**题型包括单选题，简答题，计算题，证明题**

单选题 20 10

简答题 20 4

计算题 40 3(1+1+2) 10行左右

证明题 20 2

# 数学基础

**考试相关推测：**

一些基础数学知识盲猜会出现在单选题中。

我认为主要是要知道一些符号的含义，具体定理的话应该是要结合算法放到证明题里。这部分哪怕没有单独考，复习一下也不错。

1. **整除**

a整除b，记作a|b，注意a是除数，b是被除数，反过来叫b能被a整除。

1. **最大公约数** gcd表示最大公约数。

设a、b为整数，且a、b中至少有一个不等于0，令d=gcd(a,b)，则一定存在整数x、y使得下式成立: a\*x + b\*y = d。

特别地，当a、b互素时，则一定存在整数x、y使得a\*x+b\*y=1成立。

1. **互素**

对于整数a、b，若gcd(a,b)=1，则称a、b互素。

1. **模运算和同余**

a，b对于模n同余，记作: a ≡ b (mod n)

1. **加法模逆元**

若a+b≡0 (mod n), 则称a是b的加法模n逆元，b是a的加法模n逆元。某种意义a和b互为相反数。

重要应用在于加上一个数等于减去这个数的加法逆元。

1. **乘法模逆元**

若a\*b≡1 (mod n), 则称a是b的乘法模n逆元，b是a的乘法模n逆元。a的乘法逆元记作a-1。

利用扩展欧几里德法可求乘法逆元。

例如：求13模35的乘法逆元

35/13=2……9 得到9

13/9=1……4 得到4

9/4=2……1 得到1

得到余数1后无法继续分解，把1用35和13来代替。

1 = 9-2\*4 = 35-2\*13 – (13-(35-2\*13))\*2

然后13前的系数就是乘法逆元。

# Enigma

**考试相关推测：**

必考一道计算题。

大概类型就是给你RingSetting——三个齿轮内部字母，然后三个齿轮是5个中的哪三个，最后给你plugboard，然后让你连续加密俩字母，最后输出的结果写出来，过程要写。我几乎肯定小白会出double stepping。

# MD5和SHA-1

**考试相关推测：**

可能出在简答题，或者前面单选题。

简答题会考md5计算过程中的分块及填充的过程。

单选题会考一些基础概念。

例如：MD5是有损压缩，无限的明文对应有限的128位十六进制数

SHA-1对应的长度是160位十六进制数，sha-1也是分块计算，每块也是64字节，当最后一块不足64字节也按照md5的方式进行填充。数据块的最后一定要补上表示报文总共位数的8个字节。

# ECB/CBC/CFB和RC4

**考试相关推测：**

我觉得必出一道简答题。

简答题内容就是程序填空，让你完成一种分组密码工作模式，所以复习的时候要看下咱之前写的des加密作业，或者看下源码，把整个过程理顺。

单选题也可能出，但是如果简答题能完成，我觉得不担心单选题。

比如：以下不是CBC模式特点的是：A当前块与前一块密文相关，B，加密只能串行C，解密可以并行D.可以从密文传输错误中恢复。

至于RC4了解下整体原理，打乱sbox，然后明文与sbox异或。

RC4在prepare\_key时只交换，不赋值。

# DES

**考试相关推测：**

可能考简答题，描述DES算法整体流程，但是大循环不会考？

剩下会考单选题：例如：给定一个sbox，输入一个数字，问你输出是多少？

注意查表是输入6位，输出4位。

按照简答题取准备，单选题肯定没问题。

重点了解sbox，进6出4（8组6位，共48位，出来后32位），过程，考试给表，能算；掌握把表打乱的过程，把输入信息打乱。

# AES

**考试相关推测：**

肯定是单选题重度内容。

重点是mixcolumn过程，要记住矩阵乘法。

还有不同密钥长度，上课只讲了一种，不同密钥长度如何切换有啥区别要了解一下。

AES mixcolumn环节，多项式乘法运算，必须掌握加密与解密时，矩阵不一样，3112,BD9E

能写农夫算法，会手工算（加法、减法：异或；出来mod）

# RSA

**考试相关推测：**

会考一道证明题，要知道是如何得出加密和解密公式的。

其他的可能会有些简单的计算放在单选题啥的。

知道公钥和私钥关系：d\*e=1 mod (p-1)\*(q-1)

