## 1研究背景

近年来，随着网络不断发展，人们工作与生活产生的数据也呈现指数形式的增长，海量数据存储与处理的需求日益增长，云存储技术应运而生。为了保证数据安全和用户隐私，用户一些敏感数据通常需要先加密再以密文状态上传。这种方法在云端不完全可信的情况下可以对用户数据进行保密，但是加密存储的方法也让用户无法便捷的在云端对密文进行查询。

传统的密文存储服务中，由于云端没有相应的检索功能，不能根据用户的需求查找数据，只能将用户的所有密文数据全部返还给用户，让用户在本地进行解密操作后再进行检索操作。显然这种交互方式不仅占用了大量的网络开销，而且给用户带来了额外的计算开销。因此如何在用户提交检索需求时，云端借助自身强大的计算能力实现高效检索并准确的返回数据是当今云存储安全存储的重要研究方向之一。

## 2可搜索加密技术

可搜索加密SE11（Searchable Encryption）是近年来发展起来的一种支持用户在密文上针对关键词进行查找的技术。它能够为用户节省大量的网络和计算开销，借助云端的强大计算能力准备的完成检索并返回结果，而用户在使用过程中不会向云端暴露任何隐私。

可搜索加密的方案最早由Song等在2000年提出1。可搜索加密的基本框架如图1.1，其主要过程分为四步。

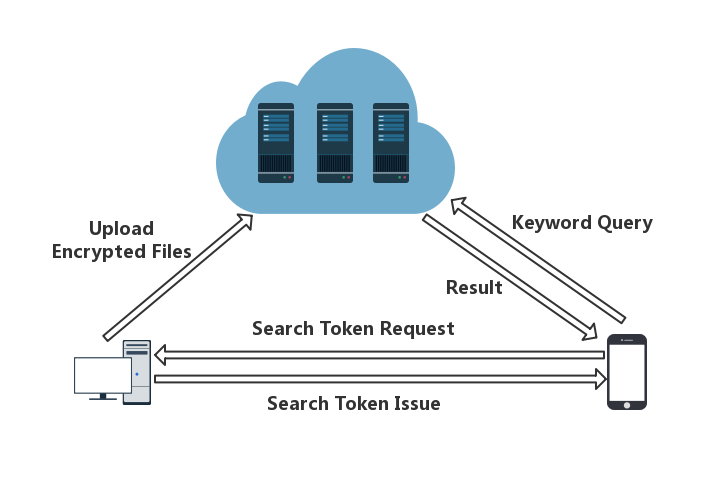
1. 数据加密：数据所有者在本地对数据进行加密，然后将密文上传至云端。
2. 陷门生成：用户结合密钥和关键词生成对应的陷门，将陷门发送给云端。
3. 检索密文：云端借助用户提交的陷门和自身维持的索引表进行检索，返回包含陷门关键词的数据。
4. 文件解密：用户使用自身密钥对云端返回的数据进行解密。

图1.1

目前，按照可搜索加密采用的加密方式可以分为对称可搜索对称加密（SSE, symmetric searchable encryption）和非对称可搜索加密（ASE, asymmetric searchable encryption）两种类型，这两种类型的加密方式来源于不同的现实问题，以及在后续解决不同的需求问题。

#### 2.1 对称可搜索加密

对称可搜索加密2的构造通常基于伪随机伪随机函数，具有计算开销小、算法简单、速度快的特点。除了加解密过程采用相同的密钥之外，其陷门的生成也需要密钥的参与。安全的SSE设计模型可以确保搜索关键词的保密性，能够抵御外部攻击和云端的内部信息泄露。

