文件编号：攻守道-SWC2020-20200148

受控状态：■受控　□非受控

保密级别：□公司级 □部门级 ■项目级 □普通级

采纳标准：CMMI DEV V1.2

数据堡垒

Data Fortress

项目开发文档

Version 6.4.0

2020.06.01

Written by 攻守道

目录

记录更改历史

序号	更改原因	版本	作者	更改日期	备　注
1	添加项目概述	V1.0.0	队员 D	2019/11/16	确定项目的定位和实现方案以及开发目标
2	添加开发计划	V2.0.0	队员 A	2019/11/18	确定项目的开发计划及进度安排
3	更新开发计划	V2.1.0	队员 B	2019/11/19	细化项目的进度安排提炼关键技术问题
4	添加可行性分析	V3.0.0	队员 A	2019/11/20	对项目可行性进行调查研究
5	更新可行性分析	V3.1.0	队员 C	2019/11/22	更新技术可行性分析
6	添加需求分析	V4.0.0	队员 D	2019/11/24	确定项目的需求模块
7	更新需求分析	V4.1.0	队员 B	2019/11/26	更新功能模块的设计要点
8	添加概要设计	V5.0.0	队员 B	2019/12/2	添加概要设计流程及要点
9	完善概要设计	V5.1.0	队员 B	2019/12/22	完善具体流程,确定程序功能
10	添加错误处理	V5.2.0	队员 D	2019/12/23	添加错误处理
11	添加系统部署方案	V5.3.0	队员 A	2019/12/27	系统部署依赖,部署环境
12	添加概要设计	V5.4.0	队员 A	2020/1/23	完善概要设计
12	添加数据库设计	V6.0.0	队员 C	2020/1/28	添加ER图及设计方案
13	完成详细设计	V6.1.0	队员 C	2020/2/10	完善详细设计,补全各功能实现细节
14	完善整体流程	V6.1.1	队员 B	2020/3/10	修改各功能模块,添加部分新功能的描述
15	添加详细设计	V6.2.0	队员 B	2020/4/20	添加详细设计
16	优化详细设计	V6.3.0	队员 B	2020/5/10	优化全部功能
17	优化全文	V6.4.0	队员 A	2020/6/1	优化行文逻辑

1 项目概述

1.1 项目背景

近年来随着移动互联网行业的快速发展，手机被赋予的功能也欲加丰富，越来越多的人选择将生活中的点点滴滴通过照片的方式记录下来，用于朋友圈分享或保存以供回忆。但很多人却在不知不觉中将自己的隐私信息全部泄露给了不法分子，而依托如今强大的海量数据分析技术，仅仅依靠无意间拍进照片中的一段文字，露出的一张笑脸，就可以获取到一个人的大量信息。

此外，随着人脸识别技术突飞猛进的发展，人脸识别的应用场景日趋丰富，人脸也被称为“行走的密码”，在带来高效便捷的同时，也夹杂着大量隐患，这种技术不仅可以用来抓取个人的面部生物信息，还可以与既有数据库中的数据进行比对。它能进一步追踪到个人的身份信息、日常的行踪轨迹、人与车的匹配、亲属关系的匹配以及经常接触人员的匹配等。由于人脸识别具有非接触性的特点，这意味着很多人可能就是在不知情的情况下分享了一张包含人脸的图片，从而被抓取到面部信息。当算法对人作出更多分析之后，成为“透明人”的可能性无疑在增加。更为重要的是，现在大量的移动 APP 需要进行人脸数据采集，但却并没有对数据本身进行加密防护，一旦数据库泄露，后果不堪设想。相关的新闻报道更是层出不穷，人脸信息大量泄露并被违法售卖，更有甚者将人脸信息和对应的全部身份信息打包售卖，究其原因还是当前市面上并不存在一种完备，安全的隐私保护方式，导致用户或企业即使有隐私保护的意识，但是却没有合适技术解决方案来满足这样的需求。

针对以上问题，我们队伍提出了一种高效且安全等级高的图像加密算法，并借助快应用“即点即用，高效快捷”的特点开发出了一款全新的以人工智能为核心的隐私保护解决方案。

图 1.1.1 图片信息泄露新闻

1.2 项目定位

1.2.1 应用场景

在用户需要对可能包含隐私信息的图片进行保存或分享时，Data Fortress 可以借助人工智能技术，发挥出强大的隐私信息自动识别并加密的功能。当拥有查看权限的用户需要查看原图时，高效的解密方案让用户使用更加便捷，同时可实现加密信息的无损恢复，真正实现了将自己的隐私掌握在自己手中。该项技术还赋予了用户极大的权限，使用户可以自主选择需要进行加密的部分。此外，为方便用户进行分享或展示等操作，Data Fortress 还实现了信息的无感加密，利用卡通玩偶图片等对出现的隐私信息区域进行遮挡，由于加密的技术框架，可以保证数据的无损恢复，和传统的贴图操作有很大差别。既可以实现信息的隐藏又避免了传统加密方法对图像视觉效果的影响。

图 1.2.1 信息泄露场景

根据不同的场景，Data Fortress 提供不同的图片部分加密类型，计划的加密类型如下

1. 人脸图片加密
2. 证件隐私保护
3. 文档隐私保护
4. 卡通隐私保护
5. 自动隐私区域定位保护

1.2.2 目标人群

Data Fortress 的目标人群为经常在公开社交网络上分享自己生活却忽略保护个人隐私的社会各阶层人群，为他们提供高效便捷的隐私保护方案。由于我们提出的加密算法效率高，可进行实时的加解密操作，可以服务于众多对时效性要求较高的实际应用场景。还可以提供给需要保存敏感数据的互联网公司进行数据本身的加密服务，同时，也可以提供给需要对数

据进行分级分发的企事业单位，如银行、上市公司、国有企业等。

1.3 项目方案

- 对于隐私数据区域定位，Data Fortress 采用机器学习方法进行人脸识别定位、文字识别定位。
- 对于原文加密，Data Fortress 采用异或加密算法加快加密速度。
- 对于异或加密掩模，Data Fortress 对于每一张图片生成一个随机种子，使用随机生成方式生成掩模。解密时只需提供加密区域以及随机种子即可解密。

1.4 项目目标

Data Fortress 以“保护您的每份重要数据”为口号，旨在使用人工智能的技术结合信息安全的图片加密算法保护用户的隐私数据安全，真正做到对数据本身进行加密，即使数据泄露也无法获取到其中的隐私信息如人脸、文字等。而当用户需要查看原图时，无需等待即可解密，从而为多媒体数据用户提供完备，便捷的数据保护措施。也可为数据管理不完善的企业提供一种安全无忧的数据保护方案。

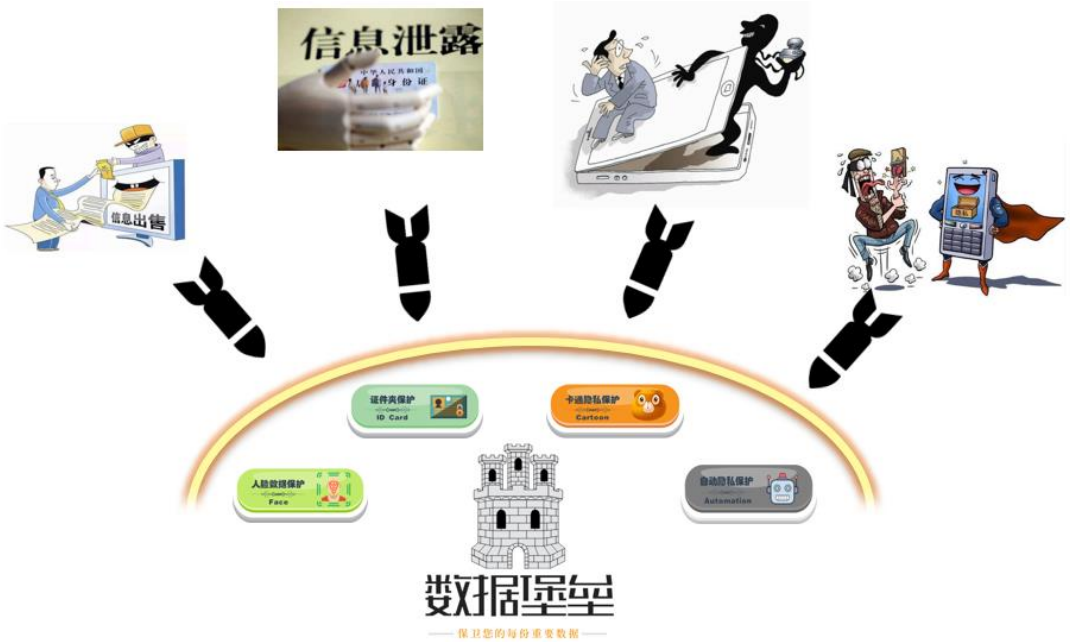

图 1.4.1 项目目标示意

1.5 项目价值

1. 保护用户隐私数据安全，自动识别隐私信息并且加密。
2. 实现合理数据分发与访问层级管理。

Data Fortress 作品紧密结合当下人工智能领域中的人脸识别、OCR 等核心技术。创新地提出了选择性加密的方法。采用合理的异或加密流程，确保了整个数据加密的安全性和应用性。为互联网多媒体隐私数据保驾护航。

