JoyLink-Bluetooth 协议 V1.93

京东智能协议组

修订记录：

版本号	修订人	修订日期	修订描述
V 1.91		2017.5.19	增加 7.2 BLE+一键配置配网流程的说明
V 1.92		2017.7.19	增加第 5 章，带安全芯片的设备认证部分，增加第 8 张配网失败后提示用户重新配置的说明
V 1.93		2017.7.21	删除了一个安全级别，只留下 plaintext;psk;ecdh

1 关键词定义及约定

1.1. 关键词定义

AES 算法标准：AES128 CBC/PKCS#5 padding。

GUID：合法用户从云端申请并写入设备的唯一 ID 值，32 字节，是设备的唯一标识。

PUID：某类产品（同一品牌、同一批次、同一规格）的标识码，6 字节固定长数字和字母组合。是系统生成的产品标识码。

BRAND:品牌编号。

CID：品类编号。

property：设备的每一项操作或能力，定义为一个属性(property)，例如：开，关，温度提升 1 度。属性对应的数值按能力不同，阈值也不同。对一个属性的一次操作会由 Operate + TLV 描述。

TLV:是一个结构体，Tag + Length +Value。

Tag: 给某个 property 遍的一个号码。

1.2. 约定

本协议只针对 BLE 设备的连接及数据传输，对传统蓝牙设备不适用。

2 协议说明

2.1. 关键词定义

JoyLink-Bluetooth 协议，按照角色可以把智能硬件分为以下三种：

蓝牙设备(Slave)： 支持 BLE 协议的设备。

控制终端(Master)： 与用户产生交互的控制端，指令的发起方，同时也是信息的查询窗口。可以是手机或者是支持 BLE 的网关。

云端： 提供后台服务、存储蓝牙设备信息的具有公网 IP 的服务器端。

本协议是描述蓝牙设备与主控、云端如何组成系统，之间如何通讯，如何管理的应用层协议。

2.2. Profile 规定

2.2.1 Profile 设计理念

系统总体架构如下图所示：

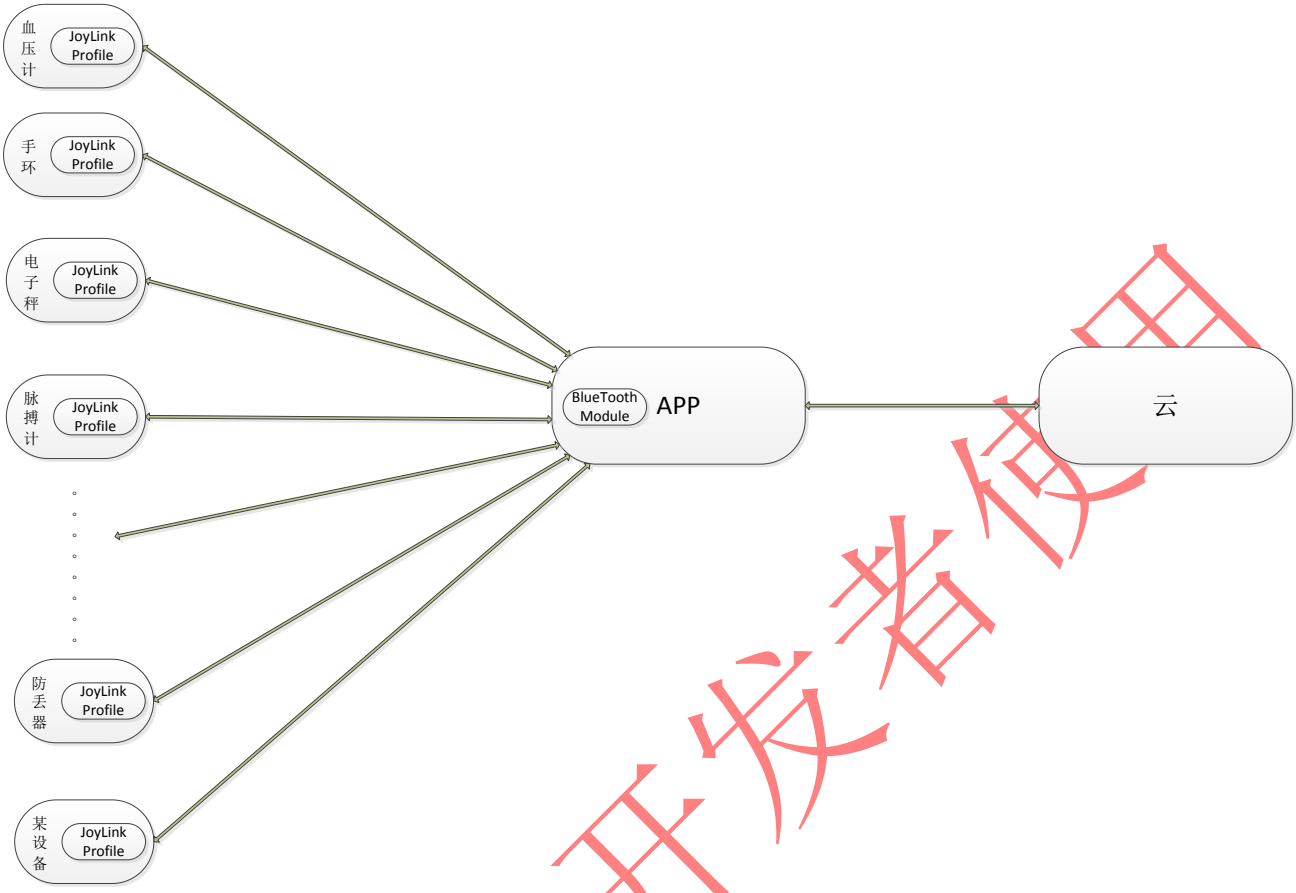


图 2.1

手机 APP 要能够通过这个 JoyLink Profile 和各种蓝牙设备交换信息，操控各种蓝牙设备。前提是这个蓝牙设备和手机 APP 都有对本 profile 的支持。

