**电子科技大学大学生创新训练项目  
结题报告**

项目名称：基于生物特征信息以及弱口令的密码生成器.

项目负责人：　　 石崇宇 　　 　　 　　 　　.

学 号：　　 2016220503003　　 　　 　　　.

联系电话：　　 15520772197　　 　　 　　 　 .  
所在学院：　　 信息与软件工程学院 　　 .

指导教师：　　 熊虎　　 　　 　　 　　 　　 .  
申请金额：　　 1000 　　 　　 　　 　　 　　 .  
申请日期：　　 2017.03　 　　 　　 　　 　　 .

电子科技大学教务处

**摘要**

本项目是以面部特征识别以及基于面部生物特征信息的加密为核心的密码生成器，主要是针对我们每个人大量的网站口令使用问题：弱口令易被破解，强口令难以记忆。

本项目所实现的面部特征识别算法以及面部特征信息加密算法均基于Dlib框架，该框架该框架集成了大量的机器学习算法，包括了一些已经完成训练的模型。经过测试，本项目的面部特征识别算法的识别率高达97.5%，算法实用性较高，满足本项目需求。

本项目的后端采用Java的Spring boot框架，共分为实现具体功能的controller层，实现各种方法类的until层，实现与数据库接口的respositoy层，实现事务管理的service层。

本项目的前端使用工具Android Studio进行开发，项目前端一共由四个Activity，工具类Util.java，用户全局变量User组成，四个Activity分别是SplashActivity启动界面，loginActivity登录界面，CameraActivity人脸图像界面，workActivity工作界面。

本项目现在已经顺利完成，已完成项目预期的功能。

关键词：人脸识别；生物特征；弱口令；强口令；

# 1 绪论

在现代日常生活中，上网已成为我们生活中重要的一部分，所以网站口令的使用是非常广泛和频繁的。但是在网站口令设置方面存在这样一个问题，如果我们使用一样的口令登陆我们所使用的每一个网站，那么一个网站口令的泄漏就意味着所有网站的登陆口令的保密性都将失去，这样的做法对于安全需求不同的各个网站来说（如安全要求高的支付宝网站与安全要求较低的小游戏网站），很明显是不适用的。

如果每次在网站注册时都使用强口令作为登陆口令，且注册的每个网站使用的强口令都不一样，那么，记忆这些复杂的强口令对于一般人来说是一个很大的难题。

诚然，现在的电脑和手机上都可以帮你在注册网站时生成一个毫无规律的强口令作为登陆口令并自动记住该口令与该网站匹配，并且在你再次登陆该网站时进行自动填充，无需我们记住复杂的强口令。但是，如果我们跟换手机或电脑，那么，我们在上一台手机或电脑上储存的密码也无法跟随转移到新的设备上，甚至，如果我们的设备丢失，那么我们的口令也就跟随着丢失，会造成我们重要的隐私丢失，甚至造成财产上不可挽回的损失。

## 1.1项目背景

在这个互联网高速发展的时代，我们每个人都无可避免的注册大量的网站和手机APP，这些账号的密码管理就成了一个令人头痛的问题。

大多数人们在注册时通常会选择有规律的弱口令密码，如123456，等等，这些密码毫无疑问是不安全的，乌云平台上提交的关于弱口令的漏洞多达778个，网上的弱口令暴力破解更是数不胜数，所以弱口令是不安全的。

然而强口令的复杂程度远超弱口令，不易被破解，但是众多的强口令导致记忆这些强口令是一件异常困难的事情。

正因为这些，本项目的基于生物特征的以及弱口令的密码生成器便是为了解决这些难题。我们的密码生成器通过生物特征，网站名称，弱口令生成一个复杂程度较高的，安全的，毫无规律的强口令，当你需要再次使用密码时，只需要输入网址和特定的弱口令即可等到强口令。

