**XXXX分工文档**

目录

[一、成员说明 2](#_Toc103453925)

[二、分工概述 2](#_Toc103453926)

[三、模块设计说明 3](#_Toc103453927)

[3.1 服务总览 5](#_Toc103453928)

[3.1.1 实现功能 5](#_Toc103453929)

[3.1.2 模块间的加载与配置 5](#_Toc103453930)

[3.1.3 模块间的调用关系 6](#_Toc103453931)

[3.1.4 模块负责人 7](#_Toc103453932)

[3.2 数据中间件 7](#_Toc103453933)

[3.2.1 实现功能 7](#_Toc103453934)

[3.2.2 模块间的加载与配置 7](#_Toc103453935)

[3.2.3 模块间的调用关系 8](#_Toc103453936)

[3.2.4 模块负责人 9](#_Toc103453937)

[3.3.4/5G无人机控制端无感知认证 11](#_Toc103453943)

[3.4.1 实现功能 11](#_Toc103453944)

[3.4.2 模块间的加载与配置 11](#_Toc103453945)

[3.4.3 模块间的调用关系 12](#_Toc103453946)

[3.4.4 模块负责人 12](#_Toc103453947)

[3.5 4/5G无人机与核心网/卫星的接入认证 12](#_Toc103453948)

[3.5.1 实现功能 12](#_Toc103453949)

[3.5.2 模块间的加载与配置 12](#_Toc103453950)

[3.5.3 模块间的调用关系 12](#_Toc103453951)

[3.5.4 模块负责人 12](#_Toc103453952)

[3.6 用户认证 13](#_Toc103453953)

[3.6.1 实现功能 13](#_Toc103453954)

[3.6.2 模块间的加载与配置 13](#_Toc103453955)

[3.6.3 模块间的调用关系 15](#_Toc103453956)

[3.6.4 模块负责人 17](#_Toc103453957)

[3.7 无人机群组间认证与配置管理 17](#_Toc103453958)

[3.7.1 实现功能 17](#_Toc103453959)

[3.7.2 模块间的加载与配置 18](#_Toc103453960)

[3.7.3 模块间的调用关系 19](#_Toc103453961)

[3.7.4 模块负责人 19](#_Toc103453962)

# 一、成员说明

现云雾混构拥有：

表1-1 成员说明

|  |  |
| --- | --- |
| 4位本科生 | 吴飞宇、豆玉赟、李学欣、刘梦洁 |

# 二、分工概述

表1-2 分工概述

|  |  |  |
| --- | --- | --- |
| 总体把控 | 吴飞宇 | 管理项目进度，把控项目进程。 |
| 无人机与控制器之间的无感知认证 |  |  |
| 无人机与核心网 或者卫星之前的接入认证 |  |  |
| 无人机如何快速处理海量并发接入问题 |  |  |
| 无人机与无人机之间的(域内或者跨域)组网 |  |  |
| 无人机与无人机之间的安全协同 |  |  |
| 通信间安全问题 |  |  |
| 无人机耗电量效率问题 |  |  |
| 无人机硬件搭载问题 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 三、模块设计说明

## 3.1 服务总览

### 3.1.1 实现功能

前端展示模块主要包含两个部分，即展示前端和数据后端。

展示前端实现了以下基本功能：

（1）项目基本静态信息展示，包括项目简介，项目创新点和项目成果等；

（2）4/5G自适应认证、用户认证和（软）生物认证的认证数据展示；

（3）4/5G自适应认证、用户认证和（软）生物认证的服务运行状态和参数控制；

（4）项目管理后台的权限控制，限制不同权限用户的操作范围。

数据后端实现了以下基本功能：

（1）4/5G自适应认证、用户认证和（软）生物认证的认证数据查询，支持分页查询；

（2）4/5G自适应认证、用户认证和（软）生物特征的认证结果聚合，支撑前端展示的数据大屏展示。

（3）设计并实现了项目管理后台的用户权限模型，根据权限获取用户可操作的路径范围，并在前端展示。

该模块与项目中的功能指标和性能指标无关。

### 3.1.2 模块间的加载与配置

展示前端上线部署时以html文件的形式表示，需要将文件挂载在Nginx上，数据后端需要以jar包的形式运行，展示前端中的部分请求依赖于数据后端。这两个部分都需要在Nginx中进行相应的配置。