# ECC

**考试相关推测：**

这一块绝对是考试重点内容。

具体来说，会考n道计算题？，所有解都会列在一张表里，加法乘法都可以直接查，但是要会算。

除了正常加密解密，两种签名手段ecnr和ecdsa也要会算，不过这两种签名公式会给不用记。

ECC了解两个点和点加法的集合意义，知道怎样加的，知道乘n怎样乘的，计算公式不用记

ECC的6要素，能够运用ECC加解密算法（公式）推到结果的正确性，能够推导签名的正确性

ECC源代码关于提取坐标的函数需要了解。

# 证明题

考试相关推测：

前面已经讲了会考一道RSA，还剩下的就是

1. 证明gcd(n,u)=an+bu (p.93)

2. 证明Euler准则(p.98)

3. 证明中国剩余定理(p.83)

其中欧拉准则我也不知道会不会考，老师上课没讲。

具体我还在问以前上过的同学。

# 复习资料

**第二章**

Enigma看老师笔记就可以，比较全的。

**（注意反射板不需要加减Δ，直接查即可）**

**第三章**

1. Md5算法的输出固定为128位，即16字节。

Md5分块计算，每块大小为64字节。若文件大小为n字节，则最后那一块的长度为n%64，其范围为【0:63】，即若最后一块恰好为64字节，则后面还有一个0字节的块，需要被填充为64字节或128字节（两个块）。数据末最后一字节必须填充为0x80，然后继续填0x00，直到前56个字节都满。最后八个字节为64位整数，其值等于message总共的位数，小端规则（不含填充内容）

1. SHA算法的输出为160位，即20字节，比md5多了32位（4字节）

SHA算法的填充方式与md5一样

**第四章**

**ECB模式：电子密码簿**

明文的每个块（8个字节）单独根据密钥和加密算法进行加密，互相之间没有联系。

加密和解密都可以并行，但对于相同内容的明文段，加密后得到的密文块是相同的。

**CBC模式：密文块链接模式**

明文的前一个块（8个字节）首先跟种子数异或，然后根据密钥和加密算法进行加密，其他块类似，但是后一个块的种子数是前一个块加密后得到的密文（8个字节）。

加密只能串行，但是解密可以并行（已知所有块的密文）。

**CFB模式：密文反馈模式**

首先将初始种子数（8个字节大小）根据密钥和加密算法进行加密，得到的结果的首字节与明文的首字节（1个字节大小）（即明文以8位为一个单位）异或，得到密文，后一个字节的加密过程中，需要将初始种子数左移8位，右边补上8个0，之前得到的密文去填补这8个0的空挡，这样的到新的种子数，继续加密，得到的结果的首字节与明文的第二个字节异或，得到密文，以此类推。

这个过程所产生的前一个密文并不是直接和后一个密文建立联系的，还跟种子数有关系，每一个明文块（8个字节）加密需要调用8次。

**密文反馈模式的优点是可以从密文传输的错误中恢复。**

因为种子数每次左移8位，如果只有其中一个字节传输、解密发生错误，只会影响接下来的8次解密，共9次解密，但之后就不会发生错误了。

**流密码算法RC4：**

加密和解密算法相同。

rc4(buf, n, &k); buf是明文块，n是明文长度，k是密钥。

目标密钥的生成需要种子密钥，种子密钥长度，由prepare\_key生成。

首先初始化目标密钥state，256个元素，一开始每个元素的值等于自己的下标，然后一一配对进行打乱，例如s[0]和s[22]配对，那么他们的值交换，最终得到密文表。（只做交换，不做赋值）

打乱的过程是设置两个下标变量index1和index2,初始都是0。

设置256次循环，counter计数，令index2等于(种子密钥查表key\_data\_ptr[index1]+state[counter]+index2)%256，然后将state[counter]与state[index2]的值交换，index1 = (index1+1)% key\_data\_len。

然后根据得到的密文表去加密得到目标密钥。

加密的过程是进行buffer\_len次循环，counter用于循环计数，将当前元素buffer\_ptr[counter] ^= state[xorIndex]; （加密/解密前，需要交换一对state）

而xorIndex = (state[x]+state[y])%256;

其中x和y初始为0，每次循环中x= (x+1)%256, y=(state[x]+y)%256,将state[x]与state[y]交换，可见每次循环还改变了密文表。