## 1.2 项目基础

经过组员讨论，我们决定采用如下图的流程来实现我们的的目标

|  |
| --- |
| 图一 密码生成器工作流程图 |

以下是搭建本项目的过程中所涉及的编程语言及相关技术

Springboot：Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。通过这种方式，Spring Boot致力于在蓬勃发展的快速应用开发领域(rapid application development)成为领导者。我们后台的就采用的Springboot框架。

MySQL：MySQL是一个关系型数据库管理系统，由瑞典MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。MySQL所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

Dlib

# 2 项目开发

## 2.1 人脸识别及基于生物特征的加密设计

在本项目中，采用的生物特征为独特的面部数据，采用了面部生物特征检测及面部特征信息识别比对技术。本次选择使用的是一个Dlib的框架，首先是由于该框架集成了大量的机器学习算法，包括了一些已经完成训练的模型（如本项目中使用的“shape\_predictor\_68\_face\_landmarks”，以及“dlib\_face\_recognition\_resnet\_model”）此外，Dlib通过干净和现代的C ++ API实现高度模块化，快速执行和简单易用。

在对生物特征数据处理中，采用了两步完成计算机对面部数据的检测，以及识别。

在面部检测中，我们使用的模型是“shape\_predictor\_68\_face\_landmarks”，这是一个由Dlib框架完成训练的模型，用于检测面部，并将面部的大概轮廓进行框选。

|  |
| --- |
| /\*模型的引入  detector = dlib.get\_frontal\_face\_detector()  shape\_predictor = dlib.shape\_predictor(predictor\_path)  /\*图片读入以及三原色的转换  img = cv2.imread(img\_path, cv2.IMREAD\_COLOR)  b, g, r = cv2.split(img)  img2 = cv2.merge([r, g, b])  /\*使用模型对面部的检测  dets = detector(img, 1) |

由于该模型使用的三原色顺序为[R, B, G]；但是使用opencv对读入的图片，其三原色顺序为[B, G, R]；故我们需要对顺序进行转换，以用于对面部轮廓的检测。

第二步，我们需要提取出面部的特征点，我们使用的模型是“dlib\_face\_recognition\_resnet\_model”，这是一个由Dlib框架完成训练的模型，用于对面部特征进行识别并进行标注，最后将返回标注点的特征坐标。

|  |
| --- |
| /\*提取出刚刚面部检测的每一张脸  for index, face in enumerate(dets):  /\*提取出面部特征点的坐标  shape = shape\_predictor(img2, face)  /\*将面部特征点重新计算出128维向量值  face\_descriptor = face\_rec\_model.compute\_face\_descriptor(img2, shape) |

在完成对面部数据的处理后，我们需要对其进行人的识别，让计算机通过计算的方式，完成对人的识别。在查阅相关资料后发现，现在使用的比较多的是通过机算其欧拉距离，判断其是否为同一个人的脸部特征数据。我们在通过计算，得到了128维的的坐标向量，刚好能够运用其中。

|  |
| --- |
| /\*欧几里德距离是指多维空间两点间的距离，这是一种用直尺测量出来的距离。如果将两个点分别标记为(p1,p2,p3....pn)和(q1,q2,q3.....qn),则欧几里德距离的计算公式为：  = (1)  /\*欧几里德距离评价[算法](http://lib.csdn.net/base/31" \t "_blank" \o "算法与数据结构知识库)是一个非常简单的计算相似度评价值的方法。它以经过人们一致评价的物品为坐标轴，然后将参与评价的人绘制到图上，并考查他们彼此间的距离远近。  图二 欧几里德评价值  /\*通过查阅文献，欧几里德距离评价，在其距离小于0.6时，即可认为是同一个人，故我们可以进行相关运算。  /\*我们计算距离的方式为，将两张图片的128维坐标一一取出，此外，计算其欧拉距离  for i in xrange(len(data1)):  diff += (data1[i] - data2[i])\*\*2  diff = numpy.sqrt(diff)  /\*最后返回的即使一个欧拉距离值，只要经过简单判断即可完成 |

