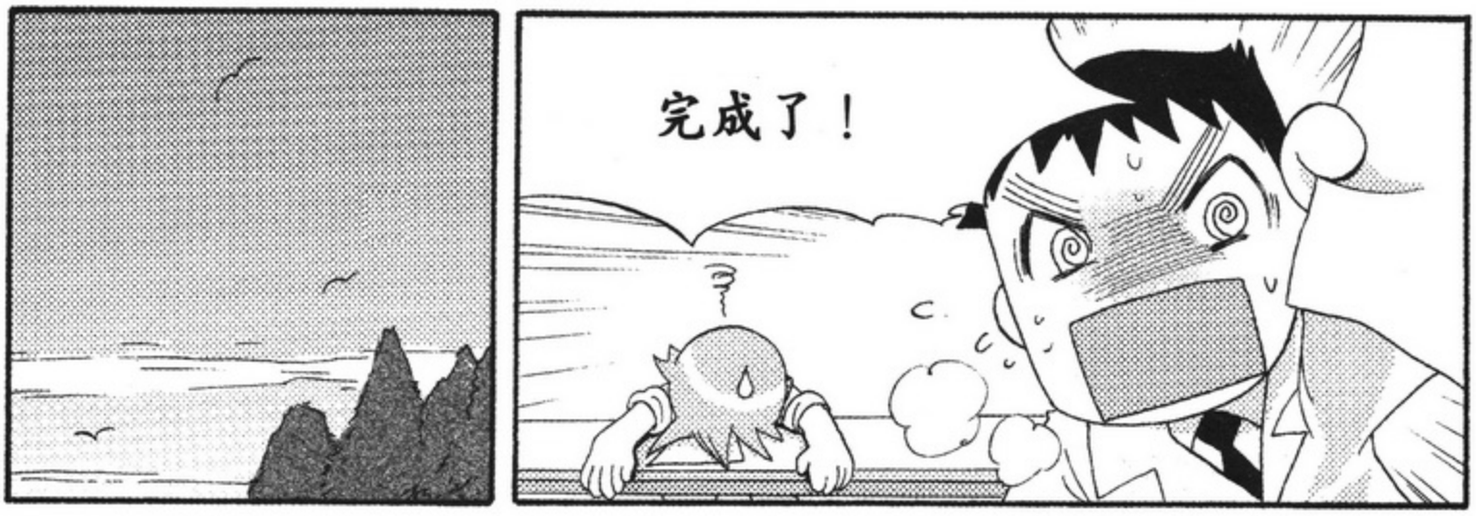
第一个计算出地球周长的人就是我！

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220
221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260
261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280
281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320
321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340
341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360
361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380
381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220
221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260
261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280
281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320
321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340
341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360
361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380
381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400



	2	3	5	7	9	11	13	15	17	19
21		23	25	27	29	31	33	35	37	39
41		43	45	47	49	51	53	55	57	59
61		63	65	67	69	71	73	75	77	79
81		83	85	87	89	91	93	95	97	99
101		103	105	107	109	111	113	115	117	119
121		123	125	127	129	131	133	135	137	139
141		143	145	147	149	151	153	155	157	159
161		163	165	167	169	171	173	175	177	179
181		183	185	187	189	191	193	195	197	199
201		203	205	207	209	211	213	215	217	219
221		223	225	227	229	231	233	235	237	239
241		243	245	247	249	251	253	255	257	259
261		263	265	267	269	271	273	275	277	279
281		283	285	287	289	291	293	295	297	299
301		303	305	307	309	311	313	315	317	319
321		323	325	327	329	331	333	335	337	339
341		343	345	347	349	351	353	355	357	359
361		363	365	367	369	371	373	375	377	379
381		383	385	387	389	391	393	395	397	399



	2	3		5		7				11		13				17		19	
		23						29		31						37			
41		43				47						53						59	
61						67				71		73						79	
		83						89								97			
101		103				107		109				113							
						127				131						137		139	
								149		151						157			
		163				167						173						179	
181										191		193				197		199	
										211									
		223				227		229				233						239	
241										251						257			
		263						269		271						277			
281		283										293							
						307				311		313				317			
										331						337			
						347		349				353						359	
						367						373						379	
		383						389								397			

✤ 素数的判定

厄拉多塞筛选法，是能够准确查找素数的方法。可是，判定比较大的数是否为素数，则需要很长时间。

因此，我们采用另外的判定算法，虽然不是 100% 准确，但误差率也不是很高。

费尔马方法说的是在某一个整数 a 与要判断是否是素数的 n 之间，如果 $a^{n-1}=1(\mathrm{mod}\ n)$ 的等式成立，则 n 可在概率上判定为素数（参照第 156 页）。但是，仍然存在着极小的误差，可能将非素数（合数）判定为素数。

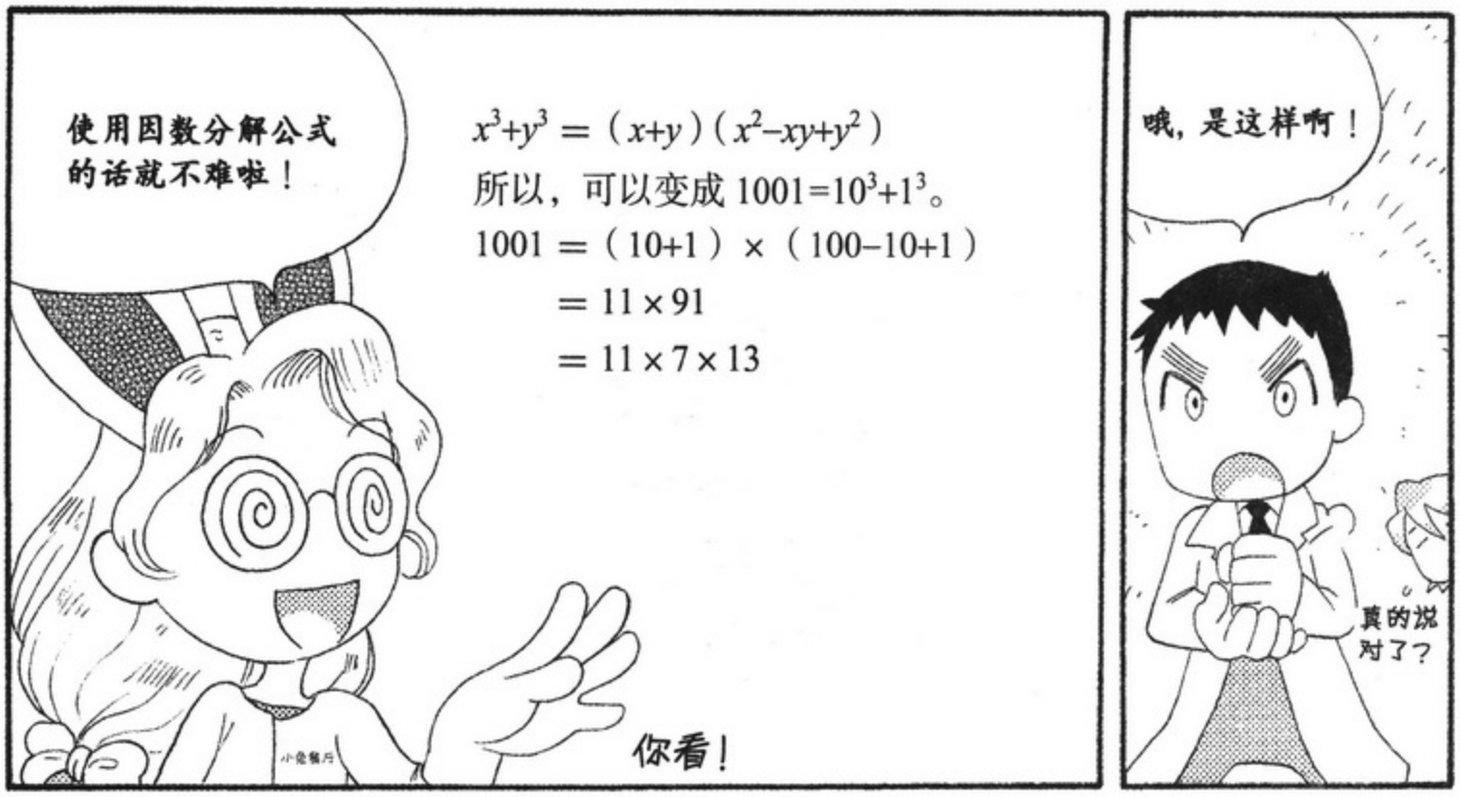
于是，出现了将费尔马算法进行改进的 Miler-Rabin 算法。在一次测试中，错误判断几率达到了费尔马算法的四分之一以下，几乎能完全准确地判断一个数是否为素数。

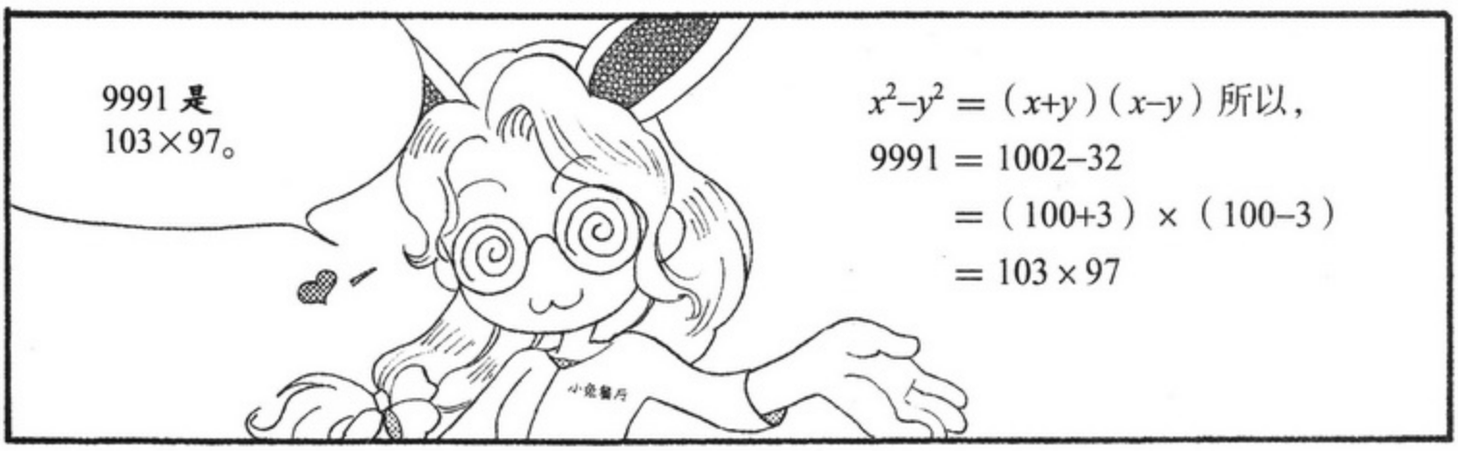

下面，我们来学习一下难懂的素因数分解吧！
理绪小姐做一下35的素因数分解吧！
简单！
是5×7。
正确！
那么77的素因数的分解呢？
是7×11。
又答对了！
正确！
这次轮到哥哥啦！
1001的素因数分解是？
啊——

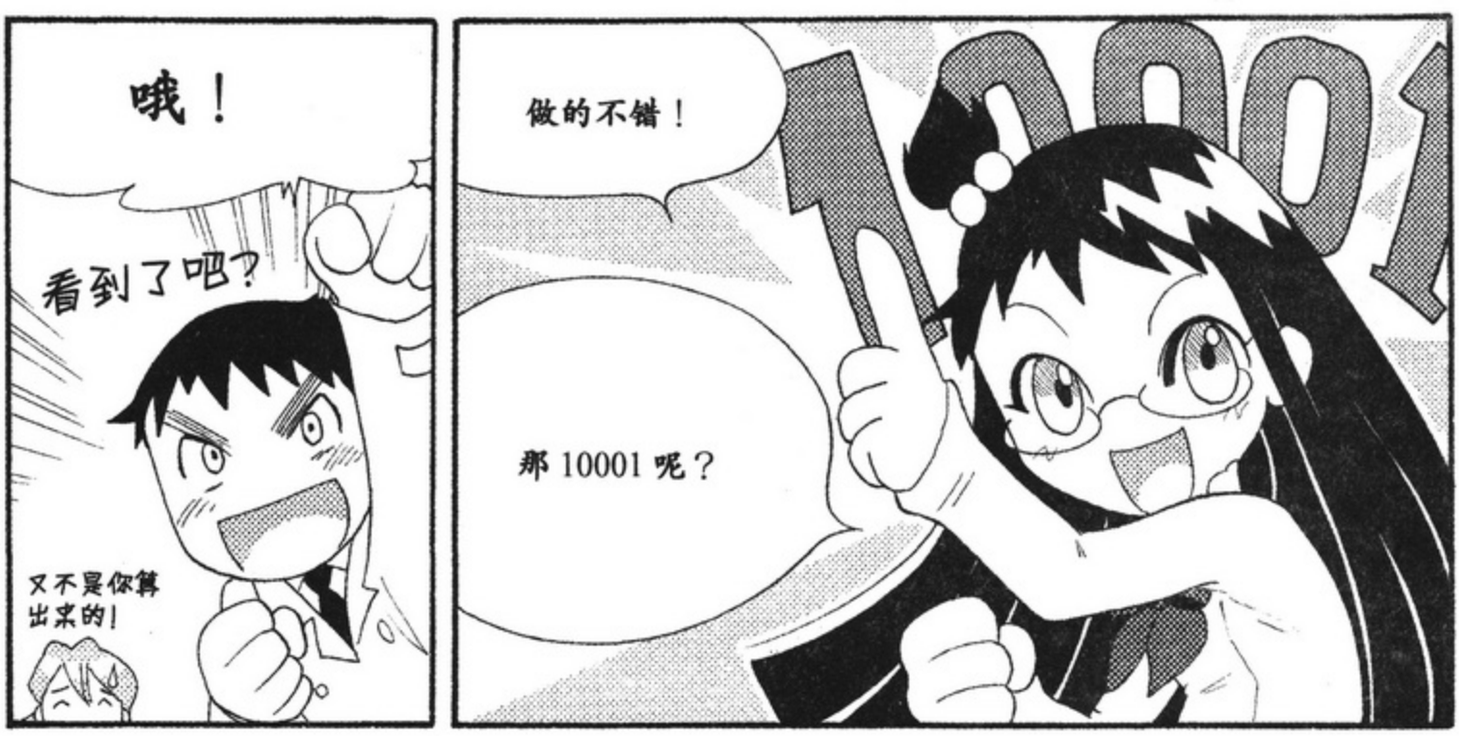

嗯
那9991的素因数
分解呢？
嗯


为什么轮到我，问题
就变难了呢？
1001是
7×11×13吧？


正确！
哦！
您的拉面，
让您久等啦！
小兔餐厅


使用因数分解公式的话就不难啦！
$x^3+y^3=(x+y)(x^2-xy+y^2)$
所以，可以变成 $1001=10^3+1^3$。
$1001=(10+1)\times(100-10+1)$
$=11\times91$
$=11\times7\times13$
你看！
哦，是这样啊！
真的说对了？


9991 是 103×97。
$x^2-y^2=(x+y)(x-y)$ 所以，
$9991=1002-32$
$=(100+3)\times(100-3)$
$=103\times97$


哦！
看到了吧？
又不是你算出来的！
做的不错！
那 10001 呢？
10001

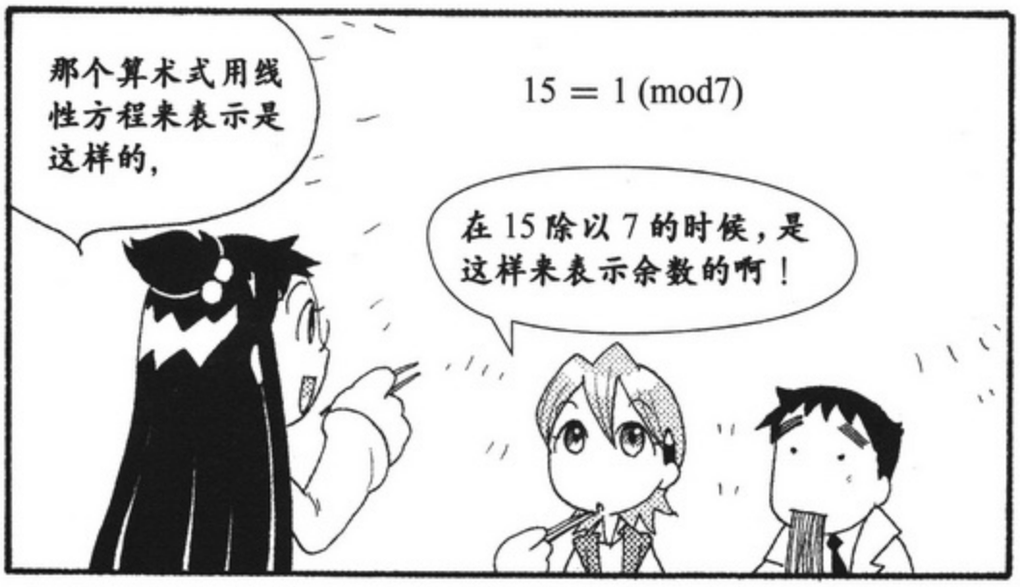
从小于$\sqrt{10001}$开方的素数
{2，3，5，7，11，13，17，19，23，29，31，37，41，43，47，53，59，61，67，71，73，79，83，89，97} 中一个一个地找出 10001 的约数候补。得出的结果是 10001=73 × 137。

3-3 取模运算

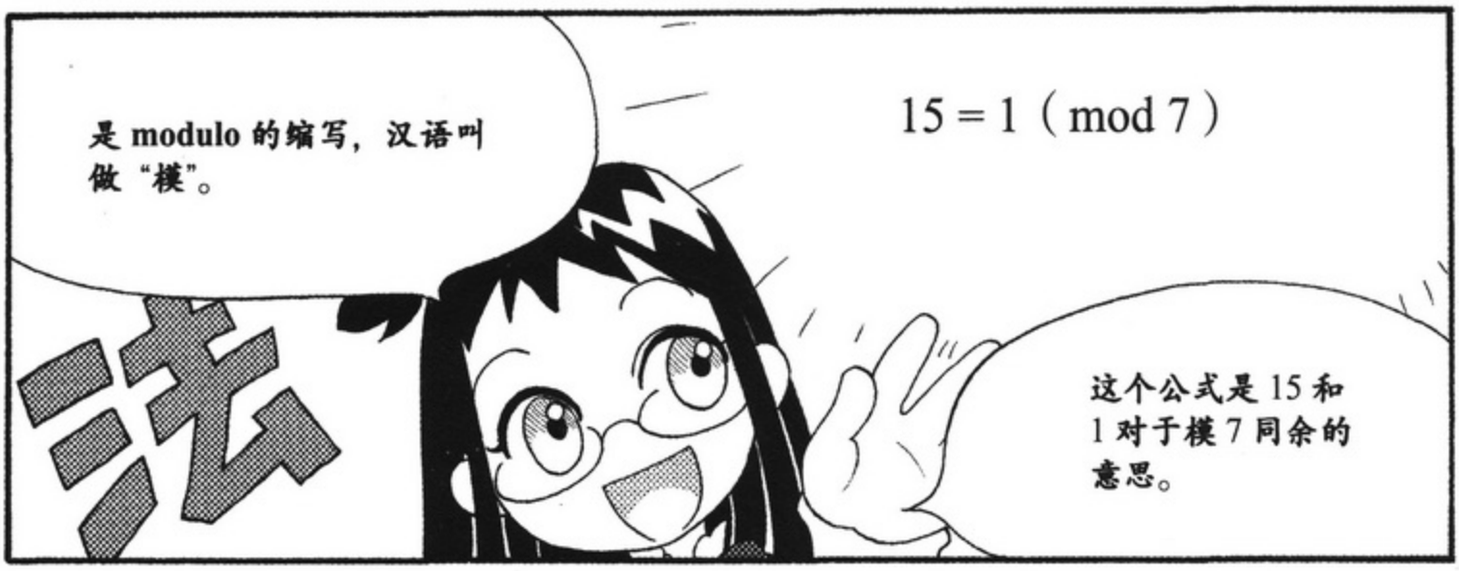

筷子
筷子


现在我们来学习一下整数除法运算的商。
如果不学好取模运算，就无法理解 RSA 密码哦！


是这么算的吧。
在小学学过的除法运算就有余数的例子 15÷7=2 余 1。
嘶嘶

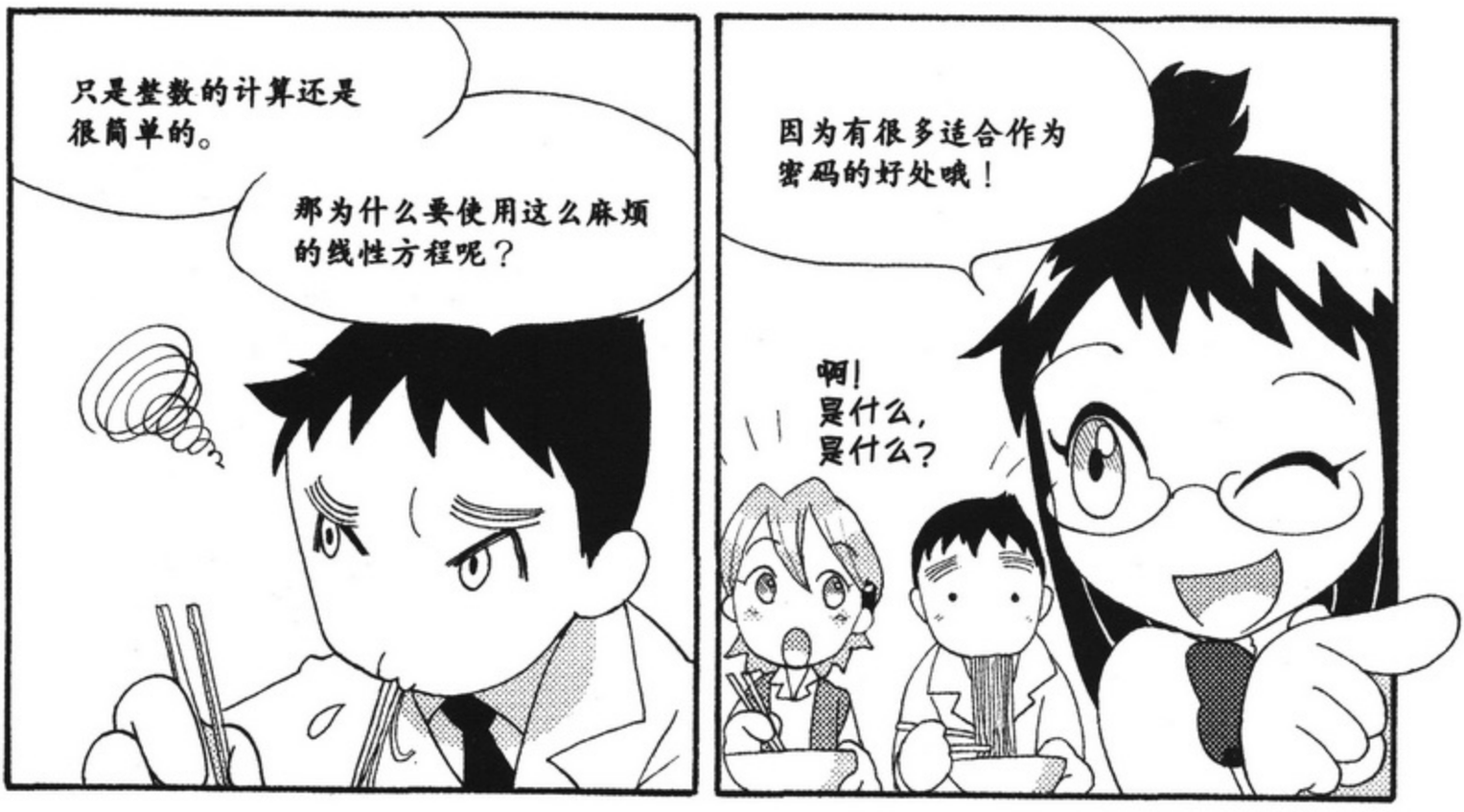

那个算术式用线性方程来表示是这样的，
15 = 1 (mod7)
在 15 除以 7 的时候，是这样来表示余数的啊！


mod 是什么？



$a = b \pmod{N}$

这种普通的线性方程，也称作模运算。是当 N 为模的时候 a 和 b 对于 N 同余的意思。所使用的符号不是“=”，而是“≡”。



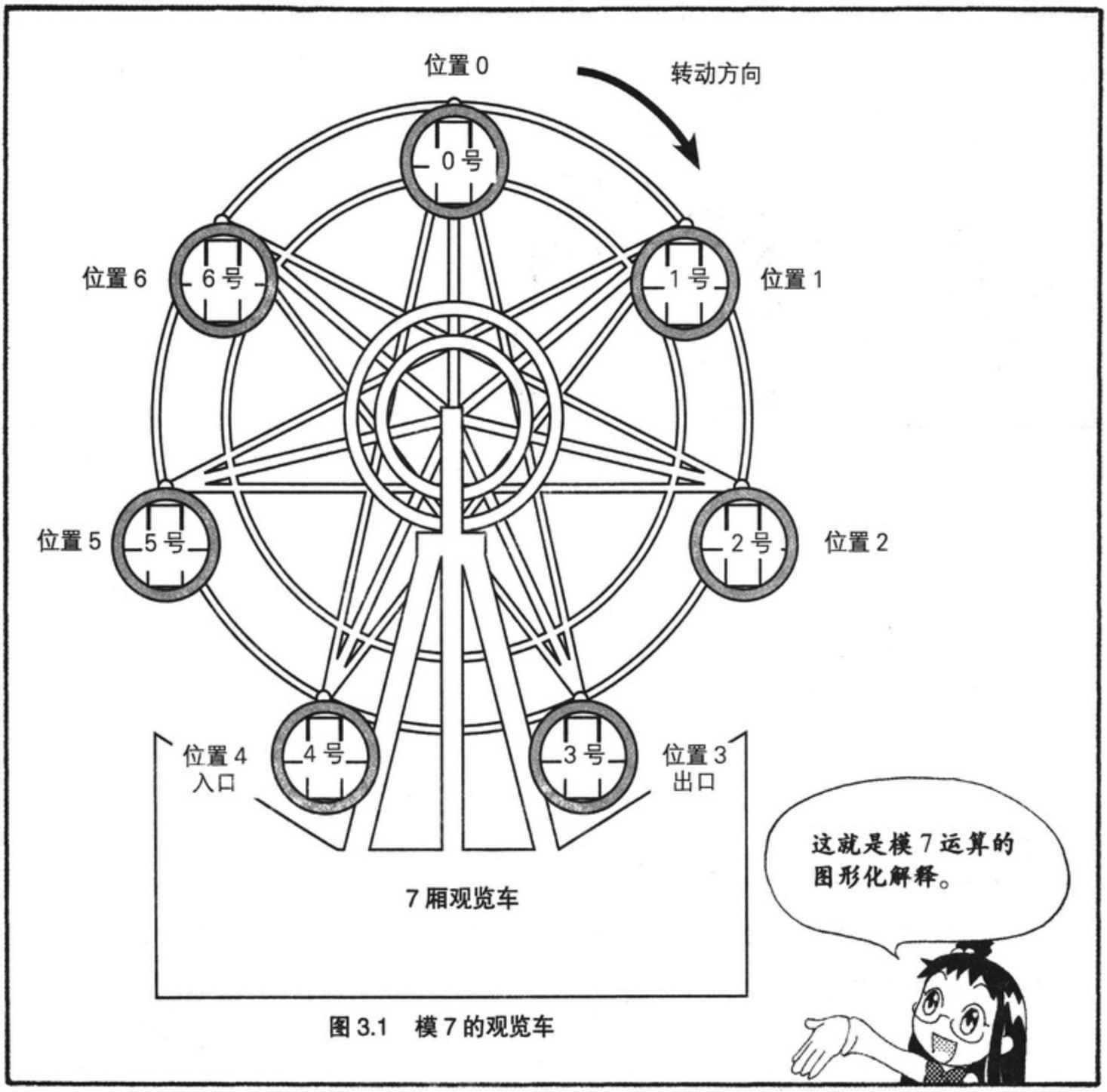
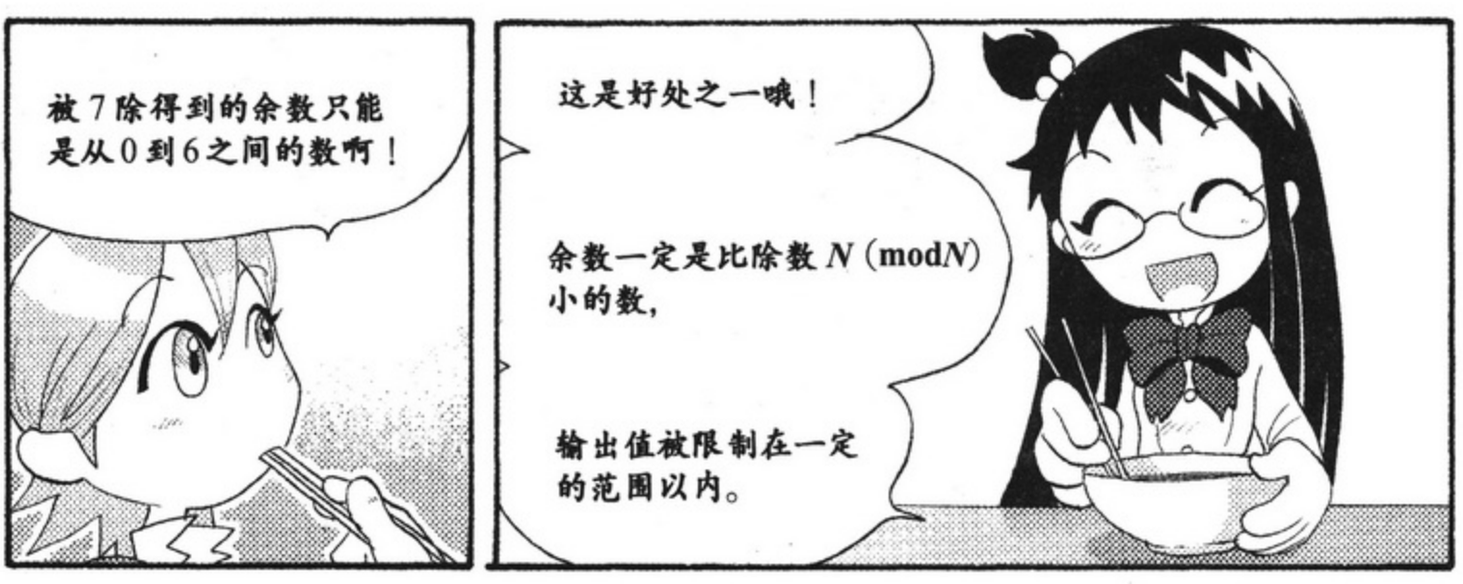

位置 0
转动方向
0 号
位置 6
6 号
1 号
位置 1
位置 5
5 号
2 号
位置 2
位置 4
入口
4 号
3 号
位置 3
出口
7 厢观览车
这就是模 7 运算的图形化解释。

图 3.1　模 7 的观览车


被 7 除得到的余数只能是从 0 到 6 之间的数啊！
这是好处之一哦！
余数一定是比除数 N (mod$N$) 小的数，
输出值被限制在一定的范围以内。

❀ 取模运算的加法运算和减法运算

采用如图 3.1 观览车为取模运算的模型。将 7 个车厢分别编为 0~6 号。并将车厢的编号，最上面设定为 0 号，按顺时针方向等角度地将 1~6 号进行定位。位置 3 的下车口与位置 4 的上车口相通。

初始状态为，0 号车厢在位置 0 上，1 号车厢在位置 1 上……所有的车厢与号码位置一致。然后，将加法运算设定为车厢顺时针旋转。

首先，观察 0 号车厢的变化。

旋转 1/7 周时，0 号车厢将从位置 0 移动到位置 1。将此定义为 +1（增加 1）。

旋转 2/7 周时，0 号车厢将从位置 0 移动到位置 2。将此定义为 +2（增加 2）。

旋转 7/7 周时，也就是完全旋转了一周，0 号车厢将从位置 0 重新回到位置 0。此时为 +7。+7 与 0 等同，即没有移动。

利用这个观览车模型，可以对所有加法运算进行说明。

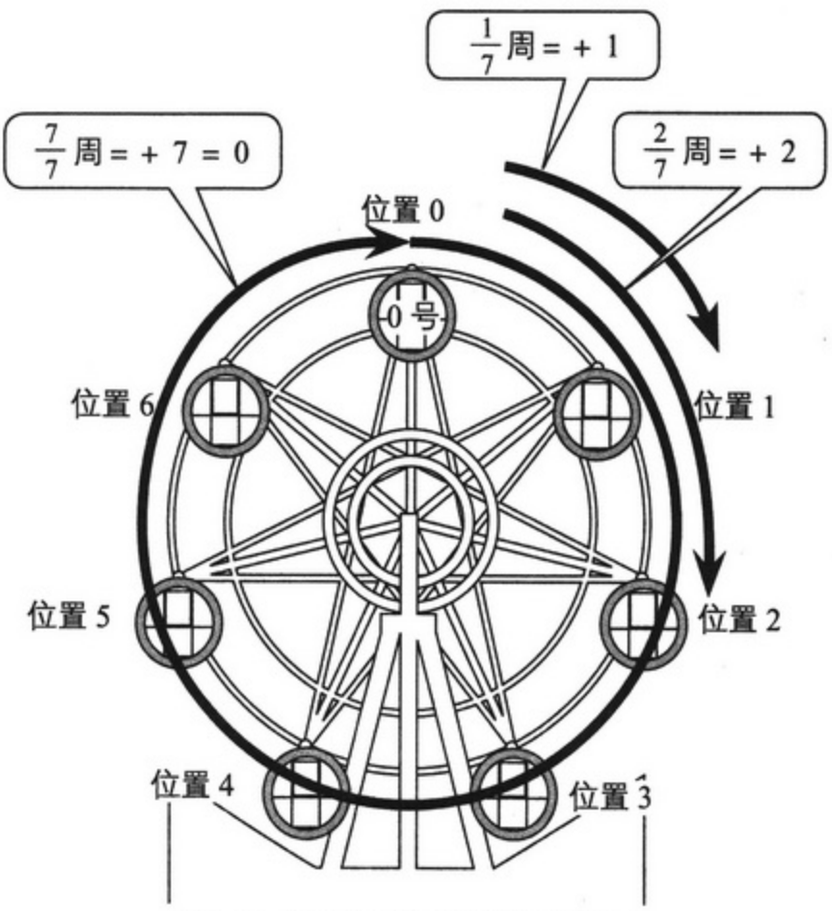


图3.2　取模加法运算的模型（1）

例如，分析 5+6 的结果。

5+6 中的 5，设定为初始状态的 5 号车厢。如果将 5 号车厢旋转 6/7 周，结果将会移动到什么位置?