展示前端的Nginx配置示例，其中root对应路径按需修改：

location / {

root /root/Document/xw/all-dist/dist;

index index.html index.htm;

}

数据后端使用8080端口，docker端口映射关系为26030:8080。数据后端的Nginx配置示例如下，其中/yunwu-demo/是展示前端的请求前缀，proxy\_pass是请求指向数据后端的地址，这两个部分按需修改：

location /yunwu-demo/ {

proxy\_buffer\_size 64k;

proxy\_buffers 4 32k;

proxy\_busy\_buffers\_size 64k;

proxy\_set\_header Host $http\_host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header REMOTE-HOST $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_pass http://172.17.17.2:26030;

proxy\_read\_timeout 150;

}

### 3.1.3 模块间的调用关系

本模块主要依赖于数据中间件的RabbitMQ解析程序组件和Docker容器探测组件。

本模块与RabbitMQ解析程序不存在直接的数据交互。RabbitMQ解析程序主要解析RabbitMQ中来自三个认证服务中的数据日志，并将其中的认证结果存入Mysql数据库，本模块从Mysql数据库中取出并聚合对应的认证数据。

本模块与Docker容器探测组件通过socket进行交互，本模块中的数据后端直接向Docker容器探测组件发送对应的命令数据即可。具体的命令、含义和返回值如下：

|  |  |
| --- | --- |
| 命令数据 | 含义 |
| docker stop 服务名 | 关闭服务所在容器 |
| docker start 服务名 | 运行服务所在容器 |
| docker inspect 服务名 | 探测服务的运行状态 |

### 3.1.4 模块负责人

负责人姓名：夏威

负责人联系方式（邮箱）：weixia1660@163.com

## 3.2 数据中间件

### 3.2.1 实现功能

本模块主要包含以下几个功能组件：

（1）socketServer

（2）RabbitMQ解析组件

（3）Docker容器探测组件

（4）请求代理组件

Docker容器探测组件实现了以下功能

（1）根据服务名关闭对应Docker；

（2）根据服务名运行对应Docker；

（3）根据服务名探测对应服务的运行状态。

socketServer组件实现了以下功能：

（1）接收自适应设备认证、用户认证、（软）生物认证服务产生的日志。

（2）将上述接收的日志转发到RabbitMQ服务器暂存，并等待RabbitMQ解析组件消费。

请求服务器实现了以下功能：

（1）根据请求的URL将安卓端发起的请求转发给对应的认证服务器，返回响应消息。

### 3.2.2 模块间的加载与配置

socketServer组件作为一个独立的jar包镜像部署，占用端口26004。

RabbitMQ服务器作为第三方应用程序，占用端口5672和15672。

Docker容器探测组件作为一个独立的war包进行部署，占用端口26043。

请求代理组件作为一个独立的jar包进行部署，使用8080端口，docker端口映射关系为26037:8080。请求代理组件的Https通过Nginx代理实现，所以请求代理组件需要在Nginx的配置请求转发。此处的/fog/是URL请求前缀，按需修改。

location ^~ /fog/ {

proxy\_pass http://172.17.17.2:26037;

# proxy\_redirect http:// https://;

proxy\_set\_header Host $http\_host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

### 3.2.3 模块间的调用关系

socketServer组件与自适应设备认证、用户认证、（软）生物认证服务以及RabbitMQ服务器存在数据依赖关系。其中，自适应认证、用户认证、（软）生物认证服务与socketServer组件通过socket进行数据交互，具体接口信息如下：

|  |  |
| --- | --- |
| 请求地址 | 172.17.17.2:26004 |
| 支持格式 | String |
| 接口描述 | 自适应认证、用户认证、（软）生物认证服务提供字符串数据，socketServer组件接收到数据后转发至RabbitMQ服务器。 |

