第十八届“挑战杯”全国大学生课外学术科技

作品竞赛“揭榜挂帅”专项赛

参 赛 作 品

基于量子比特加速检索的

密态云盘设计与实现

参赛学校： 华中科技大学

参赛成员：王子尧、罗理恒、杨明欣、王文恺、

金航宇、刘昕、陈威、郭星宇

指导老师： 王蔚、徐鹏、李霖

**摘 要**

随着互联网技术的快速发展，云存储技术已得到了广泛应用。与传统存储介质相比，云存储具有海量存储空间、数据备份便携、多端同步等特点。随着云盘的广泛应用，其安全性也逐渐得到关注，同时促使了以可搜索加密为核心的密态云盘存储技术研究。该技术目前受限于经典计算机算法的高时间复杂度，在密文数据搜索及密文数据处理上存在性能瓶颈。近年来，随着量子计算技术的发展，通过量子算法加速通用计算机处理性能已成为了一个新兴的研究方向。针对上述设想，提出了一种基于Grover算法的可扩展量子搜索加速方案，可针对乱序数据进行搜索，相比传统方案有指数级性能提升，在大规模数据处理方面具有更高的效率。本项目在设计上从用户的信息隐私安全出发，构建了存储用户数据的密态云盘，并融合了基于Grover算法的可扩展量子搜索方案达到了更好的搜索性能。本项目在具体实现中完成了云盘的web端和Android端，在量子计算平台上对量子搜索算法的可行性进行了实验验证。

结合现实应用考虑，本方案合理设想了未来量子计算平台的应用场景，有机结合了数据库功能与量子计算，深度优化了数据库搜索速度，发掘量子计算平台的商业价值。

**关键词：**Grover算法、量子搜索加速、密态云盘、量子计算应用

**ABSTRACT**

With the rapid development of Internet technology, cloud storage technology has been widely applied. Compared to traditional storage media, cloud storage features massive storage space, portable data backup, and multi-device synchronization. As the widespread use of cloud drives continues, its security has gradually gained attention, leading to research on secure cloud storage technologies with searchable encryption as the core. However, this technology is currently limited by the high time complexity of classical computer algorithms, resulting in performance bottlenecks in searching and processing encrypted data. In recent years, with the development of quantum computing technology, accelerating the processing performance of classical computers through quantum algorithms has become an emerging research direction. In response to the above scenario, a scalable quantum search acceleration solution based on the Grover algorithm is proposed, which can search unordered data. Compared to traditional solutions, it provides exponential performance improvement and higher efficiency in large-scale data processing. This project designs a secure cloud drive for storing user data, taking into account user information privacy and integrates a scalable quantum search solution based on the Grover algorithm to achieve better search performance. The project implements both a web and Android interface for the cloud drive and experimentally verifies the feasibility of the quantum search algorithm on a quantum computing platform.

Considering real-world applications, this solution reasonably envisions future applications of quantum computing platforms, combining database functionality with quantum computing, optimizing database search speed, and exploring the commercial value of quantum computing platforms.

**KEYWORDS:** Grover algorithm, Quantum search acceleration, Encrypted cloud storage, Quantum computing application.

**目 录**

[1 绪论 8](#_Toc137713123)

[1.1 研究背景与意义 8](#_Toc137713124)

[1.2 研究现状 9](#_Toc137713125)

[2 系统设计 12](#_Toc137713126)

[2.1 系统概述 12](#_Toc137713127)

[2.2 系统功能设计 13](#_Toc137713128)

[2.3 模型构建 14](#_Toc137713129)

[2.4 接口设计 14](#_Toc137713130)

[2.5 小结 17](#_Toc137713131)

[3 算法设计 18](#_Toc137713132)

[3.1 可搜索与分享的加密模型KS2E 18](#_Toc137713133)

[3.2 基于Grover算法的可扩展量子对称加密加速搜索方案 20](#_Toc137713134)

[3.2.1 Grover算法 20](#_Toc137713135)

[3.2.2 可扩展的量子搜索算法 21](#_Toc137713136)

[3.2.3 量子搜索功能设计 22](#_Toc137713137)

[3.3 小结 23](#_Toc137713138)

[4 系统实现 24](#_Toc137713139)

[4.1 系统开发环境 24](#_Toc137713140)

[4.2 系统项目搭建 24](#_Toc137713141)

[4.2.1 服务端设计实现 24](#_Toc137713142)

[4.2.2 客户端设计实现 26](#_Toc137713143)

[4.2.3 密态云盘部署 27](#_Toc137713144)

[4.2.4 量子平台连接 31](#_Toc137713145)

[4.3 系统操作流程 31](#_Toc137713146)

[4.4 系统展示 33](#_Toc137713147)

[4.4.1 系统web端展示 33](#_Toc137713148)

[4.4.2 系统Android端展示 35](#_Toc137713149)

[4.5 小结 38](#_Toc137713150)

[5 系统测试与评估 39](#_Toc137713151)

[5.1 网盘功能测试 39](#_Toc137713152)

[5.2 量子搜索测试 41](#_Toc137713153)

[5.2.1 实验环境 41](#_Toc137713154)

[5.2.2 实验测试 41](#_Toc137713155)

[5.2.3 实验过程 42](#_Toc137713156)

[5.2.4 实验评估 44](#_Toc137713157)

[5.3 小结 48](#_Toc137713158)

[6 优势分析与应用推广 49](#_Toc137713159)

[6.1 网盘安全性分析 49](#_Toc137713160)

[6.2 网盘搜索速度分析 49](#_Toc137713161)

[6.3 方案创新性分析 49](#_Toc137713162)

[6.4 方案商业价值分析 50](#_Toc137713163)

[6.5 应用推广优势 50](#_Toc137713164)

[7 总结 52](#_Toc137713165)

[参考文献 53](#_Toc137713166)

**图 目 录**

[图 2‑1 系统功能框架 13](#_Toc137632581)

[图 2‑2 系统的模型结构图 14](#_Toc137632582)

[图 2‑3 登录、注册接口与UserInfo结构定义 15](#_Toc137632583)

[图 2‑4 发送搜索令牌、搜索接口 15](#_Toc137632584)

[图 2‑5 获取文件列表、文件内容、创建文件夹接口 16](#_Toc137632585)

[图 2‑6 获取节点id、上传文件接口 16](#_Toc137632586)

[图 2‑7 一轮、二轮分享接口 17](#_Toc137632587)

[图 2‑8 删除、修改密码接口 17](#_Toc137632588)

[图 3‑1 双向结构索引 18](#_Toc137632589)

[图 3‑2 量子搜索算法门电路设计图 22](#_Toc137632590)

[图 4‑1 数据库设计概念图 25](#_Toc137632591)

[图 4‑2 Web端页面设计图 26](#_Toc137632592)

[图 4‑3 https配置图 30](#_Toc137632593)

[图 4‑4 存储服务配置图 30](#_Toc137632594)

[图 4‑5 系统操作流程图 32](#_Toc137632595)

[图 4‑6 登录页面 33](#_Toc137632596)

[图 4‑7 注册页面 33](#_Toc137632597)

[图 4‑8 注册成功 33](#_Toc137632598)

[图 4‑9 登录成功 34](#_Toc137632599)

[图 4‑10 文件上传成功 34](#_Toc137632600)

[图 4‑11 查看文件内容 34](#_Toc137632601)

[图 4‑12 分享令牌上传成功 34](#_Toc137632602)

[图 4‑13 文件接收成功 35](#_Toc137632603)

[图 4‑14 文件搜索 35](#_Toc137632604)

[图 4‑15 注册、登录页面 35](#_Toc137632605)

[图 4‑16 文件上传、创建文件夹、文件列表展示页面 36](#_Toc137632606)

[图 4‑17 文件内容浏览、搜索页面 36](#_Toc137632607)

[图 4‑18 用户信息展示、修改密码、设置、关于页面 37](#_Toc137632608)

[图 4‑19 蓝牙连接页面 37](#_Toc137632609)

[图 4‑20 转存、接收页面 38](#_Toc137632610)

[图 5‑1 本地模拟器运行量子程序进行存储 43](#_Toc137632611)

[图 5‑2 量子计算云平台查看4比特量子程序运行结果 44](#_Toc137632612)

[图 5‑3 量子计算云平台查看8比特量子程序运行结果 44](#_Toc137632613)

[图 5‑4 经典搜索与量子搜索之间理想搜索次数的差异 46](#_Toc137632614)

[图 5‑5 量子算法与经典搜索算法的加速比 46](#_Toc137632615)

[图 5‑6 不同比特搜索时理想的搜索时间 47](#_Toc137632616)

**表 目 录**

[表 4‑1 系统开发环境 24](#_Toc137573006)

[表 4‑2 请求路径url说明 25](#_Toc137573007)

[表 4‑3 Vue文件说明表 26](#_Toc137573008)

[表 5‑1 注册成功后用户信息 39](#_Toc137573009)

[表 5‑2 上传文件后的文件密钥和用户双向索引结构 39](#_Toc137573010)

[表 5‑3 test.txt的部分搜索索引（一） 39](#_Toc137573011)

[表 5‑4 test.txt的部分搜索索引（二） 40](#_Toc137573012)

[表 5‑5 搜索令牌 40](#_Toc137573013)

[表 5‑6 新的身份验证信息和重新加密的用户密钥 40](#_Toc137573014)

[表 5‑7 分享令牌和RSA加密后的文件密钥 41](#_Toc137573015)

[表 5‑8 实验环境 41](#_Toc137573016)

[表 5‑9 不同比特数对应平台搜索正确的概率值 45](#_Toc137573017)

[表 5‑10 不同比特理想搜索次数的对比（搜索次数向下取整） 45](#_Toc137573018)

[表 5‑11 量子搜索与经典搜索空间对比 47](#_Toc137573019)

# 绪论

## 1.1 研究背景与意义

随着云存储技术的不断发展和完善，云存储应用产生了一系列需求，如云文件的跨平台访问和用户之间的高效文件分享等[1]。越来越多的用户选择云存储服务来管理他们的文件，并广泛使用基于云存储服务开发的云盘软件。然而，伴随大量个人隐私数据的存储，网盘泄露个人隐私数据的风险也在逐渐增加。例如，2022年12月11日苹果公司的iCloud系统遭受网络攻击，并导致了约200张好莱坞女性艺人的私人照片遭受泄露。2017年[33]，百度网盘的数据也遭到攻击，大量私人信息，包括身份证、驾驶证照片甚至企事业单位内部通讯录等，全部遭到泄露。因此，目前云存储技术中亟需关键技术来解决个人隐私泄露问题。

目前，市面上主流的云盘软件仅考虑了文件传输时的秘密性与完整性，而没有运用加密技术保护存储文件。例如百度网盘，腾讯微云等产品在传输时采用合法用户校验、通讯隧道加密等方式来保证传输安全，但在数据存储时仅使用了访问控制技术管理明文存储的用户数据，一旦攻击者破解或越过数据库的访问控制，云盘中以明文存储的文件将会直接暴露，导致大量明文数据泄密，这将极大地损害用户的合法权益和隐私安全。

为了减少数据明文直接暴露的风险，目前研究者已开始数据库加密的系统性应用研究。详细来说，目前市场上已有BitGlass, CipherCloud, EnclaveDB[11][18]等加密数据库产品实现了以密文形式存储云盘数据。为了实现传统数据库的文件取回与分享功能，加密数据库中采用了可搜索加密技术（Searchable Encryption, SE）[12]。具体而言，用户可以利用可搜索加密来按照关键词的形式访问和分享外包的云盘文件。这种可让用户安全访问与分享文件的技术已经成为加密数据库库研究的关键技术之一[2]。此外，可搜索加密技术具有前向安全性[3] 和后向安全性[4]的严格安全性定义，使得其在执行更新和删除操作时减少信息泄露风险。

然而，可搜索加密技术使用的密文索引不具有传统数据库索引易于访问的性质[13] [14]，这使得可搜索加密访问密文索引需要巨大的性能开销，且传统搜索算法无法解决。文献[15]指出，该操作可以看作一种无序数据库搜索问题（UDS，Unsorted Database Search），即对未排序的数据库执行搜索操作。该问题在经典的计算机通常采用随机读取与遍历查找的策略，但是这些策略都需要的时间复杂度才能完成，这将在大规模请求时产生显著的时间开销。例如，Google每天需要处理超过35亿次搜索请求，平均每秒超过4万次请求，每次请求均需要遍历整个数据库（通常是密态数据库）才能得到结果。这些事实说明了降低密文索引访问开销是有待研究的关键问题。

随着量子计算产业的发展，量子搜索算法为解决传统搜索算法中索引访问效率的问题提供了新的解决思路。量子搜索算法利用了量子叠加和纠缠的特性，能够同时处理多个搜索空间中的可能解，从而提高搜索效率。其中Grover算法能够在无序数据库中快速以找到目标项，实现指数级别的搜索加速。此外，Grover算法还具有一定的量子计算抵抗性，能够保证搜索过程的量子安全性。因此，如果能够实现一套可搜索的云盘系统，以密文形式存储文件，使用Grover量子搜索算法加速密态数据库检索效率，则可以在保证了系统易用性的同时大幅提升系统的运行效率，增强安全性和可靠性，可以满足更高的市场需求和保密需要。在量子计算硬件成熟后，量子计算加速文件检索带来的多项式级时间优化可以使得云盘存储的信息量大幅提升，同时仍能保持较快的查询速度和极高的准确性，相对于现代云盘产品更有优势。

针对上述设想，本文提出了一个基于量子比特加速检索算法实现的密态云盘系统，进一步探索科大国盾量子技术股份有限公司“祖冲之二号”等量子计算云平台上的应用场景，设计实现可搜索加密模型KS2E，设计实现可扩展的量子搜索方案。此外，本系统具有高可用性，可在不同的云平台如量子计算云平台，AmazonCLoud上部署。

## 1.2 研究现状

一般的搜索问题场景为关键词搜索，在可搜索对称加密技术的研究中，关键词搜索的动态性问题已经得到了充足研究：2000年，Song等人提出了可搜索加密（SSE）的概念[23]，以便用户以安全的方式搜索存储在云中的加密数据。2012年，Kamara等人[24]首次提出了动态可搜索对称加密（DSSE）的概念，并提出了数据库更新协议和SSE的适应性安全模型。为深入DSSE的更新需求，Stefanov等人[3]在2014年提出了DSSE的前后向安全的概念。其中，前向安全指的在DSSE 方案中，未来的更新请求与以前的搜索请求不会产生任何联系；相对的，后向安全指的在DSSE 方案中，未来的更新请求与以后的搜索请求不会产生任何联系。在此概念基础上，可搜索对称加密技术不断完善。

支持搜索数据分享的可搜索加密同样是一个关键问题。密文数据分享的可搜索对称加密来源于多密钥可搜索加密的概念。2013年，Popa等人[25]提出了多密钥可搜索加密（MKSE）的概念，并在2014年基于MKSE构建了系统Mylar[26]。但此方案仍存在攻击缺陷[27]，其安全性模型和实例化方案仍需改进。此后，Hamlin等人[28]提出了MKSE的模拟安全性模型与实例化方案，Wang等人[29]在2022年提出了组合可搜索加密（HSE）的方案，方案结合了公钥加密和DSSE实现了多写者场景下的数据分享方案。在公钥领域也存在利用公钥性质实现的搜索数据分享的方案，例如Cui等人[30]提出的密钥聚合的可搜索加密技术（KASE），张等人[31]提出的云环境下的密钥聚合可搜索加密方案，甘等人[32]提出的高效密钥聚合方案。可搜索加密技术已具备较为完善的应用条件。

