《应用密码学》课程作业答案

野鸡出版社

1.0 版 序 言

这是四川大学网络空间安全学院《应用密码学》课程的作业答案[1]。大概应该也许可以带给学弟学妹一些作业上的帮助，当然，这也可以作为期末复习的资料[2]。本资料不包含课程实践、题库[3]。

ZJX

2021 年 6 月 8 日

于四川大学多学科交叉创新大楼 4 楼实训摸鱼

Sean 数字签名者：Sean 日期：2021.06.08 23:47:11 +08'00'

[1]不是参考答案，这是我一年前做的 hhh

[2]因为这是当时作业的原件，可能会有一些小问题，而我现在也没有时间进行进行完善了，所以某些地方的粗糙之处还请见谅

[3]我原本打算放上来的，但想了想，还是算了叭。课程实践调 C 的 Bug 真的很搞人，但是网上有很多参考资料 (比如 `github.com`)；题库什么的就自己去收集咯

目 录

1 数论基础复习

1. **求欧拉函数 $\varphi(98)$ 和 $\varphi(23)$ 。**

$$\varphi(98) = \varphi(2)\varphi(49) = \varphi(7^2) = 6 \cdot 7 = 42$$

$$\varphi(23) = 23 - 1 = 22$$

2. **设有限域 $\mathbf{GF}(2^8)$ 的不可约多项式为 $p(x) = x^8 + x^4 + x^3 + x + 1$，写出多项式 $A(x) = x^7 + x^4 + x^3 + x^2 + x + 1$，$B(x) = x^6 + x^4 + x^2 + x + 1$ 的二进制表示，并求 $\mathbf{GF}(2^8)$ 上的多项式加法和乘法 $A(x) \oplus B(x)$ 与 $A(x) \otimes B(x)$。**

$p(x) = x^8 + x^4 + x^3 + x + 1$ 可表示为 (100011011)

则 $A(x) = x^7 + x^4 + x^3 + x^2 + x + 1$ 可表示为 (10011111)

则 $B(x) = x^6 + x^4 + x^2 + x + 1$ 可表示为 (01010111)

$$\begin{aligned} A(x) + B(x) &= (10011111) + (01010111) \\ &= (11001000) \end{aligned}$$

$$\begin{aligned} A(x) \cdot B(x) &= (10011111) \cdot (01010111) \\ &= (x^7 + x^4 + x^3 + x^2 + x + 1) \cdot (x^6 + x^4 + x^2 + x + 1) \\ &= x^{13} + x^{11} + x^{10} + 2x^9 + 3x^8 + 3x^7 + 3x^6 + 3x^5 + 4x^4 + 3x^3 + 3x^2 + 2x + 1 \\ &\equiv x^{13} + x^{11} + x^{10} + x^8 + x^7 + x^6 + x^5 + x^3 + x^2 + 1 \\ &\equiv (x^8 + x^4 + x^3 + x + 1) \cdot (x^5 + x^3 + x^2 + x) + x^3 + x^2 + x + 1 \\ &\equiv x^3 + x^2 + x + 1 \\ &\equiv (00001111) \pmod{p(x)} \end{aligned}$$

3. **计算:(1) 3201 mod 11 (2) 9541432 mod 17**

由于 11 是质数，所以 $3^{\varphi(11)} \equiv 3^{10} \equiv 1 \pmod{11}$

$$3^{201} \equiv 3^{20 \cdot 10+1} \equiv 3 \pmod{11}$$

由于 17 是质数，所以 $954^{\varphi(17)} \equiv 954^{16} \equiv 2^{16} \equiv 1 \pmod{17}$

$$954^{1432} \equiv 2^{89 \cdot 16+8} \equiv 2^8 \equiv 2^{4^2} \equiv 1 \pmod{17}$$

4. **用 Eculid 算法求 gcd(180, 252)，并表示为 180 和 252 这两个数的带整系数线性组合**

$$52 = 180 \cdot 1 + 72$$
$$180 = 72 \cdot 2 + 36$$
$$72 = 36 \cdot 2 + 0$$

则 $GCD(252, 180) = 36$, 对上面的序列进行代换有：

$$180 = (252 - 180) \cdot 2 + 36$$
$$180 = 252 \cdot 2 - 180 \cdot 2 + 36$$
$$36 = 3 \cdot 180 - 2 \cdot 252$$

5. **求解同余方程组** $\begin{cases} 13x \equiv 4 \pmod{99} \\ 15x \equiv 56 \pmod{56} \end{cases}$

由于 99 与 101 互素

由

$$13x \equiv 4 \pmod{99}$$

运用

$$x \equiv \frac{b}{(a,m)} \cdot [(\frac{a}{(a,m)})^{-1} \mod \frac{m}{(a,m)}] + t \cdot \frac{m}{(a,m)} \mod m \ \ t = 0, 1, 2 ... (a,m) - 1$$

有

$$x \equiv 46 \pmod{99}$$

由

$$15x \equiv 56 \pmod{101}$$

有

$$x \equiv 98 \pmod{101}$$

则得到

$$\begin{cases} x \equiv 46 \pmod{99} \\ x \equiv 98 \pmod{101} \end{cases}$$

由于 $GCD(99, 101) = 1$，且中国剩余定理

$$\begin{cases} x \equiv b_1 \mod m_1 \\ x \equiv b_2 \mod m_2 \\ \qquad\qquad ... \\ x \equiv b_n \mod m_n \end{cases} m_i \quad \prod m_i = m \quad x \equiv \sum b_i M_i M_i' \pmod{m}$$

得到

$$x \equiv 7471 \pmod{9999}$$

6. **计算:(1) 5403 mod 13 (2) −234 mod 12 (3) −12 mod 234**

（1）5403=13*415+8 ANS：8

（2）-234=12*（-19）-6 ANS：6

（3）ANS：222

7. **写出 n=28，33 和 35 在 Zn 上所有可逆元。**

$$\{x|x \in [1,27], x \in \mathbf{N}^*, x = 2,4,7,8,9,10,12,14,16,18,20,21,22,24,26\}$$

$$\{x|x \in [1,32], x \in \mathbf{N}^*, x = 3,6,9,11,12,15,18,21,22,24,27,30\}$$

$$\{x|x \in [1,34], x \in \mathbf{N}^*, x = 5,10,14,15,20,21,25,28,30\}$$

8. **利用扩展的欧几里德算法计算: (1)17-1 mod 101; (2)357-1 mod 1234; (3)3125-1 mod 9987**

$$\begin{aligned} 101 &= (5) * 17 + 16 \\ 17 &= (1) * 16 + 1 \\ 16 &= (16) * 1 + 0 \end{aligned}$$

反向代入有 $1 = (-1) * 101 + (6) * 17$，则 $17^{-1} \equiv 6 \pmod{101}$

$$\begin{aligned} 1234 &= (3) * 357 + 163 \\ 357 &= (2) * 163 + 31 \\ 163 &= (5) * 31 + 8 \\ 31 &= (3) * 8 + 7 \\ 8 &= (1) * 7 + 1 \\ 7 &= (7) * 1 + 0 \end{aligned}$$

反向代入有 $1 = (46) * 1234 + (-159) * 357$，则 $357^{-1} \equiv 1075 \pmod{1234}$.

$$
\begin{aligned}
9987 &= (3) * 3125 + 612 \\
3125 &= (5) * 612 + 65 \\
612 &= (9) * 65 + 27 \\
65 &= (2) * 27 + 11 \\
27 &= (2) * 11 + 5 \\
11 - (2) * 5 + 1 \\
5 &= (5) * 1 + 0
\end{aligned}
$$

反向代入有 $1 = (-577) * 9987 + (1844) * 3125$，则 $3125^{-1} \equiv 1844 \pmod{9987}$.

2 古典密码

1. **请指出一般单表替代密码的明文空间、密文空间和密钥空间各是什么?**

   明文空间与密文空间是 26 个英文字母的集合;

   而密钥空间是 $\{\pi : \mathbf{Z}_{26} \to \mathbf{Z}_{26} | \pi \quad \}$。

2. 使用仿射密码 (mod 26) 加密信息，然后再使用另一个仿射密码 (mod 26) 加密密文，请问这样做比只使用一次仿射密码有优势么? 为什么?

   设 k_1, k_2 与 26 互质

   $$m_1 \cdot k_1 + e_1 \equiv m_2 \pmod{26}$$

   $$m_2 \cdot k_2 + e_2 \equiv m_3 \pmod{26}$$

   代入有:

   $$(m_1 \cdot k_1 + e_1) \cdot k_2 + e_2 \equiv m_1 k_2 k_2 + e_1 k_2 + e_2 \equiv m_3 \pmod{26} \tag{3}$$

   在 (1)(2) 中的 k_1, k_2 与 26 互质但 $\varphi(26) = 12$，且 $e_1, e_2 \in [0, 25]$，也就是说单个仿射加密的密钥空间 $\#K_1 = 12 * 26 = 312$，想要暴力破解是非常容易的。

   但如果使用两重仿射变换有，如 (3) 所示，实际上，$k_1 \cdot k_2$ 的值也只有 12 种情况，$e_1 k_2 + e_2$ 也只有 26 种情况...所以最后的 $\#K_2 = 12 * 26$ 还是 312，双重仿射**没有优势**，甚至更低效。

3. **试用 Playfair 算法加密明文 "Network security is very important"，设其密钥为 security.**

   5x5 字母表:

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

   (4)

   `Network security is very important`

   得到: `Mcyvpud rcurstya te qcsa tnlpuybmy`

4. **假设 M = C = Z26，Hill 密码加密密钥矩阵为** $\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$ **，试加密消息 "pay more money"。**

pay more money：

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \begin{pmatrix} 15 & 12 & 4 & 13 \\ 0 & 14 & 12 & 4 \\ 24 & 17 & 14 & 24 \end{pmatrix} \equiv \begin{pmatrix} 11 & 7 & 4 & 19 \\ 13 & 3 & 22 & 17 \\ 18 & 11 & 12 & 22 \end{pmatrix} \pmod{26}$$

