申请上海交通大学硕士学位论文

同义词搜索和抗信息泄漏的对称可搜索加密技术的研究

论文作者 _____

学　　号 _____

指导教师 _____

专　　业 计算机科学与技术

答辩日期 2015 年 1 月 16 日

Submitted in total fulfilment of the requirements for the degree of Master in Computer Science and Technology

Research on synonym search and leakage-resilient in SSE

Supervisor

Prof.

Depart of Computer Science and Engineering, School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, P.R.China

Jan. 16th, 2015

上海交通大学
学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：_____

日　期：_____年 ____月 ____日

上海交通大学
学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保　密 □，在 _____ 年解密后适用本授权书。

本学位论文属于

不保密 □。

（请在以上方框内打“✓”）

学位论文作者签名：_____　　指导教师签名：_____

日　期：_____年 ___月 ___日　　日　期：_____年 ___月 ___日

同义词搜索和抗信息泄漏的对称可搜索加密技术的研究

摘　　要

在大数据时代，越来越多的用户开始使用廉价和计算能力强大的云外包服务。然而安全因素成为了它进一步发展的主要障碍，导致该问题的原因在于：云外包商并非完全可信，致使远程外包的数据处在危险之中。一个简单的解决方案是在数据外包之前对数据加密。但是，加密限制了用户对数据的有效检索。为此，如何避免云端数据被未经授权的人所访问并维持对加密数据的有效计算，已成为云计算外包领域的研究重点。

基于安全索引的可搜索加密技术解决了该难题。可搜索加密技术主要包括：对称可搜索加密技术、公钥可搜索加密技术和隐私信息检索，它们分别解决了不同场景下的应用难题。本文仅仅关注云计算环境下的对称可搜索加密技术，其主要研究内容是对称云环境下的隐私泄漏问题和复杂条件搜索的情形。

为了弥补 Non-adaptive 安全模型中信息泄漏过多的缺陷，我们基于史密斯正交化理论提出了一个以概率抗信息泄漏的方案。与已有的可搜索加密方案相比，我们的方案在减少信息泄漏的同时，并且没有增加客户端的计算与存储开销以及双方交互的通讯开销，同时达到了相同的安全级别。不幸的是，方案提高了服务器的存储和计算开销。通过在安全和性能之间权衡，我们证明这个代价是可取的，并且使方案获得了良好的可扩展性。

另外，到目前为止，没有一个方案对复杂条件搜索下的同义词搜索问题进行研究。在用户记忆具有有限性和人员频繁变更的场景下，以前的方案显得有些难以应付，于是我们认真地研究了该问题。紧接着，我们设计了一个支持同义词搜索的方案。该方案有如下优势：我们第一次阐述了同义词可搜索问题；我们引入了同义词函数和同义词集合，增加了方案的可扩展性；并且通过严格的安全分析，证明我们提出的方案是正确的和 Non-adaptive 安全的。此外，我们提出的算法是高效的，它没有增加用户的计算和存储开销，传输开销仍为

$O(1)$，查找时间复杂度仅为 $O(p)$（p — 单词同义词集合的最大长度)。通过返回所有同义词的文档集合，该技术大大克服了已有方案的不足 — 模糊搜索不能解决所有相似搜索问题，提高了系统的可适用性。因而，我们提出的方案是非常实用的并且弥补了理论与现实的差距。

关键词： 可搜索加密　同义词搜索　信息泄漏　搜索模式　访问模式

Research on synonym search and leakage-resilient in SSE

ABSTRACT

In an era of mass data, more and more users start to utilize cloud outsourcing services of cheap and powerful computing ability. While security problem has become a main obstacle to its further moveforward. The fact that the cloud server provider is ”honest-but-curious” might make outsourced data at risk. A naive method is that these data have to be encrypted prior to outsource to the remote semi-trusted servers to combating unsolicited accesses. But unfortunately, encrypted data restricts the users' effective search ability. Therefore, this point that how to avoid accessing the outsourced data by unauthorized users while maintaining effective computing on encrypted data has been closely researched in the field of cloud outsourcing.

The searchable encryption technique based on secure index, which has been a key and lead solution, solves this issue. It mainly consists of symmetric searchable encryption (SSE), public-key searchable encryption (PKSE) and private information retrieval (PIR). And they, respectively, work out the problem of application under different scenarios. In this thesis, we only focus on SSE technique in the context of cloud computing.The dissertation concerns about researching on problems of data privacy leakage and complexity search on encrypted data in untrusted cloud setting.

In order to make up the defects of information leakage of Non-adaptive security model in the phrase of query, we based on the smith orthogonalization theory present a scheme of resisting leakage of trace in probability. Compared with the previous single keyword search works, not only does our solution leak less information, but also it doesn't increase the computing and storage overheads of clients and communication cost in the mutual interactive process. It is unfortunate that this scheme greatly improves the storage and computational overhead of servers. At last, by balancing trade-off between

security and capability, we show that it is desirable to increase this cost and extends scalability and usability of our solution in practice.

In addition, there exists no such solution focusing on the synonym search in the setting of complexity search. In consideration of finiteness of people's memory and people's frequent changing, existing schemes seem to be difficult to deal with this case. To this case, we seriously study this issue. Soon afterwards, we firstly design a formal Practical Synonym Symmetric Searchable Encryption Solution (PSSSE) in this thesis. Our proposed technique have a number of crucial advantages. To our best knowledge, we for the first time put forward the issue of the synonym keyword search over remotely encrypted data in the context of cloud computing. In this scheme, we introduce the concepts of synonym function and synonym sets to obtain better scalability. Through rigorous security analysis, we show that our proposed solution is correct and privacy-preserving (non-adaptive security) without the loss of function of the scheme. Besides, the algorithms we present are simple and effective, it almost increases no space and computing overhead of the user. The overhead of communication still remains $O(1)$. The time complexity of searching on encryption data is $O(p)$ (p — the maximum number of synonym of keyword). This scheme greatly enhances system's usability by responding all documents containing keyword and corresponding synonyms. Hence, our solution is very significant to practical use and bridges the gap of state-of-art.

KEY WORDS: searchable encryption, synonym search, information leakage, search pattern, access pattern

目　　录

表格索引

插图索引

第一章 绪论

1.1 研究背景及意义

随着网络技术的飞速发展，人们生活中的点点滴滴（从办公、娱乐到洗衣、做饭）逐渐网络化，导致网络中数据的量级呈指数增长。在这些数据的面前，先前的计算技术和存储能力显得有些力不从心。在此背景之下，一种全新的网络技术油然而生 — 即“云存储”和“云计算”技术。它们的出现分别从理论上解决了大数据的存储和计算问题。为此，实际中各大企业纷纷提出不同场景下的“云”解决方案以满足各自的需求。亚马逊（Amazon）率先推出弹性计算云（Elastic Compute Cloud — EC2）服务 [1]；Google 首席执行官埃里克 · 施密特（Eric Schmidt）在搜索引擎大会首次提出“云计算”（Cloud Computing）[2] 的概念；随后包括 IBM、Microsoft、Intel、Apple 和 YAHOO 等一些大型企业都开始部署自己的云存储和计算平台。由此，网络计算和存储的发展正式进入“云”的世界。

“云计算”是一种基于互联网的计算方式，按照这种方式，云终端将已部署的资源 (如网络、服务器、存储、应用及服务) 按需提供给云用户, 并最大程度地减少用户对资源的管理和配置 [3]。通过互联网，云计算为使用者提供强大的可伸缩和廉价的分布式计算能力，并且在使用过程中，使用者不需要了解云端基础技术的细节和具备相关的专业知识，就能对云端资源进行合理的控制和使用。根据美国国家标准与技术研究院 (NIST) 的定义 [4]，云计算提供三种不同层次的服务：

1. 软件即服务（SaaS）：用户可以通过租借云平台上的软件来为自己提供服务或无偿使用一些基础服务软件；

2. 平台即服务（PaaS）：用户利用云计算服务提供商的平台，通过免费或低价租借的方式来部署自己的软件；

3. 基础设施即服务（IaaS）：云用户可以利用云平台基础设施来获得所需服务。

当前的云计算模型按照部署方式主要包括公有云、私有云、社群云和混合云。公有云通过网络及第三方服务提供给用户使用；私有云具有许多公有云环境下的优点（如弹性，实时提供服务），两者差别主要在于，私有云资源仅需在组织内部管理和使用，不会受到网络带宽和安全疑虑的影响；社群云主要由许多利益相仿的组织掌握和使用，社群内成员可以使用共有的资源，避免了公有云开放环境的安全问题；混合云则基于经济性、可用性等的考虑，是公有云和私有云的结合。云计算部署模型从数据隐私方面的角度来考虑，具备以下特点：

- 在公有云环境下，由于数据拥有者与服务提供商站在不同的立场上并且拥有不同的利益，使得服务提供商并不被完全信任（semi-trusted）；
- 在私有云环境下，虽然云资源仅在组织内部使用，但由于云计算普遍采用虚拟化技术 [5] 来提高系统的资源利用率和降低运营成本，不同用户的数据可以同时在同一物理服务器上进行计算和存储。跨虚拟机攻击 [6] 使得用户数据有可能被同一物理服务器上的其他用户非授权访问。

由于“云”计算提供廉价的计算和强大的可伸缩，越来越多的用户将本地的数据移植到“云”平台。与此同时，一些云安全事故也频频发生（e.g.2011 年谷歌邮箱爆发大规模的用户数据泄漏事件，2014 年 Apple 公司 icloud 帐号泄漏事件）。在透明的云环境下，用户简单地将数据以明文形式存储在云服务端存在明显的缺陷 — 数据安全隐患。因此，当数据拥有者将数据存储到云服务端后，如何确保云端数据隐私，避免云端数据被未经授权用户所访问成为云端数据隐私安全研究的焦点。

为了解决该安全隐患带来的缺陷，一些学者利用传统的安全机制解决了数据隐私泄漏的问题，具体过程如下：数据拥有者在将数据存储到并非完全可信的服务端之前，首先将数据加密 [7], 然后将数据以密文的形式存储到云服务端；当用户需要使用数据时，首先将存储在服务器端的加密数据全部下载下来，然后进行解密并查找获得所需的文件。该机制虽然解决了数据的隐私问题，同时带来了新的问题 — 如何才能保证云存储和计算资源被高效地利用。这种做法不仅耗费大量的网络资源，同时也增加了用户的计算开销，这些都与云计算服务的设计理念和用户需求相违背。因此，如何高效利用云端的存储和计算能力是当前关注的重点。

为了解决上述查找效率低下的缺陷，在“云”计算平台下一种新的确保数据隐私且具备高效计算的技术被提出 — 可搜索加密技术。可搜索加密技术解决了上述难点，其过程描述如下：数据拥有者对文档进行可搜索加密，然后存储到云服务平台；当授权的用户需要查找文档时，使用搜索条件陷门（Trapdoor）生成算法将单词的陷门提交至云服务端；一旦云服务端收到搜索陷门后，便直接在用户的密文数据上进行搜索操作，并将搜索结果返回给用户。在方案整个过程中，云服务器无法知晓用户数据以及搜索条件的任何信息，保证了数据隐私性；在网络上仅传输符合搜索条件的数据，而不是所有外包的数据，大大降低了网络资源的消耗；同时请求者只需解密符合条件的文档，大大节省了用户昂贵的计算能力和存储空间。

可搜索加密技术包括对称可搜索加密 (Symmetric searchable encryption)、公钥可搜索加密 (Public key encryption with keyword search) 和多用户 (Multi-User) 可搜索加密技术。近年来，这些技术都得到的广泛的关注。它们的研究领域极其广泛，包括：单关键字搜索、多关键字搜索、模糊搜索、范围搜索、布尔搜索搜索和动态搜索等。这些方案试图提供一个具有完备功能、高性能和强安全的解决方案。相似搜索（包括模糊搜索和同义词搜索 [8]）在明文场景下得到了广泛的研究和应用，虽然这些技术在加密数据上也被关注和研究，但是其研究成果尚有不足。并且在实际研究中，人们仅仅关注模糊搜索却忽略了同义词搜索的情形。此外，这些基于逆向索引的方案在查找时的信息泄漏有所偏多 — 包括大小模式（size pattern）、访问模式（access pattern）和搜索模式（search pattern），可以得到进一步的降低。

综上所述，我们能进一步对相似搜索技术进行的研究，不仅由于其方案不够完善同时包括原有方案存在缺陷；特别是同义词搜索技术，它在实际应用中也存在着极大的需求，因而提出安全的相似搜索加密方案对理论和实际都具有远大的意义。同样地，对目前基于 semi-trusted 的云环境下的对称可搜索的方案的信息泄漏问题进行深入研究，更多地保护用户的搜索隐私，即解决如何进一步降低这些信息的泄漏的问题，不仅能加快推进其在实际中应用的进度，同时也使可搜索加密技术在安全上有所突破，树立一个新的里程碑。为此，探究实用的相似可搜索方案和减少信息在云计算可搜索场景下的泄漏，对安全的云计算的普及具有重要的意义。

1.2 研究现状及相关问题

D.Song 在 [9] 中给出了第一篇对称可搜索加密方案，方案使用传统的加密算法对文档中的每个单词单独加密，来对文档中的数据进行保护。方案的主要缺点是搜索效率低下，同时也泄漏了文章中单词的位置和出现的频率等信息。

为了加快检索，之后的所有可搜索加密方案都使用索引技术。安全索引包括正向索引和反向索引。文章 [10][11] 使用正向索引技术解决了对加密数据搜索效率低下的问题，搜索时间为 $O(n)$（n — 表示文章的数目）。这两篇文章都用到了 Bloom Filter 技术加快检索，[10] 的不足在于引入了误报率 — 与哈希函数的碰撞密概率切相关，而方案 [11] 避免了误报，但是增加了服务器的开销。

为了进一步降低查找时间，[12][13] 使用反向索引技术进一步降低了服务器的查找时间（$O(1)$）。curtmola[12] 是最早使用反向索引技术的可搜索加密方案，并给出了正式化的方案、完整的安全性定义和严格的安全证明。而 Van[13] 提出了另一种形式的安全索引方案，方案使用一个加密的二元数组维护安全索引信息，数组两维分别表示单词和文档 ID，数组元素为“1”表示对应的文档包含对应的单词。搜索时解密二维数组中对应单词所在的行，即可知晓包含该单词的文档 ID。

当前绝大部分可搜索加密方案假定服务器是诚实但好奇 (honest-but-curious) 的，即服务器严格遵守方案的算法和流程，但是因好奇而在用户搜索过程中偷偷分析文档的信息，希望了解文档的更多信息。而实际中，若服务器不遵守规定，如将篡改搜索结果，把某些文档从搜索结果中删去，客户端却无法知晓这些篡改情况。Chai [14][15] 给出了一个支持搜索结果验证的对称可搜索加密方案，可以知晓服务器是否遵守方案协定，而 K. Kurosawa[15] 中的方案可以让客户端进一步精确知晓服务器返回的搜索结果文档中，是否有误报或者漏报的情况，如果有漏报，具体是遗漏了哪些文档。但该方案的安全索引结构增加了大量的存储空间。M. Chase 在 [16] 中提出了一种可控信息揭露 (Controlled Disclosure) 的结构数据加密思想，这种数据密文可以在保持一定机密性的情况下实现快速查询。作者 Mohamad 则在 [17] 中讨论了可控信息揭露的结构数据加密应如何支持验证，以确保查询结果是正确的。

上述方案仅仅支持精确搜索，当面对输入中存在小的错误（如：将“people”误写成“peopel”）或输入单词不一致（如：data mining 与 data-mining）时，显得无能为力。[18][19][20] 提出了支持模糊搜索的可搜索加密方案。[18]

提出使用编辑距离与通配符来如何构造模糊可搜索方案，但缺乏完整的方案描述和严格的安全证明。C. Wang 在 [19] 中使用单词查找树提出了具体详细描述的模糊可搜索加密方案，该方案使用 trie 树构造安全索引降低了服务器的存储空间和查找时间。而 M. Kuzu 在 [20] 中使用 locality sensitive hashing(LSH) 算法 [21]，实现了一个具有更广义的和快速检索的模糊搜索方案。M. Chuah 在 [22] 中将模糊搜索从单个关键词拓展到多个关键词的情况。但是这些方案都存在一个缺陷 — 模糊集之间存在碰撞，降低了方案的安全性；此外，他们不支持同义词搜索。同义词指不同单词之间存在相同的含义，同义词搜索指用某单词搜索返回所有包括该单词及其同义词的文档。

到目前为止，所有基于 inverted-index 的方案都泄漏了 trace 信息（在 curtmola 方案中定义）— 包括大小模式、搜索模式和访问模式。从某种意义上来说，这样的信息泄漏仍不能被用户所接受 — 敌手能使用统计信息和查找的先验知识推断出用户搜索的单词。文章 [23] 提出了一个攻击模型，该模型能在敌手已知一定知识的条件下，利用访问模式发现用户搜索的敏感信息，并且作者针对这种攻击给出了相应的解决方案。不幸的是，方案引入了冗余信息和误报率，并且没有讨论搜索模式的泄漏所带来的安全问题。作者 Liu 在 [24] 中提出两个统计攻击模型，并在该模型下利用通用方案中的搜索模式以很大的概率推断出用户查找的单词。然后提出了一种能在该模型下避免搜索模式泄漏的新方案。但是，这方案引进了误报率和大大增加了传输开销，同时也增加了客户端的计算量。

1.3 研究成果

在本文中，我们将对称可搜索技术中同义词搜索和 curtmola 方案中所定义的信息泄漏作为研究对象，分别提出相应的解决方案。我们的研究成果包括如下两个方面：

1. 首先，我们在安全的云环境下提出了未被研究的同义词搜索问题，并设计了一个形式化的支持同义词搜索的可搜索加密方案（PSSSE)。为了描述同义词关系，我们的方案定义了同义词函数和同义词字典，并针对同义词字典构建了安全的循环链表索引结构；此外，我们的方案在增加功能的同时没有降低性能的要求，传输开销仍为 $O(1)$，查找时间复杂度仅为 $O(p)$（p — 单词同义词集合的最大长度)，计算开销为 $O(p)$。最

后，基于该方案中定义的信息泄漏，我们证明我们提出的技术达到了 Non-adaptive 的安全。此外，该方案能进一步应用到各种复杂的条件搜索方案中，弥补已有方案的不足。

2. 其次，我们提出了一个降低信息泄漏的可搜索加密方案，能在不降低功能的同时避免查找时大小模式的泄漏，仅以概率泄漏搜索时的访问模式和搜索模式。在该方案中，我们对文档进行分块处理，降低了单词与文档的关联度；同时我们在建立安全索引阶段，使用史密斯正交化理论构建的正交基对每个单词进行预处理，再构建单词的安全索引；在搜索时，我们按同样方式对单词进行预处理，然后构建正交基向量的法平面，随机选取平面内的任一向量作为单词搜索口令，即同一单词有不同的查找陷门对应，避免了服务器根据单词口令推断出系统的搜索模式；针对该方案，我们提出了对应的攻击模型和定义了信息泄漏，并通过严格的安全分析，证明了我们的方案达到 curtmola 文中所定义的 Non-adaptive 的安全级别；同时，我们的方案具有良好的可扩展性，能应用于所有单关键字的可搜索加密方案。但是，我们的方案增加服务器的存储和计算开销。