本协议专注于通信和安全，其中安全从属于通信。

2.2.2 Profile 实现形式

Profile 具体实现形式是一个 libjoylink.a 库。第三方厂商在生产设备时链接入这个 libjoylink.a，并实现这个 lib 库所要求的回调函数，并使用这个 lib 库实现自己的应用，那么它就可以被支持本 profile 的手机 APP 识别和操控。设备 App 层也需要有对协议运行的一些支持。

设备端软件架构是：

图 2.2

2.2.3 Profile 架构

本 profile 只有一个 service。该 service 由三个 characteristic 组成。

名称	值	作用
Profile Service UUID	BLE_UUID_JOYLINK_SERVICE	蓝牙设备的 service
Write Characteristic UUID	UUID_JOYLINK_WRITE	APP 向设备传输数据
Indicate Characteristic UUID	UUID_JOYLINK_INDICATE	设备向 APP 传输数据（有回执）
Read Characteristic UUID	UUID_JOYLINK_READ	设备不发广播时标识设备

```
#define BLE_UUID_JOYLINK_SERVICE      0x00, 0x00, 0xFE, 0x70, 0x00, 0x00, 0x10, 0x00, 0x80, 0x00, 0x00, 0x80, 0x5F, 0x9B, 0x34, 0xFB
#define UUID_JOYLINK_WRITE            0x00, 0x00, 0xFE, 0x71, 0x00, 0x00, 0x10, 0x00, 0x80, 0x00, 0x00, 0x80, 0x5F, 0x9B, 0x34, 0xFB
#define UUID_JOYLINK_INDICATE         0x00, 0x00, 0xFE, 0x72, 0x00, 0x00, 0x10, 0x00, 0x80, 0x00, 0x00, 0x80, 0x5F, 0x9B, 0x34, 0xFB
#define UUID_JOYLINK_READ             0x00, 0x00, 0xFE, 0x73, 0x00, 0x00, 0x10, 0x00, 0x80, 0x00, 0x00, 0x80, 0x5F, 0x9B, 0x34, 0xFB

BLE_UUID_JOYLINK_SERVICE    0000FE70-0000-1000-8000-00805F9B34FB
UUID_JOYLINK_WRITE          0000FE71-0000-1000-8000-00805F9B34FB
UUID_JOYLINK_INDICATE       0000FE72-0000-1000-8000-00805F9B34FB
UUID_JOYLINK_READ           0000FE73-0000-1000-8000-00805F9B34FB
```

128bit 的 UUID 和 16bit 的 UUID 等价，本协议也支持具体实现时使用 16bit UUID。

示意图：

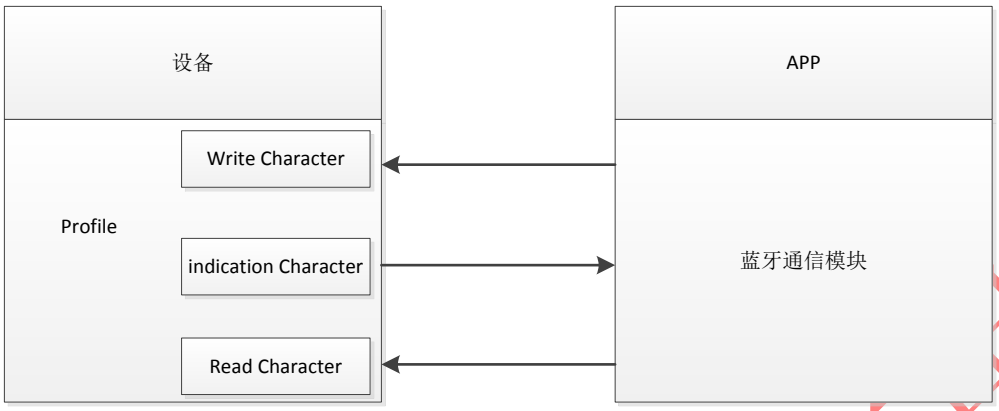


图 2.3

2.3. 通信过程描述

这里我们约定，一个 Characteristic 一次传输的数据称为一帧(frame)，一帧大小为 20 字节。
注意：应用层上的数据包（例如 1KB 大小），会分解成许多帧来传输。
数据通信过程都只支持合法的蓝牙通信。对于违反自动机运行规则的行为，SDK 都以错误码的形式返回给 APP，并且在 SDK 内部毁掉本次通信已经取得的成果，清理内存，为下次通信做好准备，并且会给对端发送一个失败信息。一个 Operate 就是一个自动机。
SDK 支持全双工操作。

2.3.1 蓝牙设备发送数据

分帧：假设设备有 1KB 数据，这 1KB 数据定义为 1 个包(Packet)，要发送给 APP。由于一个特征值长度有限（如 20 个字节），显然需要分多次才能传输完成。1KB 数据，要分成 1024B/20B = 51 个帧。剩下的 4 个字节，不足一帧（20 个字节），需补齐为一帧并对剩下的 16 个字节赋 0 值。总共是 52 帧。
发送第一个帧：把第一个帧的内容放入 indication characteristic 里面。然后通知手机读取数据，通知有两种方式， indication 和 notify，本 Profile 只支持 indication。当通知完成的时候，可以认为手机已经读完数据，
这就完成了发送第一个帧。
按照发送第一帧的步骤，依次发送剩下的帧。