**第五章**

**DES算法：Data Encryption Standard**

它属于块加密算法，明文、密钥和密文都是8个字节，加密和解密的密钥相同。

因此破解DES就是要在已知部分明文和密文对的情况下，求解密钥。

有效的攻击方法是差分分析。

1. **sbox**

sbox[8][64]分成s1到s8，每一个有64个元素，每16个元素为一行，这一行中按特定的给定顺序把0至15共16个元素分配，所以sbox中每个元素都是小于等于15的。

查表是输入6位，输出4位。

例如输入101101，那么最高位和最低位组合在一起变成11，这是第几行。中间4位代表第几列。得到的结果转换成二进制4位输出。

至于查的是s1至s8中哪一个表，是循环查表的，一共要查8组，6\*8=48位输入，4\*8=32位输出。

1. **明文**

明文64位是按照大端规则编号的。（密钥和密文应该也都是大端的）

高位置位于低地址。

1. **核心环节**

明文64位分成L32位和R32位，L32加密8次，R32加密8次，一共循环加密16次，先L32加密1次然后R32加密1次，以此类推。

在加密L32的过程中R32是有参与的，在加密R32的过程中L32也会参与，他们参与的方法都是成为密钥的一部分。

K64是64位密钥。

**首先加密L32。**

将R32展开成48位（对某些位重复），同时将K64缩减成56位（把每个字节的最右边一位砍掉，也就是第7位），然后将K56分成KL28和KR28。

将KL28和KR28都循环左移1次或2次（KL28和KR28的次数总是相同的），这里1或2次并不是说次数是随机的，而是在16轮加密循环中，通过key\_rol\_steps[16]查表得到的值确定的，总共循环左移的次数累加和为28。

static char key\_rol\_steps[16]={1,2,4,6,8,10,12,14,15,17,19,21,23,25,27,28};

后一个元素与前一个元素之差为当前轮要循环左移的次数。

循环左移都是累加的，即后一轮加密的循环左移是在前一轮循环左移的基础上再进行的。

完成循环左移后，将KL28和KR28合并并缩减成48位。

将R48和K48进行异或。

异或后的结果当作sbox的输入，输出32位S32。

将S32与L32异或得到L32’。

**接下来加密R32。**

加密流程是与之前一致的，将R32和L32的角色替换一下即可。

但要注意的是，原先的L32已经被L32’替换了，因此在加密R32的过程中，采用的都是L32’（当作原先的L32）。

将L32’展开成48位，将KL28和KR28继续循环左移1或2次，然后合并并缩减成48位，然后将L48’和K48进行异或，结果当作sbox的输入，输出32位S32，将S32与R32异或得到R32’。

1. **打乱**

**明文打乱：**

明文64位在分成L32和R32之前，会打乱，例如第0位挪到第56位。

这个挪动的位置是查表得到的。

**密文打乱：**

明文经过16轮加密出来64位密文的时候，会反向打乱，例如第56位挪回到第0位。

**密钥打乱：**

把KL28和KR28合并并缩减成48位的过程中，密钥也会打乱。

**sbox打乱：**

sbox输出的32位也要打乱。

源代码打乱都是查表法，老师讲解循环法。

打乱表ip[64]中的元素的下标表示目标位号，值表示源位号。

例如ip[0]=58，那么表示目标第0位=源第57（58-1）位。

1. **老师演示——采用循环实现打乱 按位循环查表**

明文打乱的表和密文打乱的表是互逆的。所有打乱的方法都是一致的。

假设现在进行明文打乱：

char s[8],t[8],ip[64]={....};

int i,j;

memset(t,0,sizeof(t)); //源是0的位就不复制到t了，提高效率。

for(i=0;i<64;i++)

{

n=ip[i]-1;

byte = n/8;

bit = n%8;

if(s[byte] & 1<<(7-bit)) *//s[byte] & mask[bit] , mask={0x80,0x40....8,4,2,1}*

t[i/8] |= 1<<(7- i%8); *//t[i/8] |= mask[i%8];*

}

在源代码中为了提高效率用n>>3代替n/8，然后用n&7代替n%8。

1. **des算法——采用查表法实现打乱 按组循环查表**

源64位 假设为1011 0110 1001 1111 ...