如果按顺时针移动 6 个位置，如图 3.3 所示可移动到位置 4。

用下列等式来计算：

$$5+6=4\ (\mathrm{mod}\ 7)$$

将初始状态的车厢号码（车厢的位置号码）设定为 a，将旋转的移动设定为 $b/7$，可以得出移动后的位置与加法运算的结果一致。利用此方法，算出加法运算的所有组合，可确认得出表 3.3 的结果。

下面，继续用观览车，来说明一下减法运算。

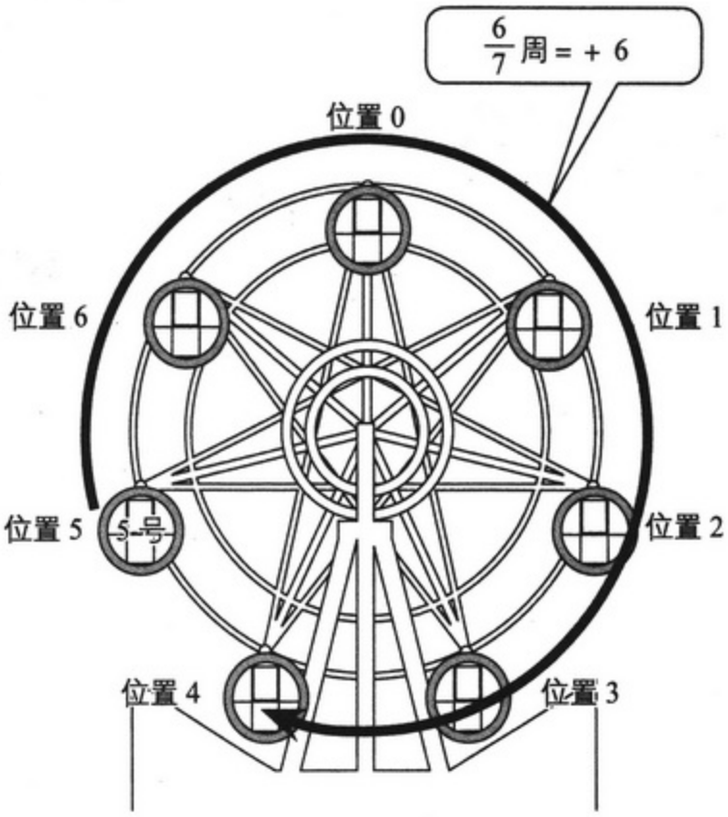


图 3.3　取模加法运算的模型（2）

表 3.3　取模 7 的 $a+b$

a \ b	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

首先，观察 0 号车厢的变化。

逆时针旋转 1/7 周时，0 号车厢将从位置 0 移动到位置 6。将此定义为 –1(减少 1)。

逆时针旋转 2/7 周时，0 号车厢将从位置 0 移动到位置 5。将此定义为 –2(减少 2)。

逆时针旋转 1 周时，0 号车厢将从位置 0 重新回到位置 0,此时为 –7。–7 与 0 等同，即没有移动。

利用这个观览车模型，可以对所有减法运算进行说明。

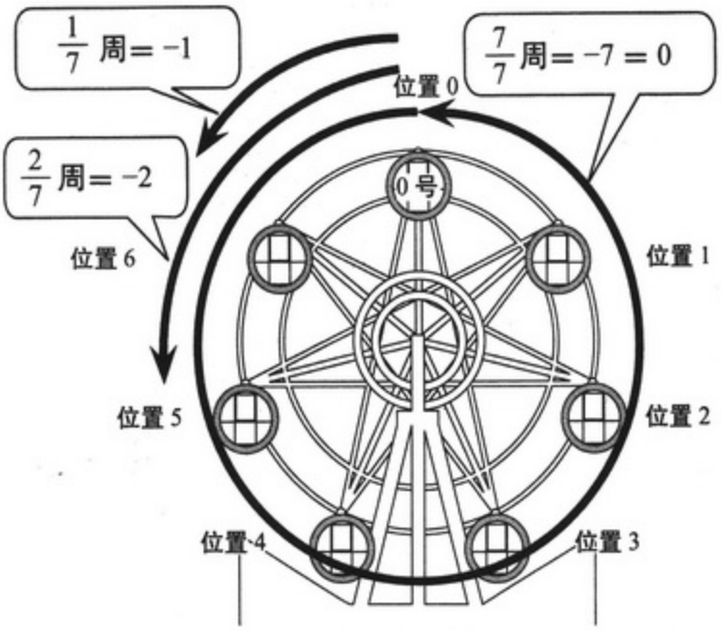


图 3.4　取模减法运算的模型（1）

例如,分析 3–4(从 3 里减去 4)的结果。

3–4 中的 3，设定为初始状态的 3 号车厢。如果将 3 号车厢逆时针旋转 4/7 周，结果将会移动到什么位置?

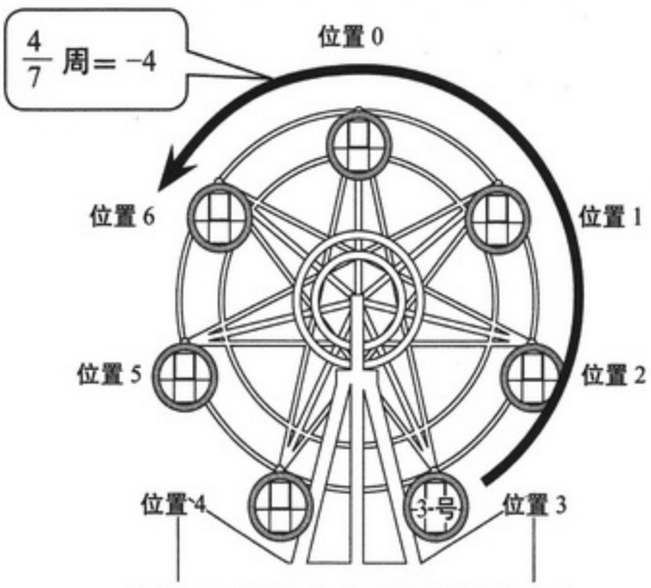


图 3.5　取模减法运算的模型（2）

如果按逆时针方向移动 4 个位置，可以知道将移动到位置 6。

即，下列等式成立。

$$3–4=6\ (\mathrm{mod}\ 7)$$

将初始状态的车厢号码（车厢的位置号码）设定为 a，将逆时针旋转的移动位置设定为 $b/7$，可以得出移动后的位置与减法运算的结果一致。利用此方法，可确认得出减法运算结果，如右表所示。

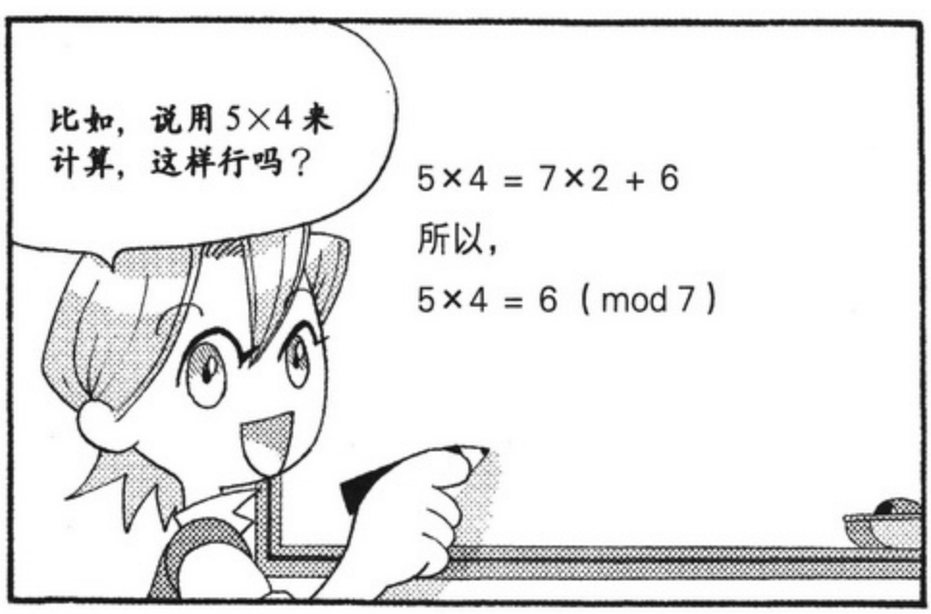
表 3.4　取模 7 的 $a–b$

a \ b	0	1	2	3	4	5	6
0	0	6	5	4	3	2	1
1	1	0	6	5	4	3	2
2	2	1	0	6	5	4	3
3	3	2	1	0	6	5	4
4	4	3	2	1	0	6	5
5	5	4	3	2	1	0	6
6	6	5	4	3	2	1	0


取模运算的加法运算和减法运算很简单吧？
我吃饱了。
简单！简单！
都是这样的话我也能学会！


时间的计算，日期的调整都是取模运算的一种哦！
CALENDER
20XX 7
SUN MON TUE WED THU FRI SAT
1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31


下面来学取模运算的乘法运算和除法运算吧！
来吧！


比如，说用5×4来计算，这样行吗？
5×4 = 7×2 + 6
所以，
5×4 = 6（mod 7）


就是这样，
模7的乘法运算表是这样的！

表 3.5　取模 7 的 $a \times b$

a \ b	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

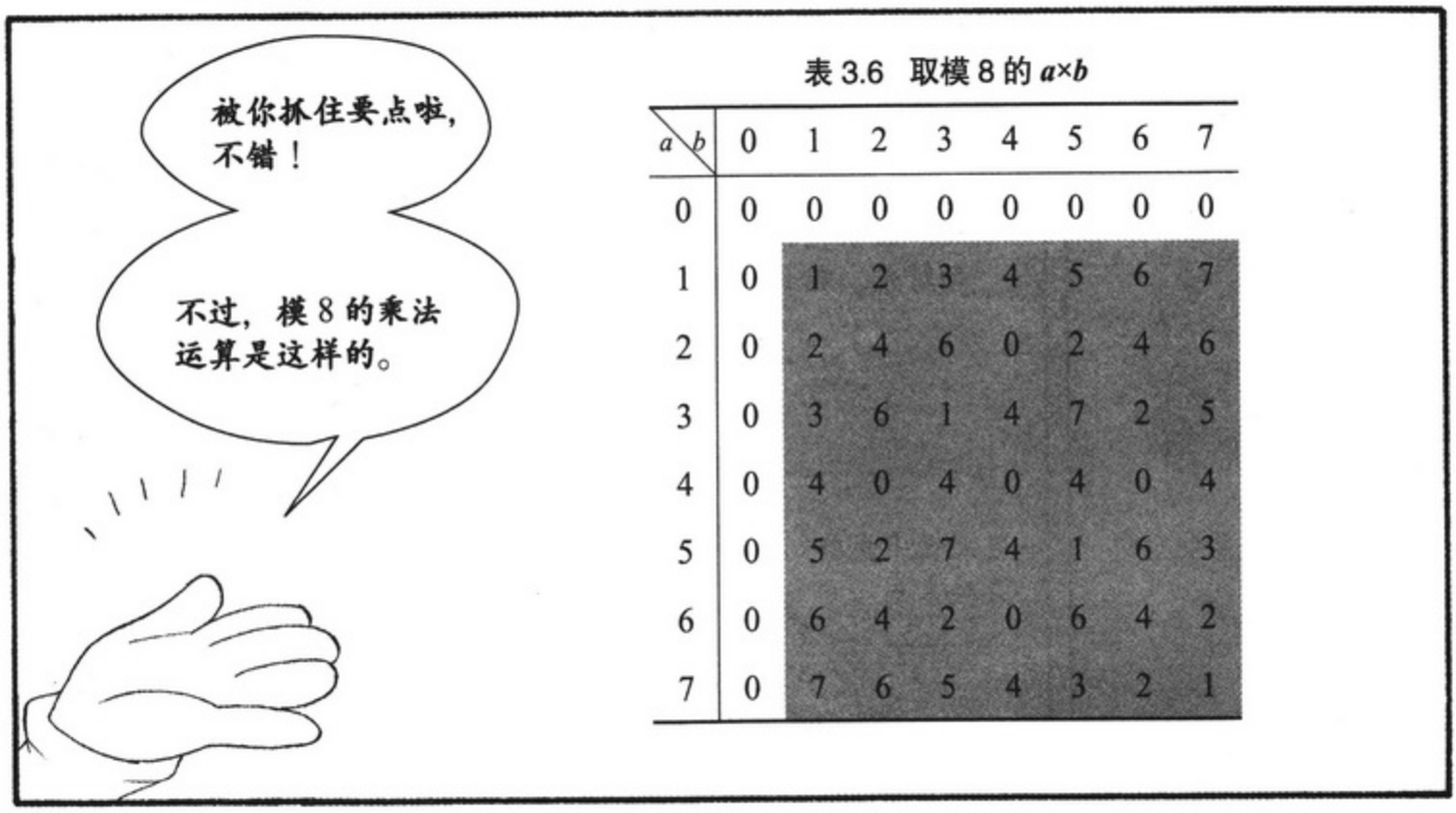
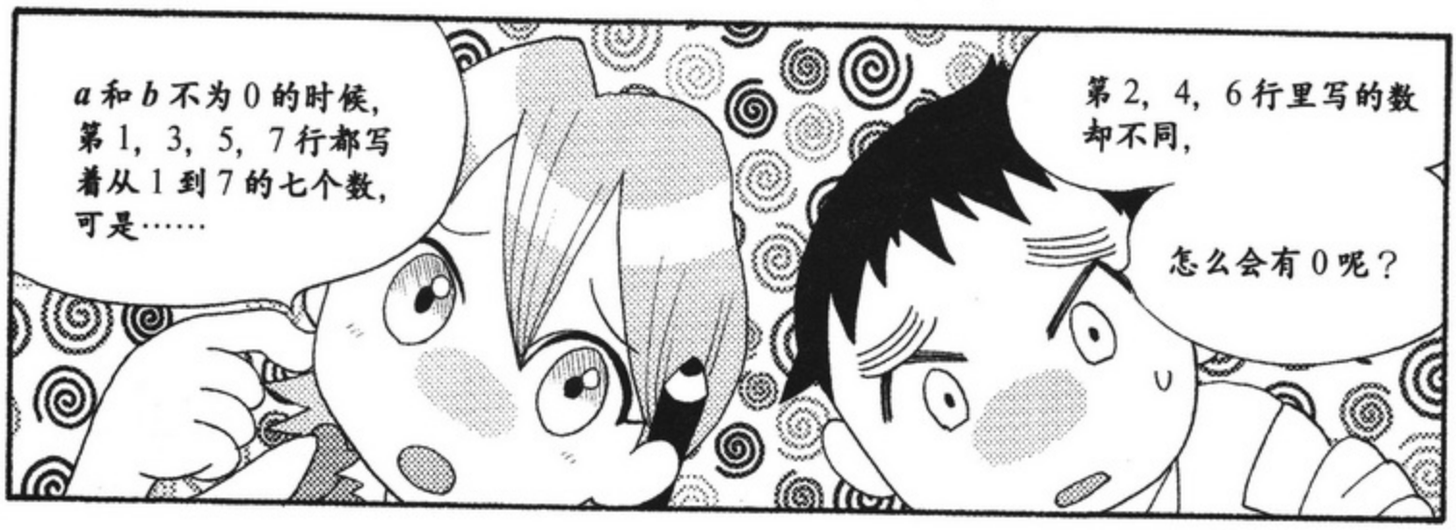


表 3.6　取模 8 的 $a \times b$

a \ b	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1



1. 日语中有些入浴剂称为“温泉素”，可以在家泡澡时用水冲开使用——译者注。

比如说8和2的话，除了1以外还有2是共同约数（公约数），所以就不是互素。那么，在模8的乘法运算表里4和6也像2和8一样有2这个公约数，所以也不是互素。

另一方面，1，3，5，7与8是互素。为什么会这样呢？请确认一下，如果两个数是互素的话，它们的最大公约数就一定为1。

像这样，所有的素数去掉它们的倍数之后会和什么样的整数互素，利用这个性质可以使用厄拉多塞筛选法来找出这些素数。

表 3.7　取模 7 的 $a \times b$

a \ b	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

从表 3.7 可以看出 1 的倒数是 1，2 的倒数是 4，3 的倒数是 5，4 的倒数是 2，5 的倒数是 3，6 的倒数是 6。

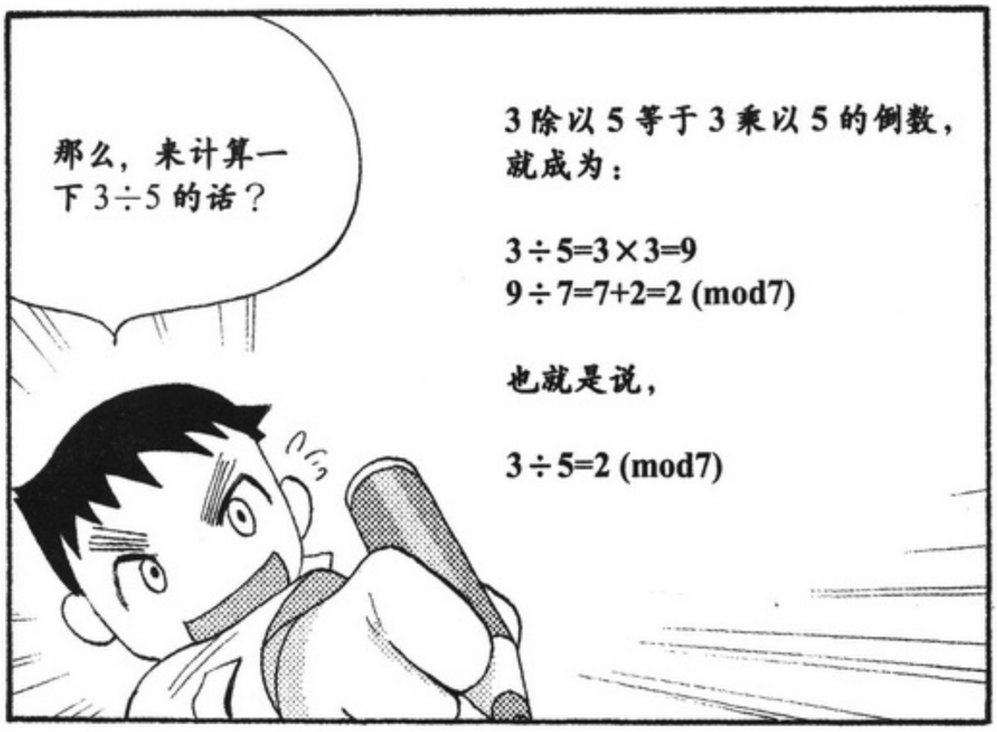


表 3.8　取模 7 的 $a \div b$

a \ b	0	1	2	3	4	5	6
0	–	0	0	0	0	0	0
1	–	1	4	5	2	3	6
2	–	2	1	3	4	6	5
3	–	3	5	1	6	2	4
4	–	4	2	6	1	5	3
5	–	5	6	4	3	1	2
6	–	6	3	2	5	4	1

✤ 取模运算的乘法运算和除法运算

乘法运算也可以通过 7 个车厢的观览车模型进行说明。

初始状态为，0 号车厢在位置 0 上，1 号车厢在位置 1 上，所有的车厢与号码位置一致。

乘法运算，可用车厢旋转的速度来考虑。

如果车厢在 1 分钟能旋转 1/7 周（即 7 分钟旋转 1 周）时，0 号车厢在 3 分钟后所移动的位置如下所示。

1（速度）×3（分钟）= 3（旋转后的位置）

如果车厢在 1 分钟能旋转 5/7 周时，6 分钟后所移动的位置如下所示。

$5 \times 6 = 30$，

$30 = 7 \times 4 + 2$（$30 \div 7 = 4$ 余 2）

因此，$5 \times 6 = 2 \pmod 7$

即 30 被 7 除，得出旋转 4 周和余 2 的结论。虽然旋转了 4 周，但因在 mod 的领域里与 0 旋转等同，所以只考虑余 2 的变化即可。

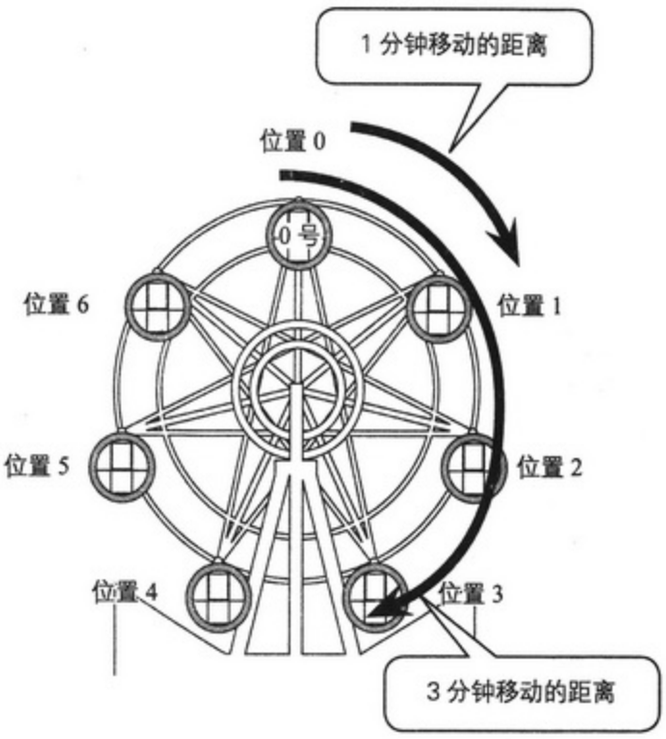


图 3.6　1 分钟旋转 $\frac{1}{7}$ 周的状态（1）

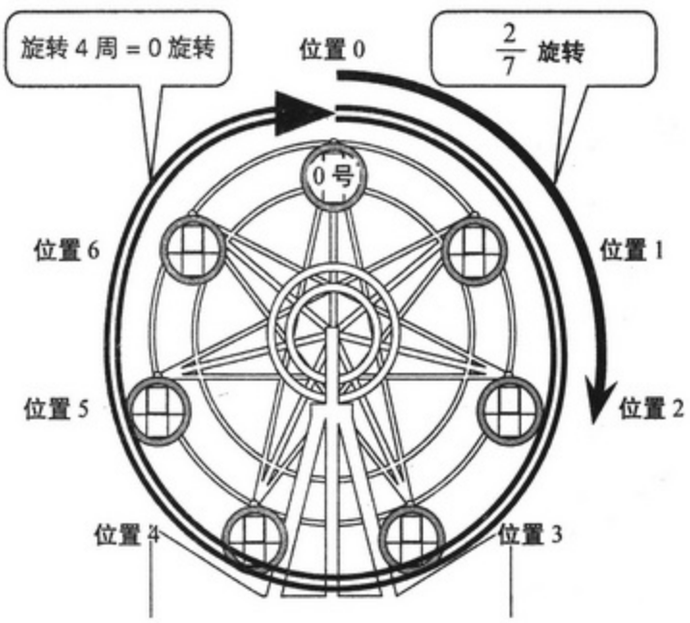


图 3.7　当 1 分钟旋转 $\frac{5}{7}$ 周时，6 分钟后的位置

除法运算与乘法运算相反。通过旋转后的位置和旋转速度，反算出车厢旋转所需要的时间。

如果车厢在 1 分钟能旋转 1/7 周，假设 0 号车厢从位置 0 移动到了位置 5，现在计算车厢旋转所需要的时间。

5（最后位置）÷1（速度）= 5（分钟）

即得出了旋转 5 分钟的时间。此外，12 分钟，19 分钟……虽然在一般（5+7*n*）的时间，都会在同一位置，但是在 mod 7 的领域里，时间上只有 0~6 分钟的区别，所以不管是 12 分钟还是 19 分钟，经过的时间都无法显示，都和 5 分钟得到同样的结果。

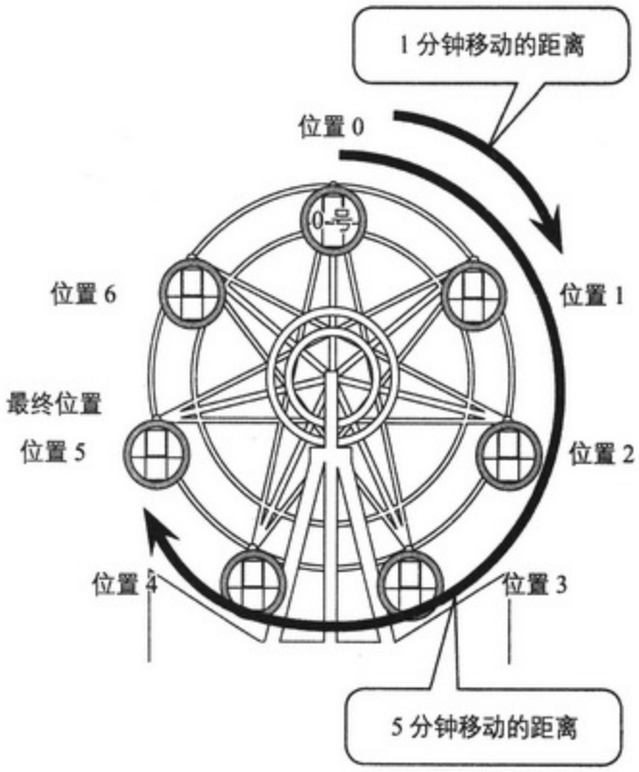


图 3.8　当 1 分钟旋转 $\frac{1}{7}$ 周的状态（2）

如果车厢在 1 分钟能旋转 2/7 周（即 7 分钟旋转 2 周）时，假设 0 号车厢从位置 0 移动到了位置 5，计算车厢旋转所需要的时间，用除法算表得出下列的结果。