得到

$$\begin{pmatrix} 15 & 12 & 4 & 13 \\ 0 & 14 & 12 & 4 \\ 24 & 17 & 14 & 24 \end{pmatrix} = \begin{pmatrix} l & h & e & t \\ n & d & w & r \\ s & l & m & w \end{pmatrix}$$

即**lns hdl ewm trw**。

5. **若已知用某仿射密码 (mod26) 加密明文字符 e、h 的对应密文字符为 F、W，试求出该仿射密码的密钥 k。**

$$\begin{cases} 4 \cdot k_1 + e_1 \equiv 5 \\ 7 \cdot k_1 + e_1 \equiv 22 \end{cases} \pmod{26}$$

得到：

$$3 \cdot k_1 \equiv 17 \pmod{26}$$
$$3^{-1} \equiv 9 \pmod{26}$$

得到：

$$\begin{cases} k_1 \equiv 9 \cdot 17 \equiv 23 \\ e_1 \equiv 17 \end{cases} \pmod{26}$$

6. **使加密算法成为对合函数的密钥 k 称为对合密钥，即对于任意明文 m，有 Ek (Ek (m)) = m。**

(1) 试给出定义在 Z26 上的移位密码体制中的对合密钥。(2) 一般而言，对于移位密码，设明文字母表和密文字母表含有 n 个字母，n 为 2 的正整数, 求出其对合密钥 k。

$$m_1 + 2k_1 \equiv m_1 \pmod{26}$$

$$k_1 \equiv 13\text{或}0$$

$$k \equiv \begin{cases} \frac{n}{2}\text{或 } 0,\text{n 为偶数} \\ 0,\text{n 为奇数} \end{cases} \pmod{n}$$

7. **假设明文是 Network security is very important，按密钥 =(4,3,5,1,2) 进行周期置换加密。**

   Netwo rksec urity isver yimpo rtant

   $$= (4, 3, 5, 1, 2)$$

   得到wtoNe escrk tiyur evris pmoyi natrt

```python
while(1):
  s = input("> ").split(' ')
  for i in s:
    print(i[3]+i[2]+i[4]+i[0]+i[1],end=' ')
```

8. **已知 Hill 密码中的明文分组长度为 2，密钥 K 是 Z26 上的一个 2 阶可逆方阵。假设明文 FRIDAY 所对应的密文为 PQCFKU，试求密钥 K。**

   FRIDAY 对应 $(5, 17, 8, 3, 0, 24)$，其以两个分组为：

   $$\mathbf{FRIDAY} = \begin{pmatrix} 5 & 8 & 0 \\ 17 & 3 & 24 \end{pmatrix}$$

   而设密钥 K 为:

   $$K = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

   PQCFKU 对应 $(15, 16, 2, 5, 10, 20)$ 有

   $$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 5 & 8 & 0 \\ 17 & 3 & 24 \end{pmatrix} \equiv \begin{pmatrix} 15 & 2 & 10 \\ 16 & 5 & 20 \end{pmatrix} \pmod{26}$$

   有

   $$\begin{cases} 5a + 17b \equiv 15 \\ 5c + 17d \equiv 16 \\ 8a + 3b \equiv 2 \\ 8c + 3d \equiv 5 \\ 24b \equiv 10 \\ 24d \equiv 20 \end{cases} \pmod{26}$$

   解得

   $$\begin{cases} a \equiv 7 \\ b \equiv 8 \\ c \equiv 19 \\ d \equiv 3 \end{cases} \pmod{26}$$

则

$$K = \begin{pmatrix} 7 & 8 \\ 19 & 3 \end{pmatrix}$$

9. **设维吉尼亚密码的密钥为 “DILIGENCE”，试对消息 “we are cryptographer” 进行加密。**

   wearecryptographer 对应 $(22, 4, 0, 17, 4, 2, 17, 24, 15, 19, 14, 6, 17, 0, 15, 7, 4, 17)$，DILIGENCE 对应 $(3, 8, 11, 8, 6, 4, 13, 2, 4)$

w	e	a	r	e	c	r	y	p	t	o	g	r	a	p	h	e	r
22	4	0	17	4	2	17	24	15	19	14	6	17	0	15	7	4	17
3	8	11	8	6	4	13	2	4	3	8	11	8	6	4	13	2	4
25	12	11	25	10	6	4	0	19	22	22	17	25	6	19	20	6	21
z	m	l	z	k	g	e	a	t	w	w	r	z	g	t	u	g	v

```
import bidict
dic = {'a':0,'b':1,'c':2,'d':3,'e':4,'f':5,'g':6,'h':7,'i':8,'j':9,'k':10,'l':11,'m':12,'n':13,
'o':14,'p':15,'q':16,'r':17,'s':18,'t':19,'u':20,'v':21,'w':22,'x':23,'y':24,'z':25}
'''
s = input("> ").lower()

for i in range(len(s)):
    print(dic[s[i]],end=',')

print()
'''
bid = bidict.bidict(dic)
s = input("> ").split(' ')
for i in s:
    print(bid.inverse[int(i)],end=',')
```

3 对称密码

1. **分组密码中的 Feistel 结构主要有什么特点？**

   1）

   Feistel 结构是典型的迭代密码, Feistel 结构保证了加密过程的可逆性。加解密相似是 Feistel 型密码的实现优点。但另一方面, Feistel 型密码的扩散要慢一些。例如, 算法至少需要两轮才能改变输入的每一位。这就导致 Feistel 型密码的轮数较多, 一般大于等于 16；

   密文的任何一位均与明文和密钥的每一位相关；而明文和密钥的任何一位的变化将影响到尽量多的密文。算法至少需要两轮才能改变输入的每一位；“加解密运算相同” 是是 Feistel 型密码的实现特点。但同时，Feistel 型密码的扩散要慢一些。如算法至少需要两轮才能改变。

   2）

   网络结构特点：

   输入为 2w 比特的分组（称为左半分组和右半分组）；

   函数 f 用于产生扩散和混淆作用，但并不要求为可逆函数；

   ki 为由密码密钥 K 生成的子密钥 (轮密钥) ；

   Feistel 网络由 n 个基本结构单元构成（但子密钥 ki 互不相同）

   3）

   分组大小：越大安全性越高，但运算速度越慢，现在多为 128bit ；

   密钥长度：越大安全性越高，但运算速度越慢，常为 128bit ；

   循环次数：越多安全性越高，但运算开销越大，常见采用 16 次；

   子密钥产生算法：算法越复杂，密码分析越困难；

   轮函数 F：复杂性越高，密码分析越困难，并不要求为可逆函数。

2. **什么是分组密码的操作模式？有哪些主要的分组密码操作模式？其工作原理是什么？各有何特点？**

   分组密码的操作模式就是分组密码算法应用于数据或文件的加密的算法。

   - 主要有 ECB, CBC, CFB, OFB, CTR；

     ECB：电子密码本 ECB 模式 (Electronic Codebook Mode) 是最简单的模式，它直接利用加密算法分别对每个明文分组使用相同密钥进行加密。设明文分组序列为 M=M1M2…Mn，相应的密文分组序列为 C= C1C2…Cn

优点：

1. 简单；

2. 有利于并行计算；

3. 误差不会被传送；

缺点：

1. 不能隐藏明文的模式；

2. 可能对明文进行主动攻击；

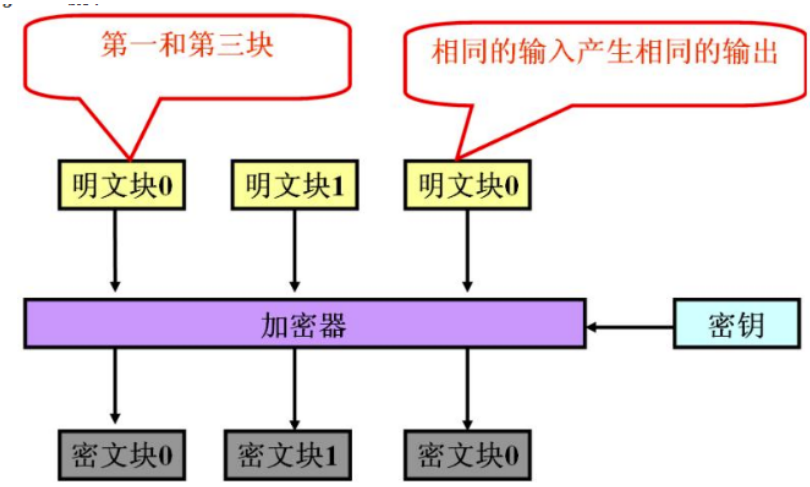

图 3.1: image-20200410180415668

CBC：CBC 模式 Cipher Block Chaining 模式 (密文分组链接模式), 在 CBC 模式中, 首先将明文分组与前一个密文分组进行 XOR 运算，然后再进行加密。

优点：

1. 不容易主动攻击, 安全性好于 ECB, 适合传输长度长的报文, 是 SSL、IPSec 的标准。

缺点：

1. 不利于并行计算；

2. 误差传递；

3. 需要初始化向量 IV

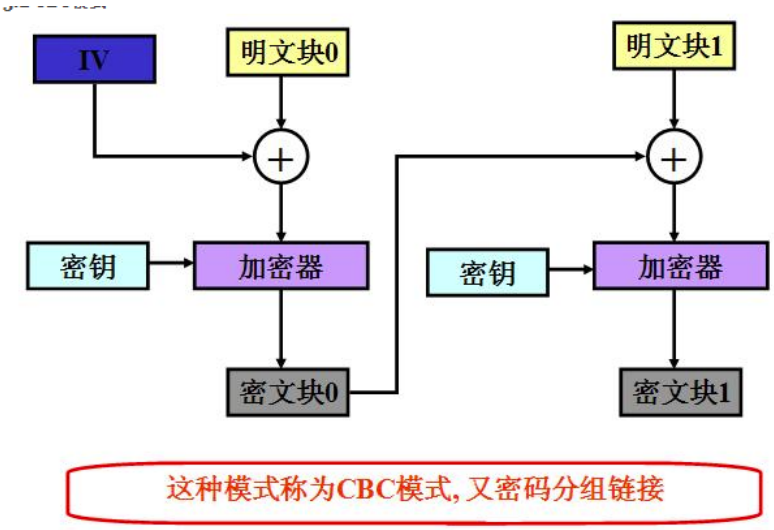

图 3.2: image-20200410180423389

CFB：CFB 模式的全称是 Cipher FeedBack 模式 (密文反馈模式)。在 CFB 模式中，前一个密文分组会被加密，然后与明文异或得到密文输出。

优点：

1. 隐藏了明文模式;
2. 分组密码转化为流模式;
3. 可以及时加密传送小于分组的数据;

