

上海交通大学

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

THESIS OF BACHELOR



论文题目： 基于感知的
Android 认证系统方案研究

学生姓名： 陈哲扬

学生学号： 5100309406

专 业： 计算机科学与技术

指导教师： 曹珍富、朱浩瑾

学院(系)： 电子信息与电气工程学院

基于感知的 Android 认证系统方案研究

摘要

近期几年以来,拥有先进与健壮的功能甚至能替代传统的大型计算机,智能手机已逐步成为许多用户的新型个人计算机。意味着智能手机也能存放了用户的各种个人信息,其中占多数为保密及有较高敏感度的信息。因此智能手机也成为很多攻击者的目标之一。

目前,安卓智能手机默认都能提供基于触摸屏使用的 Password/PIN/Pattern 认证系统作为个人手机的安全保密系统,但该认证方式仍然存在缺陷。目前两种攻击方式,肩窥攻击与污点攻击都可容易通过观察用户的解锁操作或触摸屏上的污点来获取密码。因此,使用传统 Password/PIN/Pattern 认证系统的智能手机也难以防止此攻击模型。该问题导致智能手机用户的个人信息泄露概率不断增加,成为隐私信息泄露的危机。

找出一种能对以上的攻击方式有抵抗力,尽管攻击者通过某种方式获得用户密码也无法通过手机的认证是本题目的目的。此论文将介绍签名认证系统,一种基于图像识别方式的感知认证系统方案。结合用户输入的密码内容与属性作为该系统的输入密码,使每用户都拥有属于个人的唯一密码。通过实验该系统在安卓智能手机平台上对攻击模型有较好的保密性,能降低智能手机被破锁的概率。

关键词: 安卓、智能手机、感知认证系统、肩窥攻击、污点攻击、签名认证系统

PERCEPTIVE ANDROID AUTHENTICATION SYSTEM RESEARCH

ABSTRACT

Nowadays, along with the advanced features and well-balanced abilities between performance and convenience, smartphones have become the new type of personal computer devices for a lot of users. In this era, smartphones such as Android phones are capable of storing and synchronizing personal information from the Internet inside them, which includes the sensitive ones. This makes Android smartphones become the major target for attackers to steal users' private information for bad purposes.

At the moment, most of the Android smartphones provide the Password/PIN/Pattern systems as the default security authentication. Such systems still have many weaknesses in protecting users' privacy. They are easily vulnerable to a lot of smartphone attack models such as shoulder surfing attack and smudge attack. With these two types of attack, attackers can easily through observing the users' input action or smudges left on the phones to find out the key that they use to unlock their phones. This makes smartphone security a serious problem, especially on perceptive authentication field. Hence, in this paper, SAllock, a graphical identification-based signature authentication method is introduced to counter these attack models. SAllock combines what users input and how users input as the password to unlock the phones. With this system, the variety of passwords will extend and become unique for every user. This solution can increase the safety of users' phones since the password is more difficult to memorize and imitate than standard passwords. Experiments prove that system at the movement can reduce the rate of smartphones being hacked in compared to Password/PIN/Pattern systems quite well.

Key words: Android, smartphone, perceptive authentication, shoulder surfing attack, smudge attack, signature authentication system

目 录

第一章 绪论.....	1
1.1 研究背景.....	1
1.1.1 安卓智能手机概况.....	1
1.1.2 安卓智能手机信息安全调查.....	1
1.1.3 基于感知的认证系统.....	1
1.2 国内外研究现状.....	2
1.3 研究内容和意义.....	2
1.4 论文组织结构.....	3
第二章 攻击方式分析.....	4
2.1 肩窥攻击 (shoulder surfing attacks)	4
2.2 污点攻击 (smudge attacks)	4
2.3 其他攻击方式.....	7
2.4 本章小结.....	8
第三章 现行的感知认证系统分析.....	9
3.1 Password/PIN/Pattern 认证系统.....	9
3.2 QR 码认证系统.....	10
3.3 指纹认证系统.....	11
3.4 人脸识别系统.....	12
3.5 语音认证系统.....	12
3.6 本章小结.....	13
第四章 签名认证系统设计.....	14
4.1 背景.....	14
4.2 需求分析.....	14
4.2.1 可靠性.....	14
4.2.2 可用性.....	14
4.2.3 独立性.....	14
4.2.4 唯一性.....	14
4.2.5 方便性.....	15
4.2.6 可扩展性.....	15
4.3 系统结构.....	15
4.4 笔画划分.....	16
4.5 图像对比方法.....	17
4.5.1 像素结构.....	17
4.5.2 像素结构对比.....	18
4.6 Base64 码.....	20
4.7 签名信息记录.....	21
4.8 签名样本管理.....	24
4.9 本章小结.....	24
第五章 签名认证系统实现.....	25

5.1 基础技术.....	25
5.1.1 HTML 与 CSS.....	25
5.1.2 JavaScript 与 jQuery.....	25
5.1.3 SQL.....	25
5.1.4 PhoneGap.....	25
5.2 硬件设备、软件与环境.....	26
5.2.1 硬件设备.....	26
5.2.2 软件.....	26
5.2.3 环境.....	26
5.3 系统框架构建.....	26
5.4 签名输入区.....	28
5.4.1 引入签名工具.....	28
5.4.2 存放签名数据.....	29
5.4.3 笔画的起始节点与结束节点.....	29
5.4.4 笔画对象导出.....	30
5.4.5 签名区域的干扰.....	31
5.5 笔画划分实现.....	32
5.6 图像识别实现.....	34
5.6.1 Resemble 的比较法.....	34
5.6.2 对比工具调用.....	37
5.6.3 对比状态处理.....	37
5.6.4 引入对比模块.....	39
5.7 数据库实现.....	40
5.7.1 数据库构建.....	40
5.7.2 写入签名属性.....	42
5.7.3 写入笔画属性.....	43
5.7.4 写入验证错误信息.....	45
5.7.5 数据库内容显示.....	48
5.8 文件管理实现.....	48
5.8.1 PhoneGap 的 File API 安装及使用.....	48
5.8.2 文件管理操作.....	49
5.8.3 Base64 码至二进制码转换.....	51
5.8.4 引入签名认证系统的记录过程.....	52
5.9 样本文件管理.....	53
5.9.1 获取签名和笔画信息.....	53
5.9.2 选择签名样本作为密码.....	54
5.10 系统封装.....	56
5.11 本章小结.....	56
第六章 签名认证系统样本展示.....	57
6.1 样本系统模块.....	57
6.1.1 记录.....	57
6.1.2 开锁.....	57
6.1.3 设置.....	57
6.2 样本系统展示.....	57

6.3 本章小结.....	61
第七章 系统认证实验与评估.....	62
7.1 系统实验.....	62
7.2 系统实验结果.....	62
7.2.1 针对用户本人的实验.....	62
7.2.2 针对肩窥攻击的实验.....	63
7.2.3 针对污点攻击的实验.....	63
7.3 系统评估.....	64
7.4 将来与展望.....	64
7.5 本章小结.....	65
第八章 结论.....	66
参考文献.....	67
谢辞.....	70

第一章 绪论

1.1 研究背景

1.1.1 安卓智能手机概况

为了与在 2007 年出现而首次打破典型手机概念的 iPhone 竞争, 在 2008 年底谷歌公司推出了的安卓, 而目前已成为最广用的智能手机操作系统。安卓是一种基于 Linux 内核的操作系统, 该系统的应用在一台虚拟机上运行, 而系统组件是使用 Java、C、C++ 和 XML 程序设计语言实现^[1]。一个原型的安卓系统能支持一些传统手机不可有的特性如邮件查询, GPS 定位, 账户同步化等等。随着系统及应用发展速度, 安卓智能手机已逐步成为不少用户的新一代小型计算机, 可担任与解决许多日常生活的需求而无需使用到传统计算机。由于采取开源方式, 安卓平台收到许多手机制造公司如小米、HTC、三星等等的支持和许多智能手机用户的信任^[2]。也因为这原因, 安卓系统的应用数量不断的增加, 类型也变得越来越丰富。与苹果市场不同, 安卓市场为自由市场, 意味着任何人都可自由开发和发布自己的安卓应用^[3]。广泛的应用市场, 支持多功能且方便使用的系统导致大多数安卓智能手机都存放了不少用户的个人敏感信息。这样也同时引起了隐私安全问题, 许多攻击模型已产生为了恶意地获取手机用户的个人信息。因此安卓的隐私信息安全已经成为近期的热门研究话题^[4]。

1.1.2 安卓智能手机信息安全调查

近期几年除了典型计算机平台以外, 属于智能手机的信息安全问题也逐步变成受许多关注的主题中之一^[5]。原因在于安卓手机的用户在手机上存放个人隐私信息的比率越来越高。在 2012 年加利福尼亚大学的调查中, 60% 的智能手机用户给出他们愿意存储但不愿意在手机上使用自己的个人信息^[6]。调查对象也表达他们对智能手机的隐私泄露问题比典型的计算机有更高的关心态度。和台式机、笔记本设备对比、经常随身的微小智能手机更容易收到敏感信息盗窃者的攻击^[7]。

属于物理类型的攻击如手机盗窃在 2011 年是对智能手机造成最大的威胁^[8]。由于不少手机用户存储许多个人敏感信息, 丢失自己的智能手机是造成泄露个人信息最容易发生的情况。在美国的高人口城市中, 经历过移动设备丢失会被盗窃的用户达 52%。根据一个安卓智能手机信息安全调查中, 由 Symantec 公司进行, 在北美若有 50 个智能手机被丢失在路上, 已经有 96% 的人捡到手机后访问手机内部信息、86% 查看手机用户的个人信息、83% 查看合作信息、60% 访问个人邮箱及社会网帐号、50% 运行远程管理员、43% 访问网上银行帐号^[9]。该数据体现出丢失手机时泄露个人的隐私信息有多严重。在 2012 年美国的其他调查中智能手机丢失后的找回率只有 7%, 意味着当丢失后若手机被他人访问, 许多敏感信息长久被泄露的概率是非常高。

1.1.3 基于感知的认证系统

通过传感器来接受从用户方传送过来的输入密码, 再经各种不同的识别方式来验证用户所输入的密码是否符合要求称为基于感知的认证系统。该认证方式在现代的智能手机中属于最普及的认证方式中之一。安卓智能手机可通过屏幕、麦克风、像头、速度传感器等设备来感知并与用户进行交互。从信息安全方面考虑, 通过感知方法作为验证可使用的密码类型可分成三类: 知识类、标记类及生物类^[10]。

知识类密码, 也称为文字型密码, 属于多种认证系统中最常用的密码类型。该密码类型

由于容易使用及更新而受到广泛的使用。最主要的问题存在于知识类密码的认证系统所验证的不是用户本身的所有权,而是该密码的所有权^[1]。这也意味着得到密码的人则可通过系统的认证,该认证机制可容易受密码盗窃者的攻击。知识类密码还存在记忆问题,安全性更高的密码会更难地记住。智能手机最常用的认证方式为 Password/PIN/Pattern,所接受的密码属于该类型。而用户一般是通过手机屏幕来输入密码,因此经常会在触摸屏上留下指纹的污点。攻击者可通过观察用户输入密码的行为或输入设备上留下的痕迹等方式来获得密码。本题将提到的肩窥攻击与污点攻击方式也属于该攻击类型的其中之一。

标记类密码也部分类似于知识类密码,但可解决用户对密码的记忆问题。用户向认证系统系统先分配好属于自己的标记。标记可有多种不同类型,如物理钥匙、RFID 射频识别标记、行用卡等等^[12]。标记类密码的认证方式可部分减低,但不能完全阻止标记盗窃的问题。攻击者得到用户所分配给系统的标记后仍然能非法通过系统的认证。可看出问题还是存在于该类型的认证只验证属于第三种事物的所有权而非针对用户本身拥有的独特特性。智能手机对标记类密码的验证一般是通过像头设备来识别出用户的标记。攻击者很容易想手机摄像头展示非法得到的标记来通过认证。

生物类密码,一种基于生物识别技术而产生的密码,可视为目前对密码盗窃者有最好抵抗力的密码类型。与前面两者不同,基于生物类密码的生物特征身份认证系统验证的是属于用户本身独一的特点及属性而非通过使用第三种事物作为密码^[13]。该类型密码可以为用户的指纹、语音、人脸、体形、个人习惯等等。生物类密码的最大优点为他人可观察但不可拥有或很难模仿。生物特征身份认证可以认为是目前有最高安全性的认证方法。目前该认证类型所存在的最大问题在于还未达到成熟的程度,许多现行的生物类认证系统识别率还未到达符合要求,有较高的第一型及第二型错误导致使用不方便^[14]。且有部分识别方式的密码可被模仿来非法通过验证。因此目前此认证系统收到很多公众的关注,许多开发者也以该认证方式作为自己研究及开发认证系统的方向。安卓智能手机提供各种传感器如触摸屏、摄像头、麦克风等可良好地与此认证方式配合,形成基于感知的安卓生物特征身份认证系统。

1.2 国内外研究现状

中国国内近期几年关于手机安全状况的报告大多数属于应用相关的隐私信息泄露方面。基于物理类型造成的信息安全报告还占少数,所研究的安全系统主要是针对系统软件方面^[15]。经调查也发现中国大部分手机用户仍然使用默认提供的 Password/PIN/Pattern 作为个人手机的认证方式,也有部分用户完全不设置任何密码。肩窥攻击与污点攻击也由该原因而对手机用户的个人信息造成更加严重的危害。

在国外由于智能手机的认证系统安全问题引起许多新鲜的研究方向。研究的主要的目的为找出适当并且能替代低保密效果的认证系统^[16]。新的认证方案大多数属于使用生物类密码的认证系统如人脸、语音、指纹识别等。为了防止肩窥攻击、污点攻击等攻击方式,许多自开发的感知认证系统应用已产生,带来给智能手机用户很多新鲜的认证方式选择^[17]。目前部分生物类的感知认证系统的识别效果还未达到符合的要求,因此带来给用户不方便的使用体验。可看出基于使用生物类密码的认证系统是目前最可靠且最有研究价值的方向。创造新的生物类认证方法或提高现行系统的认证效果是许多安卓平台信息安全研究者的目标。

1.3 研究内容和意义

针对本题目的研究内容为找出一种适当及新鲜的基于感知的认证系统方案。目前市场上几乎所有智能手机都能提供 Password/PIN/Pattern 作为手机的默认认证系统^[18]。该认证方式因属于知识类,无属于每个用户的独特属性所以很密码很容易猜测。经过研究 Password/PIN/Pattern 认证方式的缺陷及攻击者的攻击方式,找出能应付此攻击方式的认证

方案。研究内容也包括分析其他现行的感知认证系统来找出它们的优点及缺点。从此认证方案得出的优缺点很有作为研究自己的认证方案的参考价值,使本题目所研究的系统最后能够达到符合的需求。介绍设计思路及实现如何能使手机用户有属于自己的唯一性密码来验证自己的身份,且密码很难模仿或记住是该课题的最终目的。当样本系统完成后还需要通过多次实验来验证该系统的准确率,并从获取到的实验数据评估系统的可行性、可用性及未来的展望。

题目的研究方向对安卓平台的隐私信息安全有很大的意义。原因由于感知认证系统属于认证系统种类中最常用的认证方式之一。应用的领域很现实,如基于智能手机的屏幕开锁,一种在日常使用手机时经常及以高频率使用到的功能。研究出能够替代传统 Password/PIN/Pattern 认证方式可改变安卓手机用户对手机安全的看法。大众的手机安全认证可进一步变得更加灵活及健壮。

1.4 论文组织结构

本论文由八章组成。每张的内容体现本题目的研究过程及研究成果。

第一章介绍安卓平台系统,也是基于本题目所研究对象的运作平台。从安卓操作系统及安卓智能手机的普遍性展现出维护隐私信息安全的重要性,尤其是最常用的感知认证系统。本章节也给出目前国内与国外对该题目内容的研究现状,从而向本题目确定研究方向与研究意义。

第二章展示出目前基于感知的安卓认证系统存在的问题,也就是目前对该认证系统的攻击方式。具体内容为分析攻击者如何通过这些攻击方式破解传统的 Password/PIN/Pattern 认证系统来恶意地获取手机内的个人信息。主要的两种攻击方式是肩窥攻击和污点攻击。分析攻击模型的运作方式可清晰表示该研究内容的重要性及意义性,并且辅助研究者找出适当的认证方式来抵抗此攻击。

第三章将介绍目前的 Password/PIN/Pattern 及替代它的现行感知认证系统。分析其他感知认证系统对攻击模型的解决方式及它们本身的优缺点可让本题目的研究方向变得更稳固。

第四章介绍针对本题目内容而设计的基于签名识别的认证系统。通过对该系统的各方面需求而提出为了满足此需求所使用到的技术来组成该认证系统。该章节也给出系统的构建思路,整体框架结构及运作方式。

第五章进入签名认证系统的具体实现过程。以第四章的理论为准,描述通过使用程序设计语言实现系统的各技术及模块。从安装软件,设置环境至封装系统最终样本工作过程都有具体记录,代码展示及代码解释。

第六章以安卓智能手机应用的角度下展示样本系统。通过介绍应用的实际功能及使用,可表现出该系统的真实可行性及使用性。

第七章的内容为签名认证系统的实验过程。经过担任本手机用户与攻击者的实验者,对安装在实验手机上进行多次认证。得出的实验结果可作为基准来评估本系统的效率性,并同时找出系统在未来的展望,可升级性和可扩展性。

第八章为结论,给本题目做总结并表达研究心得。

第二章 攻击方式分析

2.1 肩窥攻击 (shoulder surfing attacks)

典型的知识类 Password 密码理论上有相当好的保密机制,但实际由于方便使用的目的,许多用户选择短且较容易记住的密码,从而影响到该密码类型的安全性。因此 Password 密码可容易被通过猜测、词典攻击、key-logger 等攻击获取^[19]。为了解决方便性和保密性的问题,基于图形的密码类型已出现。图形密码可以防止许多对文字型密码有效的攻击方式,但也很容易受到肩窥攻击的影响^[20]。



图 2-1 肩窥攻击方式

所谓的肩窥攻击,是攻击者通过直接或间接观察用户输入密码过程来获得密码信息^[21]。尽管该攻击方式的实现形式简单,无高难度的技术含量但目前仍然为许多认证系统最难以解决的危机中之一。经过调查发现图形密码是最容易受到该攻击方式的影响。一般,能应付肩窥攻击的图形密码认证系统需要较长的认证时间,因此不受到广泛用户的使用^[22]。由于智能手机的使用特征及目的与传统计算机不同,许多用户为了避免麻烦且花费多时间的认证而选择了较简单的图形密码。不少部分的密码为容易记住且输入的,该性质使肩窥攻击者更容易获得密码信息。

2.2 污点攻击 (smudge attacks)

智能手机最明显的特征是无需多种不同的输入设备如键盘、鼠标。所有用户与设备的交互将通过触摸屏来解决。经常在触摸屏上操作会难以避免留下指纹的痕迹,也称为污点。有较高使用频率且标准性如 PIN、Pattern 密码会留下很明显及突出的污点^[23]。基于污点出现的最大问题有三种:污点保留在触摸品上持久性,用户难以通过擦拭或存放手机至口袋等操作来清除留下的污点,污点可容易通过目前许多设备如摄像机、计算机等来进行信息获取和分析。

属于污点攻击方式的攻击有两种:被动与主动。被动攻击者是不能接触到用户手机的情况下,只能通过远距离使用相机等设备记录下触摸屏的状态,从而猜测出用户的密码。从以上性质,被动攻击者只可控制记录时的像头角度,而无法控制照亮环境的参数。相反,主动攻击者是拥有获得用户手机设备的能力。这样主动攻击者可随意通过调整合适的像头角度,

照亮环境等来的出最好的密码提取效果。

进行该攻击方式的模拟实验中，假设所以攻击形式都由主动攻击者执行。记录污点需要考虑的三种参数：触摸屏与污点的反光属性、位置与灯光环境的状态、像头针对手机屏幕的角度。屏幕接触到光线时会反射或扩散。反射出的光线只能通过相反的角度来观察，而扩散光线可在任何位置观察到。智能手机的触摸品属性落于两者之间。光源将反映出触摸屏上的图像。光源对屏幕的角度将取决屏幕表面的亮区与阴区，而光源大小决定照亮的范围。使用相机和光源来找出适当提取污点的角度设置如图 2-2 所示。

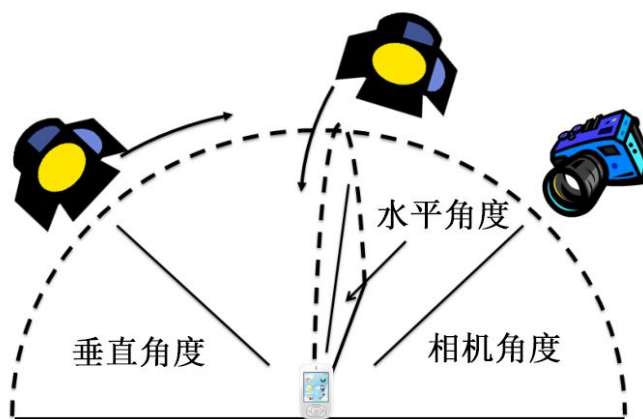


图 2-2 获取污点的像头与灯光环境设置

从宾夕法尼亚大学提供的实验信息中，从使用摄像机和灯光环境找出污点的形状来获取密码的情况下得出的成功概率相当高^[24]。实验场合包括三种：理想情况、配合手机正常操作情况、污点经过处理情况。理想情况是攻击者可容易在手机触摸屏最清晰的状态下从污点提取密码。配合手机正常操作情况可在分成两种场合：用户输入密码再对手机触摸屏上进行各种不同的滑动操作，另外场合同样情况但顺序相反。污点经过处理情况意味着手机触摸屏已经经过摩擦或用户的擦拭操作等。

