基于卫星节点的加密货币钱包安全解决方案的需求分析与设计

Release Table

Version	Date	Changes
V0.0	Oct 06, 2018	初始版本;
V1.0	Oct 22, 2018	正式版本;

1 需求分析

1.1 使用场景

随着人们对加密货币认知水平的提高，现阶段主流加密货币的单价不断抬升，加密货币交易的安全性越发引起人们的重视。尤其如交易所清算或用户进行大额交易时，由于加密货币本身运转机制不可篡改的特性，一旦出现漏洞或细小的操作失误都会引起巨额损失。因此，目前极有必要引入一种机制能够对用户钱包的关键交易进行二次确认，作用户钱包的“保险”，巩固用户钱包的安全。多重签名交易为我们提供了可能。

用户通过将自己的钱包地址和可信的第三方钱包地址合成得到一个多重签名钱包地址，并在多重签名钱包中存放自己的加密货币。当用户需要向他人转账时，用户自行发起多重签名交易。当用户本身和可信的第三方都对这笔交易签名确认后，用户才能将交易广播到公网完成转账。引入这样的流程后，用户钱包的安全性将显著提升。

1.2 功能需求

通过对用户使用场景的分析和细化，我们可以梳理出用户对方案的基本功能需求。

1.2.1 账户注册功能

用户可以创建账户，用于获取第三方钱包地址，申请第三方签名。

1.2.2 钱包地址申请功能

用户可以通过申请得到一个可信的第三方钱包地址。

1.2.3 多重签名钱包地址合成功能

可以将用户的钱包地址和可行的第三方钱包地址合成多重签名钱包地址。

1.2.4 多重签名钱包地址充值功能

用户可以向自己的多重签名钱包地址转入一定数额的加密货币。

1.2.5 多重签名交易发起功能

用户可以发起交易，用多重签名钱包中的加密货币向他人转账。

1.2.6 签名申请功能

当用户用多重签名钱包向他人转账时，用户可以申请第三方签名确认，双方都确认该笔交易后，交易才能达成。

1.2.7 交易广播功能

当用户与可信的第三方都签名确认交易后，用户可以通过交易广播功能将签名后的交易广播到公网上完成交易。

1.3 性能需求

用户容量：可以为 100 个账户提供服务；
交易频次：每天至少可以确认一笔交易。

1.4 安全需求

账户独立性要求

每个账户拥有自己独立的多重签名钱包地址，不与他人共用。

每个账户拥有自己独立的第三方，不与他人共用，且第三方拥有自己独立私钥。

第三方安全性要求

第三方节点有较高的安全性，不易被攻破。

操作安全需求

服务全程不能存在明码操作私钥的行为，如复制、粘贴等。

2 方案设计

通过对整体需求的整理与分析，可以看出：作为一套针对交易安全的解决方案，可信的第三方本身的安全性越高，方案整体的安全性就越高。因此我们提出了一套以在轨签名节点为核心，以客户端和网站服务相结合作为用户交互形式的加密货币交易安全解决方案。

以在轨节点作为第三方签名节点。因为其本身工作环境、通信体系、软硬件体系架构都十分特殊，而且具有政治象征意义，使得其本身有很高的安全性。

采用钱包客户端与网站服务相结合的方式，一方面可以提高服务本身对用户的友好程度，另一方面，可以使用户在自己的电脑上完成签名和交易广播的操作，避免操作明码私钥，提高安全性。

2.1 概要设计

2.1.1 架构设计

方案由客户端模块、网站服务模块和签名节点模块三部分组成。

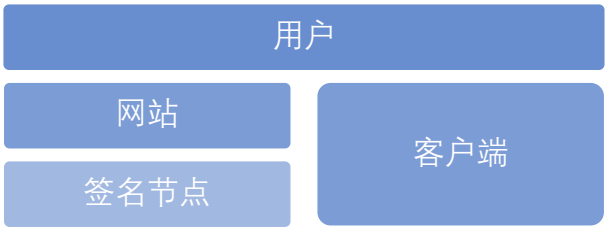

Figure-1 方案架构设计图

客户端与网站为用户操作界面，签名节点不直接与用户交互。当用户申请签名时，向网站提交签名申请，再由网站提交给签名节点。

签名节点本身是一个在轨节点，作为可信的第三方。当用户发起多签交易时，签名节点会对交易进行二次签名确认保障安全。

2.1.2 模块功能设计

按照框架设计，将功能需求拆分给以上三个模块。

2.1.2.1 客户端模块

1）多重签名钱包地址合成功能

将用户提供的钱包地址和网站申请的第三方钱包地址合成多重签名钱包地址。

2）多重签名钱包地址充值功能

用户可以向生成的多重签名钱包地址转入加密货币。

3）多重签名交易发起功能

用户可以发起多重签名交易，将多重签名钱包地址中的加密货币转账给他人。

4）交易广播功能

用户可以将收到完整的签名交易广播到公网。

2.1.2.2 网站服务模块

1）账户注册功能

处理用户的注册申请，并在后台存储用户信息。

2）钱包地址申请功能

当收到注册用户的第三方钱包地址申请后，确认其是否缴纳 token，若缴纳成功则向用户返回一个签名节点的一个钱包地址，若失败，则返回申请失败。

3）签名申请功能

当注册用户发起多重签名交易后，将已经完成自己已经签名确认后的交易信息提交给网站，网站打包发送给签名服务模块签名，若签名成功，则将完成签名的交易发送给用户，若签名失败，则向用户返回签名失败信息。

