安全支付服务 Android 应用开发指南

文件版本：3.5.1.1

2012-05-09

前言

1. 面向读者

本文档主要面向需要接入支付宝安全支付的商户的开发人员。

2. 读者所需技能

读者需有基本的程序开发背景，掌握 java 及 Android 程序开发。

3. 开发环境要求

OS: Windows, Linux, Mac, JRE1.6 及以上

SDK: SDK1.6 以上

IDE： Eclipse with ADT

建议： Android SDK 最好在线更新至最新版

版权信息

目录

第一章 安全支付服务简介

1.1 安全支付服务介绍

安全支付服务是安装在本地 Android 操作系统上的一个组件，主要用来向其它的应用程序提供便捷、安全以及可靠的支付服务。正如平常系统上所提供的其它服务，如电子邮件和电话服务一样。本文主要描述安全支付服务应用开发接口的使用方法，供合作伙伴以及其它第三方应用开发者参考。

1.2 安全支付服务业务流程

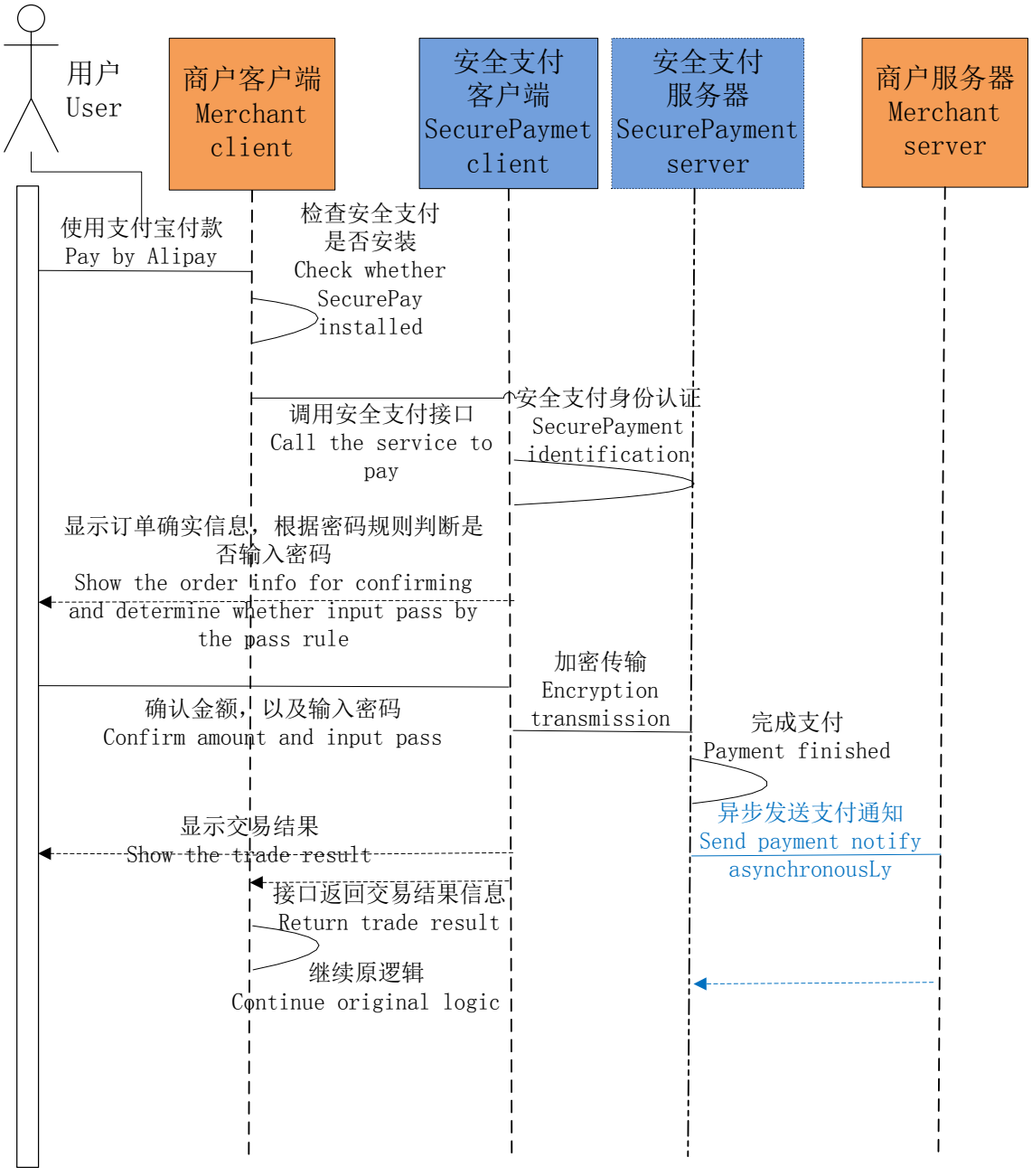

图 1-1 安全支付业务流程图

1.3 调用安全支付数据流程图

图 1-2 安全支付数据流程图

第二章 安全支付接入流程

2.1 接入前期准备

接入前期准备工作包括**商户签约**和**密钥配置**，已完成商户可略过。

2.1.1 商户签约

首先，商户需要在 https://ms.alipay.com 进行注册，并签约安全支付服务。签约成功后可获取支付宝分配的合作商户 ID(PartnerID)，账户 ID(SellerID)，如图：

图 2-1 商户 ID 获取示意图

签约过程中需要任何帮助请致电：**0571-88158090（支付宝商户服务专线）**

2.1.2 密钥配置

签约成功后，商户可登陆 https://ms.alipay.com 获取商户账号对应的支付宝公钥，具体获取步骤请见 3.2 RSA 密钥详解

接着，商户生成商户公钥和商户私钥（具体生成步骤请见 3.2 RSA 密钥详解），并登陆 https://ms.alipay.com，上传商户公钥（具体上传步骤请见 3.2 RSA 密钥详解）。