可搜索加密技术可以应用在秦盾云QinCloudDB，MongoDB等加密数据库中，直接对加密数据执行查询操作。数据库的可搜索加密技术可以直接加密数据，提供数据级安全保障，搜索功能灵活可靠，适配加密数据库多变需求应用。目前成熟的可搜索加密技术分为可搜索公钥加密技术和可搜索对称加密技术。传统加密数据库采用的是可搜索公钥加密技术，通过公钥对上传的文件进行加密，用户只需要私钥即可解密文件[16] [17]。这种加密技术虽然便于用户之间文件的传输与转存场景，但是由于公钥加密过程计算复杂，大数加乘法多，加密效率低；可搜索对称加密(Symmetric searchable encryption, SSE) [13]由于采用位运算，加密效率高，适合大规模数据场景，但是该技术下云盘文件的加解密需要文件所有者参与，文件的传输与分享不便。

相比于传统搜索算法，量子搜索算法在并行性，安全性，性能开销等方面更具优势，逐渐得到学术界及工业界的关注。Bell labs 在1996年提出的Grover作为一种在量子计算机上运行的非结构化搜索算法，其总体框架为构造一个易于实现的操作，使得初始态可以在该操作的迭代下逐渐靠近目标态，然后通过测量目标态得到搜索结果。它的复杂度为，相较于经典搜索具有指数量级的加速。Grover[6]算法使用量子硬件成功地将经典计算机检查搜索空间的次数大幅降低。目前学术界针对Grover算法进行了不同程度的优化，Seidel[20]等人提出了一种自动生成Oracle门电路的设计方法，将非结构数据库通过哈希的方式进行编码（Data Encoder），然后通过编码构造酉矩阵生成Oracle电路。Vemula[19]等人通过设计CZ门，X门等基本门电路，扩展了Grover算法搜索的量子比特数。

然而，其Grover算法的量子门电路结构特别是Oracle门电路构造理论尚不完善；随着输入量子比特数的增加，需使用的量子门电路数量及其构造难度呈指数级上升，门电路运行时间超过量子活跃时间；且该算法单次只能对固定搜索问题进行搜索，可扩展性较差。

本次赛事使用目前世界顶尖的“祖冲之2 号”级66比特可编程超导量子计算机[21]和量子计算云平台对量子计算软件开发和应用进行进一步的探索，有望进一步提升我们在量子计算语言、算法上的水平，借助量子计算技术在重点行业、新兴领域取得突破性进展。通过isQ-core语言的自定义酉门操作借助矩阵运算实现了Oracle门，并针对上文提及的Grover算法的缺陷进行了改进，合理设计量子门电路，初步实现了可扩展的Grover算法，可以在同一电路中对多组输入进行搜索。以此搜索算法为基础，本文实现了一个可以对实体数据库进行查询的量子比特加速检索算法，在保证Grover算法的时间复杂度的基础上扩展了其功能，一定程度上弥补了Grover算法的不足，满足了其实际应用的需要。作为量子检索领域少有的能够在实际使用的数据库应用中实现Grover算法功能的算法设计，本作品适合进一步进行开发拓展，应用潜力巨大。

# 系统设计

## 2.1 系统概述

整个系统是基于量子计算与C/S架构进行实现的。

量子搜索密态云盘系统是利用量子计算机进行搜索加速、以密文形式存储用户数据的在线存储系统。系统将用户数据加密后存储至数据库，保护用户信息安全，并能实现一般云盘的文件分享等过程，且具有比常规云盘更快的搜索速度。

量子搜索密态云盘系统主要分为服务器与客户端两部分，基于量子计算与C/S架构进行搭建。

服务器使用可搜索加密模型KS2E和基于Grover的可扩展量子搜索技术，用于接收客户端发送的请求报文，在处理请求报文后发送对应的响应报文给客户端；主要负责处理用户注册处理、用户登录验证以及文件分享、搜索等请求。 当对可搜索密文进行搜索时，服务器根据搜索密文和关键字构建映射关系矩阵。在接收客户端的搜索令牌后，根据令牌及文件id搜索关系的映射矩阵生成量子门电路，借此来进行搜索。

客户端通过视图给用户提供交互接口，将用户输入的信息以及操作信息进行处理后将密文搜索所需参数封装在请求报文中发送给服务器，并且在解析服务器返回的响应报文后，使用户能够看到对应的处理结果。

整个系统使用Go语言以及gin框架完成服务器开发；使用MySQL搭建关系型数据库，并且使用gorm框架对数据库进行管理；使用WebStorm作为开发工具，借助Vue框架完成Web端页面的开发；将Android Studio作为开发工具，使用Java语言来进行Android应用开发；并将量子计算云平台用于加速使用搜索令牌对可搜索密文进行搜索。

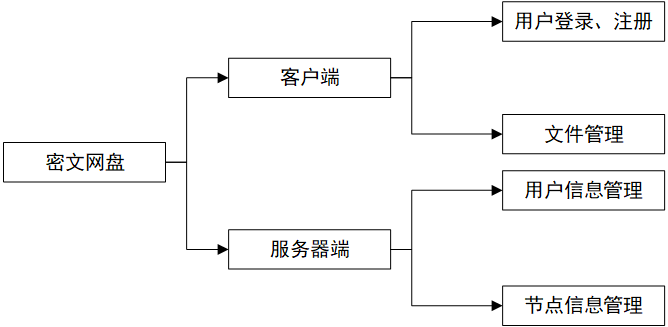


图 2‑1 系统功能框架

## 2.2 系统功能设计

1. 注册、登录和退出功能：数据库保存用户相关信息，登录成功后生成身份凭证，用于后续操作的身份验证；退出后清除该凭证。
2. 用户信息展示功能：通过前端控件展示用户个人基本信息。
3. 修改密码功能：改变用户登录密码。
4. 文件列表展示功能：根据目录id从服务器取回该目录下的文件列表信息，并通过前端控件展示给用户。
5. 文件内容浏览功能：根据文件id从服务器取回文件加密内容，解密后通过前端控件展示给用户。
6. 文件上传功能：从手机内存中选择要上传的文件，读取文件内容，提取关键字，生成文件id，并根据关键字和id生成搜索索引，将文件内容加密后同搜索索引一起上传至服务器，保存到数据库中。
7. 创建文件夹功能：在当前目录下创建一个由用户命名的空文件夹。
8. 文件搜索功能：根据搜索关键字生成搜索令牌，根据令牌从服务器取回包含该搜索关键字的文件，并通过前端控件展示给用户。
9. 文件转存接收功能：通过蓝牙进行用户面对面密文传输，接收者根据密文关键字生成新的搜索索引，保存到自己的数据库中。
10. 文件删除功能：从当前目录下删除对应文件。如果是文件夹，则批量删除文件夹包含的所有文件。

## 2.3 模型构建

根据功能模块划分的系统模型结构如图 2‑2所示。当当用户在视图模块进行操作或输入信息时，将会将其作为输入传输给与Web端组件或Android应用控件绑定的数据处理模块。数据处理模块将会使用对应的处理函数对输入进行处理，并会将处理结果封装在请求报文，再通过网络通讯模块发送给服务器接口模块。服务器接口模块将会对于请求报文进行解析与验证，并且对于不同请求路径上传的报文，将会使用不同的方法处理。除去直接封装在Go项目中的处理逻辑，在服务器进行处理请求报文时，可能还需要调用量子搜索模块以及数据库存储模块。量子搜索模块借助从请求报文中解析得到的搜索令牌，对可搜索密文进行查找。数据库存储模块主要是用于存储用户信息及节点信息。当服务器接口处理结束后，将会生成对应的响应报文发送给客户端。客户端对响应报文进行解析后，将会通过视图模块展示处理结果。



图 2‑2 系统的模型结构图

## 2.4 接口设计

（1）注册接口：客户端在注册时需要传输的请求参数如图 2‑3所示，其中username是用户名；hashId是用户身份验证信息；email是用户邮箱；biIndex是双向索引结构；key1、key2是随机生成的用户密钥。

（2）登录接口：客户端在登录时需要传输的请求参数和服务器返回的响应参数如图 2‑3所示，其中token是用户登录成功后返回的凭证；userInfo是当前登录者的用户信息；结构体UserInfo中的rootId是服务器随机生成的且不重复的当前用户节点id。

（3）发送搜索令牌接口：客户端在执行搜索操作时，首先调用该接口，其请求参数和响应参数如图 2‑4所示，其中repeated表示该项是一个集合属性。

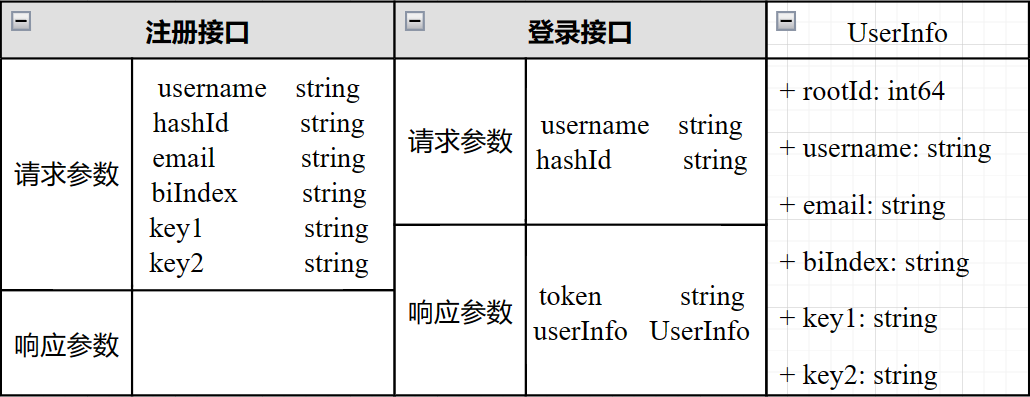


图 2‑3 登录、注册接口与UserInfo结构定义

（4）搜索接口：客户端解密（3）的返回结果，得到相应文件id集合，然后作为请求参数发送给服务器，服务器再把对应节点信息发回给客户端。Node结构中的nodeType指节点类型，是一个枚举类型；nodeId是服务器生成的节点id；nodeName是节点名称；nodeContent是针对文件节点而言，存储的是加密后的文件内容；secretKey也是针对文件节点而言，存储的是加密文件内容的密钥，它同样被加密存储；createTime和updateTime是该节点被创建时间和最近修改时间。

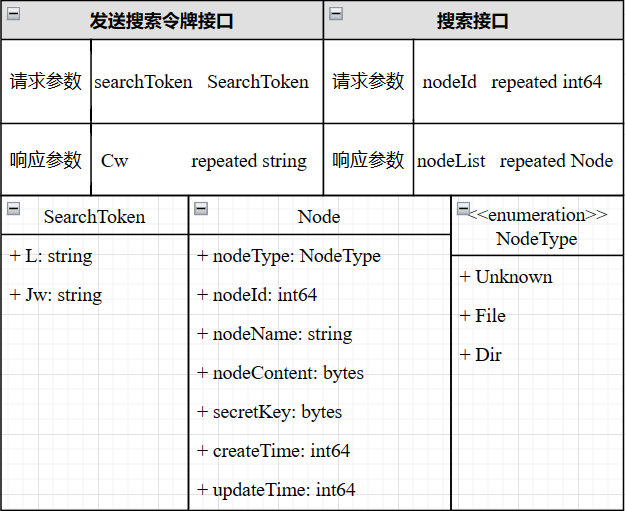


图 2‑4 发送搜索令牌、搜索接口

（5）获取文件列表接口：请求参数和响应参数如图 2‑5所示。客户端在拿到指定节点的所有子节点id后，可以调用（4）获得所有子节点的信息。

（6）获取文件内容接口：请求参数和响应参数如图 2‑5所示。考虑到在（4）中如果获取所有文件内容和对应密钥将给服务器带来巨大传输负载，且部分文件内容不会被使用，造成信道资源浪费，因此单独开辟一个接口，获取指定文件id的文件内容和对应密钥。

（7）创建文件夹接口：请求参数如图 2‑5所示。其中dirName是文件夹名称，parentId是创建文件夹时所在层级的节点id。

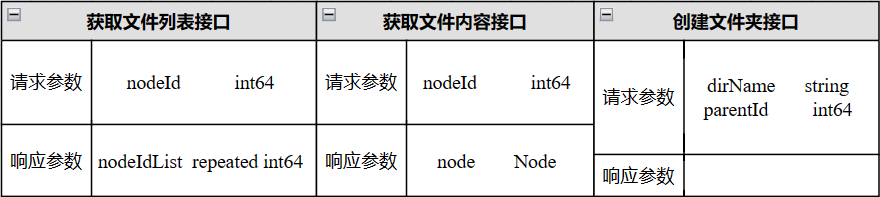


图 2‑5 获取文件列表、文件内容、创建文件夹接口

（8）获取节点id接口：在上传文件时需要利用（关键字w，文件id）生成搜索索引，所以需要先向服务器发送请求，获取由服务器生成的节点id。

（9）上传文件接口：请求参数如图 2‑6所示。其中fileName为文件名称；parentId为上传文件时所在层级的节点id；indexList是由（8）得到的节点id和文件内容关键字通过Update函数生成的搜索索引；content是加密后的文件内容；biIndex是更新后的双向索引结构；nodeId是（8）得到的节点id；fileSecret是随机生成的文件密钥，被加密后再上传。

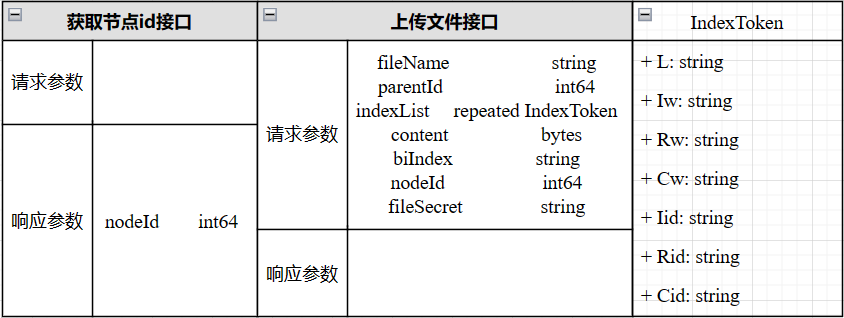


图 2‑6 获取节点id、上传文件接口

（10）一轮分享接口：请求参数和响应参数如图 2‑7所示。其中S是搜索到的Cid集合。

（11）二轮分享接口：请求参数和响应参数如图 2‑7所示。其中isShare指当前文件是否为分享文件；address表示分享文件的内容存放地址。

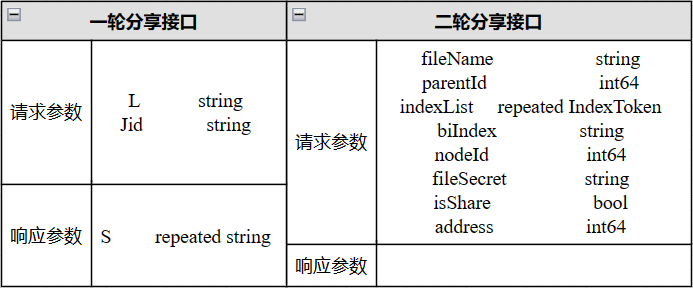


图 2‑7 一轮、二轮分享接口

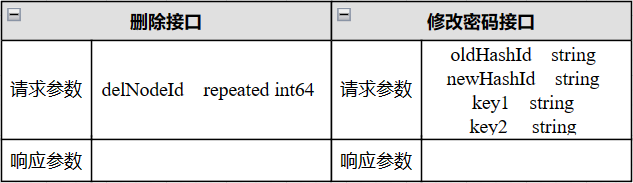


图 2‑8 删除、修改密码接口

（12）删除接口：请求参数如图 2‑8所示。其中delNodeId是要删除的节点id集合。

（13）修改密码接口：请求参数如图 2‑8所示。其中oldHashId是旧密码生成的身份验证信息；newHashId是新密码生成的身份验证信息；key1、key2是用新密码加密后的用户密钥。