RabbitMQ服务器负责汇聚由socketServer转发的日志，暂存并等待RabbitMQ解析组件消费日志。RabbitMQ通过5672端口与socketServer组件和RabbitMQ解析组件交互。

Docker容器探测组件与前端展示模块的数据后台存在依赖关系，两者通过socket进行通信，交互的方式和数据结构见3.1.3。

请求代理组件与安卓模块之间通过Https进行数据交互，具体的接口信息如下，请求数据格式见3.4、3.5和3.6中的数据格式。

（1）用户认证请求接口

|  |  |
| --- | --- |
| 请求路径 | https://cloud.gorio.top:16443/fog/userAuth |
| 支持格式 | JSON |
| 接口描述 | 安卓模块向请求代理组件提供json数据，请求代理组件将数据转发到用户认证服务，并返回请求结果。 |

（2）（软）生物认证请求接口

|  |  |
| --- | --- |
| 请求路径 | https://cloud.gorio.top:16443/fog/bioAuth |
| 支持格式 | JSON |
| 接口描述 | 安卓模块向请求代理组件提供json数据，请求代理组件将数据转发到（软）生物认证服务，并返回请求结果。 |

（3）（软）生物注册请求接口

|  |  |
| --- | --- |
| 请求路径 | https://cloud.gorio.top:16443/fog/bioRegister |
| 支持格式 | JSON |
| 接口描述 | 安卓模块向请求代理组件提供json数据，请求代理组件将数据转发到（软）生物注册服务，并返回请求结果。 |

（4）活体检测请求接口

|  |  |
| --- | --- |
| 请求路径 | https://cloud.gorio.top:16443/fog/liveCheck |
| 支持格式 | JSON |
| 接口描述 | 安卓模块向请求代理组件提供json数据，请求代理组件将数据转发到活体检测服务，并返回请求结果。 |

### 3.2.4 模块负责人

负责人姓名：李泽健

负责人联系方式（邮箱）：1565225493@qq.com

## 3.3安卓端开发

### 3.3.1 实现功能

1、实现安卓端的前端显示；

2、实现与用户认证和生物认证的交互；

3、安卓端设备认证实现了以下基本功能：

（1）向服务器发送信息并收到返回的信息。

（2）针对信息进行处理，并实现密码计算等一系列功能，最终实现APP所属UE与服务器的认证。

（3）与服务器协商密钥，并发送至中间件。

该模块与项目中的功能指标和性能指标无关。

### 3.3.2 模块间的加载与配置

安卓端设备认证模块需要使用安卓客户端启动，其配置方法与安卓客户端方法相同，与之不同的是由于使用了一系列密码算法，需要添加bouncycastle库，方法是将其jar包添加至libs文件夹中，其他不需要额外配置。

用户认证模块与生物认证模块需要使用安卓客户端注册与登录，其配置为android studio gradle-7.0.2，jdk11，其他不需要额外配置。

### 3.3.3 模块间的调用关系

本模块部署在安卓客户端内部，主要由安卓前端调用。

本模块与4/5G自适应设备认证服务器存在直接数据交互，使用TCP协议进行通信，完成认证后将协商好的Sessionkey发送至安卓前端，由前端进行后续操作。

与4/5G自适应设备认证服务器交互的方式。

定义类ClientLong，将安卓所属UE视为客户端，与4/5G自适应设备认证服务器进行长连接，其中主要函数以及输入输出具体如下表。

|  |  |  |  |
| --- | --- | --- | --- |
| **函数名** | **功能** | **输入参数** | **输出** |
| ClientLong | 构造函数，设置服务端的IP，端口 | IP（String），port（int） | 无 |
| start | 开启客户端 | 无 | 无 |
| stop | 关闭客户端 | 无 | 无 |
| sendString | 向服务端发送消息 | obj（String） | 无 |
| getdata | 提取服务端返回的消息 | 无 | Sdata（String） |