至此，接入前期准备工作完成，下一节将使用 demo 测试准备工作是否正确。

2.2 Demo

为了便于商户的接入，我们提供了安全支付 demo。通过本 demo，商户可测试 2.1 节的前期准备工作是否正确完成，同时还可参考 demo 的代码完成接入。（注意：请勿在模拟器下测试 demo，否则可能导致付款账户被锁定！）

2.2.1 Demo 配置运行

步骤 1:

解压下载的安全支付开发资料压缩包 WS_SECURE_PAY，进入目录“*WS_SECURE_PAY\Android*”，其中“*AppDemo4_0413*”即 demo 的项目文件，将其导入 eclipse，步骤如图：

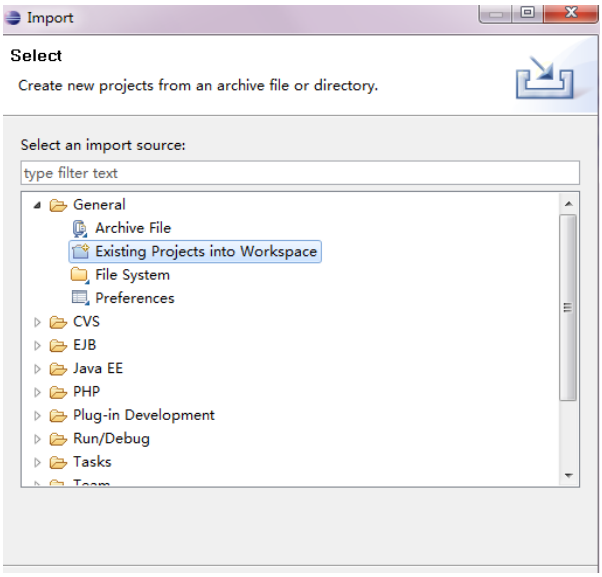

图 2- 2 Demo 导入示意图 a

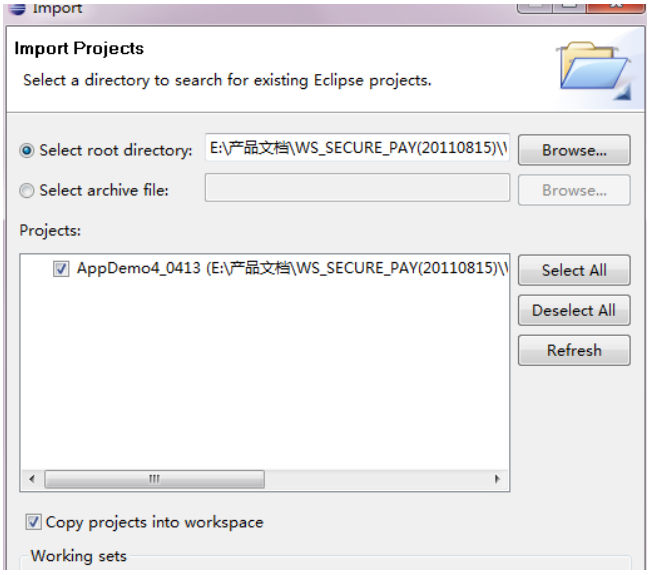

图 2- 3 Demo 导入示意图 b

项目结构如图：

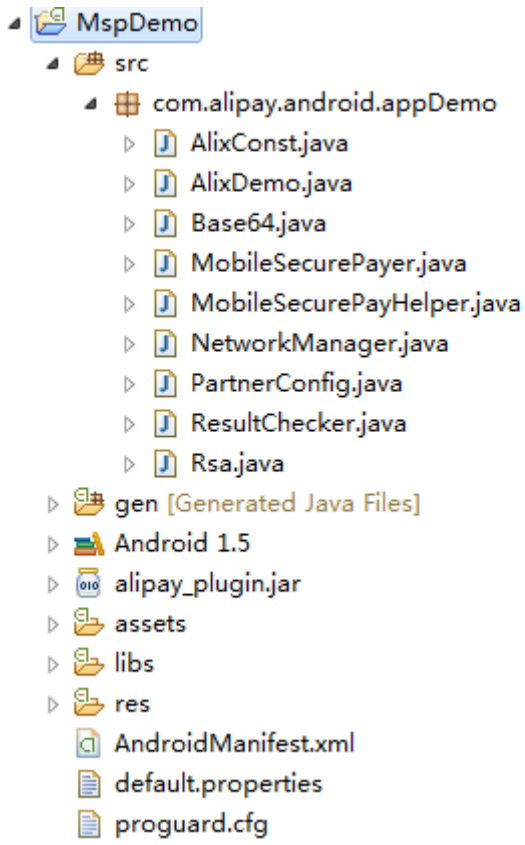

图 2- 4 Demo 项目结构图

步骤 2:

打开“*PartnerConfig.java*”文件，按照注释添加商户账号信息，具体包括：**合作商户 ID、账户 ID、支付宝公钥（即服务器公钥）、商户公钥、商户私钥**。如下图：

```
public class PartnerConfig {
    // 合作商户ID。用签约支付宝账号登录ms.alipay.com后，在账户信息页面获取。
    // partnerID. You can log in ms.alipay.com with your Alipay account
    // and get the partnerID on the page of account infomation.
    public static final String PARTNER = "";

    // 账户ID。用签约支付宝账号登录ms.alipay.com后，在账户信息页面获取。
    // accountID. You can log in ms.alipay.com with your Alipay account
    // and get the partnerID on the page of account infomation.
    public static final String SELLER = "";

    // 商户（RSA）私钥
    // private key of merchant
    public static final String RSA_PRIVATE = "";

    // 支付宝（RSA）公钥 用签约支付宝账号登录ms.alipay.com后，在密钥管理页面获取。
    // public key of Alipay.You can log in ms.alipay.com with your Alipay account
    // and get the partnerID on the page of key management.
    public static final String RSA_ALIPAY_PUBLIC = "";
    public static final String ALIPAY_PLUGIN_NAME = "alipay_plugin223_0309.apk";
}
```