表为char t[16][16][8]

第一维的16表示16组，即将源分成16组，每组4位。

第三维的8表示8个字节，因为源一共有8个字节，每一组存放4位，但由于4位可能打乱后分配在这8个字节的任意位置，所以要用8个字节的数组来存放。将16组的8个存放打乱位置的数组进行或运算就得到打乱后的结果了。

第二维的16表示源4位的16种变化，从0000到1111。

**具体查询方法：**

例如查询例子的1011，那么1011属于第0组，第11种变化

char \*p = t[0][0x0B];

p指向8个字节中的首地址

然后查询例子的0110，那么0110属于第1组，第6种变化

char \*q = t[1][0x06];

然后做8次循环将p和q数组里的8个字节进行或运算。

以此类推合并即可。

**演示代码：**

unsigned char table[16][16][8]={...};

char s[8],t[8],B4,\*p;

int i,j,k;

memset(t,0,sizeof(t));

for(i=0;i<16;i++) 进行16组查询，将s[8]分成[16][4]

{

B4 = (s[i/2] >> (i%2==0?4:0)) & 0x0F ; 判断奇偶决定是该字节的高4位还 是低4位

p = table[i][B4];

for(k=0;k<8;k++)

t[k] |= p[k];

}

在正式源代码中对应的是permute函数，源是inblock输出是outblock

permute中一次循环将每个字节的高4位和低4位同时取出，一次取2组，然后合并。

1. **根据打乱表生成查询表table[16][16][8]**

ip[64]是initial permutation，初始打乱表，用来打乱明文

fp[64]是final permutation，终端打乱表，用来打乱密文

memset(table,0,sizeof(table));

for(i=0;i<16;i++)

{

for(B4=0;B4<16;B4++)

{

for(k=0;k<4;k++)

{

if(B4 & (1<<(3-k)) == 0) 等于0就没必要保存了，初始化就是0，也可以用mask的方法查表

continue;

n = fp[i\*4+k]-1; 现在得从源位找到目标位，而不是目标位找到源位，因此采用的打乱表是fp，fp是ip的逆表。如果不用逆表就得在ip中用循环根据值来找到下标，即目标位。

table[i][B4][n/8] |= 1<<(7-n%8);

}

}

}

在正式源代码中对应的是perm\_init函数。

1. **核心函数f**

假设明文左边32位为L0，明文右边32位为R0，E(R0)表示拓展成48位的R0，56位密钥为K0,对K0左侧28位和右侧28位均循环左移1位后合并并缩减成48位得到K1。

要对L0进行加密生成L1：

f()函数中首先将R0展开成48位E(R0)，然后将E(R0) ^ K1，最后将结果进入sbox得到32位。

因此第0轮过后，L1 = L0 ^ f(E(R0),K1),R1=R0

第1轮后，L2=L1,R2=R1^f(E(L1)^K2)

**源代码讲解：**

static long32 f(ulong32 r, unsigned char subkey[8])

subkey是48位密钥K，其中每个元素的高2位恒为0。

r是L32或者R32，因此f中还要展开成48位。

展开的过程是这样的，源代码利用移位的方法分别提取6位，提取8次。

1 2 3 4 5 6 7 8 9 A B C D E F..... 1E 1F 20 先循环右移，最高位跑到左侧(大端)

20 1 2 3 4 5 6 7 8 9 A B C D E F ... 1E 1F

然后依次提取20 1 2 3 4 5，4 5 6 7 8 9,8 9 A B C D....，完成展开。

提取中每6位直接去和subkey对应的组去异或。

1. **加密函数round和解密函数reverse\_round**

加密的时候是从0到15轮，解密的时候是从15到0轮。

在加密完成16轮后会进行一次左右32位交换。

因此解密的时候还是从左边开始解密（实际上对应的就是原先的右边）

1. **大端处理**

在明文打乱的过程中是不能进行大小端转换的，因为是按照位或者字节打乱的。

打乱后要对明文块进行颠倒，转换成小端。

因为后续马上明文块就要进入f()函数，扩展成48位，而扩展过程中是多个字节采用移位的方法，大小端的结果是不同的。

**什么时候大小端会产生不一样的结果呢？**

对n个字节逐字节做处理时，大小端没有区别。因为大小端就是不同字节存放地址高低相反，对字节内部没有影响。

但如果把多个字节看成整体进行处理时，例如long32，加减乘除移位等运算都会产生不同的结果。

1. **密钥生成**

des\_set\_key函数将56位密钥，分成56个字节保存。

**第六章**

**AES算法的加密过程：**

**概述：**

首先AddRoundKey(p, k); 实际上就是异或，把明文的16个字节和密钥的16个字节（长度有16,24,32字节三种）逐字节异或，密文长度恒为16字节。