1、安卓客户端调用设备认证方式

根据不同的协议，定义相应的类，点击相应的安卓前端按钮，完成认证的功能。其中每个类都包含开启设备认证流程的函数具体如下表。

|  |  |  |  |
| --- | --- | --- | --- |
| **类名** | **功能** | **输入参数** | **输出** |
| UE\_start\_attach\_4G | 开启4GEPSAKA认证流程 | 无 | 无 |
| UE\_start\_attach\_5G | 开启5GAKA认证流程 | 无 | 无 |
| UE\_start\_attach\_EAP | 开启4GEAPAKA认证流程 | 无 | 无 |
| UE\_start\_attach\_selfadapt | 开启自适应认证流程 | 无 | 无 |

|  |  |  |  |
| --- | --- | --- | --- |
| **函数名** | **功能** | **输入参数** | **输出** |
| on() | 开启TCP客户端，向服务器发送初始信息，提示服务器开启认证流程。调用认证流程类进行认证  （UE\_start\_attach\_selfadapt类的on函数中会判断所需完成的协议类型以及协议步骤，并开启协议流程） | 无 | Sessionkey（string） |

与用户认证和生物认证都是用https进行交互，并采用json格式进行数据交互。

以下是用户注册json格式：

{

"userTelNum":"19818501743",

"userPassword":"123456",

"userName":"bbblacoo",

"userTrueName":"",

"userCarNum":"",

"idcardNum":"",

"userEmail":"xpshi1997@163.com",

"userAddr":"",

"userBankName":"",

"userBankAccNum":"",

"businessCardNum":"",

"userUkeyId":"",

"eventLog":{

"globalId":"ddd",

"parentReid":"sss"

}

}

### 3.3.4 模块负责人

负责人姓名：张冬林、闫泽润

负责人联系方式（邮箱）：1047310438@qq.com、18709199686@163.com

## 3.4 4/5G自适应设备认证前端

### 3.4.1 实现功能

仿真系统依据3GPP标准、行业现状、国密算法标准为基础，实现了基于LTE、5G网络、自适应网络三个安全仿真平台，实现了设备的接入认证、移动切换的动态链接。

### 3.4.2 模块间的加载与配置

（1）项目打包：在target的上一层目录下，打开cmd窗口，输入以下命令：

mvn clean

mvn package

（2）将Jar包放在docker中./home下，使用17002端口，docker端口映射关系为26008:17002。

（3）重启docker，输入以下命令：

docker stop UnifiedPlatform

docker start UnifiedPlatform

（4）在浏览器中输入：<https://cloud.gorio.top:16443/dyrandvoidwx/>，在该页面进行操作。

### 3.4.3 模块间的调用关系

4/5G自适应设备认证前端模块与后端模块调用。

数据交互：Socket tcp 通信

数据结构：JSON对象

### 3.4.4 模块负责人

负责人姓名：卜秋雨

负责人联系方式（邮箱）：qiuyubu@163.com

## 3.5 4/5G自适应设备认证

### 3.5.1 实现功能

设备认证组后端依据3GPP标准、行业现状、国密算法标准为基础，以及当前针对密钥生成、密钥交换、安全接入认证、安全切换、安全漫游、群组接入等相关领域的最新研究进展为基础，重点实现以下具体功能包括：

1. UE驻留基站；
2. UE附着-包括EPS-AKA，5G-AKA，EAP-AKA’，群组接入认证与自适应接入认证；
3. UE切换测量；
4. UE切换认证-包括X2切换，S1切换，N2切换，Xn切换与群组切换认证。

### 3.5.2 模块间的加载与配置

该过程面向全新部署

（1）配置环境

sudo apt install build-essential

sudo apt install redis libhiredis-dev openssl

sudo apt-get install git

wget https://github.com/Kitware/CMake/releases/download/v3.19.2/cmake-3.19.2.tar.gz

tar -zxvf cmake-3.19.2.tar.gz

cd cmake-3.19.2/

./bootstrap --prefix=/usr

make

sudo make install

hash -r

（2）配置git

#设置用户名

git config --global user.name "EV"

