天珑Android手机指纹轻触返回与快速启动功能软件V1.0

设计说明书

**深圳天珑无线科技有限公司**

目录

[第1章概述 3](#_Toc13875)

[1.1 软件产生的背景 3](#_Toc17087)

[1.2 软件功能的要求 3](#_Toc29715)

[第2章流程图 4](#_Toc20509)

[2.1 指纹快速启动设置流程 4](#_Toc19741)

[2.2 设置指纹快速启动功能截图 5](#_Toc16184)

[2.3 轻触返回功能流程 8](#_Toc6213)

[第3章 实现过程 10](#_Toc32497)

[3.1 指纹快速启动设置列表实现 10](#_Toc13735)

[3.2 应用程序数据与联系人数据获取 10](#_Toc20682)

[3.3 快速启动 11](#_Toc834)

[3.4 轻触返回 12](#_Toc23331)

[第4章 总结 14](#_Toc21714)

# 第1章概述

## 1.1 软件产生的背景

目前Android系统带有指纹传感器的手机越来越多。而手机用户一般解除手机屏幕锁后紧接着就是打开APP，如果手机指纹解锁后自动能启动APP，则能给用户带来极大的方面。另外，不管是前置指纹还是后置指纹。指纹传感器的位置都是极易方面用户单手操作的位置。而市场上很多手机的返回键要么是在最左面，要么是在最右边。单手操作都对部分人来说都不便利。而如果指纹有返回功能，也将极大的方便用户操作。于是天珑Android手机指纹轻触返回与快速启动功能软件应运而生。

## 1.2 软件功能的要求

天珑Android手机指纹轻触返回与快速启动功能软件设计要求如下:

1. 当用户开启轻触返回功能开关后，用户轻触指纹，实现与按back键相同的返回功能。
2. 当当用户关闭轻触返回功能开关时后，用户轻触指纹将不会有返回上一界面的功能。
3. 当用户录入指纹后，可以点击去设置指纹快速启动功能，启动功能包括指纹解锁后立即启动指定APP和指纹解锁后拨打指定联系人的电话。
4. 用户可以将设置过快速启动的指纹设置成仅仅具有解锁屏幕的功能。

# 第2章流程图

## 2.1 指纹快速启动设置流程

下图是程序流程图：

开始

手机当前已录入指纹

N

Y

不允许点击进入快速启动设置

点击进入快速启动设置

指纹解锁

点击列表指纹

呼叫联系人

打开应用

显示指纹列表

设置结束

本程序流程图描叙了基本工作流程。在指纹主界面判断系统是否已经录入指纹。如果没有录入，快速启动入口不允许用户点击进入。当用户已经录入过指纹后点击快速启动，进入快速启动指纹列表。列表中显示每个指纹的所绑定快速启动功能。点击指纹列表可以弹出对话框，可以选择设置打开应用、呼叫联系人和解锁。

如果选择打开应用，则可以进入选择应用界面去选择用户需要快速启动的应用。如果选择呼叫联系人，则进入联系人列表。用户可以选择需要快速拨号的联系人。如果用户选择解锁。则本指纹将不具有快速启动APP和快速拨号的功能。

## 2.2 设置指纹快速启动功能截图

（1）当用户没有录入指纹时，快速启动显示灰色，用户无法点击，如图1。当手机中有指纹后，快速启动菜单设置成可点击状态，如图2。



图1 图2

1. 点击快速启动按钮进入快速启动指纹列表界面。界面显示当前每一个指纹对应的快速启动功能，如图3。当指纹绑定的功能为解锁时，图标显示锁的样式。如果指纹绑定的是应用程序，则图标显示应用程序的小图标。如果绑定是快速拨号，图标显示的是联系人的图像。当用户没有设置联系人图形时使用默认图标。

