基于验证码的短信验证机制

2011-10-11

# 背景

基于Android手机开发的应用Inspector打算采取收费模式。希望采用的注册机制如下：

1. 每个用户可以购买多个验证码，每个验证码只能用于一部手机的一次验证。也就是，如果该手机刷新了ROM，则该APK无法再用原验证码重新激活
2. 该验证码在第一次激活号

# 方案

1. 用户安装APK后，发送一条短信（内容为验证码）到该手机。
2. 该验证码可激活设置对话框，然后再次短信验证 （确保该验证码之前没被用过）

其中1中的验证码由明码和暗码组成：明码（N位）+暗码（M位）=验证码（N+M位）

采用AES加密验证，即明码及暗码是否能满足AES算法匹配。

2采用CS结构，即用户手机作为Client，注册时向服务机（Service）发送一条短信，Service接收后在数据库中验证该机器是否可以获得有效注册，并发回注册码短信，Client手机收到短信进行校验，校验成功即注册成功，并将有效信息写入手机，下次再激活设置对话框则仅只需发送验证码到手机上即可。

下文仅详细讨论2，即手机短信验证的具体设计。

# 设计

Server以Android手机上的service形式存在，即手机即是我们的验证服务器。

## 数据库

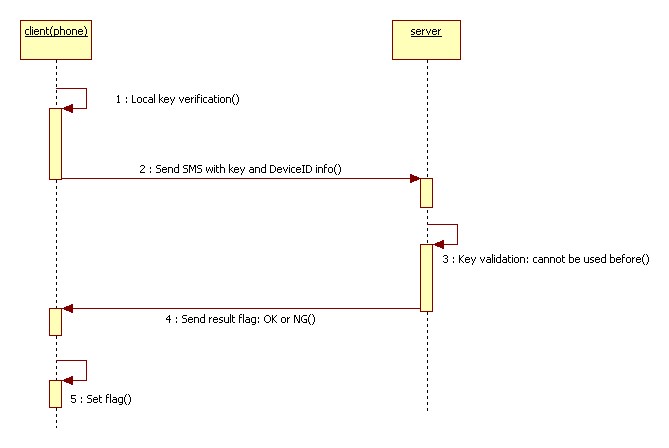
Server在Android的SQLite数据库中创建并维护的table，定义如下：

**Table1：** **inspector\_auth\_key**

**该table记录用户的有效记录数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **类型** | **能否为空** | **说明** |  |
| **\_id** | long | N | ID（自增长） |  |
| **licensekey** | varchar(16) | N | 激活码 |  |
| **deviceid** | varchar(32) | Y | 消费设备ID |  |
| **phonenum** | varchar(16) | Y | 手机号码 |  |
| **phonemodel** | varchar(16) | Y | 手机型号 |  |
| **androidver** | varchar(8) | Y | 手机Android系统版本 |  |
| **consumedate** | datetime | Y | 消费日期（激活日期） |  |
| **lastactivatedate** | datetime | Y | 最近一次激活日期 |  |
| **receivermailaddress** | varchar(128) | Y | 接收者邮箱地址 |  |
| **receiverphonenum** | varchar(16) | Y | 接收者手机号 |  |
| **sensitivewords** | varchar(128) | Y | 敏感词 |  |

## 验证流程



上图解释了短信验证流程，client通过短信发送Key和DeviceID/PhoneNumber给server，如果该Key有效（未被使用过），则通过短信发回验证结果（OK/NG），OK情况下则算是激活，可以继续设置接收邮箱等信息。

## 电文

## Client给Server发送的短信格式

Text = “Auth,” + Key +”,” + LANG + “,” + DeviceID + “,” + PhoneNumber + ,”+PhoneModel +”,”+AndroidVersion

其中LANG =CN|EN|JP，代表Client的语言

例：Auth, 8B122A1DD9, CN,13B789A23CE9125,13980065966,HTC Desire,Android 2.3.1

注意：可能存在无法取得DeviceID和PhoneNumber的情况，这时候可能会是

Auth,8B122A1DD9, , ,HTC Desire,Android 2.3.1

## Server给Client回复的短信格式

Text = Auth,” + ActivationKey + “,”+ <Result> + “,” + <Error Message>

其中Result = OK/NG, ErrorMessage 为NG情况下的出错消息，与Client电文中的LANG对应，比如在CN的情况下”该注册码已被使用”，EN的情况下”This key has already been used.”

## 超级注册码的注册电文

由于超级注册码可以在无限台机器上安装且不需server短信认证，所以仅需要将其信息发送到server保存即可，其短信格式为：

Text = “Super,” + Key +”,” + LANG + “,” + DeviceID + “,” + PhoneNumber + ,”+PhoneModel +”,”+AndroidVersion

可见其格式仅是header的不同。

## 关于获得手机号码的补充说明

对于某些号码（比如移动的动感地带和电信校园卡），是无法用下面的代码从SIM/UIM卡上获得手机号的（关于该事项可参考<http://www.eoeandroid.com/thread-6714-1-1.html>）。

((TelephonyManager)context.getSystemService(Context.*TELEPHONY\_SERVICE*)).getLine1Number();

因此，当Client发送到Server的短信不包含手机号码时，比如：

Auth, 8B122A1DD9, CN,13B789A23CE9125,,HTC Desire,Android 2.3

此时，可直接在Receiver中通过intent的DisplayOriginatingAddress获得手机号码。

public static String getSmsAddress(Intent intent) {

Bundle bundle = intent.getExtras();

Object messages[] = (Object[]) bundle.get("pdus");

return SmsMessage.createFromPdu((byte[]) messages[0]).getDisplayOriginatingAddress();

}

# 程序实现

采用通常的Android应用开发方式，Android SDK + Java + Eclipse的开发套件。

## 要点1：注册BoardcastReceiver拦截SMS

<!--接受短消息需要注册的receiver，其中的android:name是继承了BroadCastReceiver的类的位置 android:priority="100" 指定了接收器的级别，这里设置的值比系统的值要高，这意味着程序的接受 短消息要早于系统接受短消息，这也就给我们提供了拦截短消息，不让系统发出短消息提示的机会。 -->

<receiver android:name=".SmsBroadCast">

<intent-filter android:priority="100">

<action android:name="android.provider.Telephony.SMS\_RECEIVED" />

</intent-filter>

</receiver>

## 要点2：SQLite数据库的性能和扩展性

在中短期内，注册码少于5w的情形下，数据库压力不大。如果到10w级别上必须考虑扩展性。（TBD）