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

图 目 录

表 目 录

1 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$O(N)$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$O(\sqrt{N})$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 KS^2E，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$O(\sqrt{N})$，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。

2 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 KS^2E 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。

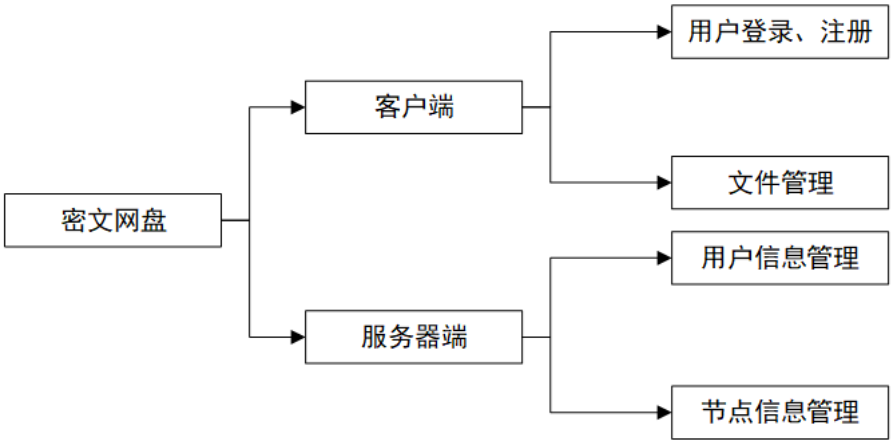

图 2-1 系统功能框架

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。

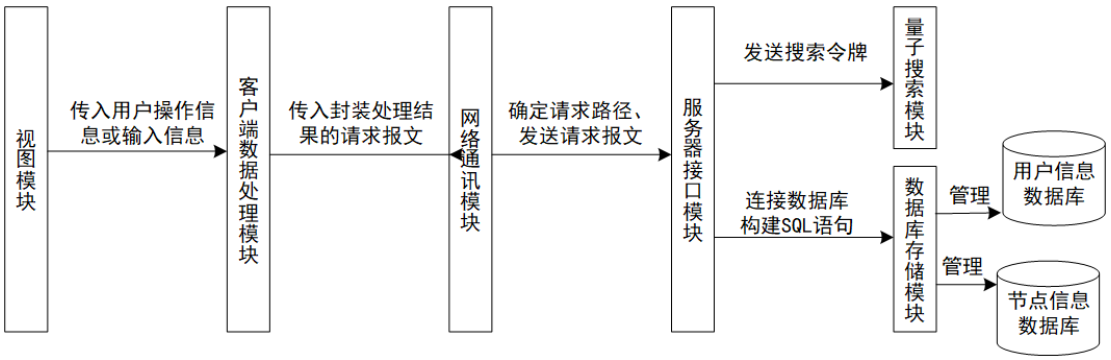

图 2-2 系统的模型结构图

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。

注册接口	
请求参数	username string hashId string email string biIndex string key1 string key2 string
响应参数	

登录接口	
请求参数	username string hashId string
响应参数	token string userInfo UserInfo

UserInfo
+ rootId: int64
+ username: string
+ email: string
+ biIndex: string
+ key1: string
+ key2: string

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。

发送搜索令牌接口	
请求参数	searchToken SearchToken
响应参数	Cw repeated string

搜索接口	
请求参数	nodeId repeated int64
响应参数	nodeList repeated Node

SearchToken
+ L: string
+ Jw: string

Node
+ nodeType: NodeType
+ nodeId: int64
+ nodeName: string
+ nodeContent: bytes
+ secretKey: bytes
+ createTime: int64
+ updateTime: int64

<<enumeration>> NodeType
+ Unknown
+ File
+ Dir

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。

获取文件列表接口		获取文件内容接口		创建文件夹接口	
请求参数	nodeId int64	请求参数	nodeId int64	请求参数	dirName string parentId int64
响应参数	nodeIdList repeated int64	响应参数	node Node	响应参数	

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。

获取节点id接口		上传文件接口		IndexToken
请求参数		请求参数	fileName string parentId int64 indexList repeated IndexToken content bytes biIndex string nodeId int64 fileSecret string	+ L: string + Iw: string + Rw: string + Cw: string + Iid: string + Rid: string + Cid: string
响应参数	nodeId int64	响应参数		

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。

一轮分享接口		二轮分享接口	
请求参数	L string Jid string	请求参数	fileName string parentId int64 indexList repeated IndexToken biIndex string nodeId int64 fileSecret string isShare bool address int64
响应参数	S repeated string	响应参数	

图 2-7 一轮、二轮分享接口

删除接口		修改密码接口	
请求参数	delNodeId repeated int64	请求参数	oldHashId string newHashId string key1 string key2 string
响应参数		响应参数	

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。

3 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id，哈希表$lastW[id]$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$lastW[id]$，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w，哈希表$lastID[w]$指向该纵链上的最后一个文件id，每次插入时都将新的文件id放在链尾，并更新$lastID[w]$；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id。

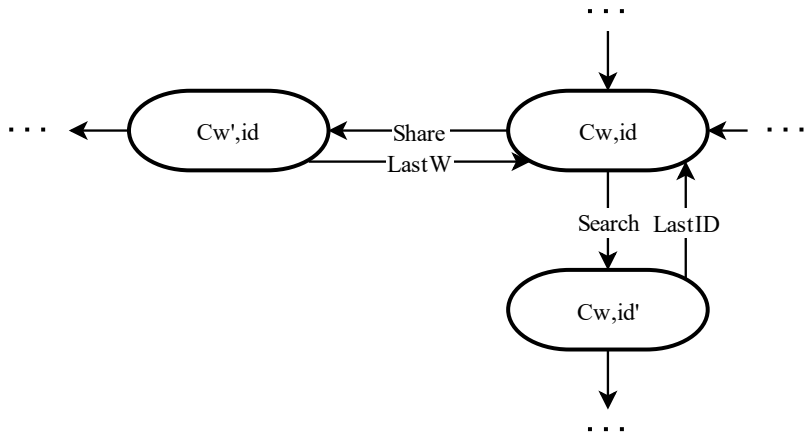

图 3-1 双向结构索引

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H，该函数输入两个长度为λ的比特串，输出一个长度为 2λ的比特串；初始化一个伪随机函数F，该函数输入一个λ长的比特串和一个任意长的比特串，输出一个λ长的比特串；初始化一个对称加密函数SE，该函数包含一个加密算法$\mathcal{E}$和一个解密算法$\mathcal{D}$；为用户随机生成λ比特的密钥$key1$、$key2$；初始化哈希表$lastW$和$lastID$。