2.3.2 蓝牙设备接收数据

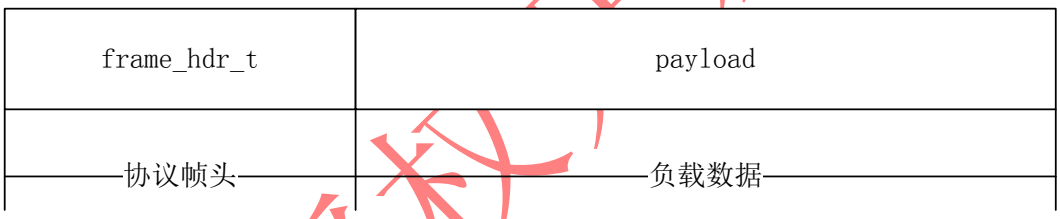
合帧：假设 APP 有 1KB 数据，这 1KB 数据定义为 1 个数据包，要发送给设备。由于一个特征值长度有限（如 20 个字节），显然需要分多次才能传输完成。1KB 数据，要分成 1024B/20B = 51 个帧。剩下的 4 个字节，不足一帧（20 个字节），需补齐为一帧并对剩下的 16 个字节赋 0 值。总共是 52 帧。

当蓝牙设备发现 write characteristic 收到数据的时候，就接收数据，直到一包数据的所有帧全部收完之后，通过合帧还原出这包数据，然后把这包数据返回给设备应用层。

3 数据通信

数据格式分两层：通信层面，业务层面。通信层面是为 SDK 服务，业务层面是为 App 服务。

3.1. 通信层面



一个 characteristic 一次传输的数据定义为一帧,每帧数据长度 20 BYTES。

payload:这个帧运输的数据。在通信层面看来，palyload 就是 raw data,没有任何格式。payload 的格式和语义要由业务层面来解析。payload 统一为 16 字节，没有例外。如果不足 16 字节就用 0x00 补齐。

加密方式：使用 128bit 的密钥，使用 AES 算法，对 payload 字段进行加密。

解密方式：使用 128bit 的密钥，使用 AES 算法，对 payload 字段进行解密。

```
typedef struct {
    unsigned char   count;     //这次传输的数据包所含有的总帧数
    unsigned char    num;      //帧序号，即当前是第几帧，从 0 开始编号
    unsigned char   keytype;   //payload 部分加解密密钥类型：
                               //0 –明文，
                               //1– PSK(Pre Shared Key)，
                               //2 – secretkey(ECDH algorithm),
    unsigned char   seq;       //本帧所属数据包的序列号,序列号从 0 到 255 循环递增
}frame_hdr_t;
```

3.2 业务层面

一包数据由一帧或者多帧的 payload 字段组合而成。一包数据能且只能是一种 Operate。
在接受一包数据时，把该包数据所有的帧收齐了之后，这些帧通过合帧操作，组成一个 seq+ operate+ length+content+ crc 的结构。
在发送一包数据时，数据构成一个 seq+ operate+ length+ content+ crc 的结构，这包数据通过分帧操作之后，放到 BLE 信道上发送出去。

seq	operate	length	content	crc
1 Byte	1 Byte	2 Byte	n Byte	2Byte

seq 字段用于超时重传造成的去重。超时的阈值由实现自己决定，协议不做限制。应答包的 seq 须与请求包的 seq 一致。
crc 字段用于检错,是对 seq+operate+length+content 字段计算校验和。

3.2.1 通信包

请求 Operate 列表如下：

编号	意义
0x01	APP 读 Device 的 property
0x11	Device 反馈 APP 读 property 命令
0x02	APP 写 Device 的 property,不带结果反馈
0x03	APP 写 Device 的 property,带结果反馈
0x13	Device 反馈 APP 写 property 命令的执行结果
0x16	Device 向 APP Indicate 数据，不带结果反馈
0x17	Device 向 APP Indicate 数据，带结果反馈
0x07	APP 反馈 device Indicate 的执行结果

每个 Operate 都有各自定义的 Content。
Length 表征 Content 的长度。

Content 由 0 个、1 个或者多个 TLV 组成。

Content：

TLV0	TLV1	……	TLVn
x Byte	y Byte	…….	z Byte

厂商的一类产品分配一个 PUID，一个 PUID 对应一组 property，每个 property 对应一个 tag。Tag 的编号是局部的而不是全局的，也就是说不同的 PUID，可以使用同一个 tag 号表示各自的 property。

每个 TLV 格式如下：

Tag ID	Length	Value
2 Byte	1 Byte	x Byte

Tag：当前通信要操作哪个 Property,它的 Tag

Length：Value 的长度

Value：这个 Property 本次通信的内容。

Value 本身可以无结构，也可以有结构。无结构比如可以是 char，short，int，long，一维数组。有结构比如可以是结构体，甚至 Value 本身也可嵌套 TLV，这个由产品根据需要自己定义，协议不做限制。

3.2.2 异常反馈包

当通信遭遇失败时，要 reset sdk，并且向对端发送异常反馈包。Operate 定义如下：

seq	operate	length	content	crc
1 Byte	1 Byte	2 Byte	n Byte	2Byte

Operate 编号	意义
0xff	这个 seq 的请求包的接收情况反馈

这个 Operate 的 Content 是 1 个字节，并非 tlv 结构，而是 unsigned char。

Length 表征 Content 的长度,值为 1。

Content 列表：

值	意义
1	crc 校验出错

2	缺帧超时，未能在一秒内传完一包数据
3	帧乱序,num 出错
4	收帧过程中收到非法序列号 seq

对端收到异常反馈包，要酌情处理，并重传这个 seq 的通信包。

3.3 补充规定

3.3.1 写结果反馈编码

关于 Operate 0x13.