缺点：

1. 不利于并行计算;
2. 误差传送：一个明文单元损坏影响多个单元;
3. 唯一的 IV;

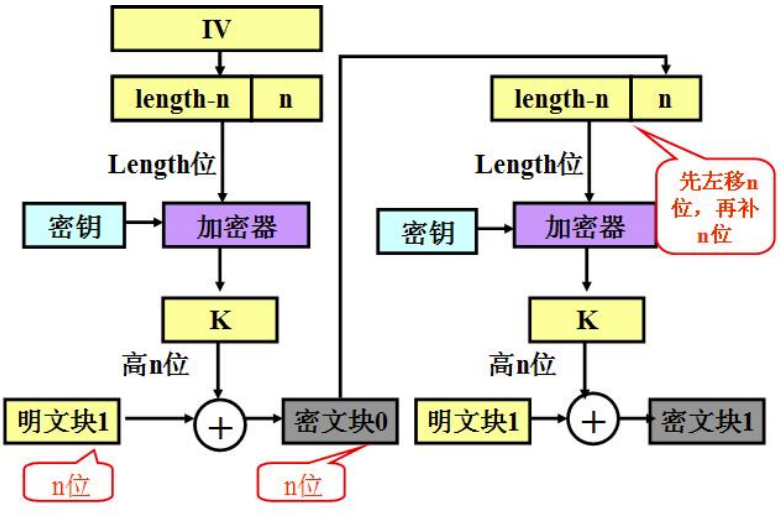

图 3.3: image-20200410180512257

OFB：OFB 模式的全称是 Output-Feedback 模式 (输出反馈模式)。OFB 模式并不是通过密码算法对明文直接进行加密的，而是通过将“明文分组”和“密码算法的输出”进行 XOR 来产生“密文分组”的。

优点：

1. 隐藏了明文模式;

2. 分组密码转化为流模式;

3. 可以及时加密传送小于分组的数据;

缺点：

1. 不利于并行计算;

2. 对明文的主动攻击是可能的;

3. 误差传送：一个明文单元损坏影响多个单元;

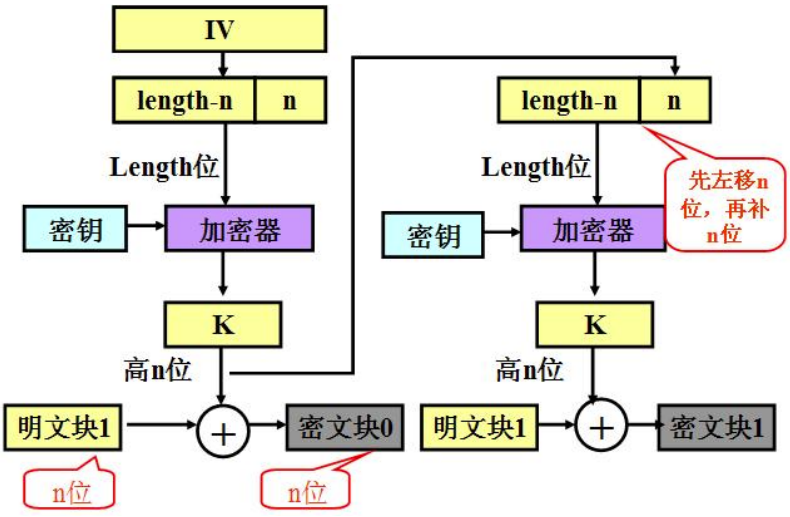

图 3.4: image-20200410180539440

CTR：CTR 模式的全称是 CounTeR 模式 (计数器模式)。CTR 模式中，每个分组对应一个逐次累加的计数器，并通过对计数器进行加密来生成密钥流。也就是说，最终的密文分组是通过将计数器加密得到的比特序列，与明文分组进行 XOR 而得到的。

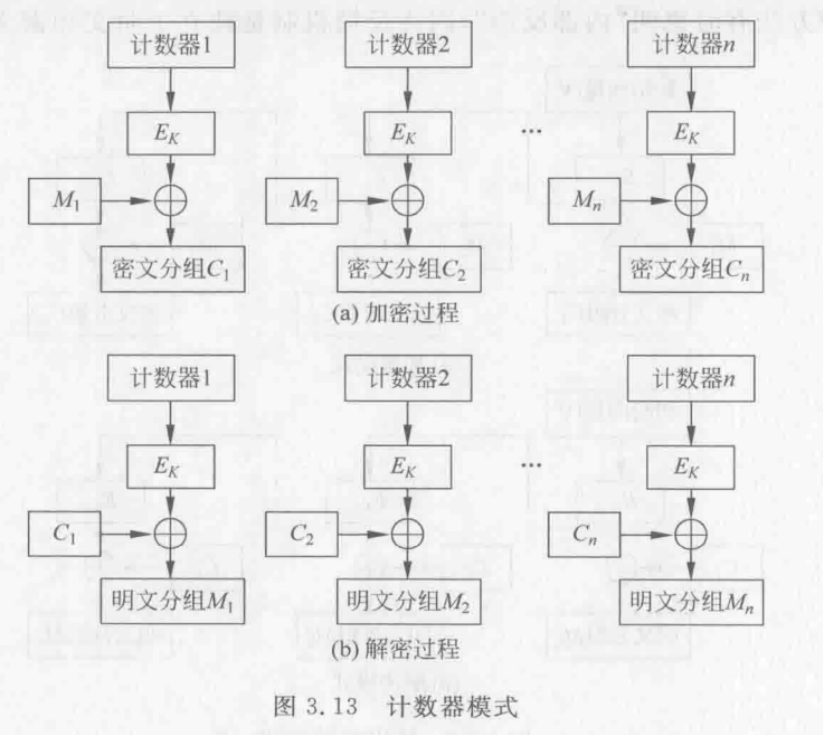

图 3.5: image-20200410181248452

总结如下：

模式	名称	优点	缺点	备注
ECB 模式	Electronic CodeBook 电子密码本模式	• 简单 • 快速 • 支持并行计算（加密、解密）	• 明文中的重复排列会反映在密文中 • 通过删除、替换密文分组可以对明文进行操作 • 对包含某些比特错误的密文进行解密时，对应的分组会出错 • 不能抵御重放攻击	不应使用
CBC 模式	Cipher Block Chaining 密文分组链接模式	• 明文的重复排列不会反映在密文中 • 支持并行计算（仅解密） • 能够解密任意密文分组	• 对包含某些错误比特的密文进行解密时，第一个分组的全部比特以及后一个分组的相应比特会出错 • 加密不支持并行计算	CRYPTREC 推荐；《实用密码学》推荐
CFB 模式	Cipher-FeedBack 密文反馈模式	• 不需要填充（padding） • 支持并行计算（仅解密） • 能够解密任意密文分组	• 加密不支持并行计算 • 对包含某些错误比特的密文进行解密时，第一个分组的全部比特以及后一个分组的相应比特会出错 • 不能抵御重放攻击	CRYPTREC 推荐
OFB 模式	Output-FeedBack 输出反馈模式	• 不需要填充（padding） • 可事先进行加密、解密的准备 • 加密、解密使用相同结构 • 对包含某些错误比特的密文进行解密时，只有明文中相对应的比特会出错	• 不支持并行计算 • 主动攻击者反转密文分组中的某些比特时，明文分组中相对应的比特也会被反转	CRYPTREC 推荐
CTR 模式	CounTeR 计数器模式	• 不需要填充（padding） • 可事先进行加密、解密的准备 • 加密、解密使用相同结构 • 对包含某些错误比特的密文进行解密时，只有明文中相对应的比特会出错 • 支持并行计算（加密、解密）	主动攻击者反转密文分组中的某些比特时，明文分组中相对应的比特也会被反转	CRYPTREC 推荐，《实用密码学》推荐

图 3.6: image-20200410181348910

3. **请证明 DES 结构的互补对称性，即 $\mathbf{DES}_{\overline{k}}(\overline{M}) = \overline{\mathbf{DES}_k(M)}$。**

   $hint: \overline{A}\,\overline{B} = A\,B \quad \overline{A\,B} = \overline{A}\,B = A\,\overline{B}$

设 $L_0,\ R_1$(不带上标'与'') **为明文消息，**

设 $L_1',\ R_1'$ 为

$$L_1' = \overline{R}_0$$

$$R_1' = \overline{L_0} \oplus f(\overline{R_0}, \overline{k_1}) = \overline{L_0 \oplus f(\overline{R_0}, \overline{k_1})}$$

而设 $L_1'',\ R_1''$（正常的 DES 加密）

$$L_1'' = R_0$$

$$R_1{}'' = L_0 \oplus f(R_0, k_1)$$

这里需要得到 $\overline{L_0 \oplus f(\overline{R_0}, \overline{k_1})}$ 与 $L_0 \oplus f(R_0, k_1)$ 的关系，考虑以下论述。

下证 $f(\overline{R_0}, \overline{k_1}) = f(R_0, k_1)$，

因为

$$f(R_0, k_1) = \mathbf{P}(\mathbf{S}(k_1 \oplus \mathbf{E}(R_0)))$$

$$f(\overline{R}_0, \overline{k_1}) = \mathbf{P}(\mathbf{S}(\overline{k}_1 \oplus \mathbf{E}(\overline{R}_0)))$$

