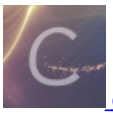
crypto 曼彻斯特编码

原创

舞动的獾 于 2019-11-08 17:40:42 发布 1842 收藏 8
分类专栏： crypto 文章标签： crypto 曼彻斯特编码



曼彻斯特编码：曼彻斯特编码将时钟和数据包含在数据流中，在传输代码信息的同时，也将时钟同步信号一起传输到对方，每位编码中有一跳变，不存在直流分量，因此具有自同步能力和良好的抗干扰性能。但每一个码元都被调成两个电平，所以数据传输速率只有调制速率的1/2。
曼彻斯特编码表示0或1有两种不同的方法：
第一种G. E. Thomas, Andrew S. Tanenbaum1949年提出的，它规定0是由低-高的电平跳变表示，1是高-低的电平跳变。
按此规则有：

- 编码0101（即0x5），表示原数据为00；
- 编码1001（0x9）表示10；
- 编码0110（0x6）表示01；
- 编码1010（0xA）表示11。
  第二种IEEE 802.4（令牌总线）和低速版的IEEE 802.3（以太网）中规定, 按照这样的说法, 低-高电平跳变表示1, 高-低的电平跳变表示0。
- 编码0101（0x5）表示11；
- 编码1001（0x9）表示01；
- 编码0110（0x6）表示10；
- 编码1010（0xA）表示00；

如何做题？
这里的话就拿i春秋上的题目开刀了
题目在i春秋上还有，以下题面均摘自i春秋

```
5555555595555A65556AA696AA6666666955 这是某压力传感器无线数据包解调后但未解码的报文(hex) 已知其ID为0xFED31F，请继续将>报文完整解码，提交hex。 tips：flag是flag{破译出的明文} 提示1：曼联
```

曼联的全称是曼彻斯特联足球俱乐部，这个提示很谐。这串二进制字符串中只包含四种字符：５６９Ａ，实际上就是0b0101 0b0110 0b1001 0b1010 ，就是曼彻斯特编码中，0和1的两种位的翻转.

解密脚本：

```
def conv(s):
    return hex(int(s, 2))[2:]

def IEEE802(bs):
    pass
    dict = {
        "0101": "11",
        "1001": "01",
        "0110": "10",
        "1010": "00"
    }
    bs=str(bs)
    print bs
    r = ""
    for j in range(0, len(bs), 4):
        i=bs[j:j+4]
        if i in dict.keys():
            r += dict[i]
    return r

n = 0x5555555595555A65556AA696AA6666666955
flag = ''
bs = '0' + bin(n)[2:]
r = ''
print bs
r = IEEE802(bs)

for i in range(0, len(r), 8):
    tmp = r[i:i + 8][::-1]
    flag += conv(tmp[:4])
    flag += conv(tmp[4:])
print flag.upper()
```

不反转的情况下是 ffff7fcbf8260aaa9f，是不存在题目所述的ID 0xFED31F，实际上按照IEEE关于令牌环或者以太网的规定，网络字节序是大端序，所以应该反转后解码，反转后脚本解出来是 fffffed31f645055f9，其中确实包含了 fed31f，所以这就是我们想要的答案。

CISCN 2016 传感器2

现有某ID为0xFED31F的压力传感器，已知测得压力为45psi时的未解码报文为：5555555595555A65556A5A96AA666666A955 压力为30psi时的未解码报文为： 5555555595555A65556A9AA6AA6666665665 请给出ID为0xFEB757的传感器在压力为25psi时的解码后报文，提交hex。 注：其他测量读数与上一个传感器一致。 tips：flag是flag{破译出的明文}

这道题的话 用上面的代码解码，分别解出来是 `fffffed31f635055f8` `fffffed31f425055d7`（都需要反转的，别忘了。）观察发现有两处不同，而且都相差33，这就很有趣，对观察出校验算法也是有帮助的，另外由于25和30相差5，30和45相差15，所以猜测答案和30的应该相差11因此最后解出来的答案就是 `FFFFFEB757375055CC`