在以上的三种情况下，针对手机触摸屏上不同的方向，使用摄像机及灯光照亮环境的不同角度提取出留下的污点形状。经过多次使用不同的角度，得出相机角度为 60 度则能达到最好的污点提取效率，而相机角度与光源垂坐标角度均为 15 得出的图像效果为最差。在理想情况，用户在屏幕上滑动密码轻重力度不同而导致能获取完整密码的概率也不一致。受轻滑动的屏幕成功获取密码概率较低，相反受重话动力的屏幕获取密码相当容易。该情况果显示 92% 的密码能通过污点可抽取及有 68% 是完全可获取。在最差的实验环境下也有 34% 能抽取和 17% 是完全可获取。配合手机操作场合中，如输入密码后继续通过触摸屏使用手机，被输入过的密码会由于其他污点的干扰而显得不大清晰，影响提取密码的概率。相反若在许多出现的干扰污点后再次输入密码，留下的污点会相当清楚及容易猜测出密码。在污点经过处理的情况下，实验将考虑的污点处理方式有屏幕与口袋的摩擦及用户对屏幕的擦拭操作。可看出经过摩擦的触摸屏只能获得部分信息，如污点的形状但无方向的信息，对手动的擦拭操作也有同样结果。图 2-3 至图 2-5 分别对应理想情况、手机操作干扰出现情况、污点经过处理情况下获得的触摸屏污点。



图 2-3 理想情况下获得的污点



图 2-4 在配上使用手机出现干扰情况下获得的污点



图 2-5 触摸屏经过手动擦拭或摩擦后获得的污点

由于 Pattern 密码的连续输入性及不广泛范围，污点攻击的影响效果相当严重。尽管从污点获得的密码信息不完整，通过推测各节点间的链接关系，攻击者可容易推出完整 Pattern 密码的形状从而对手机进行破锁。

2.3 其他攻击方式

对智能手机的传统认证系统，除了肩窥攻击与污点攻击，攻击者还可以通过其他方式如侦听、猜测、交流等来获得密码的内容。其中一种不能直接获取密码内容，但可以作为其他攻击方式的前者攻击，为物理型攻击。最普及的物理型攻击为手机盗窃。手机盗窃者得到用户的手机后可通过猜测方式来破解密码或以污点攻击的主动攻击者形式来获取密码内容。因此物理攻击也是造成智能手机用户个人信息泄漏危机中相当有危害性的攻击方式。



图 2-6 智能手机被盗窃的危机

2.4 本章小结

本章介绍了最普及可对智能手机的基于感知的安卓认证系统造成危害的攻击模型。通过分析肩窥攻击、污点攻击及物理攻击的执行方式及严重程度，可清晰地体现出本课题研究内容需要解决的问题和寻找新认证方法的目的。理解攻击模型还可帮助于设计新认证系统方案的思想，使构造出的系统能有效的应付此攻击模型。

第三章 现行的感知认证系统分析

本章节将分析一些目前现行感知认证系统的属性及基本运作方式。每种系统都有属于自己的独特认证方法及优缺点。

3.1 Password/PIN/Pattern 认证系统

目前大多数智能手机的基于触摸屏的认证系统都以 Password/ PIN/ Pattern 方式为默认，因此该认证方式也为目前最普遍的安卓手机认证系统。所输入的密码属于知识类密码，可分成三种类型：Password 密码，PIN 号码和 Pattern 图案。每种类型都有自己的优点及缺点。

Password 密码在需要认证时会打开安卓系统的键盘，该键盘可输入的字符基本上与传统计算机的键盘相同。这也表示在智能手机上使用 Password 形式的密码和在计算机上使用传统密码是相同的。Password 密码的优点在于密码内容本身的丰富性，不容易猜测。用户可自由使用各种不同字符来组成不同长度的密码。整体上若在计算机上使用会有相当好的保密效果，但从智能手机的角度来考虑会出现一定程度的问题。可看出最明显的问题是非方便性，使用手机的系统键盘输入长且复杂的密码会导致输入的时间变长，这也会增加受肩窥攻击的可能性。

PIN 号码是使用典型的手机拨打键盘作为输入工具，也就是从 0 至 9 的号码键盘。一般的 PIN 号码只有 4 位，因此输入方便及简单可视为 PIN 号码认证方式的最大优点，但同时也是 PIN 号码最不安全的地方。由于只采用 0 至 9 号码为输入内容且密码位只有 4 位，内容很容易猜测。输入的号码键盘大小也比 Password 密码的键盘大，因此留下的指印污点会相当明显^[25]。PIN 号码证方式非常容易受到肩窥攻击和污点攻击的影响。

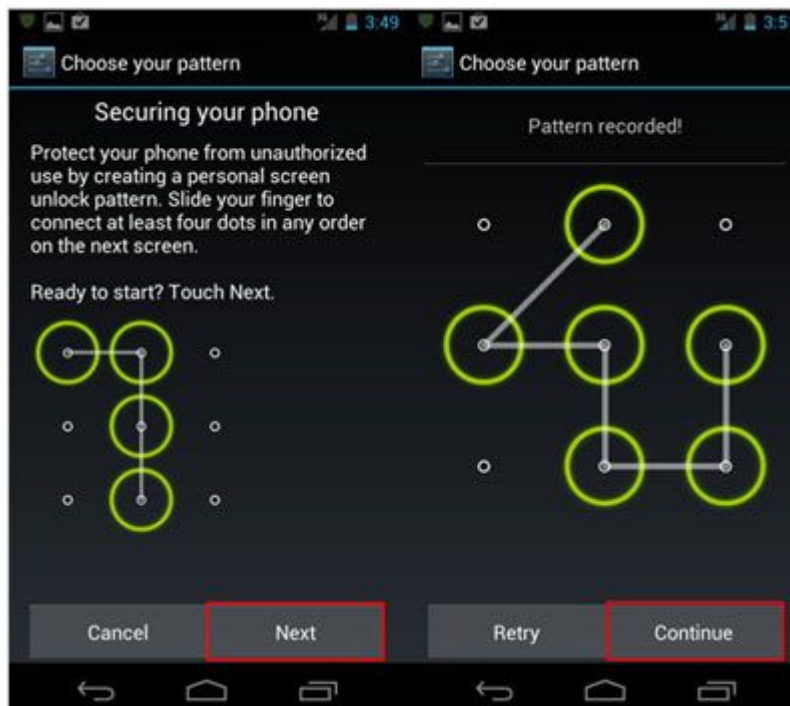


图 3-1 典型的智能手机 Pattern 图案密码

Pattern 属于图案类型的密码。输入密码的方式是通过画出触摸屏上与样本图案相同的

节点路径，记录 Pattern 图案过程如图 3-1 所示。Pattern 与 PIN 号码认证方式有相同的缺点。由于图案中的开始的节点只可能是屏幕上 9 个节点中之一，且已经出现在路径的节点不可再出现第二次，因此密码的丰富性很低^[26]。密码属于图案类型的特点也使 Pattern 密码很有形象性，很容易记住，从而受到肩窥攻击。在手机触摸屏上留下滑动节点的路径痕迹也容易成为污点攻击的目标。

3.2 QR 码认证系统

QR 码，也称为 Quick Response 码，是一种基于矩阵的二维条形码。QR 码拥有快速的解码速度，且最高有 30% 的数据在读取错误时能及时回复。QR 码主要的结构包含可识别的位置、时间标记和数据区，整体结构如图 3-2 所示。由于 QR 码拥有的优点，韩国崇实大学的一个研究项目中采取 QR 码作为一种新的认证方式^[27]。

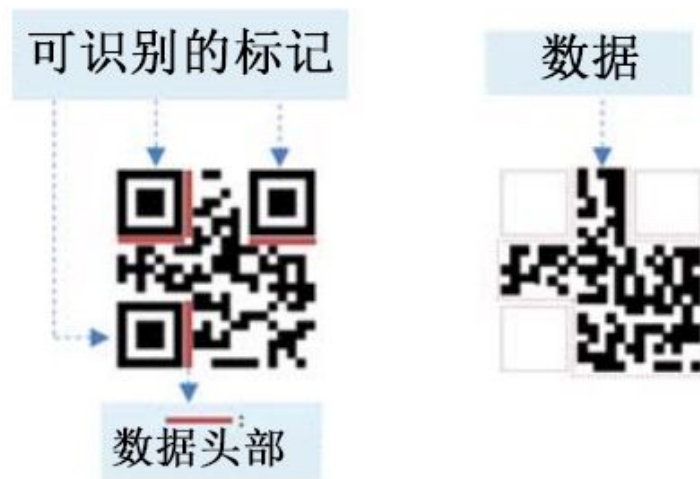


图 3-2 QR 码的结构

使用 QR 码认证方式，需要满足以下几种必要条件：

- 使用的手机设备必备摄像头
- 用户的设备需先向使用的系统注册
- 认证应用已安装在手机内
- 用户的手机可支持 SHA-1 hash 函数和 AES 算法来处理数据

基于 QR 码的验证步骤可分成四个过程：

1. 用户向认证系统发访问权限请求。
2. 认证系统获取申请者的信息，在生成 QR 码并返回给用户。
3. 用户用已经注册的设备扫描由系统提供的 QR 码，进行验证。
4. 认证系统对该设备进行验证并给出结果。

该认证方法主要使用两种协议：QR 码生成协议与认证协议。

关于 QR 码生成协议，系统先从注册的用户获取信息，存放于 CI。

$$CI: H \quad (\text{用户信息}) \quad (3-1)$$

再针对申请者生成随机数 RSN，使用 AES 算法计算出密钥，以 RSN 和 CI 的 XOR 关系来加密：

$$Enc_Value: E_KEY(H(RSN \oplus CI)) \quad (3-2)$$

最后使用 ENC_Value、RSN、TS、CI 来生成 QR 码

$$\text{QR code: ENC_Value} \parallel \text{RSN} \parallel \text{TS} \parallel \text{CI} \quad (3-3)$$

在认证协议中, 设备先扫描用户的 QR 码, 查看信息是否配对。通过连接 RSN、TS、PN 构成 RRC (Request RSN Confirm) 检查系统是否成功生成 QR 码, 若操作成功会生成 RRS (Response RSN Confirm Server)。在把 RRS、TS 和密钥发至设备:

$$\text{RRS: H(RSN} \oplus \text{MEI} \oplus \text{Key)} \quad (3-4)$$

设备经处理 PN 和 ID 来验证 RRS 并生成 MEI。把 RRS 与 $\text{H(RSN} \oplus \text{MEI} \oplus \text{Key)}$ 对比。通过验证 RSN, 设备会产生一个 CI 并同时与 RSN 进行计算来检测 QR 码中用户信息的准确性。再把 $\text{H(RSN} \oplus \text{CI)}$ 与 D Key (Enc_Value) 做一个对比。当认证过程结束后设备将构造一个 UID (User Identity Data), 和 RSN、PN、TS 同时发给系统

$$\text{UID: H(MEI} \oplus \text{PW} \oplus \text{TS} \oplus \text{RSN)} \quad (3-5)$$

此认证方式通过实验发现能相当有效地应付密码猜测, 中间人攻击, 抓包攻击等攻击方式。存在的问题是认证过程需要使用到手机的摄像头, 且由于认证过程针对设备本身的依赖性相当高, 因此对设备盗窃的物理攻击有较弱的抵抗力。

3.3 指纹认证系统

指纹属于生物类的密码, 以指纹作为认证密码是生物类认证系统目前相当普及和广用的类型。如图 3-3, 目前拥有指纹认证系统的设备都需要依赖于其他指纹扫描器, 从而造成使用中许多不方便。安卓智能手机的现行指纹认证使用 Home 按键作为指纹的感知器。根据休斯顿大学的计算机学院研究中, 将原始需要使用按钮作为指纹扫描器的感知系统移至智能手机的屏幕上是相当可行的方法, 能获得一定概率的感知效果^[28]。

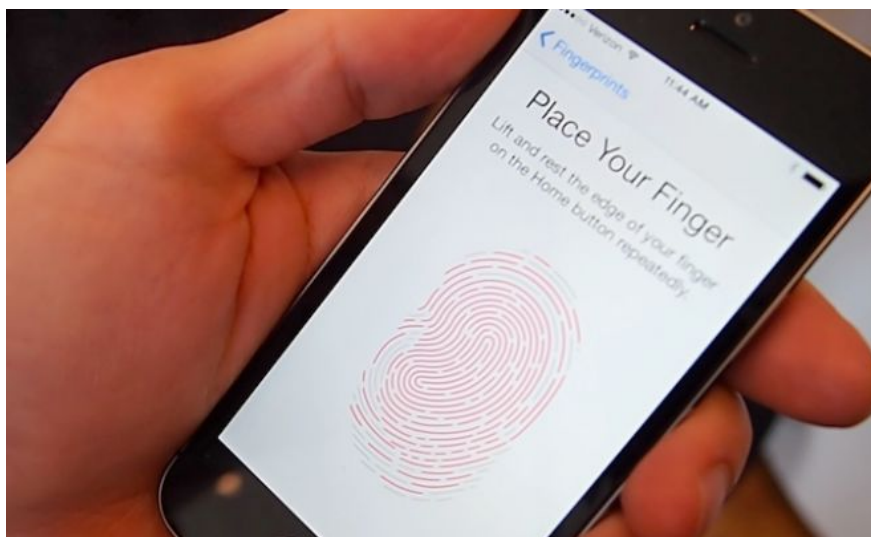


图 3-3 指纹认证系统

由于每人的指纹都是唯一, 因此用指纹作为认证密码是一种良好的思想。使用指纹可识别出用户的唯一身份, 防止他人对用户手机做出的攻击。但实际上从智能手机使用者的角度来看, 该认证方式仍然未收到大众的关注及使用。原因出现于认证机制还未达到最准确的程度, 从而影响使用体验。部分指纹认证应用已经得到多数用户的评论, 其中出现用户本人不能通过认证, 手机永久被锁住或认证时间过长。很多用户表达拒绝使用指纹作为自己智能手机的认证密码。针对安卓智能手机的指纹认证还在发展的早期阶段, 因而需要提高认证效率才能较好地应用。

3.4 人脸识别系统

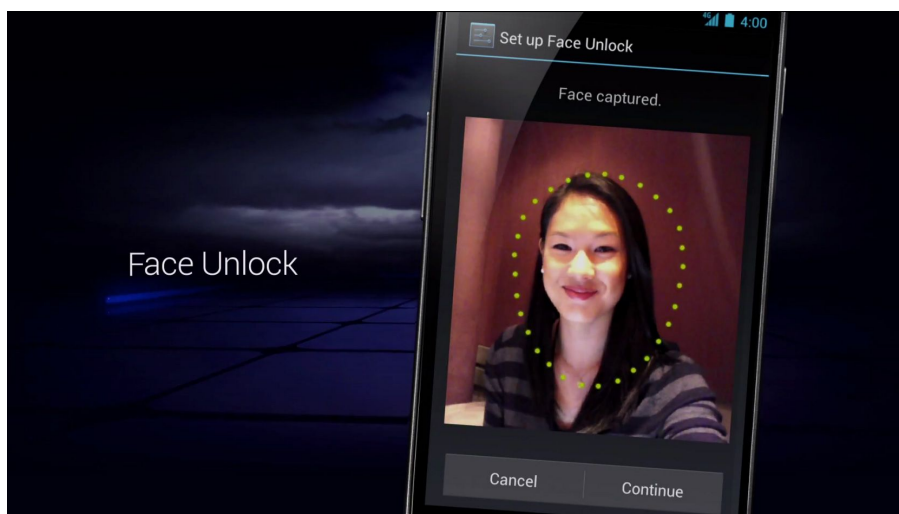


图 3-4 人脸识别认证系统

与指纹认证类似，以人脸作为生物类的密码来通过验证是人脸识别方式的特征。在智能手机中使用来接受人脸数据的设备为手机摄像头^[29]。理论上该认证方法的密码也是唯一性，但在实际用场会受到不少干扰。IBM T.J. Watson 研究中心进行的一个实验中，使用过人脸识别系统的用户返回使用不方便或认证不准确的使用体验^[30]。由于需要打开摄像头来获取人脸图像，因此获取到的信息很依赖于周围环境的照亮状态。智能手机的屏幕放光性质可造成人脸信息获取有误，环境亮度低也能导致同样的问题。每次使用需要根据不同当前环境而准确地将摄像头移动使系统能正确的获得人脸信息也是该系统的问题之一，导致使用不方便。

3.5 语音认证系统

通过语音来输入密码称为语音认证系统。原则上该认证方式也是通过输入已设置好的知识类密码来进行验证，区别在于输入的方法。不使用原理啊的输入设备如键盘，用户想设备麦克风使用语音说出密码来通过验证。



图 3-5 语音输入认证系统

也由 IBM T.J. Watson 研究中心执行的一个实验中，多数使用过语音密码认证的用户均表现低效率的使用体验^[30]。用语音输入的密码比起原来的手动输入密码更加难以记住。另外，在任何时候开锁手机需要说出密码将成为密码盗窃者的最大目标，且在不少情况下输入密码时会受到周围环境的声音而造成干扰。输入语音密码还有一个缺点是需要使用合理的音速，音量才能输入正确的密码。从语音认证方法的特征及通过实验得出的结果，该方式的认证过程时间为最长且有最低的保密性。

3.6 本章小结

本章介绍了目前许多研究中或已出现在安卓市场上的感知认证系统。经分析此认证方案可对其他研究者与开发者对安全认证方面的思想及看法有更深一步的理解。调查现行的认证系统，从而得出它们的运作方式，优点及缺点也是本题目的研究目的之一。了解此信息后在设计自己的认证方案时能有更多可考虑的方面，使系统能够补充其他系统还存在的问题。

第四章 签名认证系统设计

4.1 背景

从第二章分析可对感知认证系统造成危害的攻击模式,第三章理解现行感知认证系统的运作方式及特点,可推出需要构建的系统需要解决的主要问题。在所提到其他认证系统方案之中,多种仍然存在较高受到攻击的概率或不方便使用的性质。将问题具体化可分成三种:输入密码操作过度简单或过度麻烦、受外围环境的干扰、密码可容易被仿造。

选择的方案需要解决以上三种问题,于是本题目考虑使用签名认证方式。该方式能有较好的应付肩膀观察攻击与污点攻击,且能解决系统的使用方便性。

4.2 需求分析

4.2.1 可靠性

系统的可靠性体现在于系统能够准确的识别出本次输入签名的人是手机拥有者本人而非他人的。该元素也是系统的核心技术和最有关键性的特征。根据目前的研究现状,系统能百分之百地识别出用户签名的习惯仍然未实现到。但可考虑使用某种识别机制使系统能尽量有最低的第一型及第二型错误的概率。

4.2.2 可用性

构造出的系统必须满足对隐私信息安全有不同要求的用户。因此系统需要有灵活性,能根据用户不同的需求来做出适当的认证。系统应允许用户能管理并随意更换签名的对比样本,并且还能够记录使用过程的信息,包括错误信息及提供汇报统计功能。同时与锁住手机模块相关的功能设置如锁住时间长度,允许错误的次数等也是很重。从以上的需求分析,该认证系统的设置模块是不可缺少的部分。

4.2.3 独立性

由于所设计的认证系统属于基于安卓系统的感知类型,因此无依赖性,或独立性是非常重要的。Password/PIN/Pattern、指纹认证或其他认证方案都有共同特征是能独立地运作,无需连接网络或需要使用主机端来完成认证任务。另外当系统认证时也不必连接外围设备。这些认证系统的该设计思想相当准确,原因在于感知认证系统对智能手机平台的普遍性。只要考虑手机外层的触摸屏开锁方面,手机用户已经可在任何时间,任何地方开锁自己的手机。何时,何地区保证都能使用网络是不现实的,且在认证通过之前需要使用手机的链接网络功能意味着已经访问了手机的内部权限。该元素将会引起系统的信息安全问题。所以,本系统必为独立运作式系统。无论是签名样本文件存储、样本存储还是签名比较模块都需要在手机内局部地运作。

4.2.4 唯一性

系统的唯一性需求包括两方面:属于用户签名习惯的唯一性和属于同一个用户各签名及笔画间的唯一性。第一,每个手机用户的签名习惯都不同,因此能够记录属于用户独特的签名习惯是很重要。这也相当于用户的签名习惯不仅仅从签名的内容本身,而还从用户签名的方式体现出。此特征将决定该系统的可用性及安全性,同时也是设计本系统的难点之一。第二,用户可能同时会存储多个签名样本,且经常切换作为本次样本来进行验证。由于签名是非属于数字型,文字型密码,因此从多个签名及笔画间区分出每个元素是不可缺少的功能。实现到该特征不仅使系统的认证准确率及效率提高,且用户也容易管理自己的签名样本集。

4.2.5 方便性

若当认证的时候需要依赖于其他周围环境会对用户造成许多不方便的使用体验。如第三章所提到,使用手机摄像头来识别人脸或麦克风来做语音识别很多时候需要依赖外围环境的状态。因此这些认证方式不能保证在何时何地每次认证都能达到最高的效率。高的第一型及第二型错误会影响到认证系统的使用方便性和安全性。解决该问题的最好方案是减少系统对其他外围环境的依赖性。本题目的研究方向考虑把用户与设备间的互动只限制于通过触摸屏来进行验证。该互动方式和传统的 Password/PIN/Pattern 一样,但应用新的验证方式可仍然保证系统的安全性及效率性。

4.2.6 可扩展性

可扩展性也包括两方面:性能扩展和平台扩展。系统的开发过程中需要将每个功能模块划分清楚。各模块可独立操作且互相调用形成系统的完整运作流程。这样在后期工作可容易优化或添加新功能而不影响到整个系统的运作及结构。本系统属于基于安卓智能手机平台的认证系统,因此将系统扩展到其他移动设备平台是值得考虑的。使用跨平台的手机应用开发工具使开发系统的源代码可有重用性,从而减低开发其他平台同样应用的工作量。