图 2-5 商户信息配置截图

步骤 3：

在真机或者模拟器上运行项目（下图以模拟器为例），首次启动时，由于系统未安装安全支付服务，将出现以下提示：

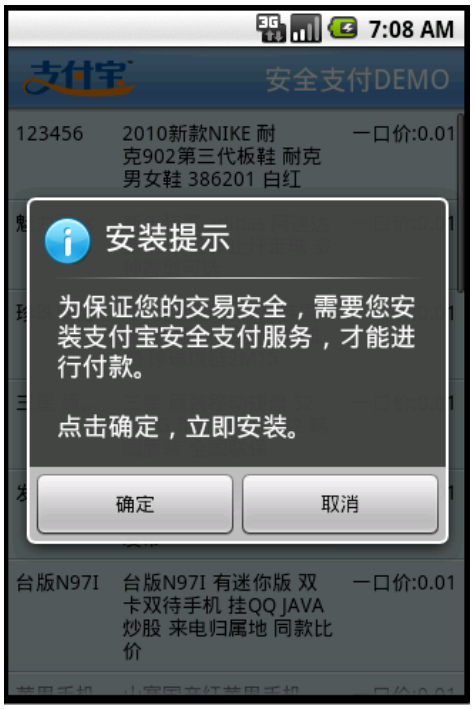

图 2-6 安全支付安装确定

此时，可以选择“确定”进行安装，安装后该提示不会再出现。此时点击任意商品进行购买，即启动安全支付：

图 2-7 安全支付启动

首次运行安全支付服务，会进行安全认证，此时选择“支付宝账户”付款（注意：付款用的账户应为某一真实账户，交易金额也是真实的！），输入账户信息，按照提示一步步进行：

图 2-8 安全认证

完成安全认证后，若出现如下提示，说明密钥配置有误，请仔细阅读 3.2 RSA 密钥详解

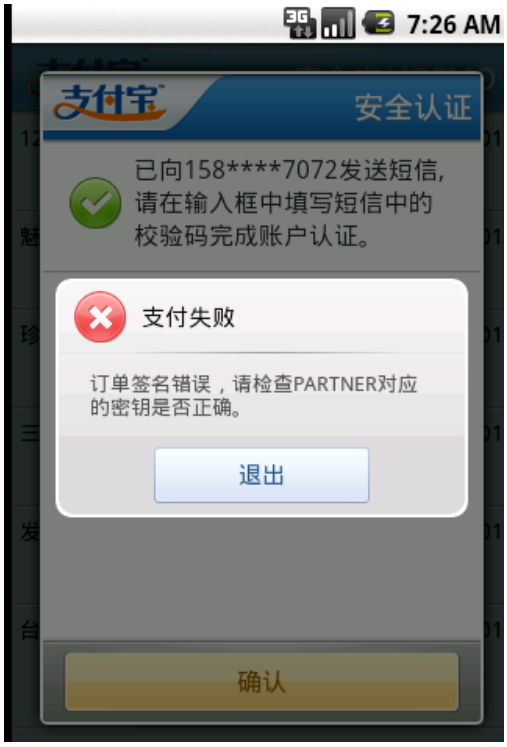

图 2- 9 验签错误-支付失败

若出现“确认支付”页面，说明密钥配置无误，接入前期准备工作全部正确完成。接下来将正式进行安全支付的接入集成。

2.2.2 Demo 结构说明

类名	说明
AlixDemo	主 Activity
Base64	Base64 编码类，签名及验签，必须
MobileSecurePayer	封装了对安全支付的调用，可参考性较强
MobileSecurePayHelper	实现安全支付插件的检测，更新下载，安装，可参考性较强
Networkmanager	网络连接管理
PartnerConfig	商户账户信息配置
ResultChecker	AlixPay 返回结果的解析处理，可参考性较强
Rsa	RSA 签名验签类，必须

2.3 安全支付集成

本章指导在商户项目中集成安全支付，关键代码以 Demo 为例。

步骤 1：添加 jar 文件

添加 demo 中的 alipay_msp.jar 包添加工程中。

步骤 2：初始化安全支付服务

在调用安全支付进行支付前，需要先初始化安全支付服务，主要代码如下：

```
    private ServiceConnection mAlixPayConnection = new ServiceConnection() {
        public void onServiceConnected(ComponentName className, IBinder service) {
            // Wake up the binder to continue.
            synchronized (lock) {
                mAlixPay = IAlixPay.Stub.asInterface(service);
                lock.notify();
            }
        }

        public void onServiceDisconnected(ComponentName className) {
            mAlixPay = null;
        }
    };
...
        // Bind the service, then can call the service.
            mActivity.bindService(new Intent(IAlixPay.class.getName()),
                    mAlixPayConnection, Context.BIND_AUTO_CREATE);
```

步骤 3：订单数据生成

在调用安全支付时，需要提交订单信息 orderInfo 其中参数以“*key=value*”形式呈现，参数之间以“*&*”分割，所有参数不可缺。示例如下：（红色参数表示该参数值需与示例一致，不可自定义；蓝色参数表示值可自定义。具体参数说明请见订单信息描述）

```
partner="2088002007260245"&seller="2088002007260245"&out_trade_no=
"500000000006548"&subject="商品名称"&body="这是商品描述
"&total_fee="30"&notify_url="http://notify.java.jpxx.org/index.jsp
"&sign="kU2Fa3x6V985g8ayTozI1eJ5fHtm8%2FJGeJQf9in%2BcVmRJjHaExbirn
GGKJ%2F7B63drqc4Kjlk%2FSg6vtSIkOtdvVBrRDpYaKxXVqkJTzRYgUwrrpMudbIj
9aMS2O3dHG0GPyL4Zb6jKDYXHabGG0aBJY3QA7JuTJ23t6SqV%2B5f1xg%3D"&sign
_type="RSA"
```