## 2.5 小结

本章通过合理的功能设计、模型构建和结构设计，逐步完成了密态云盘系统的理论构建。本部分从密态云盘的基本概念出发，开展了模块化的功能设计，将系统划分为功能模块、确定了基本功能和服务器端与用户端的开发方式、开发工具，并对设计的功能所需要用到的接口的参数名称和类型进行了详细描述，方便进一步开展算法设计。

# 算法设计

## 3.1 可搜索与分享的加密模型KS2E

可搜索与分享的加密模型（Keyword Search Shareable Encryption, KS2E）是一种能同时搜索与分享可搜索密文的方案模型。该模型由Setup、Update、Sharetoken、Share和Search五个函数组成，分别实现了系统建立、密文更新、密文分享授权生成、密文分享、密文搜索的功能。为了实现密文搜索和密文分享，描述了一个双向索引结构来存储搜索索引，如图 3‑1所示，即每条横链具有相同的文件，哈希表指向该横链上的最后一个关键字，每次插入时都将新的关键字放在链尾，并更新，在进行分享时，Share函数便通过横链找到分享文件的所有关键字；同理，每条纵链具有相同的关键字，哈希表指向该纵链上的最后一个文件，每次插入时都将新的文件放在链尾，并更新；在进行搜索时，Search函数便通过纵链找到包含搜索关键字的所有文件。

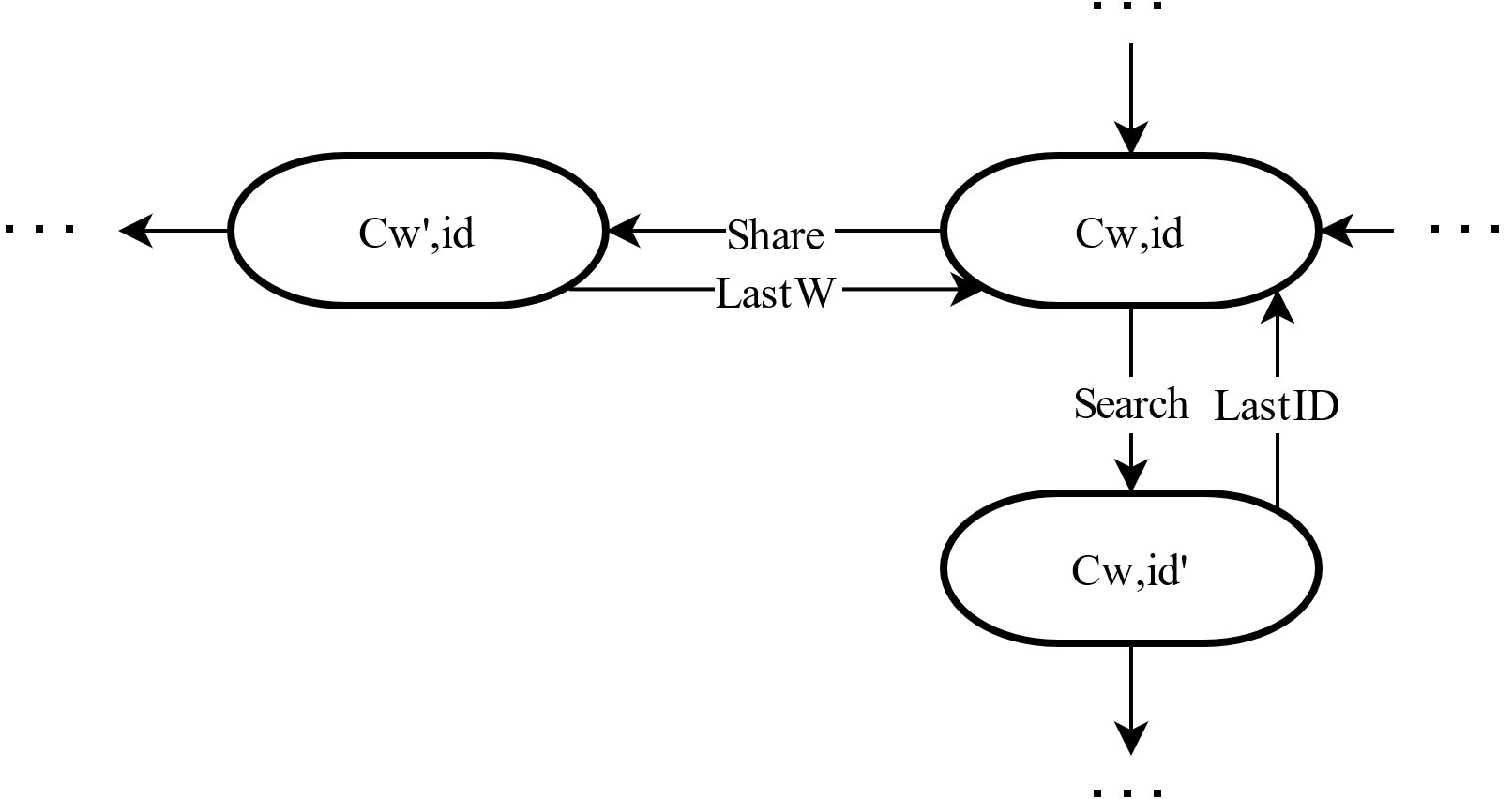


图 3‑1 双向结构索引

五个函数具体工作如下：

（1）Setup函数进行初始化工作。包括初始化一个哈希函数，该函数输入两个长度为的比特串，输出一个长度为2的比特串；初始化一个伪随机函数，该函数输入一个长的比特串和一个任意长的比特串，输出一个长的比特串；初始化一个对称加密函数，该函数包含一个加密算法和一个解密算法；为用户随机生成比特的密钥、；初始化哈希表和。

（2）Update函数利用（关键字，文件）生成搜索索引上传到服务器。搜索索引具体生成过程如下：首先生成索引标识符，其中“||”符号表示比特串拼接。然后生成两个比特的随机数和，利用用户密钥得到，。利用找到链尾，如果是空，，如果

不是空，。利用找到链尾，如果为空，，如果不为空，。最后得到搜索索引。

（3）Sharetoken函数利用要分享的文件生成分享令牌发送给接收者。分享令牌具体生成过程如下：根据找到链尾，如果为空，说明不存在对应文件，直接返回；否则，, ，，得到分享令牌。

（4）Share函数包含客户端与服务器之间的两次交互过程，第一次交互客户端根据分享令牌从服务器获取对应文件的关键字；第二次交互客户端根据接收者的密钥利用（关键字，文件）生成新的搜索索引上传到服务器。具体过程如下：服务器收到令牌，声明一个空集，当不为空时，执行循环操作：根据从数据库中找到，将放入集合中，根据横链向前迭代得到，更新。循环结束后将集合返回给用户，用户解密即可得到分享文件中的关键字，用户根据（2）中的Update函数将这些关键字利用自己的密钥重新生成搜索索引上传。

（5）Search函数根据关键字生成搜索令牌，然后用搜索令牌从服务器获取包含该关键字的文件。具体过程如下：根据找到链尾，如果为空，说明不存在包含该关键字的文件，直接返回空；否则，，将和作为搜索令牌发送给服务器。服务器声明一个空集，执行循环操作：根据从数据库中找到，将放入集合中，根据纵链向前迭代得到，更新：。循环结束后将集合返回给用户。用户解密即可得到包含关键字的文件。

## 3.2 基于Grover算法的可扩展量子对称加密加速搜索方案

### 3.2.1 Grover算法

在传统计算机的搜索算法中，对非结构化的数据进行搜索需要对每个数据进行比对。在最坏情况下，搜索时间复杂度可达，而利用量子计算的Grover算法搜索的时间复杂度为。在数据规模达到一定程度时，Grover算法可以极大地体现出量子搜索算法的优越性。

Grover算法的步骤大致分为四步：

1. 量子初态制备。
2. Oracle电路以翻转相位的形式标记目标态。

（3） 对称翻转操作使目标态幅值放大。

（4） 测量量子态以得到目标值。

Grover算法模型假设所有数据存在数据库中，假设有个量子比特，用于记录数据库中的每一个数据索引，一共表示个数据，记为个，希望搜索的数据为个，建立一个函数：

其中为搜索目标索引值。将以一个酉矩阵的形式给出，用以将转化为：

的作用是通过改变目标态相位，标记搜索问题的解。

经过后，再经过三个量子门变换放大目标态幅值：

1. 经过一个 Hardmard 门；

（2）对量子态进行一个相位变换，将 态的系数保持不变，将其他的量子态的系数增加一个负号，相当于 酉变换算子；

（3）再经过一个 Hardmard 门。

这一部分操作的数学表述为:

经过多次由以上操作组成的Grover迭代，可将量子态转换为目标态。其中迭代次数，然后对末态进行测量即可得到目标结果。

### 3.2.2 可扩展的量子搜索算法

现假设有一长度为的数组，其存储数据集合记为，下标集合记为。集合和等于同一集合，以保证数据和下标能通过相同数量的量子比特表示，输入某一数据，得到其对应下标。然后取的情况具体讨论。

为了区分不同的输入，先根据输入制备不同初态。不同数据制备的初态在量子电路上表现为对应下门的有无。如下公式所示：

假设现数组为顺序数组，设Oracle电路对应矩阵为，则初态经过Oracle的作用可表示为：

可解得，可验证其满足酉矩阵条件。

注意此处初态与Oracle矩阵相乘后的中间态只关注正负相位而不关注幅值大小，为了便于此矩阵运算计算，规定所有幅值为1。当希望更改数组顺序时，可将该矩阵的对应行互换，则可改变初态对应的转换结果。

由此可得任意长度为2的乱序数组搜索所需的Oracle矩阵，共24种。由该2量子比特Oracle矩阵表示的量子电路，均可由门，门，门组合表示。

最后将中间态经过后三步量子门操作，即可得数据所对应下标。

若需扩展搜索数据比特位数，除扩展矩阵大小外，可以将大的二进制数以两比特（或其他较小的比特位数）为一组，存放于单个数组中。搜索时以两比特为一组逐个输入。数据存放个数仍受限于单元矩阵大小，搜索数据时，当且仅当搜索下标全部一致时，数据有效；否则数据无效。

通过以上方式，可得任意比特长度搜索所需的Oracle电路对应矩阵，Oracle电路对应矩阵呈现以下规律：Oracle矩阵的第行，等于将数组中第个数据的二进制量子态依次经过门后的量子态对应矩阵除第一个幅值外的值取反后的转置除以，公式表示如下：

设

得到Oracle电路对应的矩阵后，通过isq中自定义酉矩阵的方式，定义，将Oracle电路插入整个算法中。

此数组的存数操作与矩阵行交换等价，且只能对双射关系进行搜索。在存储多位数据时，可将数据分为多组，使用多个矩阵存储进行表示。且对Oracle电路对应矩阵进行转置后，可用于数组下标取数操作。

### 3.2.3 量子搜索功能设计

使用Grover算法实现无序数据的存储与搜索的大致流程如下：用户在数据存储时上传文件，系统发出存储请求，根据用户的数据密文生成可搜索密文、数据库索引及加密文件id。通过构造数据库索引和可搜索密文搜索关系的映射矩阵生成量子门电路（如图 3‑2所示）。

当用户需要搜索某关键字时，首先客户端根据关键字生成数据库索引和前一可搜索密文的，需要通过量子门电路根据其数据库索引搜索其，用于下一含有关键字的文件搜索，同时在数据库中匹配文件id，接着结合加密id与密钥（网盘部分实现）获得文件的明文。



图 3‑2 量子搜索算法门电路设计图

图 3‑2的第一部分为数据平权与量子叠加。由于Grover算法中需要叠加量子实现翻转，反演等快速搜索策略，需要得到平权的量子态。用户在输入关键字（keyword）之后，将会被处理成数据库索引，再转换成二进制bit作为量子电路的输入。所有量子比特位（简称为qubit）的初态均为0，根据输入的不同，在量子电路中添加合适的量子非门（简称为X门），使得经过X门后qubits的位值与输入一致。随后，所有qubit经过Hadamard门（简称为H门）进行量子叠加，形成平权的叠加量子态，总态数为（n为使用的量子比特位数）。将不同输入经过H门后形成的叠加态存储到一个的矩阵中。

图 3‑2的第二部分为Oracle组合门的设计。Oracle组合门的目的是将目标量子位的相位进行反转，为了实现无序数据对有序量子态的映射，我们采用更新矩阵的方式，将数据库索引抽象为无序数据，可搜索密文抽象为其索引，即当输入为时，其目标结果应该为，将对应的量子态进行相位翻转。从实现层面，更新后的矩阵由以下公式得到：

图 3‑2的第三部分为Diffusion组合门的设计。Diffusion的目标为放大目标结果（负相位）的概率，在实现上仅需使用X门，H门与Toffili门（此处指CCZ门）。由于CZ门与CCZ门分别对应2qubits与3qubits，不符合本系统中可扩展的量子位（比特数高于4）需求。因此，团队通过自定义矩阵的方式实现了任意比特位的Multi Control Z门（简称MCZ门），该矩阵为对角矩阵，对角线上末两行值为-1，其余行对角值为1。

## 3.3 小结

为了解决可搜索对称加密技术不便于文件的分享与转存，可搜索公钥加密技术加密效率低的问题，本章设计实现了可搜索加密模型模型KS2E，并且利用Grover算法加速对称加密搜索。在传统的Grover算法的基础上本章进一步提出了可扩展的量子搜索算法，从数学原理与电路设计两方面论述了该方案的可行性，并且设计了可扩展Grover算法完成无序数据存储与搜索的具体流程。

# 系统实现

## 4.1 系统开发环境

整个系统主要由密态云盘应用组成，并通过量子计算加速了使用令牌查找可搜索密文。系统开发环境主要分为密态云盘开发与量子计算实现两部分。具体系统开发环境见表 4‑1。

表 4‑1 系统开发环境

|  |  |  |
| --- | --- | --- |
| 开发类型 | 相关工具 | 具体信息 |
| 密态云盘开发 | 操作系统 | Windows 10 |
| 架构 | C/S |
| 服务器开发语言 | Go |
| 服务器开发框架 | Gin |
| 数据库 | Mysql |
| 数据库管理框架 | Gorm |
| Web端开发工具 | Webstorm |
| Web端开发框架 | Vue |
| 页面UI框架 | Element-UI |
| Android开发工具 | Android Studio |
| Android开发语言 | Java |
| 量子平台开发 | 实验平台 | 量子计算云平台 |
| 编程语言 | Python、isq |
| 编程SDK | ezQpy |