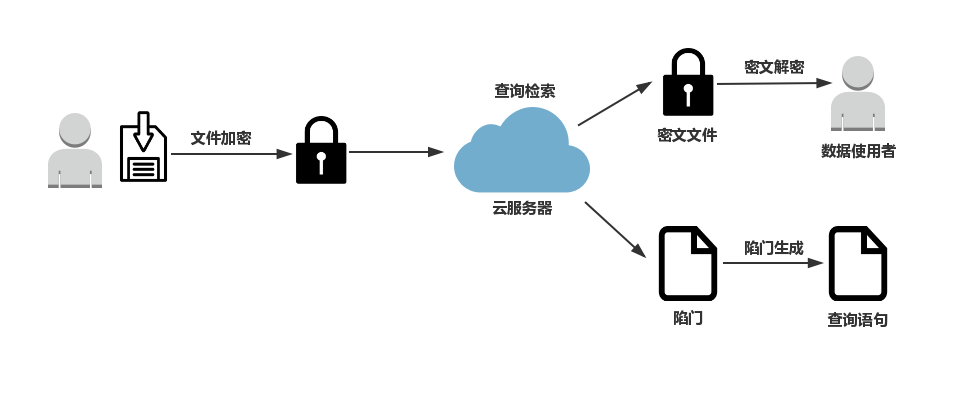
定义1（对称可搜索加密）定义在字典Δ={ W1,W2,…,Wd}上的对称可搜索加密可以描述为五元组。

*SSE=（KeyGen,Encrypt,Trapdoor,Search,Decrypt）*

其中，

1. *K=KeyGen(λ)*：λ是安全参数，该算法根据安全参数生成加密密钥K；
2. *(I,C)=Encrypt(K,D)*：输入对称密钥K和明文文件集D=(D1,D2,…,Dn)，Di∈2Δ，输出索引I和密文文件集C=（C1,C2,…,Cn）,一些不需要生成索引的方案中I∈ ；
3. *Tw=Trapdoor(K,W)*：输入对称密钥K和关键词W，输出关键词陷门TW；
4. *D(w)=Search(I,Tw)*：输入索引I和关键词陷门Tw，输出包含关键词W的文件集合D(w);
5. *Di=Decrypt(K,C)*：输入对称密钥K和密文文件C，输出相应明文文件Di。

基于上述给出的定义，对称可搜索加密的执行流程大致如下：加密过程中，用户执行KeyGen方法生成对称密钥K，使用密钥K加密文件集并上传至服务端。检索过程中，用户借助对称密钥K执行Trapdoor方法生成特定关键词W的陷门Tw，服务器使用Tw检索上传的密文文件集，返回包含特定关键词W的密文文件，用户最后用密钥K解密返回的密文文件，得到目标文件。



#### 2.2非对称可搜索加密

在对称可搜索加密中，由于对称密码体制本身的限制，通常在多用户的环境下，用户之间需要事先构造安全信道传递私密信息，进而才能事先数据的加密上传与检索。相比较之下，非对称可搜索加密3无需事先建立安全信道即可实现安全的检索功能。

定义2非对称可搜索加密算法可以描述为*PEKS=(KeyGen,Encrypt,Trapdoor,Test)*

1)*(pk,sk)=KeyGen(λ)*:λ是安全系数，该算法根据安全系数生成公钥pk和私钥sk。

2）*Cw=Encrypt(pk,W)*:利用生成的公钥pk和加密文件的关键词W，生成关键词密文Cw。

3）*Tw=Trapdoor(sk,W)*:利用生成的私钥sk和用户输入的关键词W，生成关键词W的陷门Tw。

4）*b=Test(pk,Cw,Tw)*:根据生成的公钥pk、关键词W的陷门Tw和关键词密文Cw，计算匹配相似度，输出判定值b∈｛0，1｝。

2.3可搜索加密模型的发展

随着网络技术的不断发展，云存储技术4在日常环境中的应用范围也越来越广，随之而来的就是用户对存储安全的需求。可搜索加密作为一项支持用户在密文上检索的重要技术，吸引了大量的学者进行研究和改进。不同的学者在加密搜索领域上提出了多种可搜索加密模型，这些模型在效率和安全特性上都各不相同。对于不同的加密搜索模型我们通常从三个方面进行评估，分别是查询效率、动态更新和语义安全。后续学者也是从这三个方面在之前的可搜索加密模型上进行优化。

2000年，Song等人第一次提出可搜索加密的解决方案1（如图1.2），该方案支持静态文件上的加密搜索。然而该方案实际运用中存在着很多缺陷，例如搜索性能不佳、数据库与数据规模呈线性关系以及无法证明的安全性。后续人们遵循这一设计模型，不断改善Song所提出的解决方案。



