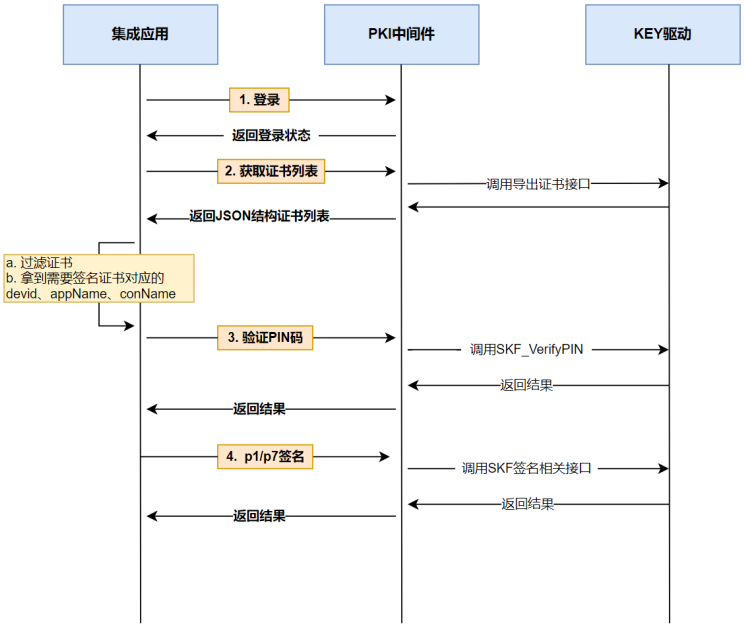
PKI中间件签名集成

时序图

- 主要关注带编号、背景为黄色的调用流程即可



集成注意事项

- 测试
  - 请先安装中间件
- 集成过程
  - 调用接口时，json数据解析/封装
  - 有效数据过滤
    - **过滤出签名key**
      - 调用getAllCert后获得响应的json数据后，匹配manufacturer字段，将KOAL、Microsoft **数据过滤后，即为实际签名使用的key
    - **过滤出签名证书**
      - 调用getAllCert后获得响应的json数据后，keyUsage字段，0是加密证书、1是签名证书、2是签名加密证书。过滤keyUsage为1的数据即可

1. WEB应用集成

1.1 业务调用流程

- **登录** ⇒ **获取证书列表** ⇒ **校验PIN码** ⇒ **数据签名/pkcs#7数据签名** ⇒ **登出**
  - 上述为集成流程，若使用js demo测试时则需要使用下述流程
    - 登录 ⇒ 获取证书列表 ⇒ **获取设备列表** ⇒ **获取应用列表** ⇒ **获取容器列表** ⇒ 校验PIN码 ⇒ 数据签名/pkcs#7数据签名 ⇒ 登出

1.2 接口调用示例

- 见js测试demo：`thriftclient_js\signClient.html`

1.3 集成接口介绍

- 详情见 **pki安全中间件接口规范** 文档
  - 登录:login
    - 5.1章节
  - 获取证书列表:getAllCert
    - 6.6章节
  - 校验PIN码:verifyPIN
    - 6.1章节
  - 数据签名:signData
    - 7.1章节
  - pkcs#7数据签名:signMessage
    - 7.2章节

2. 本地应用集成

2.1 pkiAgent动态库介绍

- 本地应用若需要集成pki中间件，需要依赖pkiAgent库

- 集成时需链接libpkiAgent4c动态库，库位于pkiAgent压缩包解压后的\pkiAgent\archs\路径

2.2 接口调用流程

- 本地应用的集成接口较简单，共5个主要接口
  - createAgent
    - 注册Agent，并注册Ukey插拔监控
  - loginAgent
    - 登录Agent
  - reqSync
    - 同步调用接口，业务调用接口，以下接口都通过此接口调用
      - 获取证书列表
      - 验证PIN码
      - p1/p7签名
  - logoutAgent
    - 登出Agent
  - releaseAgent
    - 释放Agent