4.3 系统结构

该认证系统需要满足效率性及灵活性,因而除了最核心的签名认证之外,还需要提供属于签名密码与系统相关的设置。用户开始使用系统时必须做的事情为设置自己的签名样本,使系统能开锁运作。从本次后能通过认证的用户不仅能访问手机内部内容,还可以进入系统的签名管理及设置模块。设置模块包含设置关于系统的锁住功能,系统开关等。签名管理模块包含样本管理与信息管理部分。样本管理允许用户随意添加新签名样本、删除旧样本或替换对比样本。信息管理提供给用户关于签名、笔画与验证错误的相关信息。针对本系统的用例图如图 4-1。

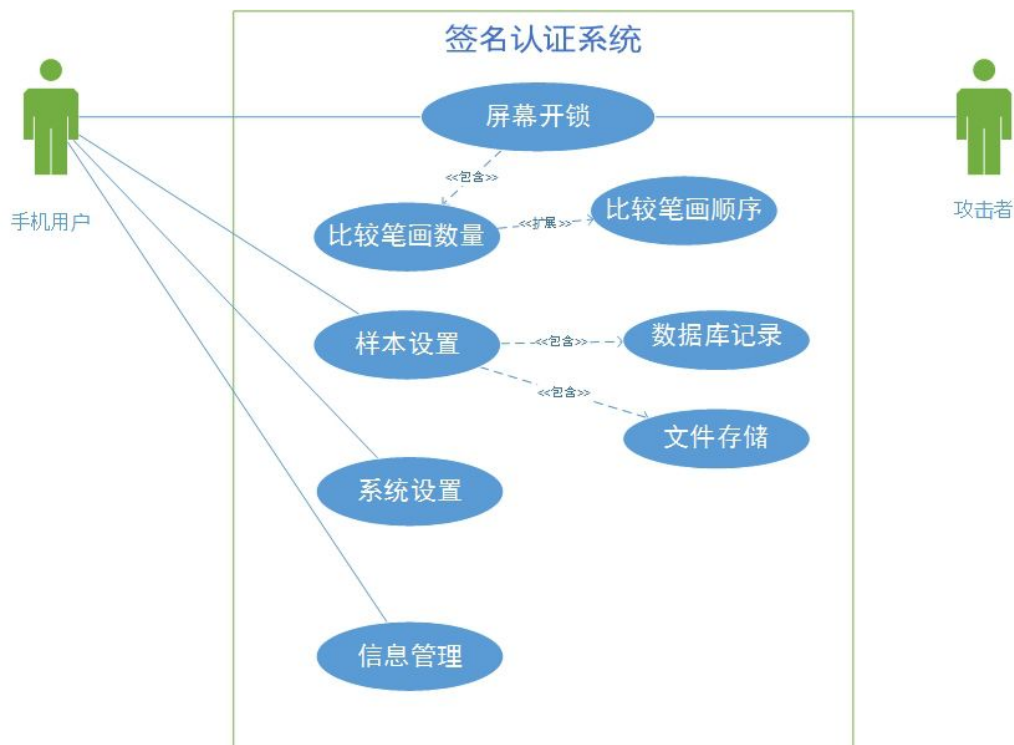


图 4-1 系统用例图

从系统拥有的模块来分析,可把系统结构分成三层:视图层、处理层和数据层。视图层

主要包含系统与用户交互的界面如输入签名区域、验证信息显示、管理样本等界面。多数在视图层的执行将传达给处理层。处理层主要的任务是接受并处理数据，在该系统中包含记录比较和整理数据。属处理层的笔画划分模块从签名区获得签名数据后将签名划分成若干个笔画，然后根据不同的使用场合讲划分后的数据传给图像比较模块或数据成。图像比较部分从笔画划分部分得到笔画数据后同时从数据层的样本文件集中取出样本数据，再执行对比工作。数据层主要有签名文件与数据库中的签名信息两部分。两者之间有密切的关系，信息的同步性必须一致，也等同于当样本文件出现数据的变动，数据库也要跟着更新自己的信息，相反也同理。数据层主要的工作是接受并存储信息与提供数据给对比过程和信息查看功能。整体的系统结构在从图 4-2 中给出。

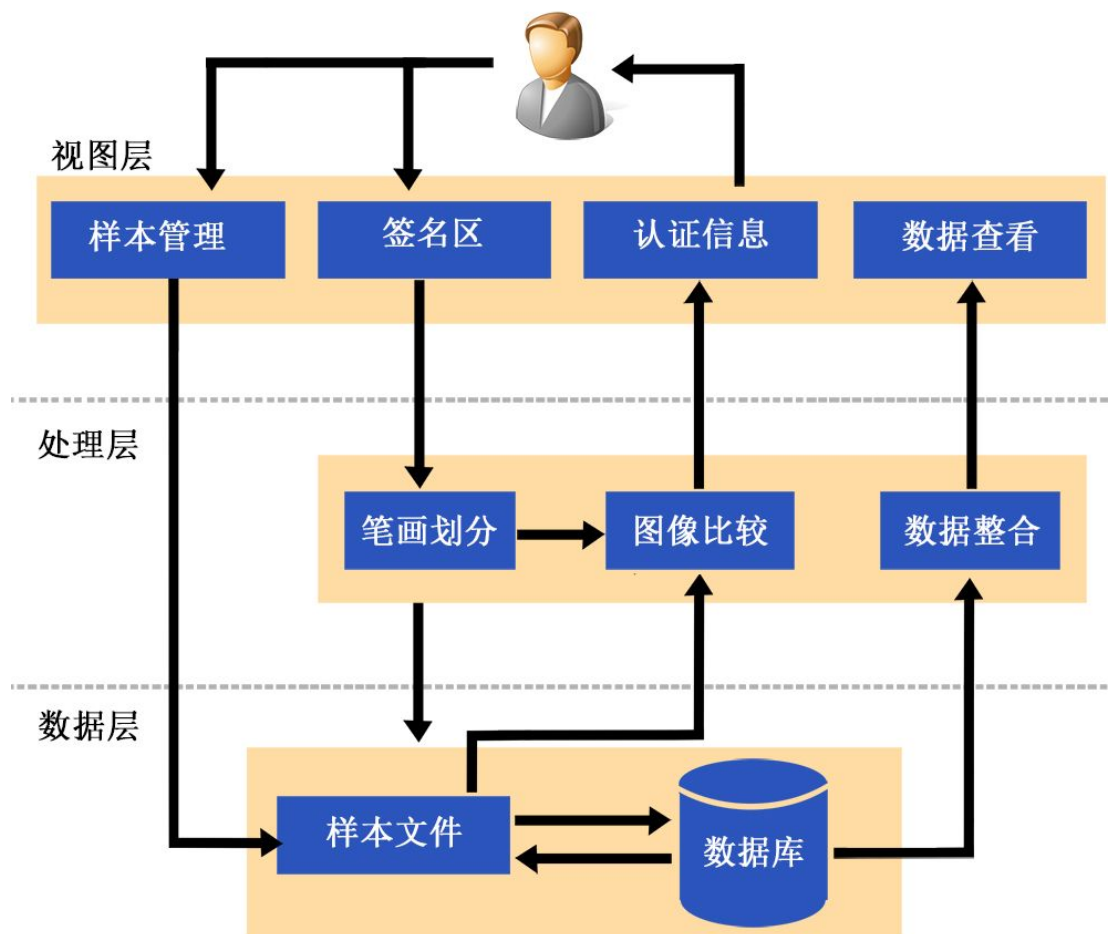


图 4-2 系统及工作流程

4.4 笔画划分

与一般的图像识别方法不同，签名认证系统不仅仅识别用户所输入的签名图案的内容，而还识别出用户输入该图案的方式。也以图像比较方法作为识别机制，但系统所对比的对象不是用户输入的整个签名，而组成签名的每个笔画。系统事先把输入的签名划分成若干个笔画，笔画的单元等于用户在屏幕上不离手的状态下签名的一个笔画。系统在对每个输入笔画与样本做对比之前需先检查用户输入的笔画数量。如笔画数量与样本的笔画数量不匹配则验证导出失败结果。该认证流程可减少系统的工作量，无需在异常情况下做笔画对比过程。比较笔画数量的想法虽然简单，容易实现但对增加该系统的安全性及认证效率有很大的意义。划分笔画来进行对比也等同于检查所输入签名的笔画顺序，从而体现出用户的签名习惯。针

对同样的一个签名内容，不同用户会有不同的签名习惯，导致构成整个签名的笔画数量和笔画顺序也不同。引入对比笔画数量特征可大大减低该系统受到污点攻击的概率。针对肩窥攻击，该签名密码形式也比传统的 Password/ PIN/ Pattern 密码难以记住及模仿。

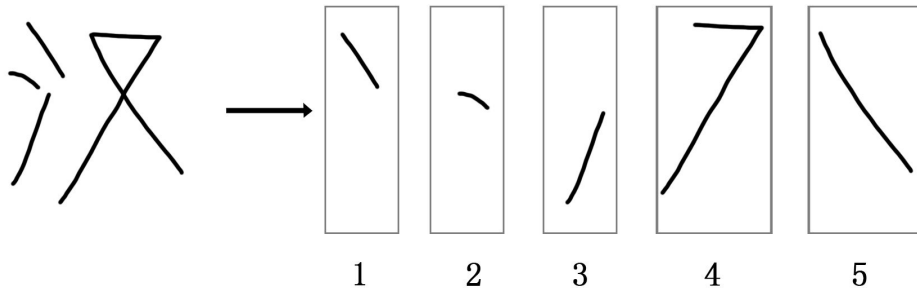


图 4-3 将签名划分成若干个笔画

本系统的签名区在接受签名时把签名及笔画对象存放在自己的数据栈中。笔画划分模块需要访问该数据栈，从而对数据栈中每个笔画进行 Base64 数据获取操作并存至自定义的数组中，该数组则为数据库记录，文件保存等操作的对象。实现思路可用以下伪代码表现：

```

Begin
  设 数组 A = 数据栈的数据;
  设 B = 新数组;
  For (i := 0 to A 的数组长度)
  {
    设 C = A[i];
    把 A[i] 推出签名区;
    设 D = A[i] 的数据;

    // 把 D 推进 B 数组
    B.push(D);
  }
End
  
```

最后得出是包含所有属于输入签名的笔画数组 B。每个笔画是一个对象，存放它本身的数据如 Base64 码，持续时间等等。

4.5 图像对比方法

4.5.1 像素结构

本系统使用的图像对方法是以属于该图像的像素为操作对象。一个像素的结构是由不同的彩色值组成。颜色值可分成四种：R（红）、G（绿）、B（蓝）和 A（亮度）。从程序设计的数据结构来看，可把每个像素看成一个数组有四个元素。从第 0 至第 3 个元素按序为 R、G、B、A，四个元素的数值范围为 0 至 255。各像素与属性之间的数据结构可从图 4-4 中看出。

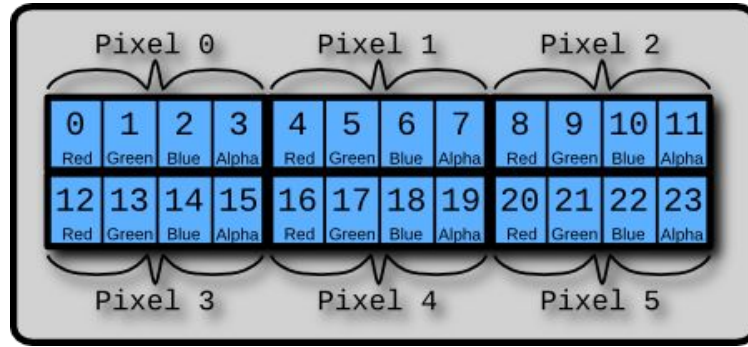


图 4-4 像素的数据结构

从一个像素跳转到下一个像素需要一个偏移量。该偏移量的计算方法为：

$$offset = (y \times w + x) \times 4 \quad (4-1)$$

其中 $offset$ 为像素的偏移量、 y 为像素目前位置的纵坐标、 w 为像素所属的图片的宽度、 x 为目前位置的横坐标。

4.5.2 像素结构对比

图像的对比流程是通过扫描两个目标图像的像素，再计算两者的差别率且导出差别的结果图案^[31]。以比较两个像素结构的 R、G、B、A 值是否相等为方法，若两者的像素颜色对应则跳往下一个像素，否则会生成一个表示错误的像素，且差别率加一。当扫描工作结束后，签名对比模块将生成一幅差别图，由扫描过程中收集到的所有错误像素组成。另外从每笔画对获得的相似度计算出平均相似度。该相似度作为决定验证是否能通过的主要元素。若得出的平均相似度大于所设置的允许相似度，则验证通过，否则失败。



图 4-5 图像对比得出的差别结果图

所谓的笔画相似度原始是从两幅笔画图计算出的差别率。之所以数据的使用法被倒过来是因为针对该算法，签名区的背景也考虑成一种颜色，而签名的笔画像素数量与签名区的像素数量之差很大。因此得出的差别率会与两者之间的差别像素量反比。

原始的差别率计算是通过取生成的差别像素与整幅图的像素之间的比率。公式如下：

$$mismatch = \frac{n_e}{w \times h} \times 100\% \quad (4-2)$$

其中 $mismatch$ 为差别率， n_e 为差别像素数量， w 和 h 分别为图像对象的宽度和高度。系统中计算出最后的签名平均相似率公式如下：

$$avgmatch = \frac{\sum_{i=1}^n mismatch}{n_s} \quad (4-3)$$

Avgmatch 为平均相似度，公式的分子部分是 1 至 n 个属于同一个签名的笔画的差别率总值， n_s 为笔画的数量。

整体的签名笔画集合比较过程可用以下的伪代码来形容。

```
Begin
//从参数获得签名 1 与签名 2 的笔画集
设 A = 签名 1 的笔画数组;
设 B = 签名 2 的笔画数组;

//比较签名的笔画数量
If(A.length != B.length)
Then
{
    //笔画数量不匹配，认证失败
    Unlock = false;
}
Else
{
    //存放差别像素的数组
    设 ErrorPixel = new Array

    //对两个笔画集中的各笔画对进行对比
    For(i=0; i < A.length; i++)
    {
        // 传入当前笔画对象
        设 strokeA = A[i];
        设 strokeB = B[i];

        //扫描两个笔画的像素
        For(j = 0; j < strokeA.lastPixel; j++)
        {
            //若像素彩色相等则返回
            If(strokeA.pixel[j] == strokeB.pixel[j]) then return;
            Else //若彩色不相等
            {
                Mismatch++;

                //生成一个错误像素并进栈
                ErrorPixel.push(createErrorPixel);
            }
        }
    }
}
```

```
//计算第i 个笔画的差别率
设  $MisRate = MisMatch / (strokeA.width * strokeA.height) * 100;$ 

//从差别像素数组构造出差别结果图
设  $DiffImage = createFrom(ErrorPixel);$ 

设  $TotalMatch = TotalMatch + MisRate;$ 
}

//计算平均相似度
 $FinalMatch = TotalMatch / A.length$ 

//从平均相似度是否符合要求给出相应的认证结果
//DefaultMatch 为允许通过的相似率
If ( $FinalMatch > DefaultMatch$ )
Then  $unlock = true$ 
Else  $unlock = false;$ 
}

End
```

4.6 Base64 码

Base64 是一种以 ASCII 格式的字符形式来显示二进制码。Base64 编码及译码技术在许多针对文字型数据处理的系统菜单应用相当广泛。本系统从签名输入区获得的签名信息也以 Base64 码表示。签名对象在与样本签名对比时也已 Base64 格式转发给签名比较模块。在存储签名样本的文件时原始的 Base64 码签名也需要通过 Base64 译码技术来转换成二进制才能存储成图片文件^[32]。

Base64 编码技术不仅有一种，但所有不同译码方法都以前 62 字符为共同的使用字符，包含 A - Z、a - z 和 0 - 9。本系统的签名接受区导出的签名使用 MIME 方式的 Base64 译码，第 63、64 字符为“+”与“/”。表 4-1 是基于 MIME 的 Base64 码索引表^[33]。

表 4-1 Base64 码索引表

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

4.7 签名信息记录

系统的识别模块能够正常运作之前需要有至少一个签名作为样本。因此使用系统时第一工作是需要执行是记录自己的签名样本。系统需要记录的对象有两种：完整签名和该签名中每个笔画元素。系统的记录工作也包含两种：记录至数据库及存储签名的图像文件。

本系统展示给手机用户的是完整的签名，这类型的对象可应用在样本管理方面但不用于对比功能。相反，签名的各笔画类型将不向用户显示，也不用于管理模块但是比较模块最关键的元素。因此在记录签名作为样本的过程中这两种对象都是不可缺少的。

数据库及文件管理系统将担任签名记录过程的所有工作，且两者有很密切的关系。当用户设置属于自己的签名样本时，整个签名和签名的各笔画的属性会同时被记录至数据库和以 PNG 格式文件存至系统的样本管理路径内。

基于数据库类型的数据在记录时针对三种对象：签名、笔画和错误类型。其中，设置属于签名和笔画这两种对象的数据表能保证两者之间有相关联系的关系也是关键的工作之一。由于基于签名的认证系统使用的密码为图像型，密码本身还未有属于自己的唯一性，因此需要手动给签名和笔画设置唯一性的属性，也就是独一的 id。使用签名的 id 和访问到属于该签名的所有笔画，于是在笔画的关系数据表中需要所属的签名 id 属性。不仅这样，能够记录所画出每笔画的持续时间也有很大的意义。目前系统还未考虑各笔画的持续时间作为对比的元素，但在未来的后期工作可利用获取到的这些持续时间作为签名的对比元素这一，从而增强系统的可靠性。签名和笔画都需要获取记录时的系统时间使用户容易观察及管理，签名对象还可以添加自定义名字的属性，从而使该数据表更有人性化。关于错误类型，此对象在认证的时候会使用到。每当用户所输入的签名不符合需求，导致验证失败，系统将记录下错误所属的类型和发生的时间。错误类型数据表能使手机用户每次访问手机能查询出验证出错信息，找出存在的不异常的验证错误从而推出手机是否被他人访问过的可能性。该数据表的数据还可在汇报统计中使用到，从错误类型的数据所画出的曲线表可让用户能更加清晰的观察到在一定时间段内的验证错误情况。从以上的分析，可推出签名、笔画和错误类型数据表拥有的属性：

- 签名：唯一的 id、自定义签名名字、构造日期
- 笔画：唯一的 id、所属的签名 id、持续时间、构造时间
- 错误类型：唯一的 id、错误类型、发生时作为样本的签名 id、发生时间

一个签名有一或多个笔画，可生成若干个错误信息。本系统针对的操作对象的类图如图 4-6 所示。

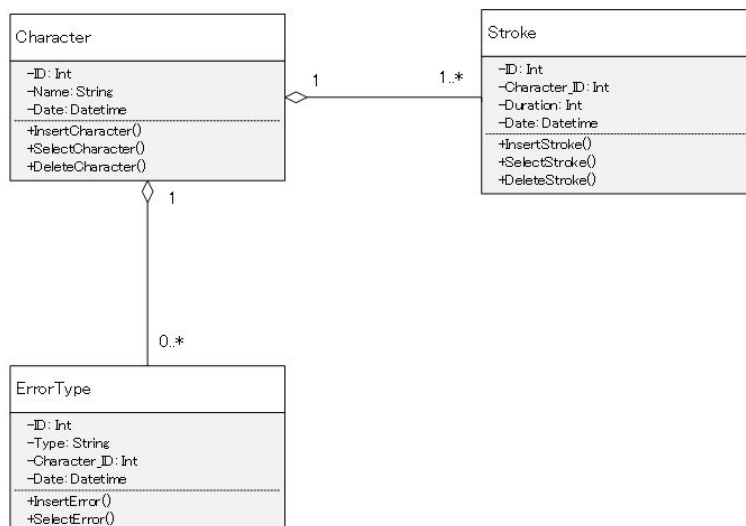


图 4-6 记录工作的操作对象类图

签名记录的第二工作为存储样本的图片文件，针对的对象有签名和笔画。由于 canvas 图像处理器导出的图像对象是 Base64 码格式，而需要存储的目标文件格式为 PNG，因此把原本的 Base64 码译码成二进制码的工作是需要的。从系统的实现方式，需要写入的二进制

码必须存放至 `ArrayBuffer` 形式的对象。因此应用 Base64 译码技术把转换出的二进制码存放至 `ArrayBuffer` 对象，在写入 PNG 文件则存储文件的流程就完成了。给签名样本的文件命名也需要考虑结构的问题。命名方式需要使各样本文件容易管理，因此最优的方法为使用数据库的数据来给样本图像文件命名。签名和笔画的命名方式如下：

签名: <签名 id>.png

笔画: <笔画 id>_<签名中的顺序>.png

其中，签名和笔画文件将分开来存储。签名文件会在记录签名信息时同时存储，而笔画文件会在记录每笔画的信息时同时存储。两种对象存储的地方也不同，由两个目录分别存储。从以上的存储顺序，存储地方和命名格式使样本签名和笔画的存储系统很清晰且有高模块化的性质。整体签名信息记录过程如图 X 所示。

实现笔画的记录伪代码如下：

Begin

构建 *characterDB* 为签名数据表;

构建 *strokeDB* 为笔画数据表;

设 *name* = 签名名字;

设 *date* = 系统当前时间;

插入 *name*、*date* 至 *characterDB* 数据库;

设 *current_id* = *characterDB* 中刚插入的签名 id;

设 *code* = 签名的 Base64 码;

// 把签名存储成 PNG 文件

Function write(current_id.png,code);

// 调用笔画划分功能

Function splitStroke();