（2）Update 函数利用（关键字w，文件id）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$L = F(key1, w||id)$，其中“||”符号表示比特串拼接。然后生成两个λ比特的随机数R_w和R_{id}，利用用户密钥$key2$得到$k_w = F(key2, w)$，$k_{id} = F(key2, id)$，$C_w = \mathcal{E}(k_w, id)$，$C_{id} = \mathcal{E}(k_{id}, w)$。利用$lastID[w]$找到链尾$id'$，如果$id'$是空，$I_w = H(F(key2, w||id), R_w)$，如果$id'$不是空，$I_w = H(F(key2, w||id), R_w) \oplus (F(key1, w||id')||F(key2, w||id'))$。利用$lastW[id]$找到链尾$w'$，如果$w'$为空，$I_{id} = H(F(key2, id||w), R_{id})$，如果$w'$不为空，$I_{id} = H(F(key2, id||w), R_{id}) \oplus (F(key1, w'||id)||F(key2, id||w'))$。最后得到搜索索引$C_{w,id} = (L, I_w, R_w, C_w, I_{id}, R_{id}, C_{id})$。

（3）Sharetoken 函数利用要分享的文件id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$lastW[id]$找到链尾w，如果w为空，说明不存在对应文件，直接返回；否则，$L = F(key1, w||id)$，$J_{id} = F(key2, id||w)$，$k_{id} = F(key2, id)$，得到分享令牌$P_{id} = (L, J_{id}, id, k_{id})$。

（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w，文件id）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$D_{id} = (L, J_{id})$，声明一个空集S，当L不为空时，执行循环操作：根据L从数据库中找到(I_{id}, R_{id}, C_{id})，将C_{id}放入集合S中，根据横链向前迭代得到$(L'||J'_{id}) = H(J_{id}, R_{id}) \oplus I_{id}$，更新$L = L', J_{id} = J'_{id}$。循环结束后将集合$S$返回给用户，用户解密$S$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w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$lastID[w]$找到链尾id，如果id为空，说明不存在包含该关键字的文件，直接返回空；否则$L = F(key1, w||id)$，$J_w = F(key2, w||id)$，将L和J_w作为搜索令牌发送给服务器。服务器声明一个空集S，执行循环操作：根据L从数据库中找到(I_w, R_w, C_w)，将C_w放入集合S中，根据纵链向前迭代得到$(L'||J'_w) = H(J_w, R_w) \oplus I_w$，更新：$L = L', J_w = J'_w$。循环结束后将集合$S$返回给用户。用户解密$S$即可得到包含关键字$w$的文件$id$。

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$O(N)$，而利用量子计算的 Grover 算法搜索的时间复杂度为 $O(\sqrt{N})$ 。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n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2^n个数据，记为N个，希望搜索的数据为M个，建立一个函数：

$$f(x)=\begin{cases}0\ (x\neq x_0)\\1\ (x=x_0)\end{cases}$$

其中x_0为搜索目标索引值。f将以一个酉矩阵U_f的形式给出，用以将$|x\rangle$转化为$|x,f(x)\rangle$：

$$|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\xrightarrow{Oracle}(-1)^{f(x)}|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$$

U_f的作用是通过改变目标态相位，标记搜索问题的解。

经过U_f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 $|0\rangle\oplus^n$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 $2|0\rangle\langle 0|-I$ 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:

$$H\oplus^n(2|0\rangle\langle 0|-I)H\oplus^n=2|\psi\rangle\langle\psi|-I$$

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$R \le \frac{\pi}{4}\sqrt{\frac{N}{M}}$，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2^n的数组，其存储数据集合记为A，下标集合记为B。集合A和B等于同一集合$\{0,1,2,3,\cdots,2^n-1\}$，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x_i，得到其对应下标y_i。然后取$n=2$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$qbit[\,i\,]$下X门的有无。如下公式所示：

$$H\oplus^2|00\rangle \to \begin{bmatrix}1\\1\\1\\1\end{bmatrix} \quad H\oplus^2|10\rangle \to \begin{bmatrix}1\\1\\-1\\-1\end{bmatrix} \qquad H\oplus^2|01\rangle \to \begin{bmatrix}1\\-1\\1\\-1\end{bmatrix} \qquad H\oplus^2|11\rangle \to \begin{bmatrix}1\\-1\\-1\\1\end{bmatrix}$$

假设现数组为顺序数组，设 Oracle 电路对应矩阵为$[a_{ij}]_{4\times4}$，则初态经过 Oracle 的作用可表示为：

$$[a_{ij}]_{4\times4}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}=\begin{bmatrix}-1\\1\\1\\1\end{bmatrix}\ [a_{ij}]_{4\times4}\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}=\begin{bmatrix}1\\1\\-1\\1\end{bmatrix}\ [a_{ij}]_{4\times4}\begin{bmatrix}1\\-1\\1\\-1\end{bmatrix}=\begin{bmatrix}1\\-1\\1\\1\end{bmatrix}\ [a_{ij}]_{4\times4}\begin{bmatrix}1\\-1\\-1\\1\end{bmatrix}=\begin{bmatrix}1\\1\\1\\-1\end{bmatrix}$$