## 4.2 系统项目搭建

### 4.2.1 服务端设计实现

服务器的主要功能就是处理客户端发送的请求，并返回正确的处理结果。在服务器中请求路径的定义以及使用场景的说明见表 4‑2。整个服务器的主体就是借助各种工具，例如MySQL数据库，Redis数据库等，来为处理请求服务。

表 4‑2 请求路径url说明

|  |  |  |  |
| --- | --- | --- | --- |
| 请求路径 | 具体使用场景 | 请求路径 | 具体使用场景 |
| /user/login | 用户登录 | /file/sendSearchToken | 搜索文件 |
| /user/register | 用户注册 | /user/changePwd | 修改用户密码 |
| /dir/create | 创建文件夹 | /file/uploadShareToken | 上传分享令牌 |
| /file/upload | 上传文件 | /file/getShareTokens | 获取分享令牌 |
| /file/search | 寻找文件节点信息 | /file/shareFirst | 接收分享文件第一轮交互 |
| /dir/get | 获取文件夹信息 | /file/shareSecond | 接收分享文件第二轮交互 |
| /node/get | 获取文件节点内容 | /node/delete | 删除文件 |

在服务器运行中，使用基于内存的Redis数据库存储用户登录随机生成的身份令牌，使用MySQL数据库存储用户信息以及文件信息。整个MySQL数据库包括六张表，具体信息如图 4‑1所示。share\_token表用于暂存文件分享令牌；file\_index表用于存储文件可搜索密文；user\_info表用于存储用户信息和用户登录验证信息；node\_rel表用于构建文件树；file\_info表用于存储文件信息；node表用于存储节点信息。



图 4‑1 数据库设计概念图

### 4.2.2 客户端设计实现

1. Web端设计

Web端的页面设计如图 4‑2所示。整个Web端网页主要是由App组件构成。这个组件中使用了Header视图组件以及使用Vue-router技术联系起来的三个视图组件。Router.js就是使用Vue-router技术时编写的配置文件，包含了三个视图组件的访问方式以及访问时传递的参数。这些视图组件包含了ElementUI提供的交互组件。

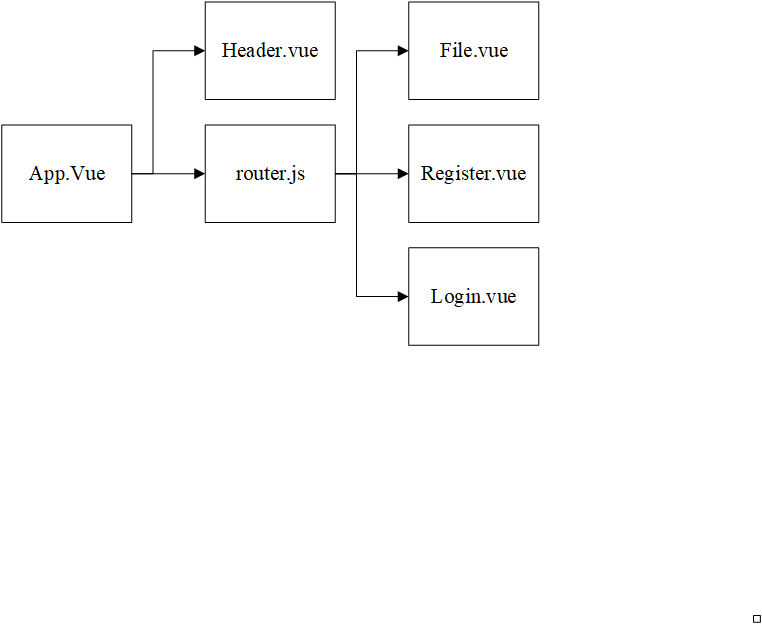


图 4‑2 Web端页面设计图

当检测到用户登录成功时，Header.vue中提供修改密码、退出登录等操作交互。当用户没有成功登录时，Header.vue将提供登录页面与注册页面自由跳转。File.vue包含所有文件管理的交互组件，包括上传文件、创建文件夹等。Register.vue提供了信息输入框，能够检查用户输入的注册信息格式。Login.vue同样提供了信息输入框，能够检查用户登录信息的格式。通过为交互组件绑定事件的方式来接收用户在页面进行的操作以及输入的信息，然后通过处理这些操作或信息，使用户对文件进行操作或查看相关信息。在处理过程中，事件处理函数将会调用全局定义的函数。在整个Web项目中重要文件的介绍可见表 4‑3。

表 4‑3 Vue文件说明表

|  |  |  |
| --- | --- | --- |
| 文件夹名称 | 子文件 | 说明 |
| global | globalConst.js | 包含对于全局常量的定义 |
| pb\_gen | common\_pb.js | 包含其他pb文件需要使用的数据类型 |
| upload\_file\_pb.js | 包含上传文件的通讯信息 |
| user\_login\_pb.js | 包含用户登录的通讯信息 |
| user\_register.js | 包含用于注册的通讯信息 |
| request | file.js | 在文件管理时与服务器通讯 |
| http.js | 封装基于https通讯的函数 |
| user.js | 在用户管理时与服务器通讯 |
| store | index.js | Vuex主模块，定义了全局变量 |
| module | 包含多个Vuex子模块文件，定义了不同模块  需要使用的全局变量 |
| utils | ByteUtils.js | 包含处理字节的函数 |
| EncryptUtils.js | 包含加解密的函数 |
| FileUtils.js | 包含随机生成文件的函数 |
| JSON.js | 包含json字符串转换的函数 |
| StringUtils.js | 包含字节数组与字符串之间的处理函数 |

1. 安卓端设计

采用Android Studio工具进行软件开发，采用前后端代码分离方式，以xml格式文件进行页面布局，以java格式文件进行后台逻辑编写。在前端页面的设计上，把简洁、易操作定为设计目标，突出软件的具体功能；采用市面上手机软件常用的底部导航栏形式，分别实现对用户文件和用户信息的管理。在后台逻辑上，实时渲染前端页面，并为前端控件设计点击事件，响应用户请求；采用protobuf技术实现与服务器对应的通信接口，用okhttp3技术实现与服务器的网络连接，并编写对应封装报文和解析报文的逻辑代码；采用蓝牙技术实现用户面对面数据传输。

### 4.2.3 密态云盘部署

1. Web端页面部署

Web页面是在WebStore中基于Vue框架进行实现。具体的部署方法如下：

* 1. 从node.js官网下载版本号为8.x.x、10.x.x、12.x.x、14.x.x、16.x.x、17.x.x之一且合适个人电脑的版本进行安装。注意选择合适的安装地址。
  2. 键盘敲击Win+R，输入cmd打开命令提示符，可以通过“**npm config get cache**”和“**npm config get prefix**”查看cache以及prefix的路径。为了数据管理，可以选择自己新建文件夹，并通过“**npm config set cache ‘xxx’**”和“**npm config set prefix “xxx”**指令进行修改。但不建议将cache以及prefix放在需要管理员权限的目录下，否则影响之后的安装。
  3. 然后在node.js的安装路径下找到node\_modules文件夹，并在用户环境变量中设置新的NODE\_PATH变量，值为node\_modules文件夹路径。将cache和prefix文件夹的路径加入到系统变量PATH中，若选择自定义两个文件的路径，则将自定义的路径添加进去。
  4. 通过分别执行“**npm install -g cnpm --registry=https://registry.npm.taobao.org**”和“**npm config set registry https://registry.npm.taobao.org --global**”配置淘宝镜像，加快第三方库的下载引入。
  5. 使用“**npm install -g @vue/cli**”指令安装Vue以及Vue脚手架。
  6. 下载WebStorm，并打开Web端网页源码
  7. 在WebStorm的终端中可以安装yarn工具，之后WebStorm将会自动将所有需要的第三方库下载进来。
  8. 在package.json文件中点击运行serve即可。

1. MySQL数据库部署
   1. 从MySQL官网找到想要下载的版本进行下载，并将解析的文件夹放置到合适的位置。文件夹路径尽可能不要包含中文。
   2. 在解压的文件夹中新建一个my.ini文件，这个文件在Go服务器的源码中存在模板。可以将其复制过来后修改其中basedir以及datadir。其中basedir设置为安装目录，datadir设置为数据库数据data存放位置。注意服务器会自动在datadir路径下新建data文件夹，不需要提前创建。
   3. 在系统环境变量PATH中添加解压文件夹中bin文件夹的路径。
   4. 以管理员身份打开命令提示符，使用“**mysqld --initialize –console**”指令进行初始化，并得到初始密码；使用“**mysqld --install mysql**”指令安装数据库；使用“**net start mysql**”启动数据库服务；然后使用“**mysql -u root -p**”并输入初始密码可以进入数据库；使用“**alter user 'root'@localhost identified by '123456';**”指令可以将数据库登录密码修改为123456。
   5. 此时可以通过账户名“root”以及密码“123456”正常访问MySQL数据库。
   6. 在安装好的Datagrip中，通过账户名和密码连接到本地数据库，进行可视化的管理。从Go项目的ddl.sql文件中复制数据库的定义语言,在新建的console中粘贴并全选执行就可以搭建服务器所需要的Mysql数据库。
2. Go服务器部署
   1. 首先下载合适的Golang的msi文件（可以前往Golang官网或国内网站Golang中文网下载）。
   2. 将GOROOT与GOPATH添加到系统环境变量中，其中GOROOT就是msi的安装地址，而GOPATH是一个自定义的目录。GOPATH中包含三个文件夹，分别为bin、src、pkg。bin将会保存使用go build指令得到的程序。src用于存放Golang源码。pkg中将会存储项目中所有使用过的第三方库。
   3. 下载GoLand软件，将Go服务器项目源码放置在src目录下并打开。
   4. 在settings中设置GOPATH路径，全局路径与系统路径有一个即可。同时settings中添加GOPROXY=https://goproxy.cn可以修改第三方库的下载服务器。然后可以关闭Goland重新打开，程序将会自动下载第三方库。
   5. 安装Redis数据库。由于Redis官方只提供了Linux版本，如果是Windows用户，则需要自己从github上下载exe文件，然后可以运行exe文件直接进行快捷安装。Linux用户需要通过命令行进行安装。
   6. 在MySQL数据库以及Redis数据库正常运行，找到main.go文件夹，点击入口函数main左侧的绿色小三角就可以运行。
3. 安卓端部署
   1. 从https://github.com/guoxingyu-del/Graduation-project.git将项目拉取到本地，然后在Android Studio中打开该项目，使用USB插口连接手机，并选择连接设备，运行程序，手机上会出现下载提示，按照提示下载该应用便可以使用该网盘软件了。
4. 项目配置
   1. Vue项目配置https。在WebStorm中的终端执行“**npm install -g mkcert**”指令安装mkcert。继续执行“**mkcert create-ca**”指令生成ca.crt与ca.key。执行“**mkcert create-cert**”指令生成cert.crt与cert.key。然后再双击ca.crt将证书安装到本地。如果双击ca.crt没有弹出直接安装，则可以直接进行系统设置，再将证书安装后，在vue.config.js中可以设置cert.crt与cert.key的路径方便读取并使用。具体的设置如图 4‑3。

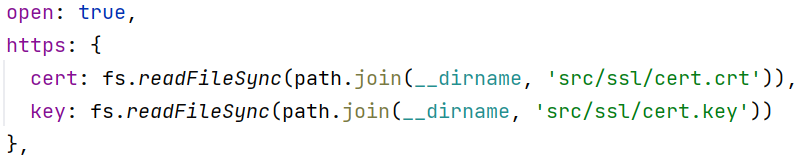


图 4‑3 https配置图

* 1. 服务器配置存储服务：除了之前提到的MySQL数据库与Redis数据库外，服务器还使用了COS对象存储来存储文件。图 4‑4中已经给出了相关存储服务的配置。针对不同的服务器，可能需要进行修改。如果没有在腾讯云上申请COS对象存储服务，可以保留对应的配置进行操作体验。



图 4‑4 存储服务配置图

* 1. 路由配置：在Vue项目的vue.config.js文件中包含了对服务器ip地址的配置。在Golang项目的server.go文件中可以设置服务器想要监听的端口。

### 4.2.4 量子平台连接

本项目使用量子计算云平台中与“祖冲之二号”同等规模的超导量子平台ClosedBetaQC进行计算。在完成上述密态云盘部署后，用户可利用自己的量子计算云平台账号，使用用户SDK密钥与即将使用的计算机名称登录量子计算云平台，编译完成的isQ代码，上传至平台进行计算，循环查询直至量子计算机运行结束后将计算结果传回本地。

## 4.3 系统操作流程

如图 4‑5新用户在注册界面按照提示输入相关信息，注册成功后进行登录。登录成功后底部导航栏默认选中“文件”一栏，即进入用户文件列表展示界面，在该界面下，用户可以进行如下操作：

1. 点击文件列表中的文件夹项，可以查看该文件夹下的文件列表；点击文件列表中的文件项，可以浏览该文件内容。点击列表中每一项后面的“…”，弹出选项框，选中“删除文件”，可以删除该项。
2. 点击左上角按钮，弹出选项框，选中“上传文件”，可以查看手机内存文件，长按文件项弹出复选框，点击“OK”可以把勾选的文件上传到网盘当前所在文件夹目录下；选中“创建文件夹”，输入文件夹名称点击“确认”按钮，可以在当前文件夹下创建一个空文件夹。
3. 点击搜索框，进入搜索界面，输入关键字进行搜索，得到包含该关键字的文件列表，点击列表项可以浏览文件内容。
4. 点击右上角按钮，弹出选项框，选中“加入会话”，进入对应界面。点击“搜索”按钮，下方会展示出扫描到的设备列表，点击某一设备项发起蓝牙连接。蓝牙连接成功后，点击“分享”或“转存”按钮，进入文件选择界面，再点击“分享”按钮将勾选文件传输给接–收方；选中“广播会话”，用户设备可被扫描到，并被动等待蓝牙连接。蓝牙连接成功后，展示接收到的文件列表，点击列表项的“…”按钮，可以选择保存或删除该项。

选中底部导航栏的“我的”一栏，切换到用户个人信息展示界面，在该界面下可以进行如下操作：

1. 点击“修改密码”，进入对应界面，按照提示输入新旧密码实现修改。
2. 点击“设置”，进入对应界面，点击“登出”，客户端清空缓存并跳转到登录界面。
3. 点击“关于”，可以查看产品信息。