又拓展函数的取反性质

$$\boxed{\overline{\mathbf{E}(R)} = \mathbf{E}(\overline{R})}$$

(6) 化为

$$f(\overline{R}_0, \overline{k_1}) = \mathbf{P}(\mathbf{S}(\overline{k}_1 \oplus \overline{\mathbf{E}(R_0)})) = \mathbf{P}(\mathbf{S}(k_1 \oplus \mathbf{E}(R_0))) = f(R_0, k_1)$$

将 (8) 的结论代入 (2)

$$R_1' = \overline{L_0 \oplus f(\overline{R_0}, \overline{k_1})} = \overline{L_0 \oplus f(R_0, k_1)} = \overline{R_1''}$$

更一般地，由轮密钥产生方式（左移取反的密钥仍然与正常的密钥互补），由类似 (1)(2)(3)(4) 的方式，显然之后

$$L_i' = \overline{L_i''},\ R_i' = \overline{R_i''} (i \geq 0)$$

对于 L_i', R_i' 型的加密方式，为明文取反，密钥取反，即 $\mathbf{DES}_{\overline{k}}(\overline{M})$ ；

对于 L_i'', R_i'' 型的加密方式（即正常 DES 加密），最后输出的是 $\mathbf{DES}_k(M)$ ，由 (10) 有

$$\mathbf{DES}_{\overline{k}}(\overline{M}) = \overline{\mathbf{DES}_k(M)}$$

4. **请用 DES 的 8 个 S 盒对 48 比特串 70a990f5fc36 进行压缩置换，用 16 进制写出每个 S 盒的输出。**

   原文：011100 001010 100110 010000 111101 011111 110000 110110

   位置：0 14 0 5 2 3 0 8 3 14 1 15 2 8 2 11

   原文：70a990f5fc36

   置换输出：$0b9158ad$

   详细过程：

   S1: 0f8368f7 0

   S2: 5bb9e556 b

   S3: 239121bc 9

   S4: 4603129fe 1

   S5: eab856f5 5

   S6: 5250887a 8

   S7: 60b336a8 a

   S8: af1a620d d

   即：$0b9158ad$

5. **设 DES 算法的 8 个 S 盒都是 S1, R0 = FFFFFFFF, K1 = 555555555555，均用十六进制表示，求：f(R0，K1) = ？**

   $f(R,K) = \mathbf{P}(\mathbf{S}(K \oplus \mathbf{E}(R)))$

   k = 0101 0101 0101 0101 0101 0101 0101 0101 0101 0101 0101 0101

   $\mathbf{E}(R)$ = 111111 111111 111111 111111 111111 111111 111111 111111

   $K \oplus \mathbf{E}(R)$ = 101010 101010 101010 101010 101010 101010 101010 101010 （2 5）

   $\mathbf{S}(K \oplus \mathbf{E}(R))$ = 0110 0110 0110 0110 0110 0110 0110 0110

   $\mathbf{P}(\mathbf{S}(K \oplus \mathbf{E}(R)))$ = 0100 0000 0111 0111 1001 0110 1011 1100

   = $407796BC$

6. **AES 的基本变换有哪些？简述其主要特点。**

   字节替代，列混合变换，行位移变换，钥加

   > 字节替代:

   字节代替变换 ByteSub( ) 是一个关于字节的非线性变换。S 盒变换是 AES 唯一的非线性变换，也是 AES 安全的关键。

（1 ）AES 使用 16 个相同的 S 盒

（2 ）AES 的 S 盒有 8 位输入 8 位输出，是一种非线性替换

S 盒第 1 步就是把各字节用其乘法逆元来代替，是一种非线性变换；系数矩阵每一行都含有 5 个 1, 说明输出中的每 1 位，都与输入中的 5 位相关；系数矩阵每一列都含有 5 个 1 ，说明改变输入中的任 1 位，将影响输出中的 5 位发生变化；相当于 DES 中的 S 盒，它为算法提供非线性变换。

> 列混合变换:

列混合变换 MixColumns() 对一个状态逐列进行变换，它将一个状态的每一列视为有限域 GF(2^8) 上的一个多项式。在列混合变换中，用一个固定的多项式 a(x) 乘以每一列所表示的多项式：

a(x)={03} x^3 +{01} x^2 +{01}x +{02}

对应 4 字节向量为 (03 01 01 02) ，模多项式为 (x^4 + 1) 。

> 行位移变换:

在行移位 (ShiftRows) 变换中，状态矩阵中的每一行将以字节为单位，循环右移不同的位移量：

第 0 行不动

第 1 行循环左移 1 个字节

第 2 行循环左移 2 个字节

第 3 行循环左移 3 个字节

> 钥加：

轮密钥加法变换 AddRoundKey( ) 简单地将输入阵列和一个轮密钥进行简单的按位异或（模 2 加）运算，轮密钥按顺序取自扩展密钥，而扩展密钥又是由原始工作密钥经过扩展后得到的。

算法利用外部输入密钥 K(密钥串的字数为 Nk), 通过密钥的扩展程序得到共计 4(Nr+1) 字的扩展密钥。它涉及如下三个模块: (1) 位置变换 (rotword)——把一个 4 字节的序列 [A,B,C,D] 变化成 [B,C,D,A]；(2)S 盒变换 (subword)——对一个 4 字节进行 S 盒代替；(3) 变换 Rcon[i]——Rcon[i] 表示 32 位比特字 [xi-1,00,00,00]。这里的 x 是（02），如 Rcon[1]=[01000000]；Rcon[2]=[02000000]；Rcon[3]=[04000000]……

AES 加密时的轮变换：

(1) 初始轮密钥加法；
(2) 中间轮轮变换；
(3) 最后轮轮变换。其中轮变换由前面四个不同的变换组成。

7. **计算四字节乘法** $(03\ 01\ 01\ 02) \otimes (0B\ 0D\ 09\ 9E)$ **模多项式** x^4+1。

$$(03\ 01\ 01\ 02) \oplus (0B\ 0D\ 09\ 9E)$$
$$(0011\ 0001\ 0001\ 0010) \oplus (00001011\ 00001101\ 00001001\ 10011110)$$

$$D = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} 9E \\ 09 \\ 0D \\ 0B \end{pmatrix}$$

因为

$$x \cdot B(x) = \begin{cases} lc(b_6b_5b_4b_3b_2b_1b_00), b_7 = 0 \\ (b_6b_5b_4b_3b_2b_1b_00) \oplus (00011011), b_7 = 1 \end{cases}$$

得到

$$\begin{pmatrix} d_0 = (02)(9E) + (03)(09) + (01)(0D) + (01)(0B) \equiv 00111010 \\ d_1 = (01)(9E) + (02)(09) + (03)(0D) + (01)(0B) \equiv 10010000 \\ d_2 = (01)(9E) + (01)(09) + (02)(0D) + (03)(0B) \equiv 10010000 \\ d_3 = (03)(9E) + (01)(09) + (01)(0D) + (02)(0B) \equiv 10101011 \end{pmatrix} \pmod{x^4+1}$$

即

$$(ab\ 90\ 90\ 3a)$$

$$171x^3 + 144x^2 + 144x + 58$$

附 D 计算过程

$$
\begin{aligned}
&00100111\\
&00011011\\
&00001101\\
&00001011\\
=\ &00111010.\\
&10011110\\
&00010010\\
&00010111\\
&00001011\\
=\ &10010000.\\
&10011110\\
&00001001\\
&00011010\\
&00011101\\
=\ &10010000.\\
&10111001\\
&00001001\\
&00001101\\
&00010110\\
=\ &10101011
\end{aligned}
$$

4 非对称密码

1. **设通信双方使用 RSA 加密体制，接收方的公开密钥是 (5，35)，接收到的密文是 11，明文是多少？**

   肉眼直接分解双素数，由 $35 = 5 * 7$, 得 $\varphi(35) = 24$，由扩展 Euclid 有私钥为 $d = 5$ ，由于 $c = 11$ ，所以 $m \equiv 11^5 \pmod{35} \rightarrow m \equiv 16$ ，即明文消息 $m = 16$ 。

2. **在 RSA 密码体制中，如果 n = 21，取公钥 e = 5，如果明文消息为 m = 8，试用该算法加密 m 得到密文 c，并解密进行验证。**

   $c \equiv 8^5 \pmod{21} \rightarrow c \equiv 8 \pmod{21}$，

   分解双素数 $21 = 3 * 7 \rightarrow \varphi(21) = 12$，由扩展 Euclid 有私钥为 $d = 5$ ，$m \equiv 8^5 \pmod{21} \rightarrow m \equiv 8 \pmod{21}$ ，即明文消息 $m = 8$ ，验证完成。

3. **（重点）在 ElGamal 密码体制中，设素数 p = 71，本原元 = 7，(1) 如果接收方 B 公钥 yB = 3，发送方 A 选择的随机整数 k = 2，求明文 m1 = 30 所对应的密文 c = ？(2) 如果发送方 A 选择另一随机整数 k'，使对明文 m1 = 30 加密后的密文 c' = (59，c2)，求 c2 = ？(3) 如果发送方 A 用相同的 k = 2 加密另外一个明文 m2，加密后的密文为 c2 = (49，13)，求 m2 = ？**

   (1) 按照 ElGamal 密码体制 $c_1 = \alpha^2 = 49$，$c_2 = 30 * y_B^2 \equiv 57 \pmod{71}$，则 $= (49, 57)$；

   (2) $c_1 = \alpha^k \equiv 59 \pmod{71} \rightarrow k = 3$，则 $c_2 = 30 * y_B^3 \equiv 29 \pmod{71}$；

   (3) $c_2 = m_2 y_B^2 = m_2 \alpha^{2x_B} \equiv 13 \pmod{71}$，由于 $y_B^{-1} \equiv 24 \pmod{71}$，所以 $m_2 = m_2 y_B^2 y_B^{-2} \equiv 33 \pmod{71}$。