CISCN 2017 传感器1

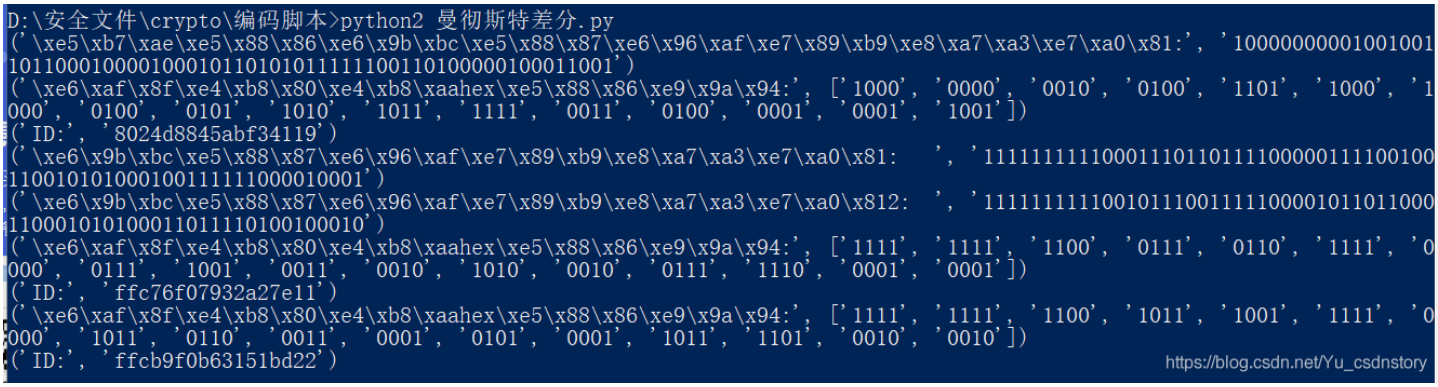
已知ID为0x8893CA58的温度传感器的未解码报文为：3EAAAAA56A69AA55A95995A569AA95565556

此时有另一个相同型号的传感器，其未解码报文为：3EAAAAA56A69AA556A965A5999596AA95656

请解出其ID，提交flag{hex（不含0x）}。

一开始觉得是曼彻斯特解码，结果发现这道题是差分曼彻斯特

```
#!/usr/bin/env python
#coding:utf-8

import re

#hex1 = 'AAAAA56A69AA55A95995A569AA95565556' # #  0x8893CA58
hex1 = 'AAAAA56A69AA556A965A5999596AA95656'

def bintohex(s1):
    s2 = ''
    s1 = re.findall('.{4}',s1)
    print ('每一个hex分隔:',s1)
    for i in s1:
        s2 += str(hex(int(i,2))).replace('0x','')

    print ('ID:',s2)


def diffmqst(s):
    s1 = ''
    s = re.findall('.{2}',s)
    cc = '01'
    for i in s:
        if i == cc:
            s1 += '0'
        else:
            s1 += '1'
        cc = i  # 差分加上cc = i

    print ('差分曼切斯特解码:',s1)
    bintohex(s1)




def mqst(s):  #只能算曼切斯特编码,无法算差分
    mdict = {'5': '00', '6': '01', '9': '10', 'A': '11'}
    a1 = ''.join(mdict[i] for i in s)
    a2 = ''.join(mdict[i][::-1] for i in s)
    print ('曼切斯特解码:   ',a1 )
    print ('曼切斯特解码2:  ',a2)
    bintohex(a1)
    bintohex(a2)


if __name__ == '__main__':
    bin1 = bin(int(hex1,16))[2:]
    diffmqst(bin1)

    mqst(hex1)
```



可以看到差分曼切斯特编码输出的是第一个ID，然后是两个曼切斯特编码的ID，都是16进制，但是好像并没有题上的0x8893CA58，仔细看第一个ID发现0x8893CA58是第一个ID的一部分，左边去掉5个字符，右边去掉4个字符。

拿AAAAA56A69AA556A965A5999596AA95656去跑脚本，得到差分曼切斯特编码为8024d8845abf34119，左边去掉5个字符，右边去掉4个字符，换成大写就是flag。