Data Fortress 可以为多种应用场景提供加密服务，如

- 身份证、学生卡等隐私数据认证提供加密解决方案
- 针对银行等需要文本数据敏感信息分发的实际场景提供解决方案

2 开发计划

2.1 最终呈现形式

Data Fortress 采用 C/S 结构为用户提供服务。

- 客户端使用快应用作为与用户交互的前端部分。
- 服务端以 Ubuntu18.04 LTS 云端服务器为基础，使用 MySql 数据库为数据的保存和密钥的存储提供服务。
- 客户端与服务端的交互分为 2 种：加密和解密。
- 客户可以选择不同的加密类型。

加密过程：

客户端使用快应用选择需要的服务，将图片上端到服务端。服务端对图片数据进行部分加密，之后将加密之后的结果返回给客户端。

解密过程：

客户端将加密后的密文传上传到服务端，服务端对密文进行解密，之后将解密之后的图片返回给客户端。

安全分享过程：

客户端包含隐私相册功能，用户可在隐私相册中查看全部加密后的图片，同时可以在相册中长按启动分享码分享功能。

2.2 主要功能描述

Data Fortress 提供以下几个部分的加密功能

1. 图片人脸加密
2. 图片文字加密
3. 证件敏感信息加密
4. 卡通人脸加密
5. 自动隐私区域识别加密

2.3 运行环境

Data Fortress 运行环境分为 2 部分

1. 客户端:

客户端运行在支持快应用的 Android 手机上。Android 手机需要提供给手机拍照功能权限，访问相册权限，同时还需要可以连接网络，作为数据传输的基础。

2. 服务端

服务端部分包括了整个应用的后台数据库模块，算法处理模块，以及相应的异常处理模块等部分，运行的环境是 Ubuntu STL18.04 操作系统，服务器后台采用 Django 进行搭建，开发库需要用到 Opencv，Dlib，hashlib 等模块，计算加密解密使用 CPU 进行运算。

2.4 验收标准

1. 功能性

程序可以自动对需要加密的图片选择正确的区域进行加密，并且能够正确进行解密

2. 时效性

从快应用端上传数据，到服务器端处理完成并且返回数据，再到快应用端展示返回结果的全部所需时间在用户的可接受范围内

3. 安全性

密钥空间足够大，使得密文在有效时间内不可能解密出原文

2.5 关键问题

技术焦点

1. 加密区域识别/选择
   a) 人脸识别
      i. 图像人脸识别
   b) 文字识别
   c) 显著性区域识别

对策：区域识别采用人工智能邻域的技术支持。交互式选择依赖于快应用的前端数据采集。

2. 部分加密安全性保障
   a) 密钥设计
   b) 加密和解密的复杂性
   c) 密钥敏感性
   d) 差异性攻击保障
   e) 抗干扰能力

对策：设计一个满足安全性的密钥和对应的加密解密策略

3. 密钥与密文对应关联管理

对策：使用数据库保存密钥，密文中保存密钥的数据库索引

设备条件

1. 计算速度需求

对策：使用高性能服务器、性能无法提高时优化算法。

2.6 进度安排

2019/11/1-2019/11/20	项目可行性分析和需求分析。
2019/11/20-2019/12/30	快应用开发、后台算法实现。产出可实现基本功能的 demo
2020/1/10-2020/2/25	快应用界面优化，提高用户交互体验，后台算法优化，提高识别准确率。产出用户体验良好，加密准确性安全性高的可上线版本。
2020/3/1-2020/3/25	收集用户体验反馈，定向修改或添加功能。
2020/3/26-2020/5/25	迭代更新后端算法、前端界面。添加分享码功能，优化前端界面

2020/5/25-2020/6/11	系统化的功能测试，整合文档，制作演示视频。

2.7 开发预算

阿里云 ECS 云服务器（2 核 CPU，4GB 内存，1M 带宽）：291.5 元/年
开发用 OPPO 等各大品牌安卓手机：1000 元/台

3 可行性分析

3.1 技术可行性分析

人脸识别算法采用技术成熟，时效性好，识别准确的 Dlib 库；加密采用了安全性高，抗差分攻击强的随机密钥生成算法以及成熟稳定的异或加密，能在保证信息安全的同时快速完成处理，给用户以无感体验。文本信息定位与识别采用了 yolo+网络结构，识别准确率高，占用资源低，可以方便的部署到服务器上。此外，我们还加入了基于 Saliency Map 的显著性区域检测，能对人首先注意到的区域进行识别与加密。

异或加密：异或操作是对数据进行位操作，加密和解密都只要一步操作，使得加密效率高。根据可证明密码学原理，如果密钥是伪随机的，并且密钥只用一次，则对应一次一密的加密算法，具有完善的保密加密性质。

Dlib：dlib 人脸检测方法采用 64 个特征点检测，效果会好于 opencv 的方法识别率会更高，同时又有更高的准确性和时效性。

Yolo+：YOLO 最大的优点就是不用再提取 proposal，直接对全图进行回归，原先的两步流程变成了一步到位，速度快，准确性也有保证。

3.2 资源可行性分析

所采用的框架及算法对算力需求不高，只需要一台带宽足够的普通服务器就可以支撑系统的运行。人脸识别和 OCR 所需的训练数据集容易取得。

算法需要后端服务器来支撑运算。Data Fortress 使用搭建 Ubuntu 18.04 LTS 操作系统的运算服务器，并且使用 MySql 数据库存储持久化数据。

人脸识别、显著图使用人脸数据进行模型训练，需要使用带有 GPU 的服务器进行模型训练。需要部署一台模型训练服务器。

OCR 识别可以使用成熟的 YOLO 框架，后期可以采集更多实际数据训练模型，从而提高识别性能和识别精准度。

前端使用部分 Android 手机的快应用功能，需要设备支持快应用功能

3.3 市场可行性分析

数字支付日渐普及，人脸及银行卡等隐私数据变得越来越重要，而互联网的发展更是增加了这些数据暴露的风险。因此，每个人都需要注重对自己隐私信息的保护，而我们的局部加密使得图片上的重要信息在得到保护的同时还具有较高的可读性，方便了用户的日常使用。而数据保护不仅对个人意义重大，对于企业则更是重点问题。如网盘图片的加密，银行卡图片信息等，都可以应用我们的接口或方法。

4 需求分析

4.1 数据需求

4.1.1 静态数据

静态数据是指在运行过程中主要作为控制或参考用的数据,它们在很长的一段时间内不会变化，一般不随运行而改变。

表 4.1.1 服务器静态数据表

名称	固定值	定义	类型
TEMP_MAX_IMG_HEIGHT	1024	图片最大宽度大小，超过此大小的图片被 resize 到此大小	string
ENCRYPTED_IMG_DIR	‘encrypted_imgs\’	加密之后的图片保存文件夹	int
DATABASE_HOST	'127.0.0.1'	数据库 IP 地址	string
DATABASE_PORT	3306	数据库端口	string
DATABASE_USERNAME	'root'	数据库用户名	int
DATABASE_PASSWORD	'root8888'	数据库用户密码	string
CARTOON_IMG_DIR	'cartoon\'	卡通小动物遮盖模板图片目录	string
HASH_COLLISION_QUERY	'SELECT hash_collision_number FROM `image_database`.`hash_collision` where hash_ID = {};'	读取 hash 冲突表 SQL 模板	string
HASH_COLLISION_INSERT	'INSERT INTO `image_database`.`hash_collision` (hash_ID,hash_collision_number) VALUES ({},0)'	插入新数据到 hash 冲突表 SQL 模板	string
HASH_COLLISION_UPDATE	'UPDATE `image_database`.`hash_collision` SET hash_collision_number = {} WHERE hash_ID = {}'	更新数据到 hash 冲突表 SQL 模板	string
IMG_ENCRYPTION_INSERT	'INSERT INTO `image_database`.`img_encryption` (hash_ID, encrypted_img_path, `key`) VALUES ({},\'{}\',%s)'	插入新数据到 img_encryption 表,不包含 mask 内容 SQL 模板	string

IMG_ENCRYPTION_WITH_MASK_UPDATE	'UPDATE `image_database`.`img_encryption` SET mask = %s WHERE hash_ID = {}'	插入新数据到 img_encryption 表,包含 mask 内容 SQL 模板	string
IMG_ENCRYPTION_QUERY	'SELECT encrypted_img_path, `key`,mask FROM `image_database`.`img_encryption` where hash_ID = {}'	查询 img_encryption 表数据 SQL 模板	string
GET_USR_HASH_ID	'SELECT hash_ID FROM image_database.img_encryption where usrId = {};'	获取单个用户上传的所有图片的 hashId	string
CARTOON_IMG_DIR	'cartoon'	卡通小动物遮盖模板图片目录	String
HASHID_MASK1	0x0F0FACBD	加密 hashID 的掩模 1	Int
HASHID_MASK2	0xABFDFACB	加密 hashID 的掩模 2	Int
HASH_LIMIT	1 << (32 - 8)	hash 值的范围	Int

表 4.1.2 前后端接口表

名称	固定值	定义	类型
upload_for_encrypt_face	' /upload_for_encrypt_face'	上传图片人脸加密接口	function
upload_for_encrypt_text	'/upload_for_encrypt_text'	上传图片文字加密接口	function
upload_for_encrypt_card	'/upload_for_encrypt_card'	上传图片证件加密接口	function
upload_for_encrypt_cartoon	‘/upload_for_encrypt_cartoon’	上传图片人脸卡通加密接口	function
upload_for_encrypt_salience	'/upload_for_encrypt_salience'	上传图片 salience 加密接口	function
upload_for_decrypt_img	'/upload_for_decrypt_img'	上传加密图片解密接口	function
req_processed	'/req_processed'	获取图片处理结果接口	function
get_all_picture	‘/get_all_picture’	获取用户上传的所有图片对应 hashID	function
get_gallery_json	'/get_gallery_json'	获取用户上传的所有图片对应的 URL 访问值，并且返回 JSON 格式的结果	function

get_encrypted_img_by_hashId	'/get_encrypted_img_by_hashId'	获取 hashID 对应密文	function
get_share_code	'/get_share_code'	获取分享码	function
get_picture_by_share_code	'/get_picture_by_share_code'	使用分享码获取解密图片	function
gallery	'/gallery'	获取用户相册	