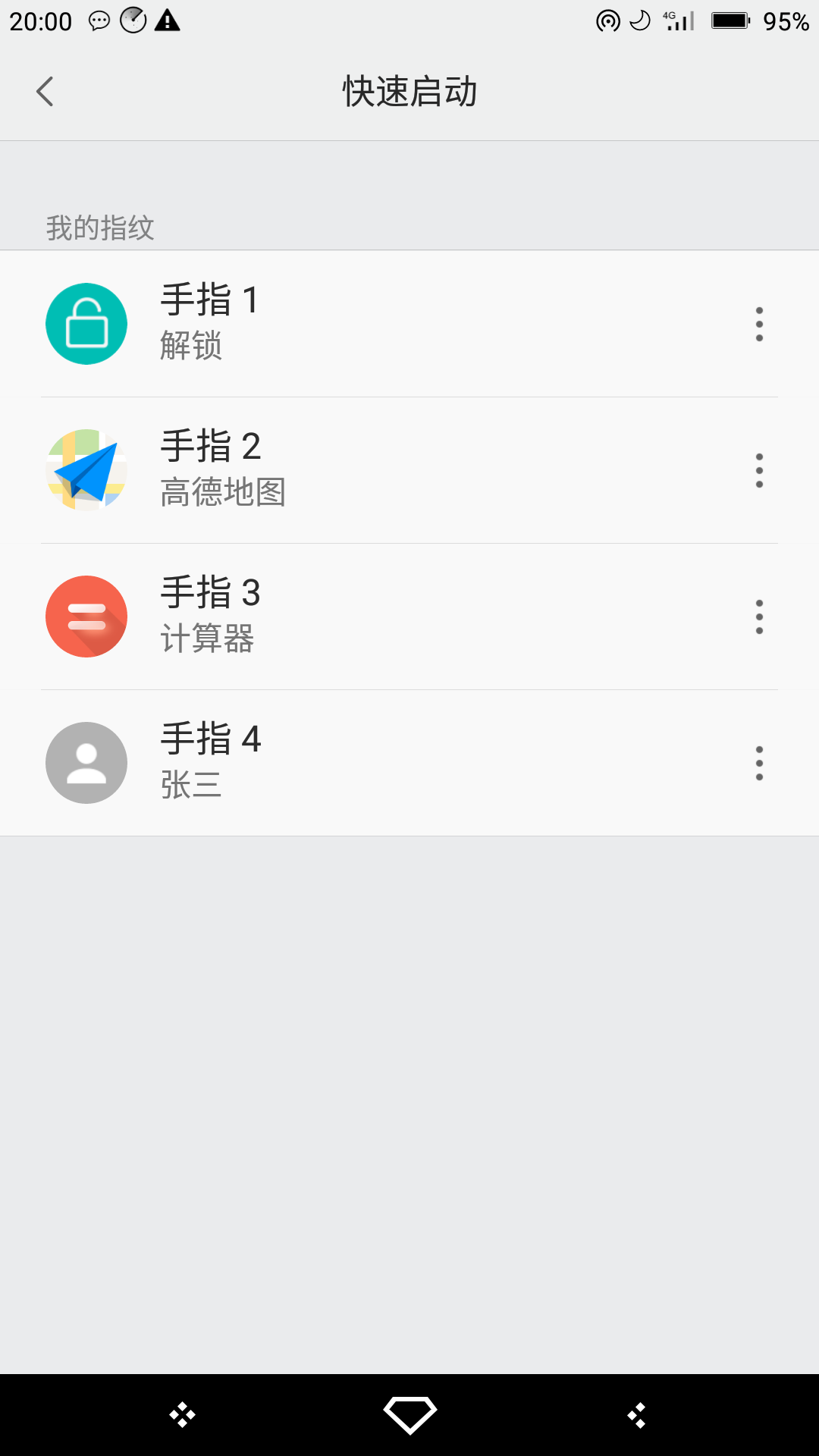


图3

（3）当点击指纹列表是弹出对话框，让用户可以设置此指纹绑定的功能，如图4。当用户点击打开应用，则会进入应用列表界面。如果用户点击呼叫联系人，则会进入联系人列表界面。当点击解锁时，直接设置当前指纹为解锁功能。



图4

1. 点击进入应用列表界面，如图5。应用安装名称分类排序。当用户点击某个应用程序时，此应用将会被设置成指纹的快速启动应用。



图5

1. 点击进入呼叫联系人界面，如图6。联系人按名字成顺序分类排列。当点击某个联系人后，此联系人将被设置成指纹快速拨号联系人。当用户有此指纹解锁手机后，手机会自动拨打此联系的号码。



