二维码离线开门生成规则

开门二维码由七部分组成，前四部分是明码，后三个部分组成的字符串通过XXTEA加密方法，进行加密处理

1. 二维码用户数据，n位

用户自定义数据，数据中不能含有符号“|”,因为这个符号是用户数据与其它数据分隔的标识符号

注：用户数据是指用户添加的另外的数据，可以自行定义 如姓名、职位等信息

1. 二维码关联卡号（8位）

二维码在开门后，会生成一条记录，记录的标识就是关联的卡号，这个卡号，一般用人员二维码在数据表中的ID号，记录上传到数据库后，可以通过这个卡号来查询到是谁开了门

注：卡号是软件人员自己定义的，一般用生成二维码的表的ID作为此二维码的卡号，历史记录中的卡号解析出来后，通过查询二维码生成表，就能知道是哪个二维码刷过的卡。此卡号和后面的二维码限制次是有关联的，如果启用限次功能，则生成一个二维码，则此卡号要递增1，保证卡号递增不重复。

1. 加密长度(2位)以及加密长度余数（2位）

注：1）加密长度，是指加密字符串每8位，则加密长度为1（也就是1个8位无符号整形），

注1：加密长度余数是加密字符串最后不够8位，剩余多少位的数值的一半。

比如 剩下2位则加密长度余数为01，剩下4位值则加密长度余数为02，剩下6位则加密长度余数为03，没有剩余则加密长度余数为00）

1. 二维码加密用户秘钥（UserKey2），4个字节

注1：XXTEA加密方法说明：

从第5-7点的形成的字符串使用XXTEA加密方法来进行加密，XXTEA加密密钥长度是64位，8个字节

加密秘钥由设备秘钥（UserKey1）和用户秘钥（UserKey2）组成，设备秘密（UserKey1）默认为0x00543210, 用户秘钥（UserKey2）自己定义,（也就是随意的1个无符号整形值，最后转成对应的字符串）

64位的加密秘钥，由4个整形值组成，变量定义：unsigned int key[4] = { UserKey2, UserKey1, UserKey2, UserKey1};

1. 管理字节 16位

开门时限 2位

00 表示没有时间限制

01 表示有开始时间和结束时间限制

开门次数 2位 取值范围00-FF

00 没有次数限制

01-FF ( 1-255次)

开门方式 2位

00 直接开门

01 以项目编号+楼栋编号参数匹配来开门

02 以设备MAC地址后六位匹配来开门

03以设备MAC地址后四位匹配来开门

开门时长 4位 （二个字节，最大表示65535秒）

0000 默认时长 FFFFFF 不开锁1

0012 18秒

开门时长，是指开门时，继电器控制门锁断电或上电的时间

扩展第二路输出 1位

0不扩展，不启用第二路输出，兼容以前格式

1扩展第二路输出（刷二维码时，第二路输出也会与第一路门锁同时动作）

扩展韦根输出 1位

0不扩展，不输出韦根卡号，兼容以前格式

1 扩展韦根输出

梯控楼层控制 1位

0不扩展，无梯控权限

1 用户人员，有梯控权限

2 物业人员，有梯控权限

3 管理人员，有梯控权限

保留3位 用作扩展用

1. 如启用第二路输出，第二路继电器输出延时时间 8位 （扩展第二路输出为0，则此项不存在），

第二路继电器延时时长，是指第二路输出继电器闭合延时时间，继电器控制外设的时长

0000 默认时长，

003C 60秒

保留4位，用作扩展

1. 如启用扩展韦根输出，输出韦根卡号 8位

输出韦根卡号 8位 此处韦根卡号，是指输出给梯控等外设的传统韦根26/34的卡号，如输出韦根卡号为韦根26，则卡号高二位会被忽略（为任意数或默写为00）

1. 开门的有效日期：开始日期和结束日期 24位，（没有启用时限，随意填24位即可）

12位的开始日期和12位的终止日期组合在一起

如：160227101256160228101256

表示此二维码的起始时间为2016年02月27日10时12分56秒

终止时间为2016年02月28日10时12分56秒

8．可开房间编号或可开MAC地址后6位或4位的组合，或梯控的项目编号+梯控楼栋编号

开门匹配认证有二种方式，一是MAC地址认证匹配方式，二是楼栋编号认证匹配方式，根据现场情况二者选一

如果选择楼栋认证匹配方式，则设备的楼栋编号需要预先设置到设备里面去，详情查看执行《二维码参数设置》文档和对应软件;

