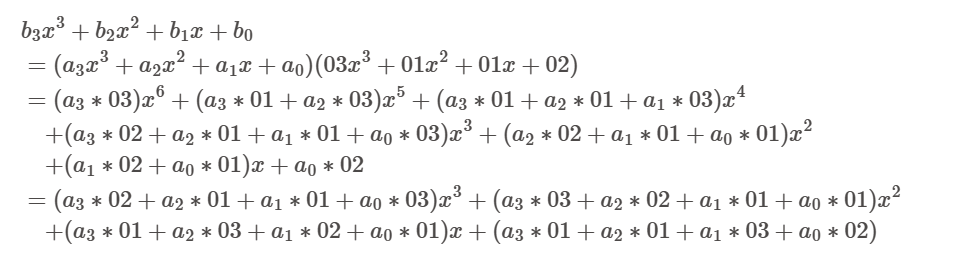
现代密码学作业——第四讲

第三节

1、

上述公式表达的即为题中所示的矩阵乘法。

2、0x87=1000 0111即a(x)=x7+x2+x+1

0x03=0000 0011

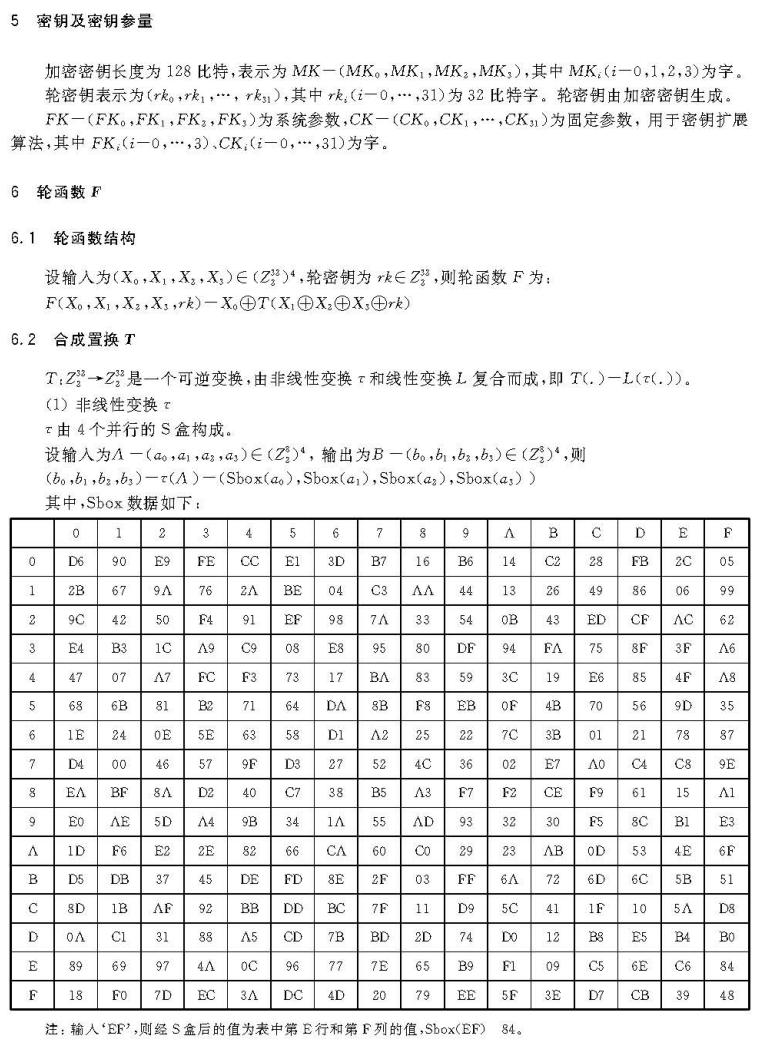
A(0)=a(x) mod m(x)= x7+x2+x+1

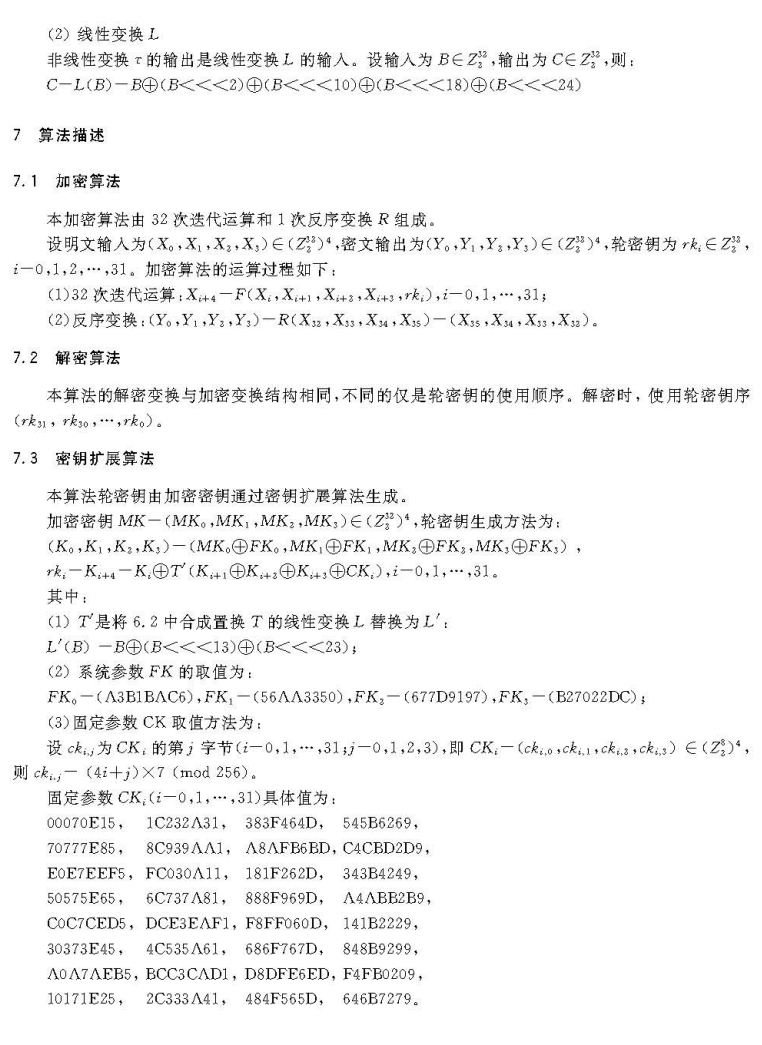
A(1)=x\*A(0) mod m(x)=x4+x2+1

a(x)\*b(x)=A(0)+A (1)=x7+x4+x

那么，0x87\*0x03=1001 0010=0x92

3、





4、

1. # S盒乘法逆元及仿射实现
2. # 求最高幂次数
3. **def** Nonzero\_MSB(value):
4. v2str = '{:09b}'.format(value)
5. **for** i **in** range(9):
6. **if** int(v2str[i]):
7. **return** 9 - i
9. # 模2除法：m为被除数。b为除数，q为商，r为余数
10. **def** Mode2\_div(fx, gx):
11. n = Nonzero\_MSB(fx)
12. m = Nonzero\_MSB(gx)
13. **if** n < m:
14. **return** [0, fx]
15. deg = n - m
16. fx = fx ^ (gx << deg)
17. [q, r] = Mode2\_div(fx, gx)
18. **return** [(1 << deg) | q, r]
20. # 多项式乘法
21. # v3 = v1 - q3 \* v2
22. **def** Calculate(v1, q3, v2):
23. value = 0
24. **for** i **in** range(32):
25. **if** (q3 & (1 << i)):
26. value = value ^ (v2 << i)
27. **return** v1 ^ value
29. # 欧几里得算法
30. **def** poly\_gcd(r1, r2, v1=1, v2=0, w1=0, w2=1):
31. **if** r2 == 0 **or** r2 == 1:
32. **return** w2
33. q3, r3 = Mode2\_div(r1, r2)  # q3(x)=r1(x)|r2(x), r2(x)=r1(x) mod r2(x)
34. v3 = Calculate(v1, q3, v2)  # v3 = v1 - q3 \* v2
35. w3 = Calculate(w1, q3, w2)  # w3 = w1 - q3 \* w2
36. **return** poly\_gcd(r2, r3, v2, v3, w2, w3)
38. **def** sym2int(sym):
39. power = [sym[i + 2] **for** i **in** range(len(sym)) **if** sym[i] == 'x']
40. **if** '+1' **in** sym: power.append('0')
41. data = 0
42. **for** p **in** power:
43. data = data | (1 << int(p))
44. **return** data
46. **def** int2sym(data):
