音频通讯SDK说明/改进文档

1. 简介

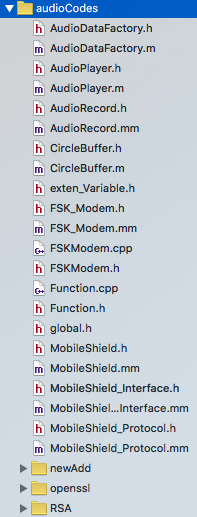
音频通讯是指使用音频作为载体进行数据交互的一种通讯方式。数据根据算法进行编码生成音频数据，通过手机进行放音，同时开启录音。录音的内容根据算法解码出程序可读数据。

公司现在的产品为二代盾，即可在电脑上使用，也可在手机上使用的U盾。在电脑上使用时，使用USB进行数据交互。手机上使用时，使用音频进行数据交互。

音频通讯的优点：成本低，U盾无需额外的设备与手机进行数据链接通道。

缺点：实现复杂，稳定性不搞，通讯速率慢。

SDK目录截图：



SDK的接口是在MobileShield\_Interface.h文件中，接口调用使用同步返回数据，就是调用SDK功能接口的时候，返回数据是通过接口返回值进行返回，优点使用方便，缺点程序会在此产生阻塞。

1. 功能分类

现有的SDK实现的公司产品当前的功能需求，但是存在代码结构混乱，没有进行框架设计，无明显层次结构，耦合度高。

具体有以下功能：

1. 数据转换 编解码

实现文件：MobileShield\_Interface，FSKModem

1. 请求数据封装

实现文件：MobileShield\_Protocol

1. 录放音

实现文件：AudioPlayer，AudioRecord

1. 内存缓存

实现文件：CircleBuffer

1. 功能说明
2. 数据转换 编解码

请求接口传入数据都是使用NS对象的数据，首先需要将NS对象的数据转换为chat\*，再将数据转换为16进制int放入一个chat数组中，对chat数组进行编码，生成音频文件。

收到录音数据，首先把录音数据放入内存缓存中，每次从内存缓存中取定长数据存入chat数组对象中，对chat数据进行编码。

1. 请求数据封装

根据U盾通讯协议，将请求数据封装成对应功能的协议数据，包括协议头数据和协议体数据

1. 录放音

使用AudioToolBox.Framework中AudioQueue类方法，需要定义录放音方式，格式，采样率，缓存等参数。

放音：创建缓冲池，一共有三个缓冲池，将数据加入到缓冲池中，开始播放缓冲池中的数据。

录音：定义缓冲池大小，一共有三个缓冲池，录音回调方法中获取录音数据，加入到内存缓存中。

1. 内存缓存

内存缓存只存放录音数据，通讯结束后，是否缓存。因为是多线程调用，内存缓存存在线程安全的隐患，加入的同步锁机制。

1. 接口描述
2. int GenRandom();

获取随机数，从U盾中获取一个int类型的随机数

1. bool auth\_keyDevice();

设备认证，检测设备的合法性

1. int open\_Application();

打开应用，非对称算法U盾必须使用。

1. NSArray \*enum\_container();

检索容器，获取已打开应用中的容器列表。

1. NSDictionary \* open\_contain(NSString \*containerName);

根据容器名打开容器，返回容器相关信息

parme containerName 容器名

1. BOOL VerifyPinCode(NSString \*pin);

校验PIN码，YES返回成功

parme pin 设备PIN码

1. NSData \* HASHInit(NSString \*inStr);

使用哈希算法进行加密

parme inStr 待加密内容

返回哈希加密后的数据

1. NSData \* RSASignData(NSString \*pin, NSData \*signData);

使用RSA算法进行签名

parme pin 设备PIN码

signData 待签名数据

返回签名后的数据

1. NSDictionary \* changePinCode(NSString \*oldPin, NSString \*newPin);

修改设备PIN码

parme oldPin 当前设备PIN码

newPin 新的设备PIN码

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. NSDictionary \* unlockPinCode(NSString \*oldPin, NSString \*AdminPin, NSString \*newUserPin);

