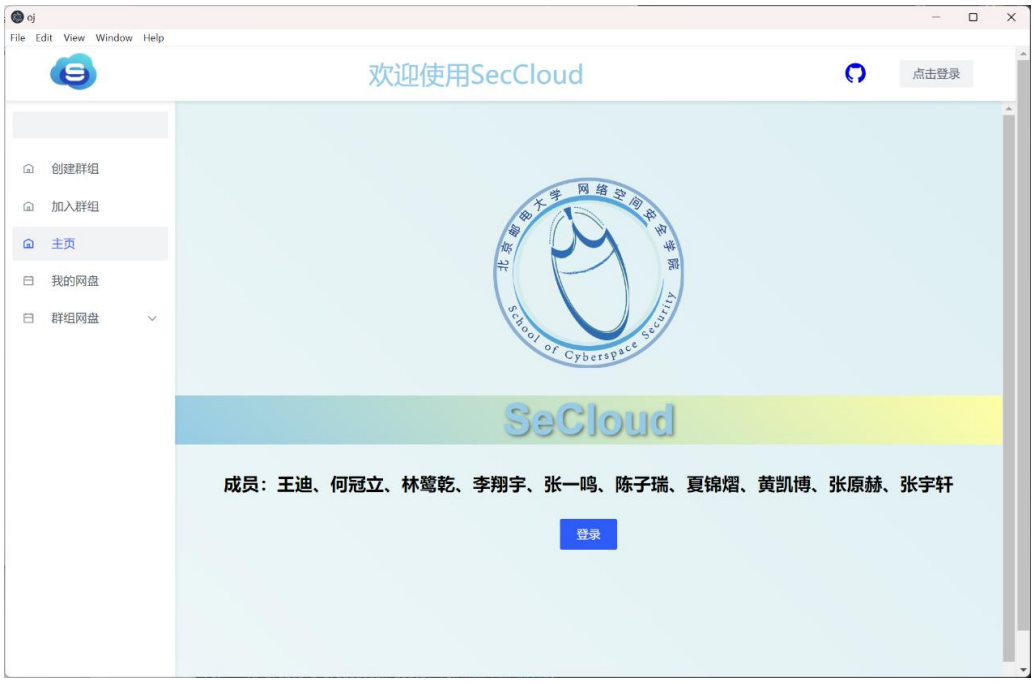
概要设计报告

SeCloud ——您的安全网盘应用

一、 设计成果、亮点概要

（一）项目简介

SeCloud 是一款集安全认证、加密传输和加密存储于一体的高级网盘应用，专注于密码学设计，实现用户中心的加密、认证体制，SeCloud 服务器无法窃取任何用户网盘数据。用户可获得与传统网盘应用相近的网盘体验。该项目从底层构建安全网盘应用，自主设计实现客户端与服务器的通信协议，并在该通信协议上增加层层密码学协议及技术，实现了安全、可信、便捷的网盘应用。



（二）项目实现

我们按照指导书，实现了所需的所有功能以及安全需求：

- ✓ **网盘基本操作：**支持所有类型文件的上传、下载、删除，文件夹创建、树形网盘结构管理。

- ✓ **安全性保证：**我们的安全网盘系统存在三类“密码”：**用户口令、会话密钥、文件加密密钥**。我们实施了一种基于硬件安全模块（U 盾）的文件主密钥管理机制，利用 DH 密钥交换协议协商会话密钥，并采用了层级密钥生成策略，保证每个文件的加密密钥都不相同；设计用户口令登录，进行多级安全保障。
- ✓ **文件加密管理：**使用层级密钥结构，由 U 盾中的主密钥生成不同的文件加密密钥，实现针对文件的“一次一密”。
- ✓ **端到端的透明加密传输：**通过经过 CA 的公钥验证服务器数字证书后，便可与服务器通过 DH 密钥协商协商会话密钥，一旦协商成功，所有通信都是加密的。我们使用 python socket 变成，实现了支持多用户同时访问一个服务器、并经过透明加密保证的通信信道，这个通信信道类似于 HTTPS 协议。
- ✓ **共享网盘设计：**我们实现了共享网盘，每个网盘的文件安全性寓于网盘文件加密密钥之中。网盘文件加密密钥由 U 盾存储，用户组内的不同用户通过 P2P 加密信道或 U 盾物理传递进行网盘文件加密密钥的传输，确保安全性。
- ✓ **安全登录认证机制：**登陆 SeCloud 需要口令，下载 SeCloud 中的文件需要 U 盾。因此我们有着双重保护机制。只有口令以及 U 盾都被攻击者窃取后，安全系统才被攻破，这大大增加了系统的安全性。并且我们支持 SeCloud 口令找回，Email 登陆机制。
- ✓ **多端登录设计：**多端登录必须在不同终端同步文件加密密钥以及用户口令，用户口令由用户记忆，而文件加密密钥由于其安全性，切忌存储在服务器，因此我们借鉴银行系统实现，将文件加密密钥存储到 U 盾中，用户在多端登录时需要携带 U 盾。
- ✓ **防篡改和防止中间人攻击：**我们使用 HMAC 进行完整性验证，防止被攻击者篡改。我们引入可信第三方 CA，进行服务器的证书验证，确保服务器的认证性，防止中间人共计。
- ✓ **加密数据找回机制：**在我们的设计框架下，U 盾是经过相关安全机构（例如银行）进行安全存储的，当平常使用的 U 盾丢失后，可以通过向安全机构申请拿到备份，此时安全机构会对用户的生物特征、社会特征、身份证等特征进行全方位验证，保证拿到备份 U 盾后后能够恢复文件加密密钥，找回加密数据。
- ✓ **用户友好的前端界面：**我们使用 electron 框架打包了基于 vue3+python 开发的客户端，