#设置git邮箱

git config --global user.email "1109444922@qq.com"

#生成git ssh密钥

ssh-keygen -C '1109444922@qq.com' -t rsa

#打印密钥，复制进git的设置

cat /root/.ssh/id\_rsa.pub

（3）克隆5G注册项目

git clone <https://git.gorio.top:16443/5gAuth/5G-register.git>

克隆后请注意查看redis的IP与端口，如果redis部署在本地，则将IP设置为127.0.0.1；否则设置为对应的IP。

编译后运行do\_register.c，选择默认注册模式即可。

（4）克隆仿真平台守护进程项目

git clone <https://git.gorio.top:16443/UnifiedPlatform/unifiedplatformguard.git>

克隆后编译即可。

（5）克隆仿真平台项目

git clone <https://git.gorio.top:16443/UnifiedPlatform/UnifiedPlatformBack.git>

代码分为带前端版本和后端版本。

后端版本：当注释掉structs/XD\_TDYTH\_UNIFIED\_common.h中的宏定义FRONTSHOW,并取消DEFAULT的注释后，可以直接编译项目并使用脚本test.sh快速打开所有终端窗口

前端版本：存在FRONTSHOW宏定义，可通过<https://cloud.gorio.top:16443>/dyrandvoidwx/进行前端展示

**运行前准备工作：**

* 全局替换地址列表（此处应该写成相对地址，但目前尚未更改）

ctrl+shift+r，将/home/ev/UnifiedPlatformBack/config.json替换为自有目录/UnifiedPlatformBack/config.json

* 更改redis配置
* 进入tools/XD\_TDYTH\_UNIFIED\_redis\_util.c，将IP的宏定义127.0.0.1解注释，如果在本部实验室环境可直接使用172的配置
* 编译

（6）设置服务自启动

即编写脚本自动运行XD\_UNIFIED\_TDYTH\_GuardProcess.c与XD\_UNIFIED\_TDYTH\_GuardClose.c。

### 3.5.3 模块间的调用关系

4/5G自适应设备认证前端模块与后端模块调用；

数据交互：Socket tcp 通信；

数据结构：JSON对象；

详细内容请自行前往目录/UnifiedPlatformGuard下查看

### 3.5.4 模块负责人

负责人姓名：易威

负责人联系方式（邮箱）：yiwei.to@qq.com

## 3.6 用户认证

### 3.6.1 实现功能

该模块实现了用户信息的注册和基于密码、验证码、TOTP的多因子用户认证。

注册服务模块：实现用户的身份信息注册服务，系统根据用户提供的个人信息在数据库完成注册服务。

认证因子分发模块：包含短信收发服务模块、TOTP生成模块。实现客户端和服务器端认证因子的同步，通过对不同模块的组合使用辅助达到确认访问用户身份、维护系统安全的目的。

认证协议管理模块：提供多端口、多身份认证协议服务。用户可以根据注册时的用户名、手机号或者邮箱，根据不同场景或安全级别选择密码认证、短信验证码认证、TOTP认证、密码和验证码认证、TOTP和密码认证、TOTP和验证码认证6种鉴别方式实现用户身份认证。实现了不同认证协议的接口和认真处理策略，为多因子认证提供实现基础，保证实体身份认证系统有效运行。

### 3.6.2 模块间的加载与配置

注册服务模块打包形式为war包，将war包挂载在容器tomcat7:v1对应目录下，配置的ip地址为172.17.17.2，端口映射关系为26038:8080，26039:9999，26040:8443。

认证中间件服务模块为new\_middle，配置在雾服务器，操作系统为Ubuntu 18.04.5 LTS，配置的ip地址为172.17.17.2，映射端口为26005，需要为日志生成目录分配读写权限。