虽然通过使用Dlib的框架，能简单的实现相关的功能；但是，由于模型建立的简单，没有经过特别训练，所以效果并不是特别完美。在此模型下，受到外界因素的影响较大（如光线，环境，分辨度等），故此，后面的工作将会重新完成自己模型的一个训练，以减少外界因素对准确度的影响。

加密算法的构造本次使用的是标准函数Hmac，Hmac是密钥相关的[哈希运算](https://baike.baidu.com/item/%E5%93%88%E5%B8%8C%E8%BF%90%E7%AE%97/10557139" \t "_blank)消息认证码；HMAC运算利用哈希算法，以一个密钥和一个消息为输入，生成一个消息摘要作为输出。

加密过程：通过将面部特征点转换为字节（bytes），作为密钥；输入的弱口令，使用字符串的形式作为消息，进行第一次Hmac加密；此后，通过第一次加密得到的消息验证码，转换为hash值，作为消息；输入的网址使用字符串形式作为密钥，进行第二次加密，得到一个消息验证码。最后采用base64的增长加密，产生出最后的密码，并输出。

|  |
| --- |
| str3 = bytes(data1)  h = hmac.new(str3)  hc = hmac.new(sys.argv[2])  hc.update(str3)  hash\_bytes = hmac.new(sys.argv[3],hc.hexdigest())  hash\_str = hash\_bytes.hexdigest()  #print(hash\_bytes.hexdigest())  base64\_str = base64.urlsafe\_b64encode(hash\_str) |

## 2.2 后台程序设计

后台整体框架：

|  |
| --- |
| 图三 后台整体框架 |

## 2.2.1 controller层

controller层主要是实现各种功能，在该项目中，各controller实现功能如下：

Logincontroller：登录与注册功能。

Mailcontroller：发送邮件功能，以及验证邮件验证码功能。

Phtotcontroller：人脸验证功能，生物特征加密功能。

Usercontroller：修改密码功能。

Webcontroller：添加网址功能，修改网址密码功能。

controller层的部分关键代码展示；

|  |
| --- |
| pyFace p = **new** pyFace(); String result = p.getFaceString(oldimagepath,changenew);*//通过人脸识别算法验证是不是该用户* **if** (!result.equals(**"Ture"**)){  **return "false"**; *//识别失败则返回false* }**else** {  *//对该User查找添加Web* webadd1 = **webRepository**.findByWebaddsAndUser(webadds1, user2);  **if** (webadd1 == **null**) {  *//如果无该web的密码。则添加，并获取密码* String shuiji = Miuntil.*getRandomString*(6);  String finalworder=(worderde+shuiji).toString();  webadd.setWorder(worder);*//添加order* webadd.setWebadd(w1ebadds1);*//添加web地址* webadd.setUser(user2);*//添加web所属User* webadd.setShuiji(shuiji);  String finalmima = mima.getmima(oldimagepath,finalworder,webadds1de);*//通过加密算法得到密码* finalmima= finalmima.substring(2,14);*//截取13位的密码作为最后密码* finalmima=AES.*encrypt*(key,finalmima);*//* webadd.setMima(finalmima);*//* **webRepository**.save(webadd);  **return** webadd.getMima();  }**else** {*//如果有该web，则返回该web密码* webadd = **webRepository**.findMimaByWebaddsAndUser(webadds1, user2);*//根据网址和用户查找* **if**(webadd.getWorder().equals(worder)){  **return** webadd.getMima();*//取得密码* }**else** {  **return "orderError"**;*//如果错误则返回orderError* } |

