DES

发表于 2021-01-04 分类于 Challenge ， 2019 ， 湖湘杯 ， Crypto
Challenge | 2019 | 湖湘杯 | Crypto | DES

点击此处获得更好的阅读体验

题目考点

- *DES加密*

解题过程

题目未知key,采用DES加密，已知Kn，也就是子密钥，已知mes加密的结果，要求求出mes以及key。 主要思路是这样，第一步通过DES子密钥求出密钥key，之后通过key直接进行解密求出mes,当然也可以直接用子密钥求解出mes。 通过子密钥求解key可以参考一道有关密钥编排的DES题目(飘零师傅NB,一搜又是你)，一步一步求解，但是密钥生成子密钥的过程中有8位是没有用的，所以逆向推导子密钥会有8位是不确定的，遍历出来会有256个密钥，这256个密钥都可以用来加密解密，而且结果都是正确的。具体参考过程如下：

```
key1 = [1,0,1,0,0,0,0,0,1,0,0,1,0,1,1,0,0,1,0,0,0,1,1,0,0,1,1,1,0,1,1,1,0,0,0,0,0,1,1,1,1,1,0,0,1,1,0,0,0]
__pc2 = [
    13,16,10,23,0,4,
    2,27,14,5,20,9,
    22,18,11,3,25,7,
    15,6,26,19,12,1,
    40,51,30,36,46,54,
    29,39,50,44,32,47,
    43,48,38,55,33,52,
    45,41,49,35,28,31
]
C1D1 = ['*']*56
for i in range(0,len(key1)):
    C1D1[__pc2[i]] = key1[i]
print C1D1
#[00000001*11111100*110*00*000
# 011001*01*1101*0001011000*01]
# you1
C0 = '0000000001*11111100*110*00*00'
D0 = '1011001*01*1101*0001011000*01'
__pc1 = [56,48,40,32,24,16,8,
    0,57,49,41,33,25,17,
    9,1,58,50,42,34,26,
    18,10,2,59,51,43,35,
    62,54,46,38,30,22,14,
    6,61,53,45,37,29,21,
    13,5,60,52,44,36,28,
    20,12,4,27,19,11,3
]
C0D0 = C0+D0
res = ['*']*64
deskey = ""
for i in range(0,len(__pc1)):
    res[__pc1[i]] = C0D0[i]
for i in res:
    deskey += i
print deskey
def zuoyiwei(str,num):
    my = str[num:len(str)]
    my = my+str[0:num]
    return my
def key_change_1(str):
    key1_list = [57,49,41,33,25,17,9,1,58,50,42,34,26,18,10,2,59,51,43,35,27,19,11,3,60,52,44,36,63,55,47,39,31,23,15,7,62,54,46,38,30,22,14,6,61,53,45,37,29,21,13,5,28,2
    res = ""
    for i in key1_list:
        res+=str[i-1]
    return res
def key_change_2(str):
    key2_list = [14,17,11,24,1,5,3,28,15,6,21,10,23,19,12,4,26,8,16,7,27,20,13,2,41,52,31,37,47,55,30,40,51,45,33,48,44,49,39,56,34,53,46,42,50,36,29,32]
    res = ""
    for i in key2_list:
        res+=str[i-1]
    return res
def key_gen(str):
    key_list = []
    key_change_res = key_change_1(str)
    key_c = key_change_res[0:28]
    key_d = key_change_res[28:]
    for i in range(1,17):
        if (i==1) or (i==2) or (i==9) or (i==16):
            key_c = zuoyiwei(key_c,1)
            key_d = zuoyiwei(key_d,1)
        else:
            key_c = zuoyiwei(key_c,2)
            key_d = zuoyiwei(key_d,2)
        key_yiwei = key_c+key_d
        key_res = key_change_2(key_yiwei)
        key_list.append(key_res)
    return key_list
deskey = "01000abc01de111f0100100h0110010i0110111j01k00L1m0n0o010p0100001q"
print key_gen(deskey)
deskey = "0100000c0110111f0100100h0110010i0110111j0110011m0100010p0100001q"
# ['10100000100101100100011000111011000001111001l000', '1k1o000000d101100n0100101001011000110110be1a0110', '01100100kn010d1001110000000111l0ae000101111110010b', '1d000110111010
# [101000001001011001000110001110111000001111001l000], [111000000011011001010010100101100011011000100110], [011001001101011001110000001111000000101111100100], [110001101101010
print key_gen(deskey)
# deskey = "0100000"+c+"0110111"+f+"0100100"+h+"0110010"+i+"0110111"+j+"0110011"+m+"0100010"+p+"0100001"+q
def bintostr(str):
    res = ""
    for i in range(0, len(str), 8):
        res += chr(int(str[i:i+8],2))
    return res
for c in "01":
    for f in "01":
        for h in "01":
            for i in "01":
                for j in "01":
                    for m in "01":
                        for p in "01":
                            for q in "01":
                                str = "0100000" + c + "0110111" + f + "0100100" + h + "0110010" + i + "0110111" + j + "0110011" + m + "0100010" + p + "0100001" + q
                                str = bintostr(str)
                                print str
```