5（最后位置）÷2（速度）= 6（分钟）

虽然得出了需要旋转 6 分钟的结果，但如何解释这个结果为好呢？

请这样理解。将最终位置 5，理解为多余旋转了 1 周。即实际上的最终位置是 2×6 = 12，但在 mod7 中则表示为 5。

因为 12÷2 = 6，所以得出正解“旋转了 6 分钟”。

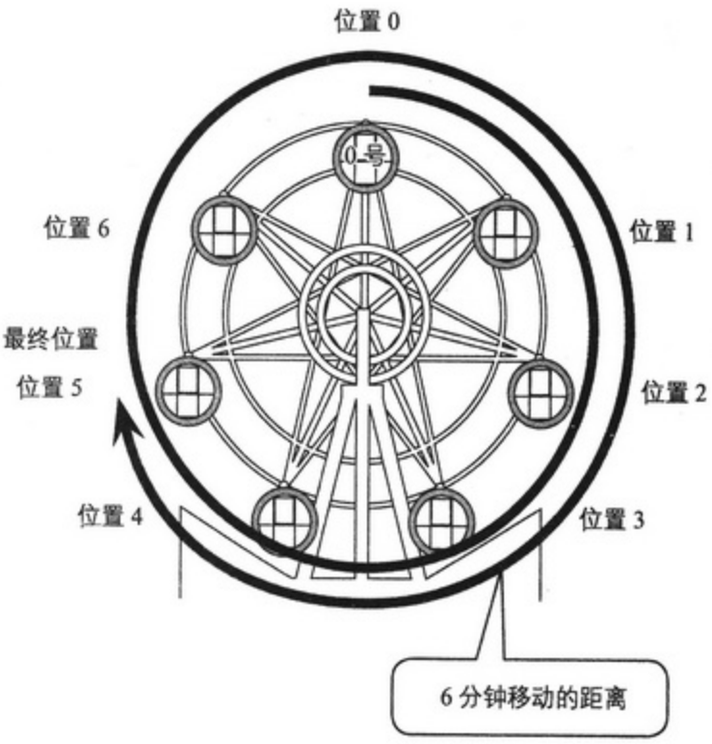
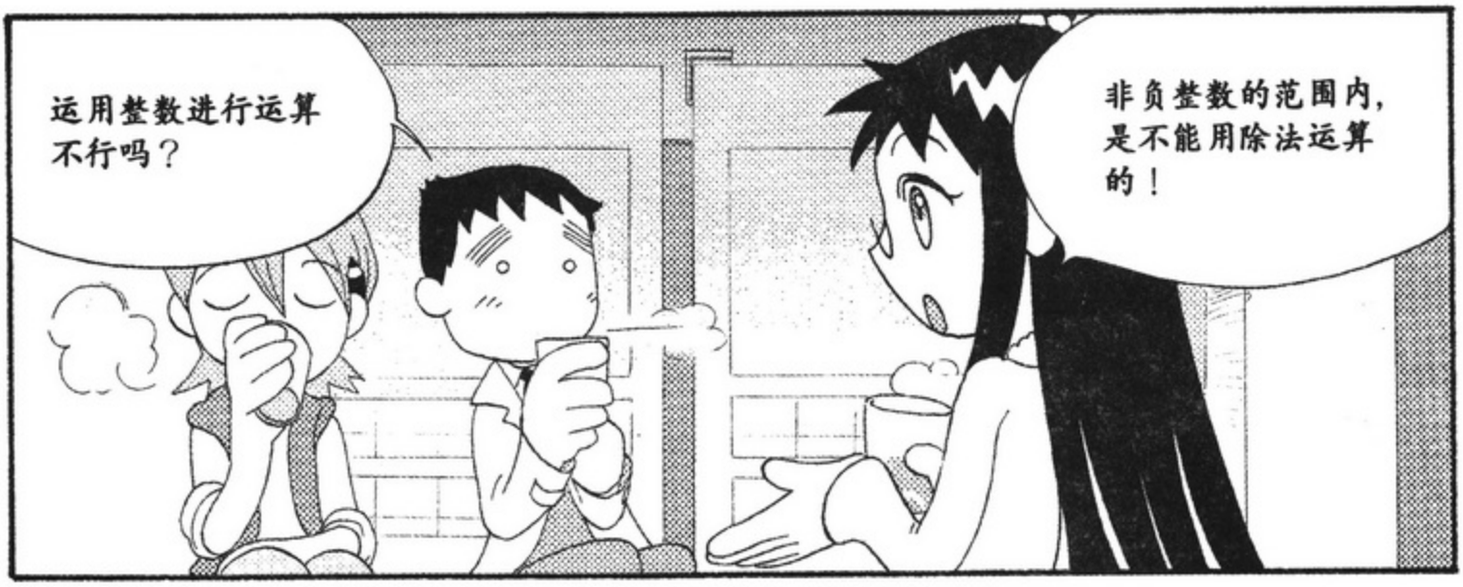


图 3.9　当 1 分钟旋转 $\frac{2}{7}$ 周的状态


综上所述，素数的模运算的四则运算就全都理解了吧？
四则运算就是加法运算、减法运算、乘法运算和除法运算这4种运算方法。


那个很了不起吗？
可以轻松自如的进行运算，非常适合在加密和解密的数学中应用哦！
无敌！
了不起


运用整数进行运算不行吗？
非负整数的范围内，是不能用除法运算的！

比如说 3÷8 的结果是分数（小数），而不是整数。

另一方面，素数 p 的取模运算，如果交换律、结合律、分配律成立，那么运算的结果一定是 $\{0, 1, \cdots, p-1\}$ 中的某一个数。

咻！

$a+b=b+a$　$ab=ba$ 是交换律
$(a+b)+c=a+(b+c)$
$(ab)c=a(bc)$ 是结合律
$a(b+c)=ab+ac$ 是分配律

这样的数学体系被称之为“域”哦！

域

“域”中具有代表性的是有理数。成为有理数要素的数是无限大的。另一方面素数 p 的取模运算的要素是 0，1，…，$p-1$，所以 p 的个数是有限的，被称为“有限域”。

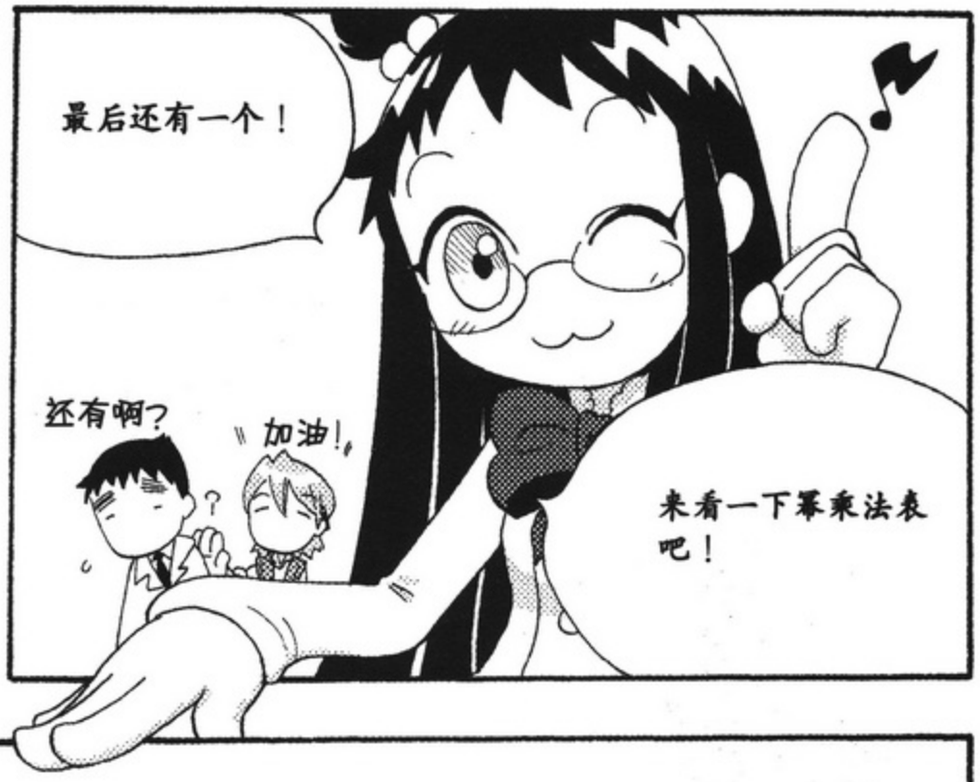
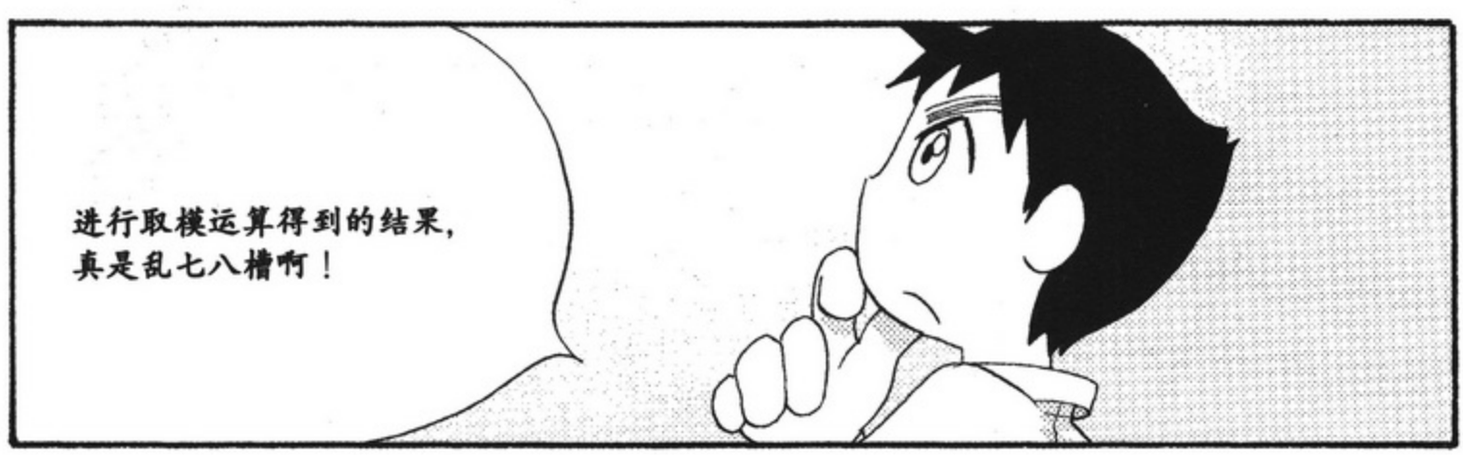


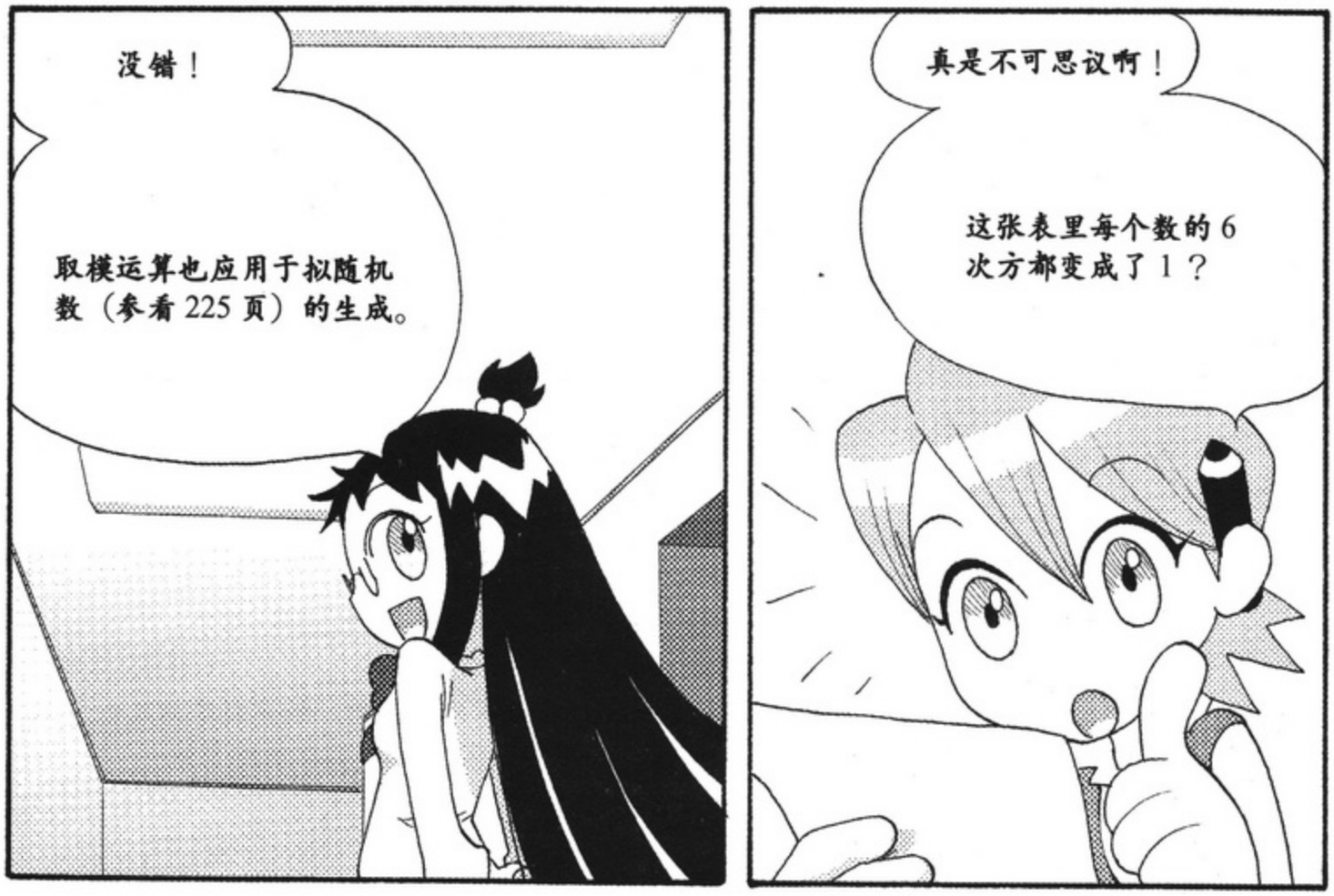
表 3.9　模 7 的 a^b（a 的 b 次方）

a \ b	1	2	3	4	5	6
1	1	1	1	1	1	1
2	2	4	1	2	4	1
3	3	2	6	4	5	1
4	4	2	1	4	2	1
5	5	4	6	2	3	1
6	6	1	6	1	6	1

表 3.9 中的各个数字的 6 次方再除以 7，每个结果都会余 1。

$$1^6 = 1 = 0 \times 7 + 1$$
$$2^6 = 64 = 9 \times 7 + 1$$
$$3^6 = 729 = 104 \times 7 + 1$$
$$4^6 = 4096 = 585 \times 7 + 1$$
$$5^6 = 15625 = 2232 \times 7 + 1$$
$$6^6 = 46656 = 6665 \times 7 + 1$$




没错！
取模运算也应用于拟随机数（参看225页）的生成。
真是不可思议啊！
这张表里每个数的6次方都变成了1？


正是这样！
兴奋
那就是和我们将要学习的费尔马小定理有关系哦！
费……费尔马？
嗯……

3-4 费尔马小定理和欧拉定理

费尔马小定理

n 是素数的时候，和 *n* 互素的某个整数 *a*（不是 *n* 的倍数的整数 *a*），有下面的公式成立。

$$a^{n-1} = 1 \pmod{n}$$

也就是说 *a* 的 *n*−1 次方除以 *n*，结果余 1。

表 3.10 取模 7 的 a^b（*a* 的 *b* 次方）

a \ *b*	1	2	3	4	5	6
1	1	1	1	1	1	1
2	2	4	1	2	4	1
3	3	2	6	4	5	1
4	4	2	1	4	2	1
5	5	4	6	2	3	1
6	6	1	6	1	6	1

✖ 数论之父费尔马

费尔马（Fermat，Pierre de Fermat，1601−1665），是17世纪具有代表性的法国数学家、法学家。在从取模运算开始的数论领域里，取得了巨大的成就。

他不仅提出了费尔马小定理，还提出了被称为费尔马大定理（最终定理）的数学理论。

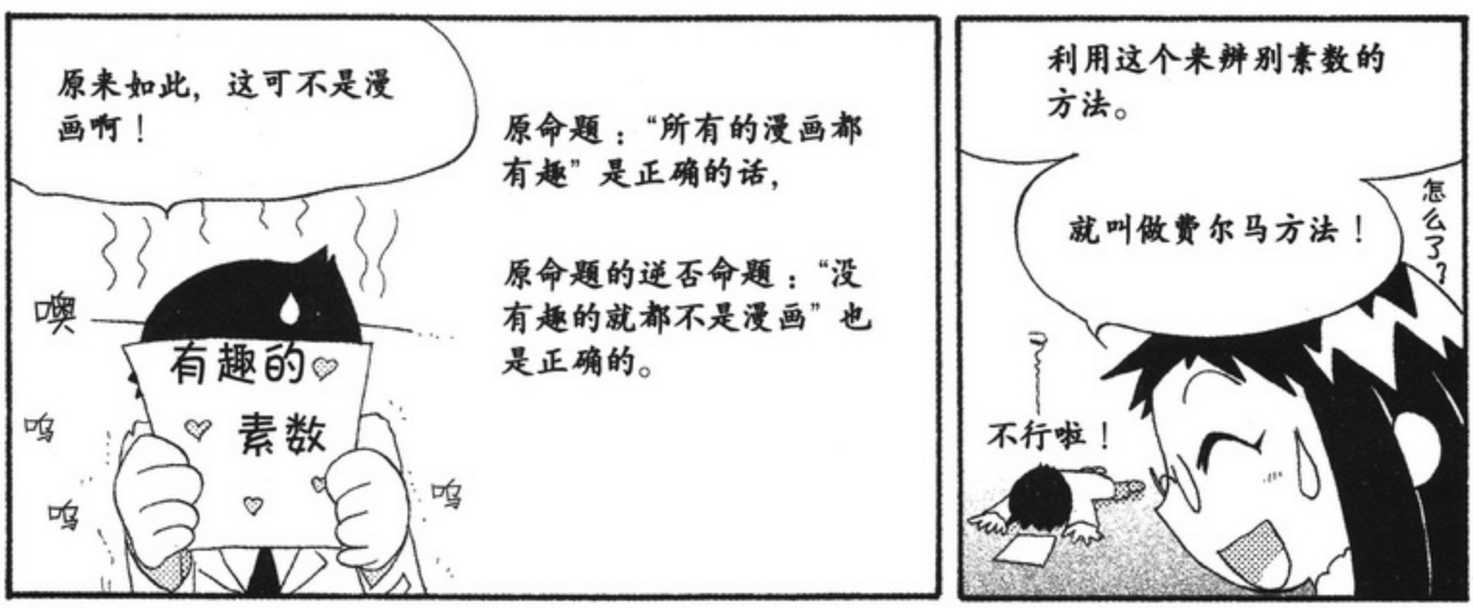
费尔马大定理是指“自然数 n 大于3的时候，没有 $x^n+y^n=z^n$ 这样的自然数（x，y，z）组合”。但是费尔马本人在生前也没有留下对这个定理的证明。

这是一个看起来内容很简单，仿佛连中学生都可以解开的、比较单纯的公式。众所周知的毕达哥拉斯定理（勾股定理）是指，“直角三角形的三边长 a，b，c，有 $a^2+b^2=c^2$ 的等式成立（例如三边长为3m，4m，5m的时候）”的理论。而费尔马的定理是将 $a^2+b^2=c^2$ 中的2替换为3以上的数字时的公式。

另外，费尔马大定理从费尔马去世后，经过了330年，在1995年才由英国的安德鲁・怀尔斯（1953—）成功地进行了证明。


辨别素数时就用费尔马小定理吧！（参看第 131 页）
取出费尔马小定理的逆否命题。即 n 和 a 互素的话，也就是，
$a^{n-1} \neq 1 \pmod{n}$
那么，n 就不是素数。


逆否命题是什么？
管 3 顿饭，还有午睡？
对原命题："如果 A 成立的话，B 就成立"来说，称"如果 B 不成立的话，A 就不成立"为原命题的逆否命题。
A 是 B
B 是 A
正确命题的逆否命题总是正确的哦！


原来如此，这可不是漫画啊！
噢
有趣的素数
呜
呜
呜
原命题："所有的漫画都有趣"是正确的话，
原命题的逆否命题："没有趣的就都不是漫画"也是正确的。
利用这个来辨别素数的方法。
就叫做费尔马方法！
怎么了？
不行啦！

✤ 费尔马方法和拟素数

用费尔马方法来判定素数的话，需要注意的是，当满足

$$a^{n-1} = 1 \pmod{n}$$

的时候，虽然 n 为素数是必要条件，但却不是充分条件。

为此，使用费尔马定理时，虽然不确定肯定是素数，但确定的可能性还是比较大的。像这样的数叫做拟素数。

比如说 n=3215031751，在与其互素的 2，3，5，7 之中

$$2^{3215031750} = 1 \pmod{3215031751}$$
$$3^{3215031750} = 1 \pmod{3215031751}$$
$$5^{3215031750} = 1 \pmod{3215031751}$$
$$7^{3215031750} = 1 \pmod{3215031751}$$

虽然都满足此式，但不是素数。因为

$$3215031751 = 151 \times 751 \times 28351$$

可以进行素因数分解。

但是，在 250 亿以下的数 n 中，2，3，5，7 这 4 个素数的 n–1 次方是 1，并且不是素数的只有 3215031751。在第 131 页介绍过的 Miller-Rabin 算法就是运用了费尔马方法，并提高了其精确度发展而来的。

✖ 欧拉定理

对于与自然数 n 互素的整数 a，下列公式成立。

$$a^{\varphi(n)} = 1 \pmod{n}$$

公式中的 $\varphi(n)$ 叫做欧拉函数。欧拉函数是，在从 1 到 n 的自然数中，与 n 互素的数字的数量表示。

接下来，因为 $a^{\varphi(n)} \times a = a^{\varphi(n)+1}$，所以下列公式也成立。

$$a^{\varphi(n)+1} = a \pmod{n}$$

为什么是这样呢？因为 $a^{\varphi(n)} = 1 \pmod{n}$，如果是 a 的 $[\varphi(n)+1]$ 次方，就会变回 a。

并且用幂次方的话，$2\varphi(n)$ 次方是 1，$[2\varphi(n)+1]$ 次方又会变回 a。把这部分用一般方法表示，对于与自然数 n 互素的整数 a，可以变为下列公式。

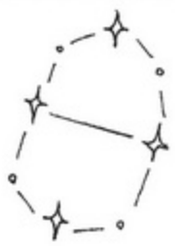
$$a^{k\varphi(n)} = 1 \pmod{n}$$

$$a^{k\varphi(n)+1} = a \pmod{n} \qquad (k \text{ 为非负整数})$$

另外，对于从 1 到 $(n-1)$ 之间的所有整数 a，下列公式成立。

$$a^{k\varphi(n)+1} = a \pmod{n} \quad \cdots\cdots (1)$$

✤ 数学家欧拉



莱昂哈德·欧拉（Leonhard Euler，1707－1783）是出生于瑞士的18世纪具有代表性的数学家。

他不仅在广阔的数学领域里有着伟大的成果，在物理学和天文学的领域里也尽显才华。

在数学界，他最为人们所熟知的就是欧拉公式（也表示为复素数的欧拉）。

$$e^{i\theta}=\cos\theta+i\sin\theta$$

这个公式是复素指数函数 $e^{i\theta}$ 和三角函数 $\cos\theta$ 与 $\sin\theta$ 中介入了虚数单位 $i=\sqrt{-1}$，结合而成的表示公式。

✤ 2个素数乘积的欧拉函数

设 N 为 2 个素数 p 和 q 的乘积。在这里为了用欧拉函数导出，请选出不与 N 互素的整数。因为 p 和 q 是素数，所以知道不与 N 互素的数是 p 的倍数和 q 的倍数。

（1）因为在从 1 到 qp 之间的 p 的倍数是 p，$2p$，$3p$，…，qp，所以有 q 个。
（2）因为在从 1 到 qp 之间的 q 的倍数是 q，$2q$，$3q$，…，qp，所以有 p 个。
（3）因为 qp 和 qp 都是 N，所以只有 1 是重复的相同的数。

也就是说，$\varphi(N)$ 是从 N（$=qp$ 个）中取出 p 个和 q 个后，再加上重复的 1 个所得到的数。即

$$\varphi(N) = pq-p-q+1=(p-1)(q-1)$$

欧拉函数 $\varphi(N)$ 变成 $(p-1)(q-1)$。

在这里 p 和 q 是素数的时候，可以用 $\varphi(pq)=\varphi(p)\varphi(q)$ 来表示。

还有就是，因为从 $a^{p-1}=1 \pmod p$，得到 $a^{p-1}=1 \pmod p$，所以将 $(p-1)$ 和 $(q-1)$ 的共同的倍数的最小的数（最小公倍数）设为 L，下列公式成立。

$$a^L = 1 \pmod p, \pmod p$$

即对于和 N（$=pq$）互素的整数 a，下列公式成立。

$$a^L = 1 \pmod N$$

也就是说，L 和欧拉函数 $\varphi(N)$ 起同样的作用。还有就是任意的两个正整数的乘积，

因为最小公倍数和最大公约数的乘积相等，所以，以（p–1）（q–1）=LG 的关系为基础，下列公式成立。

$$L = \frac{(p-1)(q-1)}{G}$$ （L 是最小公倍数，G 是最大公约数）

举例说明。例如，设 p=3，q=5。N=pq 为 15，（p–1）为 2，（q–1）为 4，$\varphi(N)$=（p–1）（q–1）是 8，2 和 4 的最小公倍数 L 为 4，最大公约数 G 为 2。这时候，对于与 15 互素的自然数 a，下列公式成立（表 3.11）。

$$a^{4k} = 1 \pmod{15}$$ （k 是非负整数）

也就是说，到 a 的 $\varphi(N)$ 次方为止，以（p–1）和（q–1）的最小公倍数 L 为周期，最大公约数的 G 次数，最少也会出现 1 次。

另外，从欧拉定理（参看第 158 页）的公式（1）中得出，从 1 到（N–1）中所有的整数对于 a 来说成为

$$a^{kL+1} = a \pmod{N} \quad \cdots\cdots (2)$$

这就成了制作 *RSA* 魔法密码的技巧。

表 3.11　2 个素数乘积的欧拉函数举例（a^b）
（N=3×5，$\varphi(15)$=8，L=4，G=2）

a \ b	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1
2	2	4	8	1	2	4	8	1
3	3	9	12	6	3	9	12	6
4	4	1	4	1	4	1	4	1
5	5	10	5	10	5	10	5	10
6	6	6	6	6	6	6	6	6
7	7	4	13	1	7	4	13	1
8	8	4	2	1	8	4	2	1
9	9	6	9	6	9	6	9	6
10	10	10	10	10	10	10	10	10
11	11	1	11	1	11	1	11	1
12	12	9	3	6	12	9	3	6
13	13	4	7	1	13	4	7	1
14	14	1	14	1	14	1	14	1
	1 周期				1 周期			

注：■■ 部分表示与公式（2）的关系。


欧拉也好，费尔马也好，我都讨厌……
哥哥……
没事的，没事的！“千里之行，始于足下”嘛！
慢慢来，你能行的！
也……也是啊。
对 对。
欧拉，费尔马
密码数学的基础就讲到这里啦！
啊，快看那个星座！
接下来，我们一起来学习公开钥匙密码RSA的实际运用吧！
哇——

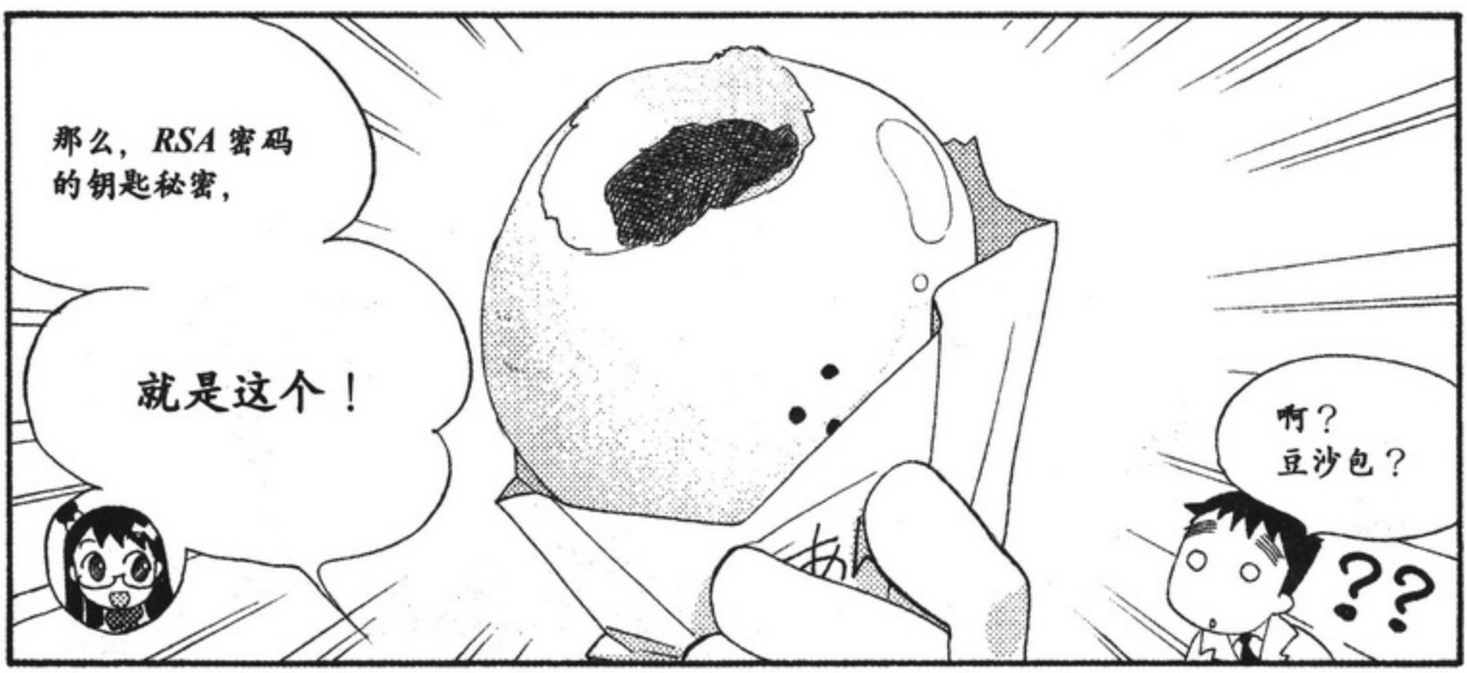
3-5 RSA密码的构成

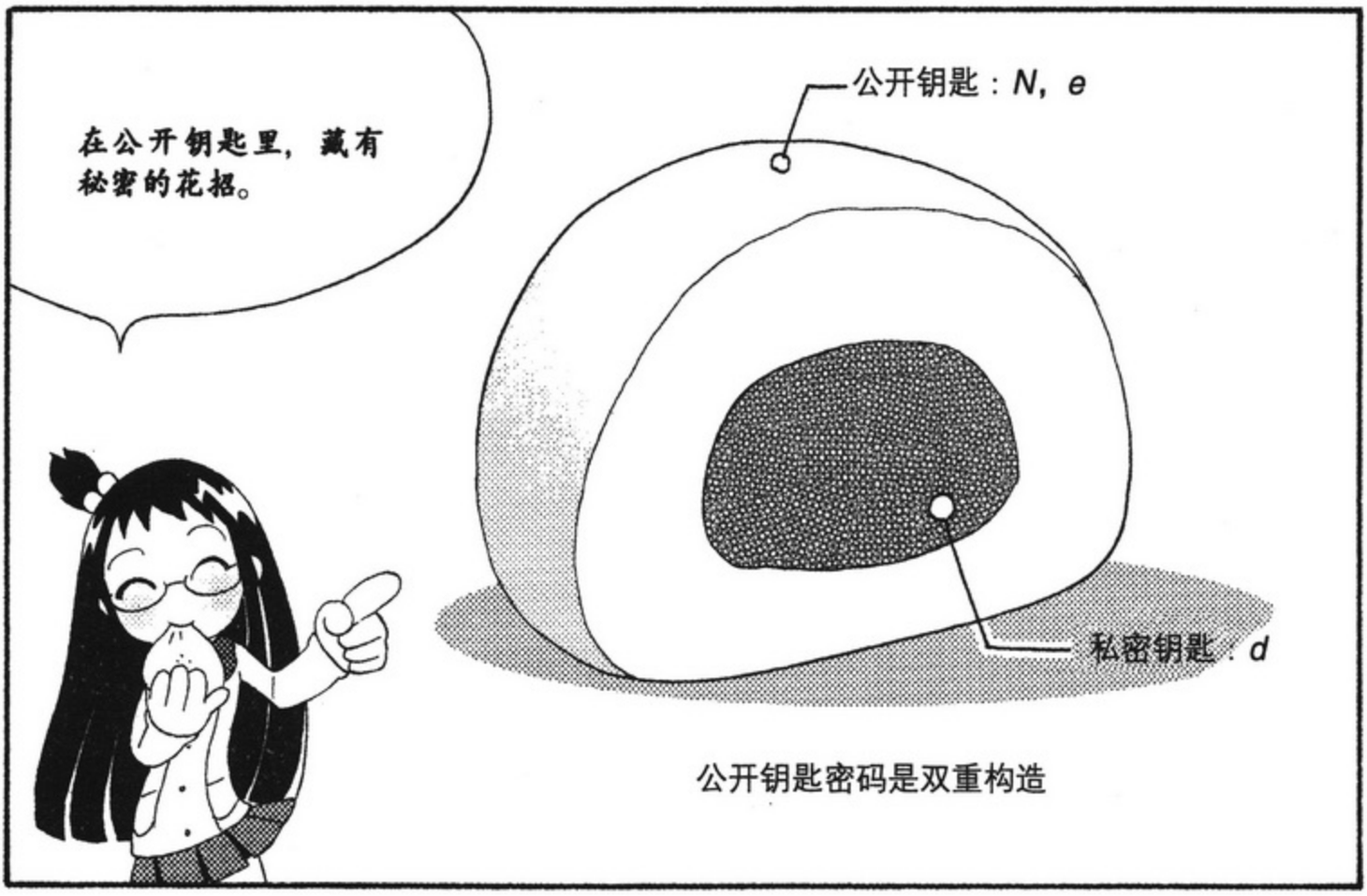

薯片
年糕
小豆汤
布丁
肉包
豆沙包
7-ELEVEN


终于从数学中解脱出来啦！
补充一下体力。


别高兴得太早！
数学知识还会出现哦！
啊——


轻松许多啊！


那么，RSA密码的钥匙秘密，
就是这个！
啊？豆沙包？
??