可解得$[a_{ij}]_{4\times4}=\frac{1}{2}\begin{bmatrix}1&-1&-1&-1\\1&1&-1&1\\1&-1&1&1\\1&1&1&-1\end{bmatrix}$，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H门，Z门，CZ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i行，等于将数组中第i个数据的二

进制量子态依次经过H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$2^{\frac{n}{2}}$，公式表示如下：

$$设H^{\oplus n}|x_i\rangle = [a_1 \quad a_2 \quad \cdots \quad a_{2^n-1}]^T$$

$$[a_i]_{1\times 2^n} = [a_1 \quad -a_2 \quad -a_3 \quad -a_4 \quad \cdots \quad -a_{2^n-1}]/2^{\frac{n}{2}}$$

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$Rs = [a_{ij}]$，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$C_{w,id}$、数据库索引L及加密文件 id。通过构造数据库索引L和可搜索密文$C_{w,id}$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Jw，需要通过量子门电路根据其数据库索引L搜索其$C_{w,id}$，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。

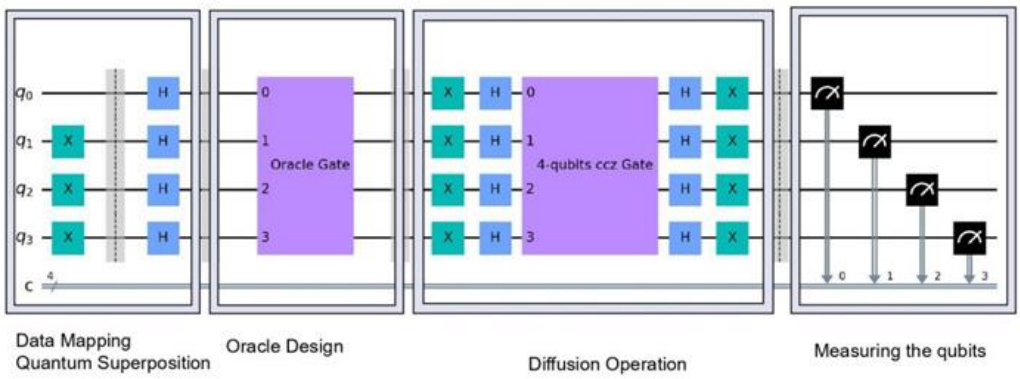

图 3-2 量子搜索算法门电路设计图

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L，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2^n（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$2^n \times 2^n$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L抽象为无序数据，可搜索密文$C_{w,id}$抽象为其索引，即当输入为L时，其目标结果应该为$C_{w,id}$，将$C_{w,id}$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：

$$Oracle_{new} = Mat_{ct2} {Mat_{ct1}}^{T}$$

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 KS^{2}E，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。

4 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。

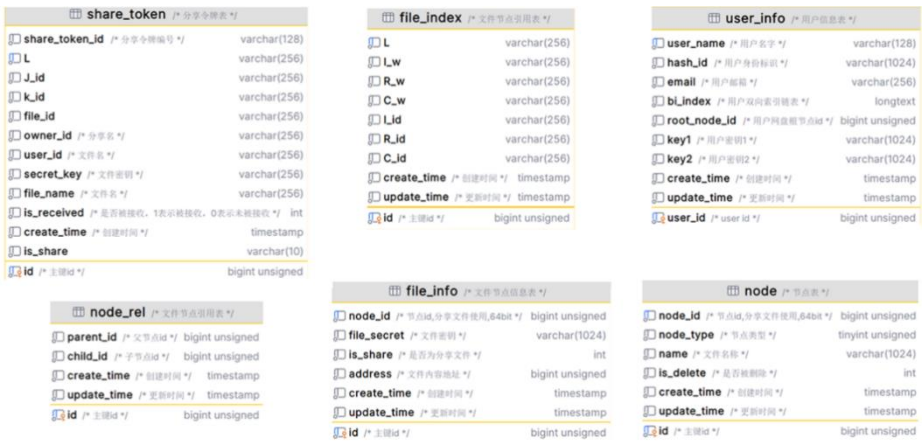

图 4-1 数据库设计概念图

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。

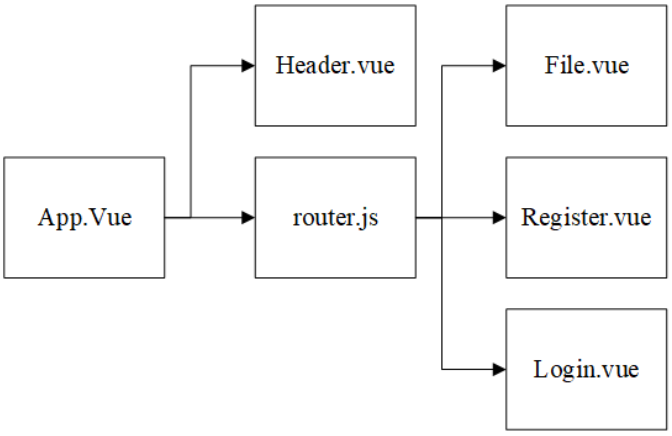

图 4-2 Web 端页面设计图

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

（2） 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：

a) 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。

b) 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。

c) 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。

d) 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。

e) 使用“npm install -g @vue/cli”指令安装 Vue 以及 Vue 脚手架。

f) 下载 WebStorm，并打开 Web 端网页源码

g) 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。

h) 在 package.json 文件中点击运行 serve 即可。

（2） MySQL 数据库部署

a) 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。

b) 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。

c) 在系统环境变量 PATH 中添加解压文件夹中 bin 文件夹的路径。

d) 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。

e) 此时可以通过账户名“root”以及密码“123456”正常访问 MySQL 数据库。

f) 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。

（3） Go 服务器部署

a) 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）。

b) 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。

c) 下载 GoLand 软件，将 Go 服务器项目源码放置在 src 目录下并打开。

d) 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。

e) 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。

f) 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。

（4） 安卓端部署

a) 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。

（5） 项目配置

a) 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。

```
open: true,
https: {
  cert: fs.readFileSync(path.join(__dirname, 'src/ssl/cert.crt')),
  key: fs.readFileSync(path.join(__dirname, 'src/ssl/cert.key'))
},
```