如果选MAC地址认证方式，需要提前将设备的MAC记录下来，以便将MAC地址设置到开门二维码中去。

1)楼栋或MAC地址数量： 2位

00 所有楼栋或设备都可以开

02 可开楼栋或房间为2个，有2个楼栋或房间的编号

2)项目编号: 8位 （以MAC地址匹配认证，则此字段不需要）

00000000 不受项目编号限制（这个功能目前废止，项目编号必须匹配上）

41590501 项目编号编制规则前6位，可参考邮政编码，规则后2位自定义

注：以楼栋编号认证，则项目编号匹配得上，才有效，项目编号不匹配，则直接返回错误。

3)楼栋或房间编号组合 8位\*N （以MAC地址匹配认证，则此字段不需要）

楼栋房间编号:03020101 8位 3栋1单元 1层01房间

0302 几栋几单元

0101 几层几房间

00000000 所有楼栋，所有层数，所有房间都有权限 00代表所有

如果是梯控含义为：

楼栋编号:03010100 3栋1单元1号电梯

楼栋编号:03010200 3栋1单元2号电梯

楼栋编号:03020200 3栋2单元2号电梯

楼栋编号:01010100 1栋1单元1号电梯

楼栋编号:00000000 所有楼栋的电梯都有权限

楼栋编号:01000000 楼栋1所有单元的电梯都有权限

楼栋编号:00010000 所有楼栋的1单元的电梯都有权限

楼栋编号:00000100 所有楼栋的所有单元的1号电梯都有权限

如果是楼层按键控制器，则只需要匹配前4位即可，如二维码楼栋编号03010100，只需要匹配前四位0301，只要是0301栋的二维码编号即可打开楼层按键

4)MAC地址低6位组合 6位\*N （以楼栋编号认证，则此字段不需要）

如果选择项目编号开门，则以上第4）点，MAC地址不要用

如果选择MAC地址开门，则以上第2），3）项目编号和楼栋房间编号不要用

加密字符：是指从以上第5（含第5点）开始，到后面的所有字符都进行XXTEA加密，以便二维码数据具有高保密性。

1. 如启用楼层控制，

楼层一共64层，16位，1位控制一层, “1”表示控制该楼层，“0”表示不控制该层。

字节顺序和楼层关系说明：8个字节，第一个字节代表1-8层，第二个字节代表9-16层，第三个字节代表17-24层。。。

一个字节由8位的“0/1”的二进制组成，每位的楼层顺序是这样的，最低位代表最低楼层，最高位代表此字节表示楼层范围的最高楼层。

例如，你要控制22层这一层，22层在第三个字节，那么前二层的字节是0x0000，第三个字节的为值为0x20（对应二进制0010 0000，从左到右第6位为1），后面还有5个字节为0x00,0x000,0x00,0x00,0x00，合起来楼层控制字符串：0000200000000000（00 00 20 00 00 00 00 00）

8个FF FF FF FF FF FF FF FF 控制所有层

8个00 00 00 00 00 00 00 00 一层也不能控制

控制 123层 07 00 00 00 00 00 00 00

控制 2 4 9 15 28 39 层 0A 41 00 08 00 40 00 00

0x0A(0000 1010) 0x41(0100 0001) 0x00(0000 0000) 0x08(0000 1000)

0x00(0000 0000) 0x40(0100 0000) 0x00(0000 0000) 0x00(0000 0000)

具体参见“离线开门二维码生成规则”的应用DEMO程序

XXTEA的加密算法，如果没有弄过，请到网上查询相关文档，以下是以C++和JAVA进行加密的范例

///////////////////////////////////////////////////////////////////////////////////////////////////

//加密解密实体函数

//XXTEA加密算法

#define MX (z>>5^y<<2) + (y>>3^z<<4)^(sum^y) + (k[p&3^e]^z)

unsigned int CQR\_SettingDlg::btea(unsigned int \*v, char n, unsigned int \*k)

{

unsigned int z=v[n-1], y=v[0], sum=0, e, DELTA=0x9e3779b9;

long p, q ;

if (n > 1)

{

/\* Coding Part \*/

q = 6 + 52/n;

while (q-- > 0) {

sum += DELTA;

e = (sum >> 2) & 3;

for (p=0; p<n-1; p++) y = v[p+1], z = v[p] += MX;

y = v[0];

z = v[n-1] += MX;

}

return 0 ;

}

else if (n < -1)

