App与嵌入式通讯协议

从 App 接到蓝牙配对命令 🚀

🟧 设备正品认证流程

步骤	方向	📩 数据内容	✍️ 用途	⚙️ 说明
1️⃣	🔐→📱 Device→App	`ID:<HEX>`	设备发起加密认证	连接后立即推送，HEX 字符串
2️⃣	📱→🔐 App→Device	`ID:<解密HEX>`	回传解密 ID	utf-8→base64 编码
3️⃣	🔐→📱 Device→App	`"VALID"`	正品认证通过	明文
4️⃣	📱→🔐 App→Device	`"validation"`	响应认证通过	utf-8→base64 编码

🟦 PIN 配对及钱包操作命令

步骤	方向	📩 数据内容	✍️ 用途	⚙️ 说明
5️⃣	📱→🔐 App→Device	`"request"`	请求用户在场确认	utf-8→base64
5.1	（Device 设备锁屏状态下 🔐）	嵌入式内部通讯	本地解锁 + 显示验证码	收到 "request" 后立即亮屏提示用户在设备上输入解锁 PIN（仅设备端处理，不经 BLE 传输）；解锁成功后生成并显示一次性验证码（如 6 位数字）；若 60 秒内未完成解锁，则不显示验证码并结束本次操作（App 端可提示超时）
5.2	（Device 设备解锁状态下）	嵌入式内部通讯	显示验证码	收到 "request" 后直接生成一次性验证码（如 6 位数字）并显示在设备屏幕上，等待用户输入到 App
7️⃣	📱→🔐 App→Device	`codeValue:<用户输入>`	回传用户输入的验证码	utf-8→base64
8️⃣	🔐→📱 Device→App	`PIN_OK` / `PIN_FAIL:<reason>`	验证结果	通过进入授权会话；失败按失败策略处理
9️⃣	📱→🔐 App→Device	`address:<chainName>`	获取各链地址	utf-8→base64，建议分批发送、指令间保留短间隔
🔟	🔐→📱 Device→App	`<prefix><address>`	返回链地址	明文，如 `ETH:0x...`
1️⃣1️⃣	📱→🔐 App→Device	`pubkey:<chain>,<hdpath>\n`	获取部分链公钥	utf-8→base64，结尾 `\n`
1️⃣2️⃣	🔐→📱 Device→App	`pubkeyData:<chain>,<pubkey>`	返回链公钥	明文

📌 关键说明

- 📦 所有 App → Device 的数据，均先 utf-8，再 base64 编码发送
- ⏳ 连续命令建议间隔 ≥ 200ms，避免设备处理拥堵
- 🔒 数据明文传递，未做加密，仅用 base64 做数据包装
- 💡 每次写入请使用 `writeCharacteristicWithResponseForService` 保证响应

从 App 接到确认签名对命令 📝🖋️

🟧 BLE 交易签名协议流程

步骤	方向	📩 数据内容	✍️ 用途	⚙️ 说明
1️⃣	📱→🔐 App→Device	`"destinationAddress:<付款地址>,<收款地址>,<手续费>,<链标识>"`	下发交易主要参数（第一步）	例 如：`"destinationAddress:0x123abc...,0x456def...,100000,ethereum"` 所有字段直接拼接，无空格；utf-8 编码后 base64 发送
2️⃣	🔐→📱 Device→App	`"PIN_SIGN_READY"` / `"PIN_SIGN_FAIL"` / `"PIN_SIGN_CANCEL"`	用户密码验证结果	明文，表示用户在设备端 PIN 校验的结果