其中 sign 值的生成，请见商品信息签名。需要特别注意的是：对数据签名后得到的 sign 值必须进行 URLEncode，之后才可作为参数。

步骤 4：调用安全支付

准备好参数后，即可调用安全支付进入支付流程并获得调用结果，代码如下：

```
result = mAlixPay.Pay(orderInfo);
```

AliXPay 函数具体说明请见 AlixPay 方法描述。

步骤 5：支付结果获取和处理

调用安全支付后，将通过两种途径获得支付结果：

1、*AliXpay.pay()*方法的返回

该方法将返回表示支付结果的字符串，详细信息详见

2、支付宝服务器通知

商户需要提供一个 http 协议的接口，包含在参数里传递给安全支付，即 notify_url。支付宝服务器在支付完成后，会用 POST 方法调用 notufy_url，以 xml 为数据格式传输支付结果，详见

接下来以 Demo 代码为例，介绍整个流程：

```
/*
 * Initialize serviceConnection to get the stub
 * Referrence class: MobileSecurePayer
*/
    private ServiceConnection mAlixPayConnection = new ServiceConnection() {
        public void onServiceConnected(ComponentName className, IBinder service) {
            // wake up the binder to continue.
            synchronized (lock) {
                mAlixPay = IAlixPay.Stub.asInterface(service);
                lock.notify();
            }
        }
/*
 * Prepare the Info which will be used to call SecurePayment service
 * Referrence class: AlixDemo
 * Referrence method: onItemClick
*/
            String orderInfo = getOrderInfo(arg2);
            String signType = getSignType();
            String strsign = sign(signType, orderInfo); //
            strsign = URLEncoder.encode(strsign);
            String info = orderInfo + "&sign=" + "\"" + strsign + "\"" + "&"
                    + getSignType();
```

```
/*
 * Call the method of pay() in SecurePayment service, this method is the
encapsulation
 * of pay().
 * Referrence class: MobileSecurePayer
*/
public boolean pay(final String orderInfo, final Handler callback,
            final int myWhat, final Activity activity) {
        // Return false if it's paying.
        if (mIsPaying)
            return false;
        // Return true if it's not paying.
        mIsPaying = true;
        mActivity = activity;
        // bind the service.
        if (mAlixPay == null) {
            mActivity.bindService(new Intent(IAlixPay.class.getName()),
                    mAlixPayConnection, Context.BIND_AUTO_CREATE);
        }

        new Thread(new Runnable() {
            public void run() {
                try {
                    /**
                     * wait for the service bind operation to completely
                     * finished. Note: this is important, otherwise the next
                     * mAlixPay.Pay() will fail.
                     */
                    synchronized (lock) {
                        if (mAlixPay == null)
                            lock.wait();
                    }

                    // register a Callback for the service.
                    mAlixPay.registerCallback(mCallback);

                    // call the MobileSecurePay service.
                    result = mAlixPay.Pay(orderinfo);
```

```
/**
  * set the flag to indicate that we have finished.
  * unregister the Callback, and unbind the service.
*/
                    mIsPaying = false;
                    mAlixPay.unregisterCallback(mCallback);
                    mActivity.unbindService(mAlixPayConnection);

                    // send the result back to caller.
                    Message msg = new Message();
                    msg.what = myWhat;
                    msg.obj = result;
                    callback.sendMessage(msg);
                } catch (Exception e) {
                    e.printStackTrace();
                    // send the result back to caller.
                    Message msg = new Message();
                    msg.what = myWhat;
                    msg.obj = e.toString();
                    // msg.obj = strRet;
                    callback.sendMessage(msg);
                }
            }
        }).start();

        return true;
    }
}
```

2.4 应用发布

目前，我们为第三方应用客户端提供了两种集成安全支付服务的方式。第一种是运行时安装，即将安全支付服务安装包 apk 与第三方应用客户端整合在一起，在恰当的时机，由第三方应用客户端释放并安装安全支付服务安装包 apk。另一种则称为动态下载安装，在此种情况下，安全支付服务安装包 apk 是预先存放在约定的远程服务器中，第三方应用客户端可以从此处下载 apk 并进行安装。

为了提升用户体验，避免捆绑安装旧版本的安全支付服务之后，接着又需要重新升级并安装成最新版本，我们目前推荐结合使用以上两种集成方式。在最新的 demo 中，会首先连接支付宝服务器，检测并判断捆绑在第三方客户端中的 alipay_plugin.apk 是否为最新版本，如果是，则直接安装。如果捆绑在第三方客户端中的 alipay_plugin.apk 不是最新版本，则从服务器中下载最新版本的安装包，然后进行安装。

2.4.1 运行时安装

运行时安装的具体步骤如下所示：

1） 将 alipay_plugin.apk 作为资源复制到第三方应用工程中的 assets 目录。

2） 第三方应用在需要付款时（或者其它恰当时机），先检测安全支付服务是否已经安装。如果尚未安装，则从 assets 目录中提取 alipay_plugin.apk 到手机存储。

3） 安装手机存储中的安全支付服务 apk。

以下是一个代码片段，演示上述过程。

```
// Install the APK of SecurePayment which is included in the merchant's App.
public boolean retrieveApkFromAssets(Context context, String fileName,
        String path) {
    boolean isRetrieve = false;
    InputStream input = null;
    FileOutputStream output = null;
    try {
        input = context.getAssets().open(fileName);
        File file = new File(path);
        file.createNewFile();
        output = new FileOutputStream(file);
        byte[] temp = new byte[1024];
        int i = 0;
        while ((i = input.read(temp)) > 0) {
            output.write(temp, 0, i);
        }
        isRetrieve = true
```

```
    } catch (IOException e) {
            e.printStackTrace ();
        }finally{
            if(output != null){
                try {
                    output.close();
                } catch (IOException e) {
                }
            }
            if(input != null){
                try {
                    input.close();
                } catch (IOException e) {
                }
            }
        }
        return isRetrieve;
    }
```

