铁塔智能锁接口方案

1. 蓝牙名称

电控锁命名规范 -TL-XXXXXXXX (XX-4字节十六进制)

蓝牙钥匙命名规范 -TK-XXXXXXXX (XX-4字节十六进制)

1. 锁密钥及加密方式

锁密钥在初始化时写入锁中，密钥是长度为8字节。对称加密采用RC4加密算法，对锁密钥钥匙进行加密。

1. 蓝牙通信方式

低功耗蓝牙采用统一的UUID进行读写操作，APP向蓝牙设备写数据使用写通道，UUID为0000ff01-0000-1000-8000-00805f9b34fb，APP从蓝牙设备获取数据使用通知通道，UUID为0000ff02-0000-1000-8000-00805f9b34fb。

1. 流程接口及定义

手机app

电控锁

返回蓝牙锁信息（设备类型1byte设备号4byte软件版本1bytes硬件版本1byte电量 1byte初始化状态1byte）

读取蓝牙锁信息

获取电控锁状态

手机APP-电控锁 接口

返回初始化锁（结果1byte1成功 其它失败）

初始化电控锁（管理密钥8byte组织密钥8byte效验和1byte）会话密钥加密

初始化电控锁

返回开锁（结果1byte1成功 其它失败）

开锁（APP用户ID 4byte锁设备号 4byte时间世纪秒 4byte动作 1byte）管理密钥加密

开锁

返回校时（1成功 其它失败）

修改管理密钥（组织密钥8byte管理密钥8byte）效验码（1byte）组织密钥加密

修改管理密码

返回请求密钥（电控锁产生8字节随机数）

请求会话密钥

返回开锁日志（APP用户ID时间世纪秒4byte）

读取开锁日志

读取开锁日志

读取结果长度不为0

蓝牙钥匙

手机app

返回蓝牙锁信息（设备类型1byte设备号4byte软件版本1bytes硬件版本1byte电量 1byte初始化状态 1byte）

读取蓝牙钥匙信息

获取蓝牙钥匙状态

手机APP-蓝牙钥匙 接口

返回初始化锁（结果1byte1成功 其它失败）

设置开锁权限（锁设备号4byte权限开始时间 4byte权限结束时间4byte效验和1byte）管理密钥加密

设置开锁权限

返回开锁（结果1byte1成功 其它失败）

开锁（ 时间世纪秒 4byte效验和 1byte）管理密钥加密 蓝牙连接钥匙后发送1次校时指令

校时

返回开锁日志（钥匙设备号4byte时间世纪秒4byte）

读取开锁日志

读取开锁日志

权限未设置完

读取结果长度不为0

返回初始化锁（结果1byte1成功 其它失败）

初始化电控锁（管理密钥 8byte组织密钥8byte效验和 1byte）会话密钥加密

初始化电控锁

返回请求密钥（电控锁产生8字节随机数）

请求会话密钥

返回校时（1成功 其它失败）

修改管理密钥（组织密钥8byte管理密钥8byte）效验码（1byte）组织密钥加密

修改管理密码

1. 接口格式及参数定义

数据加密方式定义：

RC4对称密钥加密方式（密钥长度规定为8字节）

蓝牙数据传输采用如下格式：

原始数据格式：

|  |  |  |  |
| --- | --- | --- | --- |
| T | L | V | C |
| 指令字 | 长度 | 数据内容 | 校验位 |
| 1字节 | 1字节 | 长度为length | 1字节 |

T：指令编号，长度1字节

L：数据内容长度。长度为1个字节

V：数据内容，长度为L。对应以下接口中的输入或输出参数

C：校验位，长度为1字节。值为数据内容区所有数据异或值。

1. 接口定义
   1. 读取设备信息信息

|  |  |
| --- | --- |
| **接口** | 读取门锁信息（指令字：0x01） |
| **功能描述** | 获取状态信息 |
| **输入参数** | 无 |
| **输出参数** | 设备号（4字节）、设备类型（1字节 0电控锁，1蓝牙钥匙）、硬件版本信息（1字节 高4位表示大版本号 低4位表示小版本号）软件版本信息（1字节 高4位表示大版本号 低4位表示小版本号）、状态（2字节）<电控锁指示门状态1字节 0开门 1关门 锁舌状态1字节 0锁舌缩回 1锁舌伸出 钥匙 电池电压2字节 405表示 4.05 V>、初始化状态（1字节 0：未初始化 1：已初始化） |
| **接收方** | 电控锁/ 蓝牙钥匙 |
| **发送方** | APP |
| **通信加密方式及验证过程** | 输出参数采用明文返回 |

* 1. 请求会话密钥

|  |  |
| --- | --- |
| **接口** | 请求会话密钥（指令字：0x02） |
| **功能描述** | 在初始化前请求一个随机密钥由设备端提供 |
| **输入参数** | 无 |
| **输出参数** | 会话密钥（8字节） |
| **接收方** | 电控锁/ 蓝牙钥匙 |
| **发送方** | APP |
| **通信加密方式及验证过程** | 输出参数采用明文返回 |