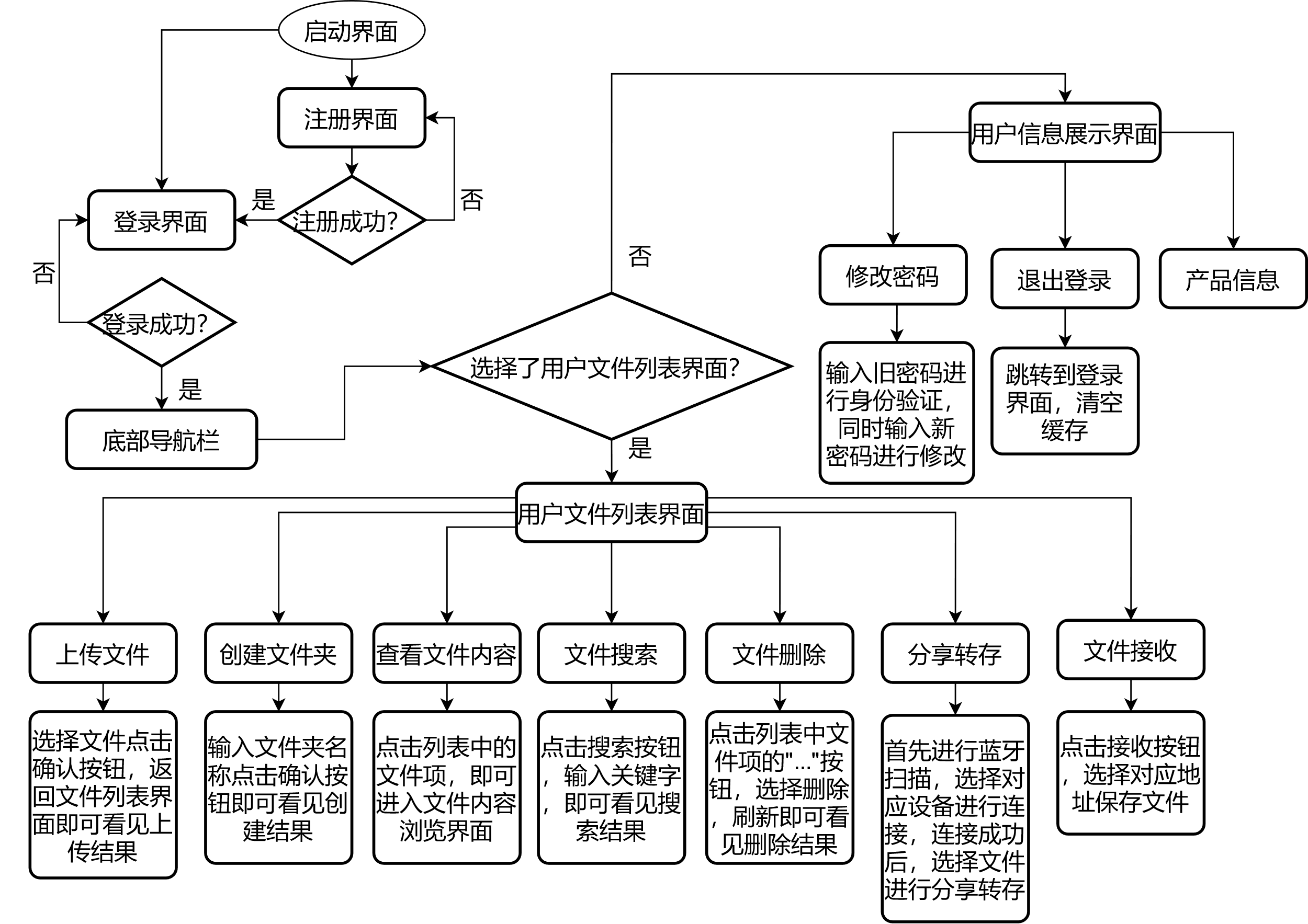


图 4‑5 系统操作流程图

## 4.4 系统展示

### 4.4.1 系统web端展示

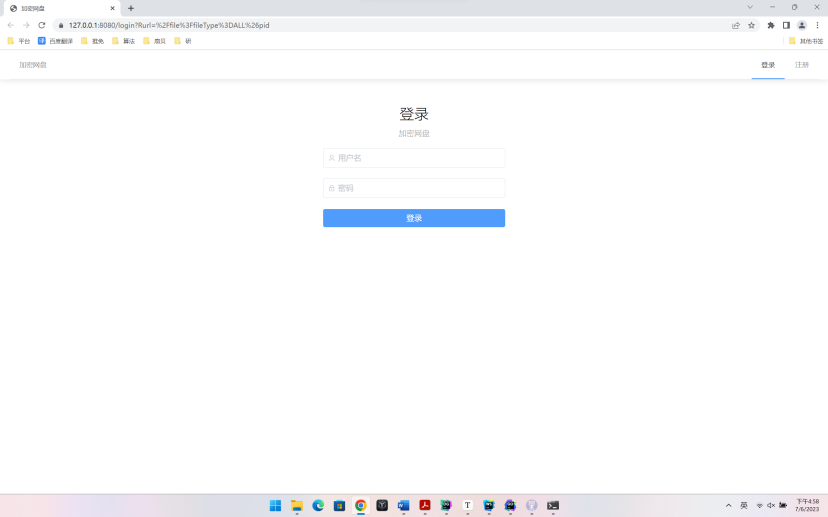


图 4‑6 登录页面

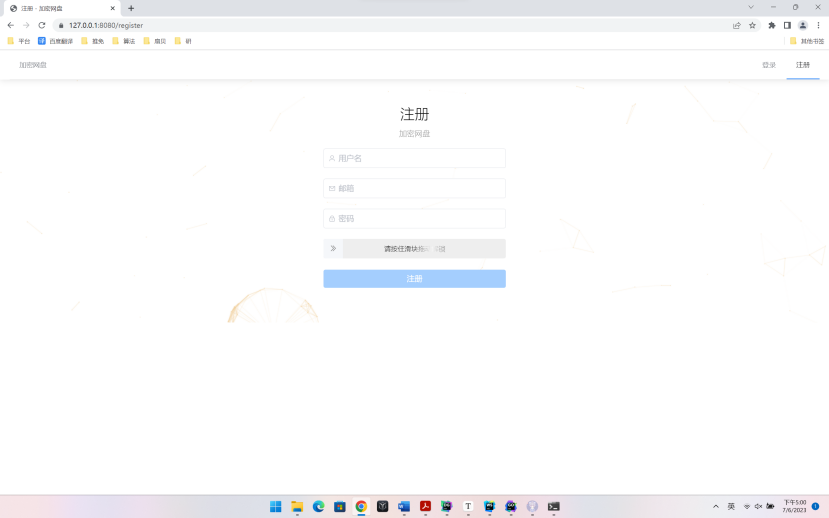


图 4‑7 注册页面



图 4‑8 注册成功



图 4‑9 登录成功



图 4‑10 文件上传成功



图 4‑11 查看文件内容

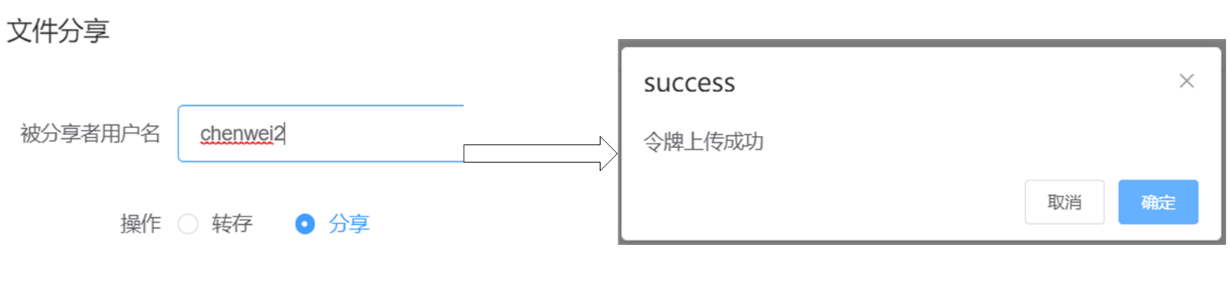


图 4‑12 分享令牌上传成功



图 4‑13 文件接收成功



图 4‑14 文件搜索

### 4.4.2 系统Android端展示



图 4‑15 注册、登录页面

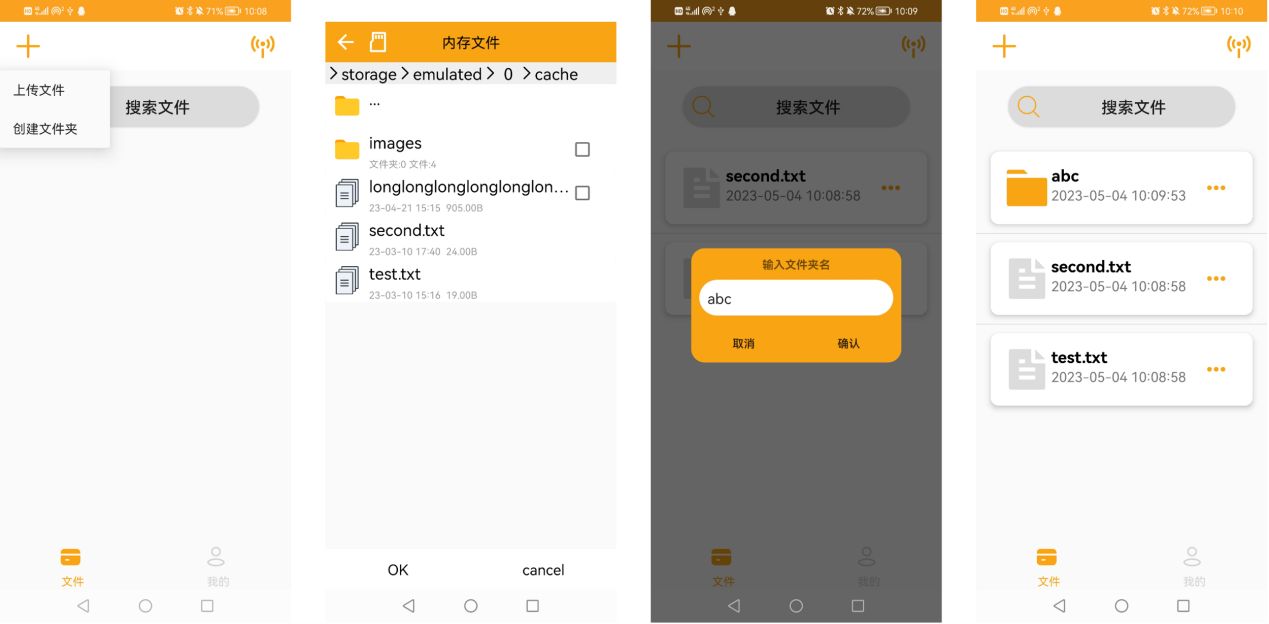


图 4‑16 文件上传、创建文件夹、文件列表展示页面

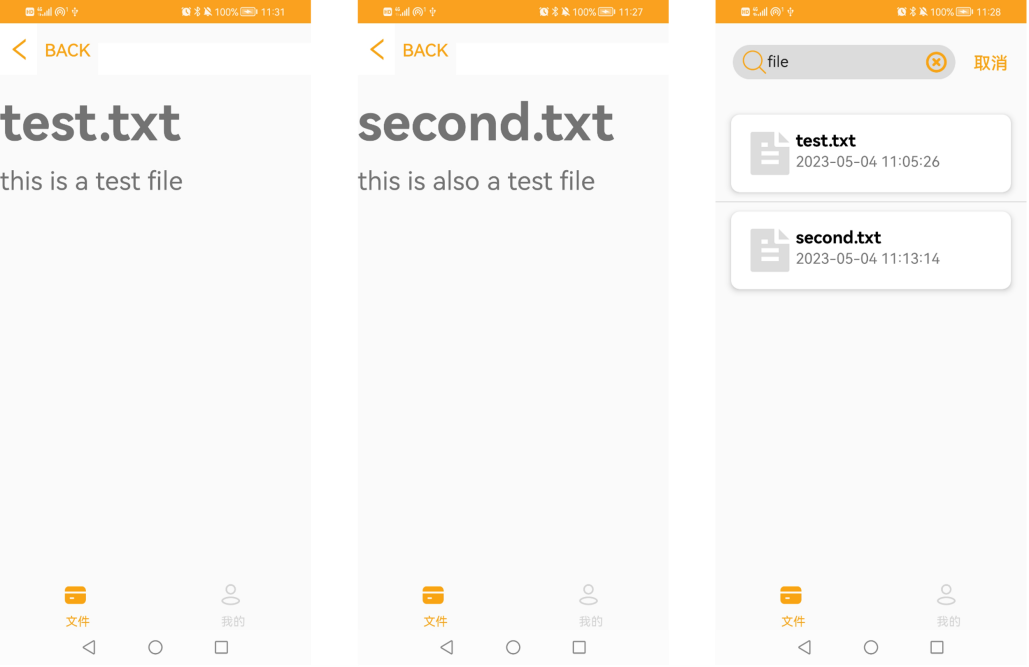


图 4‑17 文件内容浏览、搜索页面

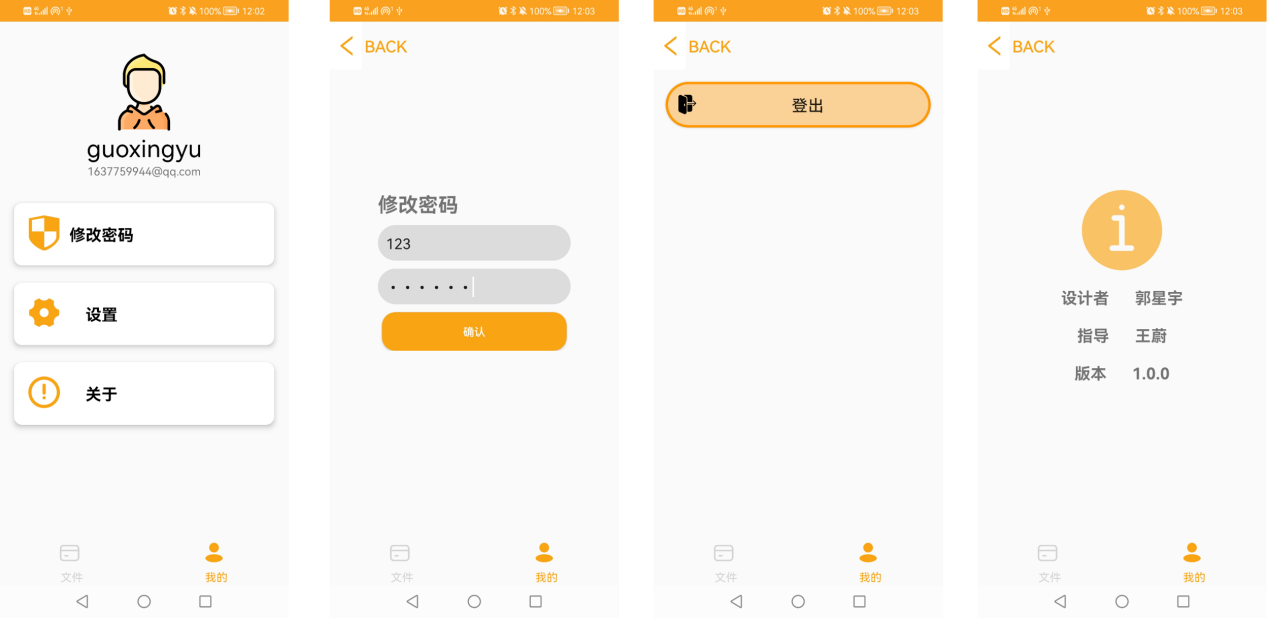


图 4‑18 用户信息展示、修改密码、设置、关于页面

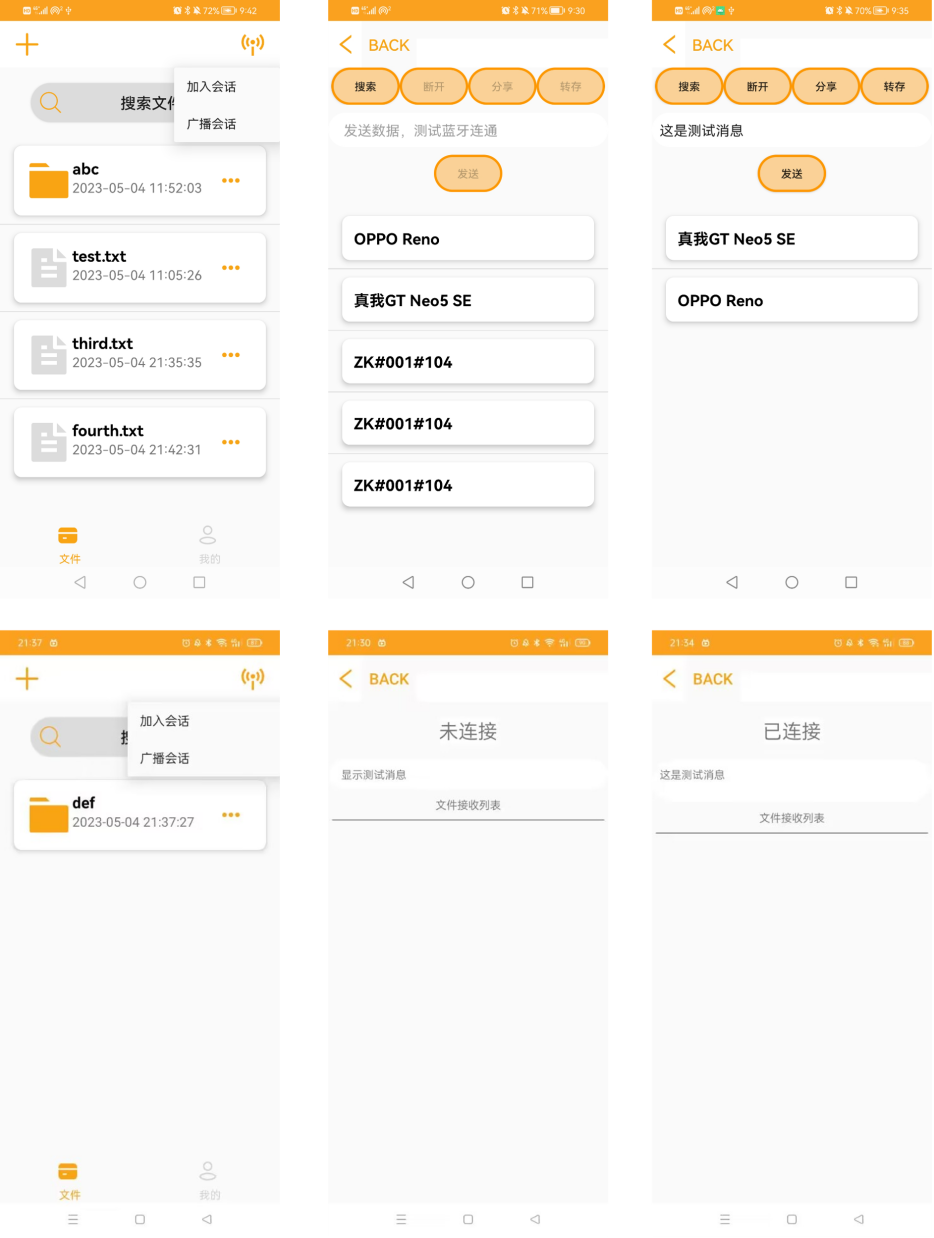


图 4‑19 蓝牙连接页面



图 4‑20 转存、接收页面

## 4.5 小结

本章对于系统实现进行了详细介绍。首先是对于密态云盘以及量子平台的开发环境，4.1节详细说明了使用的工具、开发平台等信息。然后针对系统项目搭建，4.2节主要说明了客户端以及服务端的设计思路以及实现方法、如何能够将客户端以及服务端的项目代码部署在传统计算机上运行、以及如何能够连接量子平台进行操作。在4.3节中对于系统操作流程进行了说明，给出了实际的执行逻辑。最后在4.4节中借助客户端，对于整个系统的功能进行了展示。之后在下一章将会对于整个系统进行更加全面的测试以及评估。

# 系统测试与评估

## 5.1 网盘功能测试

此部分着眼于密文数据搜索的功能，重点展现数据库中的密文数据，常规网盘功能不做测试。

测试情况见下列表格。

表 5‑1 注册成功后用户信息

|  |  |  |  |
| --- | --- | --- | --- |
| 身份验证信息 | 用户密钥key1 | 用户密钥key2 | 双向索引结构 |
| 2GHe6hc0xpgRArm88faC4PsqWC76IZecHvuG6SZr4yuqCJsLl+fdvijLBYrkdw+ioJ8j8Dp/5Bt2ZfaqTfOEdA== | DR4U7qW/7dFEJSv/ZyyAOxLbUVa+qH/N9qYkj6U796citfCQax6j5ls41/eluZt5 | 1ipZKQMDZ8t1Ybo6YNE73E9wS71UfAeCqPdtwevpZ8WqyhaQlTYvDu8ZYsOl1nGb | W3t9LHt9XQ== |

表 5‑2 上传文件后的文件密钥和用户双向索引结构

|  |  |  |  |
| --- | --- | --- | --- |
| 文件名 | 文件密钥 | 文件加密内容 | 双向索引结构 |
| test.txt | kJ+hPoqDJc0884TIjUyDmw1QpNoCp4wf2uQ8gfk1oDew4VaOHUIlzldYE3NRjv0w | 0jQYMDm+GHMojdvIfnFGolF07itoWPsLzRruetPuOUI= | W3siYSI6IjE2NTM5NjA5NDY0MjAyODk1MzYiLCJmaWxlIjoiMTY1Mzk2MDk0NjQyMDI4OTUzNiIsInRlc3QiOiIxNjUzOTYwOTQ2NDIwMjg5NTM2IiwidGhpcyI6IjE2NTM5NjA5NDY0MjAyODk1MzYiLCJpcyI6IjE2NTM5NjA5NDY0MjAyODk1MzYiLCJhbHNvIjoiMTY1Mzk2MDk0NjQyMDI4OTUzNiJ9LHsiMTY1Mzk2MDk0NjQyMDI4OTUzNiI6ImZpbGUiLCIxNjUzOTU4OTgwMTgwOTA1OTg0IjoiZmlsZSJ9XQ== |
| second.txt | pIUkULnSp72pRWEQdVSfhFjDrBdwwb2A7Qy5u+f5Te8266XKW0HZuAjmVyzSJ5jo | YZ7yqAxjYrHB/XNP5qDnoHHX+jEO4pBoj/BipOmJpwE= |

表 5‑3 test.txt的部分搜索索引（一）

|  |  |  |  |
| --- | --- | --- | --- |
| 关键字 | L | Iw | Rw |
| this | 4AJWCiJrWsqkCJpkqe9arA9WUZ6Y7PgjnUAa8mQCdjI= | GOVXsa3lVqZ4l4BjAr2+/qpXToPuBBU6Rj8YeXX4cojuAPk24XT4GkQ/8Sf2u/kF892HojeB7pCiZWhw/p9TzA== | BwIEBQUIAwUBAwkEBwQFAgMCAwQGBQcGCAQCCQIDCAY= |
| test | CMIOLieqwx9vnS5gwUC+2fxKNOjFRXjpfcTKp6bI1gM= | pZ9RBR30CAvpM3Ys+eUCb1VmN7O04LTyvv7hQeiofBbhV+z2wLY/ydGgBKa14KGlQQX2LDrS7px7xXUw+LrLww== | BwUECQkJCQEEAQcJAwcDCAUIAgQBBAUCBQkDAgQGAwE= |
| file | T21GzwV+H/Vpvve+3Eg9DzI6CEESTN46d5bcarjWmvU= | S0xhwedzLc/amLNBpNGrDfY0dL/3vxm4aTdFtKEfupmr5o+TNgUrimihfa3UChB26efDPqtHg7DpIf4lUfGQcQ== | BwADCAYBCAEJAQgBBgYACAcHBAMFBQUDAgYCAQkIBQI= |

表 5‑4 test.txt的部分搜索索引（二）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 关键字 | Cw | Iid | Rid | Cid |