2.4.2 动态下载安装

动态下载安装的具体步骤如下所示：

1） 由支付宝预先将安全支付服务 apk 放置到约定的远程下载服务器。

2） 第三方应用在需要付款时（或者其它恰当时机），先检测安全支付服务是否已经安装。如果尚未安装，则从约定的远程下载服务器提取 alipay_plugin.apk 到手机存储。

3） 安装手机存储中的安全支付服务 apk。

以下是一个代码片段，演示上述过程。

```
    // Download the APK of SecurePayment
    public boolean retrieveApkFromNet(Context context, String strurl,
            String filename) {
        boolean isRetrieve = false;
        try {
            NetworkManager networkManager = new NetworkManager(this.mContext);
            isRetrieve = networkManager.downloadToFile(context, strurl,
                    filename);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return isRetrieve;
    }
```

第三章 RSA 详解

*以下内容加*号为重点*

3.1 RSA 和 OpenSSL 介绍

3.1.1 什么是 RSA

RSA 是一种非对称的签名算法，即签名密钥（私钥）与验签密钥（公钥）是不一样的，私钥用于签名，公钥用于验签。

在与支付宝交易中，会有 2 对公私钥，即商户公私钥，支付宝公钥。

商户公私钥：由商户生成，商户私钥用于对商户发往支付宝的数据签名；商户公钥需要上传至支付宝，当支付宝收到商户发来的数据时用该公钥验证签名。

支付宝公钥：支付宝提供给商户，当商户收到支付宝发来的数据时，用该公钥验签。

3.1.2 为什么要用 RSA

使用这种算法可以起到防止数据被篡改的功能，保证支付订单和支付结果不可抵赖(商户私钥只有商户知道)。

3.1.3 什么是 OpenSSL

一句话概括：OpenSSL 是基于众多的密码算法、公钥基础设施标准以及 SSL 协议安全开发包。

3.1.4 为什么要用 OpenSSL

通过 OpenSSL 生成的签名和内置的算法可以做到跨平台，这样在不同的开发语言中均可以签名和验签。

3.2 RSA 密钥详解*

3.2.1 找到生成 RSA 密钥工具

(1)下载开发指南和集成资料，如下图，您能看到此文档说明指南和集成包已经下载了。

图 3- 1 文档下载

(2)解压下载的压缩包(WS_SECURE_PAY)，找到并解压 openssl-0.9.8k_WIN32(RSA 密钥生成工具).zip 工具包

图 3- 2 openssl

3.2.2 生成商户密钥并获取支付宝公钥

(1) 生成原始 RSA 商户私钥文件

假设解压后的目录为 c:\alipay，命令行进入目录 C:\alipay\bin，执行“*openssl genrsa -out rsa_private_key.pem 1024*”，在 C:\alipay\bin 下会生成文件 rsa_private_key.pem，其内容为原始的商户私钥（请妥善保存该文件），以下为命令正确执行截图：

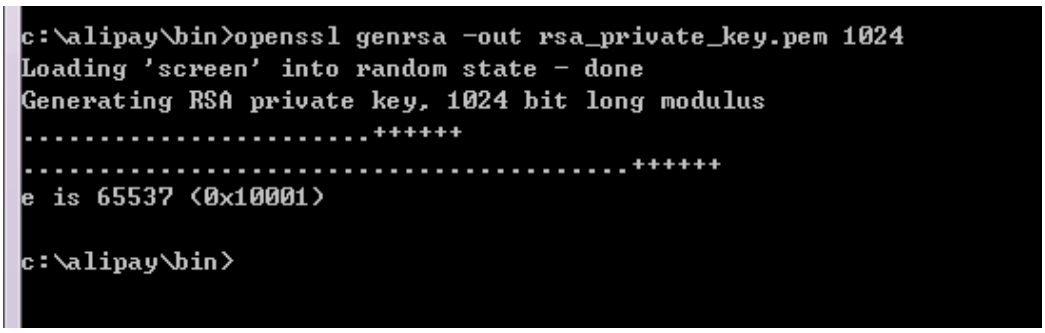

图 3- 3 生成原始 RSA 商户私钥文件

(2) 将原始 RSA 商户私钥转换为 pkcs8 格式

命令行执行“ *openssl pkcs8 -topk8 -inform PEM -in rsa_private_key.pem -outform PEM*

-nocrypt”得到转换为 pkcs8 格式的私钥。复制下图红框内的内容至新建 txt 文档，去掉换行，最后另存为“private_key.txt”（请妥善保存，签名时使用）。