图6

## 2.3 轻触返回功能流程

指纹轻触返回与快速启动功能软件开机启动service。在sercice中初始化指纹触摸管理器。指纹触摸管理器开启一个线程循环调用poll方法，当监听驱动得到指纹按下和抬起消息后，读取指纹按下或抬起的消息类型，通过注册的回调函数传递给上层。上层接收到按下或抬起事件后首先判断是否打开了轻触返回功能的开关。如果没有打开，则直接返回不去处理。如果打开了，再去判断手机当前状态是否响应轻触返回。如果是锁屏或者录入指纹界面或者在通话状态则不去响应轻触返回。否则，程序向系统发出返回事件。系统收到事件后执行返回功能。

下图是程序流程图：

开始

启动service

初始化指纹触摸Manager

获取指纹触摸按下和抬起事件

轻触返回开关已经打开

N

Y

非指纹录入和锁屏状态

N

Y

按下抬起事件间隔小于300毫秒

N

Y

向系统发出返回事件

结束

# 实现过程

## 3.1 指纹快速启动设置列表实现

在主界面获取当前指纹个数，如果个数大于0，则允许用户点击快速启动，否则不允许代码如下：

private int getFingerprintCount() {

FingerprintManagerEM fpm = new FingerprintManagerEM(mContext);

final List<Fingerprint> items = fpm.getEnrolledFingerprints();

final int fingerprintCount = items != null ? items.size() : 0;

return fingerprintCount;

}

private void updateData() {

fingerCount = getFingerprintCount();

if(mPreferenceQuickActions != null) {

mPreferenceQuickActions.setEnabled(fingerCount > 0 ? true : false);

}

当用户已经录入指纹后，点击快速启动项，启动指纹列表界面。代码如下：

Intent intent = new Intent();

intent.addFlags(Intent.FLAG\_ACTIVITY\_SINGLE\_TOP |

Intent.FLAG\_ACTIVITY\_REORDER\_TO\_FRONT );

intent.setClassName(getPackageName(),

QuickActionFingerListActivity.class.getName());

startActivity(intent);

## 3.2 应用程序数据与联系人数据获取

快速启动应用程序，首先需要回去手机中所有的应用程序信息。其代码如下：

public static void getAllApps(Context context) {

final Intent intent = new Intent(Intent.ACTION\_MAIN, null);

intent.addCategory(Intent.CATEGORY\_LAUNCHER);

List<ResolveInfo> shortcuts = context.getPackageManager()

.queryIntentActivities(intent, 0);

for (int i = 0; i < shortcuts.size(); i++) {

ResolveInfo s = shortcuts.get(i);

AppData tempData = new AppData();

tempData.SetAppName(s.loadLabel(context.getPackageManager())

.toString());

tempData.SetPackageName(s.activityInfo.packageName);

tempData.SetClassName(s.activityInfo.name);

tempData.setAppIcon(s.loadIcon(context.getPackageManager()));

Collections.sort(((AppDataListChoiceActivity) context).getListData(),

mAppDataCompartor);

}

首先创建一个特殊intent，通过它查询出所有应用程序，然后循环将应用程序的包名、类名、应用名称、以及icon保存在一个列表中，再排序。

联系人信息列表界面需要显示每一个联系人的名称、联系人头像、电话号码。所以代码如下：

ContentResolver resolver = mContext.getContentResolver();

Cursor phoneCursor = resolver.query(Phone.CONTENT\_URI,PHONES\_PROJECTION,

null, null, ContactsContract.CommonDataKinds.Phone.SORT\_KEY\_PRIMARY);

if (phoneCursor != null) {

while (phoneCursor.moveToNext()) {

String phoneNumber = phoneCursor.getString(PHONES\_NUMBER\_INDEX);

if (TextUtils.isEmpty(phoneNumber))continue;

String contactName =

phoneCursor.getString(PHONES\_DISPLAY\_NAME\_INDEX);

String sortLetters = setSortLetters(contactName);

....