4.1.2 动态数据

动态数据是指在系统应用中随时间变化而改变的数据

表 4.1.3 服务器动态数据表

名称	定义	类型
request	客户端发送的 http request 数据	WSGIRequest:
encrypt_func	用来加密图片的函数，取决于访问服务器的接口	function
img	用于进行图片缩小的中间变量	ndarray
encrypted_img	加密之后的密文图片	ndarray
rects	需要加密的所有矩形框	list
seed	加密使用的 random seed	int
id	图片 hash 得到的 hash 码，用于数据库保存	int
collision	Hash 冲突时用于唯一标志图片的 collision 值	int
keyimg	密钥图片	ndarray
key	密钥	ndarray
box	每个矩形对应的密文图片内容	ndarray
sql	访问数据库时的 sql 语句	String

4.1.3 数据词典

表 4.1.4 img_encryption

字段名	数据类型	允许空值	自动递增	备注
id	BIGINT	否	是	主键
hash_ID	BIGINT	否	否	图片加密的唯一 ID,通过算法计算 hash 得到
encrypted_img_path	VARCHAR(500)	否	否	加密后图片的保存路径
key	BLOB	否	否	加密之后图片的解密密钥
mask	MEDIUMBLOB	是	否	使用 mask 加密解密方

				法的缩小化 mask，重要用于卡通加密以及显著性加密

表 4.1.5 img_encryption

字段名	数据类型	允许空值	自动递增	备注
id	BIGINT	否	是	主键
hash_ID	BIGINT	否	否	hash 数值
hash_collision_number	TINYINT	否	否	hash 数值冲突的数量

4.1.4 数据采集

1. Cartoon 加密卡通图片模板，爬虫采集网络图片
2. 中文文字识别
3. 密钥数据由加密过中产生并且保存
4. salience 模型训练数据库
   - DUI-OMRON
   - DUTS-TE
   - ECSSD
   - HKU-IS
   - PASCAL-S

4.2 功能需求

4.2.1 功能模块

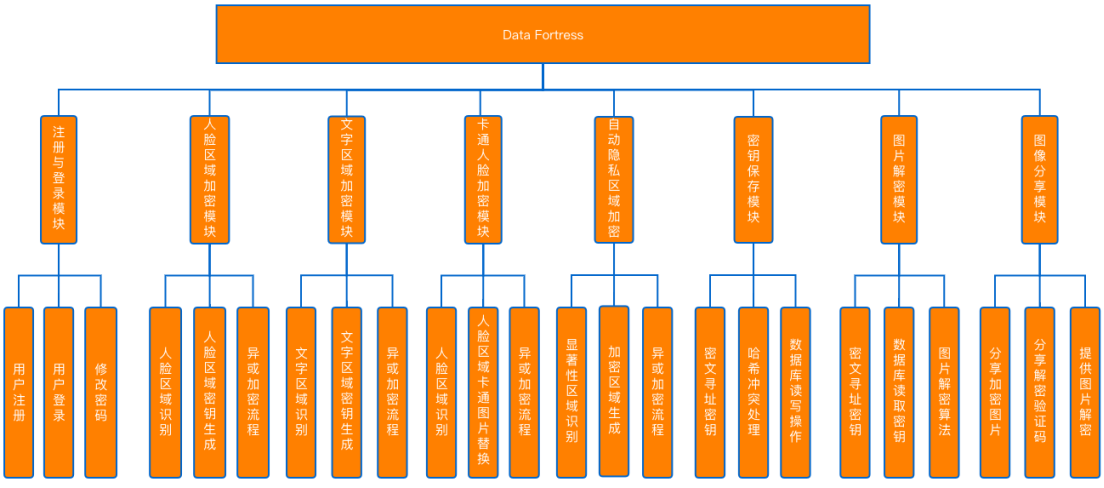

图 4.2.1 功能模块结构图

表 4.2.1 核心功能模块描述

功能模块	功能	功能描述	优先级
人脸区域加密模块	人脸区域识别	使用机器学习进行人脸区域识别	8
	人脸区域加密解密	根据识别出的人脸区域生成密钥进行加密解密	9
文字区域加密模块	文字区域识别	使用机器学习进行 OCR 识别	7
	文字区域加密解密	根据 OCR 识别的结果生成密钥进行加密解密	9
卡通人脸加密模块	人脸区域识别	使用机器学习进行人脸区域识别	8
	人脸区域卡通图片替换	使用卡通图片替换人脸，并且生成解密密钥	8
	已加密图片解密	根据密钥解密还原原始人脸	8
自动隐私区域加密模块	显著性区域识别	使用机器学习进行人脸显著性区域识别	6
	加密区域生成	利用显著性区域生成指定加密区域的掩模	6
	显著性区域加密解密	根据掩模生成加密密钥进行加密解密	6
密钥保存模块	密文寻址密钥	使用密文 hash 生成密钥数据库索引	8
	哈希冲突处理	密文 hash 冲突时，使用链地址法的类似方法解决冲突，并且在密文中保留冲突码。使得密文能够正确找到密钥	6
	数据库读写	将密钥写入数据库以及从数据库中读取密钥	4
图片解密模块	密文寻址密钥	使用加密相同哈希方法得到哈希码，再读取密文中的冲突码。组合形成密钥索引	8
	数据库读取密钥	根据密钥索引在数据库中取出密钥	6
	图片解密算法	根据上传内容选择合适的解密算法对图片进行解密	8

表 4.2.2 人脸加密用例规约

用例名称	人脸加密
功能简述	将用户上传的图片中的人脸部分进行加密
用例编号	Data Fortress 001
执行者	Server
前置条件	1. 图片不为空 2. 图片为 RGB 图片 3. 通过人脸加密接口上传
后置条件	1. 向用户返回加密后的图片 2. 服务器数据库保存解密需要的密钥
涉众利益	用户：希望得到加密之后的图片 数据库：成功保存图片对应的密钥
基本路径	1. 用户选择需要加密的图片 2. 用户使用人脸加密功能上传图片 3. 服务器接收图片 4. 使用人脸检测算法检测出图片中的人脸区域 5. 使用基于区域的加密算法将图片加密，并且生成密钥 6. 使用 hash 方法生成图片 hash 码，服务器根据 hash 码以及 hash 冲突检测结果将密钥写入数据库 7. 服务器将加密之后的图片返回给用户
扩展路径	1. 图片不为 RGB 图片，提示用户 2. 没有检测到人脸区域则直接返回原始图片
字段列表	人脸区域，密钥，密文 hash 值，密钥索引
设计规则	单独模块
未解决的问题	人脸检测侧脸准确率低
备注	单张人脸、大量人脸检测加密均成功

表 4.2.3 文本加密用例规约

用例名称	文本加密
功能简述	将用户上传的图片中的文本部分进行加密
用例编号	Data Fortress 002
执行者	Server
前置条件	1. 图片不为空 2. 图片为 RGB 图片 3. 通过文本加密接口上传
后置条件	1. 向用户返回加密后的图片 2. 服务器数据库保存解密需要的密钥
涉众利益	用户：希望得到加密之后的图片 数据库：成功保存图片对应的密钥
基本路径	1. 用户选择需要加密的图片 2. 用户使用文字加密功能上传图片 3. 服务器接收图片 4. 检测图片中的文字区域 5. 使用基于区域的加密算法并且生成密钥 6. 使用 hash 方法生成图片 hash 码，服务器根据 hash 码以及 hash 冲突检测结果将密钥写入数据库 7. 服务器将加密之后的图片返回给用户
扩展路径	图片不为 RGB 图片，提示用户
字段列表	文字区域，密钥，密文 hash 值，密钥索引
设计规则	单独模块
未解决的问题	文本检测存在误识别
备注	该部分算法基于训练数据集以及识别阈值

表 4.2.4 卡通人脸加密用例规约

用例名称	卡通人脸加密
功能简述	将用户上传的图片中的人脸部分使用卡通人物覆盖加密，并且能够解密
用例编号	Data Fortress 003
执行者	Server
前置条件	1. 图片不为空 2. 图片为 RGB 图片 3. 通过卡通人脸加密接口上传
后置条件	1. 向用户返回加密后的图片 2. 服务器数据库保存解密需要的密钥
涉众利益	用户：希望得到加密之后的图片 数据库：成功保存图片对应的密钥
基本路径	1. 用户选择需要加密的图片 2. 用户使用卡通人脸加密功能上传图片 3. 服务器接收图片 4. 检测图片中的人脸区域 5. 随机选取卡通图片覆盖原有人脸区域 6. 根据卡通图片与原图生成复原所需信息作为密钥 7. 使用 hash 方法生成图片 hash 码，服务器根据 hash 码以及 hash 冲突检测结果将密钥写入数据库 8. 服务器将加密之后的图片返回给用户
扩展路径	图片不为 RGB 图片，提示用户
字段列表	人脸区域，卡通图片，密钥，密文 hash 值，密钥索引
设计规则	依赖人脸加密模块
未解决的问题	密钥占用存储空间大
备注	无