1.4 论文结构

本文的内容分为五章，其结构安排如下：

- 第一章，绪论，首先介绍了课题的研究背景，然后介绍当前国内外在该领域的研究现状，之后介绍了本文的研究内容及所取得的成果，并对本文的结构进行了总结。

- 第二章，对称可搜索加密技术，先介绍了本课题在对称可搜索加密领域的相关知识，并介绍了通用的安全证明模型，最后分类介绍了对称可搜索加密领域具有代表性的一些成果，提出已有研究成果的不足和描述可研究的知识点。

- 第三章，抗信息泄漏的可搜索加密方案，首先描述了目前单关键词可搜索方案的信息泄漏，提出了解决该方案需要的定义及安全模型，然后提出一个首先给出同义词对称可搜索加密方案的若干定义和安全模型，并给出了方案的安全性证明。

- 第四章，同义词对称可搜索加密方案，首先给出同义词对称可搜索加密方案中的若干定义、安全模型与攻击模型，然后提出一个具有实践的安全的同义词可搜索加密方案并给出了优化方法，并对方案进行了完整安全性证明和性能分析。
- 第五章，总结和展望，系统地概述了我们在该课题中的工作与成果，并根据目前的研究现状对将来的工作作出了规划与展望。

第二章 对称可搜索加密技术

2.1 预备知识

2.1.1 数学基础知识

定义 2.1 (可忽略函数（Negligible Function）). 对于一个函数 $f: N^* \to N^*$，如果对任意给定的正多项式 $p(.)$，总存在一个足够大的数 k，使 $f(k) < \frac{1}{p(k)}$，则称函数 f 在 k 上是可忽略的。

定义 2.2 (伪随机函数（Pseudo-random Function）[25]). 对任意的函数 $\mathcal{F}$: $\{0,1\}^n * \{0,1\}^k \to \{0,1\}^m$，如果满足:

1. 对于任意给定的输入 $K \in \{0,1\}^k$ 和 $x \in \{0,1\}^n$，$\mathcal{F}(x,K)$ 总能在多项式时间内被计算;

2. 对所有多项式大小的敌手 $\mathcal{A}$，有:

$$|Pr[\mathcal{A}^{\mathcal{F}_K(.)} = 1 : K \xleftarrow{\$} \{0,1\}^k] - Pr[\mathcal{A}^{\mathcal{G}(.)} = 1 : \mathcal{G} \xleftarrow{\$} Func[n,m]]| \leq negl(k)$$

（$Func[n,m]$ 表示所有 $\{0,1\}^n \to \{0,1\}^m$ 的函数集合，$negl$ 是在 k 上的可忽略函数)。

则称函数 $\mathcal{F}$ 是伪随机函数。如果函数 $\mathcal{F}$ 是双射，则称为伪随机置换函数。

定义 2.3 (伪随机生成器（Pseudo-random Generator）[26]). 对任意函数 $\mathcal{G}$: $\{0,1\}^n \to \{0,1\}^m$，在 $m > n$ 的条件下，如果满足:

1. 函数 $\mathcal{G}$ 可以使用确定性的算法在多项式时间内被计算;

2. 对所有 t 多项式时间的算法 $\mathcal{A}$，有:

$$|Pr[\mathcal{A}(\mathcal{G}(s)) = 0 | s \xleftarrow{\$} \{0,1\}^n] - Pr[\mathcal{A}(r) = 0 | r \xleftarrow{\$} \{0,1\}^m]| < negl(k)。$$

则称函数 $\mathcal{G}$ 是在条件 $(t, negl(k))$ 下的伪随机生成器（$negl$ 是在 k 上的可忽略函数)。在多项式时间内，伪随机生成器 $\mathcal{G}$ 输出的字符串与随机字符串不可区分。

定义 2.4 (对称加密方案)**.** 对称加密方案 SKE 是由三个多项式时间算法组成的集合，定义：SKE = (Gen, Enc, Dec)。

- Gen：输入安全参数 k，返回私钥 K;
- Enc：输入密钥 K 和明文消息 m，并返回密文 c;
- Dec：输入密钥 K 和密文 c（密钥 K 与密文 c 生成时的密钥相同)，解密得到消息 m。

定义 2.5 ($PCPA$-Security)**.** 如果 SKE = (Gen, Enc, Dec) 是一个对称加密方案，$\mathcal{A}$ 是一个敌手，在下面的概率实验 $PCPA_{SKE,\mathcal{A}}(k)$ 过程中:

1. $Gen(1^k)$ 生成一个密钥 K,
2. $\mathcal{A}$ 被赋予 oracle 访问 $Enc_K(.)$ 的能力,
3. $\mathcal{A}$ 输出一条信息 m,
4. 首先 $c_0 \leftarrow Enc_K(m)$ 和 $c_1 \xleftarrow{\$} \mathcal{C}$ 分别产生两个密文 c_0 和 c_1，这里 $\mathcal{C}$ 表示 SKE 方案的整个密文空间。然后随机的选择 $b \xleftarrow{\$} \{0,1\}$，并发送 c_b 给敌手 $\mathcal{A}$,
5. $\mathcal{A}$ 被再次给予访问加密 oracle 的能力，经过多项式次数的查询后输出一个 b',
6. 如果 $b' = b$，该实验返回 1，否则返回 0。

如果对于所有的多项式敌手 $\mathcal{A}$，有:

$$Pr[PCPA_{SKE,A}(k) = 1] \leq 1/2 + negl(k)$$

这里的概率仅与 b 的选择和算法 Gen 与 Enc 引入的随机量有关，则我们认为 SKE 方案是 CPA-安全的。

2.1.2 安全索引

1. **正向安全索引（Forward-Index）**

   Goh. 等在 [10] 中给出了一种可搜索加密方案，是第一篇使用索引结构的方案。该方案使用 Bloom Filter[27] 建立正向的索引结构，从而加快单词在文章中的查找。正向索引是一种使用文档映射到对应单词集的二元组结构，即针对每个文档保存一份包含所有单词的集合，例如表2–1所示：

文档	单词
$D1$	hello, word, welcome, document
$D2$	hello, paper, good, well
$D3$	world, paper, welcome, good
$D4$	document, welcome, well

表 2–1 正向索引示例

Table 2–1 An Example of Forward-Index

   从表2–1可知，通过 Forward-Index 进行搜索时，需要遍历索引中的每个条目，即需对每个文档的 Bloom Filter 结构扫描一遍，检查该单词是否包含在其中，效率相当低下 — 与文档数目的数目相关。但是对于文档的增删改操作，索引的调整非常简单，仅需增加、修改或者删除相应的条目即可。

2. **反向安全索引（Inverted-Index）**

   2006 年 Curtmola 等在 [12] 中第一次基于 inverted-index 提出了支持高效查找和强安全（Adaptive）的可搜索加密方案。该方案的高效查找得益于对每个单词建立了一个反向索引结构，使得在查找时不需要查找每个文档，就可以找到所需要的文档信息。反向索引是一种用单词来查找相关文档的映射结构，即对每个单词，保存一份所有包含该单词的文档集合，例如表2–2 所示：

   从表2–2的结构中可知，在 inverted-index 中搜索时，只需要找到待查单词对应的条目，然后输出相应的文档集合即可。若索引结构使用哈希建表进行构造，则搜索时仅需要 $O(1)$ 的时间就能查找到所有符合条件的文档，效率相当高。但是对于文档的增删操作，索引的调整则相当麻烦，

单词	文档
hello	$D1$, $D3$, $D5$
world	$D2$, $D3$, $D5$
paper	$D1$, $D4$, $D5$
document	$D2$, $D4$, $D5$

表 2–2 反向索引示例

Table 2–2 An Example of Inverted-Index

需要修改所有包含该单词的文档集合，同时要避免信息泄漏。

2.2 对称可搜索加密方案的模型

在多用户环境下，一个典型的对称可搜索加密技术的系统模型如下图2–1所示：

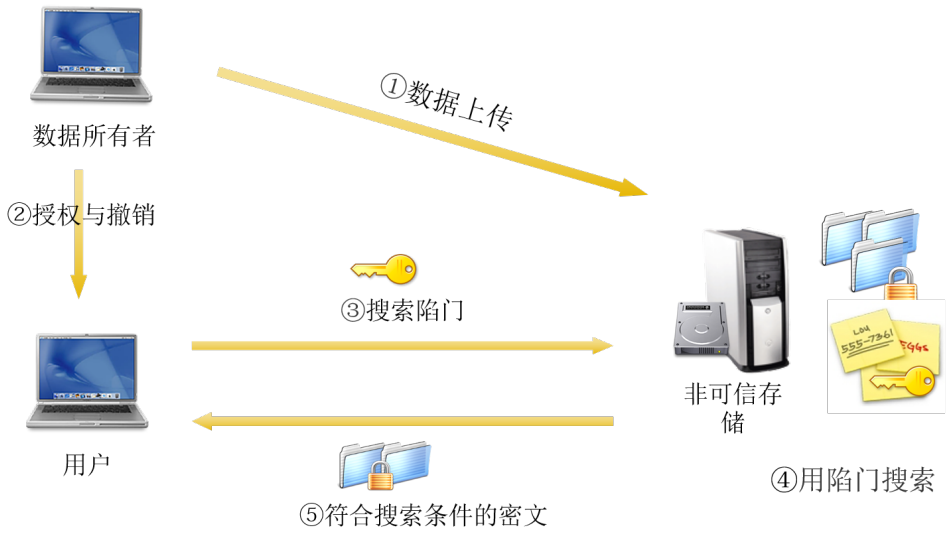

图 2–1 对称可搜索加密的系统模型

Fig 2–1 A General System Model in SSE

从图2–1可知，一个典型的模型有三部分组成：数据所有者、授权用户和非可信服务器。通常数据所有者拥有数据，一个被授权的组织可以对加密的数据进行查找。过程如下：数据拥有者上传待外包的文档并维护对加密数据的高

效检索，并且对某个组织内的所有用户进行授权或撤销；被授权的用户可随时对远程加密的数据进行搜索，并对接收的结果解密；服务器具有对加密数据进行各种条件搜索的能力，并返回符合条件的答案。我们对数据所有者给用户进行授权和撤销的细节没有考虑，可以使用各种技术完成（公钥加密或私钥加密等）— 仅仅传递密钥，信息量小。

针对该系统模型，这里简单介绍一个具有代表性的基于 inverted-index 的对称可搜索加密方案，该方案由五个算法组成：$SSE = (Gen, Enc, Trpdr, Search, Dec)$，数学定义如下：

1. $Gen(1^k) \rightarrow K$：给定安全参数 k，生成密钥 K；
2. $Enc(K, D) \rightarrow (I, c)$：给定一组文档 $D = D_1, D_2, ...D_n$，使用密钥 K 加密，生成安全索引 I 和密文文档集合 $c = \{c_1, c_2, ..., c_n\}$；
3. $Trpdr(K, w) \rightarrow t_w$：给定一个单词 w，使用密钥 K 调用陷门生成函数，输出对应的陷门 t_w；
4. $Search(I, t_w) \rightarrow R$：对陷门 t_w 在安全索引 I 上进行搜索，输出所有包含单词陷门的文档 ID 集合 R；
5. $Decrypt(K, c_i) \rightarrow D_i$：使用密钥 K 对文档 c_i 解密，生成 D_i。

数据拥有者首先调用 Gen 算法生成密钥并保存在本地，然后对需上传的文档调用 Enc 算法，生成安全索引和一组加密文档，并上传至服务器。当需要检索时，用户调用 $Trpdr$ 算法生成待搜索单词的陷门，并发至服务器；然后服务器调用算法 $Search$，使用收到的陷门在安全索引上查找，输出所有符合条件的文档 ID，并将对应的加密数据返回给请求者。最后，数据拥有者调用 $Decrypt$ 算法解密加密文档。

2.3 Non-adaptive 安全模型

该小节描述的所有定义来自于 curtmola 的方案 [12]。

对称可搜索加密技术（SSE）的安全证明模型与密码学中常用的安全证明模型（通常是 CPA 和 CCA）有所不同。根据实际的场景，当前 SSE 大多以选择关键字攻击（CKA）作为安全分析模型，典型的安全级别包括 Non-Adaptive

安全和 Adaptive 安全。在对称可搜索加密环境下的 Non-adaptive 和 Adaptive 的安全模型，如图2–2所示：

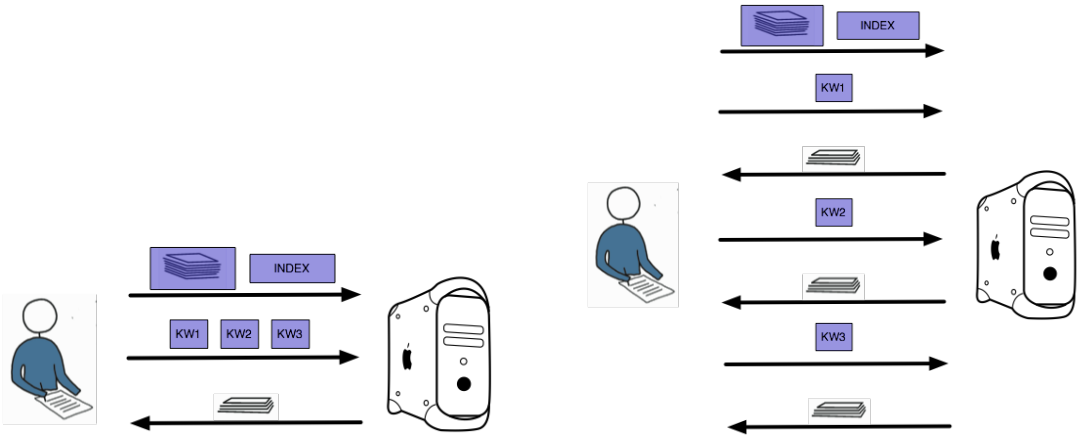

图 2–2 SSE 上 Non-adaptive 和 Adaptive 安全模型

Fig 2–2 The Security Model of Non-adaptive and Adaptive in SSE

这里，我们选择具有代表性的安全等级相对较弱的 Non-Adaptive 模型，并分别从语义安全（semantic security）和不可区分安全（indistinguishability security）进行描述。

定义 2.6 (History)**.** 假定 Δ 代表单词词典，D 表示在字典 Δ 上的一组文档集合。D 上 q 次查询的历史 (History) 定义为一个元组 $H = (D, w)$，其中 w 是包含 q 个单词的集合：$w = (w_1, ..., w_q)$。

定义 2.7 (Access Pattern)**.** 假定 Δ 代表单词词典，D 表示在字典 Δ 上的一组文档集合。D 上 q 次查询历史 $H = (D, w)$ 的访问模式 (Access Pattern) 定义为一个元组 $\partial(H) = (D(w_1), ..., D(w_q))$。

定义 2.8 (Search Pattern)**.** 假定 Δ 代表单词词典，D 表示在字典 Δ 上的一组文档集合。D 上 q 次查询历史 $H = (D, w)$ 的搜索模式 (Search Pattern) 定义为对称二元矩阵 $\sigma(H) = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,q} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ x_{q,1} & x_{q,1} & \dots & x_{q,q} \end{pmatrix}$。对于任意 $1 \le i, j \le q$，如果 $w_i = w_j$，则 $x_{i,j}$ 为 1，否则为 0。

定义 2.9 (Trace)**.** 假定 Δ 代表单词词典，D 表示在字典 Δ 上的一组文档集合。D 上 q 次查询历史 $H=(D,w)$ 的 Trace 定义为：$(H)=(|D_1|,...,|D_n|,\partial(H),\sigma(H))$，其中 $|D_i|(1\leq i\leq n)$ 代表第 i 个文档的长度。

定义 2.10 (Non-singular history)**.** 如果对任意的历史 H 满足：(1) 至少存在一个历史 $H'\neq H$，使 $\tau(H')=\tau(H)$；(2) 在给定的 $\tau(H)$ 下，历史 H' 可以在多项式时间内被发现；则称历史 H 是 Non-singular。

2.3.1 Non-Adaptive 不可区分安全

定义 2.11 (Non-adaptive Indistinguishability)**.** 在词典为 Δ，安全参数为 $k\in\mathbb{N}$ 的基础上，设 $SSE=(Gen,Enc,Trpdr,Search,Dec)$ 是一个基于安全索引的对称可搜索加密方案，且 $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)$ 为 $non-uniform$ 的敌手，考虑下面的概率试验 $\mathbf{Ind}_{(SSE,\mathcal{A})}(k)$：

$\mathbf{Ind}_{SSE,\mathcal{A}}(k)$
$K\leftarrow Gen(1^k)$
$(st_{\mathcal{A}},H_0,H_1)\leftarrow\mathcal{A}_1(1^k)$
$b\xleftarrow{\$}0,1$
parse H_b as (D_b,w_b)
$(I_b,c_b)\leftarrow Enc_K(D_b)$
$for 1\leq i\leq q$
$t_{b,i}\leftarrow Trpdr_K(w_{b,i})$
let $t_b=(t_{b,1},...,t_{b,q})$
$b'\leftarrow\mathcal{A}_2(st_{\mathcal{A}},I_b,c_b,t_b)$
if $b'=b$, output 1
otherwise output 0

假设在 $\tau(H_0)=\tau(H_1)$，$st_{\mathcal{A}}$ — 表示敌手 $\mathcal{A}$ 的状态信息的前提下，如果对于任意多项式的敌手 $\mathcal{A}$，有：

$$Pr[\mathbf{Ind}_{SSE,\mathcal{A}}(k)=1]\leq\tfrac{1}{2}+negl(k),$$

则称该 SSE 方案在 Non-adaptive 不可区分的条件下是安全的。

2.3.2 Non-Adaptive 语义安全

定义 2.12 (Non-adaptive Semantic Security)**.** 在词典为 Δ，安全参数为 $k \in \mathbb{N}$ 基础上，设 $SSE = (Gen, Enc, Trpdr, Search, Dec)$ 是一个基于安全索引的对称可搜索加密方案，$\mathcal{A}$ 为一个敌手，$\mathcal{S}$ 是一个模拟器（Simulator），考虑下面概率过程：

$\mathbf{Real}_{SSE,\mathcal{A}}(k)$
$K \leftarrow Gen(1^k)$
$(st_{\mathcal{A}}, H) \leftarrow \mathcal{A}(1^k)$
parse H as (D, w)
$(I, c) \leftarrow Enc_K(D)$
for $1 \le i \le q$
$t_i \leftarrow Trpdr_K(w_i)$
let $t = (t_i, ..., t_q)$
output $V = (I, c, t)$ and $st_{\mathcal{A}}$

$\mathbf{Sim}_{SSE,\mathcal{A},\mathcal{S}}$
$(H, st_{\mathcal{A}}) \leftarrow \mathcal{A}(1^k)$
$V \leftarrow S(\tau(H))$
output V and $st_{\mathcal{A}}$