{

/\* Decoding Part \*/

n = -n;

q = 6 + 52/n;

sum = q\*DELTA ;

while (sum != 0) {

e = (sum >> 2) & 3;

for (p=n-1; p>0; p--) z = v[p-1], y = v[p] -= MX;

z = v[n-1];

y = v[0] -= MX;

sum -= DELTA;

}

return 0;

}

return 1;

}

///////////////////////////////////////////////////////////////////////////////////////////////////

//加密函数处理

// QRString需要加密的字符串，isNewKey=0

#define QRKEY 0x00543210

CString CQR\_SettingDlg::encryptXXTEA(CString QRString,BOOL isNewKey)

{

unsigned int key[4] = {QRKEY, QRKEY,QRKEY,QRKEY};

unsigned int UserKey,UserKey2;

//将用户设置的KEY放到key[4]的key[0],key[2]

// key[0]= 设备秘钥,默认0x00543210

// key[1]= 用户秘钥

// key[2]= 设备秘钥, 默认0x00543210

// key[3]= 用户秘钥

sscanf(theApp.DeviceKeyStr, "%x",&UserKey);

sscanf(theApp.DeviceKeyStr2, "%x",&UserKey2);

key[0] = key[2] = UserKey;

if(isNewKey)

{

key[1] = key[3] = UserKey;

} else {

key[1] = key[3] = UserKey2;

}

//将16进制数值的字符串，变成8位的整形

int nIntCount=0,nRemainCount=0;

unsigned int a[100]={0};

nIntCount = TranHexStrToInt(QRString,a);//得到这个字符串有几个整形

//余数= 剩余几个字节（一个INT整数为4个字节，不够4字节（一个INT），告诉解码取4的余数是多少）

nRemainCount =(QRString.GetLength()%8)/2;

//XXTEA加密

btea(a, nIntCount,key);

//将加密后的整形值转成字符串

CString StrTmp,encryptStr;

for(int kk=0;kk<nIntCount;kk++)

{

StrTmp.Format("%08X",a[kk]);

encryptStr+=StrTmp;

}

//加密整形的个数+加密字符串

if(isNewKey)

{

nRemainCount=0x7F;

}

StrTmp.Format("%02X%02X",nIntCount,nRemainCount);

//加密的二维码字符串=加密了几个整形，最后一个整数有效字节数+加密秘钥+加密的字符串

encryptStr =StrTmp+theApp.DeviceKeyStr+encryptStr;

return encryptStr;

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////

//解码函数

// QRString需要解密的加密字符串，isNewKey=0

CString CQR\_SettingDlg:: decrypt QR(CString QRString,BOOL isNewKey)

{

int nRetCount=FALSE;

unsigned int a[100]={0};

nRetCount = decryptXXTEA(QRString,a);//得到解码的整形数有几个

if(nRetCount)

{

unsigned int key[4] = {QRKEY, QRKEY,QRKEY,QRKEY};

unsigned int UserKey,UserKey2;

//将用户设置的KEY放到key[4]的key[0],key[2]

sscanf(theApp.DeviceKeyStr, "%x",&UserKey);

sscanf(theApp.DeviceKeyStr2, "%x",&UserKey2);

key[0] = key[2] = UserKey;

if(isNewKey) {

key[1] = key[3] = UserKey;

} else {

key[1] = key[3] = UserKey2;

}

//XXTEA解密

btea(a, -nRetCount,key);

//得到加密后的字符串

//字符串最后一个字节，为了避免出现00001234前面有0000字符串

//所以先将数值转换成字符串，如果字符串长度为奇数，则前面补加0，补齐

CString StrTmp,DecryptStr;

//得到字符串（除最后一个字节外）

for(int kk=0;kk<nRetCount-1;kk++)//得到解码的字符串，除最后一个整形值特殊处理外

{

StrTmp.Format("%08X",a[kk]);

DecryptStr+=StrTmp;

}

int nRemainCount=0;

//得到4取整的余数值

sscanf(QRStringEnd.Mid(2,2), "%X",&nRemainCount);

//最后一个整形数的处理

StrTmp.Format("%08X",a[nRetCount-1]);

//没有余数 或者是新的设备秘钥（设置设备秘钥的指令是没有余数的）

if((nRemainCount==0)||(nRemainCount==0x7F)) {

DecryptStr+=StrTmp;

}

else if(nRemainCount<=3)//余数1-3，取最后一个数的后面的字符

{

DecryptStr+=StrTmp.Right(nRemainCount\*2);

}

//加密整形的个数+加密字符串

StrTmp.Format("%02X%02X",nRetCount,nRemainCount);

}

}