表 4.2.5 自动隐私区域识别加密用例规约

用例名称	自动隐私区域识别加密
功能简述	在用户上传的图片中使用显著性检测结果自动寻找敏感信息进行加密
用例编号	Data Fortress 005
执行者	Server
前置条件	1. 图片不为空 2. 图片为 RGB 图片 3. 通过自动隐私区域定位加密上传
后置条件	1. 向用户返回加密后的图片 2. 服务器数据库保存解密需要的密钥
涉众利益	用户：希望得到加密之后的图片 数据库：成功保存图片对应的密钥
基本路径	1. 用户选择需要加密的图片 2. 用户使用自动隐私区域定位加密功能上传图片 3. 服务器接收图片 4. 使用显著性检测寻找显著性区域并且生成加密位置掩模 5 使用基于掩模的加密算法进行加密并且生成密钥 6. 使用 hash 方法生成图片 hash 码，服务器根据 hash 码以及 hash 冲突检测结果将密钥写入数据库 7. 服务器将加密之后的图片返回给用户
扩展路径	1. 图片不为 RGB 图片，提示用户 2. 用户指定位置不在原图中，提示用户
字段列表	显著性区域掩模，密钥，密文 hash 值，密钥索引
设计规则	单独模块
未解决的问题	密钥占用空间大，硬盘消耗高
备注	无

表 4.2.6 密钥保存用例规约

用例名称	密钥保存
功能简述	将所有加密算法生成的密钥保存到数据库中
用例编号	Data Fortress 007
执行者	Server
前置条件	1. 加密算法提供密文 hash 码， 2. 密钥为二进制数据流，可逻辑上拆分为多个部分
后置条件	1. 服务器数据库保存密钥以及 ID 信息, D 码由 hash 码与冲突码构成
涉众利益	数据库：成功保存密钥以及 ID 用户：确保能够对信息进行解密
基本路径	1. 传入 hash 码以及密钥 2. 访问数据库并且建立连接 3. 查询冲突表获取冲突码 4. 将 hash 码与冲突码组合成为唯一 ID 5. 将密钥以及 ID 存入数据库中
扩展路径	冲突码达到上限，无法进行保存。提示维护者服务器需要升级
字段列表	密钥，密文 hash 值,冲突码，密钥 ID
设计规则	被依赖模块
未解决的问题	存在冲突上限，需要良好的 hash 算法
备注	无

表 4.2.7 图片解密用例规约

用例名称	图片解密
功能简述	将用户上传的密文图片进行解密
用例编号	Data Fortress 008
执行者	Server
前置条件	1. 用户上传的是已加密图片 2. 数据库中已经保存对应的密钥
后置条件	1. 服务器向用户返回解密成功的图片
涉众利益	数据库：提供解密需要的密钥 用户：希望得到解密后的图片
基本路径	1. 用户使用图片解密端口上传图片 2. 服务器接收图片 3. 分析图片获取 hashID,根据 hashID 在数据库中查询获取解密方法 4. 哈希接收结果并且读取密文中的冲突码，将哈希结果以及冲突码组合成为密钥索引 5. 从数据库读取密钥 6. 根据上传接口选择合适解密算法进行解密 7. 将解密结果返回给用户
扩展路径	数据库中没有密文对应密钥，解密失败
字段列表	密钥，密文 hash 值,冲突码，密钥 ID，解密算法，密钥索引
设计规则	单独模块
未解决的问题	数据库密钥丢失之后无法恢复
备注	无

4.3 性能需求

4.3.1 时间特性

图片加密过程时间特性：

1. 图片上传时间
2. 服务器预处理图片时间
3. 识别加密区域时间
4. 加密时间
5. 加密结果数据转换时间
6. 密文图片下载时间

图片解密过程时间特性：

1. 图片上传时间
2. 服务器预处理图片时间
3. 解密时间
4. 解密结果数据转换时间
5. 原文文图片下载时间

4.3.2 适应性

1. 前端与后端采用 http 进行信息交换，前后端内部发生改变两端仍可顺利交互
2. 后端采用 django 开发服务器，其适应能力强，在多种操作系统下仍可以使用，项目中已经测试操作系统 Windows 10，Ubuntu 18.04 LTS
3. 前端采用快应用的呈现方式，可在支持快应用框架的主流 Android 手机上运行

4.4 界面需求

① 登录注册界面：

提供给用户完善清晰的登录注册界面，方便用户登录注册使用，便于用户查看本地登录信息。界面原型示意图见图 4.4.1

“用户名”部分采用文本输入框接收用户输入

“密码及重复密码”部分采用密码输入框接收用户输入的两次密码

“注册按钮”部分采用自定义的 button 组件监听用户点击

“已有账号，去登陆”部分采用自定义的 button 组件监听用户点击

图 4.4.1 注册界面

图 4.4.2 用户界面

② 用户信息界面：

提供给用户查看个人信息，提供隐私相册的入口，账户管理及登录信息等功能。界面原型示意图见图 4.4.2

顶部背景图为快应用的图标，采用层叠结构，上层叠放用户头像，用户登录名。

中间“隐私相册”“分享详情”等采用自定义的按钮结构，监听用户点击事件。

③ 功能选择界面：

提供给用户选择程序功能的可交互性良好的界面，方便用户选择人脸照片加密，文件 OCR 加密隐私信息保护，加密证件夹等多种隐私信息防护功能。

顶部“All Function”主题图片轮播图部分采用 swiper 组件进行隐私防护的热点新闻展示

中间“人脸数据保护”等四个按钮部分采用图片作为按钮监听用户的点击事件

底部“首页”及“我的”采用可滑动的 tab 页组件，监听用户的点击及手势动作，滑动或点击均可切换标签页。

图 4.4.3 功能选择界面

④ 人脸加密界面：

提供用户上传人脸照片的界面，可选择直接拍摄照片上传或从相册中选择已有照片文件上传，提供上传前预览照片的功能，并能够将加密后隐去人脸的图片返回给用户。

⑤ 自动隐私保护界面：

提供用户选择上传隐私图片的界面,并且自动检测隐私区域，并能够将加密后的结果返回给用户。

⑥ 卡通加密界面：

提供卡通加密的界面，系统会自动选择合适的贴图样式，并通过人脸识别方式将人脸信息替换为卡通图片。

⑦ 证件夹加密界面：

提供证件夹信息加密功能的界面，可以检测出证件上的相应信息所处位置，并通过异或加密的方式将隐私信息全部隐去。

上述界面需求可归结为上传下载图片需求，因此均可使用如下的界面原型

顶部包含图片展示组件点击后可以进入到相册选择图片界面，选择后将替换改选择框展示在页面顶部

中间的四个按钮使用自定义的 button 组件，监听用户点击事件。

底部使用图片展示组件展示加密后的图片。

图 4.4.4 上传图片界面（图片中身份证为样片不涉及信息）

⑧ 隐私相册界面：

隐私相册的主界面是该用户加密后的图片小图预览，每张小图上有监听点击的组件，点击后进入大图预览界面，在大图预览界面长按出现对话框，长按可以选择分享或解密。

相册界面：

顶端具有下拉刷新组件，监听手势方向，如图 4.4.5 所示。

中间部分是嵌套在滚动列表中的 image 组件，每行三张，若照片数量超过一页可向下滚动，同时每张图片上还有监听点击动作的组件，点击后进入大图预览。

大图预览界面：

采用 swiper 组件实现大图预览功能，左右滑动可以切换展示的图片，在图片上有监听点击动作的组件，点击后退出大图预览功能。还有监听长按动作的组件，长按后弹出对话框，可以选择分享功能和解密功能，如图 4.4.6 所示。

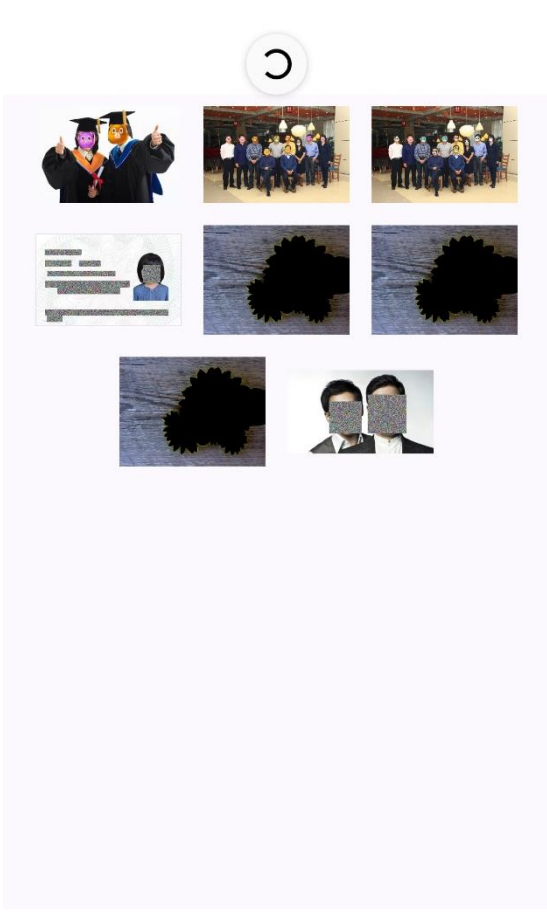

图 4.4.5 隐私相册界面

图 4.4.6 弹窗界面

⑨ 查看分享界面：

分享界面可以输入用户获取的分享码和对应的密码，从而获取到加密图片的原图未加密版本。

“分享码”部分采用文本输入框接收用户输入的分享码；

“密码”部分采用密码输入框接收用户输入的分享码密码；

“获取按钮”部分采用自定义的 button 组件监听用户点击事件。

中部采用 image 组件展示获取到的图片，如图 4.4.7 所示。

图 4.4.7 分享解密界面

4.5 接口需求

4.5.1 硬件接口

拍摄照片或视频时需要使用手机摄像头
调用方式：

使用快应用官方提供的接口，导入 system.media 模块，使用 media.takePhoto(OBJECT)方法调用手机摄像头进行照片拍摄。

4.5.2 软件接口

① 登录注册信息传输接口
② 图片信息上传接口
③ 返回加密后的图片接收接口
④ 相册管理交互接口
⑤ 错误信息处理接口

4.6 其他需求

4.6.1 可使用性

用户体验反馈与交互功能设计一直是团队的首要任务之一，因此 Data Fortress 在完成基本功能后，积极推动产品的测试与推广，并第一时间在产品中添加了反馈意见功能接口，及时收集用户的体验反馈，并进一步修改产品功能，提升用户体验。