2.1.2.3 签名节点模块

签名节点模块在收到网站提交的完成了一次签名的交易后，对交易进行二次签名。若签名成功，返回签名完成后的交易，若签名失败，返回错误信息。

2.1.3 交互设计

2.1.3.1 用户操作流程设计

1）账户注册

用户向网站提出注册申请，网站后台将存储用户信息，并向用户返回注册成功信息。

2）钱包地址申请

首先，用户向网站提供的指定钱包地址中转入指定数额的 token。转账完成后，用户向网站提交转账信息和钱包地址申请。网站将确认转账是否成功，若成功，则向用户返回钱包地址；若失败，则向用户返回错误信息。

3）多签钱包地址合成

用户向客户端输入自己的钱包地址和在网站上申请到的钱包地址，客户端合成多重签名钱包地址返回给用户。

4）多签钱包地址充值

用户向客户端转账金额，向多重签名钱包地址转入 QTUM。

5）发起多签交易

首先，用户向客户端提交交易信息并发起多签交易。客户端生成多签交易，并对多签交易签名，然后将完成一签的交易返回给用户。

6）签名申请

用户向网站提交一签交易和签名申请，网站向用户返回签名结果。

7）交易广播

用户收到签名结果后，可以选择将完成签名的交易广播到公网。用户将已经完成签名的交易输入客户端，客户端将完成签名的交易广播到公网。

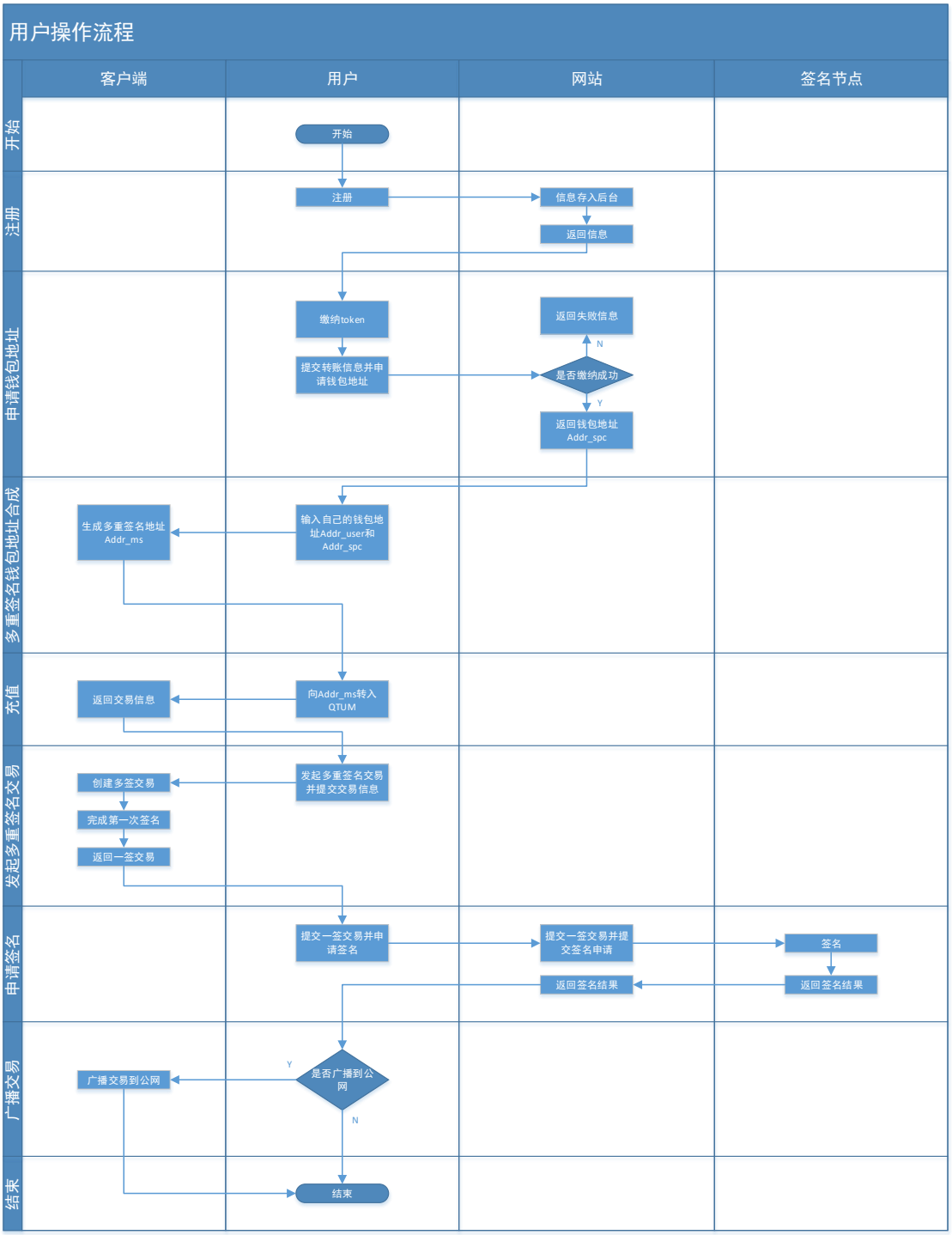

Figure-2 用户操作流程图

2.1.3.2 数据流设计

通过细化功能需求，可以将每个功能抽象为用户向模块的数据输入和模块向用户的数据输出两部分。

为了满足用户对安全的需求，在方案的数据流设计中，尽量避免了对私钥的明码操作。

对用户本身钱包地址的私钥操作也全部在用户自己的设备上完成，避免产生不必要的安全漏洞。