认证服务模块：分为controller和Server两部分，操作系统为Ubuntu 18.04.5 LTS，配置的ip地址为172.17.17.2，映射端口为26019，Server需要配置运行所需库文件，主要依赖的库文件位于项目根目录~/libs下，具体依赖的库文件如下：

|  |
| --- |
| linux-vdso.so.1 (0x00007fffec3cb000)  libgost.so => /root/sxp/userAuth/Server1213/./libs/libgost.so (0x00007fc981119000)  libxml2.so.2 => /usr/lib/x86\_64-linux-gnu/libxml2.so.2 (0x00007fc980d58000)  libsm2\_cert.so => /root/sxp/userAuth/Server1213/./libs/libsm2\_cert.so (0x00007fc980b54000)  libpthread.so.0 => /lib/x86\_64-linux-gnu/libpthread.so.0 (0x00007fc980935000)  libmysqlclient.so.20 => /usr/lib/x86\_64-linux-gnu/libmysqlclient.so.20 (0x00007fc98037d000)  libevent-2.1.so.7 => /usr/local/lib/libevent-2.1.so.7 (0x00007fc980124000)  libsm234.so => /root/sxp/userAuth/Server1213/./libs/libsm234.so (0x00007fc97ff0d000)  libSEC\_converge\_client.so => /root/sxp/userAuth/Server1213/./libs/libSEC\_converge\_client.so (0x00007fc97fd0a000)  libc.so.6 => /lib/x86\_64-linux-gnu/libc.so.6 (0x00007fc97f919000)  libcrypto.so.1.0.0 => /usr/lib/x86\_64-linux-gnu/libcrypto.so.1.0.0 (0x00007fc97f4d5000)  libdl.so.2 => /lib/x86\_64-linux-gnu/libdl.so.2 (0x00007fc97f2d1000)  libicuuc.so.60 => /usr/lib/x86\_64-linux-gnu/libicuuc.so.60 (0x00007fc97ef19000)  libz.so.1 => /lib/x86\_64-linux-gnu/libz.so.1 (0x00007fc97ecfc000)  liblzma.so.5 => /lib/x86\_64-linux-gnu/liblzma.so.5 (0x00007fc97ead6000)  libm.so.6 => /lib/x86\_64-linux-gnu/libm.so.6 (0x00007fc97e738000)  liblfh234.so => /usr/lib/liblfh234.so (0x00007fc97e521000)  /lib64/ld-linux-x86-64.so.2 (0x00007fc9815bb000)  libssl.so.1.1 => /usr/local/lib/libssl.so.1.1 (0x00007fc97e28c000)  libcrypto.so.1.1 => /usr/local/lib/libcrypto.so.1.1 (0x00007fc97dd99000)  libstdc++.so.6 => /usr/lib/x86\_64-linux-gnu/libstdc++.so.6 (0x00007fc97da10000)  libgcc\_s.so.1 => /lib/x86\_64-linux-gnu/libgcc\_s.so.1 (0x00007fc97d7f8000)  libicudata.so.60 => /usr/lib/x86\_64-linux-gnu/libicudata.so.60 (0x00007fc97bc4f000) |

数据库：安装mysql5.7、Navicat Premium，配置ip地址为172.17.17.2，映射端口为36002，执行sql文件server database back up.sql 创建所需数据库和表。

各部分通过clion进行代码的编写，随后在各模块下的根目录下进入~/build 文件夹下，打包生成可执行文件，自启动脚本如下：

|  |
| --- |
| #!/bin/bash  /etc/init.d/ssh start  cd /root/sxp/userAuth/controller/build/  ./Controller\_Epoll &  cd /root/sxp/userAuth/Server1213/build/  ./Server &  cd /root/sxp/userAuth/new\_middle0417/build/  ./new\_middle & |

### 3.6.3 模块间的调用关系

（1）密码认证：

密码认证接口：

void password\_authentication\_handler(int sockFd, HeaderControllerToServer \* header, PwAuBodySecondMidToSer \* body, PwAuBodyFourthMidToSer \* body2, int flag)；

密码认证数据包接收函数：

void pwau\_body\_second\_mid2ser\_recv(PwAuBodySecondMidToSer \* body, cJSON \* cjson);