// *A* 为存放输入签名的笔画数组

For (i:= 0 to A 的数组长度)

{

设 *duration* = *A[i]* 笔画的持续时间;

设 *code* = *A[i]* 笔画的 Base64 码;

Function write(current_id_i.png,code);

插入 *current_id*、*duration* 至 *strokeDB* 数据库;

i++;

}

End

4.8 签名样本管理

同时支持存储多个密码及能随时替换也是一个认证系统应该能满足到的需求。经常从不同场合而动态地改变自己的密码可使系统的安全性变得更可靠。因此本题目的签名认证系统也提供签名样本管理功能。该功能模块包括样本添加、样本删除及样本选择功能。用户在可访问手机的状态下随意添加新的签名样本或删除旧的签名样本, 更换对比样本每次只能选择一个。样本管理模块是该系统有较好的使用灵活性和方便性, 从该功能使用的用户可解决随时及快速更换密码问题。

样本管理模块中数据库与文件存储系统是两种很重要的角色。两者之间的密切关系将实现数据信息及文件之间的存储、转发、删除。签名认证系统的签名记录和签名对比模块是通过样本文件管理链接起来, 因此该模块的存在性和重要性也相当高。

4.9 本章小结

本章节中已讲解基于签名识别的认证系统的设计思想与方法。从前两章的信息分析可推出需求分析而表达该系统需要满足什么条件, 从而提出能够解决所存在的问题的认证方案。新的认证方案通过所介绍的设计思路可证明能解决其他认证系统的部分问题如对外围环境的依赖性、操作方便性等等。从系统的完整结构为开始点, 本章解释了所有属于系统的主要模块的性质、任务、结构及实现的方法。系统的功能、特点及运作流程通过该章节也有了详细的描述。第四章的内容可作为下一章系统实现工作的前提。

第五章 签名认证系统实现

5.1 基础技术

本系统使用网络应用程序设计语言 HTML、CSS、JavaScript 与 jQuery 来实现系统界面与内部算法。再加上 SQL 为管理关系式数据库语言。与 PhoneGap 框架配合最后可开发出基于智能手机平台的感知认证系统应用。

5.1.1 HTML 与 CSS

HTML 是超文本标记语言，作为本系统的界面设计语言。该语言无需使用编译器，在编写 HTML 文件后可直接使用浏览器预览并测试与使用。使用该语言使系统设计工作容易管理及处理错误。HTML 界面结构也集成了系统的所有逻辑库、设计风格而展示出系统功能与结构。该部分可视为用户与系统之间的交互接口。最新的 HTML5 可支持 canvas 图像处理，该性能是本系统的关键模块之一。

CSS 是串样式列表语言，该语言负责向 HTML 文件提供布局、彩色及字体设置。原始，HTML 即可通过<style></style>标签来存放某界面元素的设计风格。但当文件数过多，需要统一修改或更新时需要对所有 HTML 档案进行修改，该操作将花费很多时间及统一性很低，容易导致错误或不一致性。因此使用 CSS 语言可解决该问题。将整系统的设计风格存放在一个 CSS 列表文件中可使对系统修改或更新变得方便、节省时间且有高统一性与效率性。

5.1.2 JavaScript 与 jQuery

JavaScript 是一种动态脚本语言，经常用于网络应用程序设计。该语言作为本系统所有内置逻辑算法的实现工具。JavaScript 支持基于面向对象程序设计及动态类型转换。有许多开源软件库能提供给 JavaScript 使用，使该语言变的更灵活及高效率。JavaScript 实现的功能将以 HTML 界面作为借口来展示。本系统的签名输入区 canvas、笔画划分、图像识别、数据库与样本管理部分由 JavaScript 实现。

jQuery 是基于 JavaScript 的跨平台软件库。目前 jQuery 为 JavaScript 最广用的开源库。jQuery 可提高 JavaScript 对 HTML 元素的操作及事件处理。该库使编写 JavaScript 代码工作变得更加简洁、容易操作和方便修改。

5.1.3 SQL

SQL 是基于数据库的查询语言。本系统在记录签名、签名管理和汇报管理中需要使用到数据库。SQL 将实现本系统的数据插入，删除及选择操作。对象为签名样本、样本的个笔画及开锁错误信息。

5.1.4 PhoneGap

PhoneGap 是一款开发手机应用的框架^[34]。PhoneGap 支持使用 HTML、CSS 为界面展示，JavaScript 为逻辑算法设计组合来开发手机多平台的应用。该框架提供许多插件可支持数据库，文件系统，手机像头管理等功能。PhoneGap 的最大优点为可转换网络型应用成手机本地应用。因此本题目使用 PhoneGap 为工作环境来封装系统，可使系统以应用形式在安卓智能手机上运行。该过程有很大意义，能针对本题目的研究内容把一个基于安卓的认证系统在手机上运行，使该系统有现实性且认证实验也有更加准确的数据。不仅仅这样，使用 PhoneGap 可使系统的源代码有高重用性，经过修改属于若干手机系统的代码可扩展系统到其他平台如 iOS、BlackBerry、Windows Phone 等。

5.2 硬件设备、软件与环境

5.2.1 硬件设备

签名认证系统将以安卓应用形式在手机上运行。因此除了实现系统源代码的计算机之外，安卓智能手机是不可缺少。本题目使用三星 GT-S7562i 智能手机，运行安卓 4.0.4 系统作为样本系统的测试平台。

5.2.2 软件

实现系统需要使用的开发工具包含：

- Java JDK8 32 位版本：Java 语言开发工具包，该工具必须最初安装才可使后续的软件能正常安装及使用。
- Android SDK：Android 应用程序开发语言包（目前 Android SDK 只支持 32 位的 Java JDK）。该软件安装后需要更新及下载工具（Tools），安卓 API 及附加工具（Extra）才可使用^[35]。
- NodeJS：下载 PhoneGap 框架的工具。
- Apache Ant：一种 Java 库，用于链接及运行基于 Java 的应用程序。
- PhoneGap：开发本系统的框架软件。安装 NodeJS 后使用命令行 `C:\>npm install -g phonegap` 来下载并安装。
- Eclipse：Android 应用程序的编程软件，使用 Eclipse 以 Android Project 形式导入 PhoneGap 构造出的原始应用后可开始对系统进行编程。另外需要安装 Eclipse 的 ADT 插件。该插件包含安卓程序的开发工具包。

5.2.3 环境

安装以上的软件后需要在 Windows 系统的环境变量 PATH 中设置才能正常使用。针对 PhoneGap 而言，直接引入各工具的绝对路径是不可用行。因此为了使 PhoneGap 工作平台能正常运作，需要多定义个工具的代表变量：

```
%JAVA_HOME%: C:\Program Files (x86)\Java\jdk1.8.0
%ANT_HOME%: C:\developement\ant
%ANDROID_HOME%: C:\Users\Administrator\AppData\Local\Android\android-sdk
```

再把以上变量配上合适的路径引入环境变量 PATH。最后的 PATH 变量包含以下路径：

```
C:\Program Files (x86)\nodejs\;
%JAVA_HOME%\bin;
%ANT_HOME%\bin;
%ANDROID_HOME%\platform-tools;
%ANDROID_HOME%\tools
```

该步骤完成后，全部开发工具及环境基本上已经准备好，可接下一步开始构建系统框架。

5.3 系统框架构建

该系统使用 PhoneGap 框架来开发，因此需先使用 PhoneGap 构造一个初始手机应用。该操作通过使用如下图的 PhoneGap 密令行：

```
C:\¥Users¥Administrator>phonegap create salock
[phonegap] the options C:\¥Users¥Administrator¥salock com.phonegap.helloworld Hel
loWorld
[phonegap] created project at C:\¥Users¥Administrator¥salock
```

图 5-1 构造 PhoneGap 的应用

其中 salock 为本系统的应用名，也为该系统的主目录名。下一步，打开应用的项目目录并进行首次链接。

```
C:\Users\Administrator>cd salock

C:\Users\Administrator\salock>phonegap run android
[phonegap] detecting Android SDK environment...
[phonegap] using the local environment
[phonegap] compiling Android...
Buildfile: C:\Users\Administrator\salock\platforms\android\build.xml
```

图 5-2 链接 PhoneGap 的安卓应用

PhoneGap 将对应用的源代码文件进行编译及链接。连接生成的应用可在安卓手机上正常运行。默认原始程序为 HelloWorld 应用。

打开 Eclipse 软件，使用 File/Project 以 Android Project from Existing Code 方式导入 HelloWorld 与 HelloWorld-CordovaLib 两个项目至工作区中。HelloWorld 为本题目的主项目，所有代码编写及系统框架构建工作在该项目进行。HelloWorld-CordovaLib 向 HelloWorld 项目提供所有基于 Cordova 的软件库。该项目与主项目 HelloWorld 有密切关系，取决应用是否能在手机上正常运行，因此为工作区中不可缺少的项目。

HelloWorld 主项目中需要留意的文件目录有：

- src: 应用的插件包目录，当需要使用 PhoneGap 的 Java 插件，该插件会安装在此目录。用户也可以构造自己的插件，编写的 Java 文件也应该存放在 src 目录中。
- assets: 应用的整体核心文件，包括界面及逻辑算法处理部分。原始 assets 中的 www 目录被设为隐藏。需要选择 HelloWorld 的项目属性，在 Resource Filters 中把状态还原为默认。之后可正常访问 www 目录。该 www 目录在 Windows 操作系统下的形式为 ..\salock\platforms\android\assets\www
- www: 该文件夹中含有界面的默认文件 index.html、config.xml 标记文件及 cordova.js、cordova_plugins.js 与 phonegap.js 等软件库。另外还有 css、img、js、plugins 子目录存放串样式列表、图像文件、系统互动处理文件及应用的插件。该目录结构简洁与清晰，使构造系统工作容易操作及方便。

整体应用在 Eclipse 中的文件结构如图 5-3 所示。

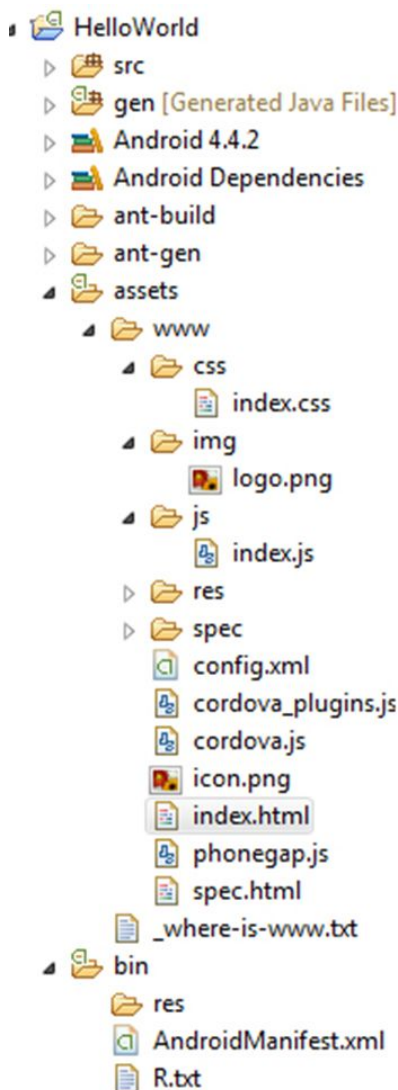


图 5-3 PhoneGap 应用的文件结构

5.4 签名输入区

可看出签名工具是该系统不可缺少的部分。签名记录和签名对比模块会使用到前签名工具来接受用户的签名。

5.4.1 引入签名工具

本系统将使用 jSignature，一种开源支持基于 canvas 的图像处理器。jSignature 可接受手写图案输入和以 Base64 码格式导出不同的格式。因方便管理及使用原因，本项目采用 PNG 格式的导出格式。该工具也支持多笔画输入，这也是能够实现笔画划分功能的重要特征。引入的时候需要在使用到的界面文件导入 jSignature.js、jSignature.UndoButton.js、jSignature.CompressorSVG.js 文件。在这些 HTML 文件中定义加载签名区的位置和属性：


```
$(document).ready(function() {  
  
    // INSTALL JSIGNATURE  
    var $sigdiv = $("#signature").jSignature({  
        color:"#000",  
        "background-color":"#fff",  
        background:"gray",  
        width:"100%",  
        height:"300px",  
        UndoButton:true  
    });  
});
```

图 5-4 在需要使用签名的界面引入 jSignature 工具及设置属性

jSignature 允许使用者自定义签名的颜色、背景颜色、背景图片、签名区域大小等。本项目使用白色加星号型曲线作为背景来提高签名输入的效率，签名颜色设置是红色。

5.4.2 存放签名数据

jSignature 签名工具中每当用户在 canvas 上画出图案，全部属于该图案的笔画对象将存放在一个数据栈中。此工具的数据栈名称是 DataEngine。DataEngine 以笔画为单位存放图案的数据，可通过 UndoButton 部分查看到该模块在取消笔画功能也被引用到。DataEngine 在 jSignature.js 中的部分定义如下图所示。

```
function DataEngine(storageObject, context, startStrokeFn, addToStrokeFn, endStrokeFn){  
    this.data = storageObject // we expect this to be an instance of Array  
    this.context = context  
  
    if (storageObject.length){  
        // we have data to render  
        var numofstrokes = storageObject.length  
        , stroke  
        , numofpoints  
  
        for (var i = 0; i < numofstrokes; i++){  
            stroke = storageObject[i]  
            numofpoints = stroke.x.length  
            startStrokeFn.call(context, stroke)  
            for(var j = 1; j < numofpoints; j++){  
                addToStrokeFn.call(context, stroke, j)  
            }  
            endStrokeFn.call(context, stroke)  
        }  
    }  
}
```

图 5-5 dataEngine 的定义

DataEngine 中定义了 numofstrokes 作为笔画的数量。stroke 是每个笔画的对象，每个对象中包含数组：x 与 y。x 数组与 y 数的结合表示在 canvas 上组成该笔画的坐标点集合。numofpoints 为一个笔画的坐标点数。DataEngine 的数据存储方式是以循环方式对每个笔画一一地记录他们的坐标点集合。此数据栈在笔画划分的模块中会被访问来提取笔画数据。

5.4.3 笔画的起始节点与结束节点

由于笔画划分会以手不离屏幕的情况下画成的笔画作为划分成的单元,能够识别出每个笔画的开始及结束节点也是很重要的。jSignature 中有 `strokeStartCallback` 与 `strokeEndCallback` 函数是与笔画的开始和结束节点相关的函数。此函数将是记录每个笔画持续时间的重要元素,后续工作中笔画的时间记录 `begin` 及 `end` 会在这两个函数中调用到。一下图是 `strokeStartCallback` 和 `strokeEndCallback` 的函数定义。

```
strokeStartCallback = function(stroke) {  
    // this = jSignatureClass instance  
  
    var startTime = MyStroke.begin();  
  
    //BEGIN DRAW STROKE  
    console.log('begin: ' + startTime);  
  
    basicDot(this.canvasContext, stroke.x[0], stroke.y[0], this.settings.lineWidth)  
}
```

图 5-6 `strokeStartCallback` 函数定义

`strokeStartCallback` 表示一个笔画的开始节点方法是构建一个基本点,也就是一个笔画对象中第一个 `x` 与 `y` 数组的元素。同时也初始化了笔画的宽度。

```
strokeEndCallback = function(stroke){  
    var positionInStroke = stroke.x.length - 1  
  
    if (positionInStroke > 0){  
        // there are at least 2 points in the stroke.we are in business.  
        var Cpoint = new Point(stroke.x[positionInStroke], stroke.y[positionInStroke])  
        , Bpoint = new Point(stroke.x[positionInStroke-1], stroke.y[positionInStroke-1])  
        , BCvector = Bpoint.getVectorToPoint(Cpoint)  
        , ABvector  
        if (BCvector.getLength() > this.lineCurveThreshold){  
            // yep. This one was left undrawn in prior callback. Have to draw it now.  
            if (positionInStroke > 1){  
                // we have at least 3 elems in stroke  
                ABvector = (new Point(stroke.x[positionInStroke-2], stroke.y[positionInStroke-2])).getVectorToPoint(Bpoint)  
                var BCP1vector = new Vector(ABvector.x + BCvector.x, ABvector.y + BCvector.y).resizeTo(BCvector.getLength() / 2)  
                basicCurve(  
                    this.canvasContext  
                    , Bpoint.x  
                    , Bpoint.y  
                    , Cpoint.x  
                    , Cpoint.y  
                    , Bpoint.x + BCP1vector.x  
                    , Bpoint.y + BCP1vector.y  
                    , Cpoint.x  
                    , Cpoint.y  
                )  
            } else {  
                // Since there is no AB leg, there is no curve to draw. This line is still "long" but no curve.  
                basicLine(  
                    this.canvasContext  
                    , Bpoint.x  
                    , Bpoint.y  
                    , Cpoint.x  
                    , Cpoint.y  
                )  
            }  
        }  
    }  
  
    var endTime = MyStroke.end();  
}
```

图 5-7 `strokeEndCallback` 函数定义

`strokeEndCallback` 是笔画结束节点的函数。通过笔画拥有的坐标点数来判别画出的笔画是直或是弯。

5. 4. 4 笔画对象导出

如何把画出的签名对象导出成可用的数据也是该签名工具的主要部分之一。jSignature 通过 `getData` 函数来转发从用户选择的导出格式给 `exportplugins` 函数来执行导出操作。

getData 先检查选择的格式是否有误，若出现异常情况会使用默认的导出格式，否则根据用户的选择导出对应的签名数据。getData 函数定义如下图。

```
, 'getData' : function( formattype ) {  
    var undef, $canvas=this.find('canvas.'+apinamespace).add(this.filter('canvas.'+apinamespace))  
    if (formattype === undef) formattype = 'default'  
    if ($canvas.length !== 0 && exportplugins.hasOwnProperty(formattype)){  
        return exportplugins[formattype].call(  
            $canvas.get(0) // canvas dom elem  
            , $canvas.data(apinamespace+'.data') // raw signature data as array of objects of arrays  
        )  
    }  
}
```

图 5-8 调用对象导出函数 getData

Exportplugins 是执行导出操作的函数。提供多种不同的格式，一下图中定义的 exportplugins 函数支持 default、native 与 image 格式。Default 为默认格式，返回对象的数据连接。Native 是属于 jSignature 自定义的格式，导出的对象是一个数组包含多个笔画对象，每笔画内又包含两个坐标点数组 x 与 y。Image 格式导出的数据是从图片信息译码成的 Base64 码。

```
var exportplugins = {  
    'default':function(data){return this.toDataURL()}  
    , 'native':function(data){return data}  
    , 'image':function(data){  
        /*this = canvas elem */  
        var imagestring = this.toDataURL()  
  
        if (typeof imagestring === 'string' &&  
            imagestring.length > 4 &&  
            imagestring.slice(0,5) === 'data:' &&  
            imagestring.indexOf(',') !== -1){  
  
            var splitterpos = imagestring.indexOf(',')  
  
            return [  
                imagestring.slice(5, splitterpos)  
                , imagestring.substr(splitterpos + 1)  
            ]  
        }  
        return []  
    }  
}
```

图 5-9 对象导出执行函数 exportplugins

5.4.5 签名区域的干扰

原始 jSignature 在签名区中会出现一条基本线。该横线在导出签名对象导出时也同时被导出，变成签名的一部分。存在基本线的签名会影响签名的信息和签名对比的效果，因此需要清除该基本线。jSignature.js 中的 basicLine 是构造签名区基本线的函数，把定义 basicLine 的代码行删除则基本线会在签名区消失，方法如下图。

```
// signature line  
ctx.strokeStyle = settings['decor-color']  
ctx.shadowOffsetX = 0  
ctx.shadowOffsetY = 0  
var lineoffset = Math.round( ch / 5 )  
// REMOVE GREY LINE  
//basicLine(ctx, lineoffset * 1.5, ch - lineoffset, cw - (lineoffset * 1.5), ch - lineoffset)  
ctx.strokeStyle = settings.color
```

图 5-10 签名区的基本线清除

5.5 笔画划分实现

将输入的签名划分成若干个笔画作为最小单元来进行各种不同的操作及记录,比较等是本系统的核心模块之一。该功能以一个独立的模块形式在系统内运作。记录、存储、对比等功均能分别或结合地调用该模块。该设计模块方法使系统的核心模块变得灵活,容易修正且不影响其他模块的运作效果。笔画划分功能由一个自定义的全局对象担任。需先创建一个 JavaScript 文件 mystroke.js,在该文件中定义全局对象 MyStroke。MyStroke 包含笔画划分功能的定义及调用签名对比的其他函数。本模块把所有笔画视为一个 MyStroke 中的成员对象 Stroke。Stroke 包含本次输入签名划分出的笔画组数组 source、作为对比样本的样本笔画组 target 数组、存放双方笔画组的相似度的 compareData 数组、验证结果状态 fail 和允许验证通过的相似率 matchMark。Stroke 对象的定义如图 5-11 所示。

```
// Stroke OBJECT
Stroke: {
  source: new Array(),           // FROM CANVAS
  target: new Array(),           // FROM DB,
  compareData: new Array(),      // MISMATCH %
  fail: false,                   // LOGIN STATUS
  matchMark: 1.5                 // FROM DB (CHARACTER TBL)
},
```

图 5-11 笔画对象定义

splitStroke 是 MyStroke 中担任划分签名成多个笔画的成员函数。该函数即将访问 jSignature 模块中的存放签名笔画数据栈 dataEngine。函数中首先需要把 dataEngine 中存储的所有笔画数据移至 strokes_data 数组。strokes_data 的每个元素为一个笔画的对象,对象的内容包含两个数组: x 数组与 y 数组。两个数组的长度总相同,意味着两数组中按序的每个 x、y 将一一配对组成一个坐标点。一序列的坐标点组成一个完整的笔画。获取数据栈的数据后先把作为显示区的 HTML 元素 extraarea 清空,则启动 formatchanged 事件。之后再给 Stroke 对象中的 source 数组初始化。接下来 splitStroke 将进入循环操作,每次循环对 strokes_data 数据栈中的每个笔画进行操作。操作内容包括:

- 使用 jSignature 模块中的 resetCanvas 功能单独地弹出该笔画至显示区。
- 获取笔画的 Base64 码: 通过 jSignature 的 getData 功能,选择的类型为“image”,也就是 PNG 图片的 Base64 码。
- 定义笔画的数据包对象 strokeData,包含笔画的 Base64 码 stroke 和同时计算笔画的持续时间 duration。持续时间是获取属于本笔画中 durations 对象的 end 元素与 begin 元素之差。计算笔画开始的时间 begin 函数和结束时间 end 函数的定义如下图。

```
// START TIME OF A STROKE
begin: function ()
{
    var a = new Date();
    var startTime = a.getTime();

    // PUSH STARTTIME TO ARRAY
    this.durations.begin.push(startTime);

    return startTime;
},

// END TIME OF A STROKE
end: function ()
{
    var b = new Date();
    var endTime = b.getTime();

    //ADD ENDTIME TO ARRAY
    this.durations.end.push(endTime);

    return endTime;
},
```

图 5-12 获取笔画开始与结束时间函数

- 把数据包 strokeData 推进输入笔画的 source 数组中：循环结束后，最后的工作是把完整的签名显示至 HTML 文件中的显示区。
针对实现笔画划分功能的 splitStroke 函数完整代码如图 5-13 所示。

```
// SPLIT AND COMPARE
splitStroke: function ()
{
    // GET THE STACK
    var strokes_data = MyStroke.jSignatureInstance.dataEngine.data;

    // CLEAN OUTPUT AREA
    $.publish('formatchanged');

    this.Stroke.source = new Array();
    this.Stroke.compareData = new Array();

    // SCAN THE STACK
    for (i in strokes_data)
    {
        var stroke = strokes_data[i];

        // OUTPUT STROKE
        this.jSignatureInstance.resetCanvas([stroke]);

        // CONVERT stroke OBJECT --> Base64 code
        var data = $("#signature").jSignature('getData', 'image');

        // DATA OF A STROKE
        var strokeData =
        {
            stroke: data, // Base64 code
            duration: MyStroke.durations.end[i] - MyStroke.durations.begin[i] // get duration
        }

        // PUSH TO SOURCE ARRAY
        this.Stroke.source.push(strokeData);

        // CONVERT ARRAY TO STRING
        $("#code").val(''+this.Stroke.source.join('','+\\n\\n'+''));
    }

    // END LOOP

    // REDRAW WHOLE SIGNATURE
    this.jSignatureInstance.resetCanvas(strokes_data);

    return true;
},
```

图 5-13 笔画划分函数

5.6 图像识别实现

图像识别是本认证系统的核心模块，工作是对从本次输入的签名所导出的图案对象与设置为样本的签名文件做对比。本项目使用由 James Cryer 开发的 Resemble 作为签名对比的工具。以图片的像素结构作为对比目标，Resemble 可识别出两幅不同格式的圖片的区别点且同时计算出两幅图的差别率。

5.6.1 Resemble 的比较法

由于本系统的对比对象是签名，而签名的彩色与签名区与的背景只是两种单颜色，因此在使用 resemble 工具时无需考虑颜色、修边等问题。接下的实现代码解释只考虑与调用到的功能的相关的部分。

首先本系统使用 resemble 时的调用语句 resembleControl 的格式如下：

```
resemble(file).compareTo(file2).onComplete(onComplete)
```

该语句表示将第一个图片文件读入 resemble 中，通过 compareTo 函数来与第二个图片文件进行对比。对比过程始终由 onComplete 函数执行。整体的调用顺序是 compareTo 函数、onComplete 函数、compare 函数、analyseImages 函数。接下将一一解释这些函数的功能及调用。

`compareTo` 主要是转发被使用来对比的第二图片文件对象。该函数返回的数值是以第二个对比图片对象为参数的 `getCompareApi` 对象。`getCompareApi` 中定义了 `onComplete` 成员函数。因此原始的主调用语句中 `compareTo(file2).onComplete(onComplete)` 意味着执行包含 `file2` 对象的 `onComplete` 函数。`compareTo` 函数的定义如下图所示。

```
compareTo: function(secondFileData){  
    return getCompareApi(secondFileData);  
}
```

图 5-14 调用图案比较函数 `compareTo`

`onComplete` 是 `getCompareApi` 的成员函数。`onComplete` 的主要的工作是可执行用户自定义的 `callback` 函数来处理对比过程的后续处理工作，同时通过 `wrapper` 函数调用 `compare` 函数来把两个图片对象进行对比。`onComplete` 的函数定义如下图。

```
onComplete: function( callback ){  
  
    updateCallbackArray.push(callback);  
  
    var wrapper = function(){  
        compare(fileData, secondFileData);  
    };  
  
    wrapper();  
  
    return getCompareApi(wrapper);  
}
```

图 5-15 `getCompareApi` 中的成员函数 `onComplete`

`compare` 也可视为核心的对比过程的前半部分函数。该函数主要通过 `onceWeHaveBoth` 函数将需要对比的两个对象的大小，包含宽度和高度，进行比较。`onceWeHaveBoth` 函数先判别两者的大小是否相等。若两者的大小不匹配，则取大的数值作为下一步正常化图片的常数。所谓的正常化是向加载两幅图的 `canvas` 载体取同样的宽度及高度大小，从而可执行对比工作。正常化后的两个图片对象会通过 `analyseImages` 来进行笔画对比工作。`compare` 的函数定义如下图所示。


```
function compare(one, two){
    function onceWeHaveBoth(){
        var width;
        var height;
        if(images.length === 2){
            width = images[0].width > images[1].width ? images[0].width : images[1].width;
            height = images[0].height > images[1].height ? images[0].height : images[1].height;

            if( (images[0].width === images[1].width) && (images[0].height === images[1].height) ){
                data.isSameDimensions = true;
            } else {
                data.isSameDimensions = false;
            }

            data.dimensionDifference = { width: images[0].width - images[1].width, height: images[0].height - images[1].height };

            analyseImages( normalise(images[0],width, height), normalise(images[1],width, height), width, height);

            triggerDataUpdate();
        }
    }

    images = [];
    loadImageData(one, onceWeHaveBoth);
    loadImageData(two, onceWeHaveBoth);
}
```

图 5-16 图案对比主函数 compare

analyseImages 是执行笔画对比的核心函数。该函数读入两个图片对象的数据，之后以像素的结构为单位，对两者进行扫描。每次扫描跳向下一个像素的偏移量 offset 为（纵坐标 × 图片宽度 + 横坐标）× 4。若扫描到的对应两个像素颜色不匹配，则生成一个错误像素及差别率加一，若两者之一为空图片则返回。最后得出的输出结果有两种：差别结果图，由所有错误像素组成的图片、差别率，通过与整幅图的像素面积取比率值。

由于 analyseImages 的计算方式也考虑背景的颜色，签名的像素数与背景的像素数之差相当大，导致差别率出现相反的结果。于是在实现系统中调用 resemble 得出的差别率可作为相似度来决定验证是否成功。analyseImages 的函数定义如下图。

```
function analyseImages(img1, img2, width, height){
    var hiddenCanvas = document.createElement('canvas');

    var data1 = img1.data;
    var data2 = img2.data;

    hiddenCanvas.width = width;
    hiddenCanvas.height = height;

    var context = hiddenCanvas.getContext('2d');
    var imgd = context.createImageData(width,height);
    var targetPix = imgd.data;

    var mismatchCount = 0;

    loop(height, width, function(verticalPos, horizontalPos){
        var offset = (verticalPos*width + horizontalPos) * 4;
        var pixel1 = getPixelInfo(data1, offset, 1);
        var pixel2 = getPixelInfo(data2, offset, 2);

        if(pixel1 === null || pixel2 === null){
            return;
        }

        if( isRGBSimilar(pixel1, pixel2) ){
            copyPixel(targetPix, offset, pixel1, pixel2);
        } else {
            errorPixel(targetPix, offset, pixel1, pixel2);
            mismatchCount++;
        }
    });

    data.misMatchPercentage = (mismatchCount / (height*width) * 100).toFixed(2);
}
```

图 5-17 图案对比执行函数 analyseImages

5.6.2 对比工具调用

系统结构中的签名对比模块需要调用到 Resemble 的对比功能，因此需要自定义调用函数。在 mystroke.js 文件中的 MyStroke 对象定义一个成员函数 compareStroke。该函数的任务为转入本次输入的签名对象和作为样本的签名给 Resemble 处理。本函数也同时实现了笔画数量的对比工作。

compareStroke 的参数是需要对比的两个签名对象 source 与 target。其中 source 为本次输入的签名笔画及，格式是 PNG 文件 Base64 码，而 target 是签名样本的笔画 PNG 文件集。

对比的最初过程是先比较两个签名对象的笔画数量，也就是 source 与 target 的数组长度。个笔画的对比过程只执行当笔画的数量匹配。笔画对比操作主要是通过循环，每次按序提取 source 与 target 数组中的笔画对象，再调用 resemble 的 compareTo 和 onComplete 函数来进行笔画对比。由于格式不同从 source 数组获取出的对象数据是 Base64 码，因此需要添加“data:”使该笔画对象能转换成可对比的图案。target 数组的对象是 PNG 文件，只需要直接读入即可。该调用语句会处理需要对比笔画的情况下的后期处理工作。在笔画不匹配的情况下则该函数立即跳往 loginFoward 函数来处理登录状态。compareStroke 的函数定义如下图所示。

```
// COMPARE
compareStroke: function (source, target) {

    // WHEN STROKE AMOUNT EQUAL
    if (source.length == target.length) {

        // SCAN EVERY STROKE IN SOURCE
        for (i in source) {

            // stroke a: BASE64 code from SOURCE
            var stroke_a = "data:" + source[i].stroke;

            // stroke b: PNG from FILE SYSTEM
            var stroke_b = target[i];

            // COMPARE EACH STROKE
            resembleControl = resemble(stroke_a).compareTo(stroke_b).onComplete(this.onComplete);
        }

        // STROKE AMOUNT ERROR
        else {

            MyStroke.Stroke.fail = true;

            salockDB.errorData = 'Stroke amount error';

            this.loginFoward();
        }

        return;
    },// END COMPARE
```

图 5-18 调用图像对比函数 compareStroke 的定义

5.6.3 对比状态处理

由于认证过程将出现不同的场合，因此需要灵活地处理各场合发生的后续工作。需要处理的场合可分成三种：笔画数量对比出错、笔画对比出错、笔画比通过。

笔画数量出错时在 compareStroke 函数会直接转至 loginFoward 函数来处理后续工作。loginFoward 根据验证通过或失败场合来做出对应的处理。若验证成功会弹出成功信息，若验证失败会清除 canvas 签名区域，用户需要重新输入签名密码。compareStroke 中笔画数量

不匹配时会立即设 `Stroke.fail` 状态为 `true`，表示验证失败，因此 `loginForward` 函数执行是只需要考虑验证失败的场合。

```
// LOGIN
loginForward: function () {

    var fail = this.Stroke.fail;

    // WHEN FAIL
    if (fail == true) {

        alert("Unlock Fail!");
        this.jSignatureInstance.resetCanvas();    // reset canvas

        // RECORD ERROR DATA
        salockDB.errorRecord();
    }

    // WHEN SUCCESS
    else {

        alert("Unlock succeeded!");
    }

    return;
}
```

图 5-19 登录状态判别函数 `loginForward`

`compareStroke` 函数的笔画对比工作的出的结果、状态与相应处理由 `onComplete` 函数来完成。`onComplete` 的任务是生成 `source` 与 `target` 的区别结果图案、计算相似度的平均数值、比较得出的相似度和设置的相似度和转发验证结果给 `loginForward` 函数。

`onComplete` 函数先从转入的数据参数 `data`，为 `source` 与 `target` 的签名笔画对象，生成 `diffImage` 区别图案。再把计算出的相似度推进 `Stroke` 对象中的 `compareData` 数组。最后选择 `compareData` 中的所有数据值取出平均数，再和设置好的允许相似度对比。从相似度是否符合要求来设置认证结果的状态，从而转发该结果给 `loginForward` 函数。`onComplete` 函数的实现代码如下图所示。

```
// COMPARE COMPLETE
onComplete: function (data){

    // DIFFERENT IMAGE
    var diffImage = new Image();

    diffImage.src = data.getImageDataURL();

    MyStroke.Stroke.compareData.push(parseFloat(data.misMatchPercentage));

    // CHECK IF COMPARE THE LAST STROKE
    if (MyStroke.Stroke.compareData.length == MyStroke.Stroke.source.length) {

        // CALCULATE AVERAGE MISMATCH
        var mark = Math.average(MyStroke.Stroke.compareData);

        // MISMATCH ERROR
        if (mark < MyStroke.Stroke.matchMark)
        {
            MyStroke.Stroke.fail = true;

            salockDB.errorData = 'Mismatch error';

        }

        // SUCCESS
        else
        {
            MyStroke.Stroke.fail = false;
        }

        // CHECK STATUS
        MyStroke.loginFoward();

    }

    // DISPLAY DIFF IMAGE
    $(diffImage).attr("width","80%").css("width","80%").appendTo('#displayarea');
    $('#displayarea').append("<hr /><p>"+"Match: "+data.misMatchPercentage+"</p>");
},
```

图 5-20 处理对比后期工作函数 onComplete

5.6.4 引入对比模块

为了使完整的对比工作流程可用，完成的执行签名对比函数 compareStroke 会与笔画划分函数 splitStroke 配合使用。因此自需要定义一个 doCompare 函数来顺序地调用此函数即可。doCompare 也是在开锁界面用户点击开锁按钮时启动的函数。doCompare 的函数定义如下图。

```
// RUN SPLIT & COMPARE FUNCTION
doCompare: function()
{
    // SPLIT
    this.splitStroke();

    // COMPARE
    this.compareStroke(this.Stroke.source, this.Stroke.target);

},
```

图 5-21 执行完整签名对比函数 doCompare

5.7 数据库实现

本系统对数据库的操作主要有记录签名属性、记录属于该签名个笔画的属性、记录验证错误信息及删除数据表中的签名与笔画信息。

从以上需求，可推出该系统需要三个数据表：签名数据表、笔画数据表与错误信息数据表。签名、笔画、错误信息这三类对象中各类的元素需要有唯一性，且有密切的关系。此数据表包含以下属性：

- 签名数据表：唯一的 id、用户自定义名字和创建日期。
- 笔画数据表：唯一的 id、对应的签名 id 和笔画的持续时间。
- 错误信息数据表：唯一的 id、对应的签名 id 和验证错误发生的时间。

其中笔画数据表与错误信息数据表的属性都与签名的唯一 id 有关系。则执行批量添加、删除等操作时可同时对三个数据表有作用。

5.7.1 数据库构建

本系统使用属于 PhoneGap 的 Cordova 内置插件 Storage API 中的 SQLite 来构建 WebSQL 数据库。需要在项目目录中的 config.xml 文件中添加该特性：

```
<feature name="Storage">
  <param name="android-package" value="org.apache.cordova.Storage" />
</feature>
```

所有与数据库相关的定义及操作都存放在自定义的 www/js/salock_database.js JavaScript 文件中。

首先定义一个全局的总对象 salockDB，该对象包含所有数据库对象及相应的操作。根据以上需求，使用 openDatabase 函数定义三个数据库对象操作如下图。

```
// DATABASE OBJECT
strokeDB: window.openDatabase("strokeDB", "1.0", "Stroke Database", 2*1024*1024),
characterDB: window.openDatabase("characterDB", "1.0", "Character Database", 2*1024*1024),
errorTypeDB: window.openDatabase("errorTypeDB", "1.0", "Error Type Database", 2*1024*1024),
```

图 5-22 数据库对象定义

该函数返回一个数据库对象，4 个参数按序为：数据库名、版本、显示名及容量大小（单位为字节）。characterDB、strokeDB、errorTypeDB 对应为签名数据表、笔画数据表及错误信息数据表。所有数据表的版本为 1.0，容量为 2MBs。

为了使数据库当每次进入应用可立即使用，需要定义一个触发接受器。触发事件为 deviceready，意义是当 PhoneGap 已经完成加载工作后才能执行 onDeviceReady 函数。把触发器定义在 init 初始化函数内，则每次访问 HTML 文件时，HTML 元素（如 body）加载完成后会自动执行该函数。

```
// CREATE EVENT LISTENER
init: function ()
{
  document.addEventListener("deviceready", this.onDeviceReady, false);

  return true;
},
```

图 5-23 基于 PhoneGap 的初始化函数 init

PhoneGap 加载完成后将自动执行 `onDeviceReady` 函数。此函数使用 SQLite 中的 `transaction` 功能来处理数据库操作。`Transaction` 函数包含三个参数：数据库操作函数（`populate`）、操作成功函数（`successCB`）与操作失败函数（`errorCB`）。该函数先执行 `populate` 函数，从 `populate` 执行得出的完成或失败状态接下执行 `successCB` 或 `errorCB`。以下图是对三个数据库对象进行创建数据表的函数代码。

```
// DEFINE DATABASE
onDeviceReady: function ()
{
    // CREATE DB
    salockDB.characterDB.transaction(salockDB.populateCharacter,salockDB.errorCB,salockDB.successCB);
    salockDB.strokeDB.transaction(salockDB.populateStroke,salockDB.errorCB,salockDB.successCB);
    salockDB.errorTypeDB.transaction(salockDB.populateError,salockDB.errorCB,salockDB.successCB);

    return true;
},
```

图 5-24 执行数据库对象的操作

`Populate` 函数使用 SQL 语言执行数据库操作，本函数的操作为创建数据表。`Populate` 中的 `tx` 参数为内体数据库对象，所有对数据库的 SQL 操作由 `tx` 的 `executeSql` 函数实现。本操作中的 SQL 语句 `DROP TABLE IF EXISTS<数据表名>` 可实现每次打开应用清空数据表。若想永久保留数据则删除该代码行。`CREATE TABLE IF NOT EXISTS<数据表名>` 语句会构建出一个数据表并同时定义数据表中的属性。由于每数据表都有无二义性的属性 `id`，因此在 `id` 属性需要添加 `INTEGER PRIMARY KEY` 定义，该定义表示 `id` 为主键，有唯一性且每次插入新元素会自动自增地生成。通过 `populate` 函数对 `characterDB`、`strokeDB`、`errorTypeDB` 构造表格的具体实现代码如下图。

```
// CREATE CHARACTER TABLE
populateCharacter: function (tx)
{
    // RESET TABLE
    // tx.executeSql('DROP TABLE IF EXISTS characterDB');

    // CREATE TABLE
    // INTEGER PRIMARY KEY: auto increase for id data
    tx.executeSql('CREATE TABLE IF NOT EXISTS characterDB (id INTEGER PRIMARY KEY, name, date)');

    return true;
},

// CREATE STROKE TABLE
populateStroke: function (tx)
{
    // RESET TABLE
    // tx.executeSql('DROP TABLE IF EXISTS strokeDB');

    // CREATE TABLE
    // INTEGER PRIMARY KEY: auto increase for id data
    tx.executeSql('CREATE TABLE IF NOT EXISTS strokeDB (id INTEGER PRIMARY KEY, character_id, duration)');

    return true;
},

// CREATE ERROR TABLE
populateError: function (tx)
{
    // RESET TABLE
    // tx.executeSql('DROP TABLE IF EXISTS errorTypeDB');

    // CREATE TABLE
    // INTEGER PRIMARY KEY: auto increase for id data
    tx.executeSql('CREATE TABLE IF NOT EXISTS errorTypeDB (id INTEGER PRIMARY KEY, character_id, type, date)');

    return true;
},

// ERROR HANDLER
errorCB: function (err)
{
    // print the error message
    alert("Error processing SQL: " + JSON.stringify(err));

    return true;
},

// DATABASE CREATE SUCCESS
successCB: function ()
{
    //alert("Database created!");

    return true;
},
},
```

图 5-25 构建数据库对象的数据表

5.7.2 写入签名属性

在系统的设置新样本界面会使用到该功能。需要在 record.html 中的按钮元素上定义用户点击按钮时对应的执行函数。在此 HTML 文件中的<script></script>中定义绑定函数,使用户点击 Record 按钮时会系统会执行 salockDB 对象中的 CharacterRecord 函数。该函数是签名属性记录的函数,此操作结束后会继续通过第二阶段为调用气压按钮来继续记录签名的笔画属性。

CharacterRecord 首先执行数据库操作函数 insertCharacterDB 来记录签名属性。insertCharacterDB 函数将从 HTML 文件中由用户输入自定义签名名字的输入框获取签名名字并存入_name,再用 Date()函数获取系统的当前时间存入_date。SQL 语句中的“?”符号为占位符。以 executeSql 执行 SQL 语句,把_name 与_date 插入 characterDB 数据表中。