从测试结果及用户反馈来分析，Data Fortress 在操作的便捷性以及照片加解密的速度和最终呈现效果上具有较大的优势，用户操作界面友好且对于加密图片的管理十分人性化。后台算法十分成熟稳定，可适用于几乎所有的相关领域，ToB 和 ToC 端均可使用，为项目日后的推广和应用打下了良好的基础。

4.6.2 可靠性

系统的可靠性是系统长期稳定运行的基石，对于我们的隐私保护应用更是如此。因此团队在进行系统架构设计时，对数据库及隐私保护加密算法设计了尽可能详尽的故障处理方案，以保证系统的可靠性。此外，团队还计划在下一阶段采用冗余技术来保证数据的可靠存储、系统可靠运行。

4.6.3 可移植性

“数据堡垒”依托快应用框架进行开发测试，可移植性高，经测试在各大手机系统平台都能够正常运行，系统级调用接口也可以正常使用，并实现设定的全部功能，且没有出现明显问题。

4.6.4 可维护性

对系统运行状况采用自动检测的方式进行实时观测，能够实现错误的快速定位和修复。同时，在服务器端存储用户的使用日志和服务器的运行日志，出现异常可以快速排查。

5 概要设计

5.1 处理流程

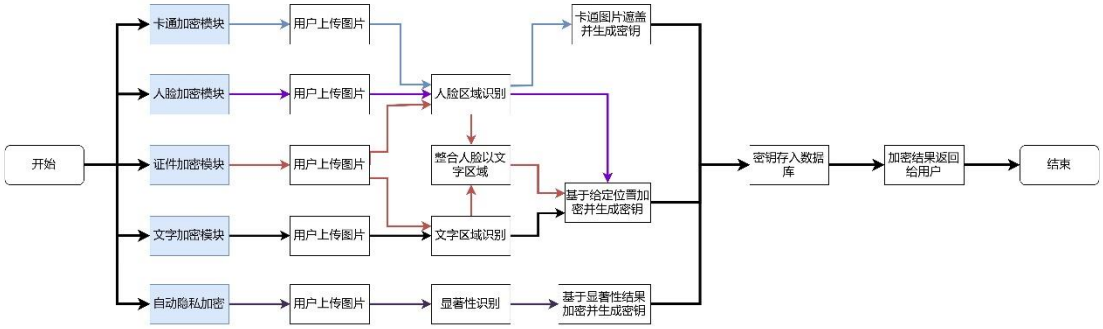

图 5.1.1 逻辑流程图

每个模块的处理逻辑中，加密之前都需要用户上传需要加密的图片。之后卡通加密、人脸加密、证件加密模块识别人脸区域，文字加密、证件加密识别文字区域，这里比较特别的是证件加密，其需要识别人脸个文字区域。通过上一步之后，加密模块获得需要加密的区域，卡通加密模块使用卡通图片遮盖人脸区域，人脸、证件、文字加密模块使用基于位置的加密算法进行异或加密。显著性加密识别显著性区域，之后根据显著性区域进行异或加密。所有模块加密之后生成密钥，将密钥保存到数据库之后再将加密结果返回给用户。至此处理流程结束。处理流程如图 5.1.1

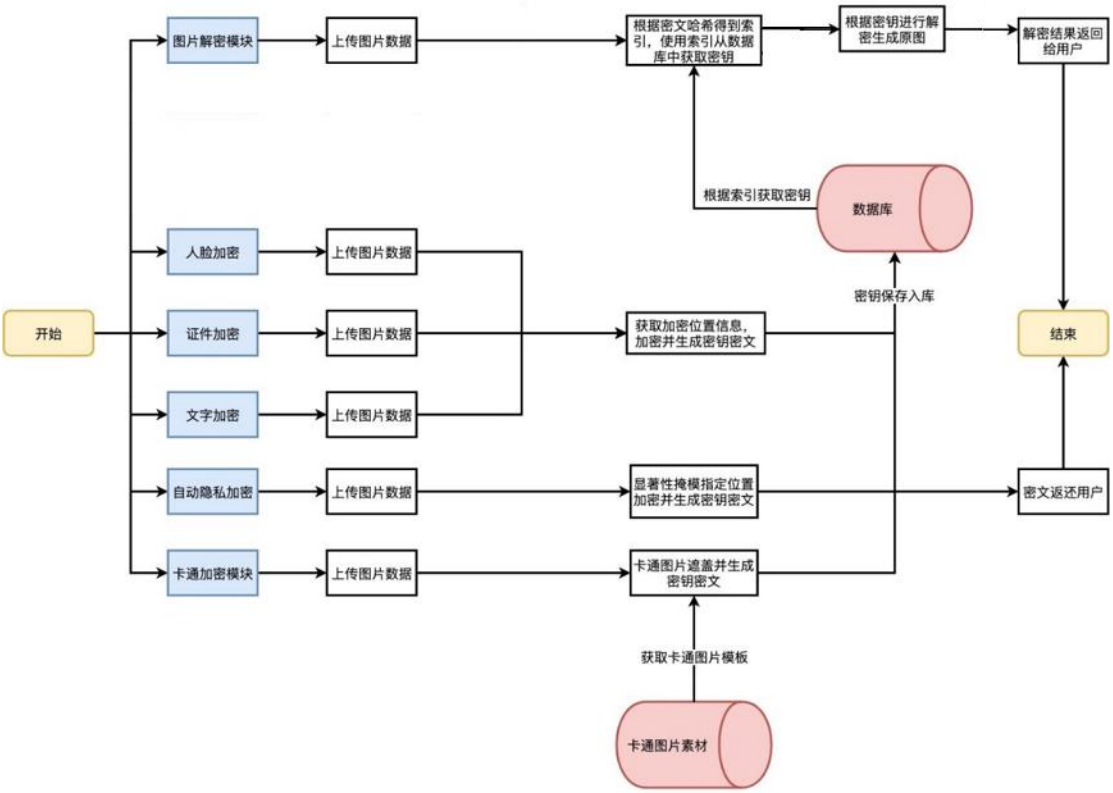

图 5.1.2 数据流程图

所有模块的数据处理流程刚开始都是用户上传图片数据。人脸、证件、文字、自动隐私加密模块使用加密算法生成密钥密文数据，卡通加密在加密过程中会访问卡通图片素材，使用已有素材进行加密，并且也生成密文密钥数据。所有的加密模块生成密文密钥之后都会将密钥保存到数据库中，将密文返回给用户。图片解密模块起始步骤也是用户上传图片数据，然后哈希密文得到数据库索引结果，再到数据库中获取密钥以及解密方法记录，之后使用密钥进行解密并且将解密结果返回给用户。所有的数据流程在图 5.1.2 中

5.2 总体结构设计

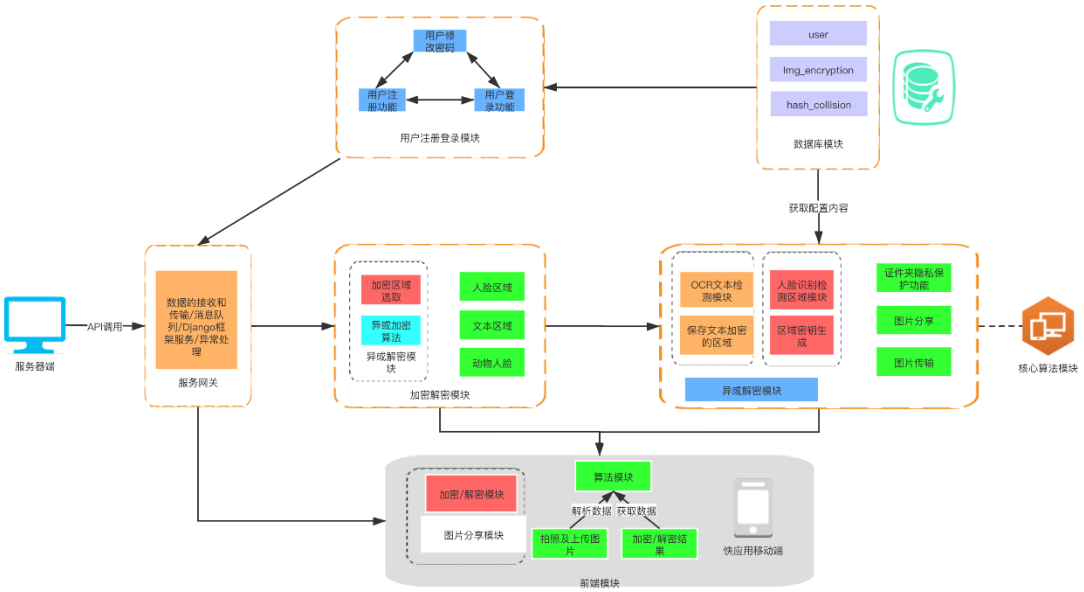

图 5.2.1 结构设计图

系统的主要内容集中在后端，服务器使用 django 作为基础，进行数据接受以及传输、各种异常处理，并且搭建核心算法模块。使用 mysql 作为数据库存储各种密钥密文数据。快应用前端负责与用户交互，上传下载图片，进行图片分享，相册查看。