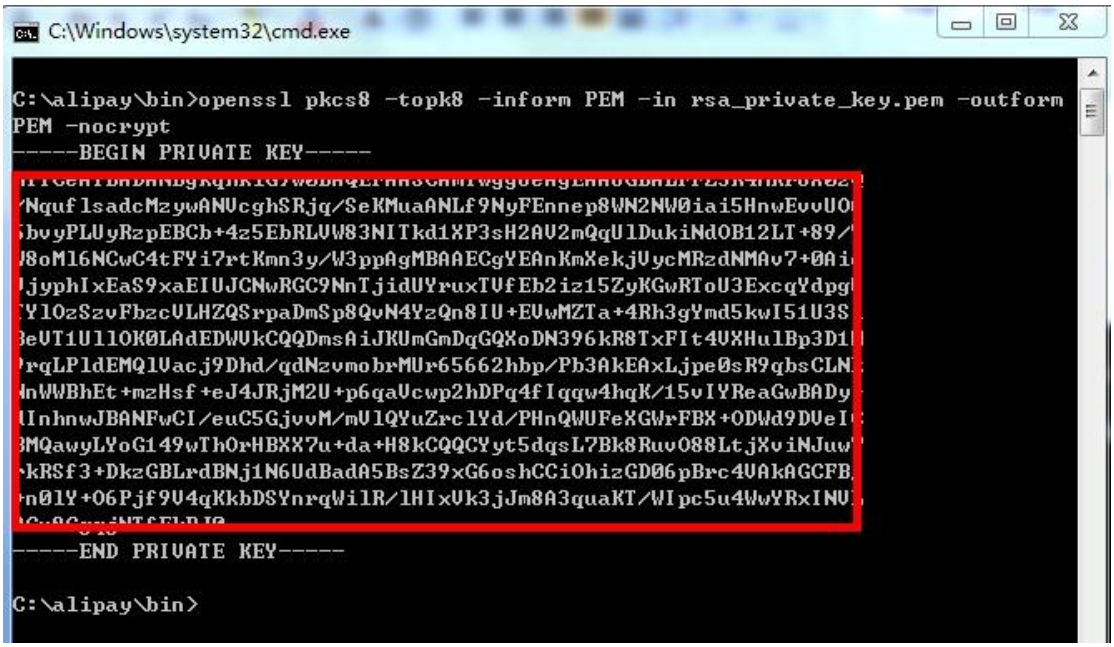

图 3-4 转换私钥格式

(3) 生成 RSA 商户公钥

命令行执行“ *openssl rsa -in rsa_private_key.pem -pubout -out rsa_public_key.pem*”，在 C:\alipay\bin 文件夹下生成文件 rsa_public_key.pem。接着用记事本打开 rsa_public_key.pem，复制全部内容至新建的 txt 文档，删除文件头“-----BEGIN PUBLIC KEY-----”与文件尾“-----END PUBLIC KEY-----”及空格、换行，如下图。最后得到一行字符串并保存该 txt 文件为“public_key.txt”。

```
1  -----BEGIN PUBLIC KEY-----
2  MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCuQBQMjMX+ossoDXoi5DlcDOsf
3  6hVT6twgwfuVbyouTSI/cHjH2xpu1S/RD4xXHBi/60GNmewAro2T70i1wxuMpgcD
4  S+3S/0z+4xyrW8ewXfeGmUVPKlyPbkmFeL/OuKWNdhpObOmCyByZPts0lkFKDPb9
5  B5lxZQzj6b+82L31kQIDAQAB
6  -----END PUBLIC KEY-----
```

去掉头尾注释、换行、空格

```
IGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCuQBQMjMX+ossoDXoi5DlcDOsf6hVT6twgwfuVbyouTSI/cHjH2xpu1S
```

图 3-5 生成公钥

(4) 上传商户公钥至支付宝

浏览器访问 https://ms.alipay.com/index.htm 并用签约帐号登录，点击菜单栏“我的产品”，右侧点击“密钥管理”，见下图红色框内

图 3- 6 商户公钥上传

点击“上传”，选择步骤(3)生成的“public_key.txt”并完成上传。

(5) 获取 RSA 支付宝公钥

成功上传公钥至支付宝后，页面显示如下：

图 3- 7 支付宝公钥获取

其中红色框内部分即支付宝公钥，请复制至新建 txt 文档，去掉换行和空格，妥善保存（用于验签收到的支付宝通知）。

3.3 RSA 签名和验签 *

建议：签名和验签尽量在商户服务器端进行，同时一些敏感数据（如公私钥等）也应存储在服务器端，避免可能的安全隐患。

3.3.1 RSA 签名

(1) 在项目中添加下面的类：

Base64.java

Rsa.java//包含了签名验签等方法

(2) 生成商品订单：

可参考 Demo 中 *AlixDemo.java* 的方法：

private String getOrderInfo()

例子：出售商品（subject）“Iphone4”，价格（total_fee）“1”元，外部交易号（out_trade_no）“zzzz”，商品描述（body）为“秒杀”，订单支付完成通知 URL（notify_url）为 http://notify.java.jpxx.org/index.jsp

则生成如下商品信息字串：

```
partner="xxxx"&seller="yyyy"&out_trade_no="zzzz"&subject="Ipone4"&
body="                                秒                                杀
"&total_fee="1"&notify_url="http%3A%2F%2Fnotify.java.jpxx.org%2Find
ex.jsp"
```

备注：notify_url 的值需要进行 URLEncode 编码。

(4) 对商品信息进行 RSA 签名

可使用 Demo 中 Rsa.java 的方法：

public static String sign(String content, String privateKey)

String content：代签名字符串（红色部分）

String privateKey：商户私钥(pkcs8 转换后的商户私钥)

返回值：签名值（蓝色部分，传递给安全支付前需要 URL 编码）

```
partner="xxxx"&seller="yyyy"&out_trade_no="zzzz"&subject="Ipone4"&
```

```
body="秒杀"&total_fee="1"&notify_url="http%3A%2F%2Fnotify.java.jpxx.org%2Findex.jsp"&sign_type="RSA"&sign="O0I1APPVQcK5bbSgdeFx9HB3Yu/U2+akTZ3T0/P7v3g7XD7TsQCprb6O9Nybr8CDIrztdUseQN/TCXuEvCU2cvCt1xX9UUyI6fOxXxQFlDWx7IE2S7Zo5wOeVWmMBnCQCV8iDjcNxGHwhtCT09bVVf0wbaOiHXvAYzWlvPhyR+0="
```

3.3.2 RSA 验签

(1) 使用 RSA 类进行验签：

public static boolean doCheck(String content, String sign, String publicKey)

String content：待验签的字符串（红色部分）

String sign：签名值（蓝色部分）

String publicKey：支付宝公钥

返回值：验签成功则返回true，反之返回false.