0x13	Device 反馈 APP 写 property 命令的执行结果

其结果反馈统一定义如下：

Result 列表如下：

编号	意义
0x00	写操作成功
0x01	写操作失败，失败原因为 1
0x02	写操作失败，失败原因为 2
0x03	写操作失败，失败原因为 3
0x04	写操作失败，失败原因为 4
0x05	写操作失败，失败原因为 5

协议在此留有扩展空间，最多可以有 255 种错误编码，产品可以根据需要自己定义错误码含义，自己扩展错误码。

另外，错误码之后可以附带错误描述信息，也可以不附带错误描述信息。举例如下：

手机写了 tag 0xXXXX,结果以如下方式反馈：

失败，不带错误描述信息：

Tag ID	Length	Value
0xXXXX	1	3

表示 tag 0xXXXX 写失败了，错误码是 3。

失败，带错误描述信息：

Tag ID	Length	Value	
0xXXXX	14	3	hardware error

成功：

Tag ID	Length	Value
0xXXXX	1	0

3.3.2 特殊 property 定义

协议使用 0xF000-0xFFFF 作为特殊 property 使用，适用于所有产品。具体产品中不能使用 0xF000-0xFFFF 的 TAG ID，防止与特殊 property 冲突。

Property	Tag ID	Type	Length	描述
PUID	0xFFFF	uchar 数组	6	这个产品的 PUID
GUID	0xFFFC	uchar 数组	32	这个产品的 GUID
LOCAL_KEY	0xFFFB	uchar 数组	16	蓝牙设备的 local key
dev_snapshot	0xFEFF	uchar 数组	x	表示蓝牙设备快照
gender	0xFEFE	uchar	1	用户性别，0 女性，1 男性
age	0xFEFD	uchar	1	用户年龄
height	0xFEFC	uchar	1	用户身高，单位：厘米
time	0xFEFA	uint	4	用于存储时间
App_pub_key	0xFEF9	uchar	21	压缩手机公钥，用于 ECDH

Dev_pub_key	0xFEF8	uchar	21	压缩设备公钥，用于 ECDH
SSID	0xFEF7	uchar	x	蓝牙配网设备，路由器 SSID
PASSWORD	0xFEF6	uchar	x	蓝牙配网设备，路由器密码
SECLEVEL	0xFEF5	uchar	1	设备安全级别
BRAND	0xFEF4	uchar 数组	2	设备品牌编号
CID	0xFEF3	Uchar 数组	2	设备品类编号
BLE_DEV_CTL	0xFEF2	uchar	1	1- 蓝牙断开连接进入广播态 2- 蓝牙断开连接并关闭蓝牙 3- 自动化测试用，复位设备重置配网 Others: reserve
WIFI CONNECT STATUS	0xFEF1	uchar	1	用于支持蓝牙配网的模块上报 WIFI 连接状态，数据值定义： 0- disconnected 1- connecting 2- connected 3- 未扫描到对应的路由 4- 因其他原因连接失败 Others: reserve

u 表示 unsigned,无符号。x 表示长度不固定，根据产品不同而不同。

4 设备发现连接

设备发现主要是指设备广播、APP 扫描到设备的过程。

蓝牙设备在未处于连接态时，需要一直处于广播态，广播数据中包含 Service UUID 和 manufacture specific data。

广播数据需要使用 TYPE=0x03(完整的 16 位 bit UUID)，在其中携带本协议的 service UUID 0xFE70，标识设备支持本协议

名称	Length	Value
Service UUID	2	0xFE70

另外使用 TYPE=0xFF(厂商自定义数据)的前 14 个字节如下数据。

名称	Length	Data
Manufacture specific data	14	PUID(6 BYTE) + MACADDR(6 BYTE) + VERSEC (1 BYTE) + EXPANDTAG(1 BYTE)

其余的广播字段，厂商可根据自己的产品自行定义。

所有通信采用小端序（Little Endian）。

设备还需要暴露一个 read characteristic。当手机上的其他 APP 连接上设备时，设备不会再广播，支持本 Profile 的 APP 会读取该特征值，以确定是否要和这个设备通信。

manufacture specific data 和 read characteristic 数据内容一致，为：

PUID	MACADDR	VERSEC	EXPANDTAG
6 BYTE	6 BYTE	1 BYTE	1 BYTE

字段说明：

1. PUID
   产品字符编号，注册产品时获取
2. MACADDR
   BLE 设备的 MAC 地址
3. VERSEC
   Version+Security Level，协议版本号+安全级别。
   其中，高半字节 H 用于标识协议版本号：
   本协议固定定义为 0x00

   低半字节 L 用于标识安全级别：
   会话密钥的生成方式分别为
   0- 安全级别为 0，详见 §6.1
   1- 安全级别为 1，详见 §6.2
   2- 安全级别为 2，详见 §6.3

4. EXPANDTAG
   扩展标识，用于标识芯片信息，我们用 0xHL 标识字节
   其中，高半字节 H 用于标识芯片类型：
   0-BLE 设备
   1-BLE+WIFI 设备，支持通过 BLE 控制 WIFI 入网
   低半字节 L 用于标识，WIFI 设备当前的入网状态：
   0- disconnected