支持跨平台、界面美观、用户友好。

（三）亮点

- ✓ **用户中心的加密和认证体制：**SeCloud 服务器无法窃取任何用户数据，从而保证了用户的隐私和数据安全。
- ✓ **基于 U 盾的多重安全机制：**通过硬件安全模块（U 盾）和多种密钥管理机制，增加了系统安全性。
- ✓ **一次一密的文件加密管理：**采用层级密钥结构和 U 盾生成的主密钥，确保每个文件都有独一无二的加密密钥。
- ✓ **端到端的透明加密传输：**利用 DH 密钥交换协议和数字证书，实现了与 HTTPS 相似的加密通信信道。
- ✓ **P2P 加密信道和 U 盾物理传递的共享网盘设计：**确保即使在多用户共享环境中，每个文件的加密密钥也都得到了安全传输和存储。
- ✓ **双重保护的登录认证机制：**除了传统的口令登录，还需要 U 盾进行文件下载，大大增加了安全性。
- ✓ **完整性验证和中间人攻击防护：**通过使用 HMAC 和可信的第三方 CA，进一步保证了数据和通信的安全性。
- ✓ **灵活的加密数据找回机制：**通过与安全机构（如银行）合作，用户可以在 U 盾丢失后经过全方位验证后拿到备份，保证数据不会丢失。
- ✓ **适应多端登录的设计：**借鉴银行系统，将文件加密密钥存储在 U 盾中，实现多端登录时的高级安全同步。

（四）技术栈：

1.后端技术栈：

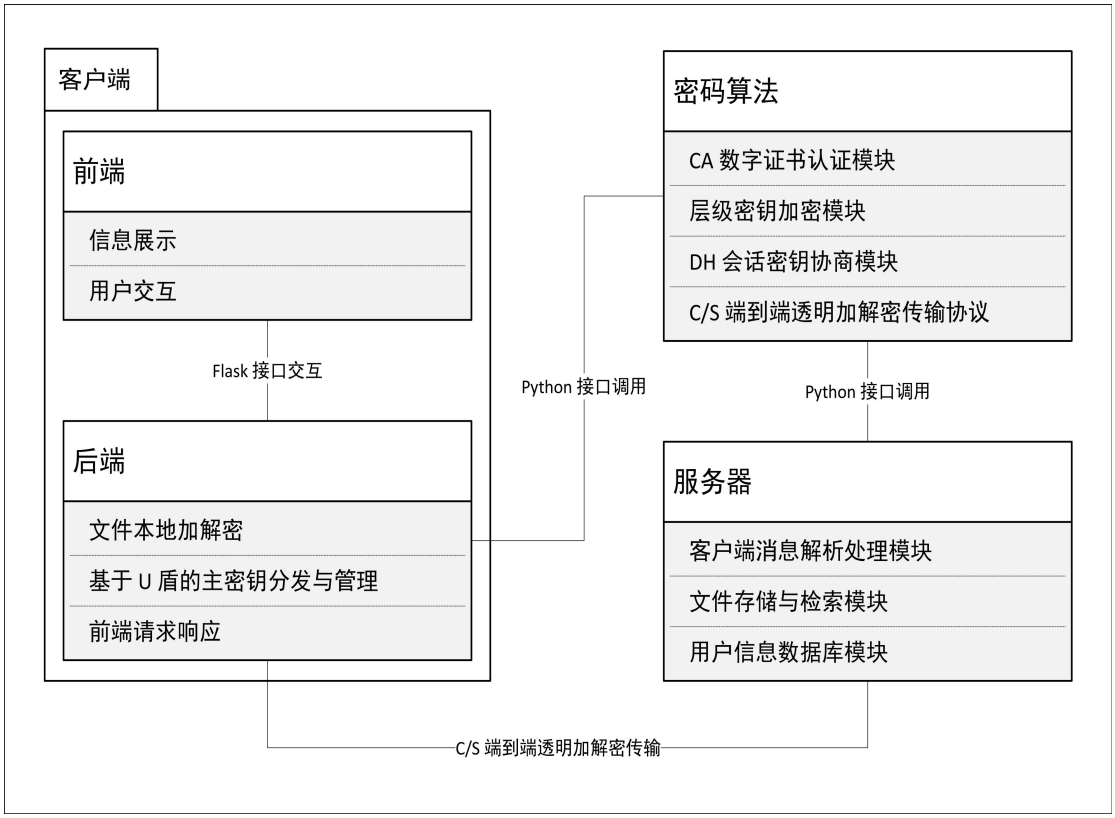
- ✓ Python 开发：使用 Python 作为后端开发语言，因为其具有丰富的库支持和快速开发的能力。
- ✓ 密码库协议、算法编程：使用 python 的成熟密码库进行密钥生成、加密和解密操作，以及实现各种安全协议。