公开钥匙密码是双重构造

✤ RSA 密码的加密和解密

设明文为 P，密文为 C，加密用下列公式表示。

$$C = P^e \ (\mathrm{mod}\, N)$$

也就是说，把 P（明文）的 e（公开钥匙的 1 个）次方的值除以 N（另 1 个公开钥匙），余 C（密文），可以进行加密。

其次，解密用下列公式表示。

$$P = C^d \ (\mathrm{mod}\, N)$$

总之，就变为 C（密文）的 d（私密钥匙）次方除以 N（公开钥匙）余 P（明文），就可以进行解密。这里的 N 是指不同的 2 个巨大数值的素数相乘的组合。


如果N是个非常大的数，用素因数分解需要很长的时间，所以结论是“密码的安全强度＝素因数分解”。
你好好听哦！
只要明白了欧拉函数$\varphi(N)$，就一定能用欧拉定理计算出来！
如果$\varphi(N)$能整除的话，N就不能进行素因数分解了吧？
啊—
知道了x以外的e，C，N，就可以解开$x^e=C \pmod N$，也就有可能破解密文。
但是，将每个数值带入方程式来导出x的方法会需要很长的时间，实际上，几乎是不可能的！
这可以说是“安全计算量的密码”。


也就是说，素因数分解问题做为“难解的数学问题”解密是很难的哦！


公开钥匙中的加密钥匙和解密钥匙有着非常重要的作用。下面我们就按照顺序学习一下钥匙生成法吧！
好可怜哦……

✤ RSA 密码钥匙的生成法

① 任意选取足够大的不同的两个素数 p，q。

$p \times q$ 的乘积 N 是公开钥匙的 1 个。

② 求取欧拉函数 $\varphi(pq) = (p-1)(q-1)$。

③ 是为了做出公开钥匙的另一个准备操作。

③ 计算（p–1）和（q–1）的最小公倍数 L。

求欧拉函数的时候，就不需要 p 和 q 了，为了不让他人知道，就取消掉吧！

④ 因为要与（p–1）和（q–1）的最小公倍数 L 互素，所以选出比 L 小的任意正整数 e。

e 是另外一个公开钥匙哦！一定要满足 $P^e > N$ 的条件来选出 e 哦！

如果 $P^e \leqslant N$，不用取模运算介入，$P^e=C$ 的时候，就不能进行运算的自由变换。

⑤ 对于任意的正整数 e，求取可以满足下列公式的正整数 d。

$$ed = 1 (\mathrm{mod}\, L)$$

只是，要设定 d 为比 $\varphi(N)$ 小，比 p 或者 q 大的数。

求下面与加密钥匙 e 对应的解密钥匙 d。

因为 $ed=1 \pmod{L}$，所以，$eb-1=0 \pmod{L}$，也就是说，$ed-1$ 是 L 的倍数。

$$ed-1 = kL \qquad (k \text{是非负整数})$$

因此，下面的公式成立。

$$ed = kL+1 \qquad (k \text{是负整数})$$

因此，用欧拉函数来解释公式(2)，对于从 1 到($N-1$)所有的自然数 P(相当于明文)来说，下面的公式成立。

$$P^{ed} = P^{kL+1} = P \pmod{N}$$

综上所述，密文 C ($=P^e$) 的 d 次方 P^{ed} 就是明文 P 的解密。

✤ 公开钥匙和私密钥匙的制作方法

现在，设两个素数为 p=5，q=11 的时候，求取公开钥匙 N 和私密钥匙 d。

步骤 1

设 p 和 q 的乘积为 N。

$$N = pq = 5 \times 11 = 55$$

步骤 2

求取 N 的欧拉函数 $\varphi(N)$。

$$\varphi(55) = (5-1) \times (11-1) = 4 \times 10 = 40$$

步骤 3

求取（p–1）和（q–1）的最小公倍数。因为是求 4 和 10 的最小公倍数，所以 L=20。

步骤 4

求取和最小公倍数 L 互素的自然数 e。能成为和 L=20 互素的自然数 e 的数为 {1，3，7，9，11，13，17，19} 这 8 个数。

步骤 5

加密钥匙 e 的逆元 d，然后求取解密钥匙。

用 mod 20 的运算，例如，从对于 e=17 的相乘的逆元 d 来考虑。

因为 ed=kL+1（mod 20），所以，17d=20k+1。

然后将上个公式变形

$$d = \frac{(20k + 1)}{17}$$

由于右边是整数，所以需要找出 20k+1 为 17 倍数的数。k 为 11 的时候，

$20k+1=221$，所以可以知道是 17 的倍数。也就是说

可以从 $221=20\times11+1=17\times13$ 得出

$$17\times13 = 1 \pmod{20}$$

由上面两个公式的关系，得出 $d=13$。

从以上的结果，可以算出下列钥匙。

公开钥匙（$N=55$，$e=17$）←加密钥匙

私密钥匙（$d=13$）　　←解密钥匙

另外，使用相同的计算方法可以得到其他的 e 和 d 的组合为

$(e=1,\ d=1)$、$(e=3,\ d=7)$、$(e=7,\ d=3)$、$(e=9,\ d=9)$、
$(e=11,\ d=11)$、$(e=17,\ d=13)$、$(e=19,\ d=19)$

所有的组里都是 $ed=20k+1$。

一般来说，加密钥匙和解密钥匙是不同的整数（$e \neq d$），因为加密钥匙 e 要尽量选择大一些的数字，所以 $e=17$，$d=13$ 是很合适的。

✤RSA 密文的生成

首先，使用 RSA 密码的公开钥匙来对密文生成的程序进行说明。

举个具体的例子，以加密钥匙 e=17 为例，把字母表的 4 个文字变成明文（GOLF）进行加密。

表 3.12　文字代码表

文字	代码	文字	代码	文字	代码
a	0	s	18	K	36
b	1	t	19	L	37
c	2	u	20	M	38
d	3	v	21	N	39
e	4	w	22	O	40
f	5	x	23	P	41
g	6	y	24	Q	42
h	7	z	25	R	43
i	8	A	26	S	44
j	9	B	27	T	45
k	10	C	28	U	46
l	11	D	29	V	47
m	12	E	30	W	48
n	13	F	31	X	49
o	14	G	32	Y	50
p	15	H	33	Z	51
q	16	I	34	⋮	⋮
r	17	J	35	空白	63

步骤 1

首先，以文字代码表为基础，选出对应的整数。

G → 32　　O → 40　　L → 37　　F → 31

步骤 2

将整数变为二进制数据的形式。

32 → 100000　　40 → 101000　　37 → 100101　　31 → 011111

步骤 3

将二进制数据用（N–1）以下的非负整数表示。在这个例子中，因为 N=55，所以 N–1=55–1=54，将 5bit 进行分区。也就是说，可以表示 5bit 的最大值是 31，在 54 以下，可以满足条件。当然，3bit 和 4bit 也可以，因为具有分区 bit 数越大加密效率就越高的性质。

100000　101000　100101　011111　　将 0 补上

10000　01010　00100　10101　11110

（最后的 0 是将 5bit 分区的时候不足的 bit，在这里为了方便所以设 0。）

步骤 4

将二进制数据更改为十进制。

10000 → 16　　01010 → 10　　00100 → 4　　10101 → 21　　11110 → 30

步骤 5

使用加密钥匙（N=55，e=17）进行加密。具体来说，就是求取十进制的数据的 17 次方，除以 55 的余数。因此，从

$16^{17}\ (\mathrm{mod}\ 55)$，$10^{17}\ (\mathrm{mod}\ 55)$，$4^{17}\ (\mathrm{mod}\ 55)$，$21^{17}\ (\mathrm{mod}\ 55)$，$30^{17}\ (\mathrm{mod}\ 55)$

的计算中可以得到密码数据。例如，通过对 16 的运算

$$16^2 = 256 = 36\ (\mathrm{mod}\ 55) \qquad 36^2 = 1296 = 31\ (\mathrm{mod}\ 55)$$
$$31^2 = 961 = 26\ (\mathrm{mod}\ 55) \qquad 26^2 = 676 = 16\ (\mathrm{mod}\ 55)$$

将上述结果，顺序代入下面的关系式：

$$\begin{aligned}
16^{17} &= 16^2 \times 16^2 \times 16^2 \times 16^2 \times 16^2 \times 16^2 \times 16^2 \times 16^2 \times 16 \\
&= 36 \times 36 \times 36 \times 36 \times 36 \times 36 \times 36 \times 36 \times 16 \\
&= 36^2 \times 36^2 \times 36^2 \times 36^2 \times 16 \\
&= 31 \times 31 \times 31 \times 31 \times 16 \\
&= 31^2 \times 31^2 \times 16 \\
&= 26 \times 26 \times 16 \\
&= 26^2 \times 16 \\
&= 16 \times 16 \\
&= 36
\end{aligned}$$

剩下的加密也可以用同样的方法进行运算（只记入结果）。

$$10^{17}\ (\mathrm{mod}\ 55) = 10 \qquad 4^{17}\ (\mathrm{mod}\ 55) = 49$$
$$21^{17}\ (\mathrm{mod}\ 55) = 21 \qquad 30^{17}\ (\mathrm{mod}\ 55) = 35$$

得到的密文为

$$36 \quad 10 \quad 49 \quad 21 \quad 35 \cdots\cdots (3)$$

将公式（3）作为文字代码，按照表 3.12 找出对应的文字，用十进制来表示：

36	10	49	21	35
↓	↓	↓	↓	↓
K	k	X	v	J

这样密文就做成了。

✤ RSA 密文的解密

这次，使用 RSA 密码的私密钥匙，对将密文解密为明文的处理程序进行说明。

举个具体的实例，使用作为私密钥匙的解密钥匙（d=13），将十进制的密码数据公式（3）解密为字母的明文，表示如下。

步骤 1

使用解密钥匙（d=13），计算 C^d（mod N）。具体来说，就是求取公式（3）的十进制数据的 13 次方除以 55 的余数，得出明文。因此，从

$$36^{13}\ (\mathrm{mod}\ 55),\quad 10^{13}\ (\mathrm{mod}\ 55),\quad 49^{13}\ (\mathrm{mod}\ 55),\quad 21^{13}\ (\mathrm{mod}\ 55),\quad 35^{13}\ (\mathrm{mod}\ 55)$$

的计算中可以得到明文数据。例如，通过 36 的计算，利用生成密文的步骤 5，可以得出

$$\begin{aligned}36^{13} &= 36^2 \times 36^2 \times 36^2 \times 36^2 \times 36^2 \times 36^2 \times 36\\ &= 31 \times 31 \times 31 \times 31 \times 31 \times 31 \times 36\\ &= 26 \times 26 \times 26 \times 36\\ &= 26^2 \times 36\\ &= 16 \times 26 \times 36\\ &= 14976\ (\mathrm{mod}\ 55)\\ &= 16\end{aligned}$$

剩下的密码数据 {10，49，21，35} 也可以用同样的方法进行计算（只记入结果）。

$$10^{13}(\mathrm{mod}\ 55) = 10 \qquad 49^{13}\ (\mathrm{mod}\ 55) = 4$$
$$21^{13}(\mathrm{mod}\ 55) = 21 \qquad 35^{13}\ (\mathrm{mod}55) = 30$$

这样，明文数据就变成了下列十进制。

16 10 4 21 30

步骤 2

将从明文数据得到的十进制改变成 5bit 的二进制。

16	10	4	21	30
↓	↓	↓	↓	↓
10000	01010	00100	10101	11110

步骤 3

为了对应表里的文字代码，将二进制数据改变为 6bit 并进行分区。

10000 01010 00100 10101 11110

↓↓↓↓↓↙ ↓↓↓↓↙↙ ↓↓↓↓↙↙ ↓↓↓↙↙↙ ↓↓↙↙↙↙ ↘ 将 0 去除

100000 101000 100101 011111

（最后的 0 是 6bit 分区的时候多余的 bit，所以将其删掉。）

步骤 4

将 6bit 的二进制数据变为整数。

100000	101000	100101	011111
↓	↓	↓	↓
32	40	37	31

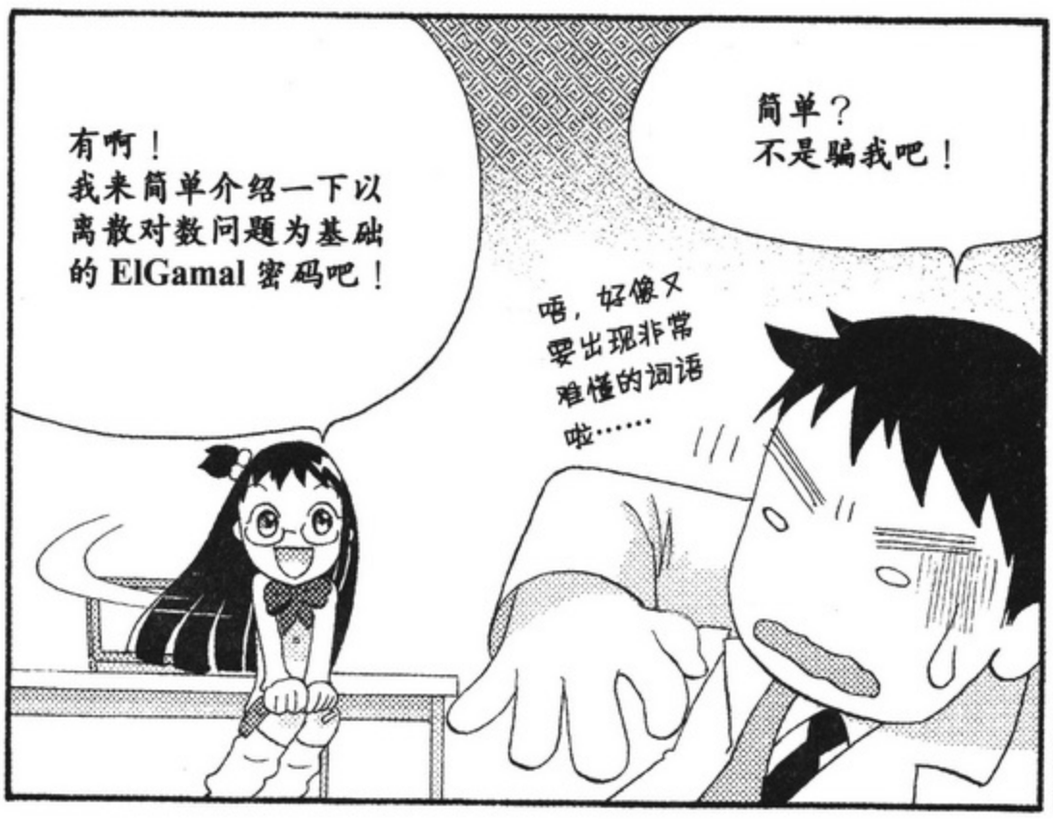
步骤 5

以表的文字代码为基础，将整数数据替换成文字。

32	40	37	31
↓	↓	↓	↓
G	O	L	F

这样，解密就完成了。

3-6 公开钥匙密码和离散对数问题



❀ 离散对数问题

请再来看一下模 7 的幂次方表。

表 3.13　模 7 的 a^b（a 的 b 次方）

a \ b	1	2	3	4	5	6
1	1	1	1	1	1	1
2	2	4	1	2	4	1
3	3	2	6	4	5	1
4	4	2	1	4	2	1
5	5	4	6	2	3	1
6	6	1	6	1	6	1

3 的幂次方那一行中，从 1 到 6 的数值没有重复，每个只出现了一次。

素数 7 的取模运算的结果是有限的，要素是

$$\{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6\}$$

但是，3 的幂次方则变成了除了 0 以外所有的都是可能的。以表 3.12 的 a=3 为例，具有从 1 到 6 的数值没有重复并且 1 个只出现 1 次的性质的数称作原根。

设素数 p 为模，原根是一定存在的，个数为 $\varphi(p-1)$。在模 7 的情况下

$$\varphi(7-1)=\varphi(6)=\varphi(2\times3)=(2-1)\times(3-1)=2$$

所以是两个。这样的话，除了 3 以外，一定还有另一个原根的数值存在。从表 3.12 来看，可以知道 5 也是原根。取素数 p 的模，设原根为 α，取模运算的任意要素 Z_i 用下列公式表示。

$$\alpha^k = Z_i \pmod p$$

（k 为非负整数，并且 $k \leqslant p-1$）

还有就是，原根 α 的幂指数 k 用下列公式表示。

$$k = \log_\alpha Z_i \pmod p$$

这个时候，把 k 称为以 α 为底的离散对数。

在这里，log 这个符号，其实并不难理解。

例如，$2^3=8$ 的公式和

$$3 = 3\log_2 2 = \log_2 2^3 = \log_2 8$$

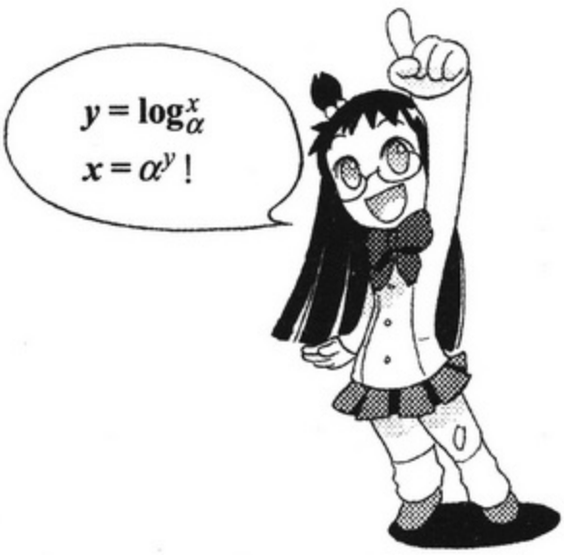
是完全相同的意思。

“2 的 3 次方等于 8”，换句话说就是“2 这个数相乘 3 次就可以得出 8 这个数”，两句话是同一个意思。

例如，在 118 页解释过的

$$\alpha^k = Z_i \ (\mathrm{mod}\, p)$$

如果知道了α和 k 和 p，求 Z_i 就很容易了。但是如果知道α和 Z_i 和 p，反过来求离散对数的 k 的话，就非常困难。这就是离散对数问题。



✤ ElGamal 密码的加密和解密

把密码的发送人设为“留香”，接收人设为“小兰”。

① 接收人小兰准备了巨大素数 q 和它的原根 α。

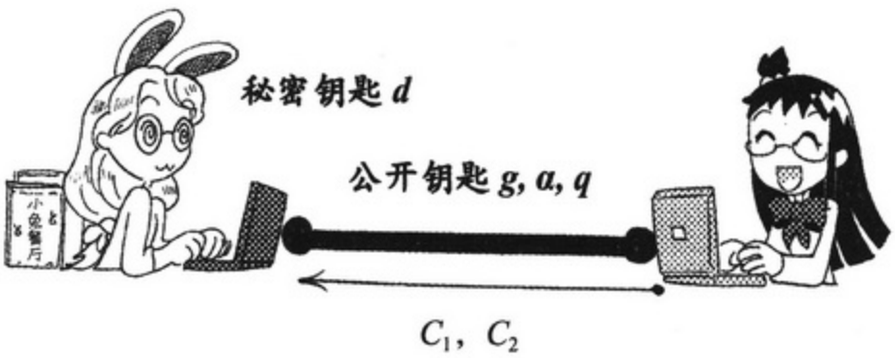
② 接收人小兰决定，随机使用私密钥匙 d

$$g = \alpha^d \pmod{q}$$

计算后，将 g 和 α 和 q 作为公开钥匙进行公开。

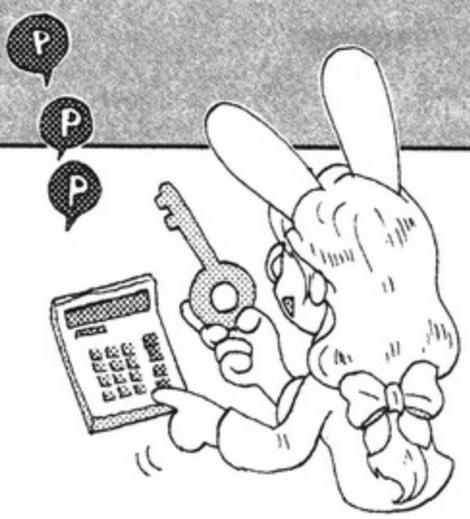
③ 发送人留香选出了随机数 r，将 $C_1 = \alpha^r \pmod{q}$ 进行计算。
并且，对于明文 P，对 $C_2 = P \times g^r \pmod{q}$ 进行计算。

④ 发送人留香将 C_1 和 C_2 向小兰进行发送。



⑤ 接收人小兰使用私密钥匙 d 计算下列公式，进行解密。

$$P = \frac{C_2}{C_1^{\,d}} \pmod q$$



是 $C_1^{\,d}=(\alpha^r)^d=(\alpha^d)^r=\alpha^{rd}=g^r$ 吧！
也就是说，

$$\frac{C_2}{C_1^{\,d}}=\frac{P\times g^r}{g^r}=P$$

总之就是解密 P。

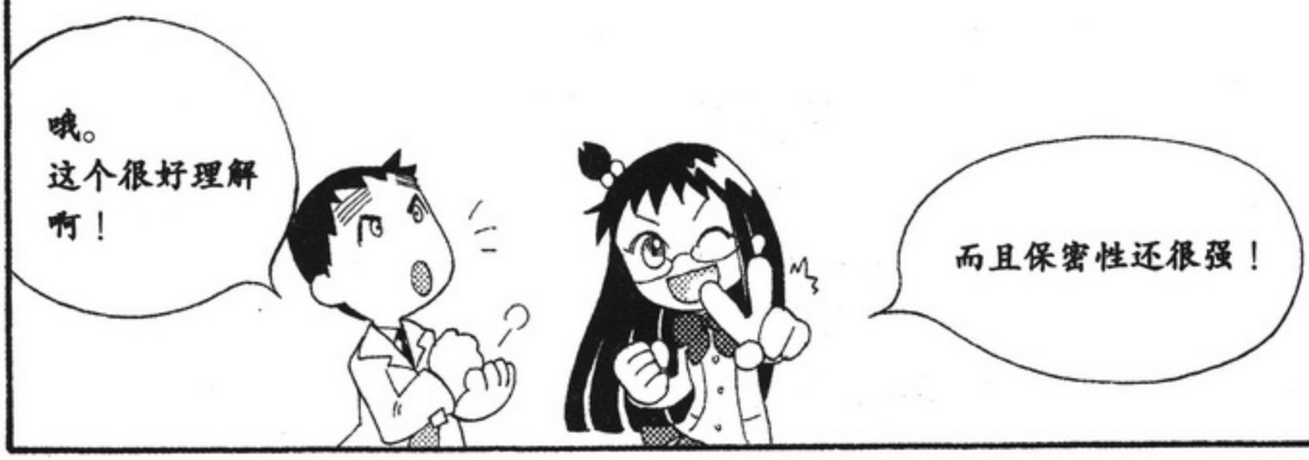
① 大的素数 p 和原始根 α 没有保密，是留香和小兰两个人共有。

▼

② 留香随机选择了数 c 进行保密，将 $\alpha^c \pmod p$ 发送给了小兰。另一边兰随机选择了数 d 进行保密，将 $\alpha^d \pmod p$ 发送给了留香。

▼

③ 留香从私密钥匙 c 中得到了钥匙 $(\alpha^d)^c = \alpha^{cd} \pmod p$。小兰从私密码钥匙 d 中得到了钥匙 $(\alpha^c)^d = \alpha^{cd} \pmod p$。
两个人都可以使用这把钥匙。



发送人：卯月兰
接收人：留香
主题：bye bye!

今天上的密码课也很辛苦吧?

目前我辞掉了店里的工作，来到了国外!