如果对于任何多项式规模敌手 $\mathcal{A}$，都存在一个模拟器 $\mathcal{S}$，使得对于任意多项式规模的区分器 $\mathcal{D}$，有：

$$|Pr[\mathcal{D}(V, st_{\mathcal{A}})] = 1 : (V, st_{\mathcal{A}}) \leftarrow \mathbf{Real}_{SSE,A}(k)] - Pr[\mathcal{D}(v, st_{\mathcal{A}}) = 1 : (V, st_{\mathcal{A}} \leftarrow \mathbf{Sim}_{SSE,\mathcal{A},\mathcal{S}}(k)]| \le negl(k),$$

则称该 SSE 在 Non-adaptive 的条件下是语义安全的。

2.4 对称可搜索加密方案分类

2.4.1 单关键字搜索

对于可搜索加密问题，Goh 在文章 [10] 中的提出了第一个基于正向索引的解决方案。方案使用一个 Bloom Filter[27] 作为安全索引。对于每个文档，映射包含的所有单词到一个 Bloom Filter；在搜索时，授权用户发送待查单词的多个哈希值作为陷门；一旦服务器收到请求后，逐个检查每个 Bloom Filter，判断对用的位置是否是“1”，如果全部都为“1”，则表示包含该单词并返回该文档，否则跳过。该方案突出的优势体现在性能上：Bloom Filter 的映射过程只

需要计算若干 Hash 函数，故索引创建过程性能较高。在搜索阶段，服务器需要对每个文档调用一次 SearchIndex 算法，而在该算法中对陷门中每个元素只需进行 Bloom Filter 数据位的比较操作，时间复杂度仅为 $O(1)$，故整体效率仍然是比较高的。但是该方案却引进了误报率，且误报率和安全索引的大小成反比关系。

Y. Chang 在 [11] 中则基于 forward-index 提出了没有误报率的的方案。其基本思想如下：将待上传文档中所有出现的单词构成一个词典，安全索引中用一比特位代表每个单词是否存在 — “1”表示存在该单词，“0”表示不存在。方案同样达到了相当高的效率，但问题在于词典的保存大大增加了存储开销的负担。文中给出了两个解决方案，第一个方案是将词典保存在客户端，这样用户每次查询则需要本地词典，这样增加了本地的存储和计算开销，并且在多用户环境下还需要同步词典；第二个方案则将词典加密后存放在远程服务器上，但查询过程则需要通讯两轮 — 一轮获得词典里索引信息，另一轮查询获得文档。

R.Curtmola 在文章 [12] 中第一次基于 inverted-index 给出了两个改进方案，并给出了详细的安全性定义和证明。第一个方案达到了 Non-adaptive 安全性，其基本思想描述如下：首先构建待上传文档的 inverted-index；然后将索引表项中的文档 ID 部分加密并随机分散到一个数组中，如下图2–3所示：

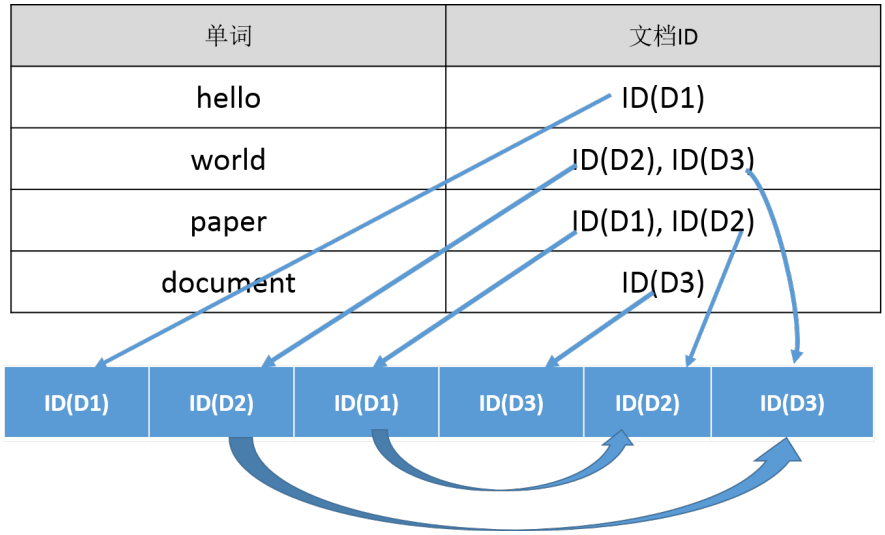

图 2–3 文档 ID 数组

Fig 2–3 The Array of Document ID

从上表可知，每个单词所对应文档的 ID 形成一个链表。数组中的每个

元素使用不同的密钥调用对称分组密码算法加密。而文档中所有的单词在 Inverted-index 则以查找表的形式存储，如图2–4 所示：

单词陷门	文档ID在数组中起始下标
T(hello)	1
T(world)	2, 6
T(paper)	3, 5
T(document)	4

图 2–4 反向安全索引中的查找表

Fig 2–4 The Look-up Table of Secure Index

查找表的条目由键值对的形式表示，其键存储的是通过伪随机函数对单词置换后的结果 — 单词的陷门，其值所存储的部分为对应文档 ID 的链表在数组中的起始位置。查找的时候，通过单词的陷门在查找表处找到其 ID 链表在数组中的起始位置，然后根据起始 ID 位置逐个解密这个链表获得所有包含该单词的文档 ID 集。

Curtmola 方案一的主要优势是服务器端的搜索效率高，得益于 inverted-index，其服务器端搜索时间复杂度为 $O(1)$，而之前的方案都只能达到 $O(n)$（n 表示文档的数目）。同时 R. Curtmola 在文中给出了另一个具有 Adaptive 安全性的方案，服务器端搜索时间复杂度仍能保持 $O(1)$，但是服务器端的存储量和单词陷门的大小均有所增加。

为了确保方案足够安全 — 不能因 ID 链表元素个数不同而导致信息泄漏，Curtmola 建议数组元素的个数应与所有文档的总长度所能容纳的最大单词数相当。这将导致在安全索引中，数组部分所占用的存储空间将会远大于所有文档的总长度。H. Lu 在文章 [28] 中基于 inverted-index 提出了一种降低安全索引结构所占存储空间的方案。方案与 Curtmola 在方案 1 的最大不同在于合并了安全索引中 ID 链表结构中相同的元素，从而极大减少了数组的长度，如图2–5所示：

基于这样的结构使得合并后每个文档 ID 在数组中只出现一次，从而减少数组元素的总数。经实际测试，方案中数组的元素个数可降至原来的 5% 以下。

单词	文档ID
hello	ID(D1), ID(2), ID(4)
world	ID(1), ID(D2), ID(D3)
paper	ID(D1), ID(D2)
document	ID(D3)

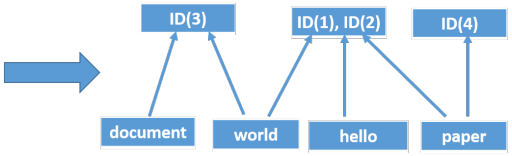

图 2–5 改进的文档 ID 链表

Fig 2–5 An Improved List of Documents ID

作者 Van 在 [13] 中基于 inverted-index 提出了另一种形式的安全索引方案，方案使用一个加密的二元数组维护安全索引信息，数组两维分别表示单词和文档 ID，数组元素为“1”表示对应的文档包含对应的单词。搜索时解密二维数组中对应单词所在的行，即可知晓包含该单词的文档 ID。

2.4.2　模糊搜索

在加密的云数据环境中，当前的相似搜索的研究集中在模糊搜索上。基于明文模糊搜索 [29] [30] [31] 和精确关键字搜索 [12] [11] [28]，Jin Li 等人于 2010 年在 [18] 中率先提出了模糊对称可搜索加密的问题，并提供了两种解决方案：直接模糊搜索方案和基于通配符的模糊搜索方案。方案用编辑距离（edit distance[32]）来衡量单词之间的相似性，即若给定单词 w_1 和 w_2，定义其编辑距离为：将 w_1 变换成 w_2 所需的最小的操作数，变换操作包括：（1）插入：在单词的某个位置上插入一个字符；（2）删除：删除单词中某个字符；（3）替换：将单词中某个字符替换为另一个字符。该方案的基本思想如下：（1）加密阶段：对文档集中每个单词 w 构建模糊集 F_w，用与 [12] 方案相同方式构建反向安全索引，然后将安全索引和加密文档外包；（2）陷门生成阶段：当搜索单词 w 时，首先构建单词 w 的模糊集 F_w，然后计算集合 F_w 所有单词的陷门并提交至服务器；（3）搜索阶段：当服务器收到请求的陷门后，在安全索引中查询是否存在和单词 w 的陷门精确匹配的索引，若存在则返回，否则返回与模糊集 F_w 所有陷门匹配的结果。

在直接的模糊搜索方案中，构建单词 w 的模糊集如下：枚举所有的单词 w' 使得 $ed(w, w') \leq d$。而在改进的模糊搜索方案中，用通配符来替代同一位置上所有不同的单词，这样大大降低了模糊集的规模，降低了网络传

输开销和服务端的存储开销。但是不幸的是，方案对服务器存储开销与编辑距离正相关，且不支持多关键字搜索。同时，不同单词的模糊集之间存在碰撞，例如当单词 "at" 和 "it" 在编辑距离为 1 时，对应模糊集分别为：$S_{at,1} = \{*at, *t, a*t, a*, at*\}$，$S_{it,1} = \{*it, *t, i*t, i*, it*\}$（$S_{w,d}$ 表示单词 w 在编辑距离为 d 时的模糊集），有 $S_{at,1} \cap S_{it,1} \neq \varnothing$。

为了解决上述方案中服务端存储过大的问题，M. Chuah 等人在 [22] 中提出了基于 BedTree[33] 的模糊可搜索加密方案。方案的具体过程如下：（1）加密阶段：对于单词 w 在编辑距离为 d 时，将其模糊集对应的陷门映射到 Bloom Filter 中，如将 $S_{w,i}$ 映射到 B_i 中（$S_{w,i}$ 表示单词 w 在编辑距离为 i 时的模糊集，B_i 表示编辑距离为 i 时模糊集所构造的 Bloom Filter），然后将单词的 $\{B_i, i \leq d\}$、单词所对应文档的 ID 信息以及单词的陷门作为一个叶子节点，通过单词的数据向量插入到 BedTree 树中；（2）陷门生成阶段：对待搜索单词 w，计算其数据向量和模糊集中每个元素的陷门，并提交至服务端；（3）搜索阶段：当服务器收到用户的查询信息后，首先通过其数据向量在 BedTree 树构造的安全索引中查找某叶子节点，再检查模糊集陷门是否存在于 B_i 中，最终返回正确匹配的结果集。该方案主要在安全索引结构上有所改进，利用 BedTree 的索引结构降低了存储空间和 Bloom Filter 减少了索引构建时间，同时能很好地扩展到多关键词搜索和增量更新。然而，方案也引入了新的的问题 — 由 Bloom Filter 引入了一定的误报率。

Cong Wang 等人在 [19] 中基于 Trie 树的结构提出了搜索效率更高且具备同等安全的模糊可搜索加密方案。他们首先对方案 [18] 提出了改进措施：对每个单词 w，根据容许的最大编辑距离 d 计算出单词的模糊集 $S_{w,d}$，然后根据 Bloom Filter 技术将每个单词的模糊集计算出的令牌映射到一个 Bloom Filter，即对每个关键字的模糊集构建一个 Bloom Filter；在搜索关键字 w 时，服务端只需搜索待搜索单词模糊集对应令牌是否存在某个 Bloom Filter 中，即可查询得到所需的结果。基于 [18] 的改进方案虽然减少服务器的存储量，但是却增加了服务端的搜索时间 — 对所有关键字构建的 Bloom Filter 索引都要搜索一次，并且由 Bloom Filter 结构带来了一定的误报率。方案的构建过程如下：（1）加密阶段：首先计算文档中每个单词的模糊集和对应令牌信息。然后对于每个令牌将其分成 N 块，并将 N 块的子结构插入到 Trie 树中，叶子节点即单词令牌的最后一块，并将单词对应文档信息放在叶子节点中或者在当前叶子节点下插入一个新的节点存放其他信息，最后将安全索引和文档密文外包到服务端；（2）

陷门生成阶段：对单词 w，计算模糊集及对应令牌，并发送给服务端；（3）搜索阶段：当服务端收到用户发来的令牌集后，首先对单词 w 的令牌按照构建索引过程分成 N 块，然后在基于 Trie 树的索引中进行精确查找，若找到返回精确查询的结果，否则对其模糊集令牌按照同样方式搜索并返回结果集。该方案主要基于不同单词之间存在相同的部分，通过相同的部分来减少存储量。该方案与方案 [18] 相比，明显减少了服务端的存储开销但是增加了常量的搜索时间。与方案 [22] 相比，该方案减少了搜索过程中信息的泄漏增强了方案的安全性，但是从数据的时间局部性和空间局部性的角度来看，由于 BedTree 是一颗 B+ 树，而 Trie 是一棵多叉树，因而 B+ 树可以利用数据局部性原理来通过减少内存在访问过程中搜索失效时数据内存时间换入换出来提高单词的搜索时间。

Deshpande 等人在 [34] 中对目前所有模糊可搜索加密方案进行了总结 — 包括一般的基于通配符构建模糊集的可搜索加密方案、基于 BedTree 树构建索引的模糊可搜索加密方案和基于 Trie 树构建索引的模糊可搜索加密方案；并提出了两种高效的模糊单词集构建方案 — 基于通配符和 Gram；并通过实验分析了各方案在不同编辑距离时索引构建的时间和搜索的时间及个方案相关优缺点。

上述方案都以编辑距离来衡量单词间的模糊程度，不可避免带来了碰撞，引入了安全隐患。为此，[35] 第一次提出了 sub-linear 的模糊可搜索加密方案。在方案中，他们使用了基于有权图的理论来描述单词间的相似性，并对此提出了一个更强的安全定义，证明之前方案并不能达到此安全。此外，他们也给出了如何在安全性和高效性做出平衡。

2.4.3 动态搜索

基于正向索引的解决方案 [10] [11] 等都能很好地适应与动态的搜索领域。对于扩展于动态搜索，索引结构的调整非常简单，仅需要在正向索引中添加或删除对应条目即可。但是正向索引的方案在查找上效率并不高，通常与文档数目相关。

为了解决上述方案中的不使用问题，Kamara 等人于 2012 年在 [36] 中第一次提出了动态可搜索加密技术的问题及相应的解决方案。在文中作者主要结合了正向索引具有的动态修改特性和反向索引的强安全性和无误报率等特征，在反向索引方案的基础上通过增加另一个正向索引，实现了具有动态修改的可搜索加密方案。该方案不仅获得了有效的搜索时间 — 线性搜索时间（与文档数

目成正比），而且确保了搜索过程中的信息泄漏，具有 Non-adaptive 的安全性。该方案的主要思想如下：在索引构建阶段，不但对每个单词构建了一个反向索引，而且对每篇文档构建了一个正向索引，正向索引和反向索引通过对偶节点（dual node）联系起来（所谓对偶节点即将正向索引结构中的 < 文档，单词 > 条目和反向索引结构中的 < 单词，文档 > 条目具有相同值的那对节点称为对偶节点）；在搜索时，只需要查询反向安全索引即可；而在添加或删除文档时，需通过一系列复杂的位操作对正向索引和方向索引进行更新，同时维护正向索引与反向索引之间的对应关系。随后，Kamara 在文章 [37] 中对 [36] 方案进行扩展，提出具有并行特征的动态可搜索加密方案，这样可充分利用基于多核和分布式的计算机来提升方案的整体性能。但是服务器端额外存储量增加了 1 倍以上。

Stefanov 等人在 [38] 中提出了具有更少信息泄漏和更高效的动态可搜索加密方案。他们指出 Kamara 方案中泄漏的信息并不仅仅限于他们方案中所定义的信息 — 泄漏搜索模式（search pattern）[24]、访问模式（access pattern）[23] 和大小模式（size pattern），同时也存在 forward privacy（即在搜索单词 w 后，然后立即添加一个包含 w 的文档，服务器不应该了解到新添加的文档包含用户刚搜索过的单词 w）和 backward privacy （即当删除包含某单词 w 的文档后，随之搜索单词 w，服务器不应该了解到刚被删除的文档包含单词 w）的信息泄漏。作者在文中第一次提出具有 forward privacy 安全的动态可搜索加密方案，但是 backward privacy 的信息泄露问题仍然没有得到解决。

2.4.4　优先级搜索

支持优先级搜索的可搜索加密技术 (Ranked Keyword Search) 是指对于文档中出现的单词，其对应的搜索结果文档具有不同的优先级 — 如文档中被搜索单词出现次数多则优先级高，反之则优先级低。搜索时，服务器按优先级先后次序返回搜索结果，或者返回指定数量的优先级最高的文档。最简单的做法将客户端预先计算的优先级加密存放到安全索引中，搜索的时候，服务器解密搜索结果文档所对应的优先级，从而进行筛选或排序。但 A. Swaminathan 在 [39] 中证实：如果服务器能直接获得优先级值的话，在特定情况下，有可能获得明文或者搜索关键词的信息。

上述方案仅仅能将结果返回给用户，不能对结果进行过滤，例如搜索结果

按优先级排序。C. Wang 在 [40][41] 中最先考虑了支持单个单词的优先级对称可搜索加密技术，给出了这类方案的具体定义，并设计了一个具体的方案。方案中对文档的优先级使用保序加密（Order Preserving Encryption）[42] 技术进行加密；对于不同的单词，使用不同的密钥加密优先级。使得服务器无法得到优先级的具体值，甚至无法评估不同关键词对应搜索结果的优先级差异，从而降低了信息泄漏，使其安全性能与之前的对称可搜索加密方案相当。

N. Cao 在 [43] 中将优先级搜索问题扩展到多个单词的情形，通过 coordinate matching[44] 综合考虑一篇文档相对于多个搜索单词的综合优先级，并给出了具体的支持多个搜索单词的优先级搜索方案。该方案的问题在于单词词典是固定的，如果需要增加新的单词的话，需要进行重构操作。Z. Xu 在 [45] 中针对这个问题作出了改进，使得新增单词时，只需要少量调整操作；方案中还考虑了单词访问频率对优先级的影响。J. Yu 则在 [43] 中做了进一步的研究，给出了安全性更强的方案。

第三章　抗信息泄露的可搜索加密

当前基于索引的可搜索加密方案都实现了云服务器中对加密数据进行检索的基本功能，并能获得 Adaptive 安全。基于正向索引的方案与使用反向索引技术的方案相比：正向索引方案中搜索效率与文档的数目成正比，而反向索引方案中搜索时间与文档大小无关仅为 $O(1)$；两者方案都泄漏了大小模式和访问模式，而反向索引方案还泄漏了搜索模式而正向技术没有；但是，可搜索加密技术处于大数据的背景下，搜索时间是方案的一个重要指标，即使正向索引方案泄漏更少的信息，仍不被实际所采纳。