5.3 功能设计

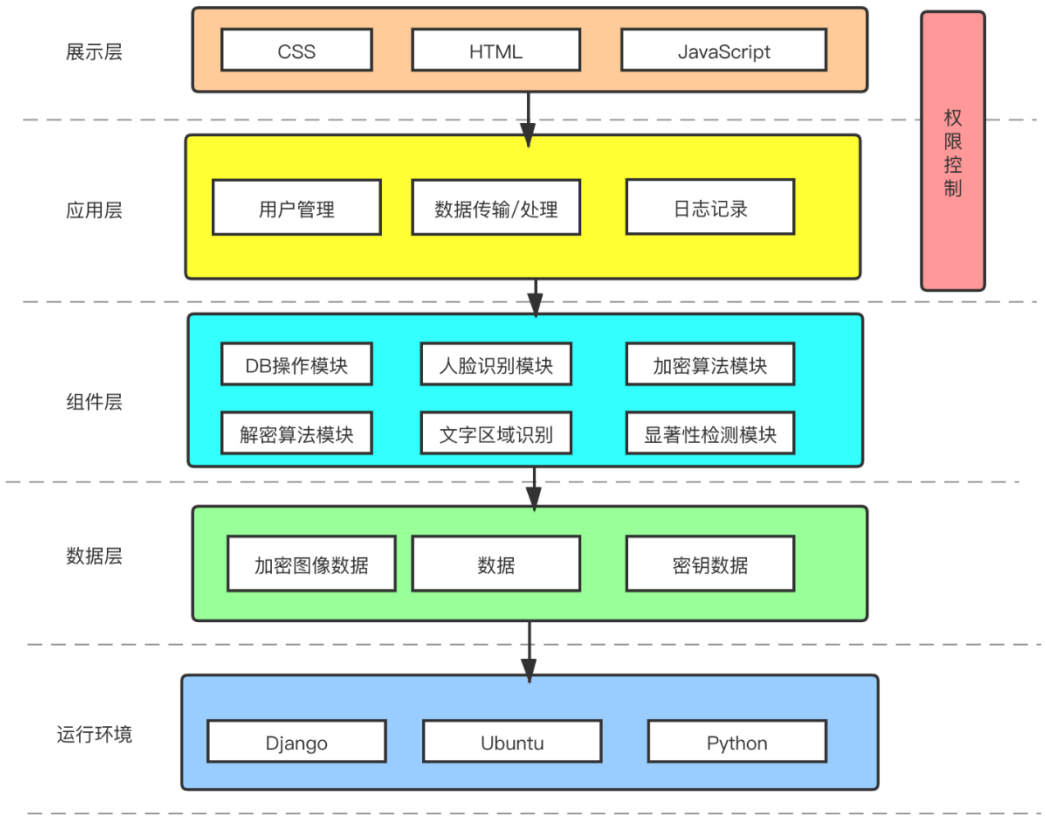

图 5.3.1 系统架构图

系统运行环境为 Ubuntu 18.04 LTS,并且使用 web 框架 django 作为基础，python 作为开发环境。

数据层中，对图像数据进行加密，密钥数据进行生成解析。

组件层中，对数据库进行操作，保存密文密钥数据，并且提供加密解密具体算法。

应用层中，负责对用户进行管理，上传下载图片数据，并且记录服务器运行日志。

展示层中，使用居于快应用的界面，其技术基础为 HTML\CSS\JavaScript 等。

5.4 用户界面设计

5.5 数据结构设计

Rect：加密方框，记录需要加密的方框位置，并且控制方框的极限位置，预留边界保存密钥信息的数据

Key:保存解密需要信息，包括 rect 数量，随机种子，每个方框的位置

Keyimg:存储密钥掩模数据，用于与密文或者原文进行异或解密加密操作

hashId: 密文 hash 得到的 hash 值

id: hashId 左移 8 位再加上冲突数量得到最终的密文 ID,通过此密文 ID,可以直接寻址到数据库中的密钥数据项

5.6 接口设计

5.6.1 外部接口

软件接口：

数据库接口：通过 pymysql 将与数据库进行连接，并且写入、读取、更新数据。

Sql 语句使用格式化字符串生成，所有格式化字符串保存在 back-end/algorithm/constant.py 文件中，并且模板都保存位一个常量字符串。字符串变量名如下

HASH_COLLISION_QUERY 读取 hash 冲突表 SQL 模板
HASH_COLLISION_INSERT 插入新数据到 hash 冲突表 SQL 模板
HASH_COLLISION_UPDATE 更新数据到 hash 冲突表 SQL 模板
IMG_ENCRYPTION_INSERT 插入新数据到 img_encryption 表,不包含 mask 内容 SQL 模板
IMG_ENCRYPTION_WITH_MASK_UPDATE 插入新数据到 img_encryption 表,包含 mask 内容 SQL 模板
IMG_ENCRYPTION_QUERY 查询 img_encryption 表数据 SQL 模板
INSERT_NEW_USER 插入新用户数据
GET_USER_DATA 获取用户数据
GET_USR_HASH_ID 获取单个用户上传的所有图片的 hashId

硬件接口：

拍摄照片或视频时需要使用手机摄像头

调用方式：

使用快应用官方提供的接口，导入 system.media 模块，使用 media.takePhoto(OBJECT)方法调用手机摄像头进行照片拍摄，使用 media.takeVideo(OBJECT)方法进行视频拍摄。

5.6.2 内部接口

服务器与客户端使用 url 作为接口，接口如下
upload_for_encrypt_face 上传图片人脸加密接口
upload_for_encrypt_text 上传图片文字加密接口
upload_for_encrypt_card 上传图片证件加密接口
upload_for_encrypt_cartoon 上传图片人脸卡通加密接口
upload_for_encrypt_salience 上传图片 salience 加密接口
upload_for_decrypt_img 上传加密图片解密接口
upload_for_decrypt_cartoon 上传人脸卡通加密图片解密接口
upload_for_decrypt_salience 上传 salience 图片解密接口
req_processed 获取图片处理结果接口
get_all_picture 获取用户上传的所有图片对应 hashID
get_encrypted_img_by_hashId 获取用户上传的所有图片对应 hashID
get_share_code 获取分享码
get_picture_by_share_code 使用分享码获取解密图片
gallery 获取用户相册

5.7 错误/异常处理设计

5.7.1 错误/异常输出信息

1. 异常:获取图片为空
   输出:图片为空,请重新选择
2. 异常:无对应的加密 ID
   输出:此图片不在数据库中,请检查图片或重新加密此图片
3. 异常:传输超时
   输出:传输超时,请检查网络连接
4. 异常:图片格式错误
   输出:不支持此格式,请检查文件类型
5. 异常:账号密码错误
   输出:账号/密码错误,请重新输入

5.7.2 错误/异常处理对策

1. 异常:获取图片为空
   处理对策:输出 warning,提示再试一次
2. 异常:无对应的加密 ID
   处理对策:输出 warning,提示用户检查图片
3. 异常:传输超时
   处理对策:尝试重传,并提示用户等待后再次尝试或选择压缩后上传
4. 异常:图片格式错误
   处理对策:提示用户重新选择文件
6. 错误:账号密码错误
   处理对策:提示用户重新输入

5.8 系统配置策略

安装 python 解释器，python 环境依赖如下
requests==2.22.0
web.py==0.40.dev0
numpy==1.17.0
PyMySQL==0.9.3
matplotlib==3.1.0
Django==2.2.1
six==1.12.0
opencv_python==4.1.0.25
dlib==19.19.0
keys==0.1-test
Pillow==6.2.1
web==0.6.0

5.9 系统部署方案

服务器端：

- 安装 python 解释器，以及上述的 python 依赖库
- 数据库：配置 mysql 80，并且创建需要的数据库 2 个。数据库结构在下文中展示
- 部署服务器端代码并且部署算法实现。
- 启动服务器，并且测试服务器功能

客户端：

- 打开快应用卡片即可使用

5.10 其他相关技术与方案

Saliency Map：将受到关注度最多的地方识别出来进行加密，这样就避免开了对整幅图片的加密，能够很大程度上减少对多余区域像素加密，时效性上会有很大的提升，同时能减少加密开销。