密码认证数据包发送函数：

void pwau\_body\_third\_ser2mid\_send(void \* p,cJSON\* cjson);

（2）验证码认证：

验证码认证接口：

void message\_authentication\_handler(int sockFd, HeaderControllerToServer \* header, PwAuBodySecondMidToSer \* body, PwAuBodyFourthMidToSer \* body2, int flag)；

验证码认证数据包接收函数：

void pwau\_body\_second\_mid2ser\_recv(PwAuBodySecondMidToSer \* body, cJSON \* cjson)；

验证码认证数据包发送函数：

void pwau\_body\_third\_ser2mid\_send(void \* p,cJSON\* cjson)；

（3）TOTP申请：

TOTP申请接口：

void token\_application\_handler(int sockFd, HeaderControllerToServer \* header, TkApBodySecondMidToSer \* body);

TOTP申请数据包接收函数：

void tokenApplication\_body\_2\_mid2ser\_recv(TkApBodySecondMidToSer \* body, cJSON \* jsonString);

TOTP申请数据包发送函数：

void tokenApplication\_body\_3\_ser2mid\_send(void \* pp, cJSON \* object);

（4）TOTP认证：

TOTP认证接口：

void token\_authentication\_handler(int sockFd, HeaderControllerToServer \* header, TkAuBodySecondMidToSer \* body, TkAuBodyForthMidToSer \* body2, int flag)；

TOTP认证数据包接收函数：

void token\_body\_2\_mid2ser\_recv(TkAuBodySecondMidToSer \* body, cJSON \* jsonString)；

TOTP认证数据包发送函数：

void token\_body\_3\_ser2mid\_send(HeaderServerToController \* header, TkAuBodyThirdSerToMid \* body2, char \* buf)；

（5）密码和验证码认证：

密码和验证码认证接口：

void password\_and\_messagecode\_authentication\_handler(int sockFd, HeaderControllerToServer \* header, PwAndMsAuBodySecondMidToSer \* body);

密码和验证码认证数据包接收函数：

void PwAndMs\_body\_2\_mid2ser\_recv(PwAndMsAuBodySecondMidToSer \* body, cJSON \* jsonString);

密码和验证码认证数据包发送函数：

void PwAndMs\_body\_3\_ser2mid\_send(void \* pp, cJSON \* object);

（6）TOTP和密码认证：

TOTP和密码认证接口：

void password\_and\_token\_authentication\_handler(int sockFd, HeaderControllerToServer \* header, PwAndMsAuBodySecondMidToSer \* body);

TOTP和密码认证数据包接收函数：

void PwAndMs\_body\_2\_mid2ser\_recv(PwAndMsAuBodySecondMidToSer \* body, cJSON \* jsonString);

TOTP和密码认证数据包发送函数：

void PwAndMs\_body\_3\_ser2mid\_send(void \*pp, cJSON \*object);

（7）TOTP和验证码认证：

TOTP和验证码认证接口：

void messagecode\_and\_token\_authentication\_handler(int sockFd, HeaderControllerToServer \* header, PwAndMsAuBodySecondMidToSer \* body);

TOTP和验证码认证数据包接收函数：

void PwAndMs\_body\_2\_mid2ser\_recv(PwAndMsAuBodySecondMidToSer \* body, cJSON \* jsonString);

TOTP和验证码认证数据包发送函数：

void PwAndMs\_body\_3\_ser2mid\_send(void \*pp, cJSON \*object);