以上操作成功后会继续调用 successCharacterDB 函数。本函数中的 results 参数为以上执

行 SQL 语句的结果对象。使用 results 对象的 insertID 函数可返回刚插入元素的 id，把获取的 id 存放 salockDB 的 currentCharID 成员对象。该 id 在记录签名的笔画时将被使用，向同一个签名的笔画组的签名属性赋值。签名记录过程到外步骤则结束，则使用 jQuery 语句 \$("#recordBtn").hide() 和 \$("#recordStroke").show() 隐藏原有的记录签名按钮并同时显示执行下一步记录工作的 Strokes 按钮。从调用 CharacterRecord 之后所有记录签名的流程实现代码如下。

```
// RECORD CHARACTER
CharacterRecord: function()
{
    salockDB.currentCharID = 0;

    this.characterDB.transaction(salockDB.insertCharacterDB,salockDB.errorCB);

    return true;
},

// INSERT CHARACTER
insertCharacterDB: function(tx)
{
    // SET VALUE to INSERT
    var _name = $("#signature_name").val();

    var _date = new Date();

    var sql = 'INSERT INTO characterDB (name, date) VALUES (?,?)';

    tx.executeSql(sql,[_name,_date],salockDB.successCharacterDB,salockDB.errorCB);

    return true;
},

// GET CHARACTER ID
successCharacterDB: function(tx, results)
{
    // GET CURRENT CHARACTER ID
    salockDB.currentCharID = results.insertId;

    MyStroke.characterCode = $("#signature").jSignature('getData', 'image');

    $("#recordStroke").show();
    $("#recordBtn").hide();

    return true;
},
```

图 5-26 记录签名信息过程

5.7.3 写入笔画属性

笔画记录为签名记录的直接后续工作。该过程与记录签名类似，同样经过点击 Strokes 按钮来调用笔画记录函数 StrokeRecord。因此也需要在 record.html 文件添加 Strokes 按钮，也同时定义绑定函数，当点击会调用 StrokeRecord 函数并开始划分同时记录笔画属性。

由于记录前需要把签名中的所有笔画进行划分，则先调用 MyStroke 对象中的笔画划分函数 splitStroke。后续的笔画属性写入操作与 splitStroke 函数异步执行。

insertStrokeDB 是写入数据库笔画属性的函数。由于与 splitStroke 异步执行，因此该函数也需要循环地操作。循环次数与 splitStroke 中的笔画划分次数相同，为输入的签名笔画栈中的笔画数 Stroke.source.length。关于每笔画的持续时间长度，MyStroke 中的 begin 及 end

函数当每次记录一个笔画的开始和结束时间时直接推进 `durations` 对象中的 `begin` 和 `end` 数组。取 `durations` 数组中同标记元素的 `end` 与 `begin` 之差的值可得当前获得的笔画的持续时间。以 `currentStroke` 成员对象为目前的笔画下标，每当从 `splitStroke` 获取到的当前笔画持续时间信息，立即写入该信息至笔画数据表。另外从 `successCharacterDB` 得出的签名 `id` 也存入数据表中，表示若干个笔画是属同一个签名。当全部笔画记录完成则把下标 `currentStroke` 还原成 0，为了下次使用。

当笔画信息写入成功会自动执行 `successStrokeDB` 函数，弹出成功通知，否则运行 `errorCB` 错误通知函数。为了避免重启应用而能方便直接记录下一个签名，接受签名 `canvas` 区域在签名与笔画完成记录后会把刚记录的签名清空。同时 `MyStroke` 中的 `durations` 对象的 `begin` 和 `end` 数组也需要清空，则能使后续的笔画持续时间能记录工作能正确地运作。记录笔画按钮 `Strokes` 被回收并同时还原原始的签名记录按钮 `Record`。

完整的数据库笔画记录过程实现代码如下图所示。

```
// RECORD STROKE
StrokeRecord: function()
{
    // SPLIT STROKE
    MyStroke.splitStroke();

    // START INSERT
    this.strokeDB.transaction(salockDB.insertStrokeDB, salockDB.errorCB);

    return true;
},

// INSERT STROKE
insertStrokeDB: function(tx)
{
    while (salockDB.currentStroke < MyStroke.Stroke.source.length) {

        // set data
        var _character_id = salockDB.currentCharID;
        var _duration = MyStroke.Stroke.source[salockDB.currentStroke].duration ;

        var sql = 'INSERT INTO strokeDB (character_id, duration) VALUES (?,?)';

        // INSERT DATA
        tx.executeSql(sql,[_character_id, _duration],salockDB.successStrokeDB,salockDB.errorCB);

        // NEXT STROKE
        salockDB.currentStroke++;
    }

    // RESET
    salockDB.currentStroke = 0;

    return true;
},

// INSERT STROKE SUCCESS
successStrokeDB: function(tx)
{
    // RESET CANVAS AFTER INSERT
    MyStroke.jSignatureInstance.resetCanvas();

    // Clean array
    // PREPARE FOR NEXT RECORD
    MyStroke.durations.begin = [];
    MyStroke.durations.end = [];

    $("#recordStroke").hide();
    $("#recordBtn").show();

    tx.executeSql('SELECT * FROM strokeDB', [], salockDB.renderDataList, salockDB.errorCB );

    return true;
},
```

图 5-27 记录笔画信息过程代码

5.7.4 写入验证错误信息

该功能将在屏幕解锁模块使用到。每当用户开锁不成功时，根据验证失败的原因（笔画数量不匹配或笔画顺序有误等）而写入相应的验证错误类型。errorRecord 函数在 MyStroke 全局对象的 loginForward 函数中调用到。

被调用的 insertErrorTypeDB 函数负责写入错误信息，其中 salockDB.errorData 是存放错误类型信息的成员变量，在 MyStroke 的笔画比较功能中依据验证失败的不同原因而取得相应的错误信息。错误类型数据表记录时也同时获取当前作为比较样本的签名 id 号及认证出错的时间。具体的实现过程如下图。

```
// RECORD ERROR
errorRecord: function()
{
    // START INSERT ERRORTYPE
    this.errorTypeDB.transaction(salockDB.insertErrorTypeDB, salockDB.errorCB);

    return true;
},

// INSERT ERROR TYPE
insertErrorTypeDB: function(tx)
{
    alert("Insert error type");

    // SET VALUE
    var _character_id = salockDB.currentCharID;
    var _error_type = salockDB.errorData;

    var sql = 'INSERT INTO errorTypeDB (character_id, type, date) VALUES (?, ?, ?)';

    // INSERT
    tx.executeSql(sql, [_character_id, _error_type, _date], salockDB.successErrorTypeDB, salockDB.errorCB);

    return true;
},

// INSERT ERROR SUCCESS
successErrorTypeDB: function(tx)
{
    alert("ErrorType query success!");

    tx.executeSql('SELECT * FROM errorTypeDB', [], salockDB.renderDataList, salockDB.errorCB );

    return true;
},
```

图 5-28 记录验证错误信息过程代码

引入错误信息的地方有三个，均属于对比签名部分。错误类型主要有两种：签名笔画数量错误或签名对比错误。根据不同场合设置相应的数据，再加上最后记录至数据工作这个引用方法如下面的图序列所示。

```
// COMPARE
compareStroke: function (source, target) {

    // WHEN STROKE AMOUNT EQUAL
    if (source.length == target.length) {

        // SCAN EVERY STROKE IN SOURCE
        for (i in source) {

            // stroke a: BASE64 code from SOURCE
            var stroke_a = "data:" + source[i].stroke;

            // stroke b: PNG from FILE SYSTEM
            var stroke_b = target[i];

            // COMPARE EACH STROKE
            resembleControl = resemble(stroke_a).compare(stroke_b);

        }

        // STROKE AMOUNT ERROR
        else {

            MyStroke.Stroke.fail = true;

            salockDB.errorData = 'Stroke amount error';

            this.loginForward();

        }

        return;
    }
}
```

图 5-29 在 compareStroke 函数设置笔画数量错误的信息



```
// COMPARE COMPLETE
onComplete: function (data){

    // DIFFERENT IMAGE
    var diffImage = new Image();

    diffImage.src = data.getImageDataURL();

    MyStroke.Stroke.compareData.push(parseFloat(data.misMatchPercentage));

    // CHECK IF COMPARE THE LAST STROKE
    if (MyStroke.Stroke.compareData.length == MyStroke.Stroke.source.length) {

        // CALCULATE AVERAGE MISMATCH
        var mark = Math.average(MyStroke.Stroke.compareData);

        // MISMATCH ERROR
        if (mark < MyStroke.Stroke.matchMark)
        {

            MyStroke.Stroke.fail = true;

            salockDB.errorData = 'Mismatch error';

        }

        // SUCCESS
        else
        {
            MyStroke.Stroke.fail = false;

        }

        // CHECK STATUS
        MyStroke.loginForward();
    }
}
```

图 5-30 在 onComplete 函数设置笔画对比错误的信息

```
// LOGIN
loginForward: function () {

    var fail = this.Stroke.fail;

    // WHEN FAIL
    if (fail == true) {

        alert("Unlock Fail!");
        this.jSignatureInstance.resetCanvas(); // reset canvas

        // RECORD ERROR DATA
        salockDB.errorRecord();

    }

    // WHEN SUCCESS
    else {

        alert("Unlock succeeded!");
        this.jSignatureInstance.resetCanvas(); // reset canvas

    }

    return;
}
}
```

图 5-31 在 loginForward 函数执行错误信息记录工作

5.7.5 数据库内容显示

为了使用户容易查看目前数据库的内容，必须向系统提供公用的数据读取功能。数据读取函数从 SQL 语言的 SELECT 语句中把输出结果以列表形式显示。参数 results 为 SQL 结果，获取 results 的所有行元素（rows.item）再用 JSON.stringify 函数转换成字符串形式。用 html() 函数把字符串形式的数据显示在 HTML 文件中的 dataView 模块。任何数据库对象都可调用 renderDataList 来读取数据表中的数据，具体代码实现如下图。

```
// PRINT FUNCTION
renderDataList: function (tx, results)
{
    // store the result to print in HTML
    var htmlstring = '';

    // number of rows in database
    var len = results.rows.length;

    // loop for value in every row in DB
    for(var i=0; i<len; i++)
    {

        // store value from DB to text
        // an item is a group of whole row in a table
        var text = JSON.stringify(results.rows.item(i));

        // display as unordered list
        htmlstring += '<li>' + text + '</li>';

        //console.log(text);
    }

    // output htmlstring to HTML document
    $('#dataView').html(htmlstring);
}
```

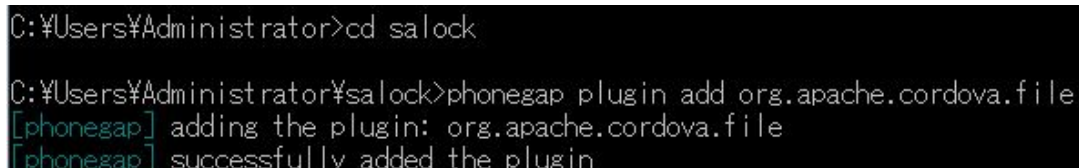
图 5-32 数据库内容显示函数 renderDataList

5.8 文件管理实现

从文件管理的需求，本认证系统必须支持基于文件处理的操作。此操作包括读写文件，删除文件和浏览文件。需要处理的文件为 PNG 图像文件，也就是签名样本文件。本系统的开发框架 PhoneGap 可支持 File API，一种可实现基于文件管理系统操作的插件。与文件处理相关的操作将被引入系统的各主功能中，包括签名样本记录、删除和选择功能。

5.8.1 PhoneGap 的 File API 安装及使用

PhoneGap 原始框架还未支持文件的操作，因此需要安装第三方的插件 File API。通过命令行方式给本系统安装该插件，如下图：



```
C:\Users\Administrator>cd salock
C:\Users\Administrator\salock>phonegap plugin add org.apache.cordova.file
[phonegap] adding the plugin: org.apache.cordova.file
[phonegap] successfully added the plugin
```

图 5-33 安装 PhoneGap 的 File API 插件

由于系统以手机应用形式在手机上运行，最适当的存储位置为属于手机内部存储区中该应用的安装目录。为了获得在内部存储的写入文件权限，需要在本项目的 config.xml 文件中

添加关于文件处理的属性设定。

```
<preference name="AndroidPersistentFileLocation" value="Internal" />
```

安装过程完成后 File API 的插件文件可在本系统的项目工作文件中查看，目录为 ..\HelloWorld\org.apache.cordova.file。Org.apache.cordova.file 包中含有与文件操作相关的本地 Java 类。系统会通过 JavaScript 函数来调用此类的功能。File API 的特征是不支持计算机的浏览器，而只能在手机上运行文件系统对象申请操作。因此文件管理的实验及修正过程必须要在手机上执行才能观察到。

5.8.2 文件管理操作

调用本地 Java 类功能由 JavaScript 对象来担任。所有与文件处理相关的操作均定义在 js/storage.js 文件中。与数据库处理类似，所有处理函数将被封装在一个总对象内，此操作组的总对象为 salockFile。除了操作类函数，salockFile 还存放其他成员变量及数组如 rootdir、fileSystem、fileEntry、imageCode。

由于该文件管理系统的默认目录为手机内部最外层的存储目录（使用的手机中该目录名为 /mnt/sdcard），因此需要定义 rootdir 作为自定义的当前使用目录路径。rootdir 的路径值是 /Android/data/com.phonegap.helloworld/files，引入应用的安装目录（绝对路径为 /mnt/sdcard/Android/data/com.phonegap.helloworld/files）。com.phonegap.helloworld 目录中的 cache 与 files 文件夹为 File API 在安装应用的时候自动生成。本项目的签名样本存储在 files 目录中，以 character 和 stroke 子文件夹分别存放签名文件和各笔画的文件。

fileSystem 是 salockFile 的样本管理对象。当对象的申请操作成功，生成的文件系统对象将传送给 fileSystem 对象，fileSystem 成为执行所有与文件相关的操作对象。

fileEntry 与 fileSystem 类似，是 salockFile 的成员文件对象，作为处理文件操作的临时文件对象变量。

imageCode 是存储目标图像的 Base64 码变量，在 salockFile 的每图像对象的操作中会把本身的 Base64 码存放至 imageCode 变量来转发给函数进行处理。

基于文件管理系统的 File API 插件仅限在智能手机上使用，因此在调用任何功能之前必须申请一个文件管理对象。为了使申请操作能运行，记录笔画 record 的 HTML 文件中 body 元素加载完毕时需要运行 salockFile 的初始化函数 init。init 的内容为添加对象申请的事件触发接收器 deviceready。当 PhoneGap 加载后执行 onDeviceReady 函数。onDeviceReady 会向系统申请一个文件系统对象。window.requestFileSystem 函数含四种变，从左至右：LocalFileSystem.PERSISTENT 为持续类型的文件系统对象，在未收到应用或用户的允许下则不可删除；0 为对象的大小，以字节为单位，设置大小为 0 意味着该文件系统无固定的容量大小；salockFile.onFSSuccess 为申请成功后的后续函数；salockFile.onError 为申请失败后的后续函数。完整文件管理系统的申请及生成的实现代码如下。


```
// CREATE EVENT LISTENER
init: function () {
    document.addEventListener("deviceready", salockFile.onDeviceReady, true);
},

// REQUEST FILE SYSTEM
onDeviceReady: function () {

    window.requestFileSystem(LocalFileSystem.PERSISTENT, 0, salockFile.onFSSuccess, salockFile.onError);

    return true;
},

// ASSIGN FILE SYSTEM
onFSSuccess: function (fs) {

    salockFile.fileSystem = fs;

    alert("file system created!");
},

//ERROR
onError: function (e) {
    alert("error: " + e.toString());
},
```

图 5-34 构建文件管理系统对象

文件管理对象成功后则可以开始使用各种不同的功能了。首先主要使用到的是文件存储，也就向新的空文件写入相关的信息。本系统的签名样本记录过程会使用到该功能。File API 支持通过 ArrayBuffer 形式来想文件写入二进制码。由于存储的文件类型为 PNG 图片，因此写入的信息是对应从该图片的 Base64 码译码出的二进制码。doAppendFile 是存储图片文件的主调用函数。为了达到较好的调用效果，该函数中含 dir 与 code 参数，对应与存储文件的名称及路径和需要写入的目标 Base64 码。code 参数转发给 salockFile 的成员字符变量 imageCode 等待处理，而 dir 路径及文件名转发给 getFile 功能来进行文件写入操作。salockFile.fileSystem.root.getFile 选择第一个参数作为目标文件，从第二布尔型参数决定是否自动生成若该文件不存在，对该文件执行第三参数的 appendFile 函数，若处理出错则执行第四参数的 onError 函数。

appendFile 的任务为处理 dir 中的目标文件，本场合的业务为写入从 Base64 译码出的二进制码。需先将接受到的 Base64 码通过 imageFile 函数转换成二进制码并返回 ArrayBuffer 类型的对象，把该对象存入 file 变量。进行写文件之前必须通过文件对象 f 的 createWriter 函数构建文件写入器 writeOb。使用 writerOb.write(file) 语句把 ArrayBuffer 对象 file 写入对应的文件中。此操作完成会弹出写入成功信息。存储文件函数的定义如下图所示。

```
// WRITE FILE CALLER
doAppendFile: function (dir,code)
{
    salockFile.imageCode = code;

    salockFile.fileSystem.root.getFile(dir, {create:true}, salockFile.appendFile, salockFile.onError);
},

// WRITE FILE
appendFile: function (f) {

    // ArrayBuffer object of image code
    var file = salockFile.imageFile(salockFile.imageCode);

    // create writer
    f.createWriter(function(writerOb) {

        writerOb.onwrite=function() {

            salockFile.writelock = false;
        }

        // write binary to PNG
        writerOb.write(file);

    })
},
```

图 5-35 转换成二进制及存储 PNG 函数

5.8.3 Base64 码至二进制码转换

原始的 Base64 码当写入 PNG 文件是不可显示的。因此在写入前必须把 Base64 码转换成二进制码。译码工作由 base64DecToArr 函数完成，而 imageFile 负责转发接受到的 Base64 码 code 参数给 base64DecToArr 函数处理。调用 base64DecToArr(code).buffer 实现 Base64 至二进制译码过程且返回 ArrayBuffer 对象。将该对象存放至 myBuffer 变量并且当 imageFile 函数的返回值。这样此函数可作为写入文件函数的写入目标参数。

```
// RETURN BINARY ARRAY BUFFER
imageFile: function (code)
{
    // decode base64 to binary
    // return an ArrayBuffer object
    var myBuffer = base64DecToArr(code).buffer;

    return myBuffer;
},
```

图 5-36 调用二进制译码 imageFile 函数

使用从 Mozilla Developer Network 提供的 Base64 码译码至二进制码并返回 ArrayBuffer 对象的 base64DecToArr 函数实现代码如下：

```
// BASE 64 --> ARRAY BUFFER
function base64DecToArr (sBase64, nBlockSize) {

    var
    sB64Enc = sBase64.replace(/[^A-Za-z0-9\+\ \/]/g, ""), nInLen = sB64Enc.length,
    nOutLen = nBlockSize ? Math.ceil((nInLen * 3 + 1 >> 2) / nBlockSize) * nBlockSize : nInLen * 3 + 1 >> 2,
    taBytes = new Uint8Array(nOutLen);

    for (var nMod3, nMod4, nUint24 = 0, nOutIdx = 0, nInIdx = 0; nInIdx < nInLen; nInIdx++) {
        nMod4 = nInIdx & 3;
        nUint24 |= b64ToUint6(sB64Enc.charCodeAtAt(nInIdx)) << 18 - 6 * nMod4;
        if (nMod4 === 3 || nInLen - nInIdx === 1) {
            for (nMod3 = 0; nMod3 < 3 && nOutIdx < nOutLen; nMod3++, nOutIdx++) {
                taBytes[nOutIdx] = nUint24 >>> (16 >>> nMod3 & 24) & 255;
            }
            nUint24 = 0;
        }
    }

    return taBytes;
}
```

图 5-37 Base64 译码函数代码

5.8.4 引入签名认证系统的记录过程

实现文件处理功能的下一步为引入本系统的结构。首先的使用场合为样本记录。可利用数据库记录的过程直接在此函数内调用文件写入功能。主要调用地方有两个：存储完整的签名和存储签名中的各笔画。此存储过程与数据库记录两大部分同时执行。

存储签名过程在数据库对象中的 `successCharacter` 函数将调用到。选择此函数作为调用该功能的地方的原因在于 `successCharacter` 函数中才能获取到当前记录的签名名字，这与向签名文件命名过程有关。文件被存储之前必须获取到整个签名的 Base64 码。通过对接受签名的 canvas 处理器的加载区 `signature` 调用 `jSignature` 模块中的 `getData` 功能，取属于 `image` 类型，也就是 PNG 格式的 Base64 码。此 Base64 码由 `MyStroke` 的 `characterCode` 成员临时存储。含有 Base64 码后的 `characterCode` 是一串两元素的数组，每个元素是一个字符串。第 0 元素的内容为 Base64 码类型，本场和为“`image/png;base64`”，而第 1 元素为该签名的 Base64 码。译码及写入过程的目标参数是 `characterCode` 的第 1 个元素。调用 `salockFile` 对象中的 `doAppendFile` 函数时会给此函数传送存储的文件名及存储位置和需要译码的目标 Base64 码字符串。签名样本文件的存储目录是自定义当前目录中的 `character` 子目录，命名格式为<签名 id>.png。记录签名属性至数据库成功后的函数 `successCharacter` 中调用的存储前面文件代码如下图：

```
// GET CURRENT CHARACTER ID
salockDB.currentCharID = results.insertId;

MyStroke.characterCode = $("#signature").jSignature('getData', 'image');

// WRITE CHARACTER TO PNG
salockFile.doAppendFile(

    salockFile.rootdir +
    "character/" +
    salockDB.currentCharID +           // NAME
    ".png",

    MyStroke.characterCode[1]         // BASE64
);
```