Table-1 用户与模块交互数据表

模块	功能	用户输入	模块输出
网站	账户注册	账户邮箱	注册成功信息
		账户密码	
	钱包地址申请	缴费交易 ID	钱包地址或申请失败信息
	签名申请	完成一签的未完成交易的 json 格式数据	签名结果数据
客户端	多签钱包地址合成	用户钱包地址	多签钱包地址
		申请的钱包地址	
	多签钱包地址充值	多签钱包地址	转账交易 ID
		转账金额	
	多签交易发起功能	充值交易 ID	完成一次签名的未完成交易 JSON 格式数据
		收款钱包地址	
		转账金额	
	交易广播功能	完成全部签名的交易的 hex 数据	广播成功信息或广播失败信息

Table-2 网站与签名节点交互数据表

模块	功能	网站输入	签名节点输出
签名节点	签名申请	完成一签的未完成交易数据	若签名成功：返回签名完成的交易 若签名失败：返回签名失败信息

2.2 详细设计

模块结构和上层应用设计完成后，根据每个模块的功能和数据流进行模块底层实现的方案设计以及模块间的通信链路的方案设计。

2.2.1 星地通信链路设计

2.2.1.1 链路组成

方案中的通信链路由**网站-地面站**和**地面站-签名节点**两部分组成。

2.2.1.2 通信流程设计

首先，网站会以电子邮件的形式将用户申请的待签名交易按固定格式编码后发送给地面站。地面站收到邮件后，会在签名节点过境时通过 UHF/VHF 信号发送给签名节点。签名节点收到数据后按照固定格式解码得到待签交易，然后对待签交易签名确认。

签名节点会将签名结果数据按照固定格式编码，再通过 UHF/VHF 信号发送给地面站。地面站在将收到的数据通过电子邮件发送回网站。网站收到数据后按照固定格式解码得到签名结果数据。