基于反向索引的可搜索加密技术的信息泄漏以 curtmola[12] 中定义的 trace — 包括大小模式（size pattern）、搜索模式（search pattern）和访问模式（access pattern）为标准，即这些方案除了泄漏 trace 信息之外，没有泄漏其它任何信息。至今为止，仍没有一个解决方案在实际上能抵御或降低该信息的泄漏。为此，从安全性角度出发，我们对以前方案进行了改进，提出了一个抗信息泄漏的可搜索加密方案。

通过使用史密斯正交化知识构建单词搜索陷门，使我们的方案在获得与通用可搜索方案有相同功能和安全的同时，避免了大小模式的泄漏，并分别仅以概率泄漏搜索模和访问模式，同时没有增加客户的负担和网络负荷。但是，该方案增加了服务器的存储开销和计算开销。

3.1　问题定义

在通用的可搜索加密模型2–1中，信息泄漏包括大小模式、访问模式和搜索模式，大小模式仅指文档的长度，很容易通过填充进行避免。下面我们分析搜索模式和访问模式，了解当前方案中导致该泄信息漏的根本原因。

3.1.1　搜索模式泄漏

搜索模式是一个二维矩阵，其信息透露了该轮搜索的单词是否在之前已被搜索过。从图2–1我们可以了解到，搜索模式的泄漏主要来源于陷门生成函数使用的是确定性算法。如我们查询的单词序列是 $(w_1, w_2, w_1, w_1, w_1)$，则请求

的搜索陷门分别为 $(T_{w_1}, T_{w_2}, T_{w_1}, T_{w_1}, T_{w_1})$，这样敌手可以通过用户搜索习惯的统计信息并结合英文单词的频率推断出所查找的单词。另外，即使我们在搜索时的陷门使用概率算法没有泄漏，但是在建立索引时，若安全索引结构中每个单词的在查找表中的内容也是基于确定性的算法，则同样地敌手在查找过程中，通过查找表中匹配的结果来分析和推断出用户的搜索模式。因而，使用确定性的算法建立陷门必然会导致搜索模式的信息泄漏，我们必须设计一种新的算法来避免或以概率揭露用户的搜索模式给敌手。一个典型的搜索模式信息泄漏的情况如表3–1所示：

文档	单词
w_1	$T(w_1)$
w_2	$T(w_2)$
w_3	$T(w_3)$
w_4	$T(w_4)$

表 3–1 搜索模式信息泄漏示例

Table 3–1 An Example Of Search Pattern Information Leakage

从上表可知，对于每个单词而言，他们所产生的陷门信息是确定的，即对同一单词必产生相同的输出陷门。为此，我们必须设计一个对某一个单词，其输出陷门是概率性确定的算法。

3.1.2 访问模式泄漏

在 q 次查询历史的条件下，访问模式是指通过单词搜索得到 q 个密文文档的集合，即通过待查单词，敌手最终能获得查找后的文档结果集，即使不能解密，仍能通过多次的查找访问模式中概率推断出查找单词的搜索模式，若每个单词有唯一的文档 ID 集，则能完全推断。如已知 5 次查询历史的单词为：$(w_1, w_2, w_3, w_2, w_1)$，敌手得到其对应的加密文档结果集，如图3–2 所示：

从表3–2可知，对每一个单词，所对应的文档的结果集是唯一确定的。当搜索时，如果服务器发现该次查询在以前某次被查询过，服务器可以直接返回以前查询所记录的结果，而不管上次查询之后文档是否发生变化，这显然破坏的方案的完整性，同时可以使用访问模式推断搜索模式。此外，如果访问模式没有被隐藏，即使搜索模式被隐藏了，我们同样可以通过访问模式以很大概率

单词陷门	查找结果
$T(w_1)$	c_1, c_3, c_4
$T(w_2)$	c_2, c_3, c_5
$T(w_3)$	c_1, c_2, c_5
$T(w_2)$	c_2, c_3, c_5
$T(w_1)$	c_1, c_3, c_4

表 3–2 访问模式信息泄漏示例

Table 3–2 An Example Of Access Pattern Information Leakage

推导出搜索模式的信息。例如：5 次查询历史仍为 $(w_1, w_2, w_3, w_2, w_1)$，使用概率陷门生成算法后的结果如表3–3所示：

单词陷门	查找结果
$T(w_1)$	c_1, c_3, c_4
$T(w_2)$	c_2, c_3, c_5
$T(w_3)$	c_1, c_2, c_5
$T(w_2')$	c_2, c_3, c_5
$T(w_1')$	c_1, c_3, c_4

表 3–3 改进后的访问模式信息泄漏示例

Table 3–3 An Example Of Improved Access Pattern Information Leakage

从上表结果可知，我们仍然能通过搜索单词的结果集推断出：$T(w_1)$ 与 $T(w_1')$ 和 $T(w_2)$ 与 $T(w_2')$ 是同一单词，虽然这个结果不是绝对的，但是敌手仍能以很大概率推断出多次搜索历史的 Search Pattern。因此，仅仅隐藏搜索模式是显然不够的，同时必须在一定程度上隐藏访问模式。

综上所述，为了完全隐藏方案中的 trace 信息，构造的方案必须具有以下特点：

1. 对于同一个单词，其对应的搜索陷门每次产生应该完全不相同，即敌手不能通过搜索陷门来推断出用户的搜索模式；并且搜索后，安全搜索的条目应有所变化，使得下次搜索时不能通过安全索引的已匹配项来推断出任何信息（如搜索模式）；

2. 对于不同的单词，其对应的搜索陷门应该存在相同，即敌手不能通过输出的结果来推断用户的访问模式；同样地，对相同单词的多次搜索结果也应该有所不同；

3. 对于所有外包的文档的密文，它们的长度必须相等，以至于服务器在查询时不能通过长度来泄漏 trace 的大小模式。

而在实际中，要实现具有上述特征的方案，必须查询时更新所有信息，这样的代价是不可取的。既然不可能完全掩盖 trace 的信息，我们只能尽可能地减少其信息的泄漏，即以条件概率来泄漏 trace 的信息量。

3.1.3 前人的工作

Liu 等人在方案 [24] 中，首先证明了在通用的单关键字的可搜索加密方案中，敌手能利用少量的额外信息和用户的搜索模式发现用户搜索潜在的关键字。然后针对该问题，他们提出了一种基于分组的可搜索加密方案，该方案隐藏了搜索模式的泄漏，同时在一定程度上也避免了访问模式的泄漏，并通过实验证明了该方案的安全性。该方案由八个算法组成，其具体思路如下：

- $K \leftarrow KeyGen(1^{\lambda})$. 方案使用安全参数 λ，输出密钥 K；
- $I \leftarrow BuildIndex(D)$. 算法从文档集 D 中提取出单词与文档间索引关系 I；
- $(C, SI) \leftarrow Encryption(D, I, K)$. 该算法对文档加密和对明文索引 I 加密生成安全索引 SI，并提交两者至服务器；
- $S \leftarrow Dividing(k, W, V)$. 该算法使用参数 k、W 和 V （其中 k 表示对单词集分组后每组的元素个数，W 是文档中所有不同单词的集合，V 表示额外的知识，被数据所有者和敌手所使用），输出分组的单词集 S，过程如下：

  1. 根据 V 的频率分布，将单词集 W 排序；
  2. 将排序后的单词集分成 $|W|/k$ 组，每组 k 个元素：$S = (S_1, S_2, ..., S_{|W|/k})$；
  3. 将每组的元素 S_i（$1 \leq i \leq |W|/k$）用洗牌算法对其进行重排。

- $(b, \mathcal{Q}) \leftarrow Query(k, w, K, S)$. 用户查询单词 w，调用算法 $Dividing$ 对单词进行重新分组。然后从分组的集合 S 中，查询得到包含单词 w 的集合 S_q；对集合 S_q 中第 i 个单词建立陷门 sq_i，记 $\mathcal{Q} = \{sq_1, ..., sq_k\}$，并记录单词 w 在集合中的位置记为 b 和更新额外信息 V；保存 b 在用户端，并提交请求 $\mathcal{Q}$；

- $R \leftarrow Search(\mathcal{Q}, SI)$. 服务器收到请求 $\mathcal{Q}$ 后，对每个元素元素 $\mathcal{Q}[i]$ 查询得到结果集 R_i，并返回结果集 $R = \{R_1, ..., R_k\}$；

- $R_b \leftarrow Extract(R, b)$. 请求者收到结果集后，根据请求单词所在位置 b，提取出目的结果 R_b；

- $D_i \leftarrow Decryption(C_i, K)$. 用户使用文档加密密钥 K 解密 R_b：$D_i \leftarrow SKE.Dec(C_i, K)$ $(C_i \in R_b)$。

该方案的缺陷在于：(1) 需要了解并利用额外的知识 V，在某些情况下，这个限制条件是很难达到的；(2) 客户需要存储分组后的单词集合 S，并且需要维持集合 S 的内容在不同授权用户之间的一致性问题；(3) 服务端返回的结果非常大，包括所有请求的单词的结果 R，导致网络间的通讯量是所需答案信息量的 k 倍，同时也增加了服务端的计算开销，与分组中每组的单词个数 k 密切相关。当 k 足够大时，计算和存储开销太大，当 k 很小时，不容易隐藏用户的搜索模式。因而如何选择常数也成了本方案的关键。并且在该方案中，如果不每次更新集合 S，如：查询单词 $(w_1, w_2, w_1, w_1, w_1)$，请求陷门如下：

$$\begin{cases} \mathcal{Q}_1 & = \{sq(w_7), sq(w_1), sq(w_{15})\} \\ \mathcal{Q}_2 & = \{sq(w_2), sq(w_6), sq(w_8)\} \\ \mathcal{Q}_3 & = \{sq(w_7), sq(w_1), sq(w_{15})\} \\ \mathcal{Q}_4 & = \{sq(w_7), sq(w_1), sq(w_{15})\} \\ \mathcal{Q}_5 & = \{sq(w_7), sq(w_1), sq(w_{15})\} \end{cases}$$

从上例可知，服务器通过对上述多次请求陷门集合进行比对，照样推断出用户的搜索模式。因而，在每次查询时，都必须对单词集进行重新分组，以避免 search pattern 的泄漏。对上例重新分组后的可能结果如下：

$$\begin{cases} \mathcal{Q}_1 & = \{sq(w_7), sq(w_1), sq(w_8)\} \\ \mathcal{Q}_2 & = \{sq(w_2), sq(w_6), sq(w_8)\} \\ \mathcal{Q}_3 & = \{sq(w_7), sq(w_1), sq(w_2)\} \\ \mathcal{Q}_4 & = \{sq(w_1), sq(w_7), sq(w_2)\} \\ \mathcal{Q}_5 & = \{sq(w_2), sq(w_1), sq(w_7)\} \end{cases}$$

通过这样的改变，服务器显然不能推断出用户的搜索模式。

Islam 等人在方案 [23] 中，提出一个新颖的攻击模型，该模型利用泄漏的访问模式和先验知识来推断用户某些敏感的信息。然后针对该攻击，提出了一个避免 access pattern 泄漏的方案。但是，该方案引入了误报率和增加通讯开销。方案思路如下:

- **建立索引：**在该阶段，用一个 $m * n$ 的二维矩阵 ID 来存储安全索引 (m—单词数目，n — 文档数目)，元素 $ID_{i,j}$ 表示单词 w_i 是否在文档 D_j 中，若在为 1，否则为 0。建立阶段，每行元素通过如下方式构建：对该单词包含在某文档中的位置必置为 1，否在在其它非 1 位置随机选择几个置为 1。然后使用近似算法将 ID 转化为安全的矩阵索引;
- **查询结果：**在查找阶段，对安全索引进行查找，返回所有符合条件的结果。

显然，在该方案中，引入了一定的误报率，并且要构造如此的近似函数也不是易事；同时，在搜索结果中通过错误的结果增加了传输的通讯开销。

3.2 方案框架

3.2.1 方案相关定义

定义 3.1 (正交投影)**.** 在相同空间中，向量 v 在向量 u 上的正交投影 p，定义为:

$$p_u(v) = \frac{\phi(u,v)}{\phi(u,u)}u$$

ϕ 是一个向量内积函数。

定理 3.1 (Gram-Schmidt). 如果 $(f_i)_{i\in\{0,n\}}$ 是一个 *pre-Hilbert* 向量空间系，ϕ 是一个向量内积函数，则必存在一个正交系 $(o_i)_{i\in\{0,l\}}$ 使：

- $\phi(o_i, o_j) = \phi(o_i, o_i)\delta_{i,j}$,
- $Vect(f_1, ..., f_l) = Vect(o_1, ...o_l)$,
- 标量积 $\phi(f_i, o_i)$ 严格为正。

Notations	Explanations
p	常量，对每个文档块复制 p 份
q	常量，对每个单词构造 q 个相关联的向量
m	常量，将每个文档分成 m 块
T	查找表，存储单词陷门内容
A	数组，存储所有单词的文档 ID 信息
A_d	数组，存储加密文档的数据块
D	集合，所有的文档
$W(D)$	数组，文档集 D 中所有不同的单词集合
$ID(w)$	数组，所有包含单词 w 的文档的 ID
$A[i]$	数组中第 i 个元素

3.2.2 方案框架

在不可信的云环境下，我们实现了一个抗信息泄漏的可搜索加密方案。我们的系统模型与通用的系统模型2–1相同，包括数据拥有者、被授权的用户和云服务提供商。数据拥有者负责数据的初始化和安全索引的构建，用户搜索并解密所需文档，而云服务提供商提供存储外包服务和遵照用户规则对外包数据进行检索并返回结果。方案思想如下：

1. 在建立索引阶段，对每个单词的处理如下：将每个单词转换向量；
2. 在查找阶段，单词查找口令按如下方式生成：将该单词按同样方式生成向量 V，然后构建向量的法平面，取法平面内的任意向量，由法平面内的任一向量与向量 V 垂直（），可构造出类似的概率算法。

该方案由七个算法组成，如下所示：

- **密钥生成：**数据所有者使用安全参数，生成解密文档和建立索引所需要的密钥；
- **文档加密：**数据拥有者对将上传的文档分块、加密并建立各块之间的联系，对每篇文档生成一个链表，同时使服务器能通过起始块恢复出完整的文档；
- **安全索引建立：**数据所有者提取待上传的文档的关键字信息，对关键字建立安全索引，并建立安全索引与加密文档之间的关联；
- **外包：**数据所有者将安全索引和加密文档上传至云服务器；
- **陷门生成：**当被数据拥有者授权的用户想要查找包含某单词的文档时，调用安全陷门生成函数计算单词的陷门，发送给服务器；
- **搜索：**一旦云服务器收到用户所提交的单词查找陷门时，使用单词陷门在安全索引上检索得到文档 ID，通过文档 ID 查找获得文档首块地址，按首块地址得到文档的所有块，回送给用户；
- **解密：**用户对收到的所有加密文档块进行解密，扔掉无用的块和恢复出完整的文档。

3.3 方案细节

在已有方案和技术的基础上，我们构造了一个抗信息泄漏的可搜索加密方案，该方案具有如下特征：没有泄漏大小模式，仅以概率泄漏访问模式和搜索模式，并与已有方案有相同的安全级别。

定义 3.2 (抗信息泄漏的可搜索加密方案)**.** 该方案包括七部分：($Setup$，$DEnc$，$BuildSIndex$，$Outsource$，$TrapdrGen$，$Query$，Dec)，其数学描述如下：

- $K \leftarrow Setup(1^k)$**.** 方案中该算法是简单的密钥生成函数。单词加密密钥从 $\{0,1\}^k$ 中随机取样，文档加密密钥调用 $SKE.KeyGen(1^k)$ 生成，生成包括加密文档和建立安全索引的密钥 K；将每个文档
- $A_d \leftarrow DEnc(D, K)$**.** 该算法使用参数 D 和 K 对文档进行分块，分别加密并存放至数组 A_d 中且维护各块之间的关联；

- $I \leftarrow BuildSIndex(D, K)$**.** 索引建立算法使用参数 D 和密钥 K，提取文档中单词 $W(D)$ 并建立安全索引同时维持单词与文档之间的关系，即通过单词能快速定位到包含单词的文档 ID;

- $Outsource(A_d, I)$**.** 该过程仅仅将加密的结果 A_d 和 I 外包至远程的服务器;

- $T_w \leftarrow TrapdrGen(w, K)$**.** 陷门生成函数是一个概率算法。主要由授权的用户所调用，用于生成查询陷门。在该方案中，算法使用同一单词 w 和 K，但每次输出不同的陷门 T_w;

- $R \leftarrow Query(w, I, A_d)$**.** 该过程是确定性的查询算法，包括使用单词陷门对安全索引进行查找获得所有文档的首块地址，然后通过该地址对文档信息进行查找获得完整的文档块结果集合 R，并返回;

- $PD \leftarrow Dec(R, K)$**.** 是确定性的解密算法，对收到的 $R_i \in R$ 中的每块调用算法 $SKE.Dec$ 进行解密，并将解密的各块连接起来形成完成的文档 PD，即为正确的文档。

从上述定义可知，$Setup$ 是简单的密钥生成算法，$Outsource$ 仅仅将数据外包，这两个算法与其它方案中算法一致，无改进之处。下面分四个过程（加密、陷门生成、查询和解密）对其它算法进行描述。其中，加密包括对文档 D 进行加密和建立查找结构 I。

3.3.1 加密

首先数据所有者调用密钥生成函数 $Setup$ 生成密钥 $K = (K_1, K_2, K_3, K_4, SK_1, ..., SK_p)$；用于对文档进行加密和建立安全索引。

文档加密（$DEnc$） 首先对所有文档 $D = (D_i, ..., D_n)$ 进行填充，使所有文档具有相同的大小。然后将每个文档 $D_i \in D$ 划分成 m 个大小相同的块 $D_i = (B_1, ..., B_m)$，并在每个文档块的后面加一个代表该块所在文件中位置的标记，如加上标记后的文件块为：$B = (B_1||1, ..., B_m||m)$；紧接着，将标记后的文件块复制 p 份并随机洗牌，然后对每块分别用加密函数 $SKE.Enc$ 和密钥 SK_i 进行加密，加密后放入数组 A_d 中，并将所有的块用一个链连接起来，第 i 个数据块在数组 A_d 结点中的信息如下:

$$A_d[addr_D(B[i])] =< SKE.Enc_{SK_i}(B_i||i), addr_D(B[i+1]) >,$$

这里 $addr_D(B[i])$ — 表示 B 中第 i 块在数组 A_d 中的地址。若取 p=2，将每个文档分成三块，则在数组 A_d 中对每个文档分块并加密建立关联后的结构如图3–1所示：