以下是一个订单支付成功完整信息的示例：

```
resultStatus={9000};
result={partner="2088002007260245"&seller="2088002007260245"&out_trade_no="600000000006891"&subject="商品名称"&body="这是商品描述"&total_fee="1"&notify_url="http%3A%2F%2Fnotify.java.jpxx.org%2Findex.jsp"&success="true"&sign_type="RSA"&sign="O0I1APPVQcK5bbSgdeFx9HB3Yu/U2+akTZ3T0/P7v3g7XD7TsQCprb6O9Nybr8CDIrztdUseQN/TCXuEvCU2cvCt1xX9UUyI6fOxXxQFlDWx7IE2S7Zo5wOeVWmMBnCQCV8iDjcNxGHwhtCT09bVVf0wbaOiHXvAYzWlvPhyR+0="}
```

备注：result返回的json字串需要处理转义字符（可参考Demo中checkSign函数的处理方式）否则可能导致验签无法通过。

(2) 为了方便商户接入安全支付服务，我们将签名和验签的方法封装成 *Rsa.java* 提供给大家使用，可从 demo 中提取。

第四章 通知结果

4.1 AlixPay 方法返回的结果

支付结果的处理可以参考 Demo 中的类 ResultChecker.java

结果信息详细描述如下：

表 4-1 AlixPay 返回结果描述表

字段名称	描述	属性	备注
resultStatus	本次操作的状态返回值。	字符串	用来标识本次调用的结果，具体可能的取值，请查看错误代码列表
result	本次操作返回的结果数据。	字符串	订单支付结果信息。字符串格式，形式一般如下： *partner=""&seller=""&out_trade_no=""&subject=""&body=""&total_fee="30"&notify_url=""&success="true"&sign_type="RSA"&sign="xxx"* 其中： *&success="true"&sign_type="RSA"&sign="xxx"* 之前的部分为商户的原始数据。 success，用来标识本次支付结果。 sign="xxx"为支付宝对本次支付结果(红色部分)的签名，商户可以使用签约时支付宝提供的公钥进行验证。

结果判断说明：

需要通过 resultStatus 以及 result 字段的值来综合判断并确定支付结果。在 resultStatus=9000，并且 success="true"以及 sign="xxx"校验通过的情况下，证明支付成功。其它情况归为失败。较低安全级别的场合，也可以只通过检查 resultStatus 以及 success="true"来判定支付结果。以下为订单支付成功的完成信息示例：

```
resultStatus={9000};
result={partner="2088002007260245"&seller="2088002007260245"&out_trade_no="600000000006891"&subject="商品名称"&body="这是商品描述"&total_fee="1"&notify_url="http://notify.java.jpxx.org/index.jsp"&success="true"&sign_type="RSA"&sign="O0I1APPVQcK5bbSgdeFx9HB3Yu/U2+akTZ3T0/P7v3g7XD7TsQCprb6O9Nybr8CDIrztdUseQN/TCXuEvCU2cvCt1xX9UUyI6fOxXxQFlDWx7IE2S7Zo5wOeVWmMBnCQCV8iDjcNxGHwhtCT09bVVf0wbaOiHXvAYzWlvPhyR+0="}
```

4.2 notify_url 通知说明

4.2.1 什么是 Notify_url

支付宝通过访问商户提供的地址的形式，将交易状态信息发送给商户服务器。商户通过支付宝的通知判断交易是否成功，具体如下：

商户地址：提供一个 http 的 URL(例:http://www.partnertest.com/servlet/NotifyReceiver)，支付宝将以 POST 方式调用该地址。

通知触发条件：交易状态发生改变，如交易从“创建”到“成功”或“关闭”。

商户返回信息：商户服务器收到通知后需返回纯字符串“success”，不能包含其他任何 HTML 等语言的文本。

通知重发：若支付宝没有收到商户返回的“success”，将对同一笔订单的通知进行周期性重发 (间隔时间为：2 分钟,10 分钟,10 分钟,1 小时,2 小时,6 小时,15 小时共 7 次)。

交易判断条件：收到 trade_status=TRADE_FINISHED（如果签有高级即时到帐协议则 trade_status=TRADE_SUCCESS）的请求后才可判定交易成功（其它 trade_status 状态请求可以不作处理）

4.2.2 Notify_url 接收数据示例

```
notify_data=<notify><partner>2088201564809153</partner><discount>0.00</discount><payment_type>1</payment_type><subject>测试商品</subject><trade_no>2012041821018998</trade_no><buyer_email>xxxx@xx.com</buyer_email><gmt_create>2012-04-18 11:06:52</gmt_create><quantity>1</quantity><out_trade_no>0418110644-1034</out_trade_no><seller_id>2088201564809153</seller_id><trade_status>TRADE_SUCCESS</trade_status><is_total_fee_adjust>N</is_total_fee_adjust><total_fee>0.01</total_fee><gmt_payment>2012-04-18 11:06:52</gmt_payment><seller_email>alipay@alipay.com</seller_email><price>0.01</price><buyer_id>2088302175666987</buyer_id><use_coupon>N</use_coupon></notify>&sign=ZPZULntRpJwFmGNIVKwjLEF2Tze7bqs60rxQ22CqT5J1UlvGo575QK9z/+p+7E9cOoRoWzqR6xHZ6WVv3dloyGKDR0btvrdqPgUAoeaX/YOWzTh00vwcQ+HBtXE+vPTfAqjCTxiiSJEOY7ATCF1q7iP3sfQxhS0nDUug1LP3OLk=
```

参数说明（注意，具体接收到的数据可能与例子有细微出入，仅确保下表内参数不变）：

notify_data：待验签数据(红色部分)，主要参数说明请见下表

sign：签名(蓝色部分)