图 4-3 https 配置图

b) 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。

```
MySQL:
  Host: "127.0.0.1"
  Port: "3306"
  User: "root"
  Passwd: "123456"

Redis:
  Host: "localhost"
  Port: "6379"
  User: ""
  Passwd: ""

COS:
  BucketURL: "https://example-1314125883.cos.ap-nanjing.myqcloud.com"
  ServiceURL: "https://cos.ap-nanjing.myqcloud.com"
  SecretID: "AKIDIWbWLUVIuOhnly3vZ4PvMcmrneNsAKMM"
  SecretKey: "ZMAxmT0DFAvZ3QfjbGmSisdFes48zlQw"
```

图 4-4 存储服务配置图

c) 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。

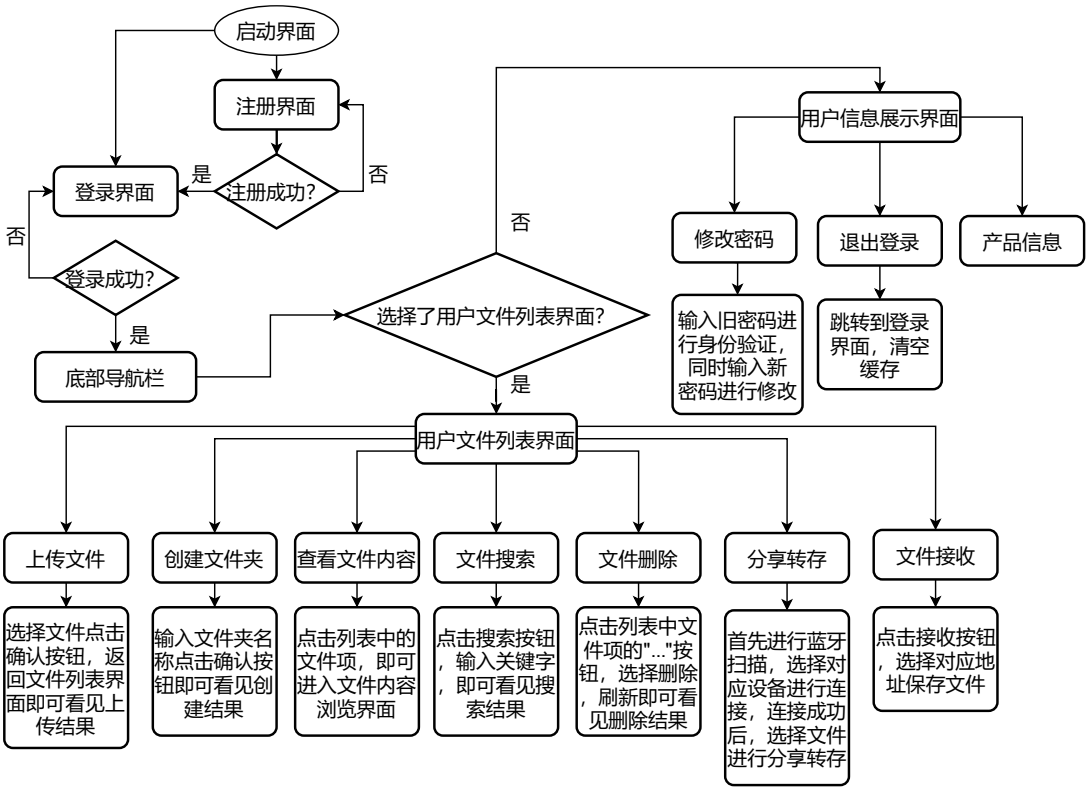

图 4-5 系统操作流程图

4.4 系统展示

4.4.1 系统 web 端展示

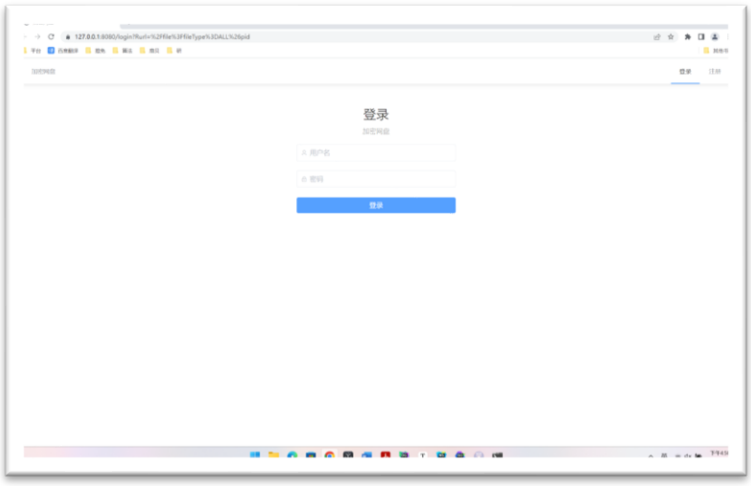

图 4-6 登录页面

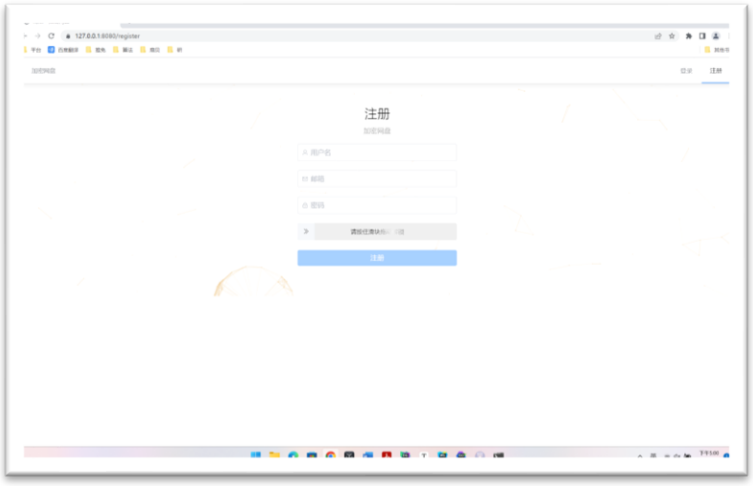

图 4-7 注册页面

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

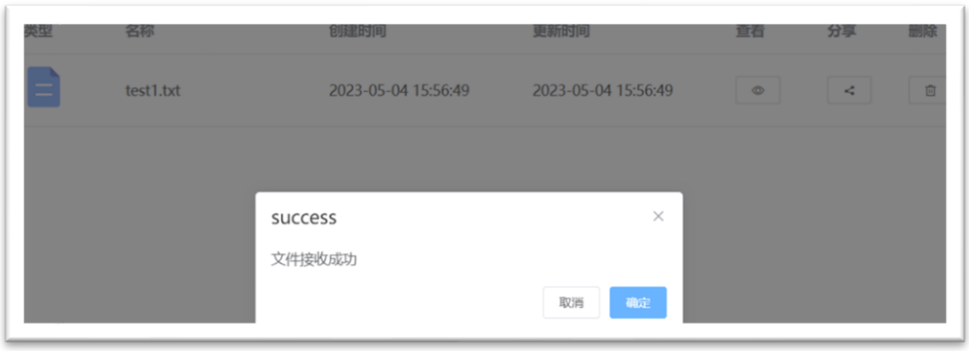

图 4-13 文件接收成功