2.3 集成示例

```
/*
*  通知回调接口
*/
bool msgNotify(const kpkiResp *pResp, void *pUserData) {
    if (pUserData) {
        const char *pData = (const char *)pUserData;
        printf("get parm=\"%s\" from callBack msgNotify\n", pData);
    }

    switch (pResp->msgType) {
        case 0x0FFF0001: {  ///设备插入
            printf("key insert\n");
            break;
        }
        case 0x0FFF0002: {  ///设备拔出
            printf("key remove\n");
            break;
        }
        case 0x0FFF0003: {  ///设备修改
            printf("key changed\n");
            break;
        }
```

```
        case 0x0FFF0004: {  /// Session关闭
            printf("session closed\n");
            break;
        }
        case 0x0FFF0031: {
            printf("get user token\n");
            break;
        }
    }
    return true;
}

int main(int argc, char **argv) {
    ///同步请求demo
    const char *pTest = "testing";
    do {
        ///初始化AGENT
        if (!createAgent(&msgNotify, NULL, (void *)pTest)) {
            printf("create Agent failed\n");
            break;
        }

        /*
            数据签名接口调用顺序（登录 -》 获取证书列表 -》 校验PIN码 -》 数据签名\pkcs7签名 -》登出），
            详细示例如下：
        */

        /*
            登录Agent
            @param {appName}应用名称，自定义
            @param {appID}应用ID，可为UUID，自定义
            @param {token}应用令牌，可为UUID，自定义
        */
        if (!loginAgent("11111-111", "22222-222", "33333-33")) {
            printf("login failed");
            break;
        }

        /*
            获取证书列表接口
            获取设备ID、设备名、应用名等参数，解析该接口获取的参数
            获取证书数据，在这一步做过滤操作
        */
        std::string List;  //用来存放getAllCertList接口掉用后获取到的数据
        getAllCertList(List);
        parseGetcertlist(List, mapParseData);

        /*
            遍历证书
            依次获取到证书SN，选择自己需要的SN（不同应用可根据需求可弹框让用户选择）
            根据选择的证书SN，获取SN对应设备的devId、appName、conName
        */
        std::map<int, CertInfo>::iterator mapParseInfo;
```

```
        for (mapParseInfo = mapParseData.begin(); mapParseInfo !=
mapParseData.end(); mapParseInfo++) {
            printf("number=%d SN=%s\n", mapParseInfo->first, mapParseInfo-
>second.SN.c_str());
        }

        while (1) {
            if (0 == mapParseData.size()) {
                printf("certs is NULL\n");
                return 0;
            } else {
                printf("please input number:");
                char c;
                ///采用%*忽略输入流中剩余的字符
                scanf("%c%*c", &c);
                std::map<int, CertInfo>::iterator mapKeyInfo;
                int keyValue = atoi(&c);
                mapKeyInfo = mapParseData.find(keyValue);
                if (mapKeyInfo == mapParseData.end()) {
                    printf(" error number,keyValue=%d\n", keyValue);
                    continue;
                } else {
                    /*
                        PIN码校验，需要传应用对应的pin码
                        接口参数设备ID、应用名称使用【获取证书列表】接口获取到的参数即
可
                        其余参数见syncVerifyPIN实现函数
                   */
                    gDevice.syncVerifyPIN(mapKeyInfo->second.devId, mapKeyInfo-
>second.appName, 1, "111111");

                    /*
                        数据签名
                        接口参数设备ID、应用名称、容器名称使用【获取证书列表】接口获取
到的参数即可
                        其余参数见syncSignData实现函数
                    */
                    // gSignx.syncSignData(mapKeyInfo->second.devId,mapKeyInfo-
>second.appName,mapKeyInfo->second.conName, ...);

                    /*
                        P7签名
                        接口参数设备ID、应用名称、容器名称使用【获取证书列表】接口获取
到的参数即可
                        其余参数见syncSignMessage实现函数
                    */
                    // gSignx.syncSignMessage(mapKeyInfo->second.devId,mapKeyInfo-
>second.appName,mapKeyInfo->second.conName, ...);
                    break;
                }
            }
        }
    } while (0);
```

```
    ///登出
    logoutAgent();
    ///释放agent
    releaseAgent();
    return 0;
}
```

2.4 集成文档

- 建议以参考demo为主