图 5-38 引入存储签名文件

另一阶段，存储签名的各笔画文件在签名记录完成后开始执行。存储笔画函数在 salockDB 对象中的 insertStrokeDB 函数使用。与签名记录类似，在 insertStrokeDB 函数中可同时取得当前签名的 id 号及在数据栈属于该签名的笔画标记。以上两种信息将给生成的笔画 PNG 文件命名。由于 insertStrokeDB 是异步与 splitStroke 函数执行，因此存储笔画操作应与记录笔画信息至数据库操作同时执行。则 doAppendFile 的调用地方为 insertStrokeDB 中记录笔画的 while 循环中。每次循环获取到的当前笔画标记 currentStroke 会与签名 id 配合地向笔画文件命名。笔画图像的名字格式为<签名 id>_<笔画标记>.png，存放在自定义当前目录的 stroke 子目录内。写入文件的目标 Base64 码是本次循环从 MyStroke.Stroke.source 获得的 stroke 数组第 1 元素。存储笔画 PNG 图像文件的代码如下图。

```
// WRITE STROKE TO PNG
salockFile.doAppendFile(

    salockFile.rootdir +
    "stroke/" +
    _character_id +                     // NAME
    "_" +
    salockDB.currentStroke +
    ".png",

    MyStroke.Stroke.source[salockDB.currentStroke].stroke[1]); // BASE64
```

图 5-39 引入存储笔画文件

这样，签名的完整记录过程可明确分为两阶段而按序执行。把记录整个签名和记录每笔画划分成两件独立且按序的工作能提高签名样本记录的准确性，且使记录流程容易观察、管理及修正。

5.9 样本文件管理

样本管理是连接记录与比较两模块的主要部分，负责从获取到的数据可转换成作为对比样本的签名。

5.9.1 获取签名和笔画信息

在签名记录过程需要记录目前的签名数量，该变量将作为显示所有签名的数量参数。使

用 PhoneGap 的 LocalStorage 在插入新签名后存放签名数量至 keyamount 变量中。可把此操作定义在一个 getAmount 函数中，该函数在从数据库选取出所有签名信息时会执行，getAmount 的函数定义如下图。

```
// GET AMOUNT OF CHARACTER IN DB
getAmount: function (tx, results)
{
    // STORE AMOUNT OF CHARACTER IN LOCAL STORAGE
    window.localStorage.setItem("keyamount",results.rows.length);
},
```

图 5-40 获取签名数量函数 getAmount

样本管理中设置选择的签名作为密码操作中，关键的操作时如何通过选择的签名取出对应该签名的所有笔画文件。在这时候数据库与文件管理的关系及文件命名方式是很重要的元素。因此首先需要从数据库从指定签名提取相应的笔画信息。定义 strokeSelect 作为调用函数，执行 strSelect 函数从以选择的签名 currentID 取出所有属该签名的笔画信息。在输出结果后需要获取笔画的总数量来想文件管理系统提取正确的文件，该操作有 getStrokeFile 函数执行。笔画信息获取的流程实现代码由下图给出。

```
// RUN SELECT STROKE FROM CHARACTER
strokeSelect: function()
{
    // START SELECT
    salockDB.strokeDB.transaction(salockDB.strSelect, salockDB.errorCB);

    return true;
},

// DO SELECT STROKE FROM CHARACTER
strSelect: function (tx)
{
    var sql = 'SELECT * FROM StrokeDB WHERE character_id =' + window.localStorage.getItem("currentID");
    //alert(window.localStorage.getItem("currentID"));

    tx.executeSql(sql, [], salockDB.getStrokeFile, salockDB.errorCB );
},

// get stroke amount
getStrokeFile: function (tx, results)
{
    window.localStorage.setItem("currentCharlen",results.rows.length);

    alert("got char length!");
},
```

图 5-41 从指定签名获取相应的笔画信息函数

5.9.2 选择签名样本作为密码

对比样本更换由选择样本模块来负责。在 settings-selectcharacter.html 界面文件中定义初始化的显示函数 show。该函数从 keyamount 取得的签名数量会按序显示目前可用的签名样本。签名的加载元素 img 也取 id 等于签名的 id，也是签名的文件名。

下一步是在 show 函数实现选择特征。由于每次能选择的样本数量只唯一，因此选择到的签名也需要获取和存储该样本的 id。此 id 会作为重要参数来提取对应的笔画文件，立即在后续的 strokeSelect 函数调用中使用到。下图是 show 函数的完整实现代码。

```
// DISPLAY CHARACTER LIST WHEN LOADED
function show () {

    // AMOUNT OF KEY
    var amount = window.localStorage.getItem("keyamount");

    // LOOP
    for (var i = 1; i <= amount; i++){

        // GET IMAGE
        var url = "/mnt/sdcard/" + salockFile.rootdir + "character/" + i + ".png";

        // PUT IMAGE IN DYNAMIC DIV
        $("<img/>").attr({

            "src":url,
            id: i
        })
        .css({width: "50%"}).wrap("<div></div>").bind( "click",

            // COLOR TOGGLE
            function ()
            {
                if( this.style.backgroundColor = this.style.backgroundColor )
                {
                    // TRANSPARENT
                    this.style.backgroundColor = "";
                }

                else
                {
                    // SET OTHER IMG TO TRANSPARENT
                    $("#content img").css("background-color", "");

                    // SET SELETED ONE TO WHITE
                    this.style.backgroundColor = "ffffff";

                    // GET CURRENT ID
                    var charID = $(this).attr("id");

                    // STORE TO LOCAL STORAGE
                    window.localStorage.setItem("currentID",charID);

                    salockDB.strokeSelect();

                }

            }

        ).appendTo("#content");
    }
}
```

图 5-42 显示签名及签名信息传送函数 show

将提取出的签名笔画文件集作为对比样本在比较界面文件 compare.html 的定义如下图：


```
// TARGET ARRAY
var stroke2 = new Array();

var charID = window.localStorage.getItem("currentID");

var len = window.localStorage.getItem("currentCharlen");

for(var i=0; i<len; i++)
{
    stroke2.push("/mnt/sdcard/" + salockFile.rootdir + "stroke/" + charID + "_" + i + ".png");
}

// PUT TEMPLATE
MyStroke.Stroke.target = stroke2;
```

图 5-43 把提取出的签名设为对比样本

5.10 系统封装

系统实现的最后工作阶段是封装系统。使用 PhoneGap 框架将完整的系统文件包装成 APK 格式的文件。样本系统在封装工作结束后可正常以安卓手机应用形式在手机上运行。先访问 salock\platforms\android\res\values 路径中的 strings.xml 文件，在该文件内容中的应用显示名改为 SA Lock:

```
<string name="app_name">SA Lock</string>
```

使用 Eclipse 的 File/Export 把项目以安卓应用 APK 格式导出。把该 APK 文件安装至安卓智能手机上可运行并开始进行实验。样本系统的运作演示在下一章将介绍。

5.11 本章小结

第五章为签名认证系统具体实现过程的章节。本章描述属于个人的系统实现方法，将在第四章还属于理论类的系统实现化至可在智能手机上运行的一个应用。通过各模块的具体实现步骤，本章体现出该系统的可行性和可实现性。由于所设置的系统类型为安卓平台的认证系统，因此样本系统能在智能手机上运行是很关键的。实现过程的完成品将是在后期进行实验需要使用到的样本系统。

第六章 签名认证系统样本展示

6.1 样本系统模块

签名认证系统的功能可分成三大部分：记录、开锁与设置。

6.1.1 记录

该部分以划分和记录笔画属性技术为核心功能。首次使用该系统时，用户需先为自己设置属于个人的签名。系统会弹出记录签名界面，用户在该界面的 canvas 区域上进行签名输入。为了达到较高的安全性，本系统建议用户选择自己最熟悉及独有的签名。用户画了签名后点击 Record 按钮，签名被记录后再点击 Strokes 按钮来完成记录笔画过程。系统将从获取到的用户样本计算出用户的个人签名习惯，包含签名的笔画数量、笔画顺序及个笔画的持续时间。最终得出的图案，包含签名与签名的各笔画以 PNG 格式存储至本系统的内置文件管理。同时该样本以及所属的笔画信息也会记录至数据库中。该图案可作为用户每次开锁的比较样本中之一。

6.1.2 开锁

以划分笔画、比较笔画数量及笔画顺序为核心技术。设置个人的签名后，每次访问手机内部信息需要通过该系统的验证阶段。目前本系统只能以应用形式运行，但方便于测试系统的验证效果。用户需先在系统的开锁界面画出自己的签名密码，再点击 Unlock 按钮。系统接收用户的输入后先把签名划分成若干个笔画，把每笔画以 Base64 码格式与 PNG 样本的笔画数量与各笔画文件进行比较。比较的前阶段为先比较签名的笔画数量是否配对。第一阶段配合会进入第二阶段为比较个笔画的相似度。若比较得出的平均相似度比率结果不足以达到系统的要求，则验证失败。当开锁失败次数超过规定次数，系统会显示警告通知并把手机锁定若干个小时。若验证通过，用户可直接访问手机信息或进入该系统的设置面板。

6.1.3 设置

设置功能可提高签名认证系统的灵活性。系统的设置面板在用户验证通过后可立即访问。设置功能包括四方面：样本管理、汇报管理、锁住设置与系统设置。

- 样本管理：与解锁密码相关的功能，此功能允许用户设置自己的密码，具体为选择内置存储中作为解锁签名的样本、添加新签名与删除旧签名。
- 汇报管理：该功能记录每次验证失败的信息后将此信息存放至系统的本地数据库。用户可通过汇报管理功能以不同形式查看在不同时间段的验证错误情况汇报。
- 锁住设置：此功能提供手机锁住开关，以及允许的解锁失败次数和锁住时间长度设置。用户可灵活根据自己的需求设置适应的锁住设置来提高保密性。
- 系统设置：若期望关闭系统但不影响其他部分的设置，系统设置功能可解决该问题。用户可在非影响系统设置的条件下随意启动或关闭系统。

6.2 样本系统展示

系统界面设计使用目前流行的单色 Metro UI 风格。结合 Metro UI CSS 库与 HTML 语言可实现简洁、清晰与容易操作与阅读的应用界面。签名认证系统根据不同模块有不同的对应彩色。包含开锁界面：浅绿色、总设置面板：灰色、样本管理：蓝色、汇报管理：深绿色、锁住设置：紫色、系统设置：橙色。使用简洁设计风格，用户可容易通过点击各菜单项来使用系统的功能。

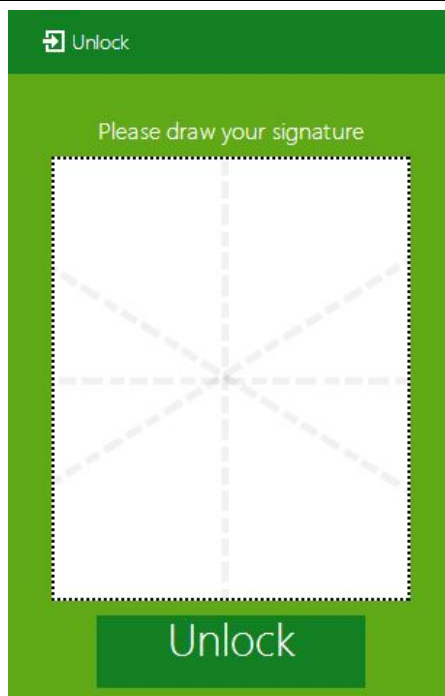


图 6-1 开锁界面

开锁界面：包含签名区域与当次开锁的允许开锁失败次数。系统将由验证成功或失败状态弹出对应的通知（验证成功通知或验证失败警告）。

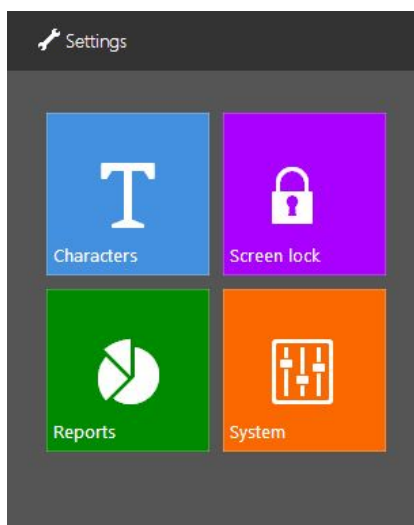


图 6-2 将笔画进一步细化成标记

总设置面板：显示四大设置功能，包含样本管理、汇报管理、锁住设置与系统设置。

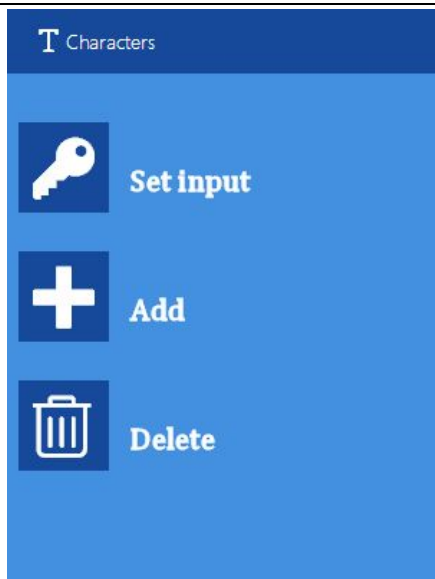


图 6-3 样本管理功能

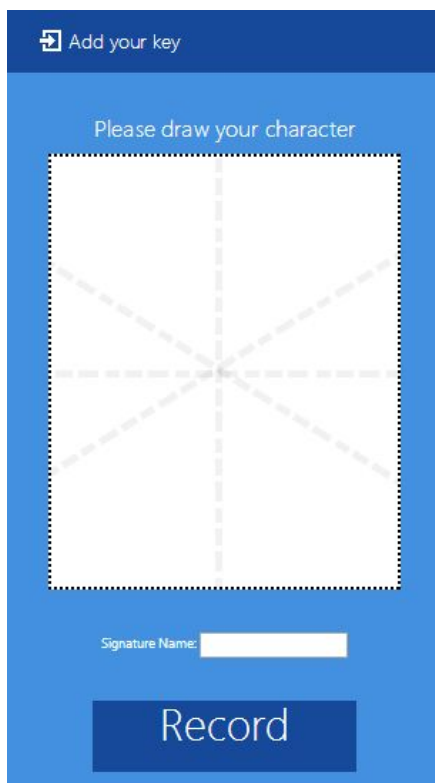


图 6-4 记录签名界面

样本管理：用户可通过点击 **Select**、**Add** 或 **Delete** 项来选择、添加或删除签名样本。密码库以 PNG 格式存放签名样本。签名样本列表的显示方式为缩略图，为使管理工作方便及容易操作。用户每次能添加一个签名且尽可选择一个样本作为系统的对比样本，或通过一次操作可批量删除旧签名样本。

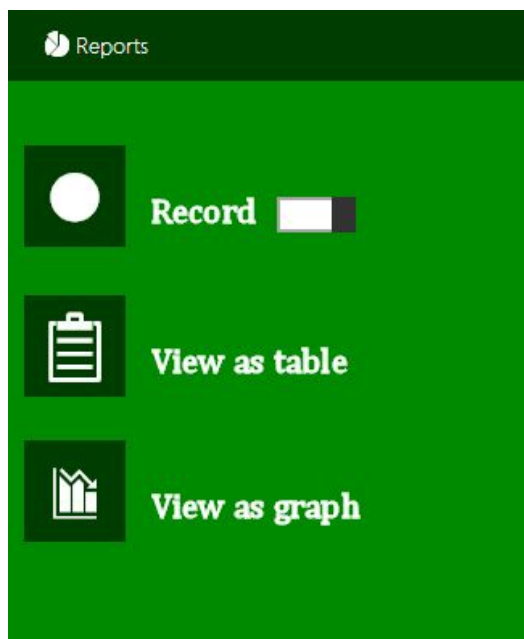


图 6-5 汇报管理

汇报管理：可随意开关验证失败记录功能。用户可选择不同的时间段查看相应的验证错误报告。错误报告可以表格（Table）或图表（Graph）形式显示，其中图表形式提供实际值型数据与比率值数据图表。

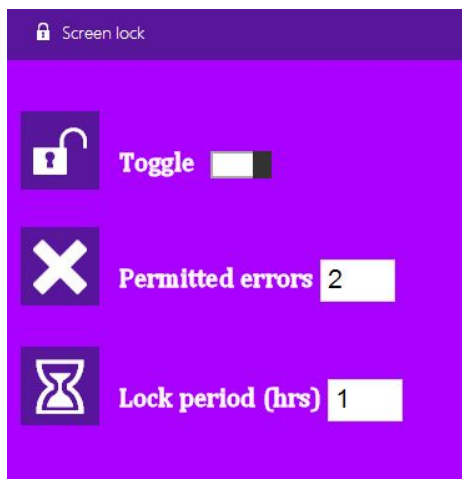


图 6-6 锁住设置

锁住设置：与汇报管理类似，用户也能随意开关手机锁住功能。该功能也能自定义每次开锁的验证错误允许次数以及锁定手机的时间长度，以小时和分钟为单位。用户选择后则生效。



图 6-7 系统开关设置

系统设置：提供切换式的系统开关，用户可通过滑动开关来使系统生效或无效。该操作不影响系统的其他设置。

6.3 本章小结

把第四章所提到的系统结构及功能，第五章所实现出的样本系统过一次演示和讲解是第六章的主要目的。本章以实际用场、实际应用和手机用户的角度来对样本系统的各模块结构及功能进行解释。后续的实验工作中，参与实验者也需要了解系统的使用方式，从而减少需要花费的时间和提高实验的效率。

第七章 系统认证实验与评估

7.1 系统实验

实验之前经过几次初步使用系统可看出签名中各笔画的相似度一般不小于 0.5，也不大于 4.0。于是以下的实验过程使用的允许平均相似度为 2.0，该数值在实验时有相对好的合理性。签名记录导出的 PNG 文件尺寸为 252x300 像素。

实验的场合可分成三种：用户本人自己开锁，肩窥攻击的攻击者开锁及污点攻击的攻击者开锁。每种场将使用同样的签名样本，让分别作为用户和攻击者的参与者尽量输入他们认为是最接近样本的签名。实验过程会记录下参与者输入的笔画数量，验证结果和平均相似率（笔画数量匹配场合下）。

7.2 系统实验结果

7.2.1 针对用户本人的实验

由手机用户本人设置签名密码，然后自己执行开锁操作。实验结果如表 7-1。

表 7-1 用户本人开锁的试验结果





签名样本	笔画数量	认证通过	错误类型	平均相似率
	4	否	笔画对比错误	0.02
		否	笔画对比错误	0.02
		否	笔画对比错误	0.02
	4	是	无	2.05
		是	无	2.01
		否	笔画对比错误	1.88
	3	是	无	2.83
		是	无	2.97
		是	无	2.59
	7	否	笔画对比错误	1.41
		否	笔画对比错误	1.31
		否	笔画对比错误	1.35

从以上数据表,可推出该系统对有短笔画的签名有较弱的认证能力。第一个签名样本中,每个笔画只是一个点,平均相似度均为 0.02,则认证总是失败。一般性的签名能达到相当好的效果,有微小的二型错误的概率。多笔画但存在短笔画的第四个签名样本认证效果还未达到最优,离通过认证的允许数值还有一段距离。

7.2.2 针对肩窥攻击的实验

在此场合,攻击者将有机会能观察到用户输入的密码,从而知道签名的图形。假设攻击者通过物理攻击得到用户手机并进行开锁操作。实验的数据结果如表 7-2。

表 7-2 肩窥攻击者的试验结果





签名样本	笔画数量	攻击者输入的 笔画数量	认证通过	错误类型	平均相似率
	4	2	否	笔画数量错误	无
		4	否	笔画对比错误	0.02
		4	否	笔画对比错误	0.02
	4	4	否	笔画对比错误	1.93
		4	否	笔画对比错误	1.91
		4	是	无	2.02
	3	4	否	笔画数量错误	无
		3	否	笔画对比错误	1.98
		3	是	无	2.23
	7	6	否	笔画数量错误	无
		6	否	笔画数量错误	无
		7	否	笔画对比错误	1.06

可看出有极短笔画的签名不适合作为对已样本。原因在于无独特特征,认证效果也是最低,无论是攻击者或用户本身都无法通过验证。第二和第三签名样本有出现一型错误的一定概率。有中等数量且较复杂笔画的签名,再配上笔画数量的比较机制对肩窥攻击能做出相当好的应付。对第四个签名样本而言,虽然目前对用户本身很容易出现二型错误,但要破锁该密码的攻击者容易遇到笔画数量错误问题。这样也体现出对于肩窥攻击,多笔画的复杂签名可使只通过一次观察获得的信息不足以完全准确。

7.2.3 针对污点攻击的实验

为了能最明显体现出该系统对污点攻击的抵抗力,本实验只考虑污点攻击的理想情况。用户先自己画出签名,攻击者在无观察到用户的情况下通过物理攻击获得用户的手机设备。接下再通过观察触摸屏上留下的污点来猜测开锁的密码。实验结果如表 7-3 所示。

表 7-3 污点攻击者的试验结果

签名样本	笔画数量	攻击者输入的 笔画数量	认证通过	错误类型	平均相似率
	4	2	否	笔画数量错误	无
		3	否	笔画数量错误	无
		3	否	笔画数量错误	无
	4	3	否	笔画数量错误	无
		3	否	笔画数量错误	无
		4	否	笔画对比错误	1.75
	3	4	否	笔画数量错误	无
		5	否	笔画数量错误	无
		3	是	无	2.10
	7	3	否	笔画数量错误	无
		4	否	笔画数量错误	无
		5	否	笔画数量错误	无

从以上的数据，笔画数量比较是该系统对污点攻击的最好应付方法。多数的攻击者通过观察触摸屏上的污点都无法能猜测出正确的笔画数量。在所有实验的签名样本中只有第三签名样本出现二型错误。此实验能表现本系统对污点攻击有一定的抵抗能力。

7.3 系统评估

目前，系统的图像识别准确率还未达到最好的效果，尤其是对含有段笔画签名的场合。攻击者经肩窥攻击多次验证仍然能在一定程度下通过验证。系统目前的一型及二型错误率还较高。但从该实验能清楚看出该系统的保密性比传统的 Password/PIN/Pattern 认证方式高，能够减低受污点攻击的概率。实验也同时展示出基于图像识别感知认证系统的运作方式。系统可识别本次输入的笔画数量与签名样本间的图像差别及差别率。主要的设置功能如样本管理、信息查看也能正常使用。