备注：在调用验签方法时，需要将”*notify_data*=”这几个字符加上，一并验签，以上红色部分

具体的验签方法请参考 3.3.2 RSA 验签

Notify_data 参数说明

参数名	说明
trade_status	用于判断交易状态，值有： TRADE_FINISHED：表示交易成功完成 WAIT_BUYER_PAY：表示等待付款 TRADE_SUCCESS：表示交易成功（高级即时到帐）
total_fee	交易金额
subject	商品名称
out_trade_no	外部交易号（商户交易号）
trade_no	支付宝交易号
gmt_create	交易创建时间
gmt_payment	交易付款时间 若交易状态是“WAIT_BUYER_PAY”则无此参数

第五章 常见问答

1. 客户端验签，报“订单信息被篡改”是什么问题？

可能有以下2种情况

a) 有可能数据在传输过程中被黑客截取和篡改

b) 检查*plaintext*(待签名的字符串)中是否有以下四个符号，如果参数当中包含了这四个字符也会报“订单信息被篡改”：

+ 加号

& 连接符

“ 双引号

= 等号

2. 客户端调用安全支付时对 body 和 subject 进行 URLEncode 会报签名错误，到底哪些需要 URLEncode？

调用安全支付接口时，只需要对参数sign进行*URLEncode*，其他参数都不能*URLEncode*，安全支付服务插件会对所有参数进行*URLEncode*，所以不用担心中文乱码

3. 上传商户公钥报格式错误怎么办？

首先确认上传的位置是否是*RSA*的下面，注意不要是*DSA*，无线目前不支持*DSA*加密；另外请检查上传的文件中是否去除注释、空格、换行等，必须是一行的字符串

附录 A 错误代码列表

以下为安全支付服务所定义的错误代码：

表 A-1 系统定义的错误代码表

错误代码	含义
9000	操作成功
4000	系统异常
4001	数据格式不正确
4003	该用户绑定的支付宝账户被冻结或不允许支付
4004	该用户已解除绑定
4005	绑定失败或没有绑定
4006	订单支付失败
4010	重新绑定账户。
6000	支付服务正在进行升级操作。
6001	用户中途取消支付操作。
6002	网络连接异常。

附录 B 安全支付服务接口

1 安全支付服务接口列表

目前 Android 平台上的安全支付服务接口如下表所示：

表B-1 支付服务接口表

接口名称	接口描述
IAlixPay	提供外部商户订单的支付功能。后续可能还会增加其它相关功能。

接口所提供的方法如下：

表B-2 IAlixPay接口方法表

方法名称	方法描述
Pay	提供外部商户订单的支付功能。
test	检查开发环境搭建是否成功，仅供测试时使用。

2 AlixPay 主要方法描述

表B-3 Pay()方法信息描述

方法原型	
String Pay(String strInfo);	
方法功能	
提供外部商户订单的支付	
参数说明	
String strInfo	主要包含外部商户的订单信息，key="value"形式，以&连接。示例如下： partner="2088002007260245"&seller="2088002007260245"&out_trade_no="500000000006548"&subject="商品名称"&body="这是商品描述"&total_fee="30"&notify_url="http://notify.java.jpxx.org/index.jsp"&sign="kU2Fa3x6V985g8ayTozI1eJ5fHtm8%2FJGeJQf9in%2BcVmRJjHaExbirnGGKJ%2F7B63drqc4Kjlk%2FSg6vtSIkOtdvVBrRDpYaKxXVqkJTzRYgUwrrpMudbIj9aMS2O3dHG0GPyL4Zb6jKDYXHabGG0aBJY3QA7JuTJ23t6SqV%2B5f1xg%3D"&sign_type="RSA" 参考订单信息描述表查看各个字段的含义。

返回值
本方法调用的结果。字符串格式，形式一般如下： `resultStatus={}; result={}` 参考支付结果信息描述表查看各个字段的含义
该方法是安全支付服务提供的核心方法，是线程安全的。属于同一进程的多个线程对此方法的调用会按顺序排队进行，前一个调用返回后，下一个调用才会开始。非相同进程的多个线程的并发调用，没有这样的限制

3 订单信息描述

表 B-4 订单信息描述表

字段名称	描述	属性	备注
partner	合作商户 ID。应用开发商与支付宝签约接入支付服务时，由支付宝分配。	字符串	非空、16 位数字。商户可以用签约支付宝账号登录 https://ms.alipay.com 获取。
seller	账户 ID。订单付款成功后，钱会打到本账号中。	字符串	非空、16 位数字。商户可以用签约支付宝账号登录 https://ms.alipay.com 获取。
out_trade_no	商家自己产生的订单编号，由商家统一定义，但此号不能有重复。	字符串	非空、64 位字母、数字和下划线组成
subject	商品名称。由商家统一定义，可重复。	字符串	非空、64 位字符（含中文字符）
body	商品的具体描述信息。	字符串	非空、1024 位字符（含中文字符）
total_fee	本次支付的总费用。所有商品的费用总和，以人民币元为单位。如 1.50。	字符串	非空、大于 0 的数字（精度不超过两位的小数，如 1.00）
notify_url	商家提供的 url。订单支付结束时，支付宝服务端会	字符串	非空、255 位，需要符合 url 编码规则。

	回调这个 url，通知商家本次支付的结果。		
sign	上述订单信息的签名。对整个定单按支付宝约定的方式的签名，签名需包括如下参数： `partner=""&seller=""&out_trade_no=""&subject="  "&body=""&total_fee="30"&notify_url=""`	字符串	非空。 商家自己生成一对公私钥。用这个私钥对订单信息签名，公钥提供给支付宝，供支付宝在验证订单签名时使用。 需要符合 url 编码规则，并且字符编码为 UTF-8.
sign_type	签名类型。由商户选择，以下算法目前可用（RSA）。	字符串	非空，定值