虽然很想和你好好聊聊……等我有时间再给你发电子邮件吧！虽然我们分开了……但我们永远都是好朋友！（⌒0⌒）


……
这么要好的朋友离开了……好难过啊！
我们先去啦！
好啊！
我当然也很难过啊！但是与此相比我更饿啊！
去吃拉面吧！
咔嗒
大家已经明白了吧？

✖专栏✖ 扩展的欧几里得辗转相除法

欧几里得辗转相除法是针对出两个自然数的最大公约数而设置的算法，也称欧几里得算法。进行计算时，比素因数分解的效率要高。使用欧几里得除法来找出自然数 a，b（$a > b$）的最大公约数时，使用以下的程序。

① 设 a 除以 b 余 r。
② 如果 r=0 的时候，当最大公约数是 b，程序结束。
③ 如果 $r \neq 0$ 的时候，a 和 b 的组合将会替换 b 和 r，返回到最初的程序。

总之，反复操作从①到③的程序，直到余数变为 0 的时候，对应的数就是最大公约数。换言之，得到余数是 0 的时候，在上一个步骤得到的余数就是最大公约数。

举个例子，用欧几里得除法求 1365 和 77 的最大公约数。

$1365 = 17 \times 77 + 56$	（通过 1365 ÷ 77 = 17 余 56 可以得出）
$77 = 1 \times 56 + 21$	（通过 77 ÷ 56=1 余 21 可以得出）
$56 = 2 \times 21 + 14$	（通过 56 ÷ 21=2 余 14 可以得出）
$21 = 1 \times 14 + ⑦$	（通过 21 ÷ 14=1 余 7 可以得出）
$14 = 2 \times ⑦ + 0$	（通过 14 ÷ 7=2 余 0 可以得出）

所以，最大公约数就是 7。因为按照顺序进行计算的话，一定能得出确切的结果，所以欧几里得除法是非常实用的。

●计算一次不定方程的解●

下面，将互素的数 20 和 17，用欧几里得算法求一下最大公约数。

$20 = 1 \times 17 + 3$ ……（1）
$17 = 5 \times 3 + 2$ ……（2）
$3 = 1 \times 2 + 1$ ……（3）
$2 = 2 \times 1 + 0$

最大公约数理所应当是 1，好像在这里失去了使用欧几里得算法的必要。但是，求结果的过程式有着很重要的利用价值。

首先将公式（1）、（2）、（3）移项，可以得到以下的 3 个公式。

$$20-1\times17=3 \cdots\cdots (4)$$
$$17-5\times3=② \cdots\cdots (5)$$
$$3-1\times②=1 \cdots\cdots (6)$$

接下来将公式（6）中的②代入到公式（5）中，注意将 3 和 17 括起来。

$$3-1\times②=3-1\times(17-5\times3)=6\times\boxed{3}-1\times17=1 \cdots\cdots (7)$$

然后将公式（7）的[3]代入到公式（4）中，注意将 20 和 17 括起来。

$$6\times\boxed{3}-1\times17=6\times(20-1\times17)-1\times17=6\times20-7\times17=1$$

将根据以上的步骤得到的结果，写成下面的公式。

$$20\times6+17\times(-7)=1$$

上面的公式变成了 $ax+by=c$ 的形式，能够成为 a，b，c，x，y 的数是所有的整数。像这样的方程式叫做一次不定方程式，求整数解 x 和 y。

也就是说，是运用了欧几里得算法的运算过程，所以 a=20，b=17 的时候，就可以得出一次不定方程式的整数解（x，y）=（6，–7）。这种方法是扩展欧几里得算法，是具有非常高的使用价值的算法。

一般来说，a 和 b 是不为 0 的整数，a 和 b 的最大公约数为 c 的时候，一次不定方程式

$$ax+by=c$$

有整数解（x_1，y_1），解的 1 组可以用扩展欧几里得算法来求取。但是，一次不定方程式的解不只有 1 组。方程式所有的整数解用任意的整数 k，表示为下列公式：

$$(x,\ y)=\left(x_1+k\cdot\frac{b}{c},\ y_1-k\cdot\frac{a}{c}\right) \cdots\cdots (8)$$

●用取模运算计算逆元●

使用公式（8）表示解的公式，一次不定方程式 $20x+17y=1$ 的全部整数解表示如下。

$$(6+17k,\ -7-20k) \cdots\cdots (9)$$

k=1 的时候，解是$(x,y)=(-11,13)$，将这个解代入到一次不定方程式 $20x+17y=1$ 中。

$$20\times(-11)+17\times 13=1$$

进行移项，调整公式。

$$17\times 13=1+11\times 20 \cdots\cdots (10)$$

仔细来看公式 10，实际上和下面的公式是同一个意思。

$$17\times 13=1 \pmod{20} \cdots\cdots (11)$$

在 168 页解释过了，当 $ed=1 \pmod{L}$ 的时候，“解密钥匙 d 就是加密钥匙 e 关于模 L 的乘法逆元”。即公式（11）中是指 13 是 17 关于 20 的乘法逆元的意思。

总之，只要使用扩展欧几里得算法就可以快速导出取模运算的逆元。因为对于公开钥匙密码来说，为了生成私密钥匙（解密钥匙），求取逆元是非常重要的，所以在密码的世界里，扩展欧几里得算法也发挥了重大的作用。

虽然可以求得 $17 \pmod{20}$ 的逆元是 17^{-1}，但是在 $16 \pmod{20}$ 的逆元是 16^{-1} 的时候会怎样呢？因为 16 和 20 的最大公约数是 4，所以 $20x+16y=4$ 也可以用前面的方法求出解。然而，为了求逆元的一次不定方程式 $20x+16y=1$，左边一定是 4 的倍数，所以正整数的解不存在。也就是说，如果两个数不互素，就无法求取逆元。使用扩展欧几里得算法来导出逆元，只能在这两个数互素的情况下进行。

最后，求 73 关于模 1001 的逆元 73^{-1} 的计算，使用欧几里得除法实际的计算一下。首先，用欧几里得算法求 73 和 1001 的最大公约数。

$$1001=13\times 73+52$$
$$73=1\times 52+21$$
$$52=2\times 21+10$$
$$21=2\times 10+1$$
$$10=10\times 1+0$$

得到了 73 和 1001 的最大公约数是 1，也就是说 73 和 1001 互素。

接下来，将这个公式改写成求余的公式

$$1001-13\times 73 = 52 \cdots\cdots (12)$$
$$73-1\times 52 = 21 \cdots\cdots (13)$$
$$52-2\times 21 = 10 \cdots\cdots (14)$$
$$21-2\times 10 = 1 \cdots\cdots (15)$$

将公式（15）中的 10 代入公式（14）。

$$21-2\times (52-2\times 21) = 1$$
$$21-2\times 52+4\times 21 = 1 \cdots\cdots (16)$$

在公式（16）中提取 52 和 21。

$$5\times 21-2\times 52 = 1 \cdots\cdots (17)$$

将公式（17）中的 21 代入公式（13）。

$$5\times (73-1\times 52)-2\times 52 = 1$$
$$5\times 73-5\times 52-2\times 52 = 1 \cdots\cdots (18)$$

在公式（18）中提取 73 和 52。

$$5\times 73-7\times 52 = 1 \cdots\cdots (19)$$

将公式（19）中的 52 代入公式（12）。

$$5\times 73-7\times (1001-13\times 73) = 1$$
$$5\times 73-7\times 1001+91\times 73 = 1 \cdots\cdots (20)$$

在公式（20）中提取 1001 和 73。

$$96\times 73-7\times 1001 = 1 \cdots\cdots (21)$$

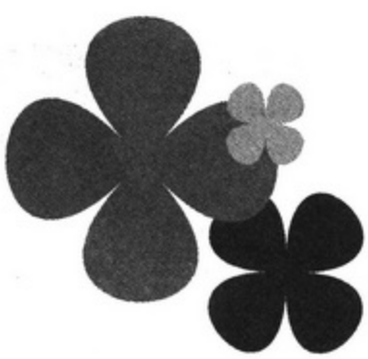
将公式（21）移项。

$$96\times 73 = 1\times 7\times 1001$$

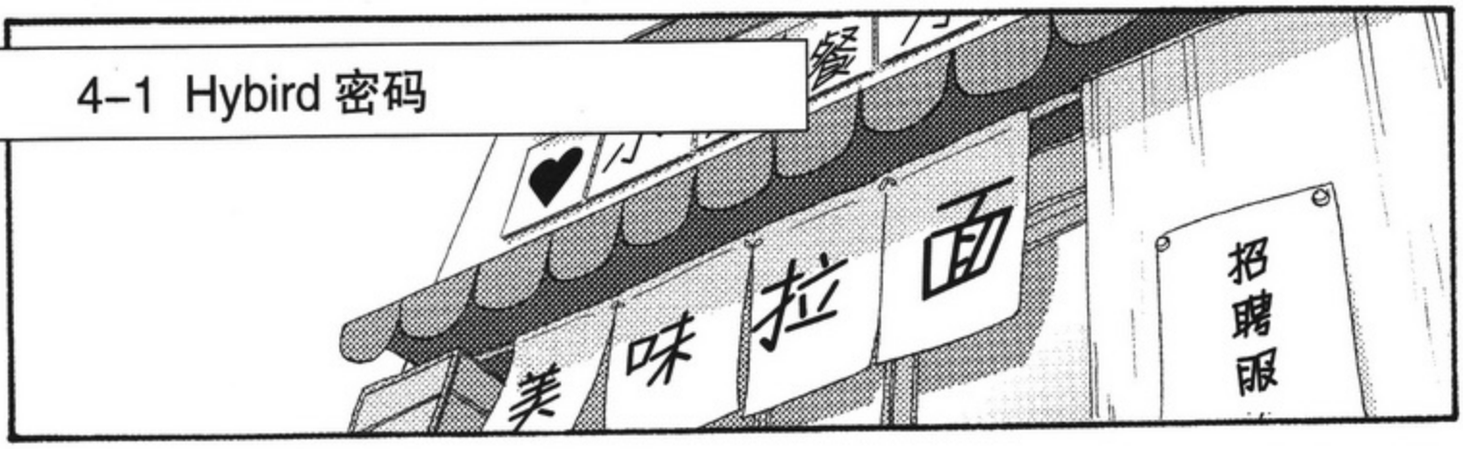
这就和 $96\times 73=1 \pmod{1001}$ 是同一个意思，所以 73 关于模 1001 的逆元 73^{-1} 是 96。

◆第4章◆

密码的实际应用



4-1 Hybird 密码


美味拉面
招聘服


招聘服务员！
急招！！


拉面还没好啊……
小兰辞掉了店里的工作，所以很忙吧……
我是小兰


既然这样，利用这段时间学学 Hybrid 密码吧！


Hybrid 就是混合的意思吧！也就是说，多个算法组合而成的，对吗？
又要学习啊……

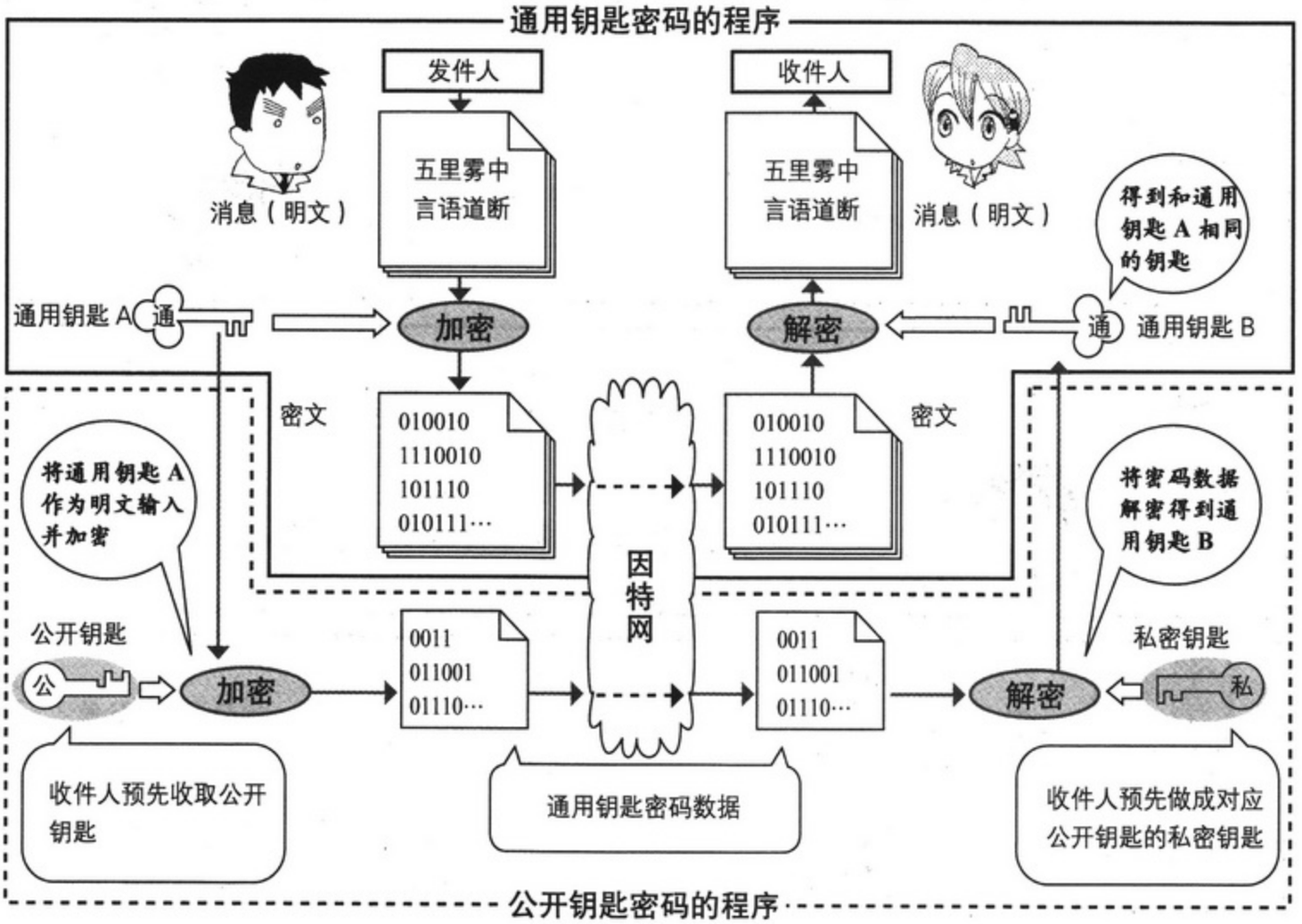


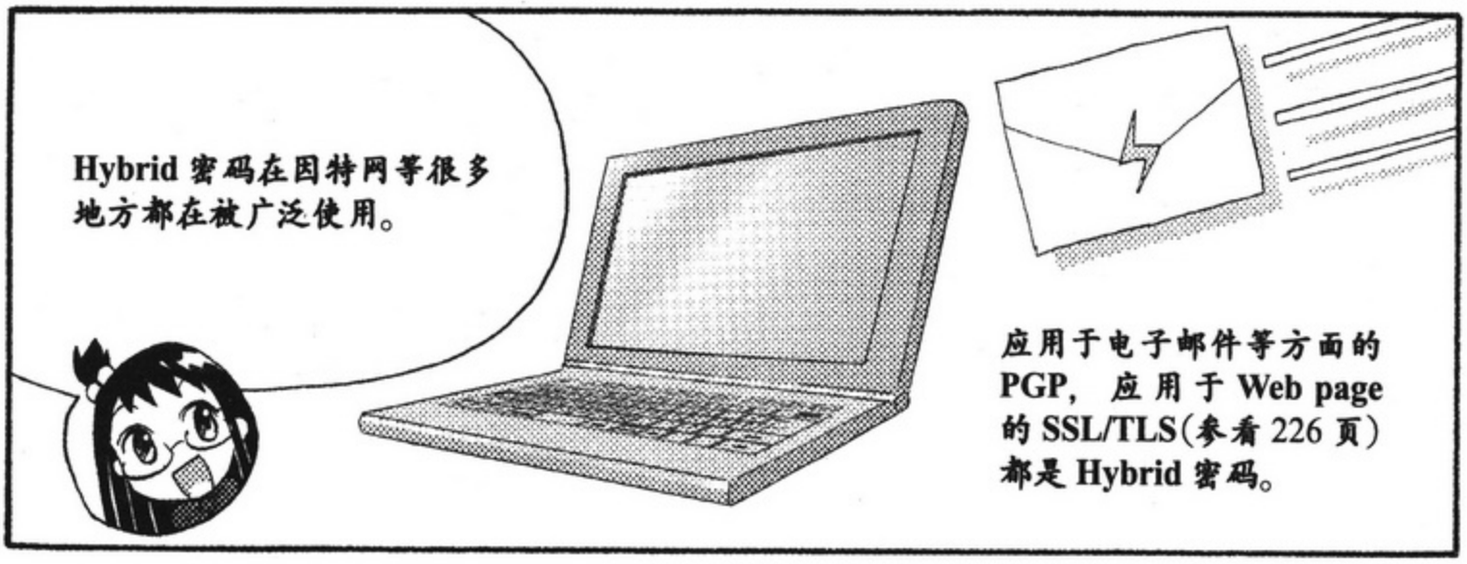
图 4.1　Hybrid 密码的加密和解密的流程

如图 4.1 所示，运用了公开钥匙密码作为“通用钥匙的加密和解密的保障”，通用钥匙密码作为“消息的加密和解密的保障”的方法。也就是说，较长的消息，用通用钥匙密码进行高速的加密和解密。通用钥匙用公开钥匙进行加密来实现安全的通信，从而避免了通用钥匙最大的缺点，也就是交换钥匙时会出现问题的情况发生。

接下来是 Hybrid 密码的实际应用，就以拉面外卖为例来研究一下吧！


原来如此，这样解密的效率就提高了呀！


拉面的效率最好也快一点啊！
饿死我啦——


Hybrid 密码在因特网等很多地方都在被广泛使用。
应用于电子邮件等方面的 PGP，应用于 Web page 的 SSL/TLS（参看 226 页）都是 Hybrid 密码。


让您久等啦！
咚——

4-2 Hash 函数和消息认证代码

✤篡改


在这之前我损失了29份拉面！
拉面？
小兔餐厅
收到了电子邮件发来的外卖订单……
第一街区的佐藤先生
小兔餐厅
小兔餐厅
美味拉面
请送1份拉面。
拉面30份，急送！
什么—！
攻击者
我来改一下！


消息被篡改啦！
呀——
好可怕
没有用密码来防止消息的篡改吗？
啊……

✤篡改的对策


防止被篡改，就要使用Hash函数啦！
hash
Hash？
和红烧牛柳有关吗？
喂，喂！
是将牛肉切细烧制的菜肴吧？
Hash就是切细的意思哦！
把牛肉换成消息来细切，做出的Hash值就是Hash函数啦！
哦，
原来如此——
好像很好吃啊！
Hash值是什么？
不能吃吗？
就是从消息中计算出来的值，类似于在犯罪调查中使用的“指纹”一样的东西！
当然不能吃啦！
是用来确保消息不会被篡改时使用的！

Hash 函数

使用 Hash 函数可以算出原消息的 Hash 值。指纹是一种用来确认身份的有效手段。所以，Hash 值也可以称为消息的指纹。Hash 函数把消息压缩成摘要，使数据量变小，固定了格式的大小。

“接收人确认消息是否被篡改”的性质叫做真实性（也称为完整性，integrity），发件人将原消息和 Hash 值一起发送的话，真实性才会得到保证。也就是说，以消息的指纹为线索可以检查出消息有没有被篡改。收件人和发件人使用同样的 Hash 值，计算出作为消息指纹的 Hash 值，与被添付的 Hash 值进行比较。如果 Hash 值相同，就说明消息没有被篡改。

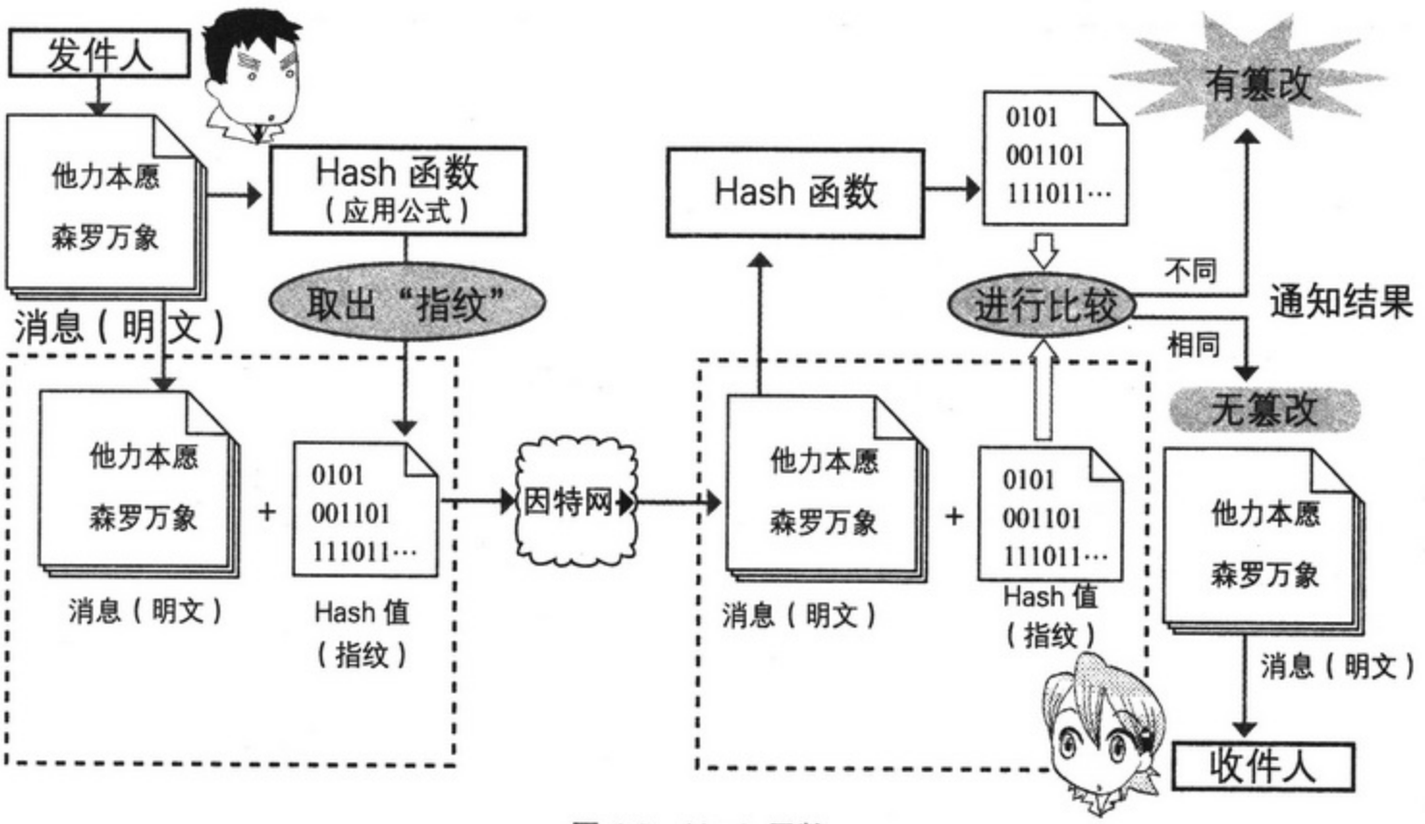


图 4.2　Hash 函数

Hash 函数是单向函数。即使从消息中计算出了 Hash 值，相反将 Hash 值还原到消息是不会发生的。这个性质叫做不可逆性。具有这个性质的 Hash 函数叫做单向 Hash 函数。

接着上面说，Hash 值一致，但是内容却不同，比较这种消息的组合很难发现不同，这种情况叫做强冲突耐性。将某个消息和另一个 Hash 值相等的消息作比较，不难发现不同，这种情况叫做弱冲突耐性。为了处理这些冲突而研发的 Hash 函数有 MD5、SHA-1、SHA-256、SHA-512、RIPEMD-160 等。

✖冒名诈骗


使用 Hash 函数来防止篡改消息不就行了？
原来如此……
但是，

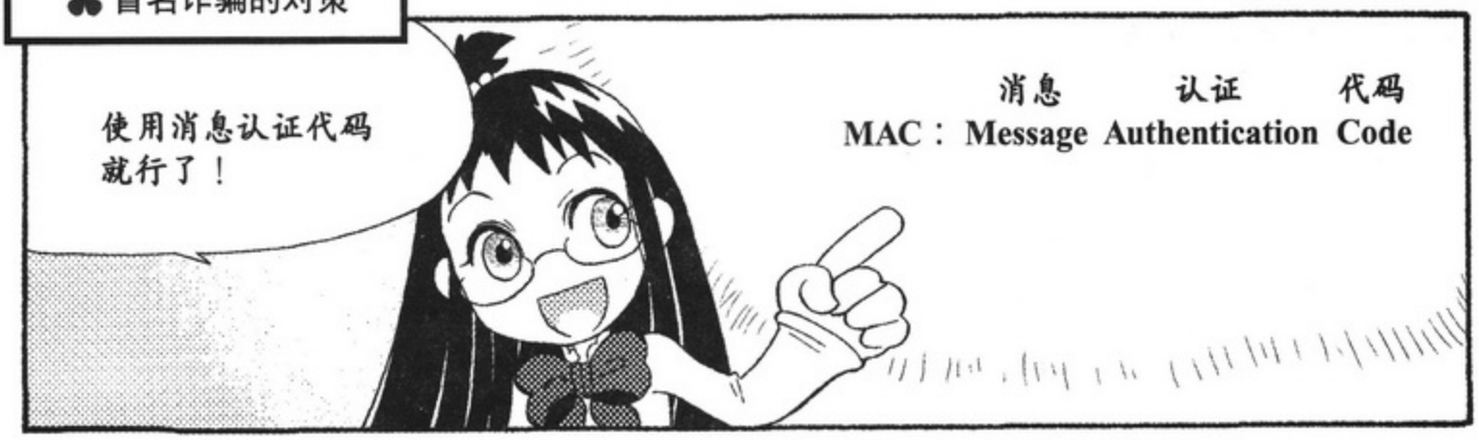

用了这个也不行啊！


今天我又损失了30份拉面！
攻击者
我来搞搞恶作剧吧！
啊——
小兔餐厅
小兔餐厅
美味拉面
我是1街区的神谷，给我10份拉面的外卖！
送5份拉面到2街区的佐藤，急送！
4街区的田中，
送3份拉面
来！
3街区的铃木，请给我7份拉面！
5街区的笹，5份拉面，拜托！


到底是谁在冒充客人！
哇……
嘣
嘣
是为了这个发的火啊……


没有防备冒名诈骗的方法吗？

✤ 冒名诈骗的对策


使用消息认证代码就行了！
消息　　认证　　代码
MAC：Message Authentication Code


消息的认证？
是什么？


从真正的发件人那里发出的消息……
2份拉面！
我是1街区的佐藤，送1份拉面来！
3街区的中井，给我送来3份面吧！
6街区的秋田，送1份拉面来！
2街区的田中，拉面
4街
认证
在收到的消息中对于冒名诈骗，到底是从非法发件人那里发出的消息，
还是从真正的发件人那里发出的消息，可以进行确认。