* 1. 初始化设备

|  |  |
| --- | --- |
| **接口** | 初始化设备（指令字：0x03） |
| **功能描述** | 在初始化锁时配置初始密钥及相关参数 |
| **输入参数** | 管理密钥（8个字节）、组织密钥（8个字节）效验字（1字节）效验采用异或效验 验证解密是否正确 不是报文结尾的效验字 |
| **输出参数** | 处理结果（1字节，1：成功 2：已安装） |
| **接收方** | 电控锁/ 蓝牙钥匙 |
| **发送方** | APP |
| **通信加密方式及验证过程** | 输入参数使用随机会话密钥RC4加密，输出参数明文传输 |

* 1. 开门

|  |  |
| --- | --- |
| **接口** | 开门（指令字：0x04） |
| **功能描述** | 蓝牙钥匙与锁连接成功，由APP发授权信息给锁，锁收到信息后进行权限检查、执行相应操作 |
| **输入参数** | APP用户ID(4字节)、锁设备号（4字节）、时间（世纪秒4字节）、动作（1字节 1：开门 2：关门） |
| **输出参数** | 处理结果（1字节，1：成功 其它：失败） |
| **接收方** | 锁 |
| **发送方** | APP |
| **通信加密方式及验证过程** | 输入参数使用会话密钥采用RC4加密算法加密，锁收到指令后，先用会话密钥解码，对于解出的授权信息，使用初始密钥再进行解码，对比用户编号，通过后执行操作，返回明文处理结果 |

* 1. 重置密钥

|  |  |
| --- | --- |
| **接口** | 重置管理密钥（指令字：0x05） |
| **功能描述** | 重置管理密钥 |
| **输入参数** | 组织密钥（8字节）管理密钥（8字节）效验码（1字节） |
| **输出参数** | 处理结果（1字节，1：成功 其它：失败） |
| **接收方** | 电控锁 蓝牙钥匙 |
| **发送方** | APP |
| **通信加密方式及验证过程** | 输入输出参数用组织密钥采用RC4加密算法加密 |

* 1. 设备日志记录

|  |  |
| --- | --- |
| **接口** | 设备日志（指令字：0x06） |
| **功能描述** | 设备对产生的开关门日志自动进行存储并在有手机连接时通过手机回传到管理平台 |
| **输入参数** | 无 |
| **输出参数** | APP用户ID或 钥匙ID(4字节) 日志时间（4字节 世纪秒） 如果电控锁或钥匙中已没用记录 返回报文长度0 返回内容 无 |
| **接收方** | 电控锁 蓝牙钥匙 |
| **发送方** | APP |
| **通信加密方式及验证过程** | 明文传输 |

* 1. 设置钥匙权限

|  |  |
| --- | --- |
| **接口** | 设置钥匙权限（指令字：0x10） |
| **功能描述** | 通过APP对钥匙进行授权 |
| **输入参数** | 锁设备号（4字节）权限开始时间（4字节世纪秒）授权结束时间（4字节世纪秒）效验和（1字节） |
| **输出参数** | 处理结果（1字节，1：成功 其它：失败） |
| **接收方** | 蓝牙钥匙 |
| **发送方** | APP |
| **通信加密方式及验证过程** | 管理密钥加密 输出参数采用明文返回 |

* 1. 校时

|  |  |
| --- | --- |
| **接口** | 校时（指令字：0x11） |
| **功能描述** | 通过APP对钥匙校时 |
| **输入参数** | 时间（4字节世纪秒）校验码（1字节） |
| **输出参数** | 处理结果（1字节，1：成功 其它：失败） |
| **接收方** | 蓝牙钥匙 |
| **发送方** | APP |
| **通信加密方式及验证过程** | 管理密钥加密 输出参数采用明文返回 |

以下是RC4算法的JAVA实现。解密和加密都是调用该方法。

public static byte[] HloveyRC4(byte[] aInput, byte[] aKey) {

int[] iS = new int[256];

byte[] iK = new byte[256];

for (int i = 0; i < 256; i++)

iS[i] = i;

int j = 1;

for (short i = 0; i < 256; i++) {

iK[i] = aKey[(i % aKey.length)];

}

j = 0;

for (int i = 0; i < 256; i++) {

j = (j + iS[i]&0xff + iK[i]&0xff) % 256;

int temp = iS[i];

iS[i] = iS[j];

iS[j] = temp;

}

int i = 0;

j = 0;

byte[] iOutputChar = new byte[aInput.length];

for (int x = 0; x < iOutputChar.length; x++) {

i = (i + 1) % 256;

j = (j + iS[i]&0xff) % 256;

int temp = iS[i]&0xff;

iS[i] = iS[j];

iS[j] = temp;

int t = (iS[i] + (iS[j]&0xff % 256)) % 256;

int iY = iS[t]&0xff;

byte iCY = (byte) iY;

iOutputChar[x] = (byte) (aInput[x] ^ iCY);

}

return iOutputChar;

}