7.4 将来与展望

从设计该系统的角度来看，还有许多的可扩展之地。目前认证失败后触发像头的思想已经足以实现但自动照相问题仍然不能解决。完成该特征可使系统在每次验证错误会拍下攻击者的图像，当手机的拥有者访问时可知道攻击者的身份从而有适当的方案来维护自己的个人信息。关于记录样本方面，系统能记录用户通过一次输入的签名直接作为标记样本，而还未

支持机器学习特征。引入机器学习的功能可让系统变得更灵活和安全，能够接受多次输入的签名来学习用户的签名习惯，产生对每用户都不同的适当认证样本。另外开锁签名和样本签名间的比较过程可再进一步优化。目前将签名划分成每个笔画的单元可继续更详细的划分，利用隐马尔可夫模型可从签名中每笔画的方向、值、弯特性来转换成多个标记，如图 7-1^[36]。以笔画标记作为最小单元，属同一笔画的标记则属同一分组，把样本与本次签名的标记分组量、每分组的持续时间及分组内的标记顺序对比会使认证过程更加明确展示出用户签名习惯唯一性，从而提高验证效率。签名认证系统也可配合其他现行的生物类感知认证系统如指纹认证来提高安全性。

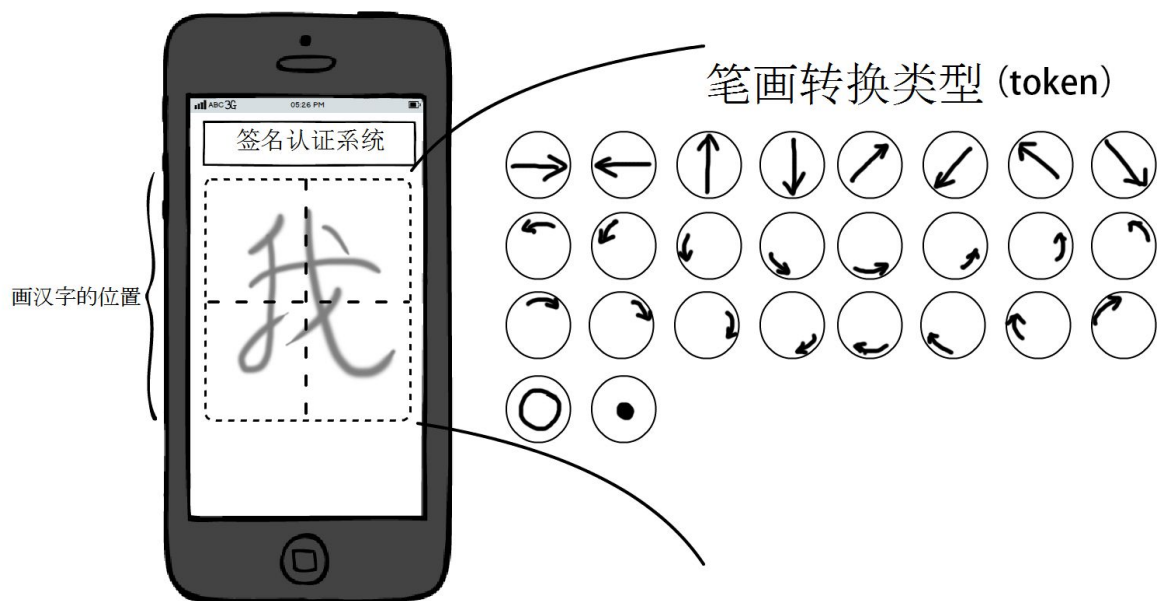


图 7-1 将笔画进一步细化成标记

7.5 本章小结

本章表现对实现出的样本系统进行各种不同场合的实验，从而检测系统对用户本身的可用性和对攻击者的抵抗性。获取到的数据是作为系统评估的主要元素。从分析系统在实际用场的的数据后也同时给出系统在未来的可扩展性。对系统继续加工和优化使系统在将来比目前更加健壮，可靠的展望也在第七章中提出。

第八章 结论

本论文通过介绍安卓平台的信息安全概况,再针对基于感知的认证系统这方向做具体的分析,包括攻击模式的分析及其他现行认证系统的分析,从而提出适当的新认证方案。根据原始系统设计思路,到最后系统的签名笔画识别认证、签名属性记录、签名样本管理、错误统计功能已经顺利的完成。通过使用 PhoneGap 开发软件框架已成功转移本系统从计算机系统至安卓智能手机平台上。在智能手机上所作的实验结果比起在计算机系统上有更加现实的性质,从而能对本系统做出正确的评估。本系统到目前为止能符合认证系统方面的独立性、唯一性、方便性及可扩展性需求。固然系统的笔画识别认证功能还未达到最优状态,但已经能部分体现出使用签名作为输入密码的优点。若对目前的系统再用心优化及添加更多的可设置功能可使该系统更近一步接近良好的质量。

通过本次的毕业设计,本人学习到了很多宝贵的知识。从本课题的研究背景和研究意义对安卓智能手机市场的信息安全情况有了充分的理解。同时通过分析基于感知的认证系统明白到一个认证系统需要满足的用户需求及基本运作方式。在构造自己的样本系统过程中学习了许多关于手机应用开发方面的程序设计技术,并且累计宝贵的经验。项目获得的实验结果也让本人清楚自己的设计思路还存在什么缺点,从而在未来可做出合理的性能优化和提升。

参考文献

- [1] Butler M. Android: changing the mobile landscape[J]. Pervasive Computing, IEEE, 2011, 10(1): 4-7.
- [2] Chen J, Advocate D. An introduction to android[J]. Google I/O, 2008.
- [3] Xuguang H. An Introduction to Android[J]. Database Lab. Inha Univeristy, 2009.
- [4] La Polla M, Martinelli F, Sgandurra D. A survey on security for mobile devices[J]. Communications Surveys & Tutorials, IEEE, 2013, 15(1): 446-471.
- [5] Becher M, Freiling F C, Hoffmann J, et al. Mobile security catching up? revealing the nuts and bolts of the security of mobile devices[C]//Security and Privacy (SP), 2011 IEEE Symposium on. IEEE, 2011: 96-111.
- [6] Chin E, Felt A P, Sekar V, et al. Measuring user confidence in smartphone security and privacy[C]//Proceedings of the Eighth Symposium on Usable Privacy and Security. ACM, 2012: 1.
- [7] Li, Bo, and Eul Gyu Im. "Smartphone, promising battlefield for hackers." Journal of Security Engineering 8.1 (2011): 89-110.
- [8] Muslukhov I. Survey: Data Protection in Smartphones Against Physical Threats[J]. Term Project Papers on Mobile Security. University of British Columbia, 2012.
- [9] Wright S. The Symantec Smartphone Honey Stick Project[J]. Symantec Corporation, Mar, 2012.
- [10] Lee, SangJun, and SeungBae Park. "Mobile password system for enhancing usability-guaranteed security in mobile phone banking." Web and Communication Technologies and Internet-Related Social Issues-HSI 2005. Springer Berlin Heidelberg, 2005. 66-74.
- [11] Hafiz, Muhammad Daniel, et al. "Towards identifying usability and security features of graphical password in knowledge based authentication technique." Modeling & Simulation, 2008. AICMS 08. Second Asia International Conference on. IEEE, 2008.
- [12] Finn, David, and Dennis Ryan. "RFID token with multiple interface controller." U.S. Patent No. 7,762,470. 27 Jul. 2010.
- [13] Tian J, Qu C, Xu W, et al. KinWrite: Handwriting-Based Authentication Using Kinect[C]//Proc. of NDSS. 2013, 13.
- [14] Wikipedia contributors. "Type I and type II errors." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 27 May. 2014. Web. 5 Jun. 2014.
- [15] 刘建奇, and 王以刚. "智能手机信息安全防范系统设计与研究." 信息安全与通信保密 2 (2007): 111-112.
- [16] Shin K I, Park J S, Lee J Y, et al. Design and implementation of improved authentication system for android smartphone users[C]//Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on. IEEE, 2012: 704-707.
- [17] Dorflinger, T., et al. "'My smartphone is a safe!'" The user's point of view regarding novel authentication methods and gradual security levels on smartphones." Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on. IEEE,

2010.

- [18] Oakley, Ian, and Andrea Bianchi. "Multi-touch passwords for mobile device access." Proceedings of the 2012 ACM Conference on Ubiquitous Computing. ACM, 2012.
- [19] Matalytski, Siarhei. "System and Methodology Protecting Against Key Logger Spyware." U.S. Patent Application 11/308,506.
- [20] Kita, Yoshihiro, et al. "Proposal and its Evaluation of a Shoulder-Surfing Attack Resistant Authentication Method: Secret Tap with Double Shift." International Journal of Cyber-Security and Digital Forensics (IJCSDF) 2.1 (2013): 48-55.
- [21] Lashkari A H, Farmand S, Zakaria D, et al. Shoulder Surfing attack in graphical password authentication[J]. arXiv preprint arXiv:0912.0951, 2009.
- [22] Wiedenbeck, Susan, et al. "Design and evaluation of a shoulder-surfing resistant graphical password scheme." Proceedings of the working conference on Advanced visual interfaces. ACM, 2006.
- [23] Von Zezschwitz, Emanuel, et al. "Making graphic-based authentication secure against smudge attacks." Proceedings of the 2013 international conference on Intelligent user interfaces. ACM, 2013.
- [24] Aviv A J, Gibson K, Mossop E, et al. Smudge attacks on smartphone touch screens[C]//Proceedings of the 4th USENIX conference on Offensive technologies. USENIX Association, 2010: 1-7.
- [25] Zhang, Yang, et al. "Fingerprint attack against touch-enabled devices." Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices. ACM, 2012.
- [26] De Luca, Alexander, et al. "Touch me once and i know it's you!: implicit authentication based on touch screen patterns." Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems. ACM, 2012.
- [27] Kim Y G, Jun M S. A design of user authentication system using QR code identifying method[C]//Computer Sciences and Convergence Information Technology (ICCIT), 2011 6th International Conference on. IEEE, 2011: 31-35.
- [28] Feng T, Prakash V, Shi W. Touch panel with integrated fingerprint sensors based user identity management[C]//Technologies for Homeland Security (HST), 2013 IEEE International Conference on. IEEE, 2013: 154-160.
- [29] Cheng, Kwang-Ting, and Yi-Chu Wang. "Using mobile GPU for general-purpose computing—a case study of face recognition on smartphones." VLSI Design, Automation and Test (VLSI-DAT), 2011 International Symposium on. IEEE, 2011.
- [30] Trewin S, Swart C, Koved L, et al. Biometric authentication on a mobile device: a study of user effort, error and task disruption[C]//Proceedings of the 28th Annual Computer Security Applications Conference. ACM, 2012: 159-168.
- [31] Di Gesu, Vito, and Valery Starovoitov. "Distance-based functions for image comparison." Pattern Recognition Letters 20.2 (1999): 207-214.
- [32] Josefsson, Simon. "The base16, base32, and base64 data encodings." (2006).
- [33] Wikipedia contributors. "Base64." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 3 Jun. 2014. Web. 5 Jun. 2014.
- [34] Wikipedia contributors. "PhoneGap." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 30 May. 2014. Web. 5 Jun. 2014.

- [35] Rogers, Rick, et al. Android application development: Programming with the Google SDK. O'Reilly Media, Inc., 2009.
- [36] Hu, Jianying, Michael K. Brown, and William Turin. "HMM based online handwriting recognition." Pattern Analysis and Machine Intelligence, IEEE Transactions on 18.10 (1996): 1039-1045.

谢辞

本论文能够顺利完成也多得曹珍富老师与朱浩瑾老师在百忙中的精心指导之下。全靠老师们热情指导态度，追求完美的精神及宝贵的建议，原始还出现很多问题的论文才能达到目前的良好状态。各位老师的优秀工作风格很深意地让本人觉得自豪。在此谨向尊敬的曹老师和朱老师表达本人的谢意和敬意。

在做毕业设计的这段时间内，也多得实验室内大家做实验的功劳，本人的认证系统实验阶段才能够完成并得出最后的系统评估。多谢各位学长、学姐及同学对本人一直以来的支持与帮助。

如果没有大家的帮助本人是不能这么顺利地自己的毕业设计论文。我再次真心的感谢各位尊敬的老师、学长、学姐及同学在本人做毕业设计这段时间内的无言帮助与支持。谢谢你们！

PERCEPTIVE ANDROID AUTHENTICATION SYSTEM RESEARCH

In recent years, smartphones have become more and more common in many people's daily life. Possessing the ability nearly equals to the classic and traditional computer system, smartphones are capable of executing several complicated tasks in form of a compact-size device. One common of many smartphone platforms is Android, which was created by Google. Many Android smartphone users treat their devices as their own personal computers, which mean a lot of private information is stored on their own phones. Users' privacy regarding their sensitive and private information leads to many cases of security attack, which the most common one is information stealing. Usually most Android smartphones provide the Password/PIN/Pattern as the default perceptive authentication system each time a person tries to unlock the phone. This method is easily vulnerable to attack models such as shoulder surfing attack and smudge attack, which attackers can obtain the content of password through observing user or smudges on the phone's touchscreen. Several reports have shown that users' information was leaked or lost despite of using the Password/PIN/Pattern system to protect their phones against thieves. Hence, this thesis introduces a new authentication method as the alternative to the traditional smartphones protection system. Signature Authentication lock, or SAlock, is the new authentication system that will be mentioned here. This system uses users' signature as the input password and the authenticating method is performed in a special way that it is possible to repel some attack models since the password is harder for attackers to memorize and imitate.

A compact and convenient device, or a smartphone to be clear, which is carried along with a person on almost time of a day and stores a lot of personal information inside can easily be the target of smartphone thieves. From the results of many surveys which were carried out from 2011 to 2013, almost 60% smartphone users state that they do store a lot of personal private information on their phones, but are not willing to perform tasks such as online shopping, bank account checking, etc. The above two facts prove that if an Android smartphone is lost, the chances of private information being revealed to anonymous people is pretty high. Physical attack such as phone stealing was one of the most serious threats to smartphone security back in 2011. Furthermore, in 2012, the rate of successfully finding back a lost phone in the U.S was only 7%, making how to protect personal information from attackers once their phones are lost or stolen a big concern for lots of smartphone users.

Even though most of smartphones nowadays present the solution of protecting the content inside the phone from being accessed from anonymous user, the current authentication systems still have many problems. The password types that are used in the common Password/PIN/Pattern authentication system have the advantages of being short, easy to enter and easy to remember. However, this also applies to attackers, who may be able to get the users' password content through observing them. Simply looking at the user from the back or even recording the progress of entering the phone's password is called shoulder surfing attack. This attack model is easy to

perform yet quite effective, since attacker can easily hack into a phone by entering the password that they just got. Another attack model is smudge attack, which is truly useful in extracting content of graphical password such as Pattern password. Using the combination of different light and camera angles, attackers can obtain the remained smudges on the touch screen of the phone which were left by user's fingerprints. Base on different circumstances, the rate of successfully retrieving the whole content of password is different. In ideal cases, where users entered the password to unlock the phone and left it to be attacked, experiments showed that the Pattern password can be fully retrievable. Even though in other cases when the password can only partly retrieved, attackers can easily guess the content of full password based on what they got. The simple mechanism of Pattern password makes it vulnerable to smudge attack including guessing the password. The same crisis also applies to PIN password since it only contains a 4-digit number.

Currently a lot of new authentication methods have been considered as an alternative solution to Password/PIN/Pattern system. The notable type among these is biometric authentication system based on identifying users' face, voice, fingerprint, etc. Biometric authentication differs from the knowledge-based password such as Password/PIN/Pattern, where the system verifies what the user know, or the ownership of the password. Such way of verifying is not unique because anyone who has the password can get through the authentication step. Using users' unique characteristic such as fingerprint, face proves a safer solution to against smartphone attackers. However, the trade-off between high security and convenience of an authentication system is difficult to solve. Several users express that they feel it is very inconvenient to use biometric method to identify themselves each time they open the phones. Longer time consumption than knowledge-based password, along with the high frequency of using screen lock feature of smartphone makes those common biometric authentication method not the best choice for personal security on smartphones. Furthermore, most of biometric authentication systems are still in the early stage of development, which means the accuracy of identifying users' password is still low and not meet the requirements of many users. Authentication methods such as facial-based, voice-based also have the disadvantages of being relied to devices such as microphones or cameras, which can create noises on different circumstances when users input their password.

Creating an authentication system that can satisfy the requirements of convenience, independent and high accuracy of identification is the purpose of this thesis. How to let the system learns down the unique properties of the users probably is the best solution to increase reliability of the system. Cutting down the dependencies to other devices also increase the accuracy and convenience of the system. Hence, using users' signature as their input password is considered here. Even for the same signature, every person's way of drawing a signature is different. The designed system, SALock will record down the users' input signature as the password template. This template later will be used in the comparison with the input signatures when users attempt to unlock the phones. Drawing signature only needs the interactions between user and touch screen, which means it is no different from the classic Password/PIN/Pattern method. This solves the convenience problem for users. The signature recording and comparison method of SALock is performed on stroke level instead of whole-signature level which can possess a better ability to defend against attack models such as shoulder surfing attack and smudge attack.

SALock's system structure can be divided into three layers: view, processing and data. View layer mainly includes all the user interfaces which users can use to interact with the system, this

also is the layer where the features of the system are displayed. The major ones are signature recording, signature comparing and system setting features. View layer will obtain the user's input data, which can be the users' signature or settings values, and transfer it to the processing layer, which will analyze, process it and store it to the data layer. The processing layer, mainly contains the stroke-splitting module, stroke-comparing module and data-analyzing module. All the input and output data will be processed here and this layer uses bidirectional transfer to send raw and processed data between view and data layer. Last but not least, the data layer that consists of database and file management modules, is the area where all data is stored and can be extracted to use for many purposes including password comparing, and information selection.

How the signature data being recorded and compared is what makes SAlock different from other biometric authentication systems. Both recording and comparing tasks are performed on stroke level, not the while signature level. That means when users input signature and want it to be recorded, the input signature will first be sent to the stroke-splitting module to split the signature to separated strokes, which a stroke is an image unit when user draw on the touch screen in the condition of finger not leaving the screen. Then, data of both whole signature and all strokes that belong to that signature will be saved to the database and file management in the form of PNG image. Once system has the template, users can freely choose which signature they want to be as the comparing template when unlock. Unlock feature of SAlock uses the stroke unit-comparing solution, which when user input their signature, it will first be transferred to the splitting module to divide it into a number of strokes. Since the amount of strokes inside a signature reflects the unique habit of drawing signature of a user, the system will first compare the quantity of strokes, if the quantity is not equal then the authenticating process will result in fail status. This can save the time for redundant processing jobs and reduce the system pressure. When the first stage, which the quantity of source and template signature matches, passed, password comparing process will enter the second phrase, which is comparing the similarities and sequences of the strokes. Step by step, each stroke from the source signature will be selected and compared with the corresponding stroke of the template signature. A match rate will be generated as the similarity result for each pair of strokes. Finally the system will calculate the average value of match rate and check if it is bigger than the permitted value or not. If the average match rate is larger the then authentication process is success, or else the system will return an error message together with resetting the signature drawing area, users then will have to draw the signature again. About the algorithm of comparing, SAlock system uses the image comparison method. The comparison module will scan the pixels of two input images to see if they are similar in color structure or not. A pixel is constructed from 4 elements including R, G, B, A. For each cycle, system will jump from the n pixel to the $n+1$ pixel by an amount equal to pixel offset value and compare the R, G, B, A values of two pixels. If the one of these values is different, then an error pixel is generated and mismatch rate will increase by 1. Finally the mismatch ratio will be calculated by comparing the amount of error pixels with the total pixel amount of the image. Along with the mismatch rate, each pair of stroke after the comparison process will also generate an image which shows the differences between two images, these images are the aggregation of all the previous error pixels. Providing these result images not only clearly demonstrates the degree of similarity between two strokes, but also helps the result of experiment period to be clearer and easier to sum up.

As the experiments were carried out, the results showed that SAlock still having problem identifying signatures that have short length strokes. Short length strokes comparing will result in

low match rate even if the stroke is correctly drawn, affecting the overall authentication for that signature. Using 4 random signatures as the authenticating template, participated users have 50% chance to pass the authentication, which means the false negative rate is 50%. In the test against shoulder surfing attack, attacker is less likely to guess the signature right when stroke quantity of a signature is high, results in stroke amount error. Attackers using this attack model still have around 17% of chance to pass through the system, which is also the false positive rate, for those signatures that have high match rate for the user. For the case of smudge attack, the system proves its effectiveness against this attack model with the stroke quantity comparison. Based on the smudges that were left on the touch screen of the phones, attackers in most of the cases cannot guess the stroke amount of a signature correctly. For signatures that have high amount of strokes, the accuracy of guessing is even lower. Usually attackers only have at most one third chance to enter a signature that matches the stroke amount, along with around 8% of chance to pass through the system.

Since SAllock is still in the early stage of development, the system still has many problems to overcome at the moment. The identification for short length strokes is still not perfect, and the balance between time consumption of authentication process and safety of system is still a difficult challenge. However, the idea of comparing signatures on stroke units has proven the effectiveness in resisting smudge attack. Using signatures that are stored and compared in stroke unit also make the input password harder to memorize through observing method in shoulder surfing attack model in compared with the Password/PIN/Pattern system. In future, implementing a better signature comparison algorithm will result in better identification ability. Adding other features such as auto photo capturing when authentication fails to get the image of the attacker is also a good idea to strengthen the usability of the system.

Key words: Android, smartphone, perceptive authentication, shoulder surfing attack, smudge attack, signature authentication system