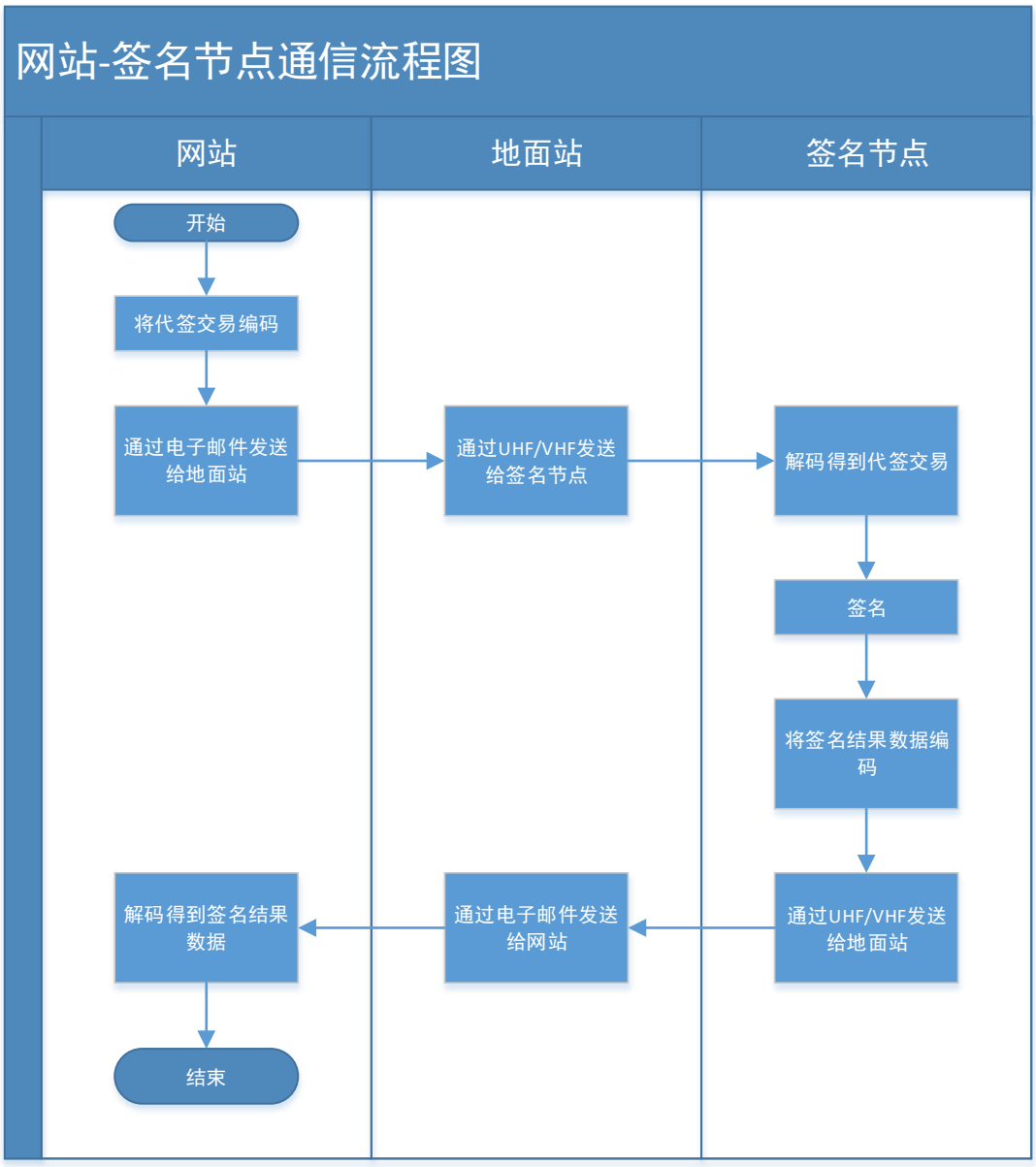

Figure-3 网站与签名节点通信流程图

2.2.1.3 通信协议设计

为了保证数据在收发过程中正确性，网站和签名节点在发送数据前会将待签交易数据和签名结果数据的各个字段按照通信协议中规定的格式进行编码。当收到数据后，也会按照通信协议规定的格式解码。

Table-3 待签交易编码格式（网站发送给卫星节点）

名称	类型	长度	说明
数据头部分			
数据头	十六进制	1byte	多重签名类型数据头，0xa1；
数据总长	十六进制	2byte	包括各字段头、各字段长度、各字段数据总体长度；
交易编码部分			
交易编码数据头	十六进制	1byte	十六进制交易编码数据头，0xba；
交易编码数据长度	十六进制	2byte	最高位表示十六进制数据的最后一个字节是否是一半有效，这个参数在把数据从十六进制转换成字符串的时候需要使用
交易编码数据	十六进制	不定长	
Txid 数组部分			
Txid 数组数据头	十六进制	1byte	Txid 数组数据头，0xca；
Txid 数组数据长度	十六进制	2byte	
Txid 数组成员个数	十六进制	1byte	
第一个 Txid 数组成员数据			
第一个成员 txid 字段长度	十六进制	1byte	最高位表示十六进制数据的最后一个字节是否是一半有效，这个参数在把数据从十六进制转换成字符串的时候需要使用
第一个成员 txid 字段数据	十六进制	不定长	
第一个成员 vout 数据	十六进制	1byte	
第一个成员 scriptpubkey 字段长度	十六进制	2byte	最高位表示十六进制数据的最后一个字节是否是一半有效，这个参数在把数据从十六进制转换成字符串的时候需要使用
第一个成员 scriptpub	十六进制	不定长	

key 字段数据			
第一个成员 redmeescript 字段长度	十六进制	2byte	最高位表示十六进制数据的最后一个字节是否是一半有效，这个参数在把数据从十六进制转换成字符串的时候需要使用
第一个成员 redmeescript 字段数据	十六进制	不定长	
第二个 Txid 数组成员数据			
…	…	…	…
私钥索引字段	十六进制	2byte	
多重签名类型数据尾	十六进制	1byte	多重签名类型数据尾，0x1a；

Table-4 签名结果数据编码格式（签名节点到网站）

名称	类型	长度	说明
签名结果数据头	十六进制	1byte	签名结果数据头，0xc1；
签名结果数据长度	十六进制	2byte	最高位表示十六进制数据的最后一个字节是否是一半有效，这个参数在把数据从十六进制转换成字符串的时候需要使用
签名结果数据	十六进制	不定长	
签名结果数据尾	十六进制	1byte	签名结果数据尾，0x1c；