图1.2

Goh等人提出了使用布隆过滤器为每个文件构造一个索引5，该方法将文件包含的关键字映射到码字存储的索引中，通过布隆过滤器的运算，就能够判断文件中是否包含特定的关键字，从而将搜索时的计算代价降低到与密文文件数成正比。但是引入布隆过滤器带来了查询的不确定性以及在检索过程中会暴露集合信息的问题，给整体模型带来了安全隐患。在2006年，由Curtmola等提出新的加密搜索模型6，该模型不仅分析了可搜索加密模型的安全性而且在检索性能上做了大量的优化工作。但是该模型的缺陷在于搜索时生成陷门的长度和整体文档长度相关以至于陷门长度过长。后续一些人也在准确关键字搜索基础上引入了模糊搜索7来增强服务端对关键字的检索能力。

原始的可搜索加密模型并不支持文件的在线删除和添加，然而现实情况中在密文检索之外有时还需要对密文本身进行增删操作。因此在密文检索基础之上，Kamara首先提出了支持动态更新的可搜索对称加密8（DSSE），并且对DSSE模型做了详细的安全分析。新的DSSE模型不仅可以提供密文搜索功能，而且还支持对文件的动态更新。但是原始的DSSE模型在交互过程中会泄露大量密文相关信息，整体模型的安全性不高。后续Kamara针对交互过程泄露隐私信息等问题做了优化，借助红黑二叉树的结构提出了新的DSSE方案9。但是作为减少隐私信息泄露的代价，新的DSSE方案增加了可搜索密文的长度并降低了检索效率。之后，Hahn提出了新的DSSE方案解决现有方案存在的问题10，该方案在检索和更新时采用新的索引结构代替构建密文结构，查询效率也进一步提升。

## 二 论文主要的研究内容

## 2.1 当今网盘系统现状

随着网络技术的兴起，用户日常生活产生的数据总量也大大提升。依托云计算平台的云存储技术应运而生。伴随用户将大量数据托管于云端服务商，对用户数据的隐私性和安全性的充分保证对当今服务商是一个巨大的挑战。我们认为当前的网盘应用服务在隐私保护方面不够完善，通常情况下，用户数据在云端以明文形式存储，即便部分网盘应用服务提供了加密存储功能，用户的数据的加密过程通常在云端完成，密钥亦保存在中心化云服务提供商处。此外，为了提升存储效率，多数云服务提供商都采用了去冗余技术，该技术的基本原理是对用户上传的文件进行哈希运算，哈希值相同的文件即被认为是同样的文件，可以指向同一物理存储位置。因此云服务提供商很容易窥探用户所存储的数据或对用户的数据进行解密。同时，用户对数据的存取操作亦完全被云服务提供商所知，因为用户对云端数据进行检索、查询的操作是完全暴露给云服务提供商的。在大数据分析方法趋于完善的今天，云服务提供商会有意识地搜集用户的各种数据并进行分析，进而实现其商业目的，或协助政府进行内容审查。Facebook公司在2018年被爆出泄露用户隐私数据给其他公司分析使用11，Facebook公司因此被推上舆论的风口浪尖。

## 2.2威胁模型

普通的云存储服务交互模式如图1.3。



图1.3

通过分析用户与服务端的交互方式，我们得出用户数据的安全和隐私风险主要来自三个个方面，首先是用户数据存储在服务端时的风险。用户通过网络将数据上传到服务端，服务端有可能在没有获得用户的允许下对用户的数据进行搜集和分析。此外，服务端也有可能被其他黑客通过非法手段入侵导致用户数据的泄露。其次，数据经过不可靠的网络信道传输也带来了额外的安全风险。现实生活中的网络由许多的路由器以及网络终端设备互联。一些别有用心的人通过向网络中的部分路由器植入特定的病毒来劫持用户传输的数据，又或者他人在数据传输过程中劫持并修改原有数据内容，这两种情况都给用户的数据安全带来了极大的威胁。最后，用户在与服务端交互过程中存在隐私信息泄露的可能。用户与服务端在传输数据之外也会有其它的交互过程，例如用户想借助服务端的强大计算能力对上传的数据进行基于关键字的检索。然而一般情况下用户在检索过程中会将关键字信息以明文方式传送给服务端，而服务端除了根据用户提交的关键字信息进行检索之外还会对用户提交的关键字信息进行收集，导致用户的行为隐私泄露。