哦！
是怎样进行确认的呢？

✤ 消息认证代码的构成

确认消息的真实性，这种进行认证的程序就叫做消息认证代码。我们一边看图 4.3，一边来学习消息认证代码的构成。

发件人将想要发送的消息和从那个消息中生成的 MAC 值一起进行发送。MAC 值是指和 Hash 值一样，有检查作用的值。

收件人将收到的 MAC 值同从消息中生成的 MAC 值进行比较，起到了确保消息真实性的作用。在这时，发件方和收件方为了 MAC 值的生成而使用通用钥匙。

两个 MAC 值相同的时候，就可以确认，从发送人那里发出的消息在途中没有被篡改（真实性），发件人确实是拥有相同钥匙，真正的发件人（认证）。

两个 MAC 值不同的时候，就可以确定，从发件人那里发出的消息在途中已经被篡改，发件人不是拥有相同钥匙的那个真正的发件人。

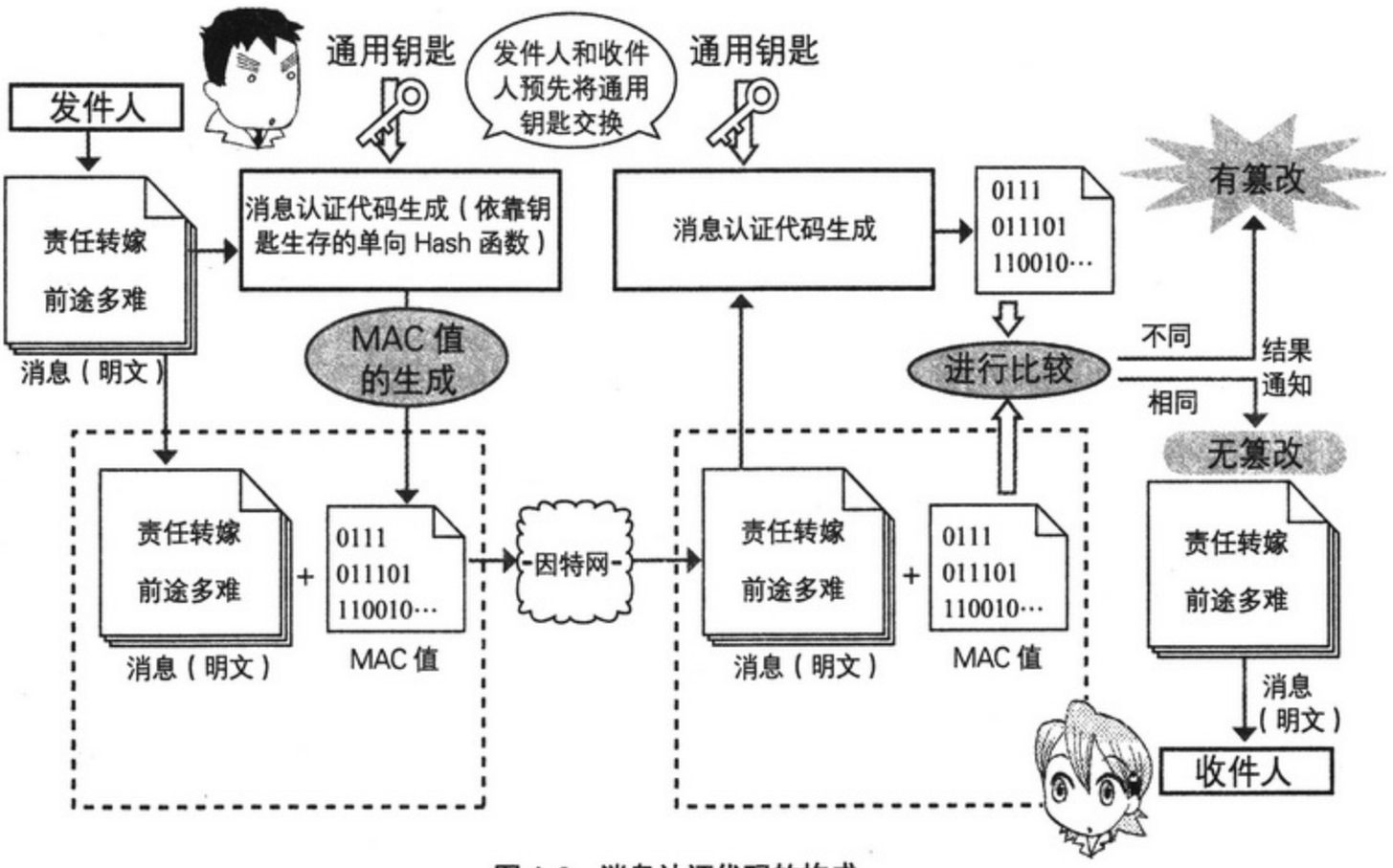


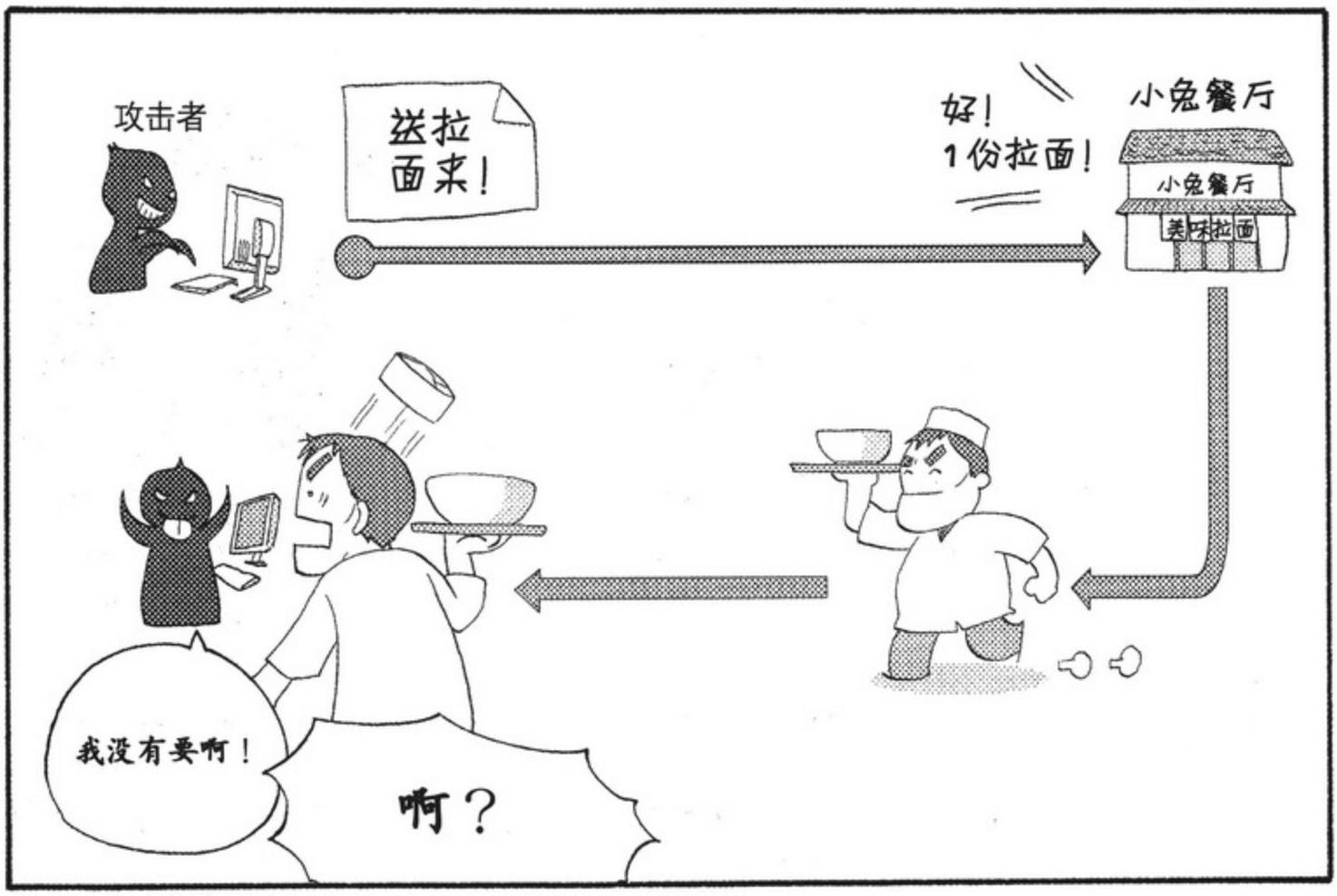
图 4.3　消息认证代码的构成

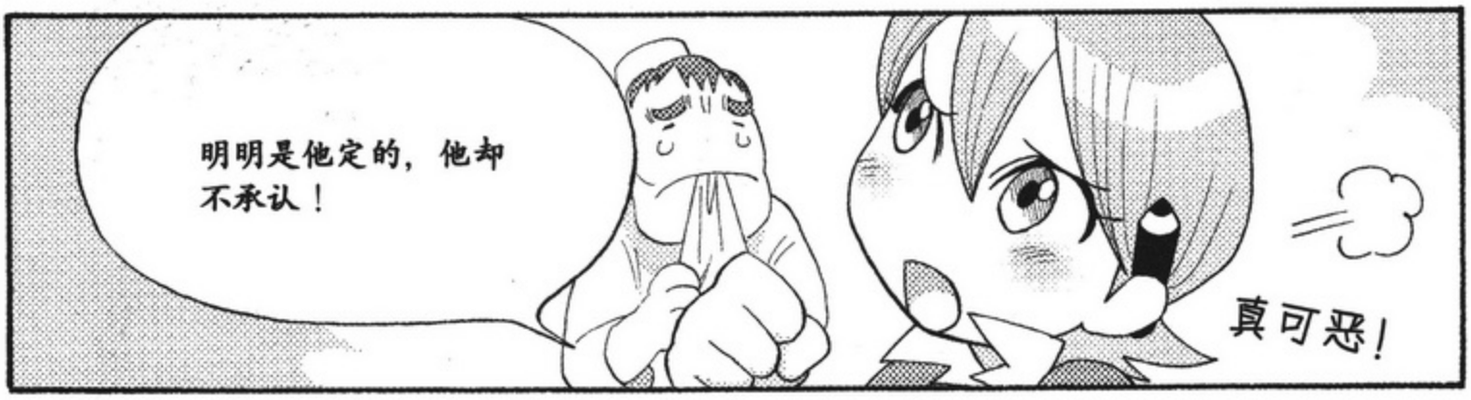
可以把消息认证代码，看成是附加钥匙的单向 Hash 函数的一种。消息认证代码基本和 Hash 值的构成一样，发件人和收件人可以对各种消息的 MAC 值进行计算并作比较，来确认真实性。

使用消息认证代码，在相互计算 MAC 值的时候，只有发件人和收件人能使用共有的钥匙。所以，可以对从原消息中计算的 MAC 值，确认发信人是否为和自己拥有一样钥匙的人。

消息认证代码的构成就是这样的，但是和通用钥匙加密一样，会出现钥匙如何能够安全共有的问题。

消息认证密码应用于国际性银行间的汇款业务，还应用于 Web 上的网上购物等领域中所使用的 SSL/TLS。


攻击者
送拉面来！
好！1份拉面！
小兔餐厅
小兔餐厅
美味拉面
我没有要啊！
啊？


明明是他定的，他却不承认！
真可恶！


用消息认证代码不能确认吗？
很可惜，消息认证密码的效力也是有限度的。
价目表
Beer

✤ 消息认证代码的两个缺点

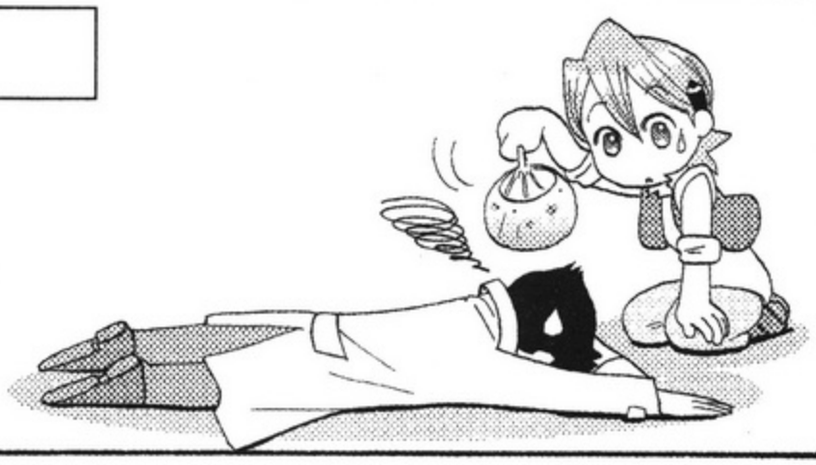
（1）无法确认发件人（Non-Repudiation）

比如说从 A 把消息和 MAC 值发送至 B，在这之后，就算 A 提出“我没有向 B 发送消息，是 B 自己捏造的”这样的说法，也没有任何办法推翻 A 的说法。就算交给第三者来判断真伪，第三者也没有办法判断出消息和 MAC 值到底是 A 做出来的，还是 B 做出来的。

（2）无法拿出给第三者的证据

从 A 发送消息和 MAC 值至 B 的时候，B 无法向第三者证明消息是由 A 发送的。消息和 MAC 值无论是 A 还是 B 都可以做出来。也就是说，对 C 无法判断到底是 A 还是 B 做出来的。

4-3 电子签名



✤ 否认的对策

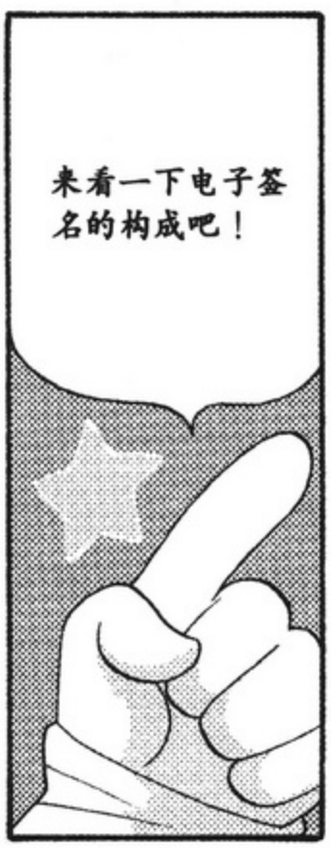


表 4.1 公开钥匙密码和电子签名

公开钥匙密码	收件人用公开钥匙加密	→	密文	→	收件人用私密钥匙解密
电子签名	发件人用公开钥匙解密	←	签名	←	发件人用私密钥匙解密

❀ 电子签名的构成

电子签名是指，发件人在被自己的私密钥匙加密过的消息中，加入签名，把签名和消息一起发送给接收人。

接收人把发送人用公开钥匙加密的签名解密，才能得到消息。然后，将经过解密的消息，与发送过来的另外一个消息进行比较。

两者一致的话，就达到了既可以检验真实性，又可以确认发送人的目的。还有，使用发送人的公开钥匙进行解密，第三者和接收人都同样可以检验签名，还可以向第三者拿出证据，这样就可以防止发送人否认了。

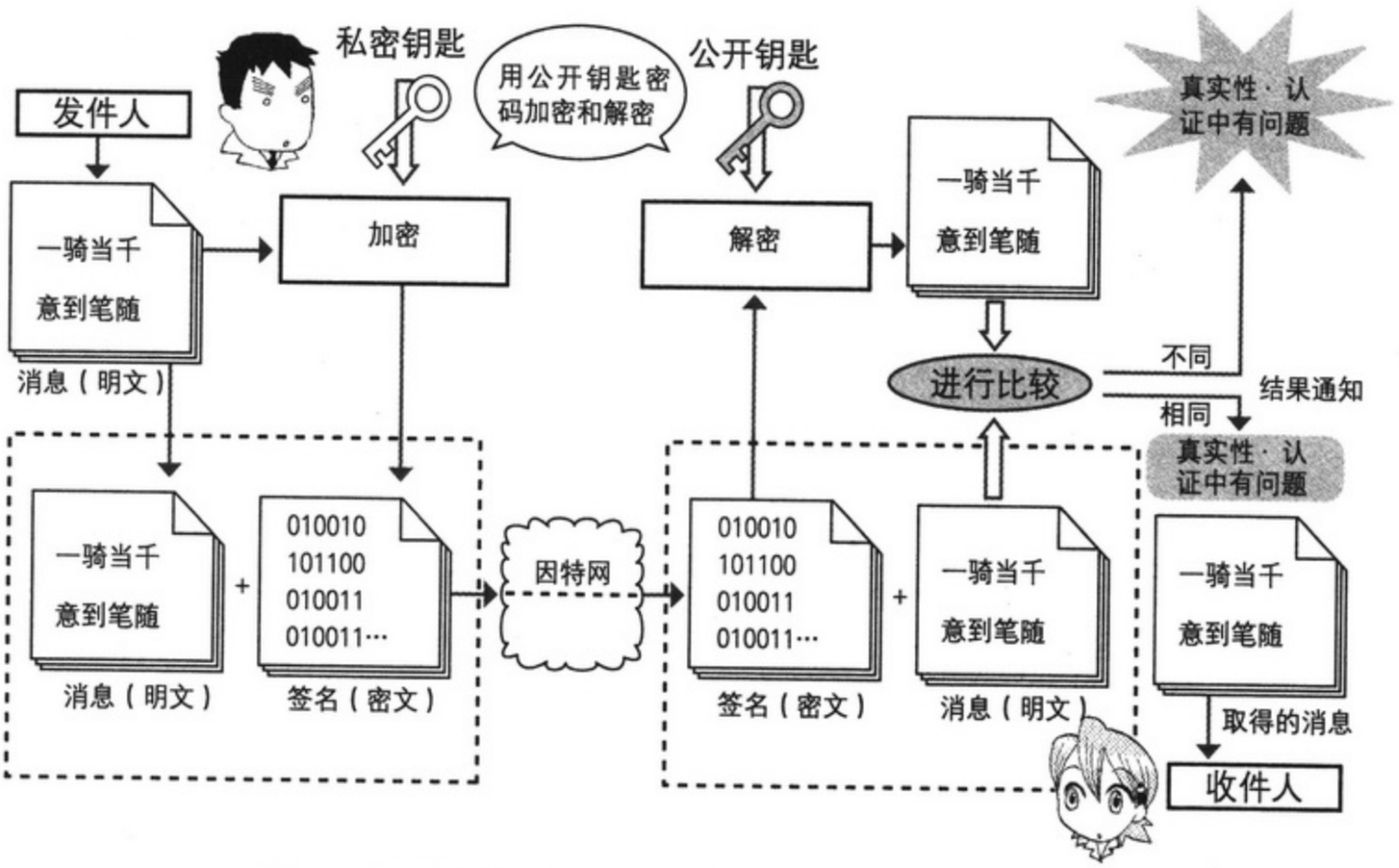


图 4.4 电子签名的构成 1（将消息原本进行加密并签名的情况）

图 4.4 为了将电子签名简化，将消息直接加密做出了签名。

从实际情况来看，要是将消息全部签名，为了避免公开钥匙密码处理时间过长的缺点，可以将消息暂且由单向 Hash 函数变成 Hash 值来做成签名。

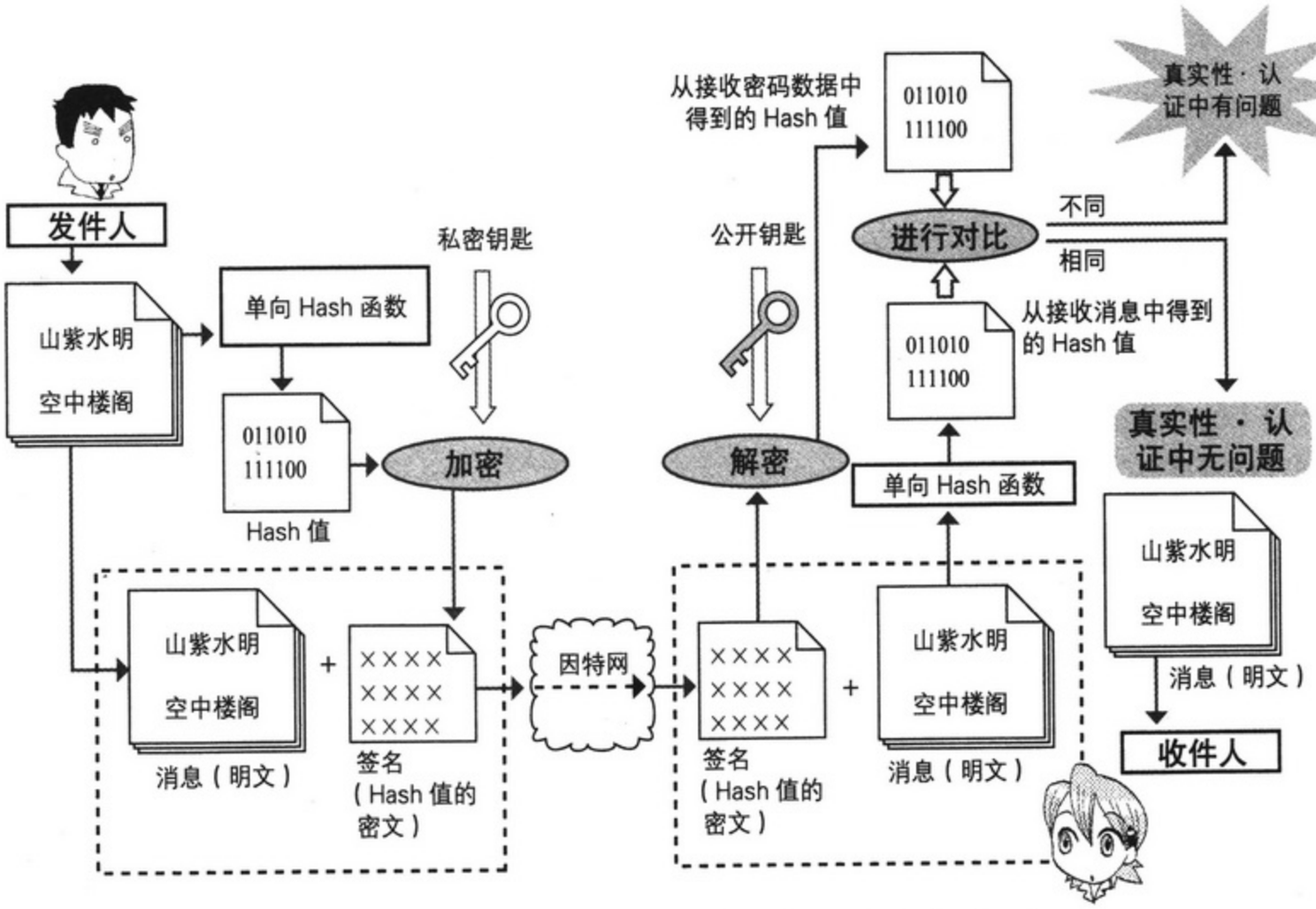


图 4.5　电子签名的构成 2（将被 Hash 化的消息进行加密并署名的情况）

电子签名也被用于制作能够证明 SSL/TLS 服务器正当性的服务器证书。证书是指在公开钥匙（指的是服务器的公开钥匙）上附加这个公开钥匙的电子签名的文件。还有就是，为了防止软件被篡改，也可以在下载用的软件上附加电子签名。

✤ 中间者攻击

把发件人 1 街区的佐藤先生作为 A，把收件人小兔餐厅作为 B。

A 用密码和 B 通信，首先必须取得 B 的公开钥匙。在此之间如有攻击者，他在 B 向 A 发送的途中将公开钥匙截获，再将自己的公开钥匙向 A 进行发送。

A 发送的密文，对于攻击者来说是用自己的公开钥匙进行加密的密文，所以有自己的私密钥匙就可以解密。然后将内容篡改后，用 B 的公开钥匙进行加密并发送，B 就没有办法进行确认了。

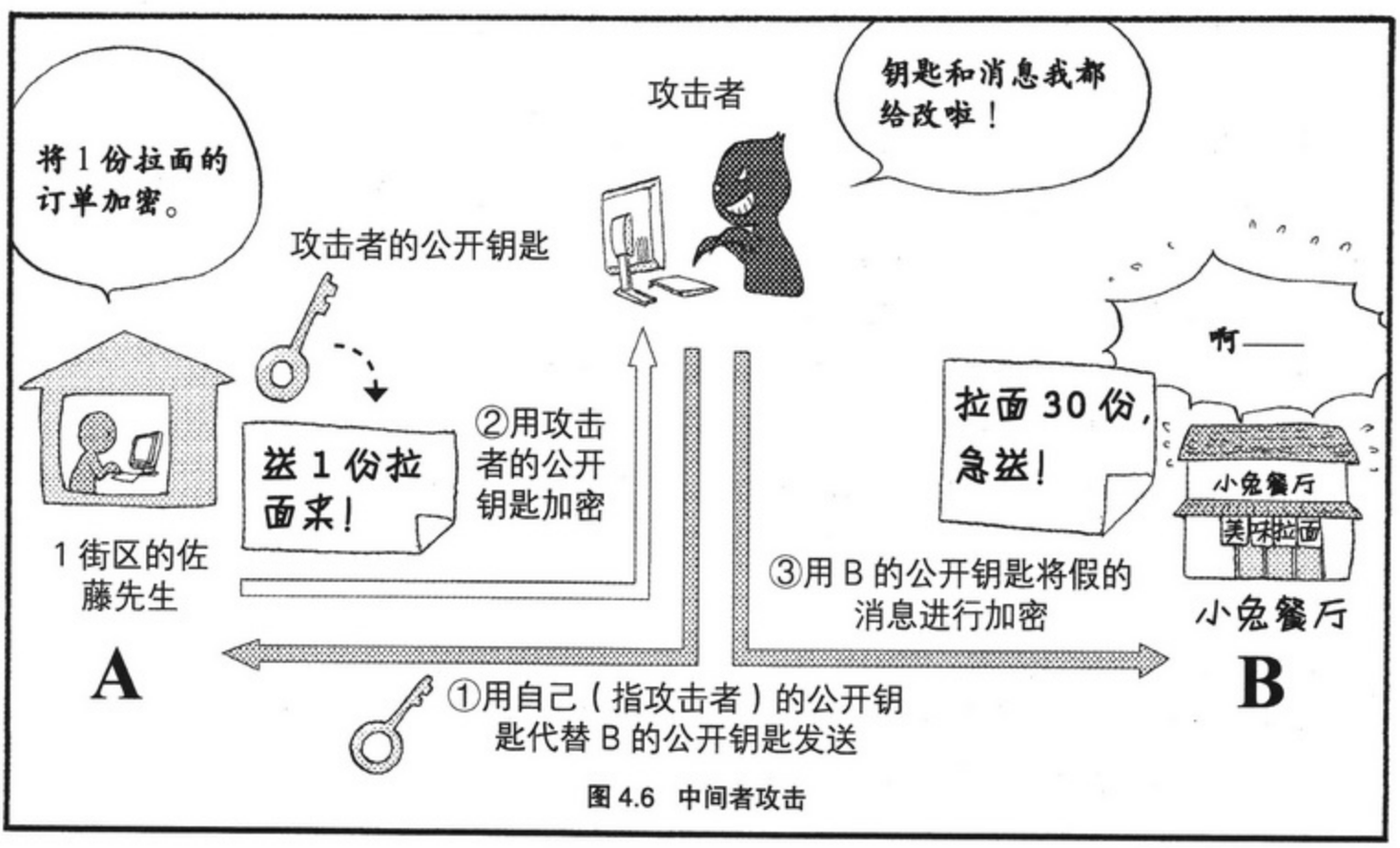


图 4.6　中间者攻击

✤ 中间者攻击的对策

✤ 证书和认证中心

证书是指，在公开钥匙上附加此公开钥匙的电子签名。证书是由认证中心发行的。想把公开钥匙公开的使用者可以在认证中心（CA：Certification Authority）将自己的公开钥匙进行登录，同时申请发行证书。

认证中心会针对申请，对想要将公开钥匙公开的使用者的正当性进行确认，如果合乎认证中心的标准，就通过公开钥匙做出电子签名，并将公开钥匙和电子签名组合做出证书。公开钥匙和私密钥匙的组合，有时候是使用者做出的，有时候是登录时认证中心做出的。

使用了证书的公开钥匙验证，保证了公开钥匙的使用者是 A。为此，使用者 A 可以从信誉很高的第三者，也就是认证中心那里得到对公开钥匙的真实性的证明。可以根据图 4.7 来看一下，下面①～⑥表示的是程序。

① 使用者 A 向认证中心申请发行自己的公开钥匙的证书。

② 认证中心在对使用者 A 的身份进行确认之后，发行证书。发行的证书，是在使用者 A 的公开钥匙上，附加认证中心电子签名。

③ 认证中心将证书储存在数据保存处。

④ 使用者 B 从数据保存处中下载使用者 A 的证书。

⑤ 使用者 B 用使用者 A 的、包含电子签名的证书对认证中心的公开钥匙进行解密。

⑥ 可以将被解密的钥匙与包含证书的公开钥匙进行比较来验证。这两个钥匙相同的话，就可以保证包含证书的公开钥匙为使用者 A 所有。

从以上的程序可以知道，使用者 B 可以得到被保证的使用者 A 的公开钥匙。如果使用被保证的使用者 A 的公开钥匙的话，对于用使用者 A 的私密钥匙进行加密的附有电子签名的消息来说，是真实消息这一点就可以被验证。真实消息就是能同时满足以下 3 个条件的消息。

① 确认消息没有被篡改。

② 确认没有第三者冒充使用者 A，发送消息。

③ 确认使用者 A 无法否认此消息是他自己发送的。

从证明公开钥匙的真实性来看，附有电子签名的消息，它的真实性可以由以上 3 个条件来保证。以这个为基础，接下来所讲述的公开钥匙密码基础设施（PKI）的构成就完成了。

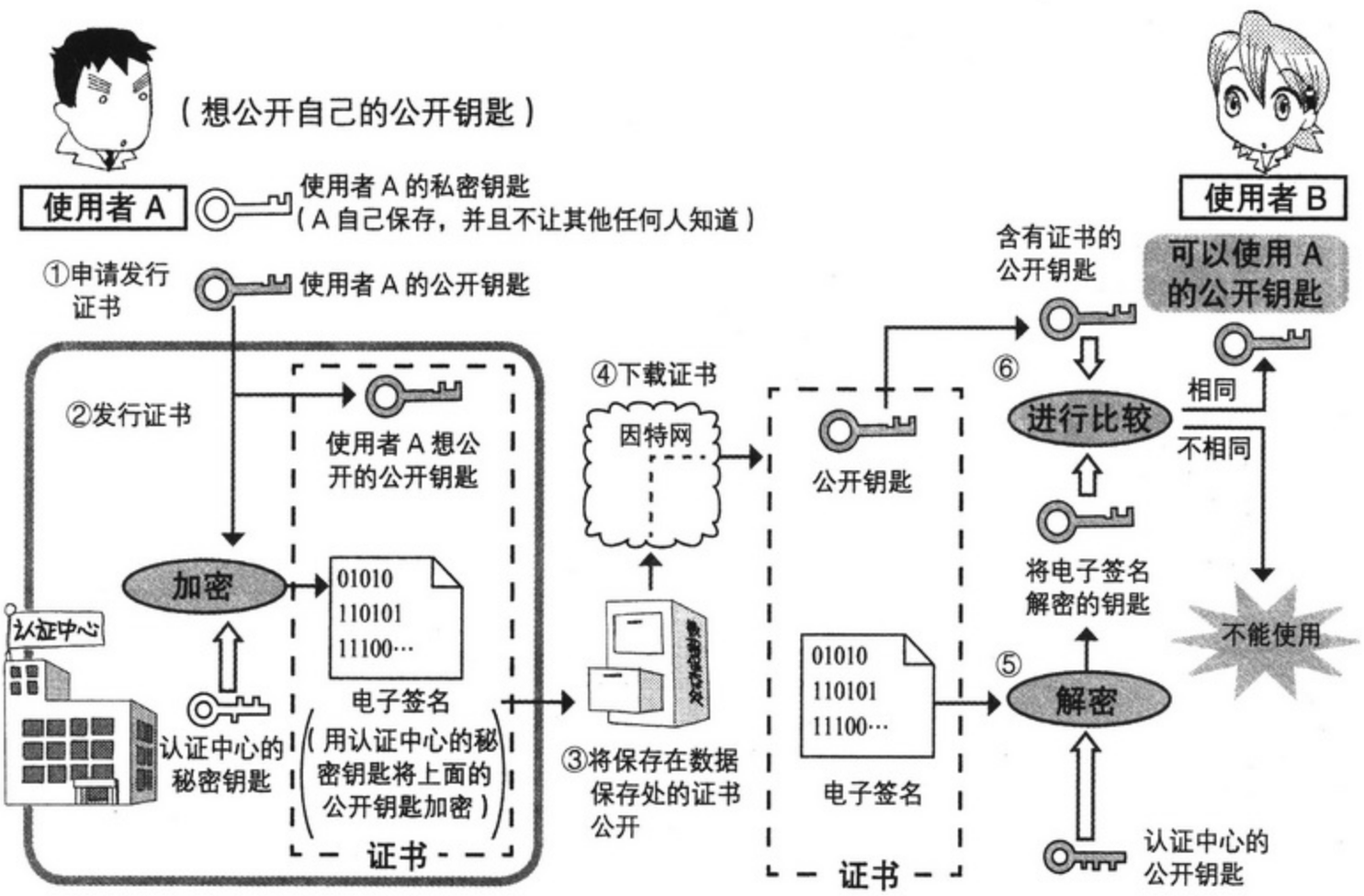
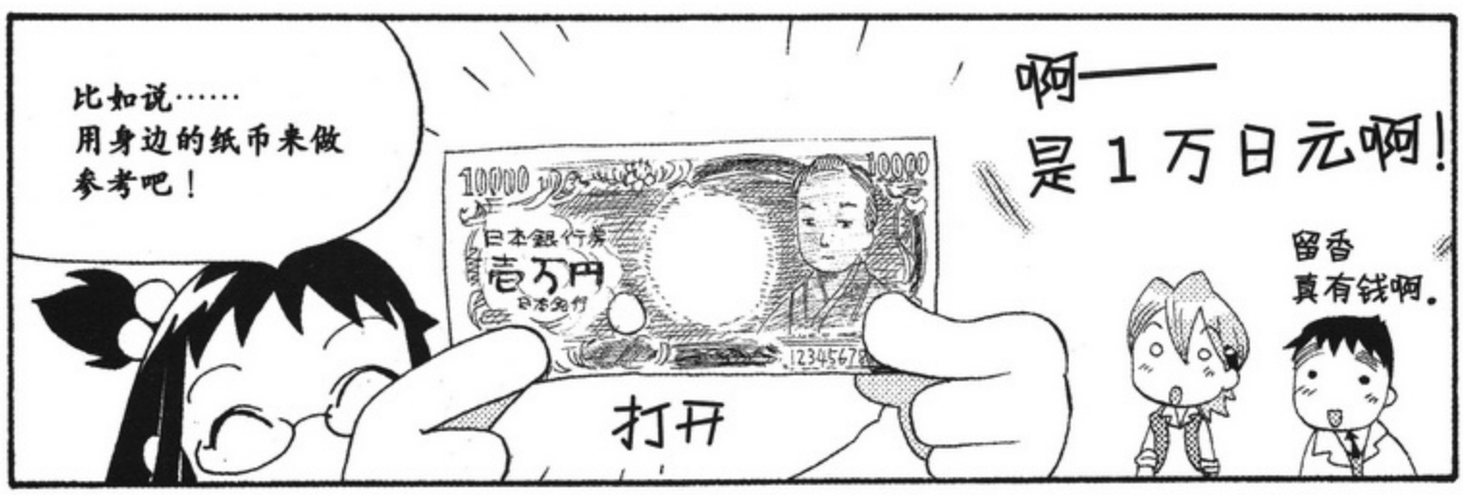
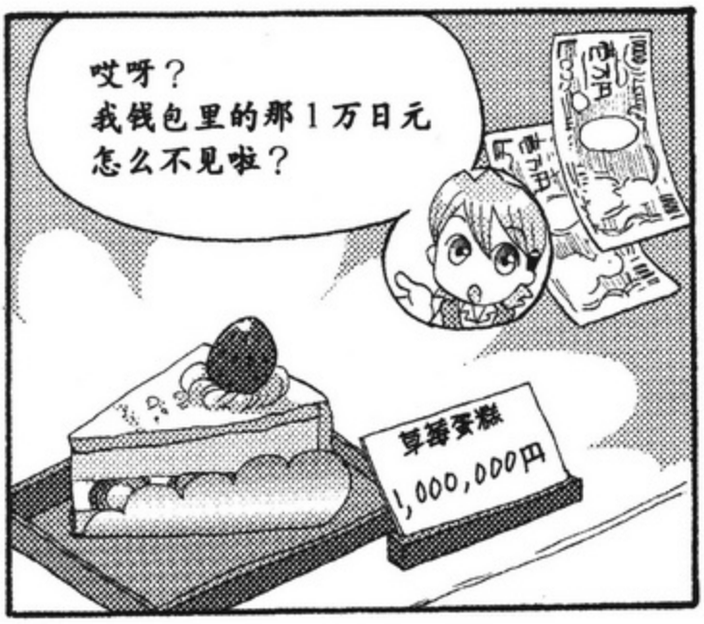
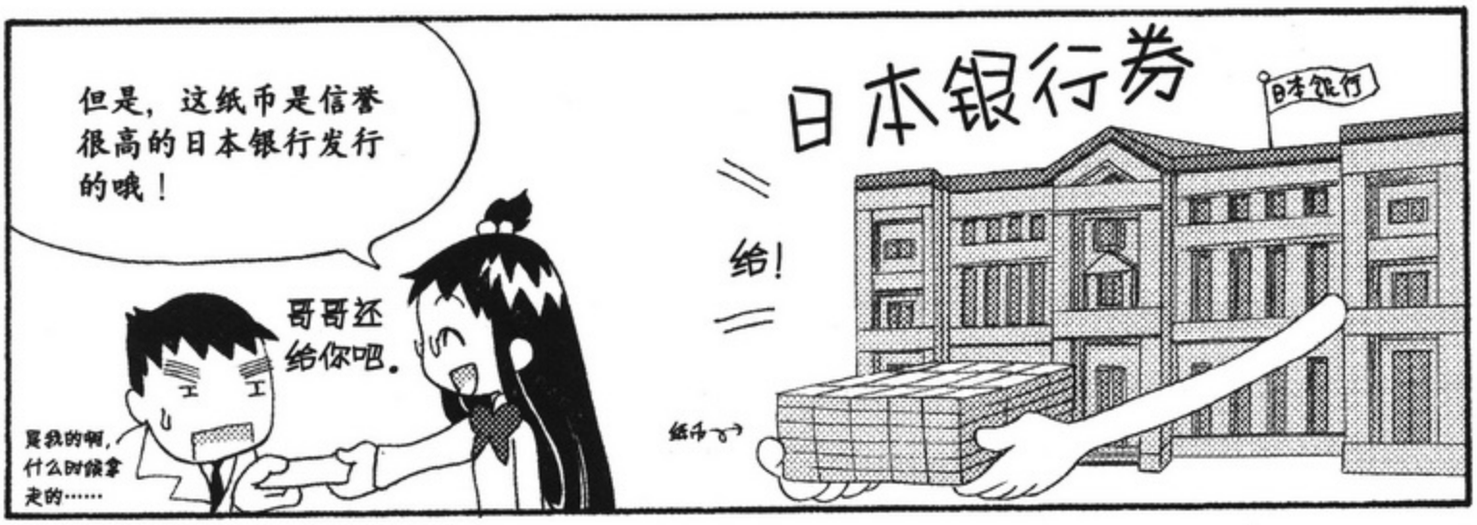