| this | BiIhTN7e2+qbxp8I/fx436yy28OjH9bsk73CDZ76Rg8= | zy1iz0TPIVEuwbw+BvioCrj78J7GQwAiTivUszUAI3lCaKf+G/B8jNxxp3gFF9f0+kiu2dT32pA9HNwEUGFfaw== | AgkFBAcGCAkJBgEDAQIJAQYCBgMHAwMEAwkEBQUDAQU= | yN87TMV38eV5tz4GKoakkg== |
| test | t+TRegRyFGr5IMOFRh25u8XOM0L8YkZHDeF9ms/8QG8= | FncoZoSPruw5PlYX4nPFenEJ+QwsimnACBix7uJi/LuwTL/eu/TtD1oDUnpe1TOxep8UVgL3XgCbhkURwsjrdg== | AwAEAgIGAQUEAgYFCQgDBwEIAAYFBgYIAgcACQADCQI= | sZeVolezwxWfKNRjUOyatw== |
| file | onCukpttcwhCsDQSbZFW0eX3UxMVJoTiJ/cnkU5icFU= | VjMEi0EEhJarE/lv5h9dM0gCIyKwujQvNbKYg58yxr44O6yU1h3Rk7cuf3Vf7Y41wym3/kpWJYXrlhMrCFMqHA== | BAcBAAUABQgDBgEIBwQFAQEFCAMGBgUCBwcHAwkEAgY= | TZY1vfP9tMIsz2USEkwhkQ== |

表 5‑5 搜索令牌

|  |  |  |
| --- | --- | --- |
| 关键字 | L | Jw |
| file | nY7fedZh+cLf1uaQJEm7re5fauorwtdx6dyNz1lLk9A= | 3HO5AWrcAZNRR60m8yZ5kyA3fJzs6ffvh+w1xXUUkRc= |

表 5‑6 新的身份验证信息和重新加密的用户密钥

|  |  |  |  |
| --- | --- | --- | --- |
| 新密码 | 身份验证信息 | 用户密钥key1 | 用户密钥key2 |
| gxy123 | hPOCV/JqOxSmUxNrRCfHFZM2ocFY3Q9KNtM4i52/wKgpBgfHu6mWhO38C1V0htpPlS0WmWnvbJFgXLgN5YshUw== | lsads5XnYJuKdl98hwGnU6PCyHXT3EAfyG/fV1WUXoql7l3b+v1K/EUSBL1OJvjf | bAiiT3F6r/SQO/5p+yYIanMfB0pEORgOs+ptvwnF3V8O+5BA+e3c13E5ps0HiOi+ |

表 5‑7 分享令牌和RSA加密后的文件密钥

|  |  |  |  |
| --- | --- | --- | --- |
| L | Jid | Kid | 文件密钥 |
| 5C6JAg1KiTZ1kOWKduVWxEylwX5WuNnT/wxERCKfRSk= | zHptGBWhO0nzGkKvydl8WPD+cPQUHXdoVEJ9KFOnqx0= | lbEkV3JkIdDz5Ha5YPl4hAQ71rIDXalKaSy6SQGfAaA= | GkfmN+YQPVIY1cwLopu6Z0rm0nZfXTKYKXA4ajHNwi8Q2dfnJz/SlrBQhKKH1vx2E2kY65zCEA0CrLFfiBDR4yFELBbq6BVCa968AiJzJW1k5nHFhyDyKvqcvjemMABtEbNPRTJMTVvP+eNpKfiGZTsRJXhAQAl721GfjgAL05c= |

## 5.2 量子搜索测试

### 5.2.1 实验环境

表 5‑8 实验环境

|  |  |
| --- | --- |
| 开发环境 | 实验环境 |
| Windows系统 | Windows系统 |
| 8核CPU | 8核CPU |
| 16G内存 | 量子计算云平台 |
| python语言 | IsQ量子编程软件 |

### 5.2.2 实验测试

可扩展的Grover算法是实现密态云盘加速检索的核心算法，Grover算法进行搜索加速的部分主要负责一下两个任务：

（1）在存储文件时将存储逻辑进行保存。客户端获取关键字（keyword）后，将其加密为与，代表数据库索引，与前一可搜索密文相关，同时设计并获取搜索函数，即用来判断搜索结果是否为对应的的函数。

（2）在查找文件时通过Grover算法进行搜索。通过设计好的量子线路和搜索逻辑对可搜索密文进行搜索，即通过，获取。

接下来，以4bit所代表的空间作为搜索空间，对密态云盘的后端Grover算法实现加速搜索的部分进行实验测试。

对于一个4位的搜索空间，我们可以将关键字与搜索密文加密为4位二进制数，例如: 0000, 0001, 0010, …, 1111。假设我们输入为0000，希望搜索到的是1010，那么对应的Oracle门将对1010返回1，对于其他密钥则返回概率趋近于0。

在存储过程中，每存入一个文件，我们都记录下和对应的映射关系。为了使Grover算法在搜索过程中保持独一性，我们通过哈希函数使得和尽可能均匀地落在搜索空间中。

在已经完成存储的搜索空间中，我们设计了一个量子线路来实现Grover算法。Grover算法通过反复应用一个变换，来增加目标状态的概率。每次迭代，Grover算法会通过应用两个变换：Hadamard变换和Oracle变换来更新量子状态。

通过对已有的4bit搜索示例，我们可以扩展其中的量子线路实现更大空间的搜索，也可以通过并行拼接，将其扩展到更高维度的搜索空间中，实现更大空间的搜索。基于量子算法的优越性，通过量子线路实现空间搜索的范围越大，对应的整体搜索算法时间复杂度就越低。

具体的实验流程将在实验过程部分展现。具体在性能评估部分，我们将对量子算法的准确性，以及基于4bit，8bit、16bit、32bit和64bit的量子算法相对于经典算法的优势进行评估。

### 5.2.3 实验过程

对于量子Grover算法进行密态云盘搜索加速的核心部分，我们可以通过isQ提供的本地模拟环境进行量子计算机的模拟，也可以在量子计算云平台通过超导量子平台ClosedBetaQC进行实验，实现真正超导量子环境下的密态云盘密钥搜索。

为了更好地进行Grover算法加速密态云盘搜索部分的测试，我们将密态云盘的核心搜索算法，Grover算法相关的存储和搜索部分单独进行实验。

运行量子程序的方法有两种，本地运行依赖封装好的isqopen库，调用isq.run()运行量子程序；云平台运行使用CloseBetaQC，在验证用户密钥（login key）和确认运行机器（working machine）之后，云平台将根据系统产生的isQ量子门电路运行量子程序。量子程序运行返回的结果为一个二进制字符串，格式与输入相同，将其转换为十进制数，即。