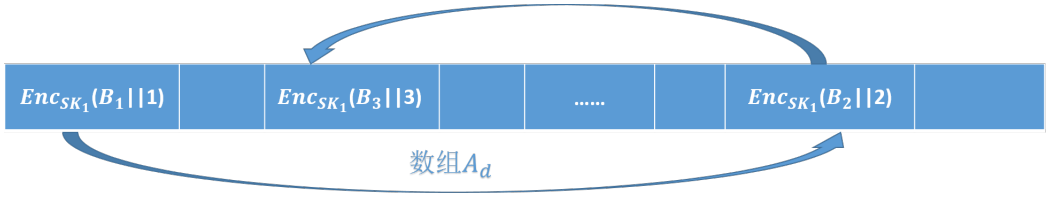

图 3–1 加密后文档块链表

Fig 3–1 Document Block List of Encryption

索引建立（$BuildSIndex$）首先对所有文档集 D，提取出 D 中所有不同的单词集合 $W(D) = \{w_1, ..., w_{|W(D)|}\}$；然后对每个单词 $w_i \in W(D)$，使用伪随机置换函数 π_{K_1} 生成一个置换后的单词集合 $PW = \{\pi_{K_1}(w_1), ..., \pi_{K_1}(w_{|W(D)|}\}$。

随后，我们选取 $|W(D)|$ 个 $l-2$ 维的标准化正交基（正交基的建立过程可以使用史密斯正交化理论构建 [46]）：$\{e_1, ..., e_{|W(D)|}\}$，$|e_i| = l-2$，$i \in [1, |W(D)|]$；然后将每个置换后的单词与对应的正交基拼接起来，则每个单词 w_i 的结构如下：$\pi_{K_1}(w_i)||e_i$，且每个结构的大小为 $l-1$ 维；再分别将集合 $[1, q]$ 中每个数与变化后的单词结构相连接，则最终对每个单词 w_i 形成 q 个 l 维的空间向量，对 $1 \le i \le |W(D)|$ 和 $1 \le j \le q$，其结构为 $f_{w_i,j} = \pi_{K_1}(w_i)||e_i||j$。对每个变量 $f_{w_i,j}$，使用伪随机函数 F_{K_2} 生成一个结果放入查找表（look-up）T 中，在 T 中每个结点存储的内容如下：

$$T[F_{K_2}(f_{w_i,j})] =< f_{w_i,j}, addr_{ID}(ID(w_i)[1]) > \oplus \mathcal{G}(P_{K_3}(f_{w_i,j})),$$

这里 F_{K_2} 和 P_{K_3} 是伪随机函数，$\mathcal{G}$ 是伪随机生成器，$addr_{ID}(ID(w_i)[1])$ 是指单词 w_i 所对应的文档 ID 集合在文档 ID 数组 A 中的第一个文档 ID 所存放的地址。

对每个单词 $w_i \in W(D)$，建立文档 ID 的集合 $ID(w_i) = \{ID(D_i)|w_i \in D_i\}$，并填充是所有的集合 $|W(D)|$ 大小相等；然后对每个 $ID(w_i$ 建立链表 L_{w_i}，使之在 $ID(w_i)$ 中每个元素（$1 \le j \le |ID(w_i)|$）在数组 A_s 中的结构如下：

$$A[addr_{ID}(ID(w_i)[j])] = (< ID(w_i)[j], addr_{ID}(ID(w_i)[j+1]) > \\ \oplus H(Q_{K_4}(f_{w_i,j}), r_i), r_i),$$

这里 r_i — 随机字符串，$ID(w_i)[j]$ — 表示集合 $ID(w_i)$ 中的第 j 个元素，$addr_{ID}(ID(w_i)[j])$ — 表示元素 $ID(w_i)[j]$ 存放在数组 A 中的地址，Q_{K_4} — 表示伪随机函数，H — 表示伪随机置换。

如 $q = 2$，对包含三个单词的文档，按照上述方式，构建完成后的安全索引结构如下图3–2 所示：

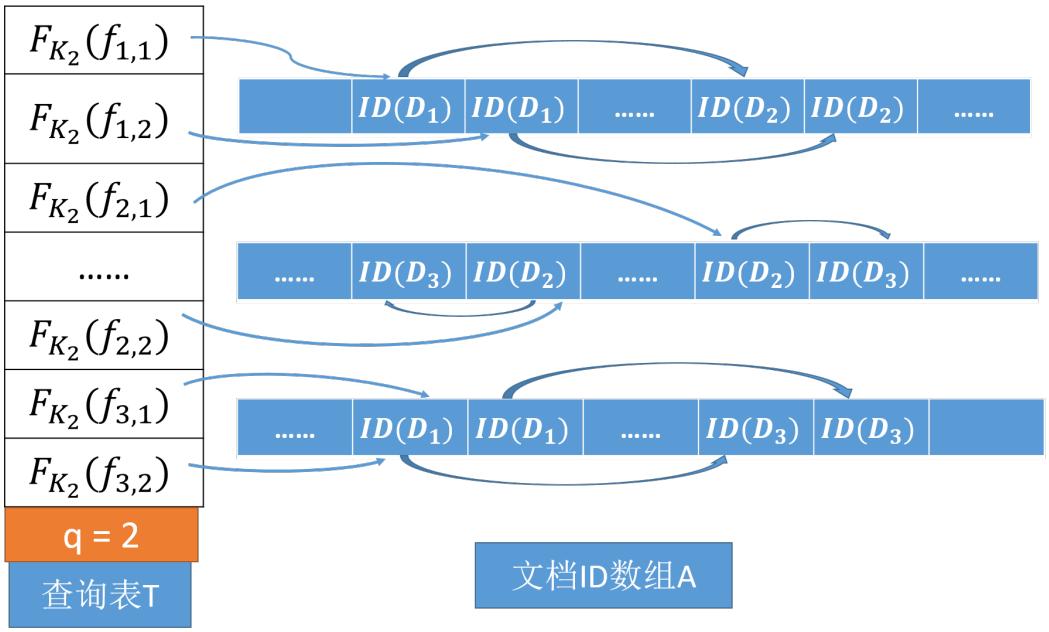

图 3–2 抗泄漏方案的安全索引

Fig 3–2 Secure Index of Resisting Information Leakage

数据拥有者在加密过程中主要使用了两个函数：文档加密函数 $DEnc$ 1和安全索引建立算法 $BuildSIndex$ — 包括两个子算法：$BuildLookup$ 2与 $BuildArray$ 3，其具体实现过程分别如下：

3.3.2　陷门生成

当授权的用户使用单词 w 查询时，首先调用函数 $BuildLookup$ 得到该单词对应的 $l-1$ 维的向量 $t(w)$，然后从集合 $[1,q]$ 中随机选取一个数 N，并将两

Algorithm 1 $A_d \leftarrow DEnc(D, K)$

Steps:

// Initialization

allocate enough array: A_d

make each $|D_i|$ equal by filling

// Build List for D

1: divide each $D_i \in D$ into m blocks: $D_i = (B_1, ..., B_m)$

2: add unique location flag to D_i, get: B =$(B_1||1, ..., B_m||m)$

3: make p copies of B, and randomly shuffle for each one

4: **for** $1 \leq j \leq p$ **do**

5: **for** $B_i \in B$ **do**

6: create a node: $A_d[addr_D(B_i)] =< SKE.Enc_{SK_j}(B_i), addr_D(B[i+1]) >$, where $addr_D(B_i)$ is randomly unique address. for the last block, having: $addr_D(B_m) = NULL$

7: **end for**

8: **end for**

9: **return** A_d;

者连接起来形成一个 l 维的向量 $f_{w,N} = (t(w)||N)$；对向量 $f_{w,N}$，构建其 l 维的法平面方程 M [47]，满足：

$$M(x_1, x_2, ..., x_l) = 0 \quad (x_i \text{ 为第 } i \text{ 维空间的标量值}),$$

在法平面 M 上，我们随机选择一个与 $f_{w,N}$ 垂直的向量 $V_{f_{w,N}}$，计算其陷门 T_w，包括：$T_w = (F_{K_2}(f_{w,N}), P_{K_3}(f_{w,N}), Q_{K_4}(f_{w,N}), V_{fw,N})$，然后将陷门提交给云服务端进行搜索。具体的描述如算法 4。

3.3.3 查询

一旦收到授权用户发送过来的请求陷门后，云服务器将进行查询并返回结果给请求者，过程如下：

1. 搜索查找表 T：解析陷门 T_w，在 look-up 中查找获得文档 ID 的入口地址；

Algorithm 2 $T \leftarrow BuildLookup(D, K)$

Steps:

// Initialization

allocate enough look-up table T

select orthogonal basis $\{e_1, ..., e_{|W(D)|}\}$ of $l-2$ dimensions

randomly select constant q

// Build Look-up Table T

for $w_i \in W(D)$ **do**

  generate: $t(w_i) = \pi_{K_1}(w_i)||e_i$, $|t(w)| = l-1$

  for $1 \leq j \leq q$ **do**

    get: $f_{w_i,j} = t(w_i)||j$

    for $f_{w_i,j}$, create a node:

    $T[F_{K_2}(f_{w_i,j})] =< f_{w_i,j}, addr_{ID}(ID(w_i)[1]) > \oplus \mathcal{G}(P_{K_3}(f_{w_i,j}))$

  end for

end for

fill remaining blanks to random value.

return T;

2. 查找数组 A：通过文档 ID 首地址得到所有文档数据块的首地址；

3. 查找数组 A_d：逐个对每个文档数据块的首地址遍历直至完成，最终得到所有的文档数据块集合；

4. 返回：将查找得到的所有结果返回给请求者。

算法5使用伪代码描述了 Query 的具体实现。

3.3.4 解密

若授权用户收到远程服务器回送的响应结果 R 后，对其进行解密，并逐步将各个数据块连接起来丢弃无用的数据块，形成完整的文档，得到用户所需求的答案 PD。解密算法 Dec 6描述如下：

Algorithm 3 $A \leftarrow BuildArray(D, K)$

Steps:

// Initialization

get $ID(w_i)$ of $w_i \in W(D)$

assure all $|ID(w_i)|$ equal

allocate enough array A

// Build Array A

for $w_i \in W(D)$ **do**

  for $1 \leq j \leq |ID(w_i)$ **do**

    create a node in A: $A[addr_{ID}(ID(w_i)[j])] =$
    $< ID(w_i)[j], addr_{ID}(ID(w_i)[j+1]) > \oplus \mathcal{H}(Q_{K_4}(f_{w_i,j}, r_i), r_i)$,
    and for last node: $addr_{ID}(ID(w_i)[|ID(w_i)|]) = NULL$.

  end for

end for

fill remaining entries of A to random value

return A;

3.4 安全性证明

基于系统模型2–1，我们选择常用的证明模型即 IND-CKA（Choosen Keyword Attatck），对查询各阶段的过程有所修改，游戏过程模拟如图3–3所示。然后证明敌手在多项式时间内仅能以 1/2+$negl(k)$ 的概率猜出游戏答案。

定理 3.2 (IND_CKA Security). *如果方案中所使用的函数 F、P 和 Q 是伪随机函数，H 和 ϕ 是伪随机置换，$\mathcal{G}$ 是伪随机生成器，并且 SKE 是 $PCPA$ 安全的，则该方案是 non-adaptive 不可区分安全的。*

证明. 该部分将使用 Curtmola 方案中提出的 Non-adaptive 不可区分游戏模型对方案的安全性进行证明。

在游戏过程中，首先敌手 $\mathcal{A}_1$ 发送的历史记录 H_0 和 H_1，满足 $|H_0| = |H_1|$ 即 $|D0| = |D1|$ 且 $|w0| = |w1|$，然后挑战者（Challenger）在过程中随机选择参数 b，敌手 $\mathcal{A}_1$ 收到记录后能访问多项式敌手 $\mathcal{A}_2$，但在猜测过程中，仅能使用挑战者获得的知识。因而，我们仅仅需要判断敌手对挑战者发起的内容在多项式时间不可区分，从如下几个方面分析：

Algorithm 4 $T_w \leftarrow TrapdrGen(w, K)$

Steps:

// Initialization

randomly select number N from set $[1, q]$

get e_i following algorithm $BuildSI$

// Build Array A

for $w_i \in W(D)$ **do**

  for $1 \leq j \leq |ID(w_i)$ **do**

    create a node in A: $A[ID(w_i)[j]] =< ID(w_i)[j], addr_{ID}(ID(w_i)[j+1]) > \oplus \mathcal{H}(Q_{K_4}(f_{w_i,j}, r_i), r_i)$

    and for last node: $addr_{ID}(ID(w_i)[|ID(w_i)|]) = NULL$

  end for

end for

fill remaining entries in A to random value

return A;

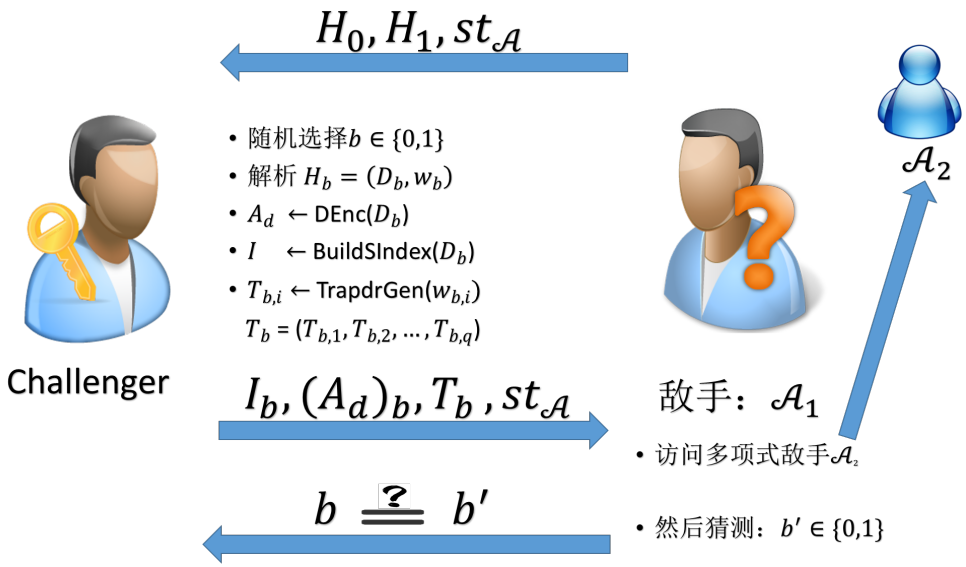

图 3–3 修改后的方案游戏模型

Fig 3–3 New Security Game Model

- $Setup$：挑战者 C 随机生成密钥 K;

Algorithm 5 $R \leftarrow Query(T_w, I, A_d)$

Steps:

// **Visit Look-up Table** T

parse T_w as: $T_w = (t_1, t_2, t_3, t_4)$

set: $T[t_1] \oplus \mathcal{G}(t_2) = (a_1, a_2)$

// **Visit Array** A

if $a_1.t_4 == 0$ **then**

  init set: $ID = \varnothing$

  get: $(id, a_2) = A[a_2][1] \oplus \mathcal{H}(t_3, a[a_2][2]))$

  put id into set ID

  if $a_2 \neq NULL$ **then**

    goto: 4

  end if

  init set: $R = \varnothing$

  // **Visit Array** A_d

  for $1 \leq k \leq |ID|$ **do**

    get block set: R_k by traversing $ID[k]$ in A_d

    put R_k into R

  end for

else

  exit;

end if

return R;

- $DEnc$：当 C 收到敌手 $\mathcal{A}_1$ 发送过来的历史信息 H_0 和 H_1 后，随机选择参数 b，然后解析历史 H_b 并对文档加密生成 $(A_d)_b$，并送至敌手；
- $BuildSIndex$：同样地挑战者收到挑战的历史记录后，对 D_b 建立安全索引 I_b；
- $TrapdrGen$： 挑战者 C 对 w_b 中每个单词，计算对应的陷门，并送至敌手 $\mathcal{A}_1$。

在上述所有过程中，挑战者仅仅使用了安全的伪随机过程和 $PCPA$ 安全

Algorithm 6 $PD \leftarrow Dec(R, K)$

Steps:

1: init: $PD = \varnothing$
// **Analyze each set** R_i **in** R
2: **for** $1 \leq i \leq |R|$ **do**
3: get corresponding SK_N of $f_{w,N}$
4: **for** $1 \leq j \leq |R[i]|$ **do**
5: decrypt: $b_j = SKE.Dec_{SK_N}(ED[i][j])$
6: **end for**
7: sort: $p_1, p_2, ..., p_m$ in actual block sort
8: get document: $D_j = p_j[1]||p_2[1]||...||p_m[1]$
9: throw dummy strings of D_j, put D_j to PD
10: **end for**
11: **return** PD;

的对称加密方案 SKE，它们仅都在多项式时间内以不大于 1/2+$negl(k)$ 的概率进行区分。因此，当敌手收到 C 发送过来的信息后，即使敌手 $\mathcal{A}_1$ 有能力访问算法 $\mathcal{A}_2$，仍不能猜中 $b' = b$ 的概率大于 1/2。所以，我们的方案是 non-adaptive 不可区分安全的。

另外，我们的方案在搜索时，一个单词对应有 q 个不同的单词陷门，并且同一个单词陷门访问文件块时有 p 种情况和相同文档中一块指向下一对应块的连接是随意的（每个文件块包含 p 份），故敌手能推断出搜索模式的概率仅为 $q * 2^p$。因不同的单词同样含有相同的文档，故能推断出搜索模式的概率小于 $q * 2^p$。

□

3.5 性能分析

到此，我们已经完成了整个方案的功能，不仅能避免 size pattern 的泄漏，以概率泄漏搜索模式和访问模式，同时也通过严格的分析证明了我们方案在我们的模型下是安全的。下面我们从三个方面对我们方案的性能进行分析 — 存储开销、计算开销和传输开销。

3.5.1 存储开销

为了减少更多的信息泄漏，方案不得不引入额外的存储开销。下面我们从两个方面分析方案的存储性能:

1. **索引结构：**在方案中，我们使用查询表 T 和数组 A 来存储安全索引。在建立查询表过程2中，对每个单词存储 p 份，因而 T 的存储空间大小为: $W(D)*q*|T[i]|$ （$T[i]$ — 表示查询表中每个结点所需要的大小)。在文档 ID 数组的建立过程 3 中，对每个单词 w，存储被文档包含的 ID 数组，并且通过填充使每个 ID 数组大小保持相等，因而数组 A 的存储空间大小至少为: $W(D)*ID(w_i)*A[i]$（$w_i \in W(D)$，$A[i]$ — 表示数组每个结点所占用空间)。

2. **文档结构：**在加密文档过程1中，首先对每个文档进行填充使大小为文档的最大大小并分块，并对每块复制 p 份，因而文档块数组 A_d 的大小至少是: $|D_i|*p$。

3.5.2 计算开销

我们从客户加密、陷门生成和服务端查询三个方面对方案的计算性能进行分析。

1. **加密文档：**在数据所有者外包数据之前，首先对文档进行加密，涉及到建立索引和文档加密。在文档加密阶段，客户的计算开销花费在对数据分块是调用函数 SKE.Enc 进行简单的加密处理，计算时间为: $|D|*m*T(SKE.Enc)$（$T(SKE.Enc)$ — 表示算法 SKE.Enc 加密所使用的时间)。索引建立的时间主要包括生成斯密斯正交基和建立安全索引，其计算时间开销为: $T(s)*|W(D)|+|W(D)|*q*ID(w_i)$ （$w_i \in W(D)$，$T(s)$ — 表示生成正交基所花费的时间)。