本文解决的

## 2.3基于可搜索加密的新型云存储系统

综上所述，我们认为现有的网盘应用服务主要存在两方面风险，即数据安全风险和用户隐私风险。在本项目的研究中，我们希望解决这两项痛点问题，设计出安全的隐私可控的新型云存储系统。在数据安全方面，我们计划引入加密存储技术并设计新型数据可靠传输模块，从而分别应对数据在云端存储和数据在传输过程中的安全问题。在隐私保护方面，我们计划引入可搜索加密技术以保证用户在数据检索过程中不会泄露行为数据，同时，我们还计划使用去中心化的密钥管理机制，使得数据访问权限完全控制在用户手中。为了使得系统具有更好的现实意义，我们将会实现原型系统，并对相关算法进行优化为#####

为了使系统效能达到实用水平，我们引入了动态累加器方法对加密搜索过程进行了优化，并进行了仿真实验，对比国际上已有文献中的加密搜索算法，本项工作中的新方法显著提升了加密搜索速度。

## 2.4动态加密搜索模型的优化

在现有的动态加密搜索模型上，学者围绕着不同方向对先前的可搜索加密模型做了更进一步的优化，例如在索引表中加上正向索引结构或者改进关键词陷门的生成过程进而支持多关键字检索，这些改进过程在原有可搜索加密模型的效率和安全方面做了很大提升。通过查阅近些年的可搜索加密技术研究进展，我们发现新的模型在具体的关键字查询过程中仍然采用类似反向索引的查询结构，考虑到查询性能对整体加密搜索模型带来巨大的效率提升，我们提出引入密码学中的动态累加器来优化查询结构的方案。通过引入动态累加器而建立新的索引表，可以在查询过程中减少在索引表中关键字的比对次数，进而提升可搜索加密模型在查询过程时的性能。(移到后面，重新组织)

密码学上的累加器最早是由Camenisch和Lysyanskaya首先提出12，其作用是对于给定的集合，一个密码学上的累加器可以将集合内一组值累加成一个值，并且在不暴露集合内部成员的情况下使输入的任意一个值证明自己被累加到这个累加值中。而动态累加器可以在O（1）时间复杂度内增加和删减累加值中的元素。2008年，Peishun Wang等人对动态累加器做了正式的定义13。借助动态累加器，我们在传统的密文-关键字索引表之外添加了新的索引表。基于新的索引表，我们在关键字的查询过程中可以通过新的索引表快速并精确的定位到关键字所在的文件位置，继而再进入相应文件的关键字集合中进行详细比对，确定具体的关键词信息。新的索引表在查询过程中可以减少与其他关键词的比对次数，从整体上优化了查询结构，进而提升了查询时的效率。

## 2.5已进行科研工作基础

#### 2.5.1基于可搜索加密网盘系统的设计与实现

###### 2.5.1.1可搜索加密网盘系统的设计



###### 2.5.1.2可搜索加密网盘系统的实现

1. 文件安全存储

考虑到文件存储在文件在服务端的安全，我们必须对待上传的数据进行预加密处理。常见的数据的加密方式包括对称加密和非对称加密。对称加密方式中加解密使用同样的密钥，加解密速度较快。非对称加密中密钥分为公钥和私钥，其中公钥用作加密数据，私钥用作解密数据。非对称加密适用于不可靠的环境中。考虑到加解密的效率，大文件的加解密主要采用对称加密的方式。因此在本研究中我们采用了AES标准作为文件加解密方式。

1. 文件可靠传输

当今互联网使用非常广泛，人们可以很方便的通过不同终端接入互联网收发信息。但是互联网就其安全性而言是无法保证的。数据在不可靠的网络环境中传输可能存在被窃取和篡改的风险。为了确保数据在不可靠的信道中安全传输，新的云存储系统中的传输模块在普通的数据传输之上增加了可靠校验和加密传输机制。