解锁设备PIN码

parme oldPin 当前设备PIN码

AdminPin 管理员PIN码

newUserPin 新的设备PIN码

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. BOOL audioRouteIsPlugedIn();

判断耳机孔是否有设备，YES有设备

1. NSString \* InterceptionFormatPayeeName(NSString \*name);

截取并格式化收款人姓名，当姓名byte字节长度大于14位，进行截取

parme name 收款人姓名

返回格式化后的姓名

1. NSDictionary \* token\_QueueTokenEX();

获取设备序列号

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. NSDictionary \* token\_ActiveTokenPlug(NSString \*tokenSN, NSString \*activeCode);

激活设备

parme tokenSN 当前设备序列号

activeCode 激活码

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. NSDictionary \* token\_UnlockRandomNo(NSString \*tokenSN);

获取解锁随机数

parme tokenSN 当前设备序列号

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. NSDictionary \* token\_UnlockPin(NSString \*tokenSN, NSString \*unlockCode);

解锁设备

parme tokenSN 当前设备序列号

unlockCode 解锁码

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. NSDictionary \* token\_UpdatePin(NSString \*tokenSN, NSString \*oldPin, NSString \*newPin);

更新设备PIN码

parme tokenSN 当前设备序列号

oldPin 设备当前PIN码

newPin 新的PIN码

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. NSDictionary \* token\_GetTokenCodeSafety(NSString \*tokenSN, int audioPortPos,NSString \*pin, NSString \*utctime, NSString \*verify, NSString \*ccountNo,NSString \* money, NSString \*name, int currency);

根据数据获取动态密码

parme tokenSN 当前设备序列号

audioPortPos 音频端口 默认为1

Pin 设备PIN码

Utctime utc时间

Verify 校验码

accountNo 收款人账号

money 金额

name 收款人姓名

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. NSDictionary \* token\_GetTokenCodeSafety\_key(NSString \*tokenSN, int audioPortPos, NSString \*pin, NSString \*utctime, NSString \*verify, NSString \*ccountNo, NSString \* money, NSString \*name, int currency);

用户转账时，生成动态密码，获取时需要通过设备按钮确认

parme tokenSN 当前设备序列号

audioPortPos 音频端口 默认为1

Pin 设备PIN码

Utctime utc时间

Verify 校验码

accountNo 收款人账号

money 金额

name 收款人姓名

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. NSDictionary \* token\_ScanCode(NSString \*tokenSN, int audioPortPos, NSString \*pin, NSString \*utctime, NSString \*verify, NSString \*ccountNo, NSString \* money, NSString \*name, int currency);

通过扫码获取动态密码

parme tokenSN 当前设备序列号

audioPortPos 音频端口 默认为1

Pin 设备PIN码

Utctime utc时间

Verify 校验码

accountNo 收款人账号

money 金额

name 收款人姓名

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. NSDictionary \* startRecordButtonAction();

等待设备按钮事件

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. NSDictionary \* startRecordButtonActionWithType(ApiType type);

根据类型等待设备按钮事件

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. void stopAudioSession();

停止录音

1. BOOL token\_CancelTrans();

取消交易

1. NSDictionary \* token\_new\_GetTokenCodeSafety(NSString \*tokenSN, int audioPortPos, NSString \*pin, NSString \*utctime, NSString \*verify, NSString \*ccountNo, NSString \* money, NSString \*name, int currency);

获取动态密码

parme tokenSN 当前设备序列号

audioPortPos 音频端口 默认为1

Pin 设备PIN码

Utctime utc时间

Verify 校验码

accountNo 收款人账号

money 金额

name 收款人姓名

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. NSDictionary \* token\_new\_GetTokenCodeSafety\_key(NSString \*tokenSN, int audioPortPos, NSString \*pin, NSString \*utctime, NSString \*verify, NSString \*ccountNo, NSString \* money, NSString \*name, int currency);