4. **（重点）利用椭圆曲线实现 ElGamal 密码体制，设椭圆曲线 E11(1，6)，基点 G = (2，7)，接收方的秘密密钥 dA = 5。求：(1) 接收方 A 的公开密钥 PA；(2) 发送方 B 欲发送消息 Pm = (7，9)，选择随机数 k = 3，求密文 Cm = ？(3) 完成接收方 A 解密 Cm 的计算过程。**

   (1) 由于 $y^{-1} \equiv 8, 2^{-1} = 6$，$\lambda \equiv \frac{3x^2+a}{2y} \equiv 8 \pmod{11}$ ，$x' \equiv \lambda^2 - 2x \pmod{p}, y' \equiv \lambda(x - x') - y \pmod{p} \rightarrow 2G = (5, 2) \rightarrow \lambda' = 8, 2G + 2G = (10, 2)$ $P_A = d_A G = 4G + G = (3, 6)$

   (2) $c_1 = kG = 3G = 2G + G = (8, 3)$,$c_2 = P_m + kP_A = (3, 5) \rightarrow c_m = \{(8, 3), (3, 5)\}$

   (3) $5c_1 = 2c_1 + 2c_1 + c_1 = 4c_1 + c_1 = (5, 2) \rightarrow (5c_1)^{-1} = (5, -2)$，所以 $P_m = c_2 + (d_A c_1)^{-1} = (3, 5) + (5, -2) = (7, 9)$

5. **分别用孙子定理和平方-乘法算法计算：** $7^{560} \pmod{527}$

   孙子定理：

$$\begin{cases} lc7^{560} \equiv b_1 \pmod{17} \\ 7^{560} \equiv b_2 \pmod{31} \end{cases}$$

   由费马小定理

$$\begin{cases} lc7^{16} \equiv 1 \pmod{17} \\ 7^{30} \equiv 1 \pmod{31} \end{cases}$$

   所以

$$\begin{cases} lc7^{560} \equiv 1 \pmod{17} \\ 7^{560} \equiv 7^{20} \equiv 5 \pmod{31} \end{cases}$$

   由 CRT 有：

$$7^{560} \equiv 31 * 11 + 5 * 17 * 11 \equiv 222 \pmod{527}$$

   平方-乘法：

   $7^2 \equiv 49 \pmod{527}$

   $7^{2^2} \equiv 293 \pmod{527}$

   $7^{2^3} \equiv 475 \pmod{527}$

   $7^{2^4} \equiv 69 \pmod{527}$

   $7^{2^5} \equiv 18 \pmod{527}$

   $7^{2^6} \equiv 324 \pmod{527}$

   $7^{2^7} \equiv 103 \pmod{527}$

   $7^{2^8} \equiv 69 \pmod{527}$

   $7^{2^9} \equiv 18 \pmod{527}$

$$7^{560} \equiv 7^{2^9+2^5+2^4} \equiv 18 * 18 * 69 \equiv 222 \pmod{527}$$

6. **RSA 公开密钥加密系统中，某用户选择 p = 43，q = 59，并取公开密钥 e = 13，请计算：(1) 私有密钥中的指数 d = ？(2) 在 Z26 空间中对明文 “public key encryptions” 加密，求出其密文数值序列**

   (1) $\varphi(43 * 59) = 2436$, 由扩展欧几里得定理有 $de \equiv 1 \pmod{2436} \rightarrow d = 937$

   (2) p u b l i c k e y e n c r y p t i o n s 得到 $(15, 20, 1, 11, 8, 2, 10, 4, 24, 4, 13, 2, 17, 24, 15, 19, 8, 14, 13$

   由加密 $c \equiv m^e \pmod{2537}$，得到 $c =$

   $(2116, 793, 1, 821, 156, 581, 2403, 140, 2130, 140, 2038, 581, 959, 2130, 2116, 2299, 156, 1616, 2038, 22$

7. **【讨论题】设系统中每个用户都有一对公私钥。假定用户 A 要把一个文件秘密发送给其他六个用户。若要求用户 A 不是给每个用户单独发送用不同用户公钥加密的秘密信息，而是给所有用户发送同一个仅含该文件的一个加密结果的消息，请给出一个实施方案。**

> 注意本题没有标准答案，我当时写的这个答案获得了较多的认可，但是老师好像没有理解下面这个方法 hhh

令用户的公钥为 (e_i, n_i)，n_i 互素，$i \in \{1,2,3,4,5,6\}$，

加密过程

使用 CRT 解如下方程

$$\begin{cases} x \equiv m^{e_1} \pmod{n_1} \\ x \equiv m^{e_2} \pmod{n_2} \\ x \equiv m^{e_3} \pmod{n_3} \\ x \equiv m^{e_4} \pmod{n_4} \\ x \equiv m^{e_5} \pmod{n_5} \\ x \equiv m^{e_6} \pmod{n_6} \end{cases}$$

得到解 $c \equiv x \pmod{n}$，其中 $n = \prod n_i$，将 CRT 的解 c 作为密文。

解密过程

由于 c 是由方程组得出的，所以 c 满足任意单个方程，如下式所示

$$c \equiv m^{e_i} \pmod{n_i}$$

所以，使用如下 $\mathbf{D}(x)$ 解密

$$m \equiv \mathbf{D}(c) \equiv c^{d_i} \equiv m^{e_i d_i} \pmod{n_i}$$

即每个用户 i 都可以恢复出消息 m。

5 散列函数与消息鉴别

1. **用 RSA 来构造一个散列函数。将输入 m 分为 k 个组，即 $m = m_1 m_2 ... m_k$，RSA 公钥为 (e, n)，定义散列函数：$h_i = (h_{i-1}^e \mod n)\ m_i\ i = 2, 3, ..., k$，其中 $h_1 = m_1$，最后一个分组的输出 h_k，即为输出的散列码。**

   试问如此构造的散列函数安全吗？为什么？

   安全的散列函数要求在知道 m 的情况下，也不能构造出 m' 使得其与 m 的散列值相等。

   发送方最后输出为

   $$h_k = (h_{k-1}^e \mod n)\ m_k$$

   攻击方构造序列 $m' = m'_1 m'_2 ... m'_{k-1}, m'_k$,按照上面方法得到的散列序列为 $H_1, H_2, \ldots, H_k$，同时使得

   $$m'_k = (H_{k-1}^e \mod n)\ m_k \oplus (h_{k-1}^e \mod n)$$

   那么在接收方校验的时候

   $$\begin{aligned} H_k &= (H_{k-1}^e \mod n)\ m'_k \\ &= (H_{k-1}^e \mod n)\ (H_{k-1}^e \mod n)\ m_k \oplus (h_{k-1}^e \mod n) \\ &= (h_{k-1}^e \mod n)\ m_k \\ &= h_k \end{aligned}$$

   这里通过构造实现了哈希碰撞，所以这个散列函数不安全。

2. **有很多散列函数是由 CBC 模式的分组加密技术构造的，其中的密钥为消息分组。例如将消息 M 分成分组 $M_1\ M_2\ ...\ M_n$ ，H_0 是初值，迭代关系为 $H_i = \mathbf{E}_{M_i}(H_{i-1})\ H_{i-1}$ ，$(i = 1\ 2\ ...\ n)$，散列值为 H_n ，其中 E 是分组加密算法。**

   （1）设 E 为 DES，已证明如果对明文分组和加密密钥都逐位取补，那么得到的密文也是原密文的逐位取补，即 $\mathbf{DES}_{\overline{k}}(\overline{M}) = \overline{\mathbf{DES}_k(M)}$ 。利用这一结论证明在上述散列函数中可以对消息进行修改（还可以改变 H）但却保持散列值不变。

   （2）若迭代关系改为 $H_i = \mathbf{E}_{H_{i-1}}(M_i)\ M_i$ ，$(i = 1\ 2\ ...\ n)$，证明仍可对其进行上述攻击。

   （1）将明文取反得到 $\overline{M}$，使用 $\overline{H_0}$，$H\_i = \mathbf{E}\_\{\overline{M_i}\}(H\_\{i-1\})\ H\_\{i-1\}$

   $$\begin{aligned} H'_1 &= \mathbf{DES}_{\overline{M_1}}(\overline{H_0})\ \overline{H_0} \\ &= \overline{\mathbf{DES}_{M_1}(H_0)}\ \overline{H_0} \\ &= \mathbf{DES}_{M_1}(H_0)\ H_0 \\ &= H_1 \end{aligned}$$

可以看到当明文取反，H_0 取反后可以对消息进行修改但却保持散列值不变。

（2）将明文取反得到 $\overline{M}$，使用 $\overline{H_0}$，$H\_i = \mathbf{DES}\{H\{i-1\}\}(M\_i)\ M\_i$

$$\begin{aligned} H_1' &= \mathbf{DES}_{\overline{H_0}}(\overline{M_1})\ \overline{M_1} \\ &= \overline{\mathbf{DES}_{H_0}(M_1)}\ \overline{M_1} \\ &= \mathbf{DES}_{H_0}(M_1)\ M_1 \\ &= H_1 \end{aligned}$$

可以看到当明文取反，H_0 取反后可以对消息进行修改但却保持散列值不变。

3. **散列函数的主要应用有哪些？**

（1）**保证数据的完整性**；例如，为保证数据的完整性，可以使用单向散列函数对数据生成并保存散列值，然后，每当使用数据时，用户将重新使用单向散列函数计算数据的散列值，并与保存的数值进行比较，如果相等，说明数据是完整的，没有经过改动；否则，数据已经被改动了。（2）**单向数据加密：如口令表加密**；应用于诸如口令表加密等场合，使用单向函数保存口令表可避免以明文的形式保存口令，至今仍然在许多系统中得到广泛的使用。（3）**数字签名**。

4. **对本章 SHA - 1 的示例，计算** $W_{16}\ W_{17}\ W_{18}\ W_{19}$。

$W_{16} = (W_0 \oplus W_2 \oplus W_8 \oplus W_{13}) <<< 1$
$= (61626380 \oplus 00000000 \oplus 00000000 \oplus 00000000) <<< 1 = C2C4C700$