2. **陷门计算：**在陷门生成算法计算过程4 中，首先需要获得待查寻单词 w 的一个正交基（计算过程与算法 BuildLookup 相同)，然后基于该向量构建该向量的法平面，因而客户端的计算开销为: $T(s)+T(M)$，这里 $T(s)$ 和 $T(M$ 分别表示生成单词 w 的基和该基向量的法平面所需要的时间。

3. **文档查询：**在算法 Query5过程中，服务器需要分别遍历安全索引和文档块数组，这里假设在数据 A 和 A_d 中单步的计算开销为 $O(1)$，则总时间计算开销为：$|ID(w_i)| * m$ （$w_i \in W(D)$，m — 表示每个单词所查找到的所有文件块总大小)。

3.5.3 传输开销

传输开销主要发生在客户与服务器之间的通讯阶段，因而，我们可以从上传文档、发送搜索陷门和返回检索结果这三个方面来分析网络的负荷。

1. **文档上传：**在文档上传阶段，客户上传至服务端的信息包括索引结构 I 和加密文档结构，因而该次通讯开销的总大小为：$|A| + |T| + |A_d|$。

2. **搜索陷门提交：**在该阶段，用户上传的陷门内容仅仅包括待查单词的几个伪随机函数，因而我们可以简单地将通讯时间看作为 $O(1)$，与其它方案的开销保持在同一个级别。

3. **检索结果返回：**在服务器返回符合搜索条件的结果至客户过程中，传输的信息量记为查询过程所得的结果大小，因而传输开销的大小同样为：$ID(w_i)| * m$ $(m \approx |D_i|)$。由于在每次用户查询过程中，都需要计算和传输该数据，因而应该尽可能地减小其大小。

第四章　同义词对称可搜索加密

到目前为止，对称可搜索加密技术已得到广泛的研究，各种复杂的条件搜索加密技术也被深入的研究并取得了实质性的成果。然而在复杂的多变的云计算环境下，我们需要研究的知识点范畴广和面对的用户群里基数大。为此，我们不得不进一步挖掘出一些尚未提出并具有实际意义的问题。在前人的工作指导下，本文解决了相似搜索的另一个场景 — 同义词搜索。

在本章，我们提出了一个支持同义词搜索并具有强安全性的对称可搜索加密方案。为此，我们首先定义了方案的系统模型和攻击模型；然后定义其算法框架；针对方案框架的各个算法，逐个给出它们的详细实现细节；最后我们对该方案进行了严格的安全性证明和性能分析。

4.1　方案模型

在该小节，我们从已有的方案中抽象出将研究的问题；然后针对我们的问题，提出了相应的系统模型和有效的攻击模型；为了实现方案的详细细节，我们引入一些相关的符号及其描述，同时定义了方案的信息泄漏；最后，我们简略地描述了方案的基本框架。

4.1.1　问题提出

可搜索加密技术解决了外包数据的安全和高效搜索的难题。然而，在互联网快速发展的时代下，人们在工作中的强度日益增加，致使人们的记忆力也随着超载工作量的过度消耗而呈现下降趋势。由于人们记忆能力的周期的缩短和人员频繁的变换，使得一段时间之后，他们对之前使用过的文档及其内容的记忆已模糊不清甚至完全遗忘。此情景的一个典型的示例如图4–1所示。首先，使用云计算平台服务的数据所有者将文档外包至不可信的云端；一段时间后，数据所有者想要查找包含“文件”含义的关键字的文档时，其可能因忘记不知道文档中到底包含“paper”还是“document”？为此，用户不得不逐个尝试所有具有该含义的单词，直至找到所需的答案为止。在按计算收费的时代，这显然是不合理的，同时也增加了网络带宽和用户的响应时间。

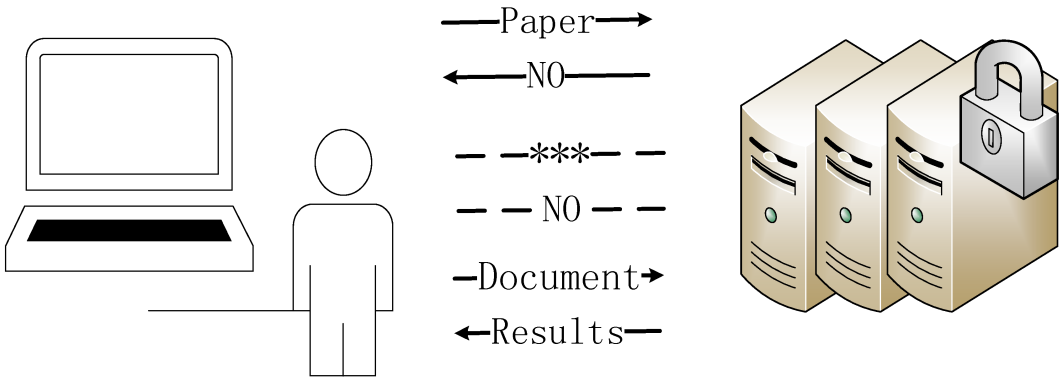

图 4–1 同义词可搜索加密方案的示例

Fig 4–1 An Example of the Synonym Search

4.1.2 系统模型

在我们的方案中，支持同义词搜索的系统模型（如下图4–2所示）包括三部分 — 数据所有者（Owner），授权用户（Users）和云服务提供商（Provider）。这三部分在系统中分别承担不同的角色。数据拥有者是数据的源头和拥有系统的抉择权，授权用户是系统中最频繁的访问者，而云服务提供商是系统的服务提供者和数据的响应者。

1. **数据所有者：** 数据所有者是系统的主体，拥有系统中最核心的部分 —数据。数据拥有者往往由一个公司、组织或团体组成。同时，数据拥有者需要考虑系统的性能、功能和其它各种因素。一般情况下，当数据拥有者想利用外包端的便利服务时，需先购买云服务，将所有的数据首先加密，然后存储到云服务提供商。

2. **用户：** 用户是系统的使用者，往往是系统中最大的人群（包括数据所有者）。用户往往不考虑系统的任何细节，只希望利用系统提供的便利和安全性。通常情况下，如果一个用户想要使用系统的查询功能，首先他必须征求数据拥有者的允许 —即获得请求数据的安全密钥，成为合法的请求者；然后利用安全密钥生成单词陷门，并提交任务给云端；最后，在收到响应结果后，对其进行解密，获得所需的文档。在我们系统中，返

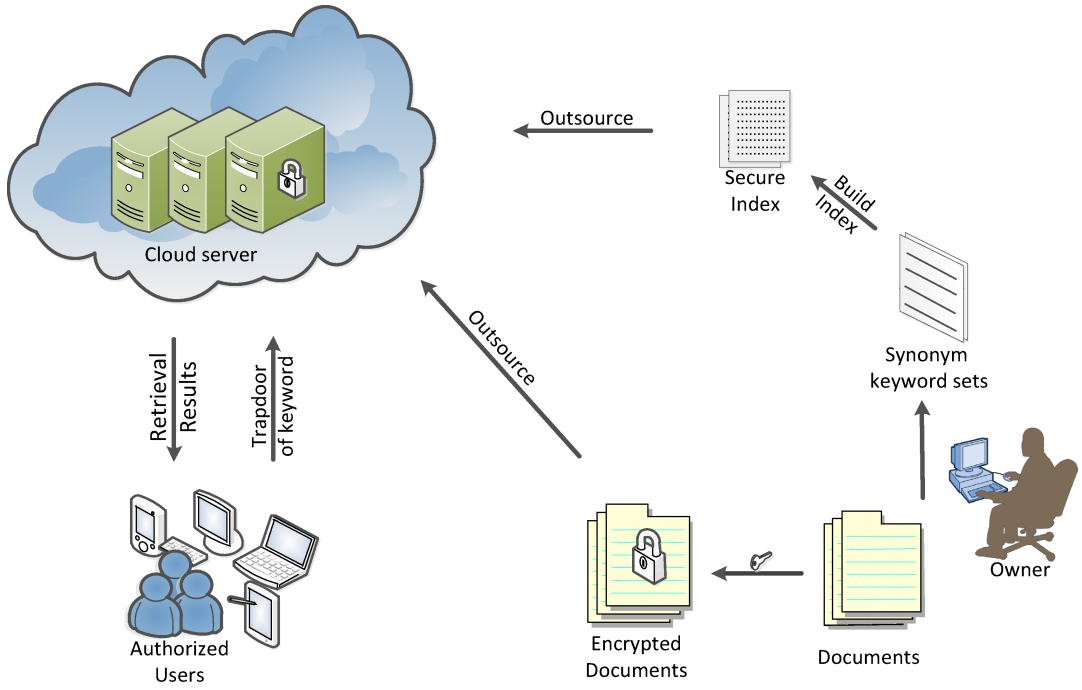

图 4–2 同义词可搜索加密系统模型

Fig 4–2 The System Model of Synonym Search

回的结果包括所有包括该单词和与它有相同含义的单词的文档。

3. **云服务提供商：** 云服务提供商是系统的客体 — 系统功能服务者，用于对外包数据进行存储和计算，是系统必不可少的一部分。通常情况下，首先接收数据所有者提交至云端的加密数据和索引，并提供对数据的计算和备份工作。当收到用户提交的查询请求时，然后利用强大的计算能力对数据进行查询操作，并将匹配的结果返回给请求者，若不存在则返回空。

在同义词可搜索加密方案中，云服务提供商返回的结果集包括两部分：(1) 包含待查单词的文档；(2) 包含与待查单词有相同含义的单词的文档。即返回的信息可描述为：$ED = \{ED_i | w \in D_i, w \in S_w\}$。并且证明我们方案具有 non-adaptive 安全。

4.1.3 攻击模型

在我们的系统中，若云服务提供商是安全并且诚实的，因而我们系统中外包的数据是安全的 — 仅使用了安全的伪随机函数和 $SKE.Enc$ 对数据加密。不幸的是，通常来说云服务提供商是“honest-but-curious” — 即遵照系统所定义的协定进行操作，同时因对数据好奇而暗地里偷偷地分析用户查询的数据，然后根据一些其他知识（例如单词的统计信息）来推断用户所搜索的真实单词，这足以说明以上方案仍存在一定的信息泄漏。为此，我们必须精确地量化出系统信息泄漏的大小，通常将云服务器视为具有最强计算能力的敌手，能分析出的数据即为系统的信息泄漏量。为了衡量我们所定义信息泄漏的正确性，引入了安全的分析模型，以证明方案除了泄漏必不可少的信息之外，不泄露其他任何的信息。为了证明方案具备 Non-adaptive 安全，假设 $\mathcal{A}$ 是多项式的敌手，$\mathcal{S}$ 是模拟者，方案的攻击模型如下:

Real$_{SSE,\mathcal{A}}(k)$

$K \leftarrow SSGen(1^k)$

$(st_{\mathcal{A}}, H) \leftarrow \mathcal{A}(1^k)$

parse H as (D, w)

$(SI, KED) \leftarrow SSEnc(D, K, SD)$

for $1 \leq i \leq q$

$T_{w_i} \leftarrow SSTrapdoor(w_i, K)$

let $T_w = (T_{w_i}, ..., T_{w_q})$

output $V = (SI, KED, T_w)$ and $st_{\mathcal{A}}$

Sim$_{SSE,\mathcal{A},\mathcal{S}}$

$(H, st_{\mathcal{A}}) \leftarrow \mathcal{A}(1^k)$

$V^* \leftarrow S(\tau(H))$

output V^* and $st_{\mathcal{A}}$

4.1.4 相关定义

- $[n]$ — 表示 n 个元素的集合，等价于 $\{1, ..., n\}$。
- D — 所有待外包的文档的集合，n 个文档的集合 $D = (D_1, D_2, ..., D_n)$。
- $ID(D)$ — 文档 D 的 ID 信息，可以用数字表示也哈希值来唯一表示。
- ED — 加密文档的结合，n 个密文文档的集合 $ED = (ED_1, ED_2, ..., ED_n)$。
- KED — 所有加密文档键值对 $< ID(D_i), ED_i >$ 的集合（$ED_i \in ED$）。

- $W(X)$ — 结构 X 中所有不同单词所组成的集合，表示为：$W(X) = (w_1, w_2, ..., w_{|W(X)|})$。
- D_w SD_w — D_w 是仅包括单词 w 的所有文档的集合；SD_w 是包括单词 w 和其同义词的文档的集合。
- SD— 同义词字典的集合，它必须包含集合 $W(D)$ 中的所有单词。根据所定义环境的不同，集合中元素也不相同。m 个元素的集合 SD，表示为：$SD = (w_1, ..., w_m)$。
- S_w — 单词 w 的所有同义词的集合，描述为：$S_w = (w_1, w_2, ..., w_{|S_w|})$。
- SID_w — 包含单词 w 的文档的 ID 集合，定义 $SID_w = \{ID(D_i) \mid w \in D_i\}$，$SID = \{SID_w \mid w_i \in W(D)\}$。
- SI — 安全索引结构，由数据拥有者生成，并存储在云服务端。在我们系统中使用数组和查询表来维护。
- T_w — 单词 w 的陷门信息。它使用伪随机函数生成，用于在搜索过程中作文查询口令。
- ESD_w — 用户在查询后，由服务器返回的加密的文件集合。

同义词函数（Synonym Function） 对于任意给定的两个单词 w_1 和 w_2，定义：

$$SF(w_1, w_2) = \begin{cases} 1 & \text{if } w_1 \text{ and } w_2 \text{ is synonym} \\ 0 & \text{otherwise} \end{cases} \tag{4–1}$$

在上述公式中，如果输出结果为 1，则称单词 w_1 和 w_2 相似，称函数 SF 为相似判断函数（简称相似函数）。

信息泄漏 在我们的方案中，我们主要分析我们所定义场景下的信息泄漏情况，我们定义该方案信息泄漏包括文档大小、访问模式和搜索模式。这里我们不考虑文档大小，仅从搜索模式和访问模式信息泄露进行分析。

查询历史： 具体定义请参见2.6。

搜索模式： 给定文档集 D 和 q 次查询历史 H，定义搜索模式如下:

$\sigma(H) = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,q} \\ x_{2,1} & x_{2,2} & \dots & x_{2,q} \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ x_{q,1} & x_{q,2} & \dots & x_{q,q} \end{bmatrix}$，其中 $x_{i,j}$ 可能取值 0、1、2。若 $x_{i,j}$ 为 0 — 表示 w_i 和 w_j 为不相等且没有相同的同义词集合 S_w；取值 1 — 表示 w_i 和 w_j 不相等但是有相同的同义字集合；取值 2 — 表示 w_i 和 w_j 为相同的单词。

访问模式： 对于给定文档集 D 和 q 次查询历史 H，定义访问模式 $\partial(H) = (SD(w_1), SD(w_2), ...SD(_q))$，$SD(w_i) = \{D(w_j) \mid w_j \in S_w\}$。

Trace： 对于给定大小为 n 的文档集 D 和 q 次查询历史 H，定义其 Trace 信息为元组：$\tau(H) = (|D_1|, ..., |D_n|, \sigma(H), \partial(H))$。

4.1.5 方案描述

下面我们简单地描述我们方案:

Synonym Searchable Encryption（SSSE）： 方案包括由 5 个基本的算法组成，定义：$SSSE = (SSKeyGen, SSEnc, SSTrapdoor, SSSearch, SSDec)$。

$\{K\} \leftarrow SSKeyGen(1^k)$： 是一个概率密钥生成算法，输入安全参数 k 和输出密钥 K。

$\{SI, KED\} \leftarrow SSEnc(D, K)$： 是方案中主要的加密模块，输入参数文档集合 D 和密钥 K，输出安全索引 SI 和加密文档 KED。主要包括文档加密函数（$SSDEnc$）和安全索引建立函数（$SSBuildSI$）。

$\{T_w\} \leftarrow SSTrapdoor(w, K)$： 是一个确定性的算法（可能是概率算法)。对于一个给定的单词 w，在密钥 K 下，生成陷门 T_w。

$\{ESD_w\} \leftarrow SSSearch(T_w, SI, KED)$**:** 是一个确定性算法。输入待查询单词 w 的陷门 T_w、SI 和 KED，查询并输出同义词结果集 ESD_w。

$\{PSD_w\} \leftarrow SSDec(ESD_w, K)$： 是一个确定性的算法，用于解密。使用返回的结果集 ESD_w 和加密密钥 K，并输出解密的文档 PSD_w。

4.2 算法框架及细节

在描述我们的方案之前，仅仅考虑一个简单的情况 — SD 中每个单词仅存在一个含义。基于这样的假设，我们设计了一个支持同义词搜索的解决方案并描述方案中各算法的详细实现流程。然后我们阐述了如何将我们方案应用于一词多义的情形并分析我们方案的优势。在详细描述我们方案之前，我们重新回顾同义词字典和同义词集合的概念。

关于 SD 的扩展： 为了使该方案更具通用性，方案中定义同义词字典 SD 为按字典排序的所有的单词集合。它不仅包括 $W(D)$ 中的所有单词，而且包含与文档中单词有相同含义的所有单词。考虑此情况的原因在于：如果我们外包有单词“Paper”但不包含单词“Document”的文档集合 D 至远程服务器；一段时间后，授权用户键入单词“Document”去查找，由于键入没有命中，服务器不得不返回无效的结果集。此时，用户不得不逐个地尝试所有具有相同含义的单词，找到所要查询的文档未知。这将大大降低方案的灵活性和可通用性。另外，用户通常希望在不同的环境下有不同的同义词集合，通常我们可能根据某领域内用户的查找知识来动态建立同义词字典。例如，在医疗行业，同义词字典的集合与其它领域可能有所不同，我们可以根据客户的要求来建立 SD。这样定制的同义词集合将使我们方案的查询性能更好和返回的结果集更有意义。

同义词集合的构建：通常情况下，同义词集合仅仅考虑文档中所有不同单词集合 $W(D)$，而忽略了其它单词。在我们构建的方案中，基于同义词字典，建立同义词集合如下：

1. 对每个单词 $w \in W(D)$，初始化单词的同义词集合 $S_w = \{w\}$；
2. 然后对每个单词 $w_i \in \{SD \setminus w\}$，计算 $SF(w, w_i)$，如果结果为 1，则插入单词 w_i 至集合 S_w，否则跳过它；
3. 最后定义文档的同义词集合 $SWS = \{S_w | w \in W(D)\}$。

4.2.1 框架详细描述

同义词对称可搜索加密： 定义同义词对称可搜索加密方案由五个算法组成即 $PSSSE = (SSKeyGen, SSEnc, SSTrapdoor, SSSearch, SSDec)$。

$SSKeyGen$ 用于生成密钥；$SSEnc$ 算法由三个子算法（$SSInitSets$ — 用于初始化文档集合，$SSDEnc$ — 加密文档，$SSBuildSI$ — 对文档建立安全索引）组成；$SSTrapdoor$ 生成单词的陷门；$SSSearch$ 通过陷门查找结果集；$SSDec$ 仅仅是个简单的解密过程。

在详细地描述我们的方案之前，这里首先定义一些将被使用的函数和数据结构，如下：

- 函数 F、G 和 P 是伪随机函数；
- $\mathcal{H}$ 是伪随机置换；
- $\mathcal{G}$ 是伪随机生成器；
- A_t 是一个足够大的查询表，用于存储到此的陷门内容，通过此可以找到包含文档 ID 的信息。$SIZE(A_t)$ 表示查询表实际所使用的大小，即未填充时所使用的大小；
- A_s 是一个足够大的数组，用户存储文档 ID 的内容，用户检索到加密文档。$SIZE(A_s)$ 表示存储文档 ID 实际所使用的空间大小。

PSSSE 方案的各个算法具体实现如下：

1. $\{K\} \leftarrow SSKeyGen(1^k)$:
   该算法从集合 $\{0,1\}^k$ 中随机选取密钥 K_1, K_2, K_3，并且调用 $SKE.Gen$ 生成 $SK \leftarrow SKE.Gen(1^k)$，输出 $K = (K_1, K_2, K_3, SK)$。

2. $\{SI, KED\} \leftarrow SSEnc(D, K)$:
   该算法包括三个子算法 $(SSInitSets, SSDEnc, SSBuildSI)$。$SSInitSets$ 用户初始化操作，$SSDEnc$ 用于加密文档，$SSBuildSI$ 用于建立安全索引。它们按如下的顺序执行（SD 在前面被建立）：

   (a) $(SWS, SID) \leftarrow SSInitSets(D, SD)$.