2.2.2 客户端

Togo

2.2.3 网站服务

Togo

2.2.4 签名节点

2.2.4.1 模块组成

签名节点由搭载平台和签名载荷两部分组成。

1）搭载平台

搭载平台顾名思义在签名节点中起到平台作用。它可以控制签名载荷的开关机，并且为签名载荷提供能源，还可以通过 UHF/VHF 收发机实现签名载荷与地面站的双向通信。

2）签名载荷

签名载荷是签名节点的核心。搭载平台收到地面站上传的待签名交易后将其发送给签名载荷，载荷签名完成后，再将签名结果返回给搭载平台，再由搭载平台发送地面站。

2.2.4.2 搭载平台与签名载荷间通信链路设计

1）硬件设计

采用 CAN 总线通信，波特率 500kbps。

2）签名载荷控制指令设计

根据业务需求，方案设计了以下五条指令，用来控制签名载荷。当搭载平台向签名载荷发送不同的指令时，签名载荷执行不同的操作并向搭载平台返回相应的结果。指令间通过消息 ID 区分。

待签交易上注指令

搭载平台将接收到的待签交易数据分多包发送给签名载荷。签名载荷收到后，将多包数据重组，解码，并对交易进行签名确认，将签名结果编码存入缓冲区。

签名结果请求指令

搭载平台向签名载荷请求签名结果数据。载荷收到后将签名结果数据发送给搭载平台。

遥测数据请求指令

搭载平台向签名载荷请求遥测数据。载荷收到后将遥测信息发送给搭载平台。

时间同步指令

搭载平台向签名载荷发送含有时间信息的数据包。载荷收到后自行按照该信息同步自己的系统时间。

总线复位指令

搭载平台向签名载荷发送总线复位指令。载荷收到后将自己的 CAN 设备重启，并清空自己的数据缓冲区。

3）通信协议设计

确定每条指令签名载荷的输入数据和输出数据后，我们为每条指令的输入和输出设计指定的数据格式，便于收发双方对数据进行拆分、重组。

待签交易上注指令

CAN 总线上数据传输是以“帧”为单位的。字节数较多的数据通过“首帧+中间帧*X+尾帧”的形式完成一次数据的传输。当前协议下，搭载平台向载荷传输待签交易数据时，CAN 帧首帧可以传输 5 个字节，中间帧可以传输 7 个字节，尾帧可以传输 3 个字节。

每次待签交易上注固定发送数据长度为 500 个字节。按照通信协议设计，分 10 次发送，每次发 50 个字节。因此每发送一次需要 1 个首帧、6 个中间帧和 1 个尾帧（5*1+7*6+3*1 = 50byte）。

Table-5 待签交易上注指令格式-首帧帧格式表

域名	说明
消息 ID	0x2D
固定字段	0x01
	0x08
	0x5A
有效负载数据	DATA0
	DATA1
	DATA2
	DATA3
	DATA4

Table-6 待签交易上注指令格式-中间帧帧格式表

域名	说明
消息 ID	0x2E
有效负载数据	DATA0
	DATA1
	DATA2
	DATA3
	DATA4
	DATA5
	DATA6

Table-7 待签交易上注指令格式-尾帧帧格式表

域名	说明
消息 ID	0x2F
固定字段	0x08
有效负载数据	DATA0
	DATA1
	DATA2

签名结果请求指令

签名结果请求指令的数据格式分两部分组成。一方面，是搭载平台向签名载荷的指令数据格式。另一方面，签名载荷收到指令后，向搭载平台传输签名结果数据时的数据格式。

搭载平台向签名载荷传输指令时，为单帧发送。

Table-8 签名结果请求指令格式-帧格式表

域名	说明
消息 ID	0x30
有效负载数据	0x1A

签名载荷收到指令后，以多帧的形式返回签名结果数据。当前协议下，签名载荷向搭载平台传输签名结果数据时，CAN 帧首帧可以传输 5 个字节，中间帧可以传输 7 个字节，尾帧可以传输 4 个字节。

每次签名结果数据下传固定发送数据长度为 792 个字节。按照通信协议设计，分 4 次发送，每次发 198 个字节。因此每发送一次需要 1 个首帧、27 个中间帧和 1 个尾帧（5*1+7*27+4*1 = 198byte）。

Table-9 签名结果数据下传数据格式-首帧帧格式表

域名	说明
消息 ID	0x431
固定字段	0x01
	0x1D
	0x1A
有效负载数据	DATA0
	DATA1
	DATA2
	DATA3
	DATA4

Table-10 签名结果数据下传数据格式-中间帧帧格式表

域名	说明
消息 ID	0x432
有效负载数据	DATA0
	DATA1
	DATA2
	DATA3
	DATA4
	DATA5
	DATA6

Table-11 签名结果数据下传数据格式-尾帧帧格式表

域名	说明
消息 ID	0x433

固定字段	0x1D
有效负载数据	DATA0
	DATA1
	DATA2
	DATA3

遥测数据请求指令

遥测数据请求指令的数据格式分两部分组成。一方面，是搭载平台向签名载荷的指令数据格式。另一方面，签名载荷收到指令后，向搭载平台传输遥测数据时的数据格式。

搭载平台向签名载荷传输指令时，为单帧发送。

Table-12 签名结果请求指令格式-帧格式表

域名	说明
消息 ID	0x30
有效负载数据	0x1B

签名载荷收到指令后，以多帧的形式返回遥测数据。当前协议下，签名载荷向搭载平台传输遥测数据时，CAN 帧首帧可以传输 5 个字节，中间帧可以传输 7 个字节，尾帧可以传输 3 个字节。