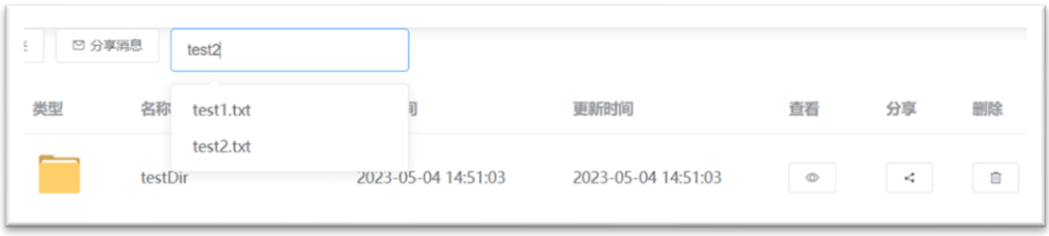

图 4-14 文件搜索

4.4.2 系统 Android 端展示

图 4-15 注册、登录页面

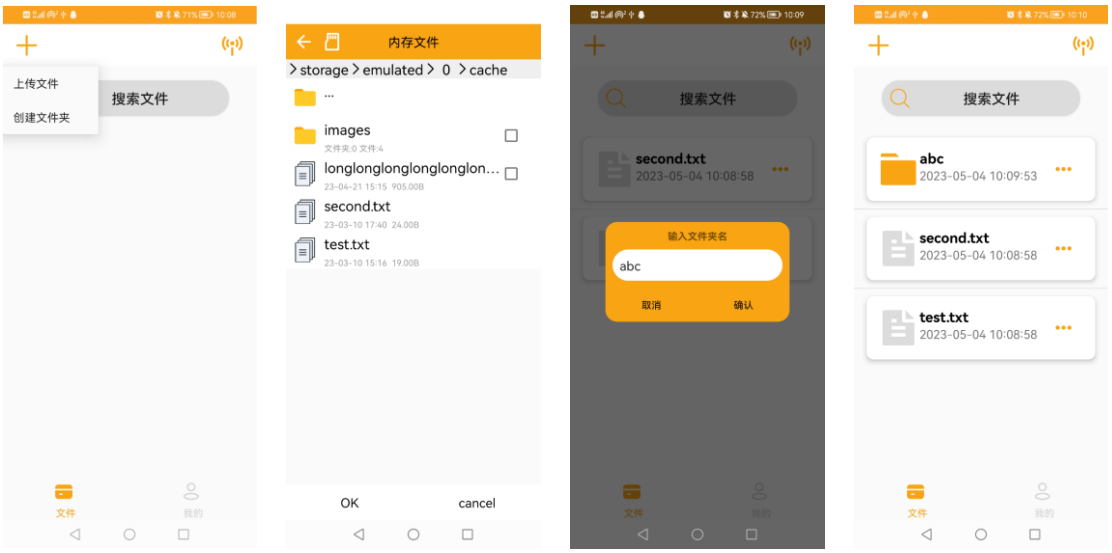

图 4-16 文件上传、创建文件夹、文件列表展示页面

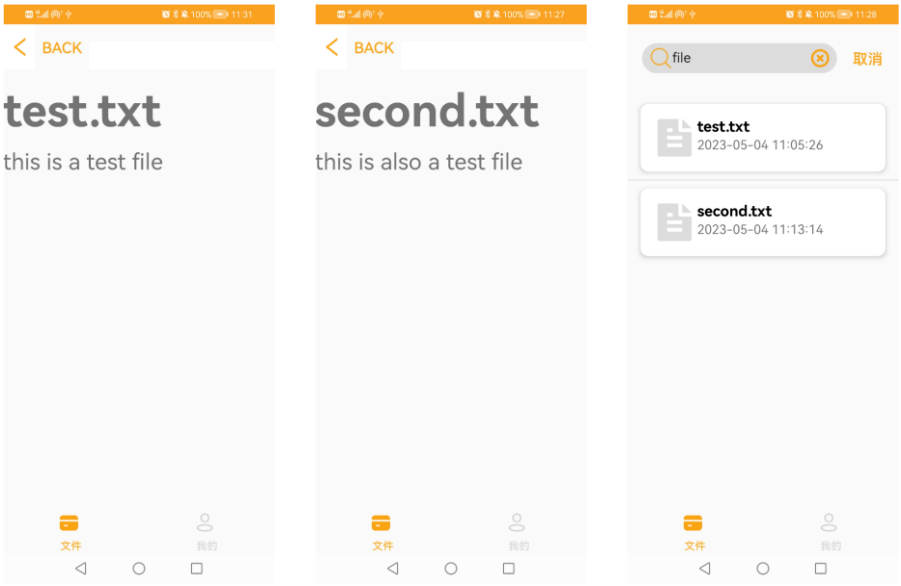

图 4-17 文件内容浏览、搜索页面

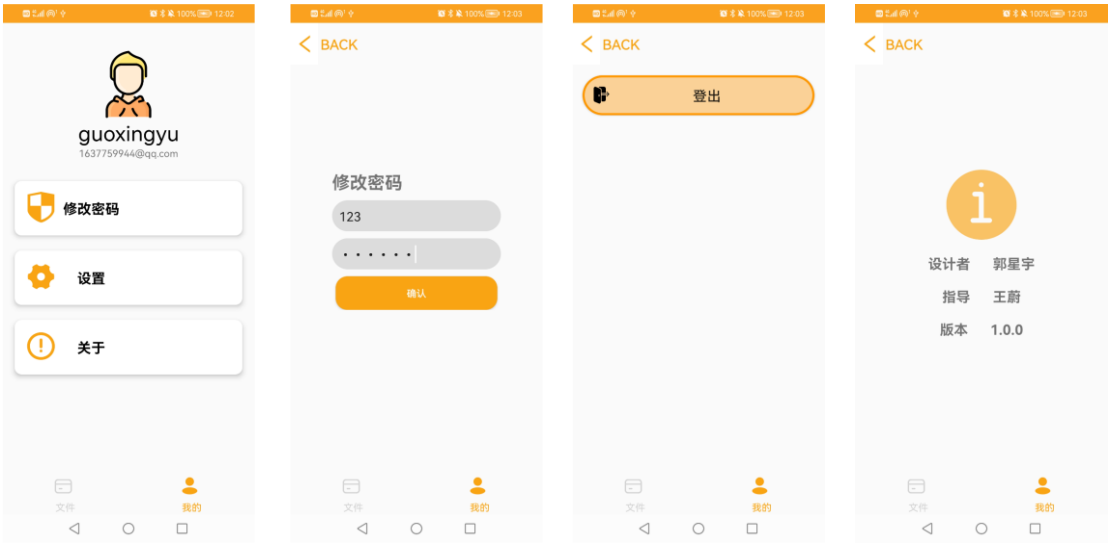

图 4-18 用户信息展示、修改密码、设置、关于页面

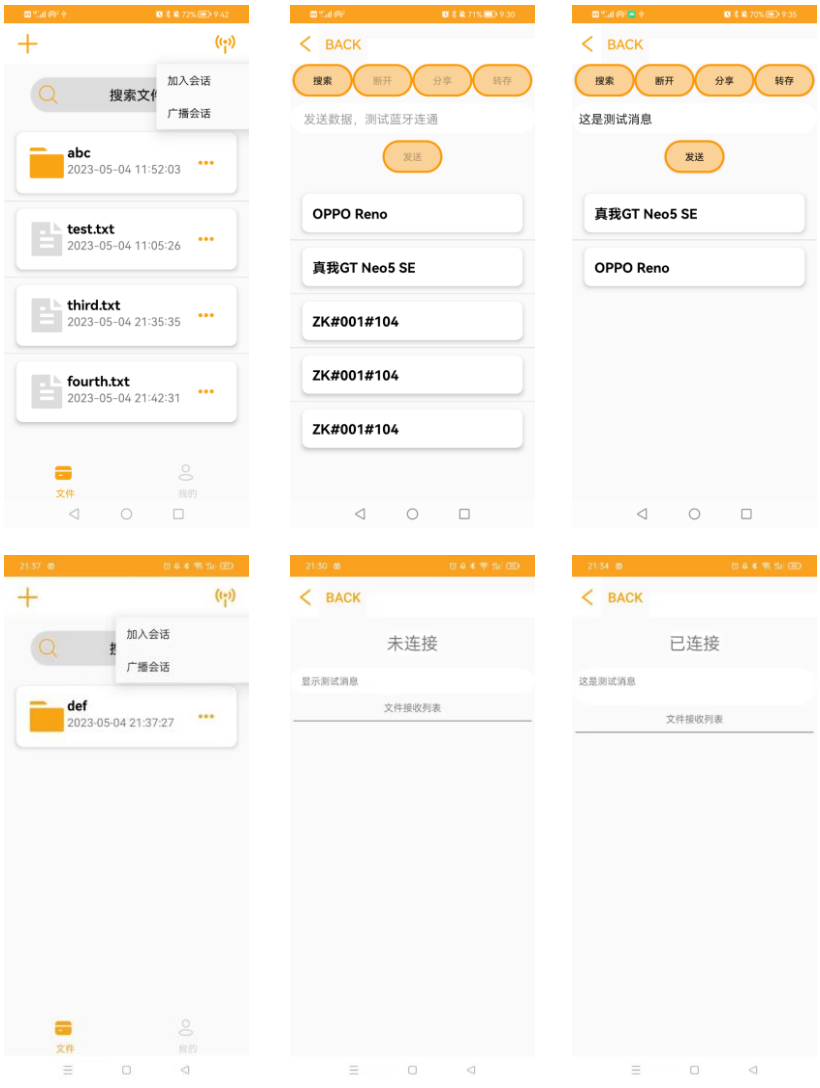

图 4-19 蓝牙连接页面

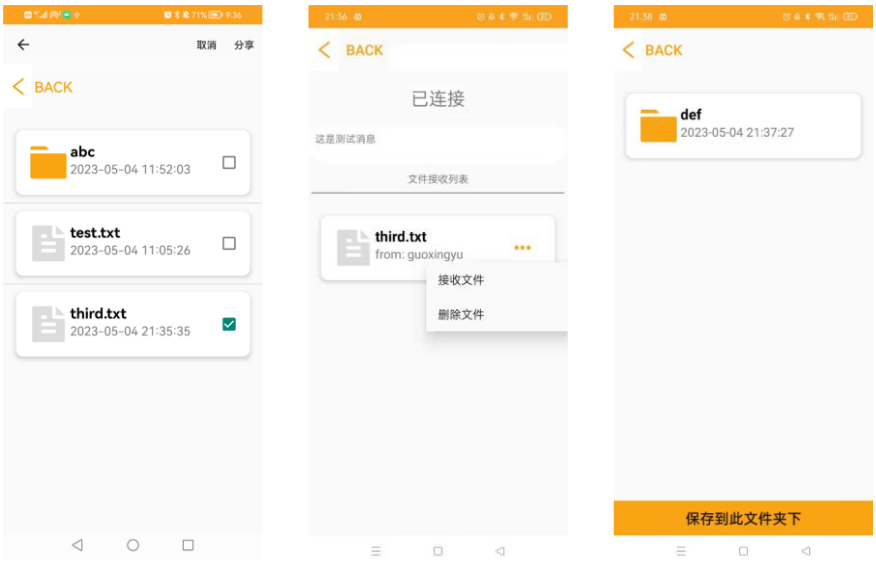

图 4-20 转存、接收页面

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。

5 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	CMIOLieqwx9v	pZ9RBR30CAvpM3Ys+eUCb1Vm	BwUECQkJCQEEA

	nS5gwUC+2fxKNOjFRXjpfcTKp6bI1gM=	N7O04LTyvv7hQeiofBbhV+z2wLY/ydGgBKa14KGlQQX2LDrS7px7xXUw+LrLww==	QcJAwcDCAUIAgQBBAUCBQkDAgQGAwE=
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