- ✓ Socket 网络编程：使用 Python 的 socket 库进行底层的 TCP/IP 通信，确保高效、稳定的数据传输。
- ✓ Flask 开发框架：使用 Flask 轻量级 Web 框架来处理 HTTP 请求和响应，简化了 Web 服务开发。
- ✓ 数据库管理：可能会用到关系数据库用于存储用户信息、文件元数据等。
- ✓ 消息队列：使用服务器端使用 socket 从底层实现消息队列

2.前端技术栈：

- ✓ Vue3 前端框架：使用 Vue3 进行前端开发，因为其提供了响应式数据绑定和组件化开发。
- ✓ Electron 跨平台 APP 开发：使用 Electron 打包应用，实现跨平台的桌面应用。
- ✓ Axios：用于客户端前端与客户端后端之间的本地通信，因为其 API 简单且支持 Promise。
- ✓ 前端路由：使用 Vue Router 进行页面路由管理，优化用户导航体验。
- ✓ 状态管理：使用 Vuex 进行应用级状态管理，简化组件之间的数据共享。
- ✓ UI 框架：使用 acro design UI 框架，以快速构建美观且一致的用户界面。
- ✓ CSS 预处理器：使用 Sass 等 CSS 预处理器，简化 CSS 编写。
- ✓ 版本控制：使用 Git 进行版本管理，以跟踪代码更改和协助多人协作。

二、 系统整体简要描述



系统架构图如上图所示，下面对各个模块进行简要介绍：

(一)客户端：

客户端包含前端和后端两部分，两部分之间使用 flask 进行交互，其中前端主要负责提供用户友好的图形界面，用于接收用户输入的信息和展示返回的结果。它使用了 Flask 这样的 Web 框架，使得与后端的交互更加方便和灵活。 后端使用前端得到的信息和服务器进行交互，使用 U 盾内的信息对文件进行加解密，并将从服务器调回的信息存入客户端或返回给前端展示，以实现数据的完整性和一致性。值得注意的是，客户端的前端虽然使用 web 框架，但是前端界面只与本地的客户端后端 python 程序通信，不会与外界网络通信。

(二)服务器：

服务器从客户端后端得到的信息提取出命令和数据，根据相应的命令做出存储信息，调取文件，返回用户请求的信息等动作。服务器的实现保证了系统的稳定性和高效性，能够应

对大量客户端的请求，提供更好的用户体验。

(三)客户端服务器通信：

客户端和服务器通信时使用 python 实现的 C/S 端到端透明加解密传输协议，使用 DH 会话密钥协商，层级密钥加密以及 CA 数字整数验证模块。这种协议的实现可以有效地防止通信过程中的数据泄露和篡改，提高了系统的安全性。

(四)客户端前后端分离：

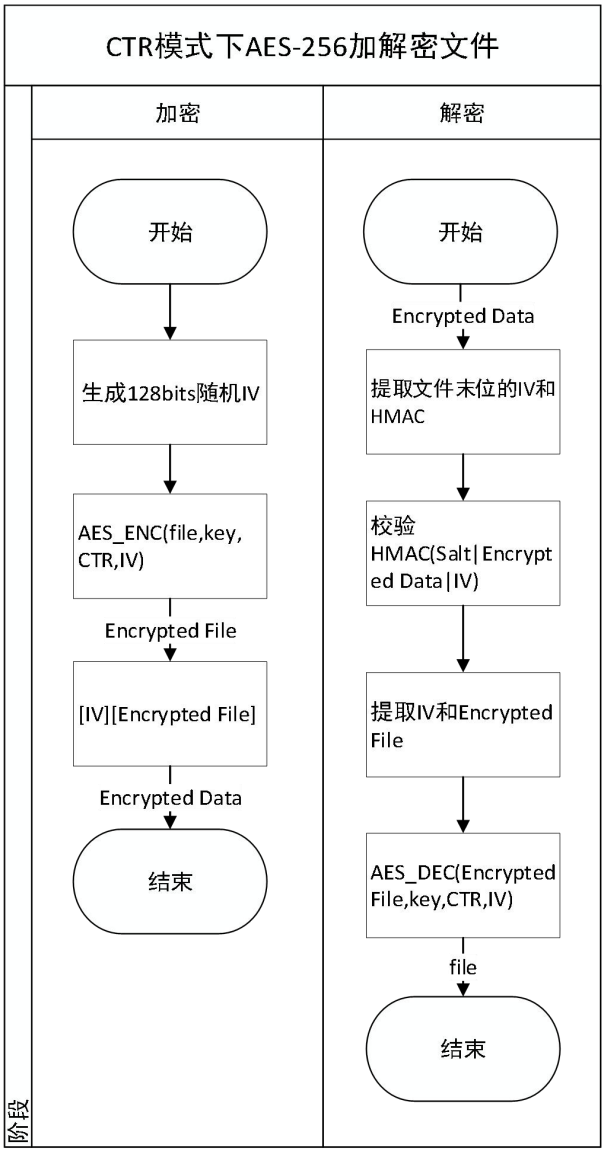
客户端使用前后端分离的模式，以确保系统的模块化和可扩展性。这种架构使得客户端前端和客户端后端分别负责与用户的交互和与服务器进行数据交换，提高了系统的性能和安全性；客户端和服务端之间的通信采用存在 CA 的加密通信，保证通信的安全性。