密钥为16字节时之后key\_round=10，24字节时，12轮；32字节时，14轮。

每个循环里会经历ByteSub-ShiftRow-MixColumn三个步骤

**ByteSub：**

是把明文每一个字节当作sbox的下标，找到的值替换明文，是用来对付差分分析的。sbox定义是unsigned char sbox[256];且sbox的元素值不重复。

**ShiftRow：**

是把明文p中的16个字节构成4\*4的矩阵，其中第0行不变，第1行循环左移8位，第2行循环左移16位次，第3行循环左移24位次。矩阵是这样排布的，unsigned char p[16]：

0 4 8 C

1 5 9 D

2 6 A E

3 7 B F

这个矩阵将p数组按顺序纵向排列的，然后循环左移完得到

1. 4 B C

5 9 D 1

A E 2 6

F 3 7 B

**MixColumn：**

最为复杂，是核心步骤

要用到的有限域是GF(2^8)，就是8位二进制的加法和乘法的域。

所谓加法运算实际上是异或，没有进位的。

所谓乘法运算实际上是列竖式(展为二进制列竖式)的方法然后每一个乘，最后用加法运算相加（异或）。

矩阵相乘时，先采用矩阵的乘法(选取相乘的行和列的时候)，然后多项式各项采用的乘法采用以上的乘法运算，各项加法的时候也采用以上的运算。

**8位数乘法实质上是多项式乘法mod (x8+x4+x3+x+1)：**

这个mod的多项式是固定的，可以保证任何数都有乘法逆元。

这里的值同时还是不可约多项式，确保可进行逆运算还原。

a \* a-1 = 1 mod n

a有乘法逆元的条件是gcd(a,n) = 1

当n为素数且a<n时，gcd(a,n) = 1

例如: 求1000 1000 \* 0000 0101 mod 0x11B

可以把上述两数乘法转化成两个多项式相乘:

(x7 + x3) \* (x2 + 1) = x9 + x7 + x5 + x3

再用手工除法求模：

mod (x8 + x4 + x3 + x + 1)

x

x8+x4+x3+x+1 x9 + x7 + x5 + x3

x9 + x5 + x4 + x2 + x

x7 + x4 + x3 + x2 + x

0-x4=x4 mod 2 加法逆元

把余式转化成二进制就是:

1001 1110

结论:

0x88 \* 0x05 = 0x9E mod 0x11B

**MixColumn中的3次多项式乘法运算（4位数乘法）：**

明文 char plain[16] = {4,3,2,1,11,22,33,44...};

密文 char cipher[16] = {14,5,0,15,...};

(3x3 + x2 + x + 2) \* (x3 + 2x2 + 3x + 4) mod (x4 + 1)

= ↓这里是低次系数

2 3 1 1 4 14

1 2 3 1 \* 3 = 5

1 1 2 3 2 0

3 1 1 2 1 15

↑这里是高次系数

x3(2\*1 + 1\*2 + 1\*3 + 3\*4)

x2(2\*2 + 1\*3 + 1\*4 + 3\*1)

x1(2\*3 + 1\*4 + 1\*1 + 3\*2)

x0(2\*4 + 1\*1 + 1\*2 + 3\*3)

以x0­系数为例:

2\*4 + 1\*1 + 1\*2 + 3\*3 = 0x08 ^ 0x01 ^ 0x02 ^ 0x05

= 0x09 ^ 0x02 ^ 0x05 = 0x0B ^ 0x05 = 0x0E

这里的前矩阵是AES算法固定的，后矩阵是从明文中取出一列。

这里的x4 + 1保证不出现3次以上的项，但这个多项式却是可约多项式。为了解决这个问题，3x3 + x2 + x + 2是特定的不可约多项式，它与0xBD9E互为乘法逆元，可以保证可进行逆运算还原，即例子中的密文矩阵乘前矩阵的乘法逆元可以得到后矩阵，即明文矩阵。