每次签名结果数据下传固定发送数据长度为 50 个字节。按照通信协议设计，共发送 1 次，每次发 50 个字节。因此每发送一次需要 1 个首帧、6 个中间帧和 1 个尾帧（5*1+7*6+3*1 = 50byte）。

Table-13 遥测数据下传数据格式-首帧帧格式表

域名	说明
消息 ID	0x431
固定字段	0x01
	0x08
	0x1B
有效负载数据	DATA0
	DATA1
	DATA2
	DATA3
	DATA4

Table-14 遥测数据下传数据格式-中间帧帧格式表

域名	说明
消息 ID	0x432
有效负载数据	DATA0
	DATA1
	DATA2
	DATA3
	DATA4

	DATA5
	DATA6

Table-15 遥测数据下传数据格式-尾帧帧格式表

域名	说明
消息 ID	0x433
固定字段	0x08
有效负载数据	DATA0
	DATA1
	DATA2

时间同步指令

搭载平台向签名载荷传输时间同步指令时，为单帧发送。

时间同步指令的有效负载数据有 6 个字节。W0~W5 为综合电子计算机的系统时间（自北京时 2018 年 1 月 1 日 0 时起毫秒累加值）。W0~W3 为累计秒数，W4~W5 为秒内毫秒计数，统一按照大端法排序。

Table-16 时间同步指令格式-帧格式表

域名	说明
消息 ID	0x18
有效负载数据	W0
	W1
	W2
	W3
	W4
	W5

总线复位指令

搭载平台向签名载荷传输总线复位指令时，为单帧发送形式。

总线复位指令的有效负载数据重复的固定内容，共 8 个字节。

Table-17 总线复位指令格式-帧格式表

域名	说明
消息 ID	0x24
有效负载数据	0xFA
	0xCE
	0xFA
	0xCE
	0xFA
	0xCE
	0xFA
	0xCE

2.2.4.3 签名载荷详细设计

1）硬件设计

签名载荷硬件采用 XC7Z030 SoC。

2）软件设计

签名业务逻辑设计

第一步：节点过境前 5min 签名载荷上电开机；

第二步：执行启动脚本，进行初始化工作（清空目录下除存储数据外的所有软件运行数据），然后启动 Qtum 客户端，启动星上软件；

第三步：监听 CAN 总线，等待 OBC 发来的消息；

第四步：收到消息，判断消息类型，若为待签交易数据（共 10 包，每包 50 个字节，共 500 个字节，不足 500 字节数据后补零。），则存入缓冲区，收包完成后开始签名计算。