三、系统分模块简要描述

(一)　客户端：

1. 文件本地加解密

包含文件加解密的内容，使用了 AES-256、数据的封装、HMAC 等技术，具体流程如下图所示，相关密码算法于后文详细介绍：



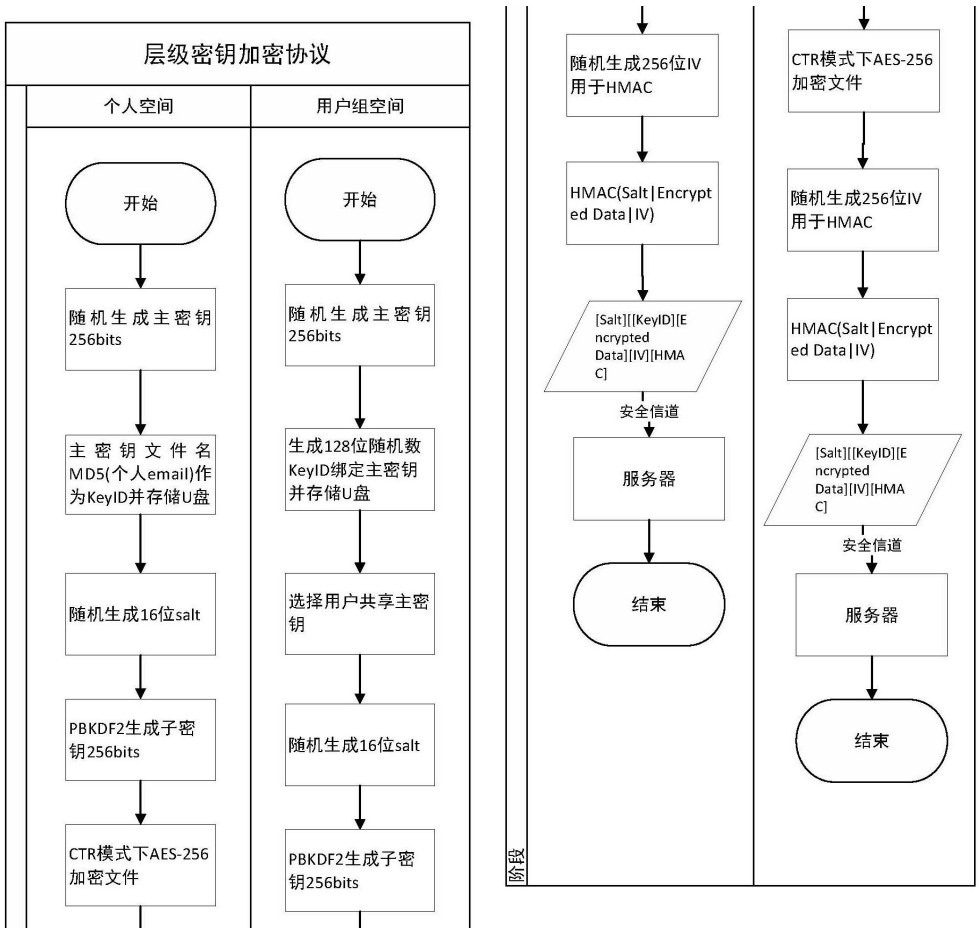
2. 基于 U 盾的主密钥分发与管理

系统采用主密钥-会话密钥-文件密钥的三层密钥体系。主密钥由硬件安全模块管理,用于导出会话密钥;会话密钥用于客户端和服务器间通信加密;文件密钥对每个文件独立生成,实现了一次一密。

我们使用 U 盾的文件加密密钥（主密钥），对每个文件生成不同的文件加密密钥，我们的层级密钥设计思路如下：

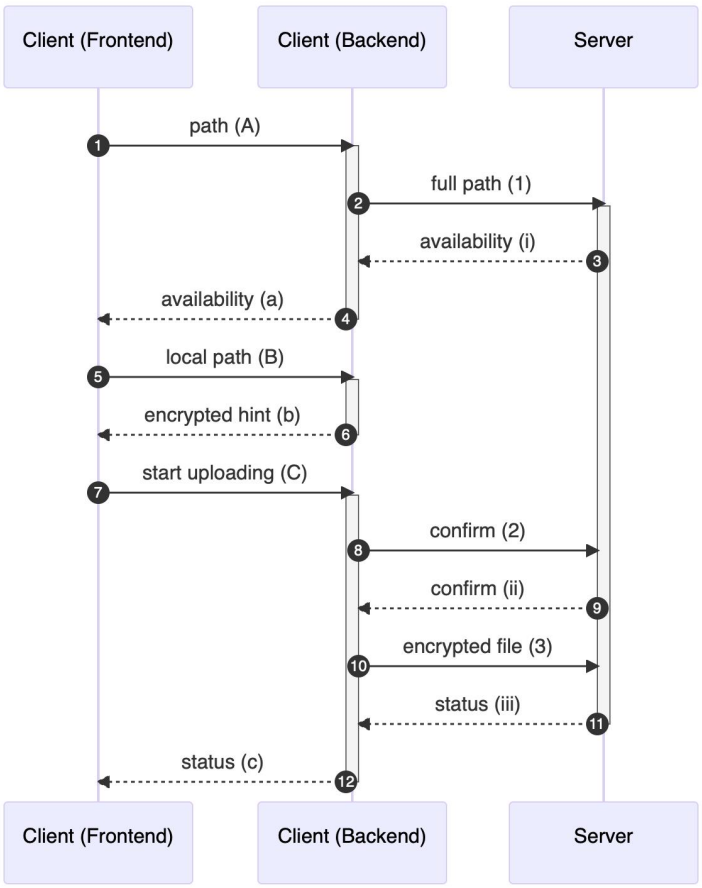
设计层面	描述
设计原因	1. 为确保数据的安全性和完整性，避免数据泄露和篡改 2. 强化密码安全，保障密码的强度和随机性的同时方便密钥管理 3. 既要单用户多终端登录实现文件解密密钥同步，又要阻止服务器解密文件而不能存储密钥在服务器上
设计方法	1. 利用PBKDF2算法结合盐值生成随机主密钥 2. 使用AES在CTR模式进行文件加密，CFB模式构建安全信道 3. 通过HMAC确保文件的完整性和安全性 4. 主密钥物理存储在U盘中，与网络隔离以增强安全性和便携性 5. 主密钥命名采用随机数或者MD5(email)命名，增加其安全性，减少主密钥信息泄露
安全性证明	1. PBKDF2算法可以防止彩虹表攻击，增强密码的安全性 2. AES加密算法提供了高级的数据加密保障，确保数据的保密性 3. HMAC验证可以防止数据被篡改，保证数据的完整性 4. U盘物理存储主密钥能有效阻止网络攻击，提高系统的安全性
防御攻击	1. 防御中间人攻击和重放攻击，通过构建安全信道和验证文件完整性实现 2. 防止密钥猜测攻击，通过PBKDF2和强随机数生成方法减少这种可能性 3. 防止网络攻击，通过U盘物理存储主密钥实现网络和物理隔离。