   - $W(D) \leftarrow D$，遍历文档集 D，提取出所有不同的单词组成集合：$W(D) = \{w_1, ..., w_{|W(D)|}\}$。

- 对每一个单词 $w \in W(D)$，浏览文档 D 构建包含单词 w 的文档的 ID 集合：$SID_w = \{ID(D_i) \mid w \in D_i\})$，定义: $SID = \{SID_w \mid w \in W(D)\}$，并且确保每个集合的大小与最大集合大小相等。若小于最大集合则使用随机生成的唯一的字符串填充。
- 对每个单词 $w \in W(D)$，按照上述同义词集合构建过程构建文档的同义词集合。如何集合大小不相等，则通过填入随机的值使得每个集合的大小相等，生成新的集合 SWS。

(b) $(KED) \leftarrow SSDEnc(SK, D)$.
对每个文档 $D_i \in D$，加密生成 $ED_i \leftarrow SKE.Enc(SK, D_i)$。然后对每个 ED_i，设置：$KED_i =< ID(D_i), ED_i >$，并定义 $KED = \{KED_i | i \in [|D|]\}$。

(c) $(SI) \leftarrow SSBuildSI(K, SWS, SID)$.
该算法用于建立方案的核心检索结构 — 安全索引 SI，加速查找过程。详细实现细节，请参考算法 7。

最后，将安全索引 SI 和加密文档内容 KED 外包到远程服务器。

3. $\{T_w\} \leftarrow SSTrapdoor(w, K)$:
授权用户使用陷门生成函数输出单词 w 的陷门内容如下：$T_w = (\ F_{K_1}(w), G_{K_2}(w), P_{K_3}(w)\ )$，并将请求送至服务器。

4. $\{ESD_w\} \leftarrow SSSearch(T_w, SI, ED)$:
一旦服务器收到远程请求者的陷门，便开始查找安全索引和加密文档，并返回匹配的答案给请求者。具体的实现过程在算法10中描述。

5. $\{PSD_w\} \leftarrow SSDec(SK, ESD_w)$:
该算法仅涉及到简单的解密过程，最终得到用户需要的解密文档：PSD_w = $\{SKE.Dec_{SK}(ED_i) \mid ED_i \in ESD_w\}$。

4.2.2 算法描述

安全索引建立算法（SSBuildSI）：是一个确定性的算法，在数据被外包之前，由数据拥有者建立。当数据所有者想要使用外包服务时，首先他必须对外

包的数据建立索引，并将其存储到数组 A_s 和查询表 A_t 中，同时维持索引之间的联系以及文档 ID 与密文文档的联系。算法7实现如下：

Algorithm 7 $SSBuildSI$

Input:

K **:** 加密密钥

SD **:** 同义词字典

SWS **:** 同义词集合

SID **:** 文档 ID 集合

Steps:

1: get: $A_s \leftarrow SSBuildArray(K, SD, SWS, SID)$ (refer to 8).

2: get: $A_t \leftarrow SSBuildLookup(K, SD, SWS, SID)$ (refer to 9).

3: set: $SI = (A_t, A_s)$

4: **return** SI;

同义词查找算法（SSSearchESR）：是一个确定性的算法，在查找阶段用于对数据进行计算。当服务器收到用户提交的单词陷门时，在安全索引中进行查找符合条件的文档 ID，最后通过文档 ID，得到密文的文档并将结果返回给用户。详细的算法细节参见10。

4.3 安全性证明

定理 4.1 (Non-adaptive 安全的 $PSSSE$)**.** *如果函数 F、G 和 P 是伪随机函数，$\mathcal{H}$ 是伪随机置换，$\mathcal{G}$ 是伪随机生成器，并且 SKE 是 $PCPA$ 安全的，则方案 $PSSSE$ 是 non-adaptive 安全的。*

证明. 对所有多项式敌手 $\mathcal{A}$，总存在一个多项式的模拟器 $\mathcal{S}$，使得它的输出 $\mathbf{Sim}_{SSE,\mathcal{A},\mathcal{S}}(k)$ 与敌手输出 $\mathbf{Real}_{SSE,\mathcal{A}}(k)$ 在多项式时间内不可区分。给定 q 次查询历史 H，模拟器 $\mathcal{S}$ 输出元组 V* = $(SI^*, KED^*, T_w^*) = ((A_t^*, A_s^*), KED_1^*, ..., KED_n^*, T_{w_1}^*, ..., T_{w_q}^*)$ 如下：

1. 模拟 A_s^*.
   如果 q = 0，则对 $1 \leq i \leq |A_s|$，$\mathcal{S}$ 选择长度为 $|A_s[i]|$ 的唯一且随机的字符串，并存储到位置 $|A_s[i]|$。如果 $q \geq 1$，然后对 $1 \leq i \leq q$，$1 \leq j \leq |S_{w_i}|$

Algorithm 8 $SSBuildArray$

Steps:

1: **while** $w \in W(SD)$ **do**
2: obtain S_w of w from SWS
3: **if** $S_w = \varnothing$ or is visited **then**
4: continue;
5: **end if**
6: **while** $w_i \in S_w$ **do**
7: get SID_{w_i} from SID.
// **build a list** L_{w_i} **for** SID_{w_i} **that is stored in array** A_s
8: **while** $1 \leq j \leq |SID_{w_i}|$ **do**
9: let $ID(D_{i,j}$ be the j^{th} ID in SID_{w_i}
10: create a node:

$$A_s[N_{i,j}] = (< ID(D_{i,j})||addr_{ID}(N_{i,j+1}) > \oplus \mathcal{H}(P_{K_3}(w_i), r_i), r_i)$$

where $addr_{ID}(N_{i,j+1})$ that is randomly and uniquely generated is the address of $ID(D_{i,j+1})$
11: for the last node in L_{w_i}, set the address of next node is $NULL$:

$$A_s[N_{i,|SID_{w_i}|}] = (< ID(D_{i,j})||NULL) > \oplus \mathcal{H}(P_{K_3}(w_i), r_{|SID_{w_i}|}), r_{|SID_{w_i}|})$$

12: **end while**
13: fill the remaining $(|A_s| - \sum_{w \in W(D)}(|S_w|))$ elements to random strings of the same length.
14: **end while**
15: **end while**
16: **return** A_s;

和 $1 \leq k \leq |SID_{w_{i,j}}|$ （$w_{i,j}$ 表示 S_{w_i} 中的第 j 个同义词），每个结点的值如下：

$$A_s[addr^*_{ID}(i,j,k)] = (< \delta(w^*_{i,j,k}), addr^*_{ID}(i,j,k+1) >$$

Algorithm 9 $SSBuildLookup$

Steps:

1: **while** $w \in W(SD)$ **do**

2: obtain S_w of w from SWS

3: **if** $S_w = \varnothing$ or is visited **then**

4: continue;

5: **end if**

// build a circled list for S_w **in** A_t

6: **while** $w_i \in S_w$ **do**

7: for w_i, create a node:

$$A_t[F_{K_1}(w_i)] =< addr_{ID}(N_{i,1}), addr(w_{i+1}), G_{K_2}(w_{i+1}) > \oplus \mathcal{G}(G_{K_2}(w_i))$$

where $addr_{ID}(N_{i,1})$ is the first address of the SID_{w_i} of keyword w_i in A_s, and $addr(w_{i+1})$ is the next synonym address of keyword w_i.

8: for the last one in S_w, set:

$$A_t[F_{K_1}(w_{|S_w|})] =< addr_{ID}(N_{|S_w|,1}), addr(w_1), G_{K_2}(w_1)) > \oplus \mathcal{G}(G_{K_2}(w_{|S_w|})).$$

9: At last, each synonym set S_w forms a circled list.

10: **end while**

11: **end while**

12: The remaining entries of A_t are filled in the random strings.

13: **return** A_t;

$$\oplus \mathcal{H}(\gamma^*(i,j,k), r_i^*), r_i^*),$$

$\delta(w^*_{i,j,k})$ 表示第 i 个单词 w_i 的第 j 个同义词所对应的第 k 个文档的 ID，$addr^*_{ID}(i,j,k)$ 表示第 i 个单词 w_i 的第 j 个同义词的文档 ID 集中的第 k 个文档的 ID 的地址，并且 $addr^*_{ID}(i,j,k)$，$\gamma^*(i,j,k)$，$\delta(w^*_{i,j,k})$ 和 r_i^* 是随机的字符串，其长度分别为 $|addr_{ID}|$，$|P_{K_3}(w)|$，$|ID(D_{i,j,k})|$ 和 $|r_i|$。

在 A_s^* 中，对于其它的结点，按 SSBuildSI 方案插入 NULL 字符串。

2. 模拟 A_t^*.

Algorithm 10 SSSearchESR

Input:

T_w **:** 单词 w 的查找陷门

SI **:** 安全索引

KED **:** 加密的文档信息

Steps:

1: init: $ESD_w = \varnothing$.

2: parse T_w as (T_1, T_2, T_3), and let $M = A_t[T_1]$.

3: compute and set $A_M = M \oplus \mathcal{G}(T_2)$, and parse $A_M = (M_1, M_2, T_2)$.

4: get all document IDs by parsing T_3 and M_1, we set: $ID(M) = \{ID_i \mid i^{th}$ ID in parsing $L_M\}$

5: set $ESD_w = ESD_w \bigcup \{KED[ID_i] \mid ID_i \in ID(M)\}$.

6: set $M = A_t[M_2]$.

7: Loop in steps [3-6] until finishing the traverse

8: **return** ESD_w;

如果 q = 0，则对于 $1 \le i \le |A_t|$，$\mathcal{S}$ 对每个结点存入长为 $|A_t[i]|$ 的随机字符串。如果 $q \ge 1$，然后对 $1 \le i \le q$ 和 $1 \le j \le |S_{w_i}|$，$\mathcal{S}$ 随机生成长度分别为 $|F_{K_1}(w)|$、$|G_{K_2}(w)|$ 和 $|G_{K_2}(w)|$ 的字符串 $\alpha^*_{i,j}$、$\beta_{i,j}$ 和 $\beta_{i,j+1}$，每个结点的内容如下:

$$A^*_t[\alpha^*_{i,j}] =< addr^*_{ID}(N_{i,j,1}), addr^*(w_{i,j+1}), \beta^*_{i,j+1} > \oplus \mathcal{G}(\beta^*_{i,j}),$$

但是最后一个节点内容为:

$$A^*_t[\alpha^*_{i,|S_{w_i}|}] =< addr^*_{ID}(N_{i,|S_{w_i}|,1}), addr^*(w_{i,1}), \beta^*_{i,1} > \oplus \mathcal{G}(\beta^*_{i,|S_{w_i}|}),$$

$addr^*_{ID}(N_{i,j,1})$ 代表 $SID_{w_{i,j}}$ 的第一个节点的地址，$addr^*(w_{i,j}$ 代表第 i 个单词 w_i 的第 j 个同义词的地址。最终，单词 w_i 的同义词集合 S_{w_i} 形成一个循环链表。

对于 A^*_t 中其它的空结点，存放随机的无意义的字符串。

3. 模拟 $T^*_{w_i}$.

对单词 w_i，令 $T^*_{w_i} = (\alpha^*_i, \beta^*_i, \gamma^*_i)$。

4. 模拟 KED_i^*.
   如果 q = 0，则 $\mathcal{S}$ 随机选择 $|KED_i|$-bit 的字符串作为 KED_i^*。若 $q \geq 1$，对 $1 \leq i \leq q$，$1 \leq j \leq |S(w_i)|$ 和 $1 \leq k \leq |SID_{w_{i,j}}|$，生成键值对 $< \delta(w_{i,j,k}^*), ED_i^* >$，$\delta(w_{i,j,k}^*)$ — 表示文档 ID，与 A_s^* 中对应的文档 ID 相同, ED_i^* 是长度为 $|ED_i|$ 的随机字符串，剩余的文档用随机的值填充。

基于上述构造的结构，用陷门 $T_{w_i}^*$ 对安全索引 SI^* 进行查找，服务器将返回对应的结果。

假设敌手 $\mathbf{Real}_{SSE,\mathcal{A}}(k)$ 和模拟者 $\mathbf{Sim}_{SSE,\mathcal{A},\mathcal{S}}(k)$ 的输出分别为 V 和 V*。我们只需要证明，在得到敌手额外知识 $st_{\mathcal{A}}$ 的前提下，区分器 $\mathcal{D}$ 仍不能在多项式时间内区分 V 和 V*，则证明该方案具有 non-adaptive 安全。

1. (区分 A_s 和 A_s^*.)
   A_s 由 $SIZE(A_s)$ 个文档 ID 密文和 $|A_s| - SIZE(A_s)$ 个随机字符串组成。如果 q = 0，A_s^* 所有元素是随机字符串，若 $q \geq 1$，在 A_s^* 中，$q * |SID_w|$ 个结点内容为：$(< \delta(w_{i,j,k}^*), addr_{ID}^*(i,j,k+1) > \oplus \mathcal{H}(\gamma^*(i,j,k), r_i), r_i)$，其它为随机字符串。在这两种情况下，所有函数都是不可忽略的函数，并且 $st_{\mathcal{A}}$ 不包含密钥 K_3 的信息，则 A_s^* 中每个元素与 A_s 中对应元素不可区分。

2. (区分 A_t 和 A_t^*.)
   A_t 由 $SIZE(A_s)$ 个 $< addr_{ID}(SID_{w_i}[1]), addr(w_{i+1}), G_{K_2}(w_{i+1}) > \oplus \mathcal{G}(G_{K_2}(w_i))$ 值和 $|A_s| - SIZE(A_s)$ 个随机字符串组成。如果 q = 0，A_t^* 由随机值组成。若 $q \geq 1$，然后对 $1 \leq i \leq q$，A_t^* 由 $q * |S_{w_i}|$ 个使用异或操作生成的随机串组成，剩余结点为随机值。由于每个结点都是使用不可忽略函数，并且 $st_{\mathcal{A}}$ 没有包含密钥 (K_1, K_2)，因而，A_t^* 与 A_t 中每个对应元素在多项式时间内不可区分。

3. (区分 T_{w_i} 和 $T_{w_i}^*$.)
   T_{w_i} 和 $T_{w_i}^*$ 都由 PRF F, G 和 PRP P 组成。由于所有函数都是多项式不可区分，且 $st_{\mathcal{A}}$ 没有泄漏密钥 (K_1, K_2, K_3) 的信息，故陷门 t_i 与 t_i^* 不能被多项式算法区分。

4. (区分 KED_i 和 KED_i^*.)
   KED_i 有文档 ID 和经有 SKE.Enc 的密文 ED_i 组成。而 KED_i^* 由随机字串 $\delta(w_{i,j,k}^*)$ 和随机消息 ED_i^* 组成。因 SKE 是 PCPA 安全的和伪随机函数不可区分行，并且密钥 SK 不被 $st_{\mathcal{A}}$ 所泄漏，故 KED_i 和 KED_i^* 不能被区分器 $\mathcal{D}$ 所区分.