$W_{17} = (W_1 \oplus W_3 \oplus W_9 \oplus W_{14}) <<< 1$
$= (00000000 \oplus 00000000 \oplus 00000000 \oplus 00000000) <<< 1 = 00000000$

$W_{18} = (W_2 \oplus W_4 \oplus W_{10} \oplus W_{15}) <<< 1$
$= (00000000 \oplus 00000000 \oplus 00000000 \oplus 00000018) <<< 1 = 00000030$

$W_{19} = (W_3 \oplus W_5 \oplus W_{11} \oplus W_{16}) <<< 1$
$= (00000000 \oplus 00000000 \oplus 00000000 \oplus C2C4C700) <<< 1 = 85898E01$

5. **什么是散列函数？对散列函数的基本要求和安全性要求分别是什么？**

散列函数 H(•) 是一种单向密码体制，它是一个从明文到密文的不可逆函数，也就是说，是无法解密的。

散列函数 H 必须具有如下性质：

(1) H 能用于任何大小的数据分组；
(2) H 产生定长输出；
(3) 对任何给定的 x，H(x) 要相对易于计算，使得硬件和软件实现成为实际可行。

(4) 对任何给定的码 m，寻找 x 使得 H(x) = m 在计算上是不可行的。(单向性质) 。散列函数具有碰撞不可避免性。但对散列函数的要求有：

(5) 对任何给定的分组 x，寻找不等于 x 的 y，使得 H(x)=H(y) 在计算上是不可行的。；

(6) 寻找任何 (x,y) 对 (x y)，使得 H(x) = H(y) 在计算上是不可行的。

6. **什么是消息鉴别？为什么要进行消息鉴别？消息鉴别的实现方法有哪些？**

消息鉴别是指在两个通信者之间建立通信联系后，每个通信者对收到信息进行验证，以保证所收到信息的真实性的过程。确保：（1）消息是由确认的发送方产生的；（2）消息内容没有被篡改过；（3）消息是按与发送时的相同顺序收到的。

实现方法：基于加密技术的消息鉴别、基于散列函数的消息鉴别。

6 数字签名

1. **数字签名的基本原理是什么？与传统签名相比有哪些优势及特点？**

   数字签名是使一个报文附加上一段起到签名作用的代码。这个代码可保证报文的来源和完整性。数字签名是使一个报文附加上一段起到签名作用的代码。这个代码可保证报文的来源和完整性。数字签名技术是将摘要信息用发送者的私钥加密，与原文一起传送给接收者。接收者只有用发送者的公钥才能解密被加密的摘要信息，然后用 HASH 函数对收到的原文产生一个摘要信息，与解密的摘要信息对比。如果相同，则说明收到的信息是完整的，在传输过程中没有被修改，否则说明信息被修改过。

   数字签名由两部分组成：带有陷门的数字签名算法和验证算法。

   (1) 手写签名: 签名认为是被签名消息的一部分。数字签名: 签名与消息是分开的，需要一种方法将签名与消息绑定在一起。
   (2) 手写签名: 验证是由经验丰富的消息接收者通过同以前的签名相比较而进行的。数字签名: 在签名验证的方法上，数字签名利用一种公开的方法对签名进行验证，任何人都可以对签名进行验证。
   (3) 手写签名: 签名的物理复制是无效的或容易识别的。数字签名: 签名代码物理上可被复制，故在数字签名方案的设计中要预防签名的再用。

2. **在 ElGamal 数字签名中，取素数 $\mathbf{p = 19}$ 与 Z19 上的一个本原元 $\mathbf{= 13}$，设用户 A 选定随机 $\mathbf{dA = 10}$ 作为自己的私钥。（1）求出用户 A 的公开验证密钥 $\mathbf{eA = ?}$（2）若用户 A 需要对报文 m 进行签名，设 $\mathbf{h(m) = 15}$，用户 A 产生的随机整数 $\mathbf{k = 11}$。计算消息 m 的数字签名，并对其进行验证。**

   （1）$e_A \equiv \alpha^{d_A} \equiv 6 \pmod{19}$

   （2）$r \equiv \alpha^k \equiv 2 \pmod{19}$，$s \equiv (h(m) - d_A \cdot r)k^{-1} \equiv 11 \pmod{18}$

   即消息签名为 $(2, 11)$，在验证时，$r^s \cdot e_A^r \equiv 2^{11} \cdot 6^2 \equiv 8 \equiv \alpha^{h(m)} \pmod{19}$，验证成功。

3. **设素数 $\mathbf{p = 11}$，$\mathbf{= 2}$ 是 Z11 上的本原元，用户 A 选择 $\mathbf{dA = 7}$ 作为其私钥，消息 m 的散列码 $\mathbf{H(m) = 10}$，系统选择随机数 $\mathbf{k = 7}$，试计算完成用户 A 采用 ElGamal 数字签名算法对消息 m 的签名过程以及用户 B 对签名的验证过程。**

   $e_A \equiv \alpha^{d_A} \equiv 7 \pmod{11}$

   $r \equiv \alpha^k \equiv 7 \pmod{11}$，$s \equiv (h(m) - d_A \cdot r)k^{-1} \equiv 3 \pmod{10}$

   即消息签名为 $(7, 3)$，在验证时，$r^s \cdot e_A^r \equiv 7^3 \cdot 7^7 \equiv 1 \equiv \alpha^{h(m)} \pmod{11}$，验证成功。

4. **在 ECDSA 数字签名算法中，设椭圆曲线为 $\mathrm{E}_{11}(1,6)$，选择 G = (2，7) 作为基点，G 的阶为 n = 13。假设用户 A 选择 dA = 5 作为自己的私钥，签名随机数 k = 2，消息 m 的散列码 H(m) = 11，试：（1）计算用户 A 的验证公钥 PA；（2）计算用户 A 对消息 m 的签名，并对其进行验证。**

（1）$P_A = d_A G = (3,6)$

（2）$kG = 2G = (5,2) \to r \equiv 5 \pmod{13}$，$s \equiv k^{-1}(h(m) + d_A \cdot r) \equiv 7 \cdot (11 + 5 \cdot 5) \equiv 5 \pmod{13}$

签名为 $(5,5)$，验证的时候，就是凑 $kG = s^{-1}(h(m) + d_A \cdot r) \cdot G$，则 $s^{-1}(h(m) + d_A \cdot r) \cdot G \equiv 8 \cdot 5 \cdot (5G) + 8 \cdot 11G \equiv 288G \equiv 5 \pmod{13}$，得到 $v = 5 = r$，验证完成

5. **（选做）假设 Alice 利用 ElGamal 签名方案对两个消息 m1 和 m2 进行签名，分别得到签名 $r\ s_1$ 和 $r\ s_2$，两个签名中的 r 是一样的，试：（1）假设 gcd(s1 - s2，p - 1) = 1，利用签名信息计算出签名随机数 k 和签名密钥 x。（2）若 gcd(s1 - s2，p - 1) 1，情况又将如何？**

（1）$r \equiv \alpha^k \pmod{p}$，

$$\begin{cases} s_1 \equiv k^{-1}(h(m_1) - x \cdot r) \pmod{p-1} \\ s_2 \equiv k^{-1}(h(m_2) - x \cdot r) \pmod{p-1} \end{cases}$$

$s_1 - s_2 \equiv k^{-1}(h(m_1) - h(m_2)) \pmod{p-1}$，由于 $p-1\ h(m_1)\ h(m_2)$ 已知，由于 $\mathrm{GCD}(s_1 - s_2\ p-1) = 1$, 所以 $\mathrm{GCD}(h(m_1) - h(m_2)\ p-1) = 1$，

由拓展欧几里得求解出 $h^{-1} = (h(m_1) - h(m_2))^{-1} \pmod{p-1}$，代入有，

$$h^{-1}(s_1 - s_2) \equiv k^{-1} \pmod{p-1}$$

，此时再由拓展欧几里得可得 $k \equiv ((h(m_1) - h(m_2))^{-1}(s_1 - s_2))^{-1} \equiv (h(m_1) - h(m_2)) \cdot (s_1 - s_2)^{-1} \pmod{p-1}$，

已知 k 后，假设 $\mathrm{GCD}(r\ p-1) = 1$，

$$k \cdot s_1 - h(m_1) \equiv -x \cdot r \pmod{p-1}$$

$$x \equiv -(k \cdot s_1 - h(m_1)) \cdot r^{-1} \pmod{p-1}$$

，即得到了

$$\begin{cases} k \equiv (h(m_1) - h(m_2)) \cdot (s_1 - s_2)^{-1} & \pmod{p-1} \\ x \equiv -(k \cdot s_1 - h(m_1)) \cdot r^{-1} & \pmod{p-1} \end{cases}$$

（2）若 GCD$(s_1 - s_2\ p - 1) > 1$，那么

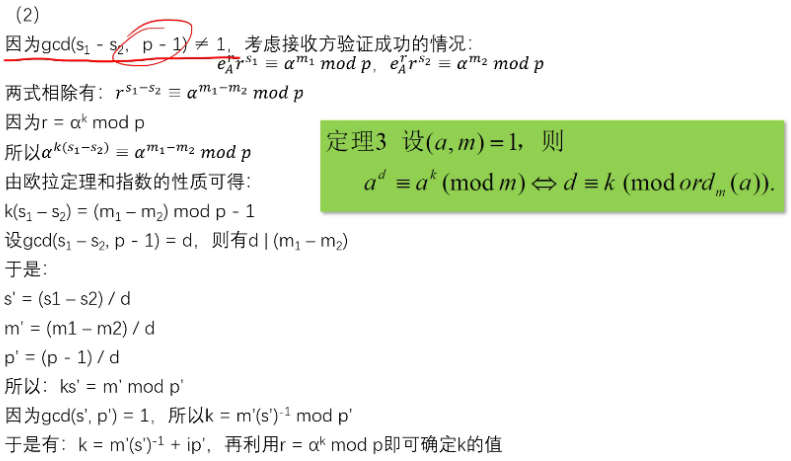

图 6.1: image-20200610141211534

7 密钥协商与管理

1. **为什么在密钥管理中要引入层次结构？**

   可减少受保护的密钥的数量，又可简化密钥的管理工作。一般将密钥划分为三级：主密钥，二级密钥，初级密钥。

   1) 安全性大大提高使得 *静止的密钥系统* → *动态的密钥系统*，下层的密钥被破译将不会影响到上层密钥的安全。在少量最初的处于最高层次的密钥注入系统之后，下面各层密钥的内容，可以按照某种协议不断地变化（例如可以通过使用安全算法以及高层密钥动态地产生低层密钥）。