1- connecting
2- connected
3- 未扫描到对应的路由
4- 因其他原因连接失败

5.设备认证

为防止攻击，协议提供一种基于安全芯片的认证机制。此认证方式，从产品的角度保证产品不被攻击，协议建议但并不强制使用。

5.1 设备证书写入

（1）（A1）颁发机构读取授权模块的公钥，使用自己的私钥对授权模块的公钥进行签名，作为授权证书写入到授权模块中（A2）,完成授权方对设备厂商或者制造厂商的授权。

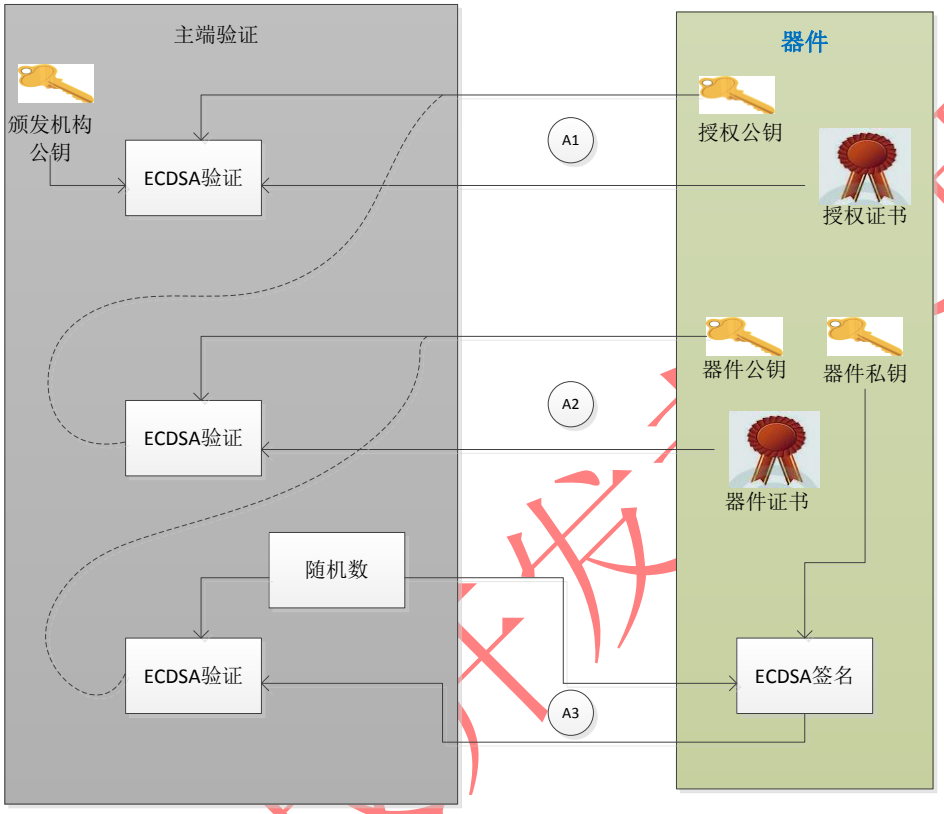
（2）（A3）授权模块读取安全芯片的公钥，使用自己的私钥对安全芯片的公钥进行签名，作为器件证书写入到安全芯片中（A4）

（3）授权模块将由（1）生成的授权证书和自己的公钥写入到安全芯片中，以完成安全芯片的写入流程。

说明：

授权方、授权模块和安全芯片各自生成自己的公私钥，私钥不能被读出或泄露。

5.2 设备认证



（1）（A1）验证端读取存储在安全芯片中的授权公钥与授权证书，使用授权方的公钥进行验签，以验证授权证书。

（2）（A2）验证端读取器件的公钥与器件证书，使用授权公钥对器件证书进行验签，以验证器件证书。

（3）（A3）验证端生成一组随机数发送给安全芯片进行签名，完成对芯片能力的验证。

6 安全机制

加密强度： AES 对称密钥长度为 128bit。

AES 算法标准：AES128 CBC/PKCS5 padding。

AES 加解密的对象是对每帧的 payload 加解密。

6.1 安全级别为 0

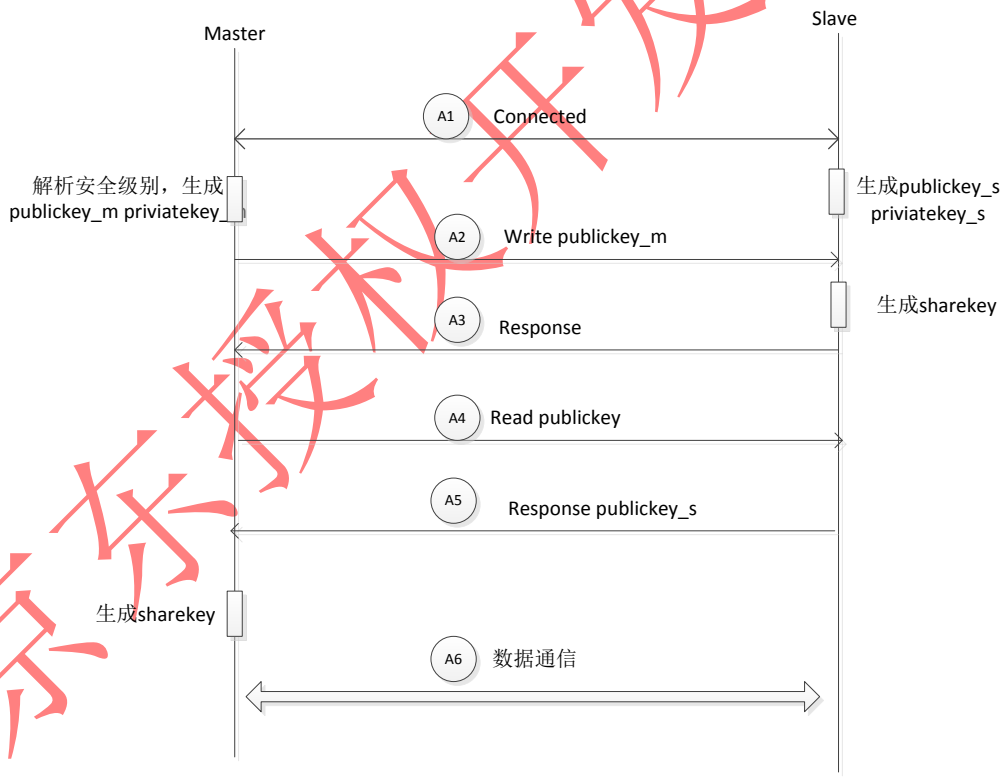
通信双方（手机，设备）在应用层直接使用明文传输，此时在 BLE 协议层可以使用 No Security, Just Works, Passkey Entry, Out of Band 等四种安全模式中的任意一种，具体根据产品的需求决定。