5C6JAg1KiT Z1kOWKduV WxEylwX5W uNnT/wxER CKfRSk=	zHptGBWh O0nzGkKvyd l8WPD+cPQ UHXdoVEJ9 KFOnqx0=	lbEkV3JkId Dz5Ha5YPl 4hAQ71rID XalKaSy6S QGfAaA=	GkfmN+YQPVIY1cwLopu6Z0rm0nZfXT KYKXA4ajHNwi8Q2dfnJz/SlrBQhKKH1 vx2E2kY65zCEA0CrLFfiBDR4yFELBbq 6BVCa968AiJzJW1k5nHFhyDyKvqcvje mMABtEbNPRTJMTVvP+eNpKfiGZTsR JXhAQAl721GfjgAL05c=

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L与Jw，L代表数据库索引，Jw与前一可搜索密文相关，同时设计并获取搜索函数$f(x)$，即用来判断L搜索结果是否为对应的$C_{w,id}$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$C_{w,id}$进行搜索，即通过L，获取$C_{w,id}$。

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L为 0000，希望搜索到的$C_{w,id}$

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L和$C_{w,id}$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L和$C_{w,id}$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$C_{w,id}$。

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。

```
eighthalf@eighthalf-virtual-machine ~/Quantum_program/Diana$ python3 CloseBetaQC.py