第五步：签名完成，等待 OBC 签名结果数据请求；

第六步：收到签名结果数据请求，将签名结果数据打包发送给 OBC，共 4 包，每包 198 个字节，共 792 个字节，不足 792 个字节数据后补零；

第七步：等待关机；

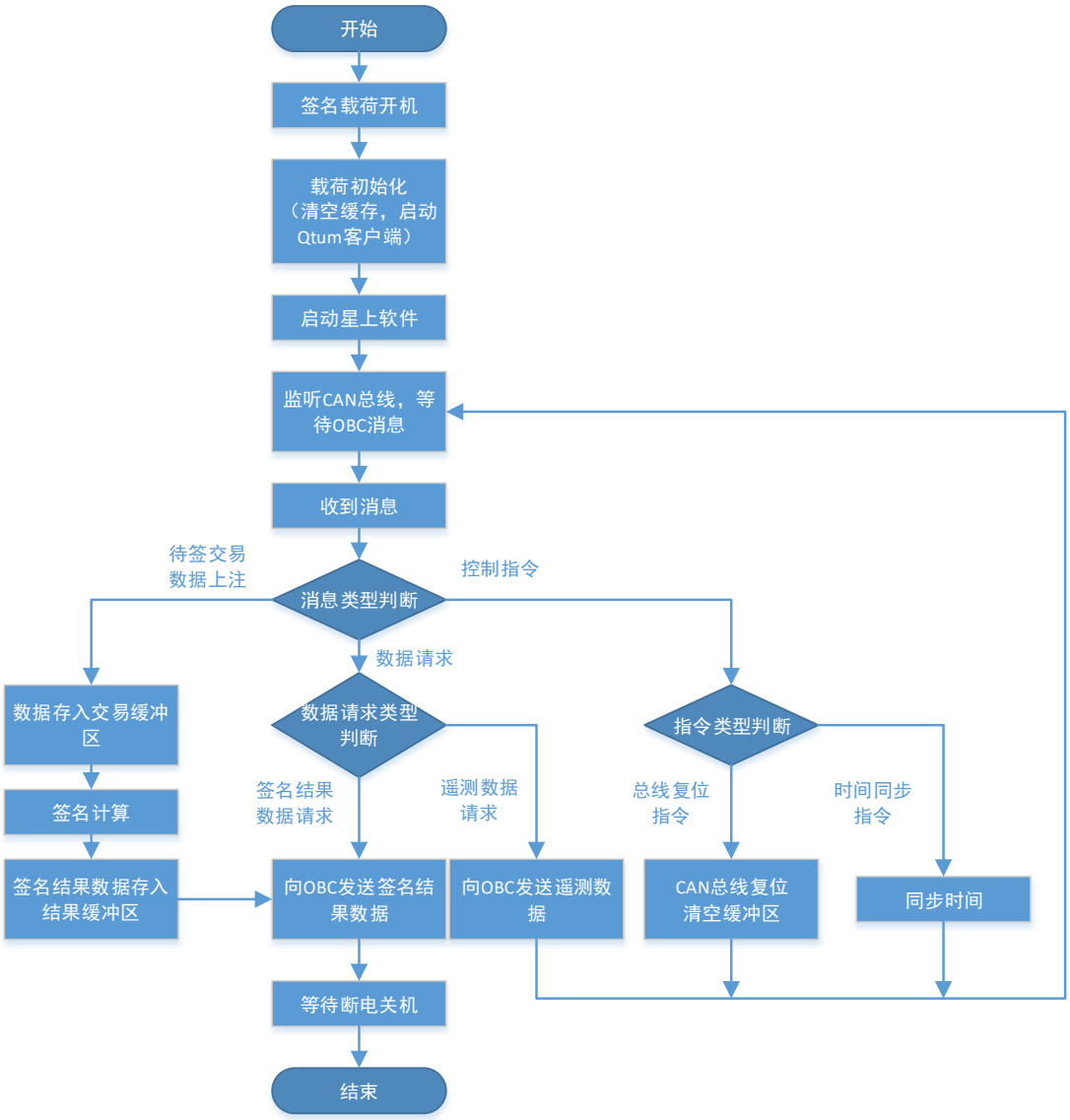

Figure-4 逻辑流程图

主要接口实现设计

INT spcRestartNum (VOID)

函数名称: spcRestartNum

功能描述: 处理设备重启次数

输　入　: NONE

输　出　: ERROR_CODE

实现流程:

1.打开 fram 设备文件

2.读 fram 中的重启次数到全局缓冲区中

3.将缓冲区中的数值加一

4.再将缓冲区中的数值写入 fram 中

5.关闭 fram 设备文件

INT spcSlaveProtocolRecvStart (char *pcdevname)

函数名称: spcSlaveProtocolRecvStart
功能描述: CAN 测试
输　入　: CAN 设备路径
输　出　: ERROR_CODE
实现流程:

1.创建 spcPthreadRecv 线程，线程参数为 CAN0 设备路径
2.线程阻塞等待子线程返回

```
static VOID　*spcPthreadRecv(VOID　*pvArg)
```

函数名称: spcPthreadRecv
功能描述: CAN 驱动测试 pthread 接收线程
输　入　: pvArg　　线程参数（CAN 设备路径）
输　出　: 线程返回值
实现流程:

1.根据参数打开 CAN0 设备文件
2.设置 CAN0 设备波特率
3.进入 while(1)循环:
　开始读 CAN 设备
　若没有数据则等待一段时间，再次读设备
　若有数据，则进入数据处理程序

```
void　spcSlaveProcessRcvData(INT　iFd, CAN_FRAME　canframe)
```

函数名称: spcSlaveProcessRcvData
功能描述: 从机处理从主机接收到的数据
输　入　: canframe：一个 can 帧数据
输　出　: None
实现流程:

1.判断收到的 CAN 数据的 ID 是否匹配
2.判断收到的数据帧类型
　1）签名结果数据请求
　2）遥测数据请求
　3）CAN 复位命令（本函数内执行 io 命令）
　4）时间同步（本函数调用系统调用）
　5）待签交易数据上注

```
static VOID　*spcMasterRequestTelemetryData(INT　iFd)
```

函数名称: spcMasterRequestTelemetryData
功能描述: 从机处理主机发送的签名结果数据请求指令
输　入　: pvArg：创建线程时传入的参数
输　出　: None
实现流程:

1.将签名结果数据按照通信协议打包
2.向 OBC 发送签名结果数据

static VOID *spcMasterRequestTestData(INT iFd)

函数名称: spcMasterRequestTestData
功能描述: 从机处理主机发送的测试数据请求命令
输 入 : pvArg: 创建线程时传入的参数
输 出 : None
实现流程:
1.将问候语传入缓冲区
2.获取启动时间
3.获取重启次数
4.进入数据打包函数