6.2 安全级别为 1

协议层采用 no security，应用层通信双方（手机，设备）直接使用预共享秘钥加密传输。

6.3 安全级别为 2

协议层采用 no security，应用层安全方案如下：
通信双方使用 ECDH 算法协商出 secretkey，使用这个 secretkey 对通信进行加解密。具体流程如下



1. (A1)Master 与 slave 建立 BLE 连接，若设备支持安全级别 3，双方各自生成自己的 publickey 与 priviatekey
2. (A2)Master 将自己的 publickey_m 写入到设备中。
3. (A3)Slave 收到 publickey_m 后，用 publickey_m 与 priviatekey_s 生成 sharekey。并返回结果给 Master
4. (A4)Macter 发送读取指令，获取 slave 端公钥 publickey_s
5. (A5)Slave 将自己的公钥 publickey_s Indicate 给 Master,Master 收到设备的公钥后生成 sharekey

6. (A6)双方用各自生成的 sharekey 进行数据通信

7 数据通信流程

7.1 控制终端读蓝牙设备数据

分两步：

（1）手机端使用 operate 0x01 读 property。

（2）设备端使用 operate 0x11 返回相应 property 的值。

协议支持批量读。

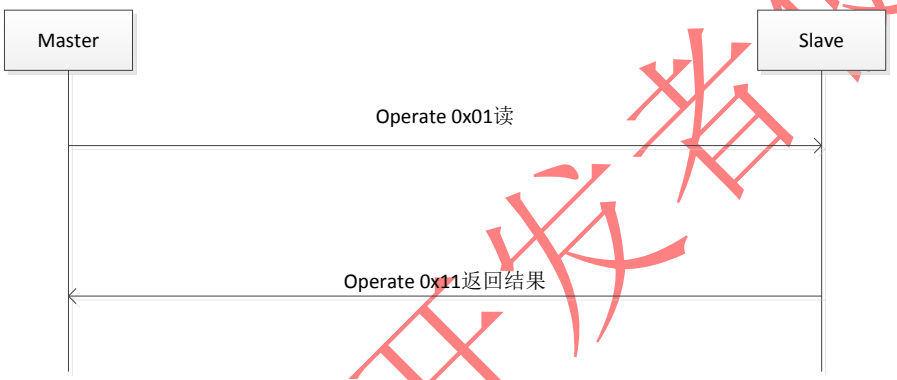


图 6.1

7.2 控制终端写蓝牙设备

分两种。带结果反馈和不带结果反馈。

7.2.1 不带结果反馈

手机端使用 operate 0x02 写设备即可。

协议支持批量写。

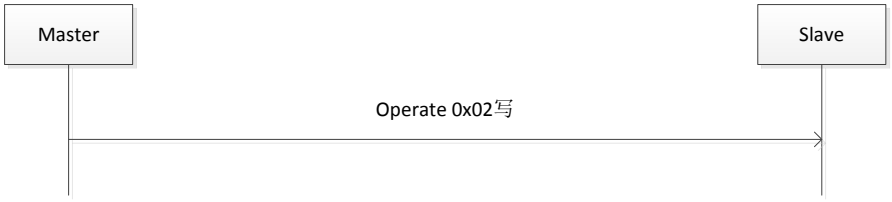


图 6.2

7.2.2 带结果反馈

分两步：
（1）手机端使用 operate 0x03 写 property。
（2）设备端使用 operate 0x13 返回写 property 的执行结果。
协议支持批量写。

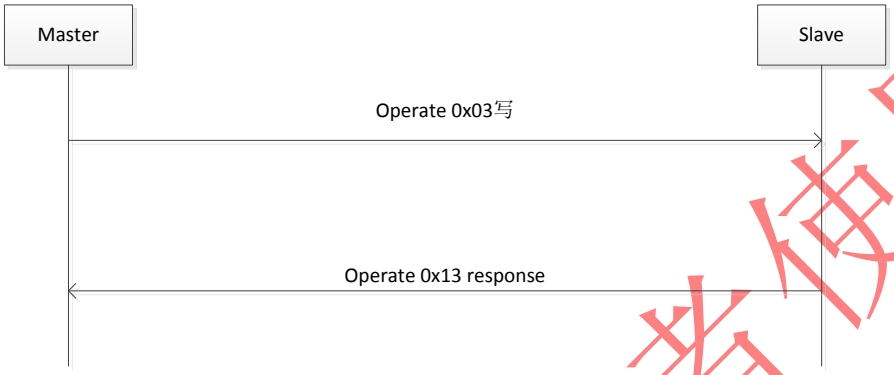


图 6.3

写结果反馈统一定义如下：

编号	意义
0x00	写操作成功
0x01	写操作失败，失败原因为 1
0x02	写操作失败，失败原因为 2
0x03	写操作失败，失败原因为 3
0x04	写操作失败，失败原因为 4
0x05	写操作失败，失败原因为 5
0x06	写操作失败，失败原因为 6