流程图如下：（右图接左图）



3. 前端请求响应

包含了客户端前后端之间的交互和客户端后端和服务器的交流，在后端得到信息后，将信息发回给前端进行界面更新以及显示。



(二)服务器：@夏锦熠 （每个小标题都要有一张图以上）

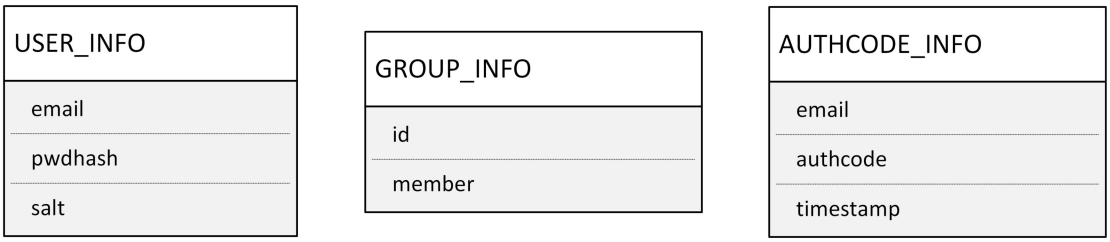
1.客户端消息解析处理模块

客户端消息解析处理模块用于解析客户端经由 C/S 端到端透明传输协议传递的操作请求，并返回请求结果给客户端。

2.文件存储与检索模块

文件存储与检索模块用于将用户与群组加密的文件保存在服务器，区分每个用户或群组对文件的访问权限，并记录用户空间与群组空间的文件目录结构、文件大小、文件修改日期等信息。

3.用户信息数据库模块



图：数据库模式

用户信息数据库模块用于存储并维护用户及群组相关的各种信息。数据库由三张表构成。其中，用户信息表（USER_INFO）用于存储用户的邮箱（email）、口令哈希值（pwdhash）与口令盐值（salt）信息；群组信息表（GROUP_INFO）用于存储群组标识符（id）、群组成员邮箱（member）信息；验证码信息表（AUTHCODE_INFO）用于存储指定邮箱（email）的验证码（authcode）与发送时间戳（timestamp）信息。