用户转账时，生成动态密码，获取时需要通过设备按钮确认

parme tokenSN 当前设备序列号

audioPortPos 音频端口 默认为1

Pin 设备PIN码

Utctime utc时间

Verify 校验码

accountNo 收款人账号

money 金额

name 收款人姓名

返回 设备操作之后解析数据，加上相应提示信息的自定义包。

1. int MyOpenFile();

读取设备证书文件，返回1 成功

1. int check\_cert();

检测是否有新证书

1. NSArray \*readLocalFile();

获取本地已存证书文件列表

1. NSInteger newRSASignData(NSString \*pin, NSData \*signData);

使用RSA算法进行签名

parme pin 设备PIN码

signData 待签名数据

返回签名是否成功

1. NSData \* waitRSASignData(NSInteger length);

等待签名数据

1. NSDictionary \* GetPinInfo(int pType);

获取PIN信息

1. NSInteger newECCSignData(NSString \*pin, NSData \*signData, NSInteger p1);

使用ECC算法进行签名

parme pin 设备PIN码

signData 待签名数据

返回签名是否成功

1. NSArray \* newRSASignDataByRandom(NSString \*pin, NSData \*signData, NSData \*random);

根据随机数进行ECC算法签名

parme pin 设备PIN码

signData 待签名数据

random 随机数

返回签名后的数据

1. NSArray \* newECCSignDataByRandom(NSString \*pin, NSData \*signData, NSInteger p1, NSData \*random);

根据随机数进行ECC算法签名

parme pin 设备PIN码

signData 待签名数据

random 随机数

返回签名后的数据

1. NSData \* GetRandom();

获取随机数

1. NSString \*getICCardNum();

获取IC卡卡号，需要刷卡

1. BOOL SufficientMoney(NSString \*money);

充值

parme money 充值金额

返回充值是否成功

1. SDK流程

改进说明

1. 对外接口窄化

当前接口比较复杂并且有很多接口出现重复，优化方案：根据类型划分，相识类型接口窄化为一个接口，传入数据统一使用参数类。不同接口使用不同的参数类。

1. 功能模块化 分层

优化方案：

1. 音频通讯透传层

说明：当前音频通讯其实已经分层，但是需要进行代码整理和优化

还有就是一个对设备兼容性的可配置管理。

1. 功能协议层

说明：功能协议就是现在的组包操作，但是当前所有的组包都放在一个文件中。优化为通过一个adapter类根据参数类操作请求类型进行组包，每个包都存在独立的文件中。

1. 临时数据层

说明：存放录音数据和解析成功后的返回数据。

因为录音数据其他线程会同步读取，所有存在线程对数据同步操作安全性问题。

当录音结束时，清理临时数据和录音写入有个时间差，也要注意写入数据的线程安全。停止录音并不是马上停止，会有个延时。

1. 音频编解码层

协议层生成数据包后进行编码生成音频数据。

录音数据进行解码，每次解码长度1024\*8bit

1. 数据解析层

解码后的数据需要进行解析，并做相应的处理，生成一个dictionary数据包。

1. 线程支持同步异步可选

当前模式是同步阻塞式，在进行某项操作的时候不能进行其他操作。需要添加一个异步方式。

1. 去掉多余代码

有很多无用的代码，去掉那些多余的代码。

1. 问题
2. 功能协议层如果使用operationQueue来控制，能否实现异步？
3. 停止录音和clean CacheData的时间间隔？
4. Sync中断处理当前使用NSRunLoop方式，参照ASI的sync进行优化是否更合理？
5. 定时器？GCD定时器和NSTimer实现机制不一样，不能使用NSTimer，但是stopTimer时，对线程数据安全的影响？
6. 资源使用率？
7. 流程图

同步

接口

异步

通知中心

临时数据

编解码

音频通讯

设备

协议