步骤	方向	📨 数据内容	✍️ 用途	⚙️ 说明
3️⃣	🔐→📱 Device→App	`"Signed_OK"` / `"Signed_REJECT"`	设备确认交易参数	明文：`"Signed_OK"` 表示设备同意处理交易；`"Signed_REJECT"` 表示用户拒绝，App 应立即终止流程
4️⃣	📱→🌐 App→Server	POST `{chain, from, to, txAmount, ...}`	获取 nonce、gasPrice 等预签名参数	App 向后端 API 发 POST 请求，获取当前链的参数
5️⃣	📱→🌐 App→Server	POST encode 接口请求体	获取 presign 数据（hex/json）	返回链对应预签名数据，用于冷钱包签名
6️⃣	📱→🔐 App→Device	`"sign:<链标识>,<BIP44路径>,<presign数据>"`	下发预签名数据	例如：`"sign:ethereum,m/44'/60'/0'/0/0,0xabc..."` utf-8 编码后 base64 发送
7️⃣	🔐→📱 Device→App	`"signResult:<签名数据>"` 或 `"signResult:ERROR"`	返回最终签名结果或错误	明文，如 `"signResult:0xf86b..."` 为签名数据，如失败返回 `"signResult:ERROR"`
8️⃣	📱→🔐 App→Device	`"BCAST_OK,<链标识>,<txHash>"` 或 `"BCAST_FAIL,<链标识>,<错误码>"`	通知设备广播结果	utf-8→base64 编码发送，`"BCAST_OK"` 表示广播成功并附带交易哈希；`"BCAST_FAIL"` 表示广播失败并附带原因码

从 App 接到收藏 NFT 命令 🖼️🔗

步骤	方向	📨 数据内容	✍️ 用途	⚙️ 说明
1️⃣	📱→🔐 App→Device	`"DATA_NFT_TEXT<n>SIZE"`	NFT 名称传输头（标志+长度）	n 为 NFT 名称 utf-8 字节数。utf-8 转 base64 发送
2️⃣	🔐→📱 Device→App	`"GET1"`, `"GET2"`, ...	请求下一个 NFT 名称分包	设备每次请求一包（最多 200 字节）
3️⃣	📱→🔐 App→Device	NFT 名称分包，base64	分包发送 NFT 名称正文	按 200 字节/包，utf-8→base64
4️⃣	🔐→📱 Device→App	`"FINISH"`	NFT 名称传输结束	明文
5️⃣	📱→🔐 App→Device	`"DATA_NFT_IMG<m>SIZE"`	NFT 图片传输头	m 为图片 base64 字节数。utf-8 转 base64 发送
6️⃣	🔐→📱 Device→App	`"GET1"`, `"GET2"`, ...	请求下一个 NFT 图片分包	每次最多 200 字节
7️⃣	📱→🔐 App→Device	NFT 图片分包，base64	分包发送 NFT 图片	200 字节/包，图片原始 base64
8️⃣	🔐→📱 Device→App	`"FINISH"`	NFT 图片传输结束	明文

从 App 接到 OTA 固件升级命令 ⚙️⬆️

步骤	方向	📨 数据内容	✍️ 用途	⚙️ 说明
1️⃣	📱→🔐 App→Device	`"DATA_OTA<文件字节数>SIZE"`	固件头部	例：`"DATA_OTA163840SIZE"`，utf-8→base64 发送，通知设备准备接收
2️⃣	📱→🔐 App→Device	固件内容分包，每包 200 字节，HEX 字符串	固件分包数据	每包 HEX 字符串（200 字节），utf-8→base64，无需等待应答
3️⃣	🔐→📱 Device→App	`"OTA_OK"` / `"OTA_FAIL"`	设备确认接收或异常	升级完成或异常时回包

🗂️ 字段示例与流程说明

- **固件头部示例：**
  `DATA_OTA12032SIZE`
  - 发送内容为 `DATA_OTA12032SIZE` → utf-8 转 base64
- **固件分包示例：**
  - 第 1 包 HEX 为 `aabbcc...`（200 字节，HEX 字符串长度 = 400）

- 发送内容为 `aabbcc...` → utf-8 转 base64

- **分包机制**：
  - 固件全部数据，循环 offset 每 200 字节，分多包下发
  - App 无需等设备 GET 或确认，可 **顺序连续** 写入

```
+--------------------------------+
|       正在升级...              |
|--------------------------------|
| 已发送: 3/48 包 ( 7% )         |
| [████▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒] |
| 请勿断开设备或关闭 App          |
+--------------------------------+
```

从 App 接到“确认地址”命令 🏦🔎

步骤	方向	📩 数据内容	✍️ 用途	⚙️ 说明
1️⃣	📱→🔐 App→Device	`"verify:<chainName>"`	显示地址请求命令	如：`"verify:bitcoin"`，utf-8→base64
2️⃣	🔐→📱 Device→App	`"Address_OK"`	地址显示完成反馈	明文
3️⃣	🔐→📱 Device→App	其他提示/错误码	异常/扩展命令	如遇异常可扩展 `"Address_FAIL"` 等命令