在传输模块中，我们采用UDP作为传输协议来传输数据。相比较TCP传输的方式，UDP传输数据速度更具有可控性，传输时也不需要建立连接以及传输过程没有消息边界限制。同时UDP是不可靠的传输协议，因此我们需要通过对UDP数据报传输内容上封装上我们的标识，以此实现数据在传输过程保持安全性和可靠性。

1. 简单实用tcp做传输为什么不好（tcp阻塞式的，并且有流控，对复杂网络环境适应较差）
2. 我们实现了基于udp的可靠传输（在应用层实现了可靠传输、能够较好地支持复杂网络环境（如果丢包或者是乱序到达也能够保持保持较高传输速度）、数据包payload（数据载荷）是加密的）
3. 可搜索加密技术引入

可搜索加密技术主要用来解决服务端在密文上进行检索的问题。在普通的云存储服务中，考虑到文件存储的安全性，用户会对待存储的数据进行预加密后再上传，同时用户又希望云端在不解密文件的情况下针对用户请求的关键字进行密文检索，而传统的检索方法在加密文件上无法工作，为了弥补这一缺陷，我们在新型安全云存储系统中引入了可搜索加密技术。通过引入可搜索加密技术，新的云存储系统可以在不暴露用户数据内容的情况下完成对密文的搜索。同时，可搜索加密技术亦能保证用户在提交密文检索请求时用户输入的关键词信息不会泄露给服务端，进一步保护了用户的隐私信息。

1. 可搜索加密算法的优化

可搜索加密技术的发展过程已经在上面有所介绍，但是纵观可搜索加密的发展历史，我们看出可搜索加密的研究主要聚焦于功能和安全两大方面。在关键词查询这一方面，现有的主流可搜索加密模型大都采用了反向索引的设计，反向索引可以在重复查询的情况下大幅度优化查询效率，但是在平均情况和最坏情况下表现仍待改进，而一个好的可搜索加密模型能否大规模应用关键就在于其运行效率。而加密搜索模型其核心功能就是其查询功能，高效的查询算法不仅能节省服务器资源并降低运营成本，同时能够给用户带来更加好的使用体验。我们在研究过程中针对当前流行的加密搜索模型中的查询算法进行优化，经过优化后，理论上加密搜索算法的最坏时间复杂度从O(m\*n)降低到了O(m+n)。

一个密码学上的累加器是一个[单向的](https://zh.wikipedia.org/wiki/%E5%96%AE%E5%90%91%E5%87%BD%E6%95%B8)[隶属函数](https://zh.wikipedia.org/wiki/%E9%9A%B8%E5%B1%AC%E5%87%BD%E6%95%B8)。它可用于识别一个元素是否为一个[集合](https://zh.wikipedia.org/wiki/%E9%9B%86%E5%90%88)的成员，且不会在过程中暴露[集合](https://zh.wikipedia.org/wiki/%E9%9B%86%E5%90%88)中的成员。累加器可以极大加快对集合内元素进行批量证明。在我们的改进过程中，借助累加器这一性质我们构建一个新的索引结构，在查询过程中通过这个新的索引结构可以快速的确定关键字所在的范围，提升查找效率。

当前主流的可搜索加密模型的原型是DSSE模型，DSSE模型中采用了反向索引来加速对关键字的查找过程。具体的查找过程中，如果反向索引中存在待查询的关键字则可以快速返回查找结果，但是在反向索引未命中的情况下，需要进入原始的文件-关键词索引表进行一一比对，而这种情况下的计算开销是巨大的。考虑到这种情况下的计算开销问题，我们通过密码学上的累加器来减少普通情况下的关键词比对计算开销。具体的做法为，通过引入动态累加器，查找过程中如果反向索引没有命中则进入我们建立的累加值索引表进行查找以确定关键词所在的范围。借助动态累加器的特性一次进行多个元素验证的这一特性，我们可以减少很多不必要的元素比对过程，提升了查询的效率。

时间复杂度分析：理论上来说，如果文件总数为m，单个文件关键字数量平均为n，之前算法（哪一个？）最坏情况下的理论时间复杂度

O（m\*n）

而新的改进算法在最坏情况下的理论时间复杂度为

O(m+n)。

4.3新的算法改进

借助动态累加器，我们添加以下算法

1. *(AC,Info)=Accumulator(Values)*

算法的输入Values的数据结构是一个列表，内容为由单个文件所有关键词组成的陷门集合。该算法的返回值AC是一个陷门集合的累加值,Info则是累加值相关辅助信息，用于后续证明。