### 3.6.4 模块负责人

负责人姓名：石小平

负责人联系方式（邮箱）：xpshi1997@163.com

## 3.7 （软）生物认证

### 3.7.1 实现功能

主要实现人脸识别和声纹识别功能，其中利用雾节点的计算能力将非隐私性的软生物特征融合生成密钥来加密生物特征、保证传输过程中生物特征的安全性和隐私性。

实体认证系统支持2种以上生物特征的身份鉴别及认证方案，确保生物特征模板数据存储与外包安全。

### 3.7.2 模块间的加载与配置

安装：Ubuntu18、python3.7、mysql

终端运行：

sudo apt install python3-pip

sudo apt install libopencv-dev

sudo apt-get install build-essential cmake

sudo apt-get install libgtk-3-dev

sudo apt-get install libboost-all-dev

sudo apt-get install libx11-dev libatlas-base-dev

sudo apt-get install python3.7-dev

sudo apt-get install python3-tk

sudo apt-get install libboost-python-dev

sudo apt-get install libbz2-dev

sudo apt-get install libicu-dev

pip3 install -r ./program/requirements.txt

安装完成后进入numpy所在目录： /numpy/core/function\_base.py，修改def linspace()函数中，将非注释第一行num = operator.index(num)改为num = operator.index(int(num))。

进到项目代码./program/auc/face/face\_recognition目录，在终端打开，运行sudo python3 setup.py install

运行：

python3 ./program/auc/auc\_new.py

python3 ./program/cloud/cloud\_new.py

python3 ./program/fog/fog\_new.py

若把项目部署在docker中，可在/root下的run.sh脚本中写入以下内容，即可服务自启：

#!/bin/bash

/etc/init.d/ssh start #自动运行ssh，请勿去掉此行，否则重启后无法使用ssh连接

#请在下面输入想要自启的命令

python3 /home/program/auc/auc\_new.py &

python3 /home/program/cloud/cloud\_new.py &

python3 /home/program/fog/fog\_new.py

注：auc、cloud配置文件在./program/config\_new.ini，fog配置文件在./program/config\_fog.ini，保存了IP地址、端口、数据库等相关信息，可更改。

### 3.7.3 模块间的调用关系

本模块FCC 服务本地监听地址为(‘0.0.0.0’, 8021)，公网映射地址为(‘cloud.gorio.top’, 26005)，Cloud服务本地监听地址为(‘0.0.0.0’, 8022)，公网映射地址为(‘cloud.gorio.top’, 26001)，AUC服务本地监听地址为(‘0.0.0.0’, 8023)，公网映射地址为(‘cloud.gorio.top’, 26002)，其他模块可通过公网映射地址与本模块各服务进行socket通信。

本模块与外部环境进行数据交互的方式为json数据包格式。

### 3.7.4 模块负责人

负责人姓名：刘梦如

负责人联系方式（邮箱）：mengrliu@163.com

## 3.8 活体检测

### 3.8.1 实现功能

主要实现人脸识别前对用户进行活体检测，验证用户是否为真实活体本人操作。可有效抵御照片、换脸、面具、遮挡以及屏幕翻拍等常见的攻击手段，从而帮助用户甄别欺诈行为，保障用户的利益。

活体检测达到98%的正确率。

### 3.8.2 模块间的加载与配置

（1）模型训练

安装：python3.6

终端运行：

sudo apt install python3-pip

pip3 install libopencv-dev

pip3 install -r ./Documents/training/requirements.txt

安装完成后进到项目代码./Documents/training目录，在终端打开，运行python3 train\_Fusion\_CyclicLR.py

（2）模型使用

安装：python3.6

终端运行：

pip3 install -r ./Documents/training/requirements.txt

安装完成后进到项目代码./Documents/CandF目录，在终端打开，运行python3 fog\_huoti.py

注：fog\_huoti.py文件中的BIOMETRIC\_ADDRESS为生物识别的地址及端口，FOG\_HUOTI\_LISTEN\_ADDRESS为活体检测监听地址及端口，LOG\_ADDRESS为认证失败后日志发送的地址及端口，可根据实际情况进行更改。

### 3.8.3 模块间的调用关系

本模块服务本地监听地址为(‘0.0.0.0’,2029)，公网映射地址为(‘cloud.gorio.top’, 26004)，其他模块可通过公网映射地址与本模块各服务进行socket通信。

本模块与外部环境进行数据交互的方式为json数据包格式。

### 3.8.4 模块负责人

负责人姓名：闵静怡

负责人联系方式（邮箱）：Caroline\_mj@163.com