欢迎进入量子比特搜索系统！
请选择本地模拟或真机运行（输入0为ClosedBetaQC，或1为本地模拟）：0
请输入要搜索的比特数：4
请继续输入要搜索的加密数据库索引：（字符串输入，输入Q停止）：
o0jn0VfPiytFqKTGcyulci64TX74HRyL+ElS5l4PvkI=
加密数据库索引的哈希值： 5
可搜索密文： 1
请继续输入要搜索的加密数据库索引：（字符串输入，输入Q停止）：
ThnNx6gK+07LBwUzGmxcYXBK5LikPRGW7/UrC4gjL+s=
加密数据库索引的哈希值： 13
可搜索密文： 2
请继续输入要搜索的加密数据库索引：（字符串输入，输入Q停止）：
Q
现在您可以开始进行搜索！
请输入加密数据库索引ThnNx6gK+07LBwUzGmxcYXBK5LikPRGW7/UrC4gjL+s=
11 01
程序已在量子云平台上运行
['1', '1']
查询实验结果请等待：4.20秒
查询实验结果请等待：1.58秒
搜索结果的高两位为： 00
['0', '1']
查询实验结果请等待：4.21秒
查询实验结果请等待：1.43秒
搜索结果的低两位为： 10
文件id的可搜索密文（二进制）为： 0010
文件id的可搜索密文（十进制）为： 2
请输入加密数据库索引Q
感谢您的使用！
```

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L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L）,将其转化为二进制的 bit 字符串，并应用 Grover 算法搜索$C_{w,id}$，具体来说，使用数据库索引（即L）在数据库中搜索可搜索密文（即$C_{w,id}$），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。