1. *Result=FindValue(AC,Token,Info)*

算法的输入是AC和Token，Token是由单个关键词生成的单个陷门，AC为关键词集合通过累加器产生的累加值,Info为该累加值伴随的辅助信息。该函数判断给定集合的属性值判断当前token值是否在集合内，返回值Result是一个布尔类型的数值。

1. *(AC’,Info’)=AddEle(Elements,AC,Info)*

算法的输入是待添加关键词陷门集合Elements,原有累加值AC和原有辅助信息Info。算法的作用是向现有累加值中增加新的元素。算法返回值为新的累加值AC’和新的累加值辅助信息Info’。

1. *(AC’,Info’)=DelEle(Elements,AC,Info)*

算法的输入是待删除关键词陷门集合Elements，原有累加值AC和原有辅助信息Info。算法

的作用是在之前累加值中删除指定的元素。算法返回值为更新后的累加值AC’和更新后的辅助信息Info’。

## 3仿真实验

可搜索加密模型在检索关键词时会带来额外的计算开销，因此相比较普通的关键词查找，可搜索加密在检索时也会更加消耗时间

普通检索和可搜索加密查询时间对比

1000个文件，每个文件随机5个关键词，其中横轴为查询关键词个数，纵轴为搜索方式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 500 | 1000 | 3000 | 5000 |
| 普通检索 (s) | 0.12 | 0.17 | 0.27 | 0.46 |
| 加密搜索 (s) | 30.55 | 67.65 | 195.45 | 331.63 |

3000个文件，每个文件随机5个关键词，其中横轴为查询关键词个数，纵轴为搜索方式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 500 | 1000 | 3000 | 5000 |
| 普通检索 (s) | 0.19 | 0.29 | 0.85 | 1.36 |
| 加密搜索 (s) | 108.23 | 217.91 | 654.35 | 1030.71 |

5000个文件，每个文件随机5个关键词，其中横轴为查询关键词个数，纵轴为搜索方式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 500 | 1000 | 3000 | 5000 |
| 普通检索 (s) | 0.39 | 0.42 | 1.25 | 2.07 |
| 加密搜索 (s) | 160.45 | 323.73 | 972.90 | 1465.80 |

不同可搜索加密方法查询时间比对

1000个文件，每个文件随机5个关键词，其中横轴为查询关键词个数，纵轴为搜索方式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 500 | 1000 | 3000 | 5000 |
| 加密搜索 (s) | 30.55 | 67.65 | 195.45 | 331.63 |
| 反向索引 (s) | 29.23 | 60.99 | 179.93 | 294.81 |
| 区域查询 (s) | 6.58 | 12.57 | 36.03 | 58.1 |

3000个文件，每个文件随机5个关键词，其中横轴为查询关键词个数，纵轴为搜索方式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 500 | 1000 | 3000 | 5000 |
| 加密搜索 (s) | 108.23 | 217.91 | 654.35 | 1030.71 |
| 反向索引 (s) | 106.94 | 201.97 | 590.98 | 860.88 |
| 区域查询 (s) | 21.28 | 41.76 | 120.13 | 173.95 |

5000个文件，每个文件随机5个关键词，其中横轴为查询关键词个数，纵轴为搜索方式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 500 | 1000 | 3000 | 5000 |
| 加密搜索 (s) | 160.45 | 323.73 | 972.90 | 1465.80 |
| 反向索引 (s) | 158.96 | 307.90 | 854.93 | 1319.55 |
| 区域查询 (s) | 32.07 | 62.63 | 158.15 | 263.31 |