求出256个密钥后用任一解密可以得到mes。然后就是让人秃头的地方，讲道理这里一般都有个啥暗示可以筛选或者看出来一个明显特定的密钥，开始我盲猜要是可见字符，然后全部都是，然后又根据其它flag盲猜是uuid,然后全部都不是。 然后比赛快结束的时候，木的办法的我只好遍历了256个flag，开始一个一个试，在经历了卡顿掉线重新登陆提交的20分钟后我试出来了，具体结果就不说了，这个key确实是一个特定的词，但是混在256个里面很难看出来。密码表如下

```
@nHdnfDB, @nHdnfDC, @nHdnfEB, @nHdnfEC, @nHdngDB, @nHdngDC, @nHdngEB, @nHdngEC,
@nHdofDB, @nHdofDC, @nHdofEB, @nHdofEC, @nHdogDB, @nHdogDC, @nHdogEB, @nHdogEC,
@nHenfDB, @nHenfDC, @nHenfEB, @nHenfEC, @nHengDB, @nHengDC, @nHengEB, @nHengEC,
@nHeofDB, @nHeofDC, @nHeofEB, @nHeofEC, @nHeogDB, @nHeogDC, @nHeogEB, @nHeogEC,
@nIdnfDB, @nIdnfDC, @nIdnfEB, @nIdnfEC, @nIdngDB, @nIdngDC, @nIdngEB, @nIdngEC,
@nIdofDB, @nIdofDC, @nIdofEB, @nIdofEC, @nIdogDB, @nIdogDC, @nIdogEB, @nIdogEC,
@nIenfDB, @nIenfDC, @nIenfEB, @nIenfEC, @nIengDB, @nIengDC, @nIengEB, @nIengEC,
@nIeofDB, @nIeofDC, @nIeofEB, @nIeofEC, @nIeogDB, @nIeogDC, @nIeogEB, @nIeogEC,
@oHdnfDB, @oHdnfDC, @oHdnfEB, @oHdnfEC, @oHdngDB, @oHdngDC, @oHdngEB, @oHdngEC,
@oHdofDB, @oHdofDC, @oHdofEB, @oHdofEC, @oHdogDB, @oHdogDC, @oHdogEB, @oHdogEC,
@oHenfDB, @oHenfDC, @oHenfEB, @oHenfEC, @oHengDB, @oHengDC, @oHengEB, @oHengEC,
@oHeofDB, @oHeofDC, @oHeofEB, @oHeofEC, @oHeogDB, @oHeogDC, @oHeogEB, @oHeogEC,
@oIdnfDB, @oIdnfDC, @oIdnfEB, @oIdnfEC, @oIdngDB, @oIdngDC, @oIdngEB, @oIdngEC,
@oIdofDB, @oIdofDC, @oIdofEB, @oIdofEC, @oIdogDB, @oIdogDC, @oIdogEB, @oIdogEC,
@oIenfDB, @oIenfDC, @oIenfEB, @oIenfEC, @oIengDB, @oIengDC, @oIengEB, @oIengEC,
@oIeofDB, @oIeofDC, @oIeofEB, @oIeofEC, @oIeogDB, @oIeogDC, @oIeogEB, @oIeogEC,
AnHdnfDB, AnHdnfDC, AnHdnfEB, AnHdnfEC, AnHdngDB, AnHdngDC, AnHdngEB, AnHdngEC,
AnHdofDB, AnHdofDC, AnHdofEB, AnHdofEC, AnHdogDB, AnHdogDC, AnHdogEB, AnHdogEC,
AnHenfDB, AnHenfDC, AnHenfEB, AnHenfEC, AnHengDB, AnHengDC, AnHengEB, AnHengEC,
AnHeofDB, AnHeofDC, AnHeofEB, AnHeofEC, AnHeogDB, AnHeogDC, AnHeogEB, AnHeogEC,
AnIdnfDB, AnIdnfDC, AnIdnfEB, AnIdnfEC, AnIdngDB, AnIdngDC, AnIdngEB, AnIdngEC,
AnIdofDB, AnIdofDC, AnIdofEB, AnIdofEC, AnIdogDB, AnIdogDC, AnIdogEB, AnIdogEC,
AnIenfDB, AnIenfDC, AnIenfEB, AnIenfEC, AnIengDB, AnIengDC, AnIengEB, AnIengEC,
AnIeofDB, AnIeofDC, AnIeofEB, AnIeofEC, AnIeogDB, AnIeogDC, AnIeogEB, AnIeogEC,
AoHdnfDB, AoHdnfDC, AoHdnfEB, AoHdnfEC, AoHdngDB, AoHdngDC, AoHdngEB, AoHdngEC,
AoHdofDB, AoHdofDC, AoHdofEB, AoHdofEC, AoHdogDB, AoHdogDC, AoHdogEB, AoHdogEC,
AoHenfDB, AoHenfDC, AoHenfEB, AoHenfEC, AoHengDB, AoHengDC, AoHengEB, AoHengEC,
AoHeofDB, AoHeofDC, AoHeofEB, AoHeofEC, AoHeogDB, AoHeogDC, AoHeogEB, AoHeogEC,
AoIdnfDB, AoIdnfDC, AoIdnfEB, AoIdnfEC, AoIdngDB, AoIdngDC, AoIdngEB, AoIdngEC,
AoIdofDB, AoIdofDC, AoIdofEB, AoIdofEC, AoIdogDB, AoIdogDC, AoIdogEB, AoIdogEC,
AoIenfDB, AoIenfDC, AoIenfEB, AoIenfEC, AoIengDB, AoIengDC, AoIengEB, AoIengEC,
AoIeofDB, AoIeofDC, AoIeofEB, AoIeofEC, AoIeogDB, AoIeogDC, AoIeogEB, AoIeogEC,
```

FLAG

```
flag{AnHengDB}
```


 # Challenge # 2019 # Crypto # 湖湘杯

Pwn1

give me your passport