VOID spcReturnTestTelemetryData (INT iFd, UCHAR *pucBuf, INT iDataLen)

函数名称: spcReturnTestTelemetryData
功能描述: 从机处理主机发送的请求测试数据命令
输 入 : pvArg: 创建线程时传入的参数
输 出 : None
实现过程:
1.将遥测数据按照协议打包
2.发送打包好的数据

INT spcRcvDataToCmdString (CHAR *ucDest, INT iLen, UCHAR *ucSrc)

函数名称: spcRcvDataToCmdString
功能描述: 把接收到的压缩数据转换成签名时执行的字符串命令
输 入 : ucDest: 存储结果的缓冲区
: ucSrc : 接收的压缩数据
: iLen : ucDest 缓冲区长度
输 出 : ERROR_CODE
实现流程:
1.解析 JSON 格式数据
2.在待签交易前拼接签名命令字
3.存入命令缓冲区

INT spcProcessingJsonData (UCHAR *ucOutData, size_t sOutLen, CHAR *cJsonStr, size_t stLen)

函数名称: spcProcessingJsonData
功能描述: 处理 json 数据
输 入 : cJsonStr : json 字符串数据
: sOutLen : ucOutData 缓冲区长度
: sLen : cJsonStr 字符串长度
输 出 : ucOutData: 压缩后的 hex 字段数据
实现流程:
按照格式解析 JSON 数据

接口调用关系

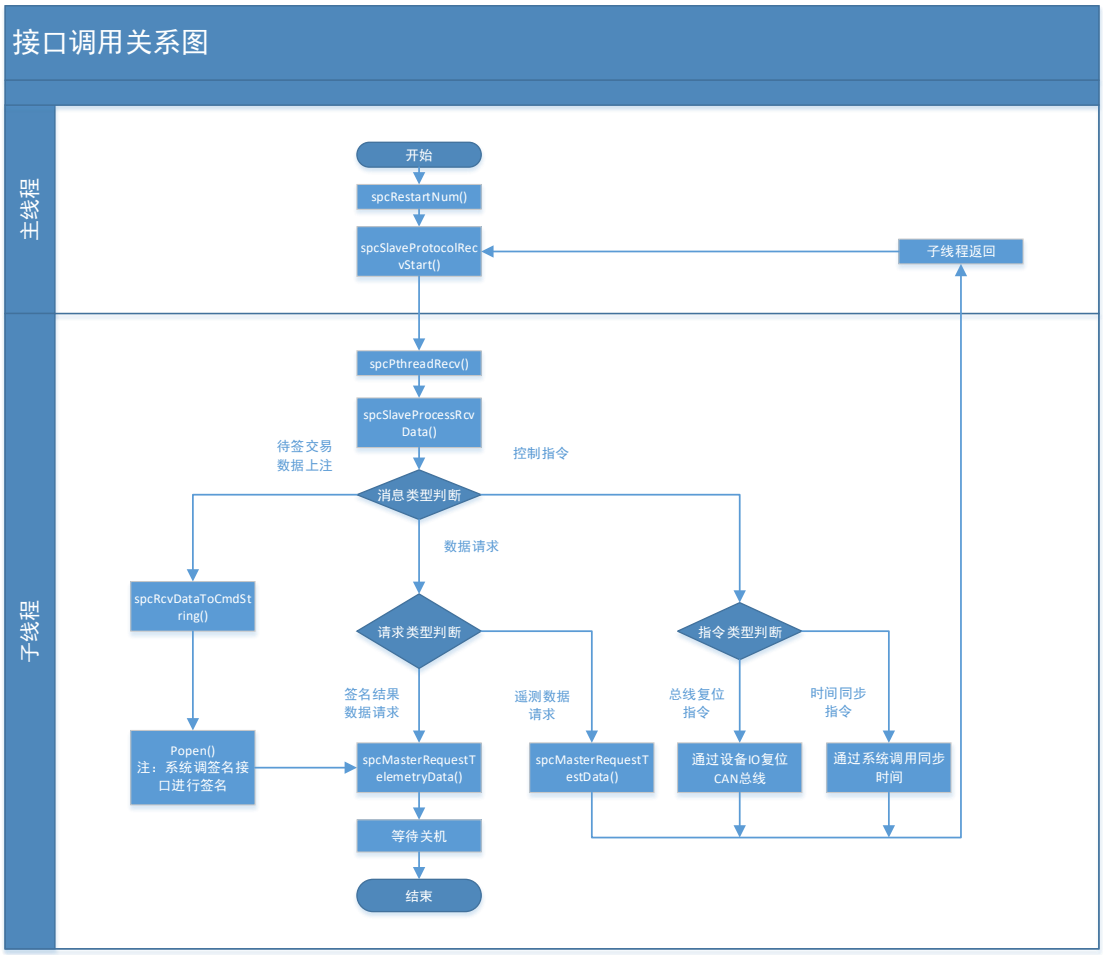

Figure-5 接口调用关系图