通过以上初步实验结果我们可以得出以下结论

1. 无加密情况下的关键词检索相比较基本的加密检索在速度上快了很多，可以看出可搜索加密牺牲了检索效率从而保证查询的安全性和隐私性。
2. 反向索引的引入在关键词重复查询的过程中可以加速关键词的查找过程，但是在检索关键词重复概率较小的情况下反向索引的加速效果不是十分明显。
3. 新的区域搜索算法在实际的模拟实验中在运行时间上基本上基本上相当于原有运行时间的五分之一，而之所以五分之一的原因在于我们在生成文件关键词时指定关键词个数为5。按照之前的理论计算，如果文件总数为m，单个文件关键字数量平均为n，最坏情况下基于反向索引的查询算法时间复杂度O(m\*n）而新的区域搜索算法在最坏情况下的理论时间复杂度为O(m+n)。实验结果基本符合我们的预期。

## 4实验环境

机器配置如图1.4

|  |  |  |  |
| --- | --- | --- | --- |
| 配置信息 | 操作系统 | 数量 | 服务器/客户端 |
| CPU：英特尔i7-8700  内存：16GB  显卡：GTX-1060-3G | Windows 10 | 2 | 服务器 |
| CPU：Intel i5-8250U@??GHz  内存：8GB DDR4  显卡：MX150 | Windows10 p | 1 | 服务器 |
| CPU：AMD 1700  内存：16GB DDR4  显卡：nvidia GeForce GTX 1050ti | Win10 | 1 | 客户端 |



## 5、开发工具及开源框架

visual studio 2017、Hadoop、SQL Server、Redis、Docker、Wireshark、Chrome。

## 6、工作安排

本课题研究时间为2019年3月到2020年3月。工作进程安排为三个阶段：

前期2019 年3月-4月：

文献阅读、文献整理及开题报告撰写

中期2019年4月—2019年12月：

5月-6月，实现加密搜索模块，完成单元测试，改进搜索算法；

6月－9月，完成加密搜索网盘原型设计，进行集成测试；

9月－11月，对系统进行优化和改进；

11月－12月，开展实验，收集数据并与文献中的相类似的工作进行比较，总结新型系统的优缺点。

后期2019年12月－2020年4月：

整理实验并完成论文初稿；

论文修改；

论文定稿。

参考文献

1. Song XD, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: Proc. of the IEEE Symp. on Security and Privacy. IEEE Press, 2000. 44-55. [doi: 10.1109/SECPRI.2000.848445]
2. LI J W, JIA C F, LIU Z L, et al. Survey on the searchable encryption[J]. Journal of Software, 2015,26(1):109-128.
3. DAN B, CRESCENZO G D, OSTROVSKY R, et al. Public key encryption with keyword search[M]//Advances in Cryptology - Eurocrypt. Berlin Heidelberg: Springer, 2003:506-522.
4. Cloud storage[EB/OL].[2019-03-26]. <https://en.wikipedia.org/wiki/Cloud_storage>
5. GOH E J. Secure indexes. Cryptography ePrint archive[R]. 2003.
6. CURTMOLA R, GARAY J, KAMARA S, et al. Searchable symmetric encryption: improved definitions and efficient constructions[C]//ACM Conference on Computer and Communications Security.ACM, 2006:79-88.
7. LIU Z, MA H, LI J, et al. Secure storage and fuzzy query over encrypted databases[C]//International Conference on Network and System Security. 2013:439-450.
8. KAMARA S, PAPAMANTHOU C, ROEDER T. Dynamic searchable symmetric encryption[C]//ACM Conference on Computer & Communications Security. 2015:965-976.
9. KAMARA S, PAPAMANTHOU C. Parallel and dynamic searchable symmetric encryption[M]//Financial Cryptography and Data Security. Berlin Heidelberg: Springer, 2013:258-274.
10. HAHN F, KERSCHBAUM F. Searchable encryption with secure and efficient updates[C]//ACM CCS. 2014:310-320.
11. Facebook–Cambridge Analytica data scandal[EB/OL].[2019-04-14]. <https://en.wikipedia.org/wiki/Facebook%E2%80%93Cambridge_Analytica_data_scandal>
12. CAMENISCH J, LYSYANSKAYA A. Dynamic accumulators and application to efficient revocation of anonymous credentials[M]//Advances in Cryptology[C]//CRYPTO. 2002:61-76.
13. WANG P, WANG H, PIEPRZYK J. A new dynamic accumulator for batch updates[C]//International Conference Information and Communications Security (ICICS 2007). 2007: 98-112.