   (2) 为密钥管理自动化带来了方便开放式的网络应用环境不可能再进行人工密钥分配。层次化密钥结构中，除了一级密钥需要由人工装入以外，其他各层的密钥均可以由密钥管理系统按照某种协议进行自动地分配、更换、销毁等。

2. **密钥分配的安全性应当注意什么问题？公开密码分发与秘密密钥分发的区别是什么？**

   注意

   被动威胁：窃听；主动威胁：篡改、重放、冒充；

   区别

   公开密钥分发：有两个密钥，在密钥分发时必须确保用于解密或数字签名的私钥的秘密性、真实性和完整性；秘密密钥分发：只有一个密钥，因此在密钥分发中必须同时确保密钥的秘密性、真实性和完整性。

3. **密钥分发与密钥协商过程的目标是什么？各又有何特点？**

   密钥分发与密钥协商过程都是用以在保密通信双方之间建立通信所使用的密钥的协议 (或机制)。

   分发：保密通信中的**一方**利用此机制生成并选择秘密密钥，然后将其安全传送给通信的另一方或通信的其他多方。

   协商：通信**双方（或更多）**可以在一个公开的信道上通过相互传送一些消息来共同建立一个安全的共享秘密密钥。在密钥协商中，双方共同建立的秘密密钥通常是双方输入消息的一个函数。

4. **简述 Diffie-Hellman 的密钥协商协议的原理和局限。**

   原理

Diffie-Hellman 安全性由求解有限域上离散对数的困难性保证。A 与 B 想要通讯，α 是 p 的本原元，

A 选择随机数 $r_A \in [2, p-2]$，发送 $S_A \equiv \alpha^{r_A} \pmod{p}$，

B 选择随机数 $r_B \in [2, p-2]$，发送 $S_B \equiv \alpha^{r_B} \pmod{p}$，

A、B 之后计算

$$S_K \equiv S_B^{r_A} \equiv S_A^{r_B} \pmod{p}$$

得到共享密钥 S_K。

局限

由于 Diffie-Hellman 没有提供身份鉴别，所以容易受到中间人攻击。

5. **设 Diffie-Hellman 密钥交换协议的公开参数 $\mathbf{p = 719}$ 和 $\mathbf{= 3}$，并且用户 A 和用户 B 想建立一个共享的秘密密钥 K_S。若用户 A 向用户 B 发送 191，用户 B 以 543 回答。设用户 A 的秘密随机数 $r_A = 16$，求建立的秘密密钥 K_S。**

$$K_S \equiv 543^{r_A} \equiv 543^{16} \equiv 40 \pmod{719}$$

6. **什么是公钥数字证书，它主要包括哪些基本内容？在应用时该如何确认证书以及证书持有人的有效性和真实性？**

答：

公钥证书是一种包含持证主体标识、持证主体公钥等信息，并由可信任的 CA 签署的信息集合。主要用于确保公钥及其与用户绑定关系的安全；

包含

用户的名称，用户的公钥，证书的有效日期以及 CA 的签名等；

验证

任何一个用户只要知道签证机构的公钥，就能检查对证书的签名的合法性；使用证书持有人的公钥加密信息来确认证书持有人的有效性和真实性。

三趟消息机制

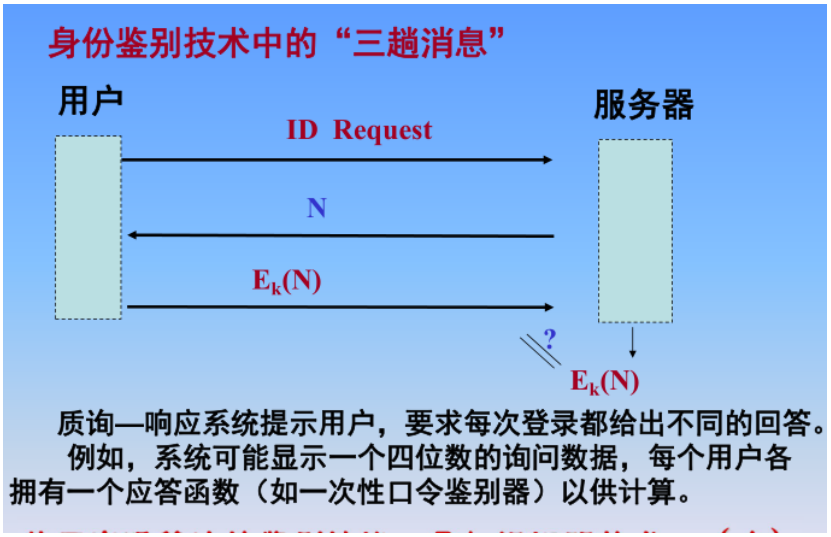

图 7.1: image-20200627002417479

8 密码技术与应用

1.【简答题】

简述链路加密与端到端加密的主要特点。

（5.0 分）

- 我的答案：

链路加密与端到端加密的层次不同，链路加密：加密过程对用户透明; 通常在物理层和数据链路层实现; 可确保整个通信链路上的传输机密性; 要求节点具有安全性加密设备、策略管理方面的开销较大。端到端加密：实现成本低；在跨越不同网络结构的节点处不会出现明文，可防止对网络节点的攻击; 可提供一定程度的认证; 加密可以由用户提供或选择，对用户来说采用这种加密方式比较灵活。密钥建立较复杂，易于受到通信流量分析的攻击；

2.【简答题】

请简述密钥环在报文的传输和接收的应用过程。

（5.0 分）

- 我的答案：

传输：PGP 利用用户 ID，从用户的私钥环中取出用户的被加密私钥。如果用户 ID 为缺省值，则从私钥环中取出第一个私钥。然后 PGP 提示用户输入通行短语（pass phase）用于恢复用户被加密的私钥。再利用解密得到的用户的私钥 SK 产生消息签名。接着 PGP 通过伪随机数发生器 PRNG 产生一个会话密钥 Ks，用 Ks 对消息与签名进行加密，接着使用对方公钥对 Ks 进行加密，加入到最后的报文。接收：PGP 从收到消息的会话密钥部分取出公钥 ID，并根据公钥 ID 从自己的私钥环中取出相应的被加密的私钥，然后提示用户输入通行短语（pass phase）以恢复私钥 SK ，再使用自己的私钥 SK 恢复出会话密钥 Ks，并进而解密消息与签名，然后验证签名。总的来说，通过用户的 pass phase 哈希后得到解密的 password，利用这个 password 对加密过的私钥进行解密。传输方要用私钥对消息进行签名，接收方要用私钥解密会话密钥。

3.【简答题】

SET 协议的双重数字签名中只有客户用自己的签名私钥完成了一次数字签名，为什么却称为双重数字签名？

（4.0 分）

- 我的答案：

因为签名的时候用了订购信息摘要、支付信息摘要的合集，然后对合集进行哈希后再用用户的私钥加密，这样完成的签名完成了对订购与支付的连接，一次性签了两个，别人不知道私钥所以无法完成这样的连接，所以称为双重数字签名。

4.【简答题】

什么是 PKI？简述 PKI 技术的主要意义。

（5.0 分）

- 我的答案：

PKI ，Public Key Infrastructure，公钥基础设施。一个基于公开密钥理论与技术实施和提供安全服务的具有普适性的安全基础设施的总称，它并不特指某一密码设备及其管理设备。PKI 至少包含证书机构、注册机构、证书库、证书撤消列表、密钥备份恢复、自动密钥更新、密钥归档、交叉认证等。提供鉴别、完整性、机密性服务。意义：PKI 提供的上述安全服务恰好能满足电子商务、电子政务、网上银行、网上证券等金融业交易的安全需求，是确保这些活动顺利进行必备的安全措施，没有这些安全服务，电子商务、电子政务、网上银行、网上证券等都无法正常运作。即 PKI 为日常生活办公等提供了必要的条件。

5.【简答题】

结合第 4 题，简述 PKI 在实际应用中存在的主要问题。

（6.0 分）

- 我的答案：

PKI 的重要性不言而喻，所以针对 PKI 进行攻击也是一种思路，这样的安全是集中管理的，而现在推崇去中心化，所以在这点上它可能存在问题。同时，如果同时有大量请求，容易造成其崩溃。在 PKI 建设中，针对技术问题和法律问题，在很多地方缺乏策略和指南或者存在错误的策略和指南; 实施费用高，特别是在实施一些非标准的接口时资金压力更大; 互操作问题依然突出，PKI 系统与其他系统的集成时面临已有系统的调整甚至替换的问题; 使用和管理 PKI 需要更多培训，PKI 的管理仍旧有严重障碍。

9 序列密码

1.【简答题】

什么是序列密码？比较序列密码与分组密码的不同。

（5.0 分）

我的答案：

序列密码，是对称密码的一种，由密钥或种子密钥通过密钥流发生器得到的密钥流为：K=k1k2 …kn，这样的密钥流就称为序列密码（流密码）。

不同：

- （1）序列密码以一个符号（如一个字符或一个比特）作为基本的处理单元，而分组密码以一定大小的分组作为基本的处理单元；
- （2）最大的不同：序列密码使用一个随时间变化的简单的加密变换，即不同时刻所用的密钥不同，而分组密码在不同时刻所用的密钥是相同的。