(三)安全层：

1. CA 数字证书模块

在我们的网络磁盘空间项目中，CA（证书授权中心）数字证书模块是用于验证用户和服务器之间通信的真实性和安全性的关键组件，该模块包含以下内容。

1.1 生成和分发证书：本模块负责生成数字证书，并将其分发和服务器，以用于后续的通信安全验证。

1.2 验证数字签名：CA 模块将验证数字签名，以确保通信双方的身份和数据的完整性。

1.3 公钥基础设施：利用服务器模拟 CA 设施来管理和维护数字证书，包括证书的申请、颁发和撤销。

2. 层级密钥加密模块

密钥加密模块是设计来确保数据存储和传输的安全性和保密性。

2.1 主密钥与会话密钥：实现一个层级密钥系统，其中主密钥用于生成文件加密密钥，而文件加密密钥则用于加密实际的文件。

2.2 密钥的安全管理和更新：用硬件安全模块（U 盾）模拟硬件安全模块（U 盾）实现密钥的安全管理和周期性更新，以提高系统的安全性。

3. DH 会话密钥协商模块

该模块是用于在通信双方之间安全地协商和建立会话密钥的。

3.1 密钥协商：利用 Diffie-Hellman 算法进行密钥协商，以安全地生成和共享会话密钥。

3.2 证书验证模块：通过配合数字证书和签名验证，增强 DH 协商的安全性，防止中间人攻击。

4. C/S 端到端透明加解密传输协议

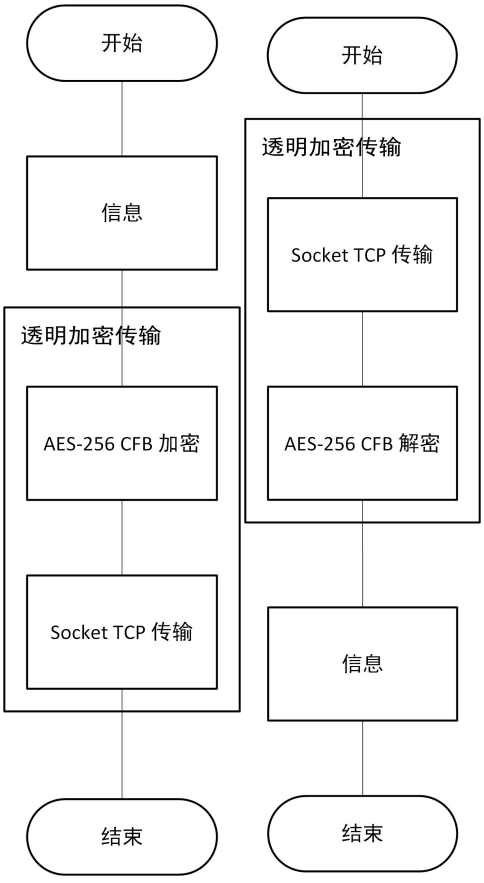
该模块用于确保客户端和服务器之间的数据传输是安全和透明的。

4.1 透明加解密框架模块：自定义加解密传输协议，使得数据在客户端和服务器之间传输时可以自动进行加密和解密，而无需用户干预。

4.2 数据完整性验证模块：HMAC 确保数据在传输过程中的完整性，防止数据被第三方篡改。

4.3 数据加密模块：AES-256 的 CFB 模式加密确保数据在传输过程中的机密性，防止数据被第三方窃取后获得重要信息。

(四)通信层：



图（a）发送信息过程　　图（b）接收信息过程

通信层用于服务器与客户端之间的通信。为了方便其他模块通信，通信层设计了一套端到端的透明传输协议。如上图所示，客户端与服务器只需要调用信息发送接口，而不需要关注 CA 认证、密钥交换、对称加密的具体流程。

(五)　UI 页面：

1.登陆注册页面：

1.1 登陆界面

采用 Vue 3 和 Arco Design 组件库构建页面基本框架，通过与后端的 API 交互来实现用户登录、注册、找回密码等操作，并根据不同的情况给出不同的消息提示，登陆成功跳转到主页面，同时提供了密码可视化、密码重置等多项便捷操作功能。

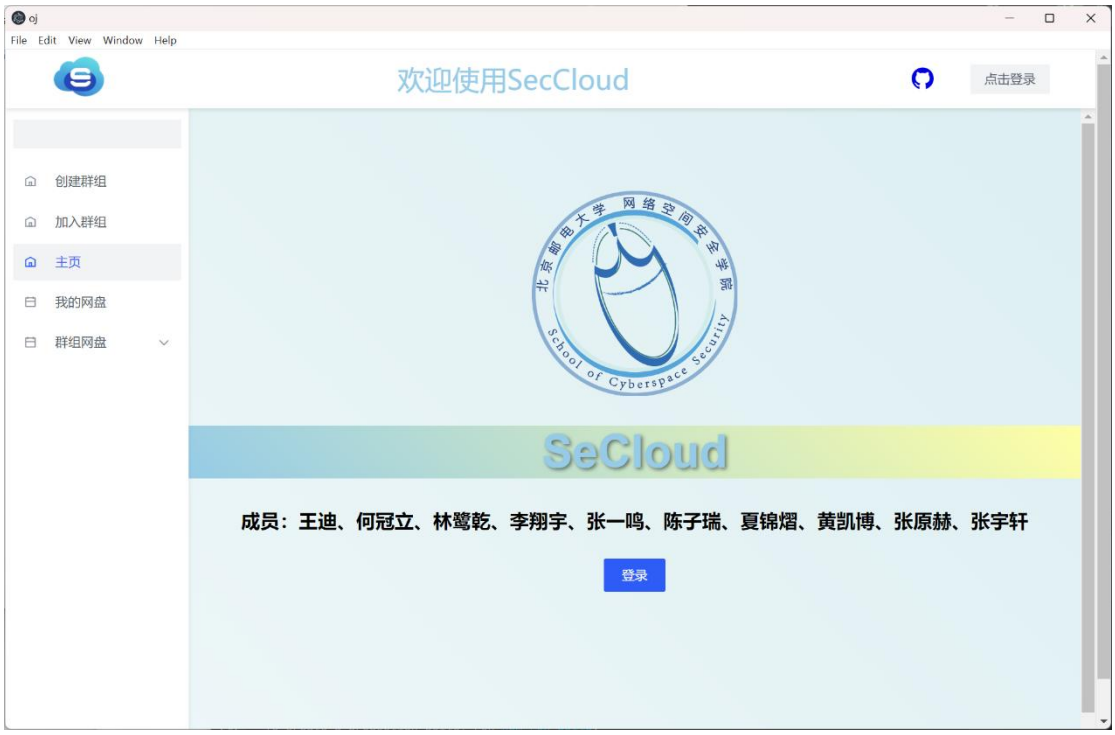
1.2 注册界面及密码找回页面

类似登陆界面，同样采用了 Vue 3 和 Arco Design 组件库来构建，实现了一个注册/密码找回页面，通过与后端的 API 交互来实现用户填写邮箱、密码、密码确认和验证码来进行注册或找回的功能，并提供密码二次确认、密码可视化、重置等便捷功能，且验证码可以实时快速准确安全地发送到注册人提供的邮箱。