## 3.3 快速启动

在手机锁屏后，用指纹解锁屏幕时，当指纹识传感器识别到指纹验证正确后会回调函数,代码如下：

public void onAuthenticationSucceeded(AuthenticationResult result) {

int fingerid = resultEM.getFingerprint().getFingerId();

if(isScreenON(mContext)){

mHandler.sendMessage(mHandler.obtainMessage(

MessageType.TINNO\_MSG\_VERIFY\_SUCCESS, MessageData));

}

在onAuthenticationSucceeded函数中首先获取当前验证的指纹是哪个指纹。然后通过Handler发送验证成功的消息给到EncryptService中，并携带指纹id数据。EncryptService接收到TINNO\_MSG\_VERIFY\_SUCCESS消息后调用startQuickBootFunc(fingerid)函数判断当前指纹有没有绑定快速启动。代码如下：

public boolean startQuickBootFunc(int fingerId)

{

{

final QuickBootData quickBootData =

mFingerPrintDataUtils.getQuickBootDataFromFingerData(fingerId);

if(!quickBootData.isAppEmpty()){

className = quickBootData.getClassName();

packageName = quickBootData.getPackageName();

startAppFromQuickBoot(packageName, className);

return true;

}

if(!quickBootData.isPhoneEmpty()){

phoneNumber = quickBootData.getPhoneNumber();

makeCallFromQuickBoot(phoneNumber);

return true;

}

return false;

}

如果当前指纹绑定了快速启动的应用程序，则通过应用程序的包名和类名启动它，并唤醒系统，解除屏幕锁。如果没有，则去判断是否有设置快速拨号。如果有，则唤醒系统，解除屏幕锁，并发起拨打电话。

## 3.4 轻触返回

手机开机后自动启动EncryptService，在onCreate方法中调用初始化fpEventManager，并调用start函数进行native接口的初始化。代码如下：

public boolean start(int s) {

if(s == 0) {

start();

}

else {

Fp\_native fn = new Fp\_native();

fn.init(1, fpEventManager.this);

}

return true;

}

Fp\_native调用init接口中会往native中注册一个回调函数。当有轻触事件时，则调用该函数将事件传递到java层。init代码如下：

public static boolean init(int s, OnReportListerner l) {

if(native\_init(s) < 0) {

return false;

}

Fp\_native fn = new Fp\_native();

fn.native\_setObj(fn);

fn.setOnReportListerner(l);

return true;

}

在native\_init是个native方法。此函数会创建一个线程，在线程中用poll方式对指纹驱动层进行事件的监听。当手指触摸指纹，不管是放下还是抬起手指，指纹传感器都会在驱动中上事件。代码如下:

void \*thread\_do\_poll(void \* arg)

{

struct pollfd pfd;

int ret = -1;

pfd.fd = \_dev\_fd ;

pfd.events = POLLIN;

while (1){

ret = poll(&pfd, 1, -1);

if (ret == 0){

LOGD(TAG "-->[%s], ret == 0.\n",\_\_func\_\_);

}else{

sem\_post(&wk\_sem);

}

}

}

当捕获到指纹按下抬起事件后goodix\_update\_finger\_state函数会通过ioctl的方式读取指纹的状态是按下还是抬起。然后调用fn.setOnReportListerner注册的回调函数来将事件传递到fpEventManager中的OnReport函数中，OnReport函数代码如下：

public boolean OnReport(int e) {

Log.i(TAG, "OnReport:" + e);

mEh.fp\_handleKeyEvent(e);

return false;

}

在fp\_handleKeyEvent函数中判断当前轻触返回开关是否打开。如果没有打开，这直接返回不处理。否则继续判断是否在录入指纹或锁屏状态，如果是此两种状态，则同样不处理。当满足条件后调用sendDownAndUpKeyEvents函数。此函数记录在指纹down事件时记录当前时间。当指纹up事件产生时计算两事件的时间差。如果时间差小于300毫秒，则通过simulateKeyStroke函数向系统下发返回键事件。simulateKeyStroke函数代码如下：

private void simulateKeyStroke(final int KeyCode) {

new Thread(new Runnable() {

@Override

public void run() {

try {

Instrumentation in = new Instrumentation();

in.sendKeyDownUpSync(KeyCode);

} catch (Exception e) {

Log.e(TAG, "simulateKeyStroke:"+e);

}

}

}).start();

}

# 总结

天珑Android手机指纹轻触返回与快速启动功能软件界面友好，操作简单，将指纹传感器功能扩展优化，设计新颖，并且功能稳定，极大的提高了用户体验。此外本软件用极简介的代码实现了本功能，代码复用率高，兼容性好，软件独立，方便在各android系统移植。综合来说本软件设计新颖方便实用，是一款不可多得的好软件。