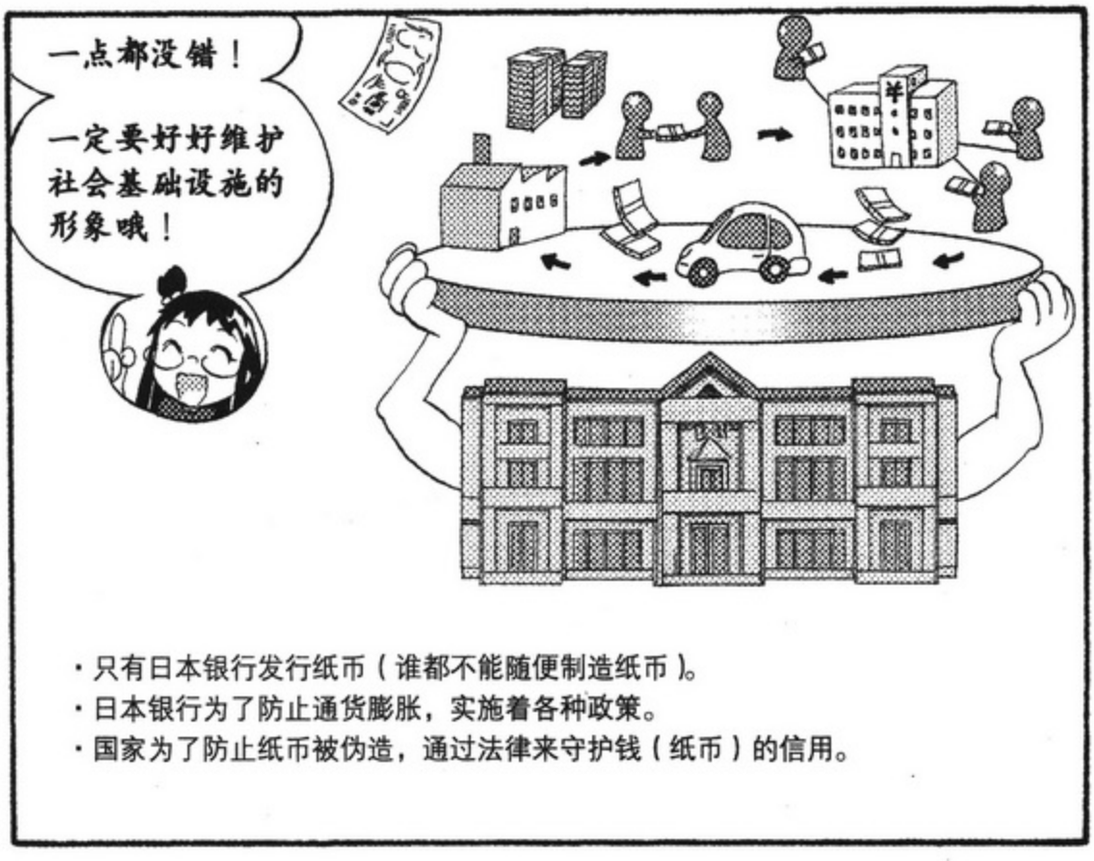
图 4.7　发行证书的流程

4-4 公开钥匙密码基础设施（PKI）





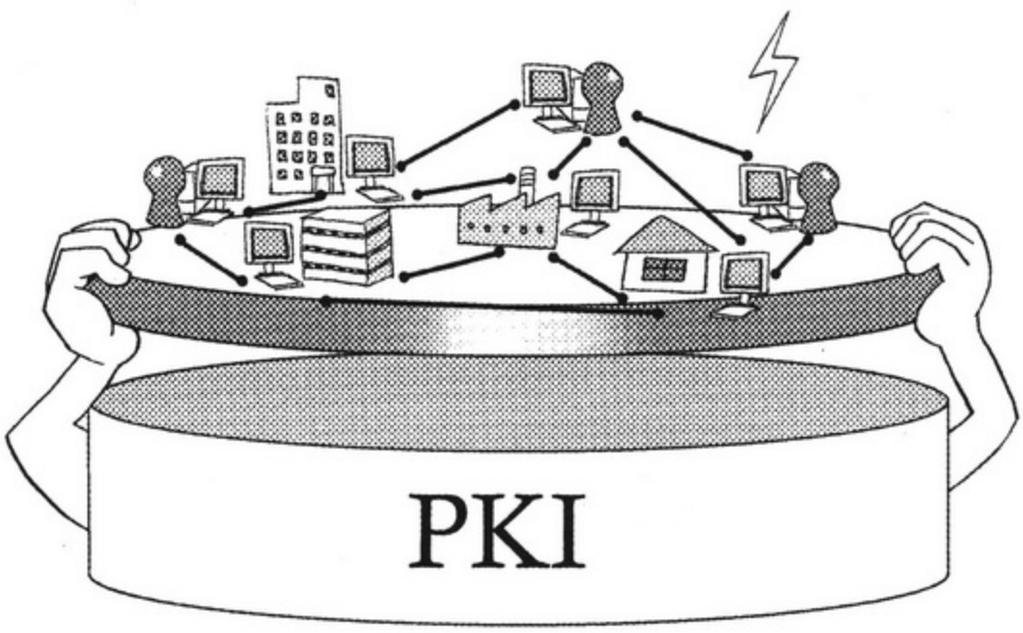




- 只有日本银行发行纸币（谁都不能随便制造纸币）。
- 日本银行为了防止通货膨胀，实施着各种政策。
- 国家为了防止纸币被伪造，通过法律来守护钱（纸币）的信用。

就和保证纸币的安全性和可信性的各种社会基础设施一样。使用公开钥匙密码的系统，由叫做 PKI 的社会基础设施来保证信息的安全性和可信性。

总之，为了保证使用公开钥匙密码电子邮件的交换和网上的商品交易，等等行为的安全进行，才有了叫做 PKI 的社会基础设施。



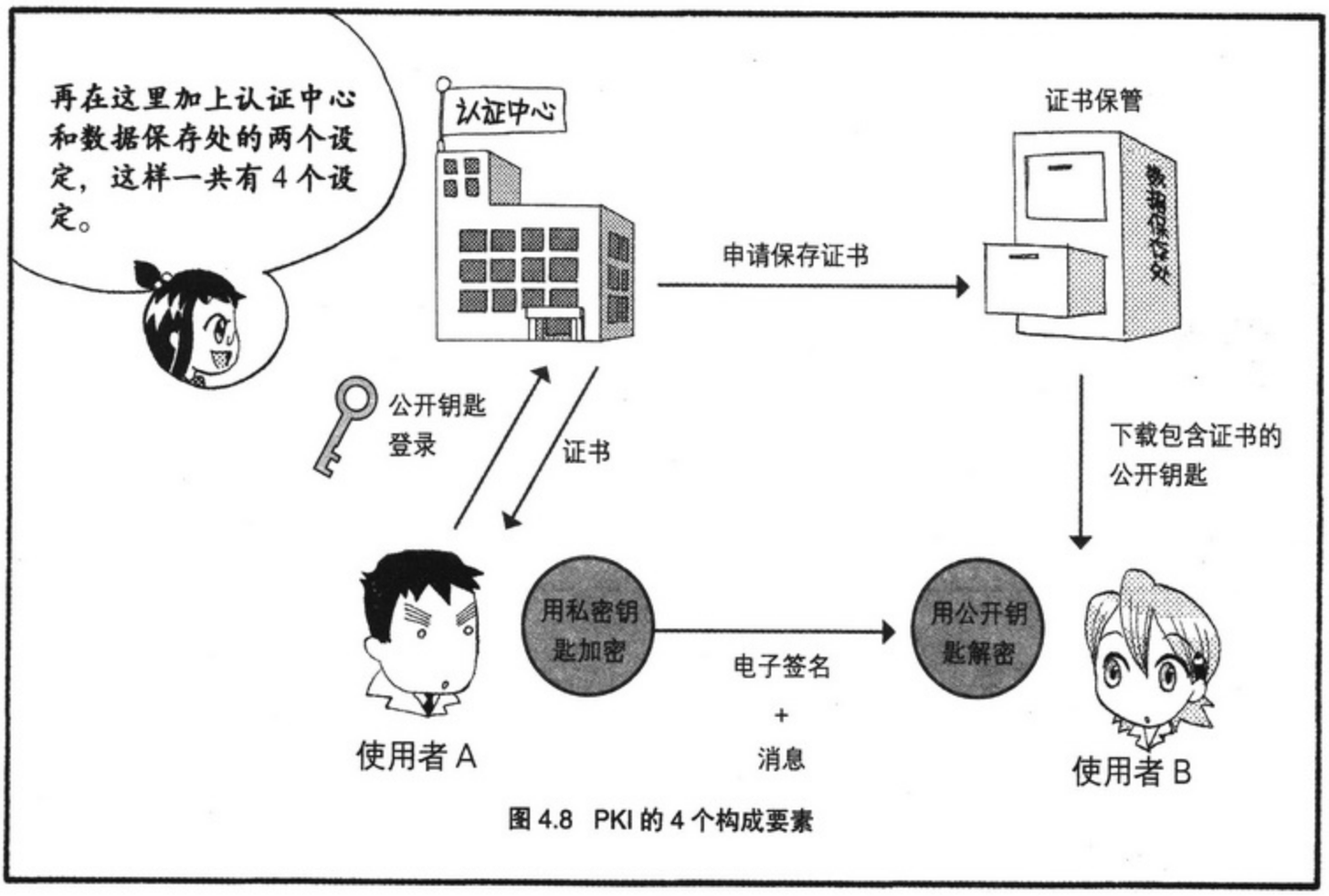
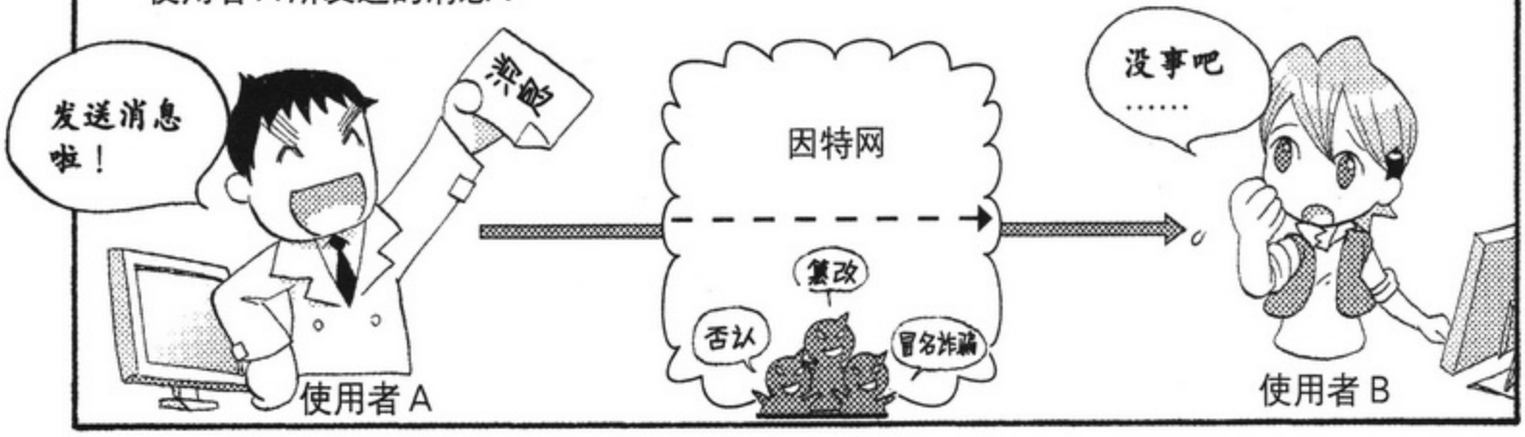
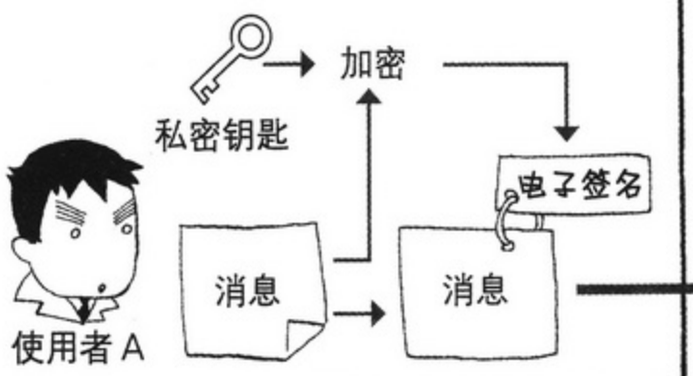


图 4.8　PKI 的 4 个构成要素

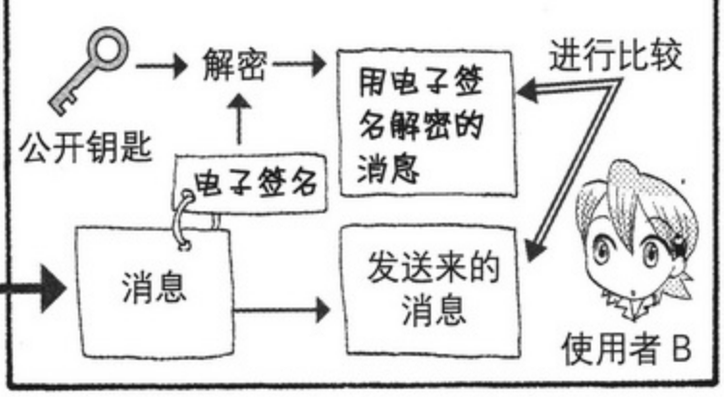
① 使用者 B 希望得到没有被篡改、没有被冒名诈骗、不能被否认的，使用者 A 所发送的消息！



② 使用者 A 将使用自己的私密钥匙做出的电子签名附加在消息上，向使用者 B 发送。



③ 使用者 B 用使用者 A 的公开钥匙验证接收到的电子签名，如果是正常，那消息就是正确的。

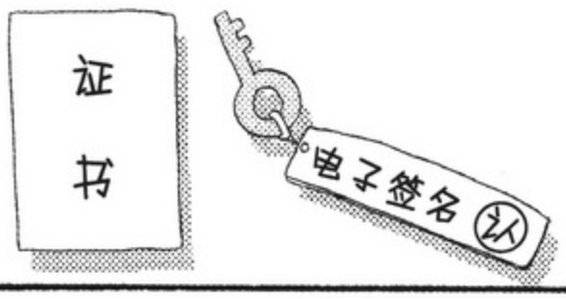


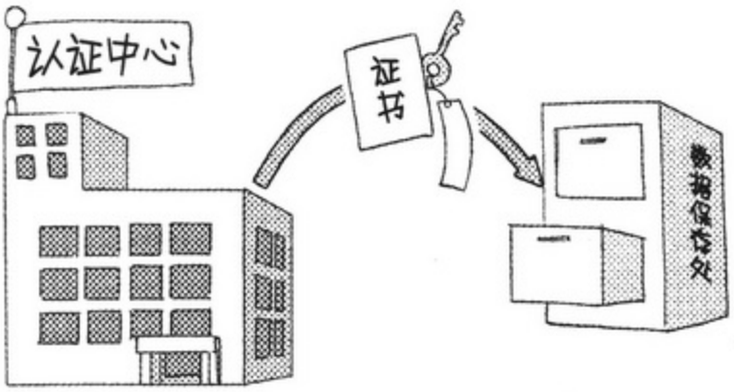
④

⑤

⑥ 使用者 A 将公开钥匙在认证中心登录，得到了认证中心发行的证书。

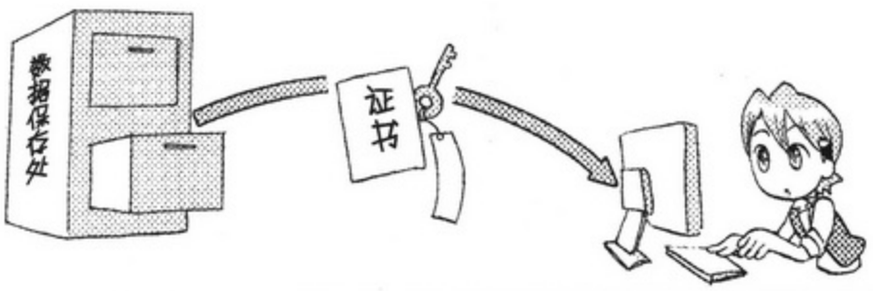
⑦ 证书的内容是使用者 A 的公开钥匙和通过认证中心附加的电子签名。



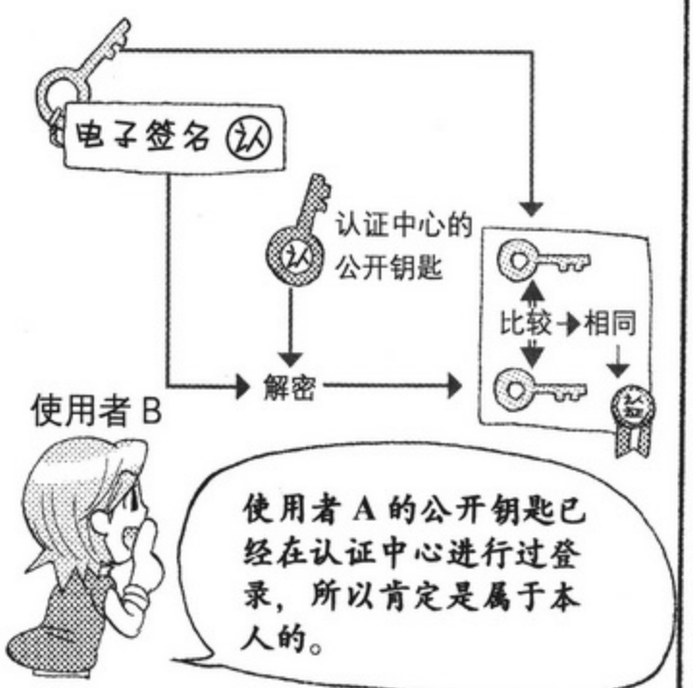


⑧ 认证中心通过数据保存处来保管证书。

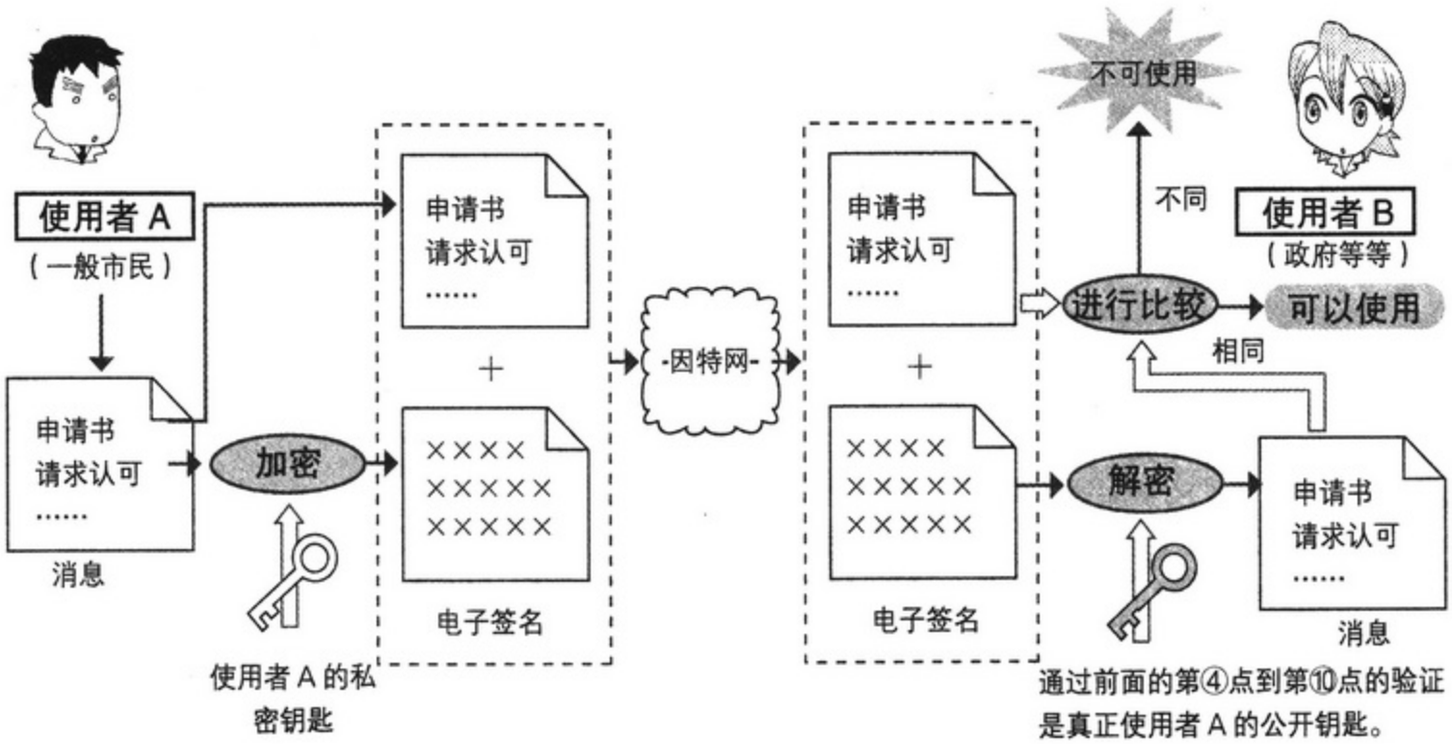
⑨ 使用者 B 从数据保存处下载使用者 A 的证书。



⑩ 使用者 B 将使用者 A 的包含有证书的公开钥匙与用电子签名解密后所得到的公开钥匙进行比较，如果相同的话，认证完毕。



⑪ 验证完毕的话，就可以知道使用者 A 的包含有证书的公开钥匙是属实的。所在第③点中得到的消息也是属实的（篡改、冒名诈骗、否认都被排除在外了）。



因为使用者 A 的公开钥匙已经得到确认，作为消息取得的申请书等和用电子签名解密过的文件是相同的话，那么，申请书等就没有被篡改、冒名诈骗以及使用者 A 对提出的事无法否认。

图 4.9　利用 PKI 申请手续范例


可是……
使用者B必须做这些麻烦事吗？
可能做不到啊。
没关系！
实际的程序都是使用Web浏览器、专用软件登录卡、存储读卡器，等等的硬件设备自动进行的。
证书是什么？
就是这个啦！
证书
常规
详细信息
证书路径
显示(S): <所有>
域
值
序列号
3e 7c 79
签名算法
md5RSA
颁发者
Thawte Code Signing CA,...
有效起始日期
2004年8月3日 8:41:58
有效终止日期
2006年8月22日 5:24:45
主题
Long Beach, California,...
公钥
RSA (2048 Bits)
Web浏览器有显示证书的功能。在证书中，记载有版本信息、序列号码、签名算法、发行者、有效期、公开钥匙信息等内容。
好了，密码入门讲座到这里就结束了！
你们辛苦啦！
终于结束啦！
叮咚

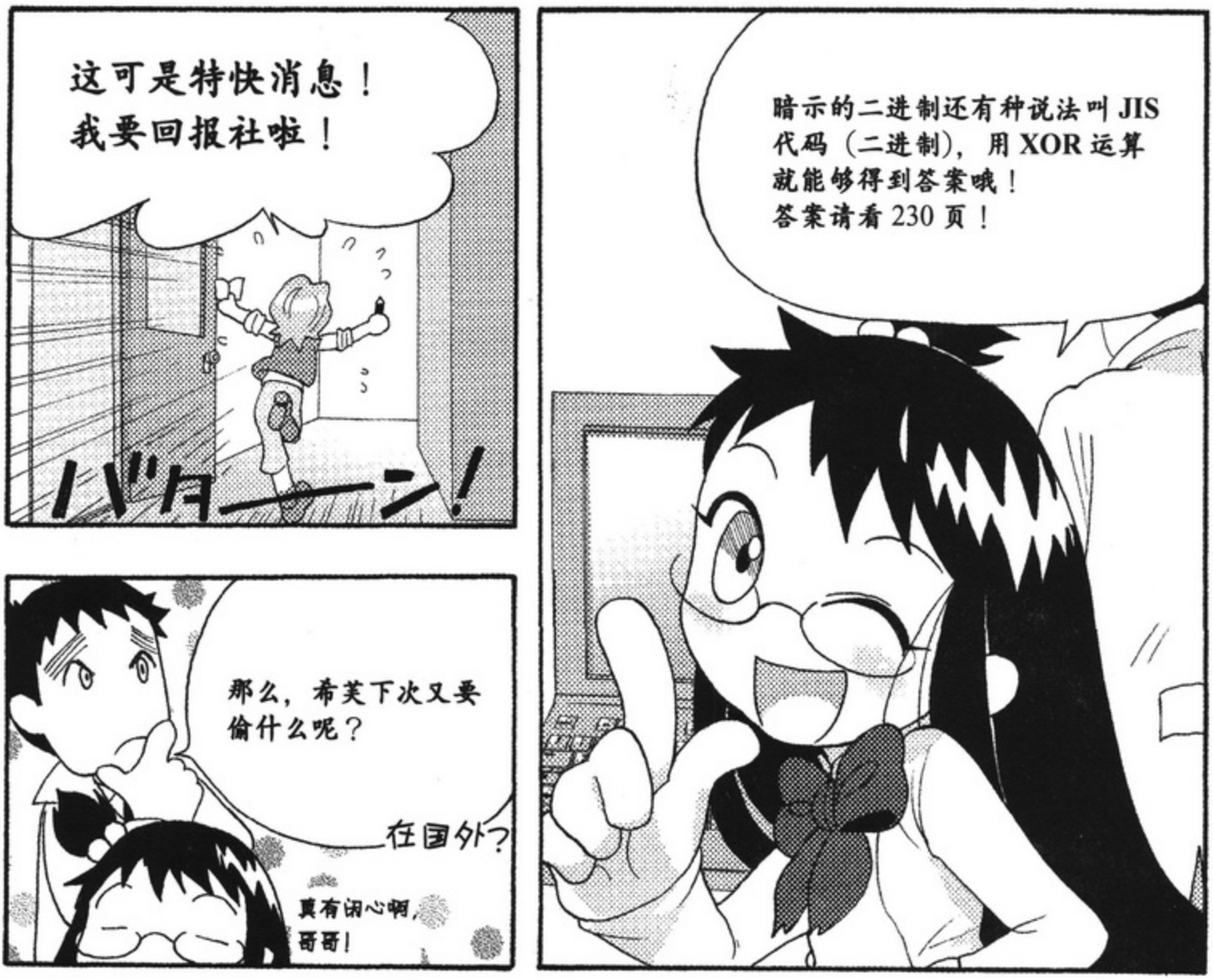
看了第 181 页，电子邮件前几行的文字，就会知道希芙到底是谁了。

发件人：小兰（怪盗希芙）
收件人：小香
主题：你好吗？

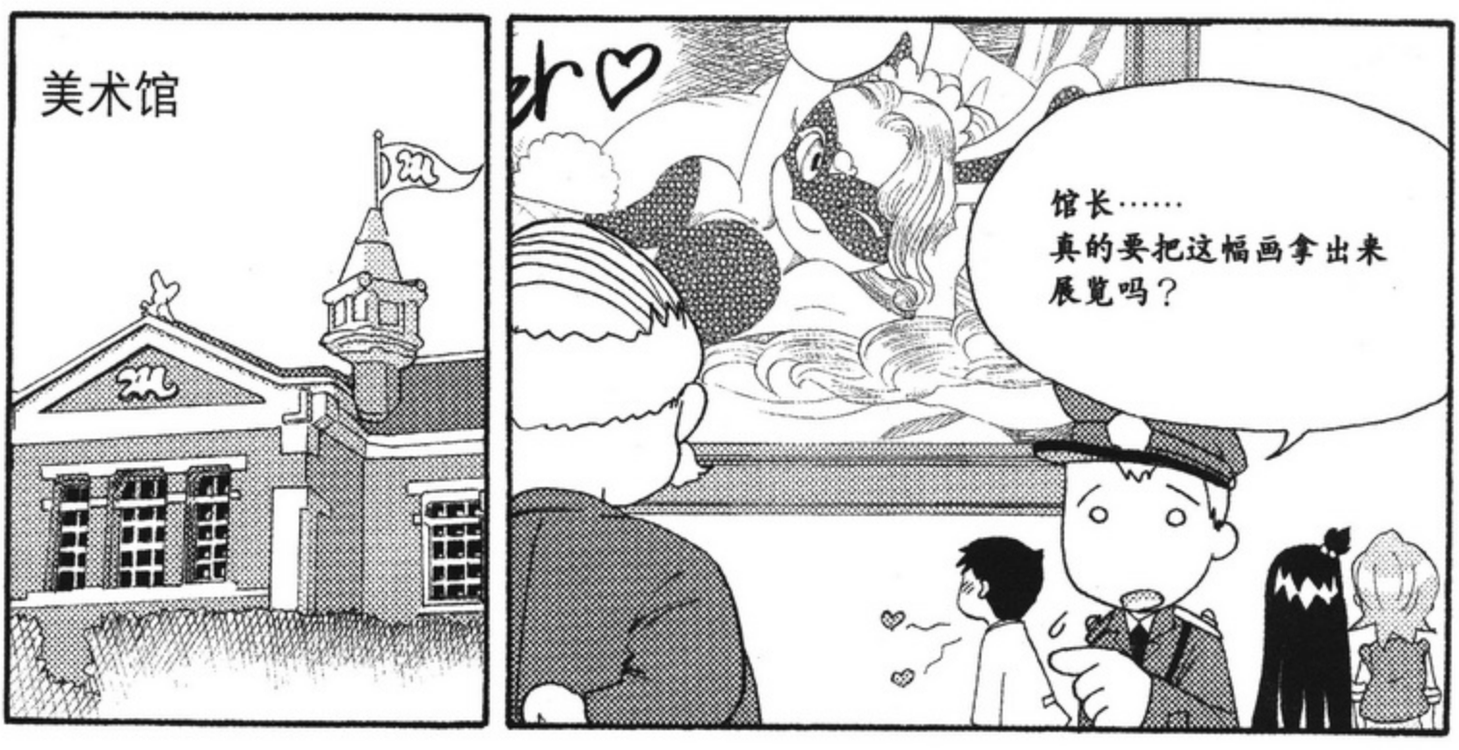
我已经归还了名画《微笑的麦当娜》和那颗绿宝石（和失窃的保险公司做了笔交易）。现在，我在自由的国家了，马上又有大事要做哦！给你个二进制的暗示……