因此, 该方案是 non-adaptive 安全的。 □

4.4 性能分析

至此，我们已完成了对同义词可搜索加密方案的构建。首先，它实现了构建的最初目标 —支持同义词搜索；紧接着通过严格的安全分析，证明我们方案达到所要求的安全模型。但是，我们尚未对我们方案进行详细的性能分析。基于对称可搜索加密环境下已实现方案的各个详细算法实现细节，下面我们从三个方面 — 存储开销、计算开销和通讯开销分别对该方案的性能进行分析。

4.4.1 存储开销

在明文可搜索方案中，为了支持同义词搜索，搜索引擎不得不额外承担计算并存储单词的同义词集合，这样显然在原有的可搜索环境下大大地增加了一定的存储开销。为此，根据明文同义词搜索环境的特征，我们总结：为增加同义词搜索功能，不得不付出额外的存储代价（在客户或者服务端)。

在我们的方案中，由于我们要增加同义词功能并且要保证数据的隐私，我们清楚地了解到支付额外的存储开销是使我们方案支持同义词搜索的必要条件，这也是我们方案所采用的策略。我们的方案主要由（$SSKeyGen, SSEnc, SSTrapdoor, S$ 五部分组成。在 $SSKeyGen$、$SSTrapdoor$、$SSearch$ 和 $SSDec$ 这些模块中，仅仅用于辅助作用（与方案的存储开销没有任何关联)；$SSKeyGen$—仅仅用于生成密钥；$SSSearch$— 用于搜索并返回结果集；而 $SSDec$ 则仅用于加密返回的结果集。因而，在我们的方案中，仅仅 $SSDec$ 与系统的存储性能息息相关。在 $SSDec$ 模块中，我们包括了三个算法（$SSInitSets, SSDEnc, SSBuildSI$)，在算法 $SSDEnc$ 中，我们仅仅加密了明文文档，与前人的可搜索加密方案一致，下面我们详细地分析算法 $SSInitSets$ 和 $SSBuildSI$。

在 $SSInitSets$ 中，我们需要对每个单词建立同义字集合（即对每个单词 w，我们需要构造它们的同义字集 $S_w = \{w_i | SF(w_i, w) = 1\}$），我们知道在原有的可搜索加密方案中，并不需要建立同义字集合，因而我们方案与基本方案相比，同义字集合的增加是必不可少的。对于每个单词 w，我们知道产生同义字集合的大小为 $|S_w|$, 所有单词所构成的同义字总大小为 $SO(SSSE) = \sum_{i=1}^{|W(D)|}(|S_{w_i}|)$ （其中，$|W(D)|$—指文档中所有不同单词集合的大小)。我们清楚地知道同义字的总大小随着文档的动态增加而增大，并且在不同领域或环境下，我们为了增进方案的可用性和可扩展性，我们所定义的同义词集合大小还会有所增加。这仅仅是没有考虑服务器可能会暗地里偷偷的查看我们的信息 —这样服务器在查询过程中，根据长度来判断同义词的语义而导致信息泄漏。为了保证服务器不能通过同义词集合的长度来推断我们的信息，我们必须对每个同义词集合插入一些冗余信息以保证所有同义词集合的大小相同，因为我们得到同义词集合的新的大小为 $MAXSO(SSSE) = max(|SWS|) * |W(D)|)$，冗余信息的存储大小为 $MAXSO(SSSE) - SO(SSSE)$. 另外，在算法 SSBuildSI 过程中，为了增加同义词搜索功能，我们在查询表 A_t 的每个节点中增加了信息: $addr(w_{i+1}), G_{K_2}(w_{i+1}$，而这些信息与在 $SSInitSets$ 过程中生成的同义词集合大小相关，因此，我们方案增加的存储开销仅与同义词集合的大小相关。从上面分析中，简单看似我们在索引数组中每个条目的开销是原有方案的三倍并且要增加每个单词的同义词集合的大小，但是对于大量文档的存储语境来说可以忽略不计，下面我们用一个实例来进行推断。

在这个例子中，我们假设用户存储文档的数目为 1000，每个文档的大小为 500KB。在这些文档中，假设不同的单词的数目是 1000，而对于每个单词 w，他们的同义词集合大小为 10，并且在存储同义词索引信息时，对于每个单词计算他们的哈希值的大小为 128bits(16bytes)，而存储 $addr(w_{i+1}), G_{K_2}(w_{i+1})$ 占用的存储开销为 32bytes。因此，我们的总的存储信息大约为大小大约为 6MB，而在我们不支持模糊搜索方案中大小仅仅是 2MB，文档的信息大小为 500MB。在不考虑文档大小和文档 ID 存储的情况下，因为我们在方案中增加的存储开销是很大的（大概是原有方案的三倍)。但是当我们将放入整个方案中时，模糊单词集合的大小显得忽略不计。因此从存储开销着手，我们方案与原有方案—不支持模糊搜索的方案性能相当。

综上，我们的方案与不支持同义词搜索的方案在存储性能上基本持平，仅

仅在索引数组的构建存储上有所增加，大概为原有的 3 倍以上，而整个方案在文档不太小的情况下，基本与基本可搜索方面为 1:1；显然，这样的存储性能增加，对于增加同义词搜索功能的方案来说，我们是可以接收并得到应用的。

4.4.2 计算开销

同义词可搜搜加密方案的计算过程贯穿于整个方案中，而 $SSKeyGen$、$SSDEnc$ 和 $SSDec$ 都是基本的对称可搜索加密算法，在此我们不作讨论，我们主要分析方案中 $SSInitSets$、$SSBuildSI$、$SSSearchESR$ 的性能计算开销。

- **SSInitSets 过程.** 在 $(SWS, SID) \leftarrow SSInitSets(D, SD)$ 过程中，我们逐个遍历文档，并且对每个单词 w 维护一个文档 ID 集合 SID_w，在这个集合中每个元素是包含该单词的文档的 ID。与此同时，对每个单词 w，遍历同义词字典，找出所有相同含义的单词形成同义词集合 S_w。由算法描述可知，其计算时间开销为：$\sum(|D_i|) + W(D) * |SD|$ $(D_i \in D)$。

- **SSBuildSI 过程.** 客户端的计算开销主要集中在 $SSBuildSI$ 阶段，在这个阶段的输出结果为安全索引 $SI = (A_t, A_s)$，因而我们可以简单地分析为主要的计算时间花费在对这两个数组的填充过程中，大约为：$|A_t| + |A_s|$. 从算法7中，我们可以清楚地评估出，构建 A_s 所用的时间复杂度为 $|W(SWS)| * |S_w| * |SID_w|$，而构建数据结构 A_t 所用的时间复杂度为：$|W(SWS)| * |S_w|$.

- **SSSearchESR 过程.** 在构建好安全索引和数据被外包后，数据的交互在于 $SSSearchESR$ 部分，因而该算法性能指标对整个系统非常重要，并且是系统最常访问的过程。在算法10中，其计算时间主要集中在如何解析出待查单词的所有的同义词集并对其中每个单词查找所包含的文章，评估查找这部分的时间大小为：$|S_w| * |SID_w|$.

在我们的方案中，$SSInitSets$ 的计算开销主要花在建立同义词集合阶段，并且仅仅在初始时才计算一次，以后保持不变；$SSBuildSI$ 的计算开销主要是通过同义词集合建立安全索索引，也仅仅是一次性计算；而 $SSSearchESR$ 过程中，每次我们查询都需要查询，并且查询主要消耗在遍历索引数组和单词对

应文档 ID 的数组过程中。综上，我们方案的主要计算开销是 $SSSearchESR$ 过程，我们知道这个过程的计算开销为 $|S_w| * |SID_w|$ （$|SID_w|$ — 指单词 w 所在文档的数目），与原有的非同义词可搜索方案相比，仅仅增加了对同义词集合的访问，而每个单词同义词的数目也不大，并且在不同场景下，可一次的书目会更小，因而我们是完全可以接受的；除此之外，我们还在我们的补充方案中提出了基于 Trie 树结构的方案，这样将进一步减小我们的查找开销。

4.4.3 传输开销

在对称可搜索加密场景下，为了利用远程服务器强大的计算能力和廉价的服务，我们不得不在客户和服务器之间进行频繁的通讯。同理在我们的同义词可搜索加密中，方案的传输开销主要来源于与服务器之间的交互。下面我们从三个方面的方案的通信开销进行阐述 — 客户外包数据、授权用户提交的搜索陷门和服务器的返回结果。

1. **数据外包.**数据拥有者完成对待外包的数据进行安全操作后 — 主要是建立安全索引和加密文档数据过程。外包数据的通信开销主要发生在：数据拥有者将加密文档和安全索引提交给服务器的过程中。外包的数据主要包括加密文档和安全索引，我们知道要想外包数据，对文档的提交是必不可少的，因此这里我们主要分析外包的安全索引的大小。在 $SSBuildSI$ 过程中，安全索引包括两部分信息（索引查找表和存放文档 ID 的数组）；对于索引查询表 A_t，由存储开销部分的分析，我们知道它的大小为：$|W(D)| * |S_w|$ — 即所有不同单词的大小和单词同义词集最大长度的乘积。而对于文档 ID 链表所形成数组的大小主要为取决于每次单词被文档包含的次数，表示为：$|SID_{w_i}| * |W(D)|$ （$w_i \in W(D)$）— 即单词的数目与所有单词被文档包含最多次数的乘积。这些信息会因支持同义词搜索而比原有方案有所增加。

2. **搜索陷门提交.** 当授权用户对外包数据进行搜索时，他们将提交单词的陷门给服务器。在我们方案中，提交单词 w 的陷门信息是：$T_w = (F_{K_1}(w), G_{K_2}(w), P_{K_3}(w))$。我们显然知道搜索的陷门是固定的三个哈希值，在任何时候保持不变，因而这样的传输数据是合理并且高效的。与非同义词可搜索加密技术相比，我们的方案传输的信息会增加一些，但

是这对于一个功能增加的方案来说显得微不足道，我们认为这样的设计是非常合理并且有实际意义的。

3. **响应结果集返回.** 在算法 $SSSearchESR$ 中，服务器查找到匹配结果后，将其返回给请求的用户。在这个返回过程中，服务器响应的结果信息包括所有包含待查单词同义词的文档，其最大开销是与在存储时每个单词的最大存储开销一致，即 $|SID_w| * |S_w|$ $(w \in W(D))$。这个结果与简单的可搜索加密方案相比，显然会大得多，由于我们的方案是实现同义词查找的情形，因而其增加的开销也是必不可少的。

综上，我们方案中各个阶段的传输开销都是必要的，并且不存在冗余信息，是实现同义词搜索必不可少的。并且外包的数据通讯虽然较简单方案有所增加，但是这样的通讯是一次性的，仅在初始外包数据时才传输，并不会影响用户使用性能和增加网络的有用流量。而在陷门提交阶段，传送的仅仅是哈希信息值，对网络负荷毫无影响。而在响应结果集中，这样的开销增加是必须的，并且是有效的增加，返回的信息都是用户所需要的。

第五章 总结与展望

5.1 全文总结

本文首先总结和分析了云计算环境下的对称可搜索加密技术，包括对称可搜索加密中的各个子问题（精确搜索、模糊搜索、范围搜索、布尔搜索以及动态搜索)；然后详细地分析了对称可搜索加密环境下的模糊搜索加密技术的研究现状和主要的研究内容，以及对该方案中的不足进行了阐述。通过对对称搜索环境下已有方案的研究，我们总结出已有方案普遍存在的一个安全缺陷 — 搜索时信息泄漏过大的问题，并针对该问题提出了一个能降低在查找过程中信息泄漏的方案 —即抗信息泄漏的可搜索加密方案（基于史密斯正交化的原理)，该方案证明了在通过增加存储开销时减少信息的泄漏。除此之外，我们挖掘出该语境下一个新的复杂对称可搜索加密技术问题 — 同义词对称可搜索加密技术；在文中我们阐述了同义词搜索与模糊搜索的关联。随之通过对该问题的详细分析和设计，我们提出了一个确保低通讯开销、少信息泄漏兼高搜索性能的同义词搜索方案；在方案中，我们引入了同义词集合和同义词判断函数，并且详细地描述了方案的算法实现和证明了方案的安全性。此外，我们阐述我们方案通讯开销为 $O(1)$，查找开销为 $O(p)$ （p — 表示单词同义词集合的大小)。

综合上述问题，在文中，我们提出了两个解决方案 —抗信息泄漏的可搜索加密和同义词可搜索加密。在方案中，我们证明了它们的可行性和性能以及安全性；并描述上述方案都能很好地应用于实际中已提出的方案之中，进一步扩展了已有方案的功能、安全性和可扩展性；但是，我们的方案中都对服务器端的存储开销有所增加。

5.2 未来展望

在大数据时代的背景下，云计算技术正以着飞速的速度发展和应用。但是安全的云计算可搜索加密方案仍处于理论的阶段，虽然有些安全的云计算系统已被开发，但是不足以面对现实的需求，主要由于这些系统难以在可应用和强安全性之间达到平衡 — 可用性好不够安全，而足够安全则功能太过于单一而

不被使用，并且这些系统功能也非常单一。为了开发出兼容诸多优势的方案，我们还有很大一段距离要走 — 主要是还有很多难题需要被提出和进一步研究。从本文的研究方向上来看，主要可以从如下几个方向进行进一步研究：

1. 在已实现的同义词可搜索加密方案中，并不能实现同义词集合随着上下文环境的变化而变化 — 即同一份文档集在不同环境下有不同的同义词集，或在不同时间段同义词随着环境来进行动态改变。最终，希望在不久，能构建一个同义词动态变化的同义词可搜索加密方案。

2. 通过我们的方案的详细描述，我们了解了同义词和模糊搜索之间的紧密联系。接下来我们希望将同义词搜索和模糊搜索结合起来，实现一个确保安全的相似可搜索加密方案 — 同时兼容模糊和同义词搜索功能。

3. 由于现有的模糊搜索方案都是基于通配符的解决方案，这些方案有一个明显的缺陷 — 高存储开销。接下来，我们希望能提出一个低存储的模糊搜索方案。

4. 我们希望将我们的方案扩展到动态的场景下 — 同时支持文档的动态的修改和同义词搜索，使方案更具可实现性和灵活性。

参考文献

[1] Walker E. Benchmarking Amazon EC2 for high-performance scientific computing[J]. Usenix Login, 2008, 33(5):18–23.

[2] Bogatin D. Google CEO's new paradigm:'cloud computing and advertising go hand-in-hand'[J]. ZDNet.[Online]. Available: http://blogs. zdnet. com/micro-markets, 2006.

[3] Mell P, Grance T. The NIST definition of cloud computing[J]. National Institute of Standards and Technology, 2009, 53(6):50.

[4] Fox A, Griffith R, Katz R. Above the clouds: A Berkeley view of cloud computing[J]. Dept. Electrical Eng. and Comput. Sciences, 2009, 28:13.

[5] Chiueh S N T c, Brook S. A survey on virtualization technologies[J]. RPE Report, 2005:1–42.

[6] Ristenpart T, Tromer E, Shacham H, et al. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds[J]. 2009:199–212.

[7] Biham E, Shamir A. Differential cryptanalysis of the data encryption standard[M], Vol. 28.[S.l.]: [s.n.] , 1993.

[8] Grefenstette G. Explorations in automatic thesaurus discovery[M].[S.l.]: [s.n.] , 1994.

[9] Song D X, Wagner D, Perrig A. Practical techniques for searches on encrypted data[J]. 2000:44–55.

[10] Goh E J, et al. Secure Indexes.[J]. IACR Cryptology ePrint Archive, 2003, 2003:216.

[11] Chang Y C, Mitzenmacher M. Privacy preserving keyword searches on remote encrypted data[J]. Applied Cryptography and Network Security, 2005:442–455.

[12] Curtmola R, Garay J, Kamara. Searchable symmetric encryption: improved definitions and efficient constructions[J]. 2006:79–88.

[13] Van Liesdonk P, Sedghi S, Doumen J, et al. Computationally efficient searchable symmetric encryption[J]. 2010:87–100.

[14] Chai Q, Gong G. Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers[J]. Communications ICC, 2012 IEEE International Conference, 2012:917–922.

[15] Kurosawa K, Ohtaki Y. UC-secure searchable symmetric encryption[J]. Financial Cryptography and Data Security, 2012:285–298.

[16] Chase M, Kamara S. Structured encryption and controlled disclosure[J]. Advances in Cryptology-ASIACRYPT 2010, 2010:577–594.

[17] Boneh D, Franklin M. Identity-based encryption from the Weil pairing[J]. SIAM Journal on Computing, 2003, 32(3):586–615.

[18] Li J, Wang Q, Wang C, et al. Fuzzy keyword search over encrypted data in cloud computing[J]. INFOCOM, 2010 Proceedings IEEE, 2010:1–5.

[19] Wang C, Ren K, Yu S, et al. Achieving usable and privacy-assured similarity search over outsourced cloud data[J]. INFOCOM, 2012 Proceedings IEEE, 2012:451–459.

[20] Kuzu M, Islam M S, Kantarcioglu M. Efficient similarity search over encrypted data[J]. 2012:1156–1167.

[21] Indyk P, Motwani R. Approximate nearest neighbors: towards removing the curse of dimensionality[J]. 1998:604–613.

[22] Chuah M, Hu W. Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data[J]. 2011:273–281.

[23] Islam M S, Kuzu M, Kantarcioglu M. Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation[J]. NDSS, 2012:31–45.

[24] Liu C, Zhu L, Wang M, et al. Search pattern leakage in searchable encryption: Attacks and new construction[J]. Information Sciences, 2014, 265:176–188.

[25] Bellare M, Canetti R, Krawczyk H. Pseudorandom functions revisited: The cascade construction and its concrete security[J]. 1996:514–523.

[26] Håstad J, Impagliazzo R, Levin L A, et al. A pseudorandom generator from any one-way function[J]. SIAM Journal on Computing, 1999, 28(4):1364–1396.

[27] Gremillion L L. Designing a Bloom filter for differential file access[J]. Communications of the ACM, 1982, 25(9):600–604.

[28] Jin C, Gu D, Tang Y, et al. Reducing extra storage in searchable symmetric encryption scheme[J]. 2012:255–262.

[29] Ji S, Li G, Li C, et al. Efficient interactive fuzzy keyword search[J]. Proceedings of the 18th international conference on World wide web, 2009:371–380.

[30] Li C, Lu J, Lu Y. Efficient merging and filtering algorithms for approximate string searches[J]. Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on, 2008:257–266.

[31] Behm A, Ji S, Li C, et al. Space-constrained gram-based indexing for efficient approximate string search[J]. Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on, 2009:604–615.

[32] Ristad E S, Yianilos P N. Learning string-edit distance[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1998, 20(5):522–532.

[33] Zhang Z, Hadjieleftheriou M, Ooi B C, et al. Bed-tree: an all-purpose index structure for string similarity search based on edit distance[J]. Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, 2010:915–926.

[34] Balamuralikrishna T, Anuradha C, Raghavendrasai N. Fuzzy Keyword Search over Encrypted Data in Cloud Computing[J]. Asian Journal Of Computer Science and Information Technology, 2013, 1(3).

[35] Boldyreva A, Chenette N. Efficient Fuzzy Search on Encrypted Data.[J]. IACR Cryptology ePrint Archive, 2014, 2014:235.

[36] Kamara S, Papamanthou C, Roeder T. Dynamic searchable symmetric encryption[J]. 2012:965–976.

[37] Kamara S, Papamanthou C. Parallel and dynamic searchable symmetric encryption[J]. 2013:258–274.

[38] Stefanov E, Papamanthou C, Shi E. Practical Dynamic Searchable Encryption with Small Leakage.[J]. IACR Cryptology ePrint Archive, 2013, 2013:832.

[39] Swaminathan A, Mao Y, Su G M e. Confidentiality-preserving rank-ordered search[J]. Proceedings of the 2007 ACM workshop on Storage security and survivability, 2007:7–12.

[40] Wang C, Cao N, Li J, et al. Secure ranked keyword search over encrypted cloud data[J]. 2010:253–262.

[41] Wang C, Cao N, Ren K, et al. Enabling secure and efficient ranked keyword search over outsourced cloud data[J]. Parallel and Distributed Systems, IEEE Transactions, 2012, 23(8):1467–1479.

[42] Boldyreva A, Chenette N, O'Neill A. Order-preserving encryption revisited: Improved security analysis and alternative solutions[J]. 2011:578–595.

[43] Cao N, Wang C, Li M, et al. Privacy-preserving multi-keyword ranked search over encrypted cloud data[J]. Parallel and Distributed Systems, IEEE Transactions, 2014, 25(1):222–233.

[44] Witten I H, Moffat A, Bell T C. Managing gigabytes: compressing and indexing documents and images[M].[S.l.]: [s.n.] , 1999.

[45] Xu Z, Kang W, Li R, et al. Efficient Multi-Keyword Ranked Query on Encrypted Data in the Cloud.[J]. 2012:244–251.

[46] Daniel J W, Gragg W B, Kaufman L, et al. Reorthogonalization and stable algorithms for updating the Gram-Schmidt factorization[J]. Mathematics of Computation, 1976, 30(136):772–795.

[47] Schwarzenberger R L. Vector bundles on the projective plane[J]. Proceedings of the London Mathematical Society, 1961, 3(1):623–640.

攻读学位期间发表的学术论文目录

[1] Zuopeng Liu, Haining Lu, Ning Ding. A Practical Synonym Symmetric Searchable Encryption. Computer Science and Communication Engineering, 2015.