2.UI 主页面：

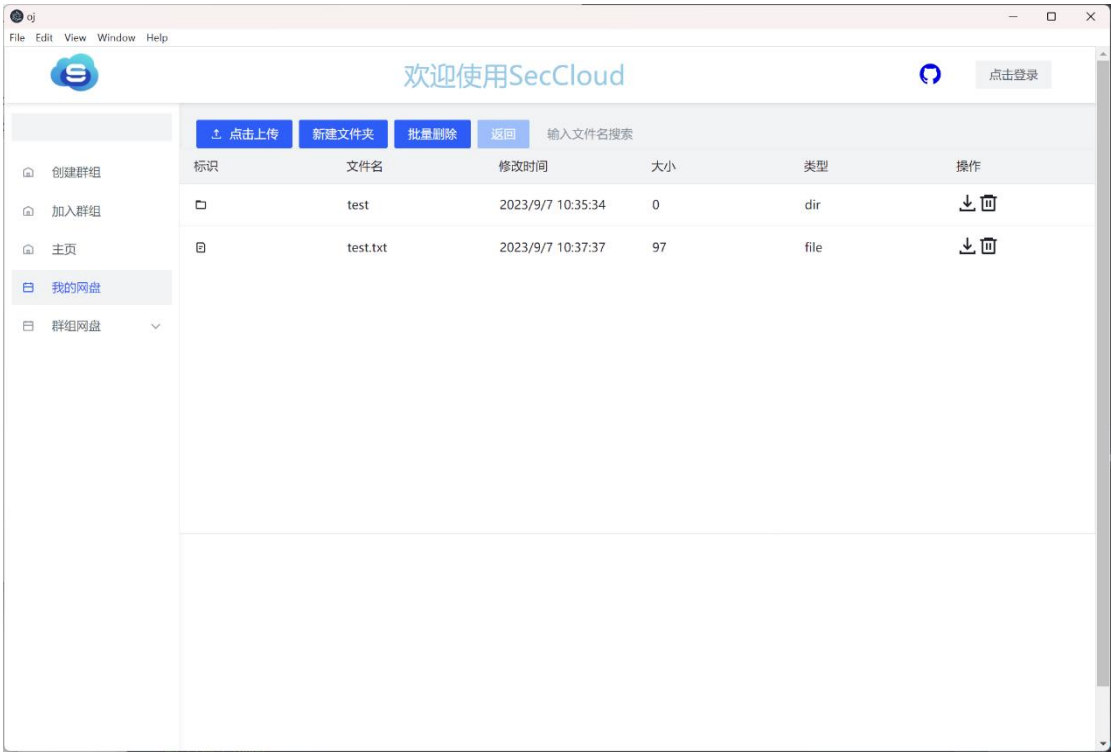
2.1 主（home）页面

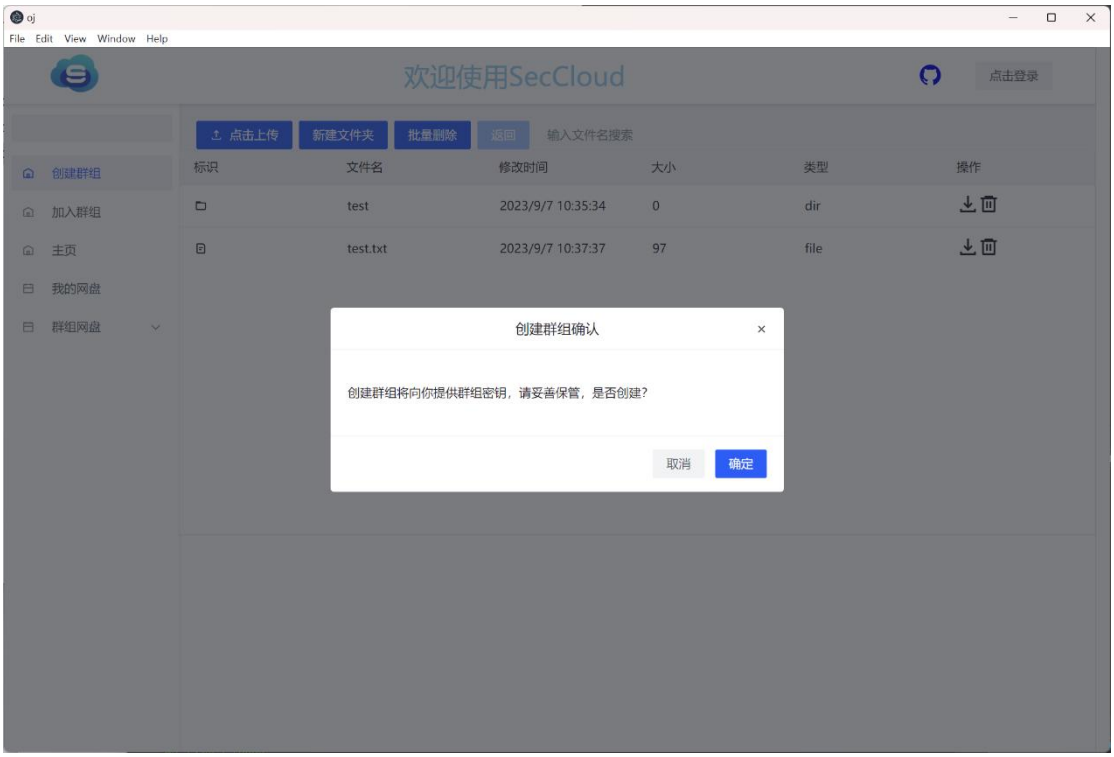
以网络空间安全学院院徽、项目名 SeCloud 及成员名单为主要元素，配以渐变帧动画的背景，提供一个颜色对比舒适、美观简洁的网盘主页面，并提供快捷的登录按钮。