可以继续扩展。

7.3 设备主动向控制终端 indicate 数据

分两种：不带结果反馈和带结果反馈。

7.3.1 不带结果反馈

Device 使用 operate 0x16 主动向 APP 发送数据，不带结果反馈。

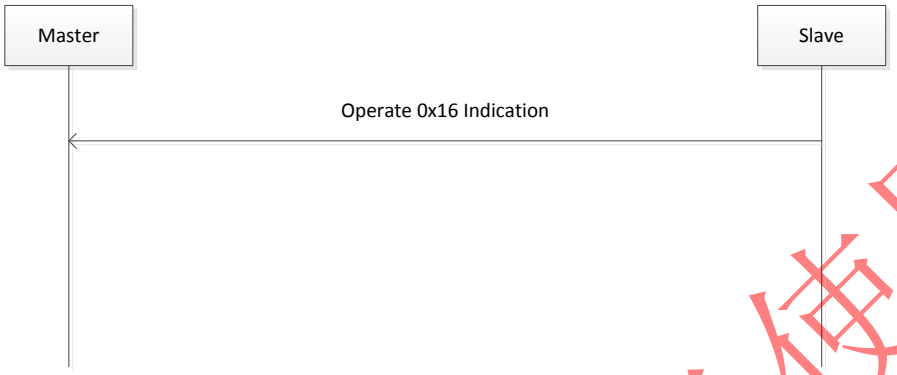


图 6.4

7.3.2 带结果反馈

Device 使用 operate 0x17 主动向 APP 发送数据, APP 使用 0x07 反馈结果。

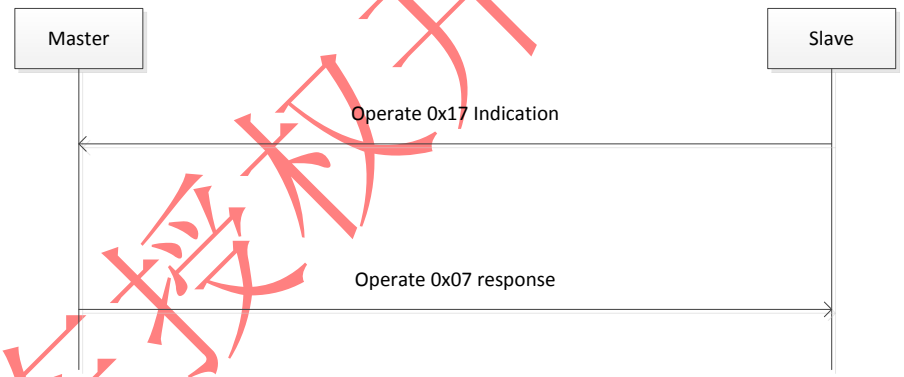


图 6.5

8 Wi-Fi 配网实现

8.1 BLE 配网流程

对于同时支持 WIFI 与 BLE 的芯片或模组，joylink 协议提供一种通过 BLE 对 wifi 模组进行入网配置的功能，流程见图 7_1。

模块需要保证设备每次进入配网模式最长时间为 10 分钟，之后失效，若重新进入配网模式，必须重新上电或重置入网。

设备收到配网信息以后，协议提供两种监测设备 WIFI 连接状态的方式：L1 通过 BLE 广播的方式，对于多个设备同时配网的情况，建议采用这种方式；L2 手机与设备不断开蓝牙连接，设备通过特殊的 property 主动上报连接状态，对于单个设备配网的方式，我们建议扫描二维码的方式并比对 MAC 地址，以减小误绑定。