2.【简答题】

同步序列密码和自同步序列密码的特点分别是什么？

（5.0 分）

我的答案：

同步序列密码：（1）无错误传播。同步序列密码各符号之间是独立的。因此，一个传播错误只影响一个符号，不会影响到后继的符号。（2）有同步要求。在同步序列密码中，发送方和接收方必须保持精确的同步，用同样的密钥并作用在同样的位置上，接收方才能正确的解密。通信中丢失或增加了一个密文字符，则接收方将一直错误，直到重新同步为止，这是同步序列密码的一个主要缺点。但同时也能够容易检测插入、删除、重播等主动攻击。如 OFB。

自同步序列密码：（1）有限错误传播。设自同步序列密码的密钥序列产生器具有 n 位存储，则一个符号的传输错误将影响到后面 n 符号的解密，但不影响后面第 n+1 个及其以后符号的解密。（2）自同步。由同步序列密码的有限错误传播特性知，只要接收方连续收到 n 个正确的明文符号，通信双方的密钥序列产生器便会自动地恢复同步，因此被称为自同步序列密码。如 CFB。

3.【简答题】

为什么序列密码的密钥不能重复使用？

（3.0 分）

我的答案：

序列密码是一种伪随机数，想要模仿“一次一密”加密算法，其安全性特别依赖密钥流生成器生成的密钥流的周期、复杂度、伪随机特性等，密钥的重复使用将导致其安全性降低。

4.【简答题】

三级线性反馈移位寄存器在 c3 = 1 时可有几种线性反馈函数？设其初始状态为 (a1，a2，a3) = (1，0，1)，求各线性反馈函数对应的输出序列和周期。

（8.0 分）

我的答案：

四种

a3

101 110 011 （101…，周期 3

a2+a3

101 110 111 011 001 100 010 (101…，周期为 7

a1+a3

101 010 001 100 110 111 011 (101…周期为 7

a1+a2+a3

101 010 (101…，周期为 2

批语

输出序列和周期正确，但是线性反馈函数不对，参考第九章教程，PPT 上面的不对

更正

c3 对应的是 a1，所以，反馈函数是：

a1

a1+a2

a1+a3

a1+a2+a3

5.【简答题】

设四级移位寄存器的反馈函数为 f(a1，a2，a3，a4) = a1 a4 1 a2a3，初始状态为 (a1，a2，a3，a4) = (1，1，0，1)，求此非线性反馈移位寄存器的输出序列及周期。

（6.0 分）

1. 注意序列密码这里很坑的，输出就是 **a1** 的值！
2. 还有，注意 **a4~a1** 的排列顺序和题目给的反馈函数的顺序（相反的）
3. 并不是序列密码输出添加到状态中，而是反馈函数的输出添加到状态中
4. 添加移出，从第 **2** 点的排布来看，是淘汰低位，将反馈函数的输出补到高位

我的答案：

a4	a3	a2	a1	f(A)	输出
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	0	1	0
1	1	1	1	0	1
0	1	1	1	1	1
1	0	1	1	（开始重复）	

.....

图 9.1: image-20200627003250455

输出序列：11011

周期：5

10 附录

10.1 求 GCD 与逆元

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <conio.h>
using namespace std;

int x,y,q;
void extend_Eulid(int a,int b)
{
    if(b == 0){
        x = 1;y = 0;q = a;
    }else{
        printf("%d=(%d)*%d+%d\n",a,a/b,b,a%b);
        extend_Eulid(b,a%b);
        int temp = x;
        x = y;
        y = temp - a/b*y;
    }
}
int main()
{
    char c='1';
    if (c=='1'){
        l1:int a,b;
        cin>>a>>b;
        if (a<b) swap(a,b);
        extend_Eulid(a,b);
        printf("\n%d=(%d)*%d+(%d)*%d\n",q,x,a,y,b);
        while(x<0) x+=b;
        while(y<0) y+=a;
        printf("%d=(%d)*%d+(%d)*%d\n",q,x,a,y,b);
        goto l1;
    }
    else
    {
        while(1){
            int a,b;
            cin>>a>>b;
            printf("%d=%d*(%d)+(%d)",a,b,a/b,a%b);
        }
    }

    return 0;
}
```


10.2 辅助做题的代码

10.2.1 week04_T2

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

List = list(range(0,26))
prime =[]
mutiPrime = set()

for i in range(0,26):
    if maxx(i,26)==1:
        prime.append(i)
'''
for i in prime:
    for j in prime:
        mutiPrime.add((i*j)%26)
'''
for i in prime:
    for j in range(0,26):
        mutiPrime.add((i*j)%26)

print(mutiPrime)
```

10.2.2 week04_T7

```
while(1):
    s = input("> ").split(' ')
    for i in s:
        print(i[3]+i[2]+i[4]+i[0]+i[1],end=' ')
```

10.2.3 week04_T9_ 由字母生成数字或者反着来

```
import bidict
dic = {'a':0,'b':1,'c':2,'d':3,'e':4,'f':5,'g':6,'h':7,'i':8,'j':9,'k':10,'l':11,'m':12,'n
↪  ':13,
'o':14,'p':15,'q':16,'r':17,'s':18,'t':19,'u':20,'v':21,'w':22,'x':23,'y':24,'z':25}

```

```
s = input("> ").lower()

for i in range(len(s)):
    print(dic[s[i]],end=',')

print()
'''
bid = bidict.bidict(dic)
s = input("> ").split(' ')
for i in s:
    print(bid.inverse[int(i)],end=',')
'''
'''
25 12 11 25 10 6 4 0 19 22 22 17 25 6 19 20 6 21
'''
```

10.2.4 week04_ 生成字母对照数字 0 开始表

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
int i=0;
    char c='a';
int main()
{

    for(c=97;c<123;c++)
    {
        printf("%c:%d, ",c,(int)c-(int)'a');
    }
    system("pause");
    return 0;
}
```

10.2.5 week05_T5_ 输出 P 盒

```
s1 = input("> ")
s = str("9")
s = s + s1
print(s[16], s[7], s[20], s[21], s[29], s[12], s[28], s[17], s[1], s[15], s[23], s[23],
↪  s[5], s[18], s[31], s[10], s[2], s[8], s[24], s[14], s[32], s[27], s[3], s[9], s[19],
↪  s[13], s[30], s[6], s[22], s[11], s[4], s[25])
#1100000101010010111110101010110
```

10.2.6 week05_T7_ 计算字节乘法

```
key = "00011011"
while(1):
    s = input("> ")
    if s[0]=="0":
        print(s[1:]+"0")
    else:
        l = list()
        m = list()
        l.append("0")
        for i in range(len(s)-1):
            l.append(s[len(s)-1-i])

        for i in range(len(s)):
            if l[len(s)-1-i]!=key[i]:
                m.append("1")
            else:
                m.append("0")
        for i in m:
            print(i,end="")
        print("")
```

10.2.7 week09_T4_ECC 点乘

```
"""
    考虑K=kG ，其中K、G为椭圆曲线Ep(a,b)上的点，n为G的阶（nG=O∞ ），k为小于n的整数。
"""

def get_inverse(mu, p):
    """
    获取y的负元
    """
    for i in range(1, p):
        if (i*mu)%p == 1:
            return i
    return -1


def get_gcd(zi, mu):
    """
    获取最大公约数
    """
    if mu:
        return get_gcd(mu, zi%mu)
    else:
        return zi
```

```
23
24  def quicktimes(k,x,y,a,p):
25      if k==1:
26          return x,y
27      if k==2:
28          return addPoint(x, y, x, y, a, p)
29      else:
30          x1,y1 = quicktimes(k//2,x,y,a,p)
31          x2,y2 = addPoint(x1, y1, x1, y1, a, p)
32          if k%2==0:
33              return x2,y2
34          else:
35              return addPoint(x2,y2,x,y,a,p)
36
37  def addPoint(x1, y1, x2, y2, a, p):
38      x3 = x1
39      y3 = y1
40      lamb = 0
41      if x1==x2 and y1==y2:
42          inverse_2 = get_inverse(2,p)
43          inverse_y1 = get_inverse(y1,p)
44          print("inverse2:",inverse_2, " inv_y1:",inverse_y1)
45
46          lamb = (((3*x1*x1+a)*inverse_2*inverse_y1)%p+p)%p
47      else:
48          inverse_ = get_inverse(((x2-x1)%p+p)%p,p)
49          lamb = (((y2-y1)*inverse_)%p+p)%p
50
51      x3 = ((lamb**2-x1-x2)%p+p)%p
52      y3 = ((lamb*(x1-x3)-y1)%p+p)%p
53      print("lambda:",lamb," x3:",x3," y3:",y3)
54      return x3,y3
55
56  if __name__ == "__main__":
57      print("*************ECC椭圆曲线加密*************")
58      #ecc_main()
59      addPoint(10,2,2,7, 1, 11)
60      #print(quicktimes(288,2,7,1,11))
61      #addPoint(3,5,5,-2, 1, 11)
62      #print(get_inverse(17, 31))
```

10.2.8 week09_T5_ 快速模幂乘

```
1  p = 19
2
3  def quick1(a,b):
4      if b == 0:
```

```
        return 1
    elif b==1:
        return a
    else:
        t = quick(a,b//2)
        print(a,"^",b//2,"=",t)
        if b%2 == 1:
            return ((((t*t)%p)*a)%p+p)%p
        else:
            return ((t*t)%p+p)%p
def quick(a,b):
    res = 1
    cnt=1
    while (b):
        if (b&1):
            res=(a*res)%p
        a = (a*a)%p
        b >>= 1
        print(cnt,": a= ",a,"b= ",b,"res=", res,sep=" ",end="\n");
        cnt+=1
    return res

print(quick1(13,15))
```

10.2.9 week09_T6_RSA 加密数字串

```
p = 2537
def quick(a,b):
    if b == 0:
        return 1
    elif b==1:
        return a
    else:
        t = quick(a,b//2)
        if b%2 == 1:
            return ((((t*t)%p)*a)%p+p)%p
        else:
            return ((t*t)%p+p)%p

s = input("> ").split(",")
for i in s:
    print(quick(int(i),13),end=",")
print("\n")
```