E B D 9 14 4

9 E B D \* 5 = 3

D 9 E B 0 2

B D 9 E 15 1

**用农夫算法分步计算0x05 \* 0x43 mod 0x11B:**

a = 0000 0101

b = 0100 0011

p = 0

a / 2 \* b \* 2 = a \* b

b \* a = b \* (0000 0100 + 1)

为了防止a的个位在除2的过程中丢失，修改成

b \* 1 + b \* 0000 0100

将b的值放入p，p = 0100 0011

然后将a>>1 = 0000 0010，b<<1 = 1000 0110

然后由于a的个位是0，不担心个位丢失

直接将a>>1 = 0000 0001，b<<1 = 1 0000 1100，b超过8位必须mod 0x11B，得到b = 0001 0111

接着a的个位又是1，于是修改式子，并将b的值与p相加(异或)，p = 0100 0011 ^ 0001 0111 = 0101 0100

接着将a>>1 = 0000 0000，b<<1 = 0010 1110

由于其中一个值为0，结束运算，p为结果。

**AES算法密钥的生成过程：**

以16个字节的密钥为例：

把16个字节的Key分成4个long类型的初始key，假设为k[0]\k[1]\k[2]\k[3]

对16个字节的加密来说，key\_round = 10，因此后面还要生成40个long，从k[4]到k[43]，其中k[i]与k[i-4]是有关的。

生成步骤如下：

k[4]=k[3]，然后把k[4]循环左移8位ROL(k[4],8)，再把k[4]中的每个字节代入sbox替换ByteSub(&k[4],4)，然后要把k[4]的首字节异或轮常数，再把k[4]与k[0]异或，k[5] = k[4]，k[5]与k[1]异或，k[6] = k[5]，k[6]与k[2]异或，以此类推，得到40个long。

**第七章**

**RSA算法:**

首先，选取两个大素数：p和q，计算乘积：n=p\*q

其中n公开，p、q保密。

然后随机选取加密私钥d，使d和(p-1)\*(q-1)互素。

接着要找出公钥e，使得：

e\*d = 1 mod ((p-1)\*(q-1))

按下面的公式进行加密明文m：

c=me(mod n)

按下面的公式进行解密密文c：

m=cd(mod n)

**RSA算法证明：**

由于密文c的加密c=me mod n，解密的时候m=cd mod n

所以明文m不能大于n。

我们可以在明文m划块的时候，例如n是128位的FE....，明文块只填充127位，那么一定能保证明文m不大于n。

我们还可以明文块最高位留出比如7个位，随机填充数值，这里填充的数值保证不大于n即可，那么可以在保证m不大于n的基础上，使得相同的明文产生不同的密文。

**方法1:**

设m是明文, c是密文, c=me mod n。现证明m=cd mod n。

因为φ(n)= φ(p\*q) = φ(p)\*φ(q)= (p-1)(q-1)，

又因为ed = 1 mod (p-1)(q-1)，

所以一定可以找到一个k使得ed = 1 + k(p-1)(q-1)成立，

于是cd = med = m1 + k(p-1)(q-1) = m \* m k(p-1)(q-1)

= m \* (mφ(n))k = m \* (1)k = m mod n

**为什么(mφ(n))k= (1)k mod n ? Euler定理**

a = a' mod n

b = b' mod n

则一定有a\*b = a'\* b' mod n, 这是因为:

a = kn+a'

b = jn+b'

a\*b=(kn+a')(jn+b') = kjnn + a'jn + b'kn + a'b'

**上述证明的前提是gcd(m,n)=1。**

**那如果m和n不互素呢？**

当gcd(m,n) ≠ 1时, 则一定有gcd(m,n)=p或gcd(m,n)=q。现假设gcd(m,n)=p，即m是p的倍数，则m与q一定互素。于是有:

mφ(q) = 1 mod q 🡺 m(q-1) = 1 mod q 🡺

m(q-1)\*k(p-1) = 1 mod q 🡺 mφ(n)\*k = 1 mod q 🡺

mφ(n)\*k = q\*s + 1 🡺 m \* mφ(n)\*k = m\*q\*s + m 🡺

mφ(n)\*k+1 = cp \* q \* s + m 🡺 mφ(n)\*k+1 = m mod n

🡺 med = m mod n

**第八章**

**一个椭圆曲线由六个要素构成，该曲线上的所有整数点构成一个群**

a/b/p/基点G/G的阶/余因子

y2 = x3 + ax + b (mod p)