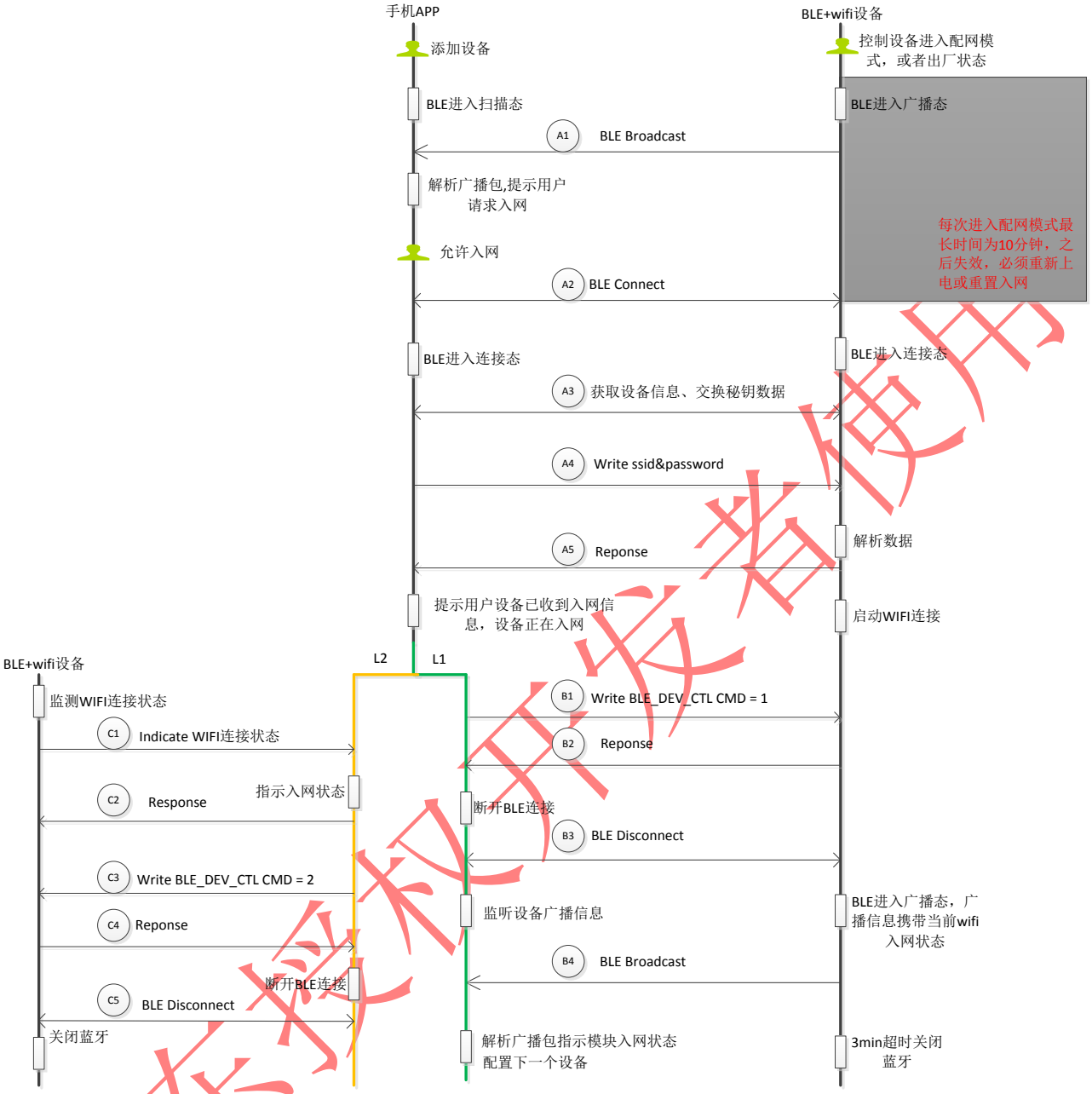


图 7_1

1. 手机默认进入扫描态，监听广播信息。
2. （A1）设备处于广播态，广播设备信息，广播帧见 4.设备发现连接。
3. 手机端有添加设备操作时，解析广播数据，判断设备是否支持 joylink 协议（判断 ADtype=0x03 的 UUID 是否等于 0xFE70），及模块信息中数据是否支持 wifi 配网及 wifi 配网状态（ATtype=0xFF 中 EXPENDTAG 字段），若满足配网条件，则提示用户设备请求入网（设备信息根据 PUID 从云端获取）。
4. （A2）若用户允许设备入网，则手机端与设备建立 BLE 连接。
5. （A3）设备连接成功后，首先获取设备信息、交换秘钥数据。
6. （A4）将 SSID 与 PASSWORD 按 3.3.2 定义的 property 写入到设备中。
7. （A5）设备解析数据并返回解析结果。
8. 若返回信息成功，手机端选择提示设备已收到入网信息，wifi 处于 connecting 状态，设备

将入网信息告知 wifi 模块启动连网。

9. 手机端选择流程 L1 或 L2 获取蓝牙配网状态，L1 通过 BLE 广播告知 WIFI 连接状态，L2 通过特殊 property（见 3.3.2 定义）告知连接状态。

L1（多个设备配网-通过 BLE 广播的数据监测配网状态，对多个设备同时配网时建议采用这种方式）：

1. （B1）手机端写入 BLE_DEV_CTL 命令给设备,参数等于 1，告知设备 BLE 断开连接后进入广播状态
2. （B2）设备返回结果给手机端
3. （B3）手机端收到 response 后断开蓝牙连接
4. （B4）设备广播数据中携带 wifi 连接信息（ATtype=0xFF 中 EXPENDTAG 字段）,手机端监听广播数据中设备的连接状态。
5. 设备延时 3 分钟关闭蓝牙

L2（单个设备配网-BLE 不断开连接，通过特殊的 property 主动上报配网状态，对于单个设备配网的方式，建议扫描二维码的方式并比对 MAC 地址，以减小误绑定。）：

1.（C1）设备端监测 wifi 连接状态，使用命令上报连接结果（成功 or 错误）
2.（C2）手机端指示连接状态并返回 response
3.（C3）手机端写入 BLE_DEV_CTL 命令给设备,参数等于 2，告知设备 BLE 断开连接后直接关闭蓝牙
4.（C4）设备返回 response,手机端断开蓝牙连接，设备不再进入广播态，关闭蓝牙

注：若设备入网失败，手机 APP 应提示用户，重新入网

8.2 BLE+一键配置流程说明

使用 BLE 进行 WIFI 配网只是提供一种配网方式，不强制单独使用，产品可选择 BLE 配网、一键配置和 softap 的方式的组合。本节对 BLE 配网+一键配置的方式进行说明。

1. 用户触发配网操作后，设备端同时启动 BLE 配网和一键配置流程。
2. 手机端根据其流程做配网操作
3. 若设备端通过 BLE 配网流程获得路由器信息，则关闭一键配置模式，连接路由器。
4. 若设备端通过一键配置流程获得路由器信息，设备关闭一键配置模式，连接路由器，BLE 继续在广播态通过广播包向外告知当前的入网状态。3min 后自动关闭蓝牙。