## 2.2.2 repository层

Repository层的功能是实现与数据库的接口，实现一系列的数据库操作

UserRepository：针对User实体类的数据库的操作。

WebaddRepository：针对Webadd实体类的数据的操作。

部分代码展示如下：

|  |
| --- |
| User findByName(String name); *//通过用户名查找User* User findPassWordByName(String name);*//通过用户名查找Password* User findPhotoByName(String name);*//通过用户名查找照片* User findByPassword(String password);*//通过Password查找User* User findByMail(String mail);*//通过邮件地址查找User* **void** deleteByName(String name);*//通过用户名删除User* **void** deleteById(Integer id);*//通过Id删除User* User save(User user);*//保存User* @Modifying @Transactional @Query(value = **"update User c set c.password = :password where c.name = :name"**,nativeQuery = **true**) **int** updatePasswordByName(@Param(**"name"**) String name, @Param(**"password"**) String password);*//通过用户名更新password* |

如findByNAme，findPassWordByName为数据库查找操作，save为数据库的保存操作，deleteByName，deleteById为数据库删除操作。

## 2.2.3 until层

该层主要包括了各种工具类，包括随机生成验证码类，随机生成6位字符串类，AES加解密类，Java命令行调用python程序进行面部特征信息识别的类等等。

该层中各类的名称及功能如下

AES：AES的加密和解密。

ChangeImage：对照片进行调整，以便进行人脸识别。

Getmima：根据用户的生物特征和弱口令以及网址进行加密，得到密码。

Miuntil：得到六位随机的字符串。

pyFace：Java命令行调用Python程序进行人脸识别。

RandomUtil：得到四位随机数字。

关键代码展示如下；

|  |
| --- |
| String[] arguments = **new** String[]{**"python"**,**"/root/InnovationProject/first.py"**,oldimage,newImage};*//要调用的程序类型，路径以及所需参数* **try**{  Process process = Runtime.*getRuntime*().exec(arguments);*//调用python程序* BufferedReader in = **new** BufferedReader(**new** InputStreamReader(process.getInputStream()));*//得到结果输出流* String line = **null**;  **while** ((line = in.readLine()) != **null**) {  **result** = line; *//转化结果格式* }  *//System.out.println(result);* in.close();  **int** re = process.waitFor();  **return result**;*//输出结果* }**catch**(Exception e){  e.printStackTrace();  **return result**;  } } |

## 2.2.4 后端API接口设计

/login 登录

/adduser 注册

/email/sendTemplateEmail 发送邮件

/email/yz 验证邮件验证码

/image/yzimage 人脸识别及生成密码

/image/setimage3 存储第一张图片

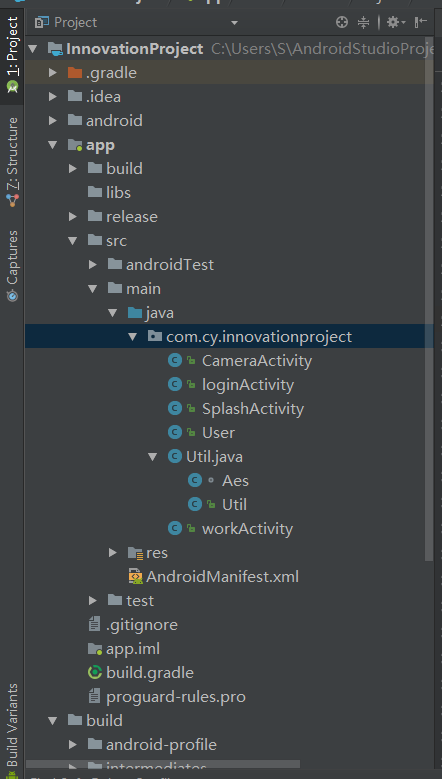
/user/updatepass 更改用户密码

/webadd/updatemima 更改网站密码

## 2.3 前端设计

前端为安卓app，使用工具Android Studio进行开发，Android Studio为谷歌推出的Android集成开发工具，支持Windows、Mac、Linux等操作系统，基于流行的Java语言集成开发环境IntelliJ搭建而成。

整个项目的结构如下图所示：



图四 前端整体框架

项目前端一共由四个Activity，工具类Util.java，用户全局变量User组成，四个Activity分别是SplashActivity启动界面，loginActivity登录界面，CameraActivity人脸图像界面，workActivity工作界面。