异或加密: 异或操作是对数据进行位操作，加密和解密都只要一步操作，使得加密效率高。

Dlib：dlib 人脸检测方法采用 64 个特征点检测，效果会好于 opencv 的方法识别率会更高，同时又有更高的准确性和时效性。

Yolo+：YOLO 最大的优点就是不用再提取 proposal，直接对全图进行回归，原先的两步流程变成了一步到位，速度快，准确性也有保证。

6 数据库设计

数据库中一共有 3 个表，分别为`user`，`img_encryption`，`hash_collision`。

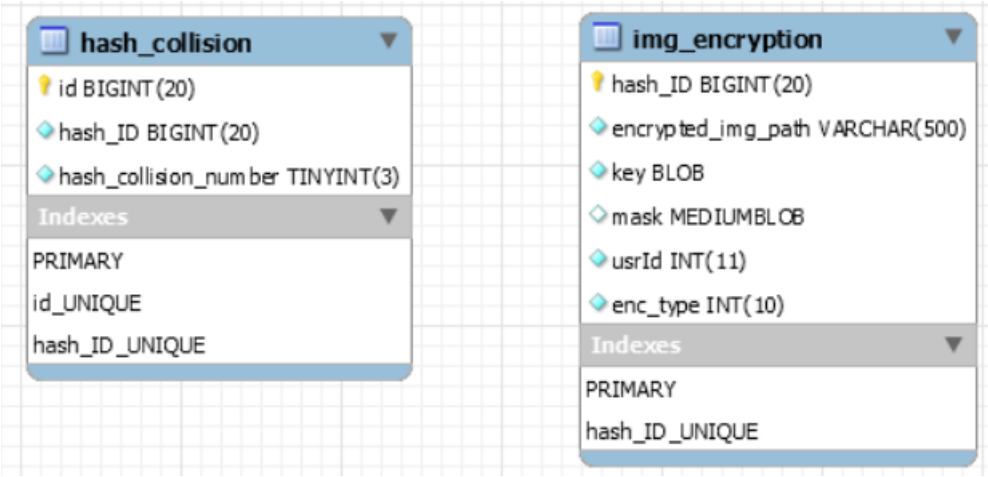

图 6.1 ER 图

表 6.1 哈希冲突信息表

表 hash_collision			
属性	数据类型	是否主键	备注
id	BIGINT(20)	是	
hash_ID	BIGINT(20)	否	hash 数值
hash_collision_number	TINYINT(3)	否	hash 数值冲突的数量

hash_collision 表保存 hash 冲突的信息，不同图片 hash 得到的结构可能相同，通过此表得到 hash 冲突的数量，解决 hash 冲突问题。

表 6.2 密钥表

表 img_encryption			
属性	数据类型	是否主键	备注
hash_ID	BIGINT(20)	是	图片加密的唯一 ID,通过算法计算 hash 得到
encrypted_img_path	VARCHAR(500)	否	加密后图片的保存路径
key	BLOB	否	加密之后图片的解密路径
mask	MEDIUMBLOB	否	使用 mask 加密解密方法的缩小化 mask
usrId	INT(11)	否	上传用户 ID
Enc_type	Int(10)	否	加密方法记录

img_encryption 表保存图片加密的密钥，密文地址，二进制掩模数据。usrId 属性是参考 user 表的外键，用来记录上传用户的信息。并且对 usrId 建立了索引，方便搜索。

7 详细设计

7.1 登录注册模块

7.1.1 功能描述

1. 索取用户信息

Data Fortress 属于面向用户的数据安全服务平台，更强调对数据的安全防护功能，因此真实姓名，身份证号等信息不是必须在此处获取的信息，目前所需的用户信息仅有用户的手机号码，用来进行登录验证和密码找回等必要环节。

2. 登录注册

为了降低用户的跳出率提升转化率，所以为用户提供了友好快捷的注册和登陆方式，只需输入用户名（手机号）和密码即可完成登录，并且能够选择登录后记住密码，下次进入时即可自动完成登录环节。

7.1.2 性能描述

登录和注册环节对网络性能的要求较低，只需要保证用户使用的设备可以正常连接互联网上传和接收信息即可。

对设备性能要求低，只要保证用户使用的设备可以正常加载快应用即可。

7.1.3 输入

1. 注册环节：只需输入用户名或手机号+两次验证密码
2. 登录环节：用户名+密码

7.1.4 输出

使用 Toast 提示用户登录注册情况：

1. 若登录成功则输出“登录成功，欢迎”字样，并自动进入用户信息界面
2. 若登录失败则提示“登录失败，请核对用户信息”字样

7.1.5 程序逻辑

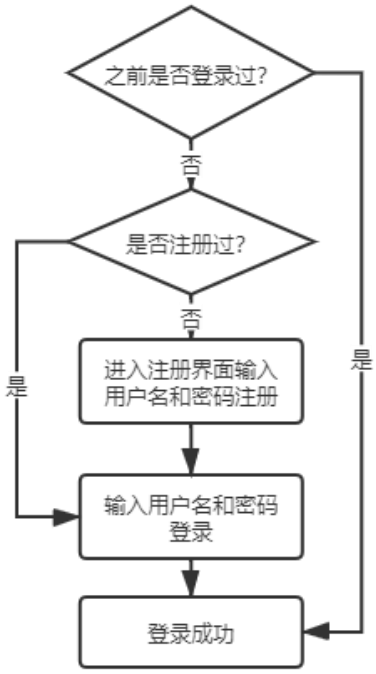

图 7.1.1 登陆注册逻辑

7.2 人脸区域加密模块

7.2.1 功能描述

将用户上传图片中所有的人脸区域进行加密。人脸加密模块首先使用 dib 中的人脸识别接口获取人脸区域，之后使用加密算法对图片进行加密，并且保存密钥然后返回加密结果

7.2.2 性能描述

图片上传下载需要用户使用的设备可以正常连接互联网，要求用户网络信号比较好。

图片人脸区域检测需要消耗比较大的计算资源，使用现有 Ubuntu 服务器的计算时长在用户可接受范围内。

图片加密过程消耗计算资源少，可以在极短时间内完成。

7.2.3 输入

用户上传需要加密的图片，以及用户自身的 ID 信息。

7.2.4 输出

加密之后的图片，图片中的人脸区域都被随机噪声覆盖。解密图片的密钥

7.2.5 程序逻辑

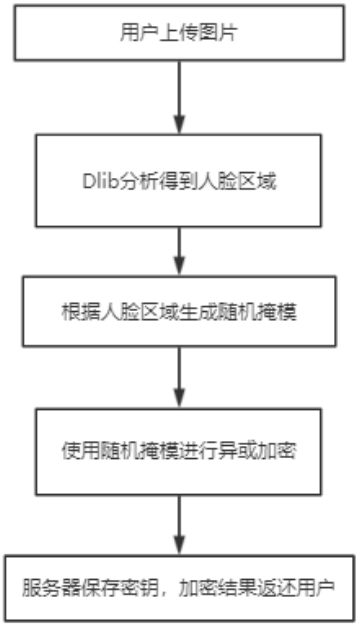

图 7.2.1 人脸加密程序逻辑

7.3 文字区域加密模块

7.3.1 功能描述

将用户上传图片中所有的文字区域进行加密。首先使用 yilo 网络获取文字区域，并且将文字区域归一化表示为矩形框，然后生成矩形框数组，之后使用加密算法对图片进行加密，并且保存密钥然后返回加密结果

7.3.2 性能描述

图片上传下载需要用户使用的设备可以正常连接互联网，要求用户网络信号比较好。

图片人脸区域检测需要消耗比较大的计算资源，使用现有 Ubuntu 服务器的计算时长比较短，识别得到文字区域速度比较快。

图片加密过程消耗计算资源少，可以在极短时间内完成。

7.3.3 输入

用户上传需要加密的图片，以及用户自身的 ID 信息。

7.3.4 输出

加密之后的图片，图片中的文字区域都被随机噪声覆盖。解密图片的密钥

7.3.5 程序逻辑

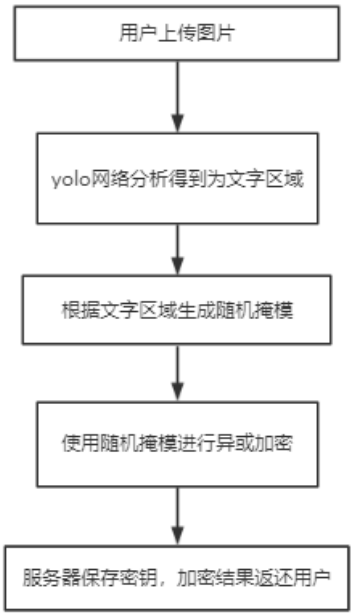

图 7.3.1 文字加密程序逻辑

7.4 卡通人脸加密模块

7.4.1 功能描述

将用户上传图片中所有的人脸区域进行卡通图片遮挡。卡通人脸加密模块首先使用 dib 中的人脸识别接口获取人脸区域，之后使用加密算法对图片人脸区域进行卡通图象遮挡并生成解密密钥，并且保存密钥然后返回加密结果

7.4.2 性能描述

图片上传下载需要用户使用的设备可以正常连接互联网，要求用户网络信号比较好。

图片人脸区域检测需要消耗比较大的计算资源，使用现有 Ubuntu 服务器的计算时长在用户可接受范围内。

图片加密过程消耗计算资源少，可以在极短时间内完成。

密钥大小比较大，占用 KB 量级的存储空间

7.4.3 输入

用户上传需要加密的图片，以及用户自身的 ID 信息。

7.4.4 输出

加密之后的图片，图片中的人脸区域都被卡通人脸遮挡。以及解密图片的密钥。

7.4.5 程序逻辑

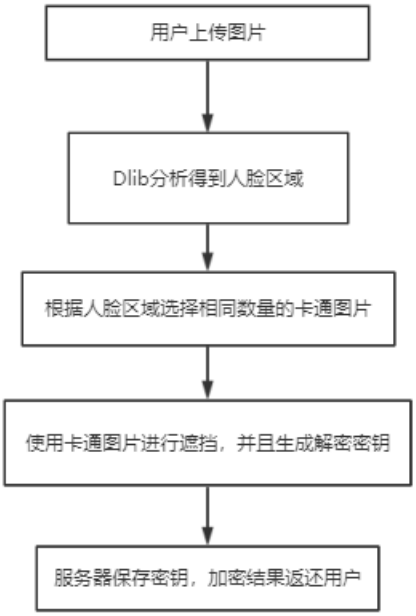

图 7.4.1 卡通加密程序逻辑

7.5 自动隐私区域加密模块

7.5.1 功能描述

将用户上传图片中的自动识别到的隐私区域进行加密。使用 salience model 对图片进行显著区域提取，得到一张显著图掩模。之后将显著区域进行加密，加密结束之后保存密钥然后返回加密结果