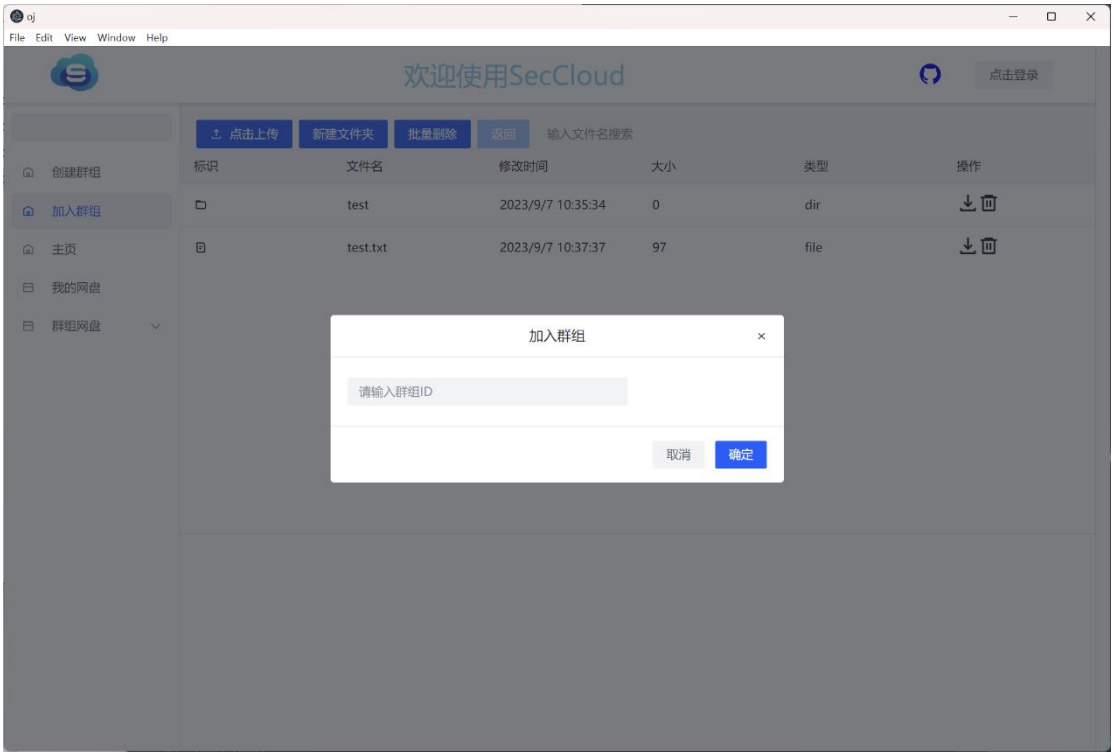


2.2 网盘页面

网盘页面采用 Vue3 及 Acro design 组件库，配以 TypeScript 前端逻辑，实现了一个美观快捷用户友好的网盘操作页面，具体来说，提供上传、新建文件夹、批量删除、返回等多项操作并为每个操作配有相应的响应弹框。对于存储在网盘上的文件，提供图标标识、文件名、修改时间、大小、类型、操作的具体信息。在左边框为用户提供创建群组、加入群组、群组选择及隐藏等按钮进行相应操作。


oj
File Edit View Window Help
欢迎使用SecCloud
点击登录
点击上传
新建文件夹
批量删除
返回
输入文件名搜索
创建群组
加入群组
主页
我的网盘
群组网盘
标识
文件名
修改时间
大小
类型
操作
test
2023/9/7 10:35:34
0
dir
test.txt
2023/9/7 10:37:37
97
file


oj
File Edit View Window Help
欢迎使用SecCloud
点击登录
点击上传
新建文件夹
批量删除
返回
输入文件名搜索
创建群组
加入群组
主页
我的网盘
群组网盘
标识
文件名
修改时间
大小
类型
操作
test
2023/9/7 10:35:34
0
dir
test.txt
2023/9/7 10:37:37
97
file
创建群组确认
创建群组将向你提供群组密钥，请妥善保管，是否创建?
取消
确定



2.3 其他部件

我们同时为项目 SeCloud 设计了一个应用图标，其中背景蓝色的云代表 Cloud，而云上的字母既像 S 又像 E，正好对应项目产品名的前缀 Se。在 head 栏的右边，我们也提供了 Github 的图标，并设置了一个超链接，点击该图标即可跳转到我们项目的 Github 代码仓库。