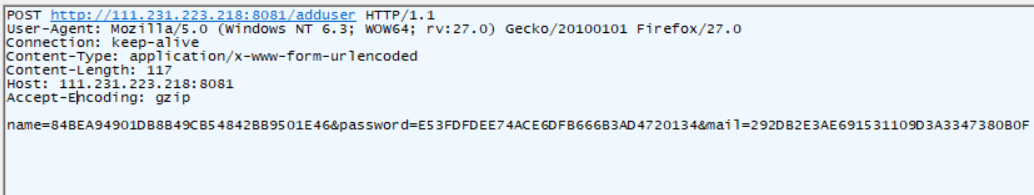
SplashActivity界面为启动界面，在首次启动app的时候，app的启动方式为冷启动，系统需为进程分配资源，app的UI布局与渲染等会消耗一定的时间，在这个时间内，app的界面是空白的。为了用户更好的体验，可自定义一个style并在manifest配置文件中修改启动主题，替换原来的空白界面。

loginActivity界面主要用于用户的登录，除了基本的登录功能，Activity中还有用户的注册功能以及用户忘记密码找回的功能，注册和找回密码功能以Dialog弹框的形式展现。用户忘记密码部分采用邮件验证的机制。

CameraActivity界面主要为调用安卓设备的前摄像头进行拍摄用户人脸图片收集信息，整个Activity以surfaceView和imageView作为布局控件，当用户进入CameraActivity界面时，点击屏幕即可拍照获取人脸信息，并上传至服务器进行人脸识别。

workActivity界面为工作界面，用户进入工作界面后可输入所要登录的网站的域名，以及用户自己方便记忆的简单口令，当点击确定后进入CameraActivity界面进行拍照人脸识别，服务端完成人脸识别后返回相应的结果时，从CameraActivity界面调回workActivity界面，根据服务端返回的结果为用户显示信息。当识别不通过或者口令错误时提示用户；当识别通过后，以弹框的形式展示出改网站的登录密码。

前端与服务端的通信数据均使用加密算法AES进行加密传输，确保了数据在传输过程中不被他人盗取或篡改。下图为注册用户时向服务端发送的http报文：



图五：Http报文

前端拍照获取服务端返回的部分代码展示：

当用户输入的网站域名和口令不为空是执行下列代码：

|  |
| --- |
| EditText webAddressEdit = findViewById(R.id.*autoCompleteTextView*); /\*\* EditText passwordEdit = findViewById(R.id.*editText4*); \* 获取输入框控件并取 User.*webAddress* = webAddressEdit.getText().toString(); \* 得用户输入的数据 User.*order* = passwordEdit.getText().toString(); \*/ String webAddress = User.*webAddress*; saveHistory("history",urlE); //网址存入本地历史Intent intent = new Intent(workActivity.this, CameraActivity.class);  startActivity(intent)； //跳转CameraActivity进行拍照 |

CameraActivity取得用户人脸信息发送给服务端，服务端进行人脸识别并返回结果，跳转workActivity，workActivity判断返回数据，如果识别成功返回用户网站的密码，则执行以下代码：

|  |
| --- |
| try {  *str*=Aes.*decrypt*(*seed*,User.*webPassword*); //对返回的数据进行解密得到密码 } catch (Exception e) {  e.printStackTrace(); } User.*webPassword*="null"; //将用户全局变量密码置空  /\*\*  \* 构造一个弹框，显示用户密码，并添加复制按钮操作  \*/ final AlertDialog.Builder builder = new AlertDialog.Builder(workActivity.this); builder.setMessage(*str*); builder.setTitle("密码"); builder.setPositiveButton("复制", new DialogInterface.OnClickListener() {  @Override  public void onClick(DialogInterface dialog, int which) {  ClipData mClipData = ClipData.*newPlainText*("Simple test", *str*);  ClipboardManager mClipboardManager;  mClipboardManager = (ClipboardManager) getSystemService(*CLIPBOARD\_SERVICE*);  mClipboardManager.setPrimaryClip(mClipData);  Toast toast = Toast.*makeText*(workActivity.this, null, Toast.*LENGTH\_SHORT*);  toast.setText("复制成功");  toast.show();  } }); builder.show(); |