首先我们对量子搜索系统进行本地测试，具体测试过程如图 5‑1所示。

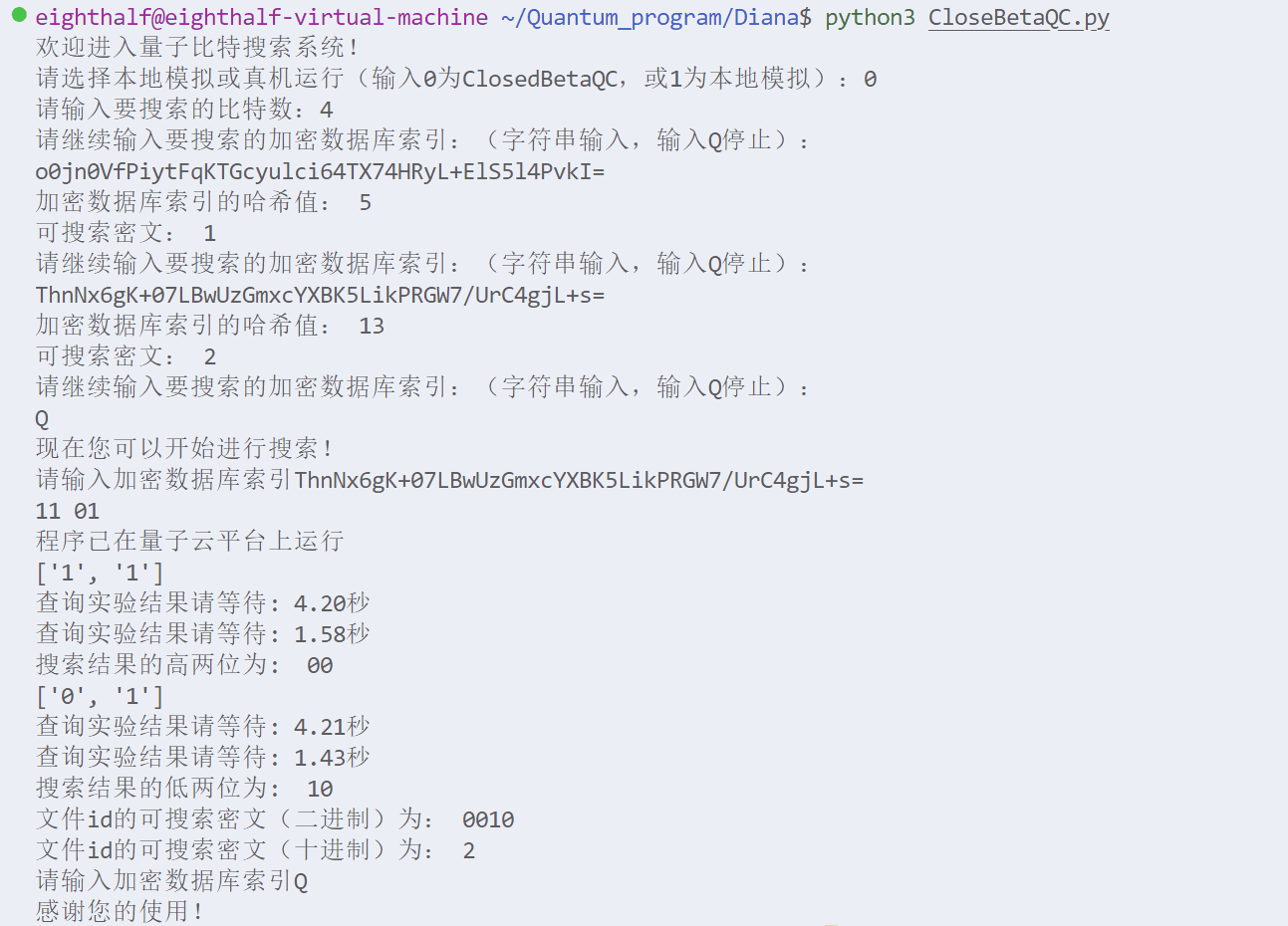


图 5‑1 本地模拟器运行量子程序

图 5‑1为量子搜索系统在本地运行的示意图。运行上述量子搜索系统时，用户首先需要在命令行中输入数据库索引与可搜索密文，系统将根据其映射关系，构建矩阵，以便接下来的搜索过程。当用户需要执行上述的搜索操作时，用户需要在命令行输入关键字，系统寻找到关键字对应的数据库索引（即）,将其转化为二进制的bit字符串，并应用Grover算法搜索，具体来说，使用数据库索引（即）在数据库中搜索可搜索密文（即），用于下一含有关键字密文文件id的查找，接着结合数据库中匹配到的密文id与密钥获取明文文件id。用户想结束输入，仅需在命令行窗口输入“exit”即可。

程序在超导量子平台ClosedBetaQC进行实验时，可以通过超导量子平台实现对于搜索算法真正的加速。同时在不超过12bit时，我们可以通过量子计算云平台查看对应的结果，如图 5‑2所示为4比特实现密态云盘搜索量子程序运行结果。

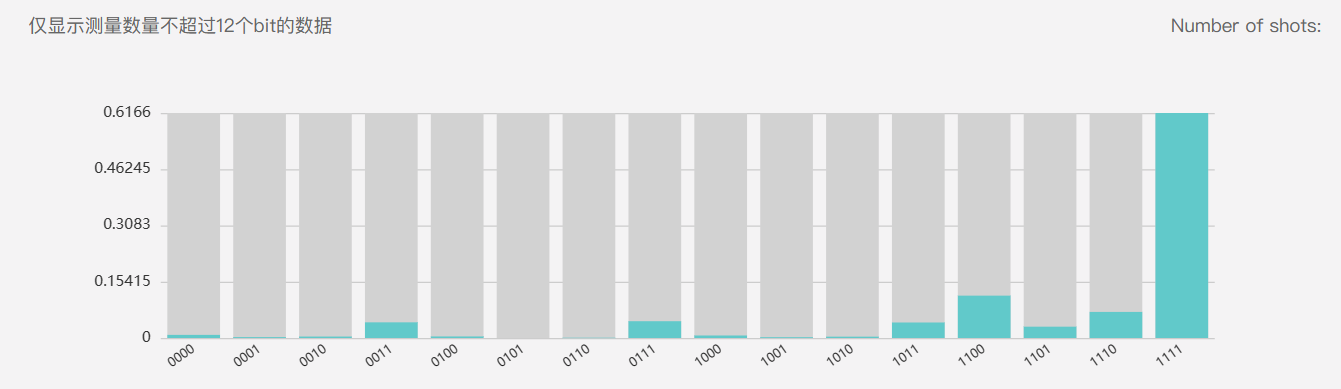


图 5‑2 量子计算云平台查看4比特量子程序运行结果

图 5‑2为4比特实现密态云盘搜索量子程序运行结果。

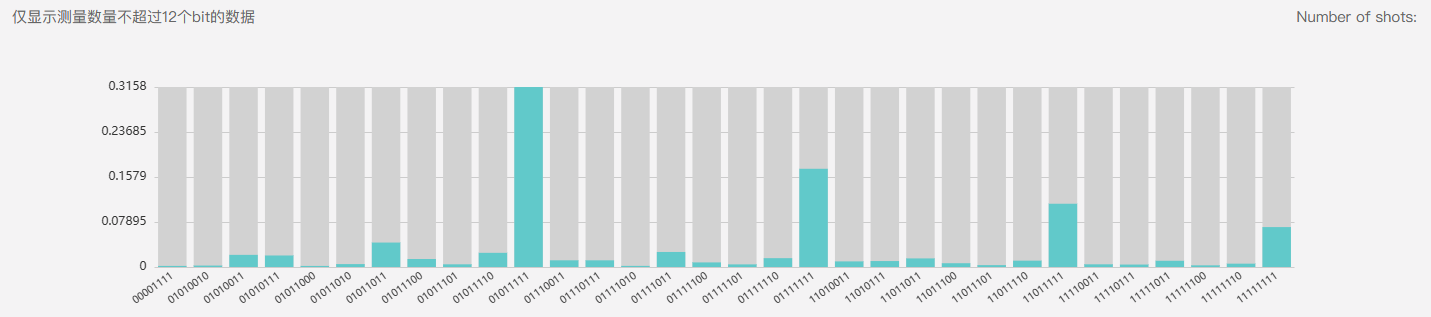


图 5‑3 量子计算云平台查看8比特量子程序运行结果

在分析实验结果时，需要考虑量子计算机的噪声和误差。使用量子Grover算法加速搜索得到的结果，服务器进一步对密态云盘上的文件进行处理。

### 5.2.4 实验评估

（1）搜索正确性测试

量子搜索的准确性是由量子算法的设计和实现决定的。在理想情况下，量子搜索可以实现100%的准确性，即在搜索空间中找到目标项。然而，在实际情况下，量子搜索的准确性受到量子比特的误差和纠错等问题的影响，可能无法保证100%的准确性。

因此，通过在量子计算云平台上进行多次测试，统计对应正确索引搜索概率，以衡量量子搜索部分的准确程度。计算连续5次在超导量子平台ClosedBetaQC进行实验返回结果的平均值作为最终的概率值，具体结果如表 5‑9所示。

表 5‑9 不同比特数对应平台搜索正确的概率值

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比特数目（bit） | 搜索空间 | 量子搜索  理想次数 | 正确结果  概率值 | 正确结果概率值  是否为最大 |
| 1 | 2 | 1 | 0.92 | 是 |
| 2 | 4 | 1 | 0.82 | 是 |
| 4 | 16 | 3 | 0.65 | 是 |
| 8 | 256 | 12 | 0.42 | 是 |

从上表中可知，随着比特数的增加，量子比特测量后可能的排列组合数目呈指数增加，量子搜索的正确结果只有一个，因此出错数量所占的比例也在增大，最终正确结果出现的概率也越来越小，但是整体的正确情况符合预期。

（2）量子搜索的速度优势

表 5‑10 不同比特理想搜索次数的对比（搜索次数向下取整）

|  |  |  |  |
| --- | --- | --- | --- |
| 比特数目（bit） | 经典搜索  理想次数 | 量子搜索  理想次数 | 加速比 |
| 1 | 1 | 1 | 1.00 |
| 2 | 2 | 1 | 2.00 |
| 4 | 8 | 3 | 2.67 |
| 8 | 128 | 12 | 10.67 |
| 16 | 32,768 | 201 | 163.02 |
| 32 | 2.15e+9 | 51471 | 41722.20 |
| 64 | 9.22e+18 | 3.37e+9 | 2.73e+9 |

表 5‑10是在量子计算环境与经典计算环境，固定相同比特的搜索空间，通过Grover算法进行搜索和通过经典遍历搜索的方法，在理想状态下实现对无序的密态云盘搜索的平均搜索次数对比。

为了更直观地展示经典搜索与量子搜索之间理想搜索次数的差异与比特数目之间的关系，将比特数目作为横坐标，经典搜索和量子搜索的理想搜索次数作为纵坐标进行绘图，如下图 5‑4所示。

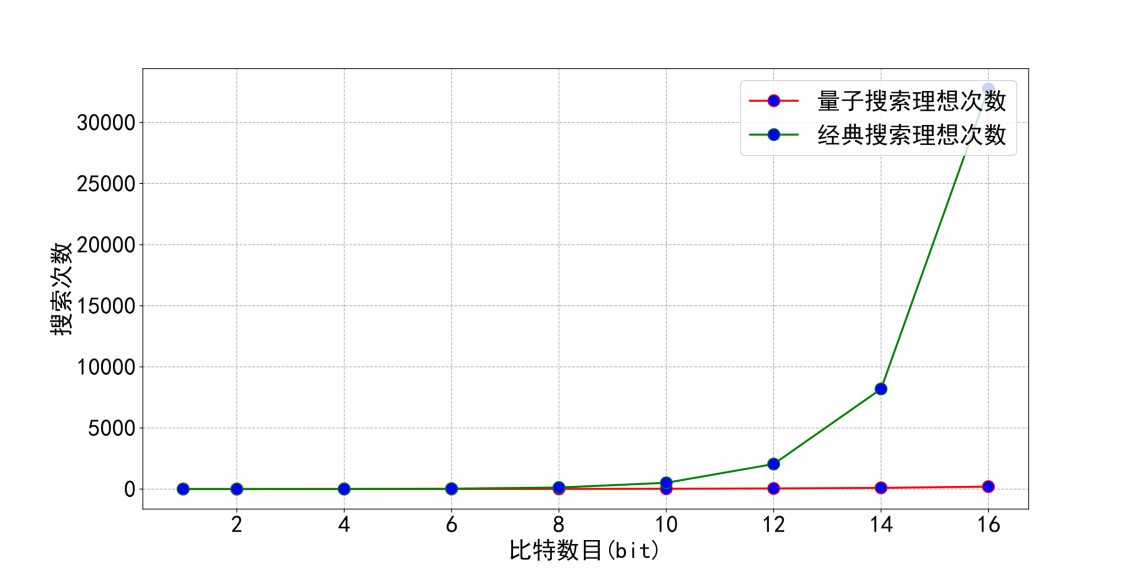


图 5‑4 经典搜索与量子搜索之间理想搜索次数的差异

由上图可知，随着比特数目的增加，经典搜索与量子搜索之间理想搜索次数的差异逐渐变大。为了更加明显地看到量子搜索相比于经典搜索的优越性，本项目进一步计算了量子算法与经典搜索算法的加速比，具体如图 5‑5所示。

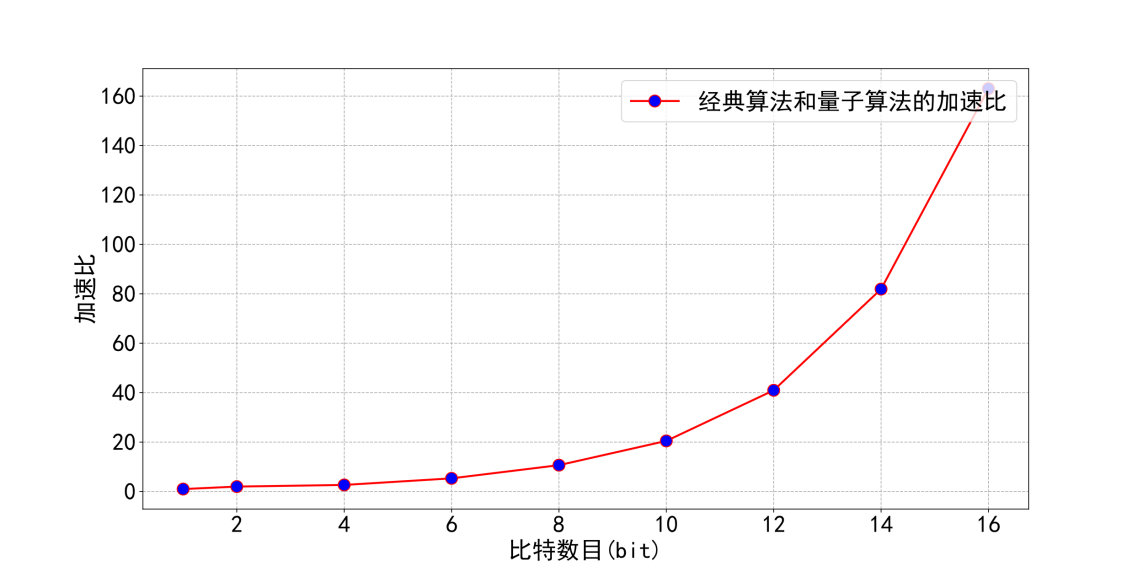


图 5‑5 量子算法与经典搜索算法的加速比

由此可见，量子搜索的超高性能随着搜索空间的增大不断突显出来。

（3）量子搜索的空间优势

量子搜索Grover算法的空间优势主要体现在其空间复杂度方面。与经典的线性搜索算法相比，Grover算法具有更小的空间复杂度，可以使用更少的内存来存储搜索空间。

具体来说，搜索一个大小为N的密态云盘空间时，Grover算法只需要个量子比特来表示搜索空间中的所有项。而对于经典的线性搜索算法，需要使用的内存来存储搜索空间中的所有项，因此其空间复杂度为。具体空间对比如表 5‑11所示。