47. int2str = '{:09b}'.format(data)
48. sym = ''
49. **for** i **in** range(9):
50. **if** int(int2str[i]) == 1:
51. **if** 8 - i:
52. sym += '+x^%d' % (8 - i)
53. **else**:
54. sym += '+1'
55. **return** sym[1:]
57. **def** xor(a, b):
58. **if** a == b:
59. **return** '0'
60. **else**:
61. **return** '1'
63. **def** fangshe(a):
64. a = a[::-1]
65. c = '11000110'
66. b = ''
67. **for** i **in** range(7, -1, -1):
68. b += xor(xor(xor(xor(xor(a[i], a[(i + 4) % 8]), a[(i + 5) % 8]), a[(i + 6) % 8]), a[(i + 7) % 8]), c[i])
69. **return** b
71. **if** \_\_name\_\_ == '\_\_main\_\_':
72. bi = bin(int(input('请输入两位16进制数：'), 16))[2:]
73. **print**("16进制转为为2进制为：{}".format(bi))
74. xstr = ''
75. **for** i **in** range(len(bi)):
76. **if** bi[i] == '1':
77. **if** i < len(bi) - 1:
78. xstr += "+x^{}".format(len(bi) - i - 1)
79. **elif** i == str(len(bi)):
80. xstr += '+1'
81. xstr = xstr[1:]
82. inverse = int2sym(poly\_gcd(283, sym2int(xstr)))
83. data = hex(sym2int(inverse))
84. **print**('你输入的多项式 :', xstr)
85. **print**('默认既约多项式 :', int2sym(283))
86. **print**('乘法逆元为 :', inverse)
87. **print**('乘法逆元的16进制：', data)
88. **print**('---- 多项式乘法逆元求解完成 ----')
89. data = bin(int(data, 16))[2:].zfill(8)
90. **print**('最终结果为：', fangshe(data))
91. **print**('16进制表示为', hex(int(fangshe(data), 2)))

第四节

1、OFB模式（输出反馈模式）：密码算法的输出会反馈到密码算法的输入当中。其是将“明文分组”和“密码算法的输出”进行XOR来产生“密文分组”。优点：传输过程中密文在某位上发生的错误不会影响解密后明文其他位。缺点：失去同步，如果加密端和解密端移位寄存器不同步，那么恢复的明文将是一些无用的杂乱数据，任何使用OFB的系统必须有检测失步的机制，并用新的初始向量IV填充双方移位寄存器重新获得同步；抗消息流篡改攻击能力不如CFB。

2、CCM模式（认证保密模式）：CCM是CTR加密模式和CMAC认证算法的混合使用，常用在需要同时加密和认证的领域，比如WiFi安全中的WPE协议，它就使用了AES-CCM模式。其首先使用CBC-MAC模式来认证传输帧，然后使用CTR模式来加密帧。