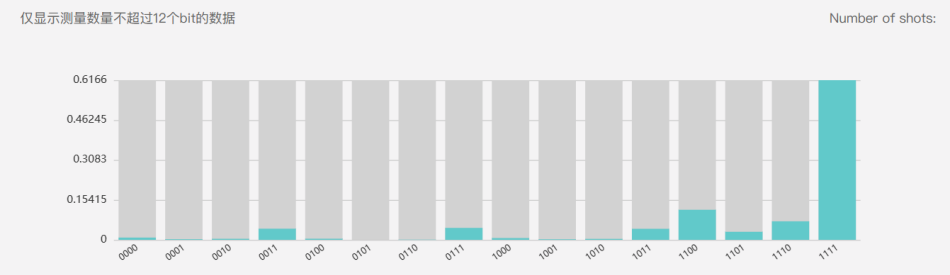

图 5-2 量子计算云平台查看 4 比特量子程序运行结果

图 5-2 为 4 比特实现密态云盘搜索量子程序运行结果。

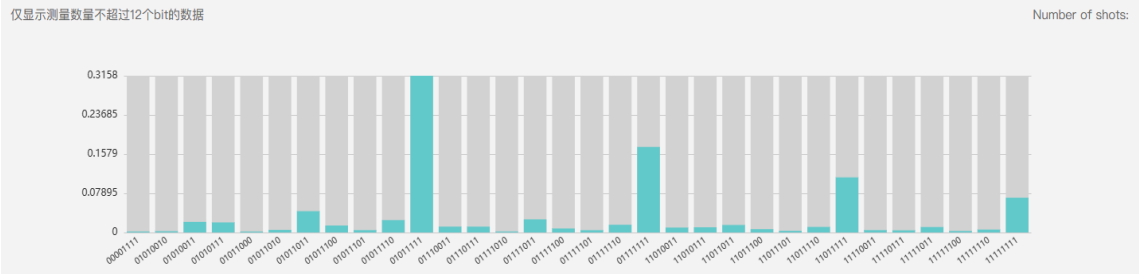

图 5-3 量子计算云平台查看 8 比特量子程序运行结果

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。

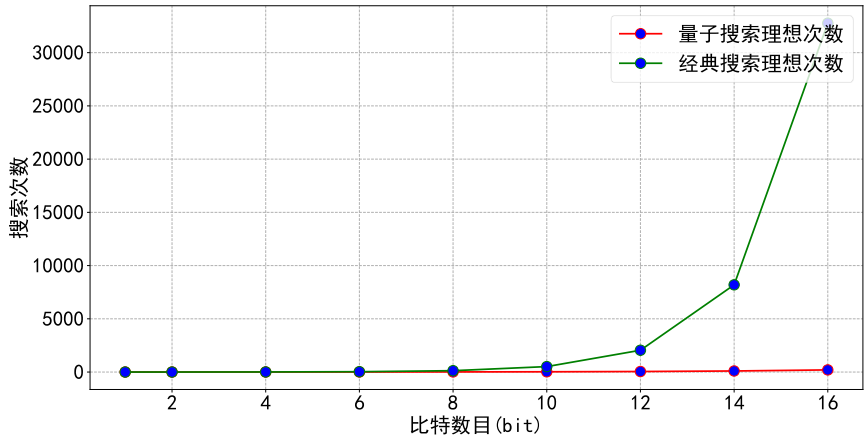

图 5-4 经典搜索与量子搜索之间理想搜索次数的差异

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。

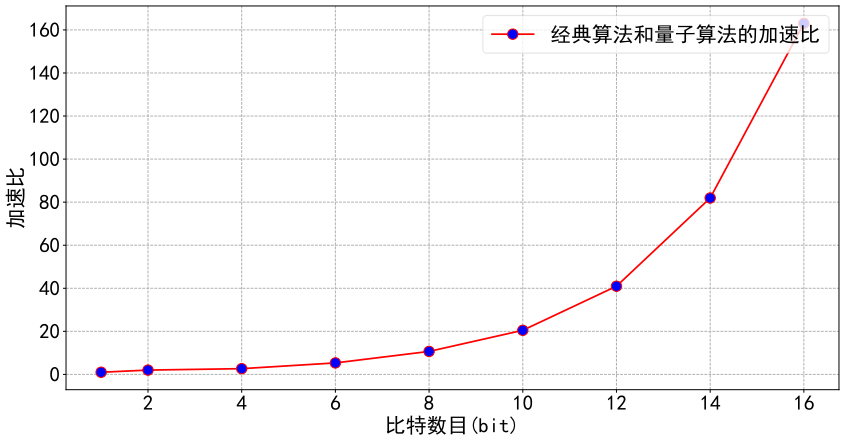

图 5-5 量子算法与经典搜索算法的加速比

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$O(logN)$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$O(N)$的内存来存储搜索空间中的所有项，因此其空间复杂度为$O(N)$。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。

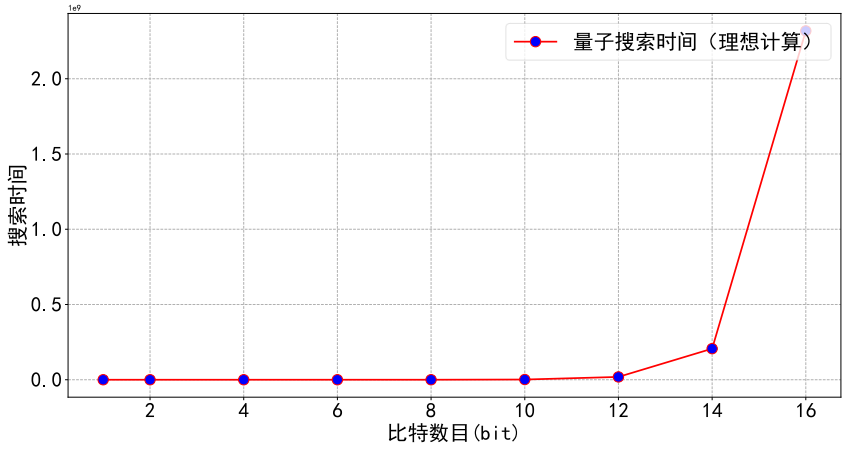

图 5-6 不同比特搜索时理想的搜索时间

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。

6 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$O(\sqrt{N})$。

6.3 方案创新性分析

（1）算法创新

使用可搜索加密模型 KS^2E，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$O(N)$。

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$O(\sqrt{N})$，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，n位量子比特能加速2^{n-1}数量级的索引数目（对于一步$O(N)$时间复杂度的检索，能将其

至少优化到$O(\sqrt{N})$），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。

7 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$O(N)$降低到 $O(\sqrt{N})$，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.