00001011　00000110　00000110
00000001　00010111　00000111
00001010

好了，再见啦！


这可是特快消息！
我要回报社啦！
バタ——ン！
暗示的二进制还有种说法叫 JIS 代码（二进制），用 XOR 运算就能够得到答案哦！
答案请看 230 页！
那么，希芙下次又要偷什么呢？
在国外？
真有闲心啊，哥哥！

数日后……


美术馆
馆长……
真的要把这幅画拿出来展览吗？


当然啦！
我要把它代替名画《微笑的麦当娜》挂上去！
微笑的麦当娜
……
宝石也归还了，可以安心啦！
太好啦！太好啦！
兄
弟！
从现在开始更要注意信息的防范，让我们共建安全的社会吧！
好啊！

✤专栏✤ 零知识对话证明

最近，使用信用卡支付的时候，信用卡信息被非法读取和没有购物却被要求付款的事件层出不穷。像这种情况，在认证本人身份的同时，当事人的秘密也会有外泄的危险。所以，在不会泄漏任何秘密（零知识）的前提下，让对方确认（证明）身份（信用卡的真实性，authenticity）的方法就变得非常必要了。

作为对这种需求所做出的回应，在 1985 年，Goldwasser，Micali，Rackoff 提出了“零知识对话证明”这个概念。零知识对话证明是指，将自己持有的信用卡的真实性向对方（信用卡公司）证明的方法。当时提出，在信用卡的秘密信息（例如十进制 100 位以上随机数密码）不会被泄漏的前提下进行认证的观点，“让公众相信，不需要泄露代表信用卡秘密信息的随机数，只需使用作为证明自己身份的随机数即可进行身份认证”。这个观点让大家觉得自我感觉非常好。像这样的事情，可以用以严谨的密码数学理论为基础的数理魔法来实现。

关于这种方法，我们分为准备阶段和实行阶段来进行说明。

●准备阶段●

对于零知识对话证明从数学魔法入手，首先必须要设置可以信赖的中心机构（验证者）。举个例子来说明：

① 设定向全部用户公开的合数 *N*

验证中心准备两个素数（p，q），取其乘积，合数 N，即

$$N = pq \quad \cdots\cdots (1)$$

然后将 p 和 q 秘密化。实际上要使用 80 位左右的巨大数值的素数，在这里举个简单的例子，使用两位的素数 p=13 和 q=19。这两个素数的乘积为 N 的话，

$$N = 13 \times 19 = 247$$

得出 3 位的合数（实际上，无论是什么样的计算机，都会生成一个不可能对合数 N 做素因数分解的，即计算量大到不可能完成运算的那种巨大数值的素数）。将这个合数 N 向全体用户公开。

②将全体用户的 ID 在验证中心登录

ID 就是每个用户所公开的数值（与公开钥匙相同），并且与其本人一一对应。

也就是说，每个用户都可以识别的公开数值就是 ID。各个用户将 ID 在中心进行登录。比如，用户 A 的 ID 用 ID_A 来表示。

③ 中心对于各个用户的私密钥匙的计算和通知

中心以各个用户登录的 ID 为基础，将那个 ID 的平方根作为域来计算出模。虽然实数平方根运算计算起来很容易，但是只有在知道整数的合数 N 的两个素数 p 和 q 的情况下，它的平方根才能够很容易地被计算出来。这种叫做零知识对话证明的数理魔法的技巧，就是利用了这个平方根计算的困难性而做成的。

在这种例子的情况下，因为只有中心知道素数 13 和 19 并计算各个用户登录的 ID 的平方根，所以秘密不会泄漏。假设用户 A 的 ID（ID_A）是 101，这个时候它的平方根就是 71。

$$\sqrt{101} \pmod{247} = 71$$

当然，反之将 71 的 2 次方，可以变为 $71^2 \pmod{247} = 101$。把这个 71 作为用户 A 的私密钥匙 S_A，秘密地发送给用户 A，实际上，如果使用 100 位以上的数字，则用 A 是记不住的。在一般情况下，用户 A 的 ID（ID_A）和私密钥匙 S_A 之间，存在着下面的关系。

$$\sqrt{ID_A} \pmod{N} = S_A \cdots\cdots (2)$$

$$(S_A)^2 \pmod{N} = ID_A \cdots\cdots (3)$$

另外，私密钥匙 S_A 的目的是，不仅仅是为了确认用户 A 这个人，还为了确认用户 A 所持有的信用卡的真实性。所以，A 不需要像记下银行的储蓄卡密码一样记住 S_A。其他的用户也是按照这样的程序分别发送私密钥匙。

●实行阶段（认证流程）●

用户 A 如何向用户 B 证明，自己就是真正的用户 A（自己所持有的信用卡是真的）。下面就是这个证明的程序。

步骤 1　用户 A 向用户 B 提出的证明申请（其一）

首先，用户 A 选择适当的随机数进行二次乘方，然后除以合数 N 求余，这个余数 y_A，即

$$y_A = (r_A)^2 \pmod N \quad \cdots\cdots (4)$$

然后发送给用户 B。比如说，用户 A 选择了 50 作为随机数，这个时候，

$$y_A = 50^2 = 2500 = 30 \pmod{247}$$

所以，将 30 发送给用户 B。

步骤 2　用户 A 向用户 B 提出的证明申请（其二）

接下来用户 A 通过自己从中心取得的私密钥匙 S_A 和步骤 1 选择的随机数 r_A 的乘积，以合数 N 作为域进行计算，即

$$z_A = S_A r_A \pmod N \quad \cdots\cdots (5)$$

之后向用户 B 发送。以前面选择的随机数 r_A 为 50，作为例子的话，

$$z_A = 71 \times 50 = 92 \pmod{247}$$

所以，就变为将 92 发送给用户 B。

步骤 3　用户 B 对于用户 A 真实性的确认操作（其一）

那么，用户 B 将用户 A 发送来的 z_A，进行 2 次乘方运算，然后再以合数 N 为域进行计算，即

$$v_A = (z_A)^2 \pmod N \quad \cdots\cdots (6)$$

$$= (S_A r_A)^2 \pmod N \quad \cdots\cdots (7)$$

之后，这个例子是 z_A=92，如下所示。

$$v_A = 92^2 = 8464 = 66 \pmod{247}$$

步骤 4　用户 B 对于用户 A 真实性的确认操作（其二）

接下来用户 B 在步骤 3 中求出了 v_A，用 v_A 除以步骤 1 中用户 A 发送的 y_A，进行计算。

$$\omega_A = \frac{v_A}{y_A} \pmod N \quad \cdots\cdots (8)$$

$$= v_A \times (y_A^{-1}) \pmod N \quad \cdots\cdots (9)$$

当然，所有的计算都是以合数 N 为域进行的。y_A^{-1} 和 y_A 是倒数关系，也就是说 y_A^{-1} 是满足下面公式的值。

$$y_A \times (y_A^{-1}) = 1 \pmod N \quad \cdots\cdots (10)$$

从这个例子，可以得出 v_A=66，y_A=30，且 $y_A^{-1}= 30^{-1} \pmod{247} =140$，所以最后就出现了用户 A 的 ID（$ID_A$）。

$$w_A = \frac{66}{30} \pmod{247} = 66 \times 30^{-1} \pmod{247} = 66 \times 140 \pmod{247} = 101$$

从以上的步骤 1 到步骤 4 的处理来看，在发件人真的是用户 A 的情况下，用户 B 对用户 A 的真实性可以进行确认。在步骤 4 中，叫做 101 的用户 A 的 ID（ID_A）出现的理由是，将用户 A 的私密钥匙 S_A 进行二次乘方运算，就可以变成 ID。也就是说，通过公式（8），来考虑公式（3）、公式（4）、公式（5）的话，可以看出

$$w_A = \frac{[(\text{用户 A 的私密钥匙 } S_A) \times (\text{随机数 } r_A)]^2}{(\text{随机数 } r_A)^2} \quad \cdots\cdots (11)$$

$$= \frac{(S_A r_A)^2}{(r_A)^2} = (S_A)^2 = ID_A = \text{用户 A 的 ID} \quad \cdots\cdots (12)$$

的关系成立。

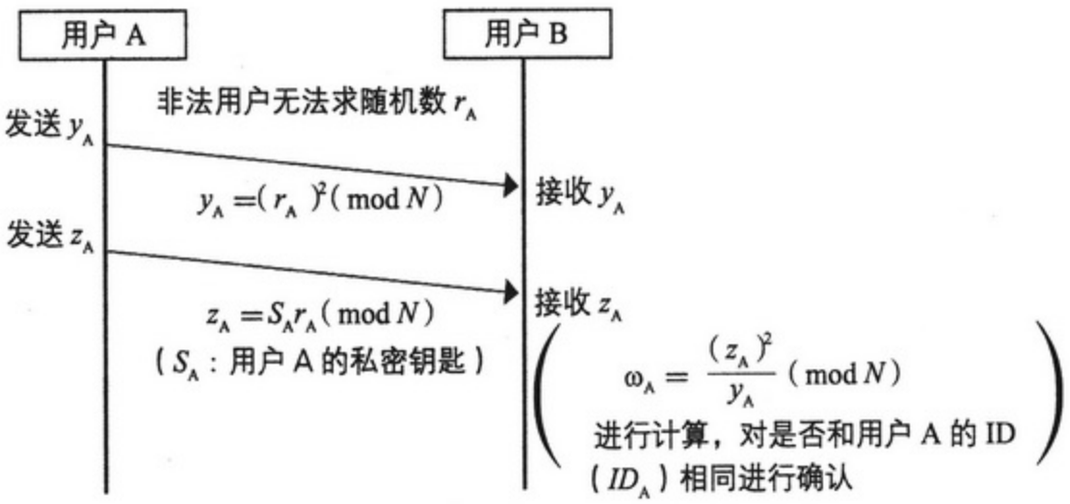


图 4.10 对于零知识对话证明（真实性）的确认程序

●冒名诈骗的方法●

通过图 4.10，来参考一下在被公开的用户 A 的 ID_A 是 101 的情况下，带有恶意的用户 X 非法变成用户 A 的方法。当然，设定用户 X 对于用户 A 的私密钥匙 S_A 的事一无所知。

为此，用户 X 要满足

$$e^2 = ID_A \times f \pmod{247} \quad \cdots\cdots (13)$$

的关系决定 e 和 f 用最初的步骤 1 来发送 f，然后用步骤 2 来发送 e。

那么，作为满足公式（13）的例子，e= 25，f = 82，实行步骤 3 和步骤 4，就得到了用户 A 的 ID（ID_A=101）。

用户 X 做出了合适的 e 和 f，分别利用了在公式（5）的 z_A 和公式（4）的 y_A 相同这一点，计算出公式（6）和公式（9），对用户 A 公开的 ID（ID_A=101）的出现进行确认。

这个计算是以合数 N（这个例子是 N=13 × 19=247）为域来进行取模运算的。

步骤 3

$$v_A = e^2 = 25^2 = 131$$

步骤 4

$$w_A = \frac{e^2}{f} = e^2 \times f^{-1} = v_A \times 82^{-1}$$

在这里，从 $82^{-1} \pmod{247} = 244$，得到 w_A=131 × 244（mod 247）=101

从以上的结果可以看出，即使用户 X 不知道私密钥匙 S_A 和随机数 r_A，也可以做出被用户 A 公开的 ID_A（=101），还可以变成用户 A，进行非法操作（图 4.11）。

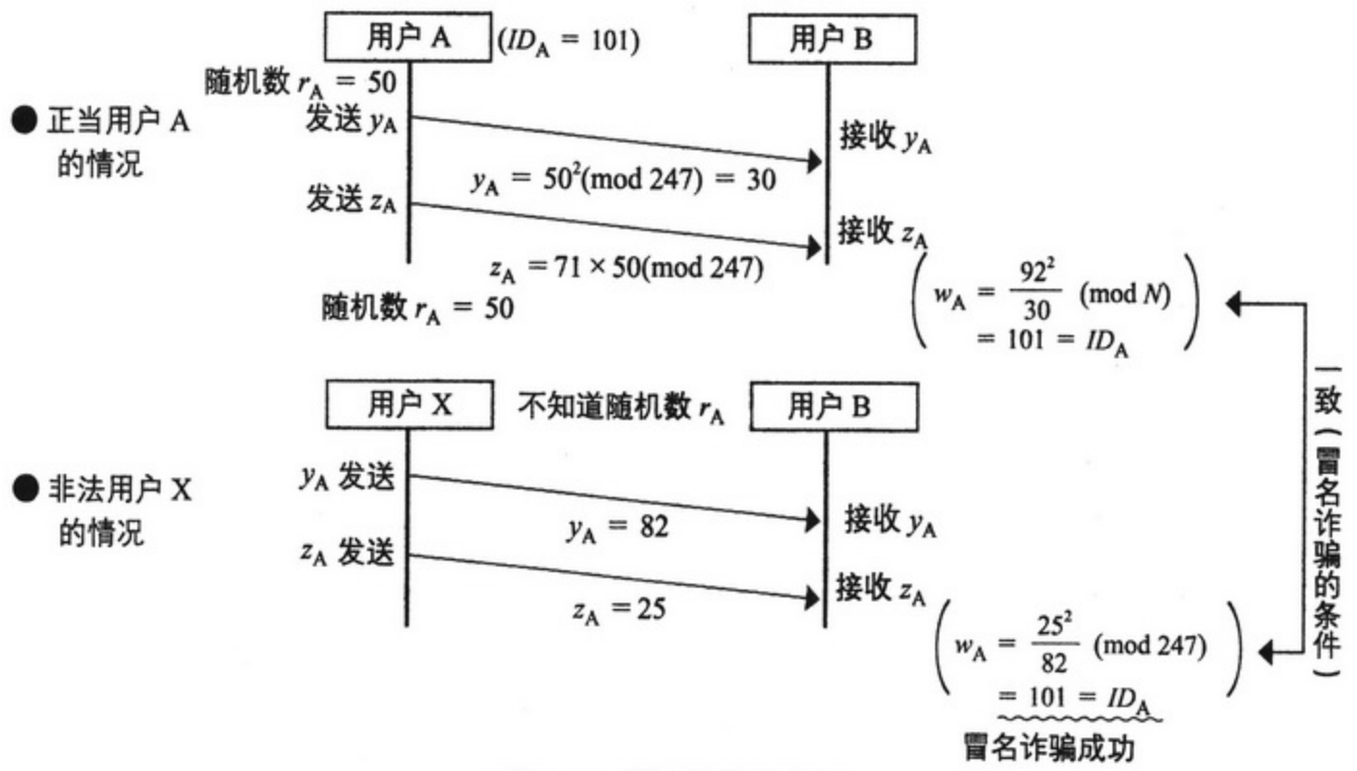


图 4.11　冒名诈骗的范例

像这样逃脱图 4.10 的确认，从被公开用户的每个 ID 信息，来满足公式（13）就可以。具体地说，不知道用户 A 的私密钥匙 S_A 的非法用户 X，首先确定合适的 e 的 2 次方，然后除以 ID_A，f 即

$$f=\frac{e^2}{ID_A}\ (\mathrm{mod}\ N) \quad \cdots\cdots (14)$$

求得之后，最初将 f 发送，然后发送 e，就算得不到用户 A 不知道的随机数 r_A，也可以简单地越过用户 B 的确认操作（步骤 3 和步骤 4）。

●使用零知识对话证明的冒名诈骗防止法●

那么，怎么才能防止冒名诈骗呢？为此，图 4.12 所表示的确认操作必须进行复杂化处理。用户 B 收到了发送人发来的 y_A 和 z_A，将或是 0 或是 1 的值发送，与其对应返回的值进行检验，来确认发送人是否使用了正确的程序。

根据这个道理，只要原来的素数不被暴露，就能保证信息的零知识性、真实性和防止冒名诈骗的发生。

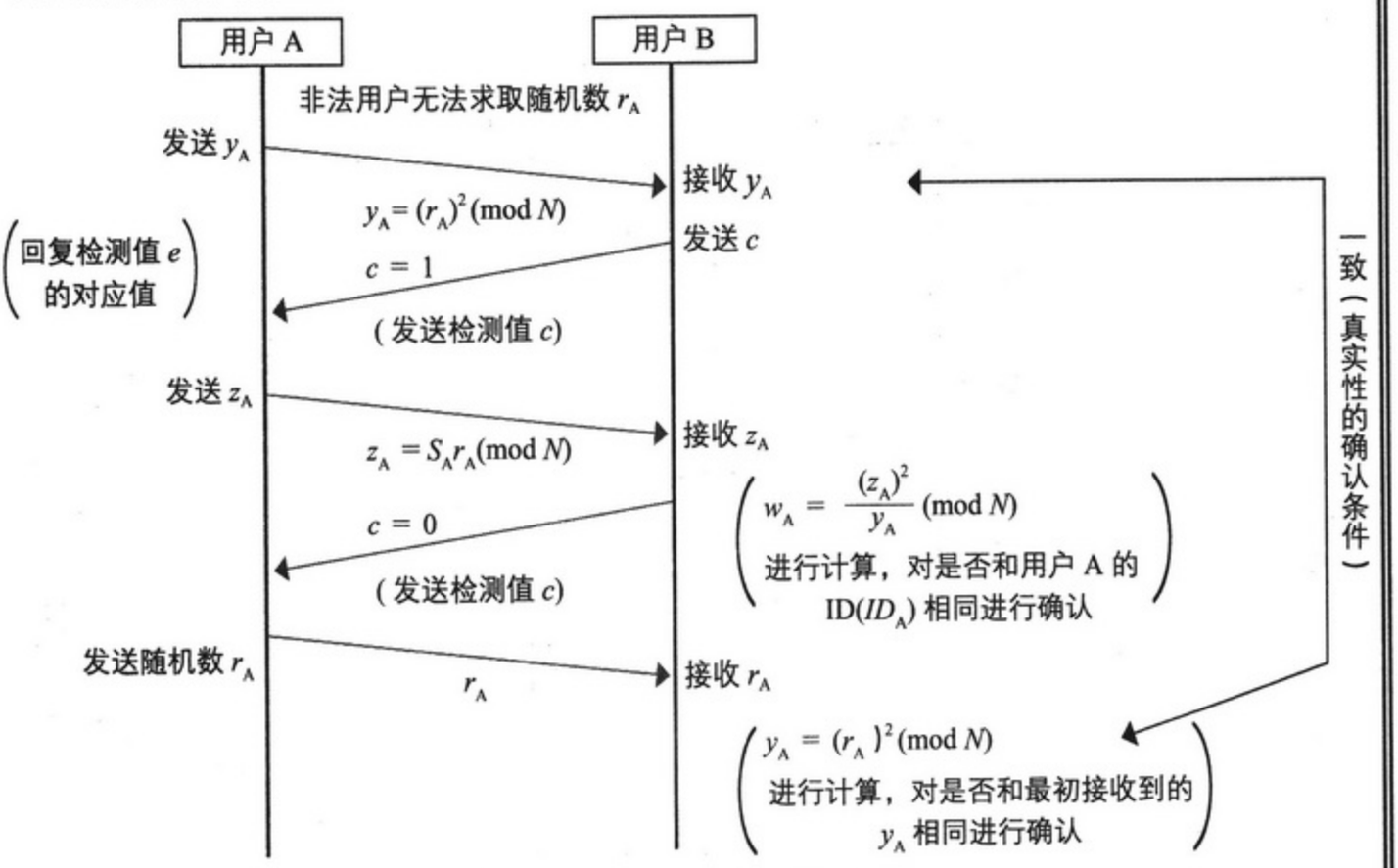


图 4.12　防止冒名诈骗真实性的确认程序

❀补充说明❀

◎ 拟似随机数和密码的安全性

随机数就是没有次序（random）的数字列，是起到维护安全性的基础设施技术的一种。比如说，在公开钥匙密码的情况下，进行加密和解密时，使用的钥匙就是用随机数做成的。因为可能会出现每次都使用同一个钥匙进行解密的情况，所以，要在使用公开钥匙密码时，生成新的钥匙来提高安全性。要是被人知道了随机数，很可能会出现密码被破解、金钱被盗取、个人信息被泄漏等问题。所以要设置不能被破解的密码，就要积极利用随机数所拥有的不可预测性（不能根据过去的数字列，计算出现在的数的性质）。

随机数和拟似随机数，以及最近备受关注的物理随机数（真性随机数），都有很大的区别。在这里，因为拟似随机数是由一定的计算式生成的，它有周期性，并且会出现相同的模式，所以不能成为“完全的没有次序的数字列”。因此，能推断出随机数，就有了破坏安全性的风险。具有代表性的拟似随机数生成器有线形合同法、平方取中法、M 系列、BBS（Blum-Blum-Shub）法、使用单向 Hash 函数的方法、密码的使用方法，等等。

另外，物理随机数是以自然界的物理现象为基础生成的随机数，可以实现完全的没有次序的数字的列，永远地持续生成没有次序的数字的列。比如说，在半导体回路中电流通过的时候，从发出的杂音中就能生成随机数。预想今后，为了构筑坚固的维护安全性的基础设施，应用物理随机数的领域会更多。

◎ PGP

取 Pretty Good Privacy 的每个单词第一个字母进行排列而成，直译就是“非常好的隐私”的意思，它是在 1990 年的时候，菲利普•齐默曼（Philip R. Zimmermann）开发出来，并得到广泛使用的软件。

PGP 几乎拥有现代密码软件所有的功能。也就是说，通用钥匙密码（AES、3-DES 等）、公开钥匙密码（RSA、ElGamal 等）、电子签名（RSA、DSA）、单向 Hash 函数（MD5、SHA-1、RIPEMD-160 等）、证书的制作等功能，都可以实现。

◎ SSL/TLS

它是通过在网上进行购物等时候，使用的通信协议（进行通信前的决定），来检验通信内容的认证和真实性时使用的电文认证代码。例如，用 Web 浏览器发送信用卡号码时，要准备作为将通信进行加密的协议 SSL（Secure Socket Layer）或者是 TLS（Transport Layer Security），将号码的交换进行加密，可以防止被盗取。另外，对于 SSL/TLS 的通信来说，URL 不是 http://，而是由 https:// 开始的。

接下来，对于发送电子邮件的 SMTP（Simple Mail Transfer Protocol）和对于接收电子邮件的 POP3（Post Office Protocol）的协议也可以用 SSL/TLS 进行加密来保护。

◎ 量子密码

被称为是绝对安全的密码。通常的光通信的 1bit 按照脉冲计算，包含光的最小单位（光子）1 万个以上。量子密码是记载了 1 个光子为 1bit 的信息，用光子的偏光状态（向电磁波的震动）来区别 0 和 1。如果这样做的话，就无法分解光子并盗取光子进行观测，因为光子的偏光状态可以变化，就可以知道光子被盗取了（量子力学中保证了“盗取的不可能性”）。这种“盗取的不可能性”和加密钥匙只能使用一次就换掉的“One time pad 的解密不可能性”进行组合，作为实现绝对安全的密码，向着实用化方向加速发展。

◎ 活体认证

活体认证是指可以利用个人所特有的数据（例如，指纹、静脉、面容、虹膜（眼球的茶色部分）、掌形（手掌的形状）、DNA（遗传因子）等）来进行对本人身份的认证。像我们生活中的事物有 ATM（自动提款机）和出入时认证本人身份的经脉认证系统，还有用“手指”或“手掌”进行身份认证的方法。

◆ 参考文献 ◆

・三谷政昭『やり直しのための工業数学――情報通信と信号解析――暗号、誤り訂正符号、積分変換』（CQ 出版）2000
・結城浩『暗号技術入門』（ソフトバンククリエイティブ）2003
・辻井重男『暗号と情報社会』（文藝春秋）1999
・岡本龍明，山本博資『現代暗号』（産業図書）1997
・伊藤正史『暗号理論』（ナツメ社）2003
・若林宏『よくわかる最新暗号技術の基礎と仕組み』（秀和システム）2005
・サイモン・シン，青木薫訳『暗号解読』（新潮社）2001
・セアラ・フラナリー，デイヴィッド・フラナリー，亀井よし子訳『16 歳のセアラが挑んだ世界最強の暗号』（NHK 出版）2001
・辻井重男『暗号 ポストモダンの情報セキュリティ』（講談社）1996
・一松信『暗号の数理』（講談社）2005
・ジョセフ・H・シルヴァーマン，鈴木治郎訳『はじめての数論』（ピアソン・エデュケーション）2001
・ベネディクト・グロス，ジョー・ハリス，鈴木治郎訳『数のマジック』（ピアソン・エデュケー ション）2005
・鷲浩二『Excel で学ぶ暗号技術入門』（オーム社）2006
・神保雅一監修，イオタゼミ著『なるほどナットク！暗号がわかる本』（オーム社）2004

※提示：怪盗希芙最后留下的密码，

00001011 00000110 00000110 00000001 00010111 00000111 00001010

其实是英文“Liberty”（自由）的编码

01101100 01101001 01100010 01100101 01110010 01110100 01111001

经过异或运算，成为了

01100111 01101111 01100100 01100100 01100101 01110011 01110011

按照日本准则标准（JIS），以上密码即“goddess”，也就是“女神”。

（O-3661.0103）
责任编辑：唐　璐　赵丽艳
责任制作：董立颖　魏　谨
封面制作：铭轩堂设计：13671110894 昭志辉瑞　轩雅静氣

用漫画这种形式讲数学、物理和统计学，十分有利于在广大青少年中普及科学知识。

周恩来、邓颖超秘书，周恩来邓颖超纪念馆顾问
中日友好协会理事，《数理天地》顾问，全国政协原副秘书长

用漫画和说故事的形式讲数学，使面貌冷峻的数学变得亲切、生动、有趣，使学习数学变得容易，这对于提高全民的数学水平无疑是功德无量的事。

《数理天地》杂志社　社长 总编
"希望杯"全国数学邀请赛组委会 命题委员会主任　周國鎮

用漫画的形式，讲解日常生活中的数学、物理知识，更能让大家感受到数学殿堂的奥妙与乐趣。

《光明日报》 原副总编辑
中华炎黄文化研究会 常务副会长　鲁諍

科学漫画是帮助学习文科的人们用形象思维的方式掌握自然科学的金钥匙。

中国人民大学外语学院日语专业 主任
大学日语教学研究会 会长　成同社

在日本留学的时候，我在电车上几乎每次都能看到很多年轻的白领看这套图书，经济实惠、图文并茂、浅显易懂，相信这套图书的中文版也一定会成为白领们的手中爱物。

大连理工大学 能源与动力学院 博士 副教授　曹建東

我非常希望能够在书店里看到这样的书：有人物形象、有卡通图、有故事情节，当然最重要的还有深厚的理工科底蕴。我想这样的书一定可以大大提升孩子们的学习兴趣，降低他们对于高深的理工科知识的恐惧感。

北京启明星培训学校 校长

书中的数学知识浅显实用，漫画故事的形式使知识贴近生活，概念更容易理解。

北京大学 数学科学学院 博士　张磊

上架建议：科普/漫画
ISBN 978-7-03-025320-0
9 787030 253200 >
定价：29.80元

科学出版社 东方科龙
http://www.okbook.com.cn
zhaoliyan@mail.sciencep.com