基点G是满足该曲线方程的某个点(x,y)

G的阶n代表n\*G=0，即恰好完成一个循环的次数

余因子是满足这个曲线方程的点的个数

余因子=曲线的阶即曲线上点的个数/G的阶,此值通常=1

真正使用时的余因子一定等于1，非1的余因子是不安全的。

**椭圆曲线在素域上的运算规则：**

(1) P+O=O+P=P

(2) 如果P=(x1,y1), Q=(x2, y2), 且有x1=x2及y1=y2=0,或有x1=x2及y1=-y2≠0, 则P+Q=O;

(3) 如果P=(x1,y1), Q=(x2, y2), 且排除(1)(2), 则

P+Q=(x3,y3)由下列规则决定:

x3 = λ2 – x1 – x2

y3 = λ(x1-x3) - y1

当P≠Q时, λ=(y2-y1)/(x2-x1);

当P=Q时, λ=(3x12+a)/(2y1);

**2. Euler准则(p.98)**

y2 = x mod p

设p>2是一个素数, x是一个整数, gcd(x,p)=1, 则

(1) x是模p的平方剩余当且仅当

x(p-1)/2 ≡ 1 (mod p)

(2) x是模p的平方非剩余当且仅当

x(p-1)/2 ≡ -1 (mod p)

所谓平方剩余代表对不同的y，和固定的p，可能出现的x就是模p的平方剩余。非平方剩余则

**3. 点加运算举例(p.127)**

Q = k\*P 其中k<n

已知P及Q的情况下，求k很困难。

设α=(2,7), 计算2α=α+α:

λ=(3x12+a)/(2y1)=(3\*22+1)/(2\*7)=

13/14=13\*14-1=2\*3-1=2\*4=8 mod 11

x3=λ2 – x1 – x2=82-2-2=60=5 mod 11

y3 = λ(x1-x3) - y1=8\*(2-5)-7=

8\*8-7=64+4=2 mod 11

因此2α=(5,2)

**7.2 用ECC算法加密解密(p.129)**

Menezes-Vanstone公钥密码体制 p.129

**1. 公钥及私钥**

公钥点R=d\*G

私钥d是一个随机数, 且d<n, 其中n是G的阶

**已知基点G和公钥R，在短时间内解不出私钥d**

这个问题称为ECDLP问题

**2. 加密**

r = (k\*G).x ; 其中k是一个随机数且k<n (.x表示取该点的x坐标)

; r不可以mod n

s = m\*(k\*R).x mod n; 其中m是明文，也要<n

密文包括两部分:r,s

**3. 解密: 红色的r是一个点，有x和y坐标，r=k\*G**

m = s/(d**r**).x = m\*(k\*R).x/(d\*(k\*G)).x

= m\*(k\*d\*G).x/(k\*d\*G).x

**4. 举例**

设曲线方程是y2=x3+x+6 (mod 11)

基点G=(2,7), G的阶n=13

设私钥d=7, 则公钥R=dG=7\*(2,7)=(7,2)

设随机数k=6, 明文m=9, 则

密文第1部分**r**=kG=6\*(2,7)=(7,9)

密文第2部分s=m\*(k\*R).x=9\*(6\*(7,2)).x=9\*(8,3).x

=9\*8 mod 13 = 7

m=s/(d**r**).x= 7/(7\*(7,9)).x =

=7/(8,3).x = 7/8 = 7\*8-1 mod 13 = 7\*5 mod 13

=9

**7.3 用ECC算法签名验证**

**1. ecdsa(elliptic curve digital signature algorithm)**

**(1)签名**

**r = k\*G ; k是随机数**

**s = (m+r\*d)/k ; m是明文或hash, d是私钥**

**(2)验证**

**(m/s)\*G+(r/s)\*R == r**

**(m/s)\*G+(r/s)\*R = mG/s + rR/s =**

**(mG+rdG)/s = (m+rd)G/((m+rd)/k) = kG**

**如果伪造m或d，都无法通过验证。**

**2. ecnr**

**(1)签名**

**r = k\*G+m**

**s = k-r\*d**

**证明题**

看老师的课件即可。