表 5‑11 量子搜索与经典搜索空间对比

|  |  |  |
| --- | --- | --- |
| 比特数目（bit） | 量子搜索  搜索项数 | 经典搜索  搜索空间 |
| 1 | 1 | 2 |
| 2 | 2 | 4 |
| 4 | 4 | 16 |
| 8 | 8 | 256 |
| 16 | 16 | 65,536 |
| 32 | 32 | 4.29e+9 |
| 64 | 64 | 1.84e+19 |

因此，Grover算法的空间优势使得它在处理大型搜索空间时更加高效。

（4）量子搜索之间的对比

由于现在的量子计算机还处于发展初期，很多经典的量子算法按照理想状态下是运行不起来的，因此需要设计出与计算机高度契合的算法。平台上一个单比特基础门约为30ns，一个双量子比特门约为100ns。因此在实际量子计算时，需要将量子门的计算时间考虑在内。最终计算不同比特搜索时理想的搜索时间，结果如图 5‑6所示。

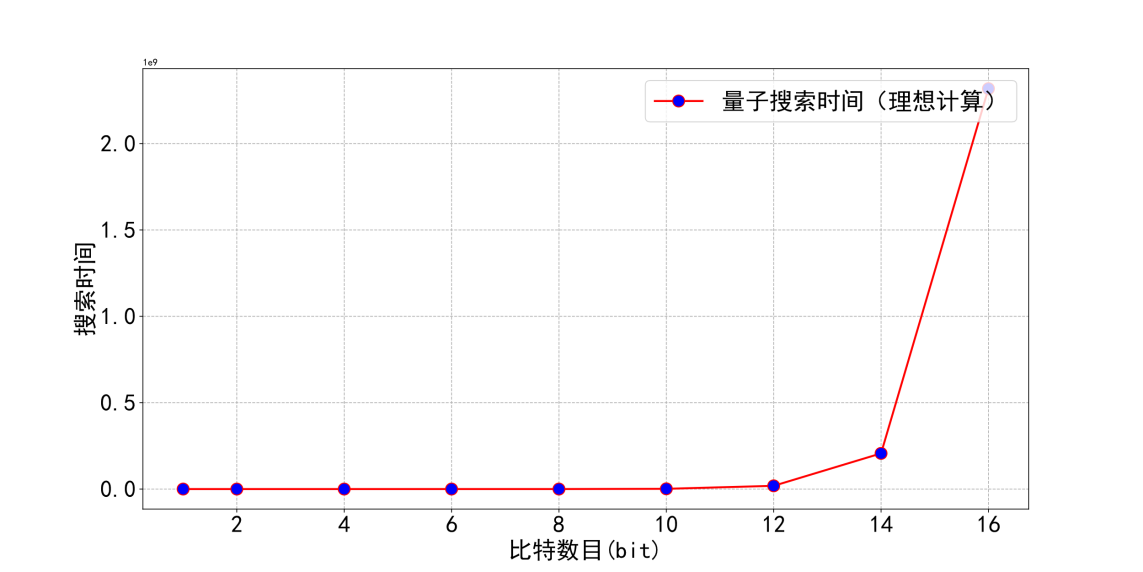


图 5‑6 不同比特搜索时理想的搜索时间

从上图可以看出，随着量子比特数目的增加，搜索时间呈指数型增长，说明在本项目设计的量子线路中，转化成的单比特门和多比特门的数量也在呈指数增长，这说明在目前量子计算机发展的初期阶段，并非使用越多数量的比特进行搜索，最终能够达到的效果越好。

因此本项目使用量子算法与经典算法的思路相结合，在搜索过程中，可以通过对量子比特进行拼接的方式，使得其能够进行无限范围内扩展的空间搜索，这同时也对密钥的哈希映射提出了更高的要求。在理想状态下，哈希映射后的哈希索引能够均匀地落在存储空间。同时本项目也进一步减少了过程中量子门的使用，使得通过量子电路的时间不断缩短，量子的可扩展性不断提高。因此本项目的方案的效果是可以随着量子计算机的发展迭代不断优化的。

## 5.3 小结

本章主要完成对于网盘功能的测试和量子搜索的测试。在网盘功能测试部分，主要针对密文数据搜索的功能进行演示，重点展现数据库中的密文数据；在量子搜索测试部分，主要针对可扩展的Grover算法进行数据库搜索方面的测试，同时对于量子搜索的准确性、空间优势和速度优势进行了进一步的评估，最终验证了量子算法相对于经典算法的优越性。在下一章将进一步进行密态云盘的优势分析和应用推广方面的介绍。

# 优势分析与应用推广

## 6.1 网盘安全性分析

本密态云盘是在全密态环境下运行的，具体保护措施包括：

1. 在客户端与服务器间通信时采用HTTPS协议。
2. 对用户隐私数据采用AES、SHA和HMAC等算法进行加密后才存储到数据库。
3. 用户间进行蓝牙通信时，采用RSA算法加密传输的文件密钥，防止被恶意窃取。
4. 采用的可搜索加密方案被证明是前向安全和后向安全的。

在以上安全措施的共同保障下，本密态云盘能够防止用户隐私数据泄漏，具有很好的安全性。

## 6.2 网盘搜索速度分析

本项目实现了多比特可扩展Grover算法，利用量子计算机突破经典算法搜索加速的瓶颈，将数据搜索的时间复杂度降低至。

## 6.3 方案创新性分析

（1）算法创新

使用可搜索加密模型KS2E，实现以密文数据库为基础的密态云盘，具有远高于市场上已有网盘的安全性。

在Grover算法的基础上，针对数据库搜索，提出了一种可扩展比特位的量子加速搜索方案。

（2）应用创新

本方案着眼于以数据库为实体的网盘搜索加速。对于乱序数据的搜索，经典搜索算法受限于计算机系统结构，需对数据进行遍历操作，最坏情况下的时间复杂度为。

本方案将量子计算平台与数据搜索的实际应用场景结合，在尽量保证搜索正确性的基础上，将Grover算法应用于密态云盘的搜索加速，利用量子计算机突破经典算法搜索加速的瓶颈，将数据搜索的时间复杂度降低至，提高了网盘在大规模数据处理环境下的效率。

## 6.4 方案商业价值分析

现在市面上的网盘简要分为加密和非加密存储两种，其中加密网盘主要有Sync，Tresorit，MEGA，Woelkli。非加密存储网盘主要有onedrive，iCloud，Google drive，百度云。还有一些支持用户自主选择本地加密或服务器端加密的网盘，比如国内的阿里云。而本项目的密态云盘具有以上产品几乎所有的优势。

（1）功能上：本文的产品综合了以上两种产品的独有功能——既拥有加密云盘的加密存储、加密文件分享功能，也具备常规非加密云盘的检索功能。对于需要处理极大量云数据并且同时对安全性有着高要求的客户，云上密文关键内容检索能协助他们更安全地找到包含这些关键内容的数据部分并且准确下载这些数据。而传统的方案只能通过牺牲对服务器运营商的安全性来做到云检索，抑或是选择将大量数据全部下载到本地再明文检索，这样对比下显然有着非常大的差距。

（2）效率上：常规的密文陷门（trapdoor）检索功能对服务器性能支出有着额外的时间和性能开支，但是对关键加解密步骤实行的量子算法加速可以使密文检索有着更接近传统明文检索的效果。换而言之，经过量子加速的密文检索算法能更好适配有着极大量加密云数据的客户，能在保证安全的基础上从各方面节约他们的时间。

综上，本项目的“密态云盘+量子加速密文检索”是行业内最适合顶尖客户的高需求严标准产品，在兼顾安全性和生产力的工具中具有不可替代的地位。

在当前加密算法安全性被量子攻击威胁的形势下，密态云盘将会是网盘发展的新趋势。本项目的产品不仅做到了在数据存储上抗量子攻击加密，“量子加速+密文检索”的组合更是有着巨大的现有和潜在市场。随着量子计算的飞速发展和密码学进入后量子时代，本项目的产品因处于这两门新技术的直接交叉区而有着巨大的发展潜力，适合提前布局。

## 6.5 应用推广优势

量子加速效果指数型正相关于目前可调用量子比特数。根据本项目目前的研究结果，现有的Grover算法在和本项目的密文陷门搜索算法进行适配后，位量子比特能加速数量级的索引数目（对于一步时间复杂度的检索，能将其至少优化到），加速效果明显。

因此，本项目的“加密搜索+量子加速”方案在目前的少比特量子计算机上就已经存在较高的应用价值，并且方案的高度通用性和易扩展性使得仅需要单纯增加量子比特数目就能获得较好的优化提升。

由于密文陷门检索有着比常规明文检索更高的算法复杂度，量子算法在其上的优化效果会更明显，有着更高的应用价值。换而言之，量子计算对云盘类产品的算法优化在本项目的产品上具有先天的适配性。

在应用推广上，本项目的网盘还具有良好的可移植性。在软件实现时，本项目进行了应用程序界面抽象化和代码模块化，封装程度较高，软件功能完备，系统成熟，部署软件在服务器上成本极低。

# 总结

本项目实现了量子比特加速检索的密态云盘设计，创新性地使用了Grover算法加速无序数据库搜索的方式去尝试解决这一学术界与工业界都普遍难以解决的问题，相较于传统的顺序检索方案，将时间复杂度从降低到，极大地提升了检索效率。

在软件开发层面，本项目实现了web端与Android端的开发，完整实现了注册，登录，退出等用户交互功能，同时开发了文件列表展示，文件内容浏览，文件接收，文件创建，文件删除等功能，总体上完整开发了网盘的所有功能，具有界面美观，用户交互流畅，可以跨平台使用的优势，并将Grover算法与网盘密文搜索结合，通过Grover算法实现搜索加速，相较于传统算法具有较大的优势。

在量子算法层面，本项目实现了多比特可扩展Grover算法。本项目对传统的Grover算法只能进行固定搜索这一问题做了进一步的改进，改进后的Grover算法可以实现传统数据库的增删改查功能。同时考虑到现阶段量子计算机正在快速发展，量子比特数不断增加的现状，本项目做了更进一步的扩展，使得扩展后的Grover算法可以实现自定义比特数大小的查找，用户可以根据所使用量子计算机的量子比特数量自定义查找范围。

本项目结合现有量子计算云平台，利用量子计算机进行数据库搜索加速，更好地体现了量子计算在未来的价值和发展前景。

参考文献

1. Konstantopoulos M, Diamantopoulos P, Chondros N, et al. Distributed personal cloud storage without third parties[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(11): 2434-2448.
2. Chang Y C, Mitzenmacher M. Privacy preserving keyword searches on remoteen crypted data[C]. in: ACNS: vol. 5: 3531. 2005: 442-455.
3. Watanabe Y, Ohara K, Iwamoto M, et al. Efficient Dynamic Searchable Encryption with Forward Privacy under the Decent Leakage[C]. in: Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy. 2022: 312-323.
4. Stefanov E, Papamanthou C, Shi E. Practical dynamic searchable encryption with small leakage[J]. Cryptology ePrint Archive, 2013
5. Shor P W. Algorithms for quantum computation: discrete logarithms and factoring[C]//Proceedings 35th annual symposium on foundations of computer science. Ieee, 1994: 124-134.
6. Grover L K. A fast quantum mechanical algorithm for database search[C]//Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. 1996: 212-219.
7. Harrow A W, Hassidim A, Lloyd S. Quantum algorithm for linear systems of equations[J]. Physical review letters, 2009, 103(15): 150502.
8. Zalka C. Grover’s quantum searching algorithm is optimal[J]. Physical Review A, 1999, 60(4): 2746.
9. Boyer M, Brassard G, Høyer P, et al. Tight bounds on quantum searching[J]. Fortschritte der Physik: Progress of Physics, 1998, 46(4‐5): 493-505.
10. Long, G., Liu, Y. Search an unsorted database with quantum mechanics. Front. Comput. Sc. China 1, 247–271 (2007).
11. Priebe C, Vaswani K, Costa M. EnclaveDB: A secure database using SGX[C]//2018 IEEE Symposium on Security and Privacy (SP). IEEE, 2018: 264-278.
12. Wang Y, Wang J, Chen X. Secure searchable encryption: a survey[J]. Journal of communications and information networks, 2016, 1: 52-65.
13. Bossuat A, Bost R, Fouque P A, et al. SSE and SSD: Page-efficient searchable symmetric encryption[C]//Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part III 41. Springer International Publishing, 2021: 157-184.
14. Cash D, Tessaro S. The Locality of Searchable Symmetric Encryption[C]//Eurocrypt. 2014, 8441: 351-368.
15. Long G, Liu Y. Search an unsorted database with quantum mechanics[J]. Frontiers of Computer Science in China, 2007, 1: 247-271.
16. Boneh D, Di Crescenzo G, Ostrovsky R, et al. Public key encryption with keyword search[C]//Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23. Springer Berlin Heidelberg, 2004: 506-522.
17. Deshmukh D A P, Qureshi D R. Transparent Data Encryption--Solution for Security of Database Contents[J]. arXiv preprint arXiv:1303.0418, 2013.
18. Crypteron. Cipherdb developer’s guide. , 2014.
19. Vemula D R, Konar D, Satheesan S, et al. A Scalable 5, 6-Qubit Grover's Quantum Search Algorithm[J]. arXiv preprint arXiv:2205.00117, 2022.
20. Seidel R, Becker C K U, Bock S, et al. Automatic generation of Grover quantum oracles for arbitrary data structures[J]. Quantum Science and Technology, 2021.
21. Wu Y, Bao W S, Cao S, et al. Strong quantum computational advantage using a superconducting quantum processor[J]. Physical review letters, 2021, 127(18): 180501.
22. Banker K, Garrett D, Bakkum P, et al. MongoDB in action: covers MongoDB version 3.0[M]. Simon and Schuster, 2016.
23. Song D X, Wagner D, Perrig A. Practical techniques for searches on encrypted data[C]//Proceeding 2000 IEEE symposium on security and privacy. S&P 2000. IEEE, 2000: 44-55.
24. Kamara S, Papamanthou C, Roeder T. Dynamic searchable symmetric encryption[C]//Proceedings of the 2012 ACM conference on Computer and communications security. 2012: 965-976.
25. Popa R A, Zeldovich N. Multi-key searchable encryption[J]. Cryptology ePrint Archive, 2013.
26. Popa R A, Stark E, Valdez S, et al. Building web applications on top of encrypted data using Mylar[C]//11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14). 2014: 157-172.
27. Grubbs P, McPherson R, Naveed M, et al. Breaking web applications built on top of encrypted data[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 1353-1364.
28. Hamlin A, Shelat A, Weiss M, et al. Multi-key searchable encryption, revisited[C]//IACR international workshop on public key cryptography. Springer, Cham, 2018: 95-124.
29. Wang J, Chow S S M. Omnes pro uno: Practical multi-writer encrypted database[C]//USENIX Security. 2022.
30. Agrawal R, Kiernan J, Srikant R, et al. Order preserving encryption for numeric data[C]//Proceedings of the 2004 ACM SIGMOD international conference on Management of data. 2004: 563-574.
31. 张玉磊, 刘祥震, 郎晓丽, 等. 云存储环境下多服务器的密钥聚合可搜索加密方案[J]. 电子与信息学报, 2019, 41(3): 674-679.
32. 甘庆晴, 王晓明. 云环境下一种高效的密钥聚合加密方案[J]. 计算机工程, 2016, 42(2): 33-37.
33. "2014年8月名人照片泄露事件." 维基百科: 自由的百科全书, 2023.