# 3 测试

在项目中，人脸识别的准确性和基于生物特征生成的密码的可用性无疑是最重要的。所以我们找了240组一个人的不同照片以及159组不同人的照片对我们的面部识别程序进行了测试。

我们采用的面部识别技术是通过特征提取器提取出面部的68个特征点如下图所示并取用其特征点之间的一个欧拉距离作为评判标准，建立了一个欧拉距离评判模型，当欧拉距离小于0.6时，我们就判定为同一个人。

|  |
| --- |
| 图六 生物特征信息点提取 |

下图为我们对240组相同人不同照片进行面部识别的测试结果。

|  |
| --- |
| C:\Users\jaska\Documents\Tencent Files\648757299\Image\Group\42CLMV`65L5A@%AK~SH0)]0.png |

由上图可知，‘在同一个人不同照片的测试中，其特征点之间的欧拉距离都集中在0.6即我们的判别是否为同一个人的分界点(低于0.6即判定为同一个人)，而在159组不同人的面部识别中，结果如下图所示，准确率高到97.5%(即判定结果为不是同一个人)，由此可见，我们的面部识别算法具有很高的准确率。

|  |
| --- |
| C:\Users\jaska\Documents\Tencent Files\648757299\Image\Group\$]@0GJSX$@~]1(FVVNH@QPE.png |

而在基于生物特征信息的加密算法，本项目主要采用了hmac的一个消息验证码的单项加密信息摘要，我们采用了12561个测试用例对该算法进行了测试。

首先，通过对不同的照片输入，相同的密钥，测试结果如图九所示可以发现其生成的强口令最长为6位（全长44位），占比约为14%（图五），可以说是相对安全的。

|  |
| --- |
| C:\Users\jaska\Documents\Tencent Files\648757299\Image\Group\X8`C2BK16UJLX9O}SQ1GFHF.png |

其次，其次，在使用相同的面部数据时，输入的弱口令及网址均相同时（这里指单次相同，并不是每一次输入都相同），测试结果如图十所示，其强口令之间最长的字串为7位，占比为16%，安全系数较高。

|  |
| --- |
|  |

最后，在使用相同的面部数据时，输入的弱口令不等同于网址时，测试结果如图十一所示，其强口令最长的字串则会下降到5位，占比13%，安全性极高。

|  |
| --- |
|  |

由上面针对面部特征检测以及基于生物特征加密的测试来看，在同一个人不同照片的情况下，识别率高达100%，而在不同人的情况下，我们的面部特征识别具有高达97.5%的高识别率。

而生物特征信息加密的测试中，密码串重复程度较小，复杂程度高，无特殊规律，符合强口令的特征。

由上述测试可知，本项目使用的面部特征识别及面部特征信息机密算法，均有很高的使用性，符合项目要求。

# 4 项目总结

目前项目已经基本完成了预期的基本功能，生物特征加密由前期的指纹识别及基于指纹信息的加密变更为面部特征识别以及基于面部特征信息的加密，因为，就目前的手机而言，我们无法从手机上获取到指纹信息，所以，经过小组讨论，我们将指纹信息改为面部特征呢信息。

项目已按原计划按时完成，但是，由于人脸识别和人脸加密的处理时间较长再加上服务器配置较低，导致用户得到密码的时间较长，这是我们后期需要优化改进的地方，在本次项目中，前后端及算法成员均获益良多，尤其是关于如何安排计划，如何解决问题，如何与团队成员交流等方面。

总的来说，本项目进展顺利，已按原计划顺利完成，项目依然存在不足之处，还有需要改进的地方，需要团队继续努力。