7.5.2 性能描述

图片上传下载需要用户使用的设备可以正常连接互联网，要求用户网络信号比较好。

图片显著区域区域检测需要消耗比较适中的的计算资源，使用现有 Ubuntu 服务器的计算时长比较短。

图片加密过程消耗计算资源少，可以在极短时间内完成。

密钥大小比较大，占用 KB 量级的存储空间

7.5.3 输入

用户上传需要加密的图片，以及用户自身的 ID 信息。

7.5.4 输出

加密之后的图片，图片中的显著区域被加密。以及解密图片的密钥。

7.5.5 程序逻辑

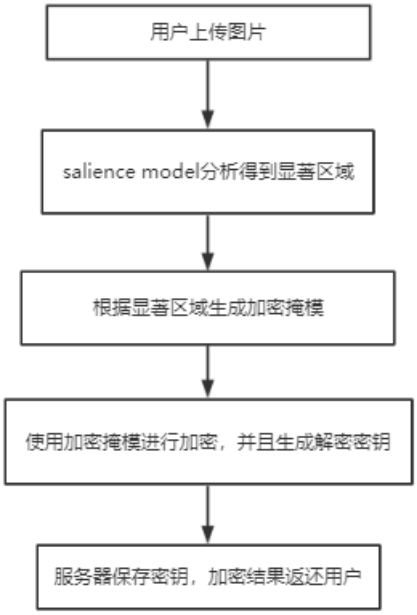

图 7.5.1 自动加密程序逻辑

7.6 密钥保存模块

7.6.1 功能描述

将密钥保存到数据库中。使用密文哈希方法进行索引，并且提供了哈希冲突的解决方法。

7.6.2 性能描述

密钥保存采用 mysql 数据库，需要占用的空间不大，存储时间短。

7.6.3 输入

需要保存的密文以及对应的密钥

7.6.4 输出

无

7.6.5 程序逻辑

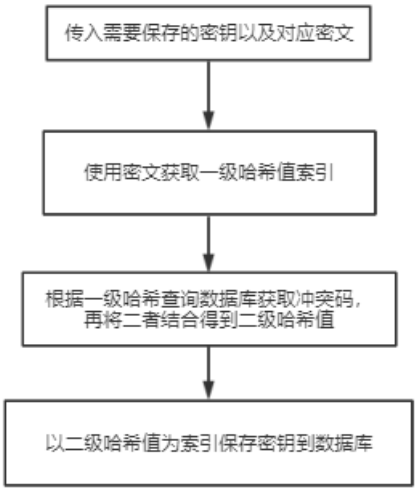

图 7.6.1 密钥保存逻辑

7.7 图片解密模块

7.7.1 功能描述

将图片进行解密，首先使用密文图片获取一级哈希，然后利用一级哈希与图片的哈希冲突数量合成二级哈希结果。利用二级哈希结果作为索引访问数据库，获取图片的加密方式信息，以及密钥。根据数据库中保存的加密方式，系统自动匹配加密算法。然后调用对应的解密算法进行密文解密。

7.7.2 性能描述

解密需要访问数据库，需要数据库支持。
数据读取之后，对图片进行解密需要计算资源少，解密速度比较快。

7.7.3 输入

加密之后的密文图片。

7.7.4 输出

解密之后的图片

7.7.5 程序逻辑

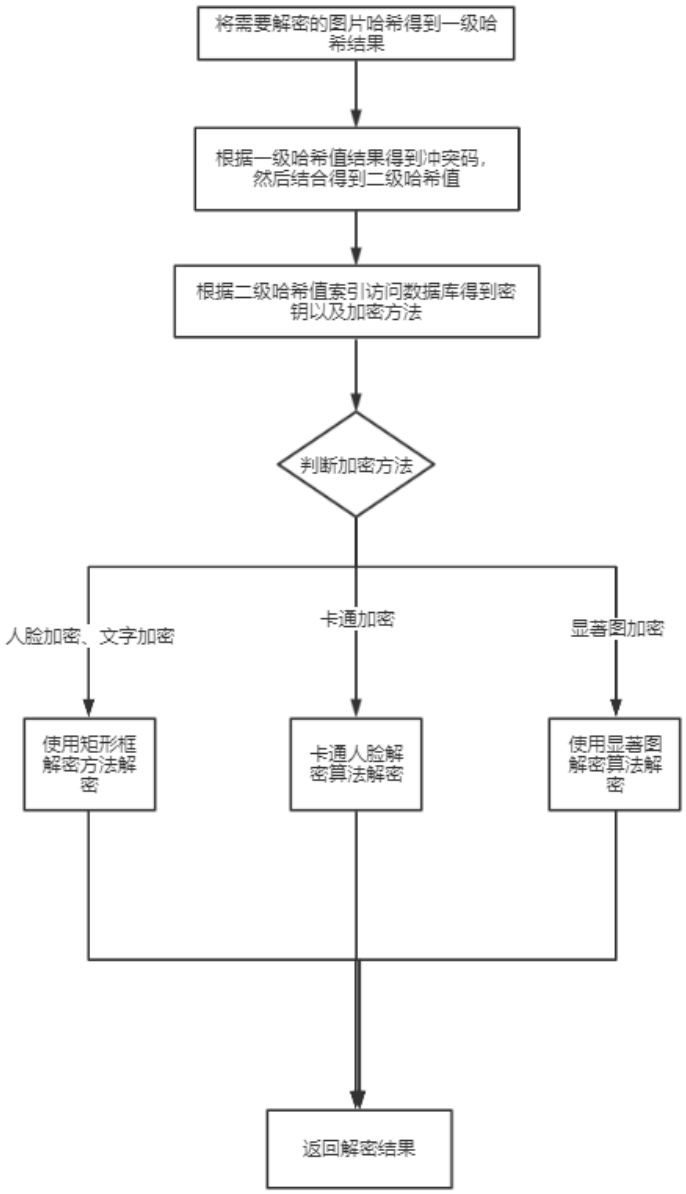

图 7.7.1 解密逻辑

7.8 图片分享模块

7.8.1 功能描述

用户通过上传图片或者查看相册指定需要分享的图片，系统为每一张图片生成对应的分享码。

用户获取分享码以及分享密钥之后，可以分享给其他用户。其他用户可以利用分享码以及密钥在系统中获取图片解密结果。

通过对解密模块的控制，结合分享码机制，能够实现数据分发功能。

7.8.2 性能描述

1. 获取分享码：需要访问服务器数据库，读取密文信息，之后生成分享码。计算资源占用少，处理时间短
2. 获取分享图片：需要调用数据解密模块，使用数据库，计算资源占用少，处理时间短。

7.8.3 输入

1. 获取分享码：需要分享得图片二级哈希索引。
2. 获取分享图片：分享码以及分享密钥。

7.8.4 输出

1. 获取分享码：分享码以及分享密钥。
2. 获取分享图片：分享图片得解密结果。

7.8.5 程序逻辑

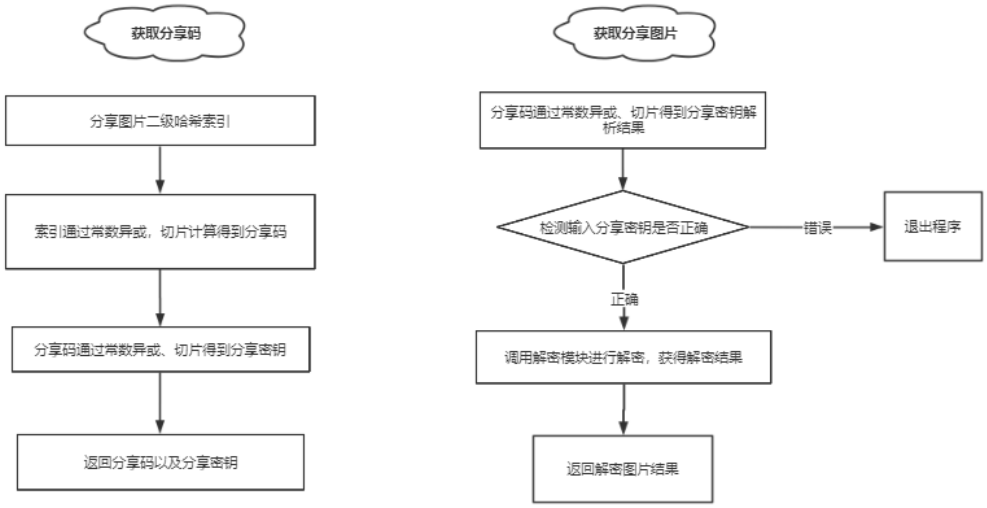

图 7.8.1 分享逻辑