Computer Science and Application 计算机科学与应用, 2016, 6(12), 778-785
Published Online December 2016 in Hans. http://www.hanspub.org/journal/csa
http://dx.doi.org/10.12677/csa.2016.612093

Hans汉斯

Searchable Symmetric Encryption System Based on Graphic Database

Chen Tian, Lei Fan

School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai
Email: s.holmestian@hotmail.com, fanlei@sjtu.edu.cn

Received: Dec. 1st, 2016; accepted: Dec. 14th, 2016; published: Dec. 22nd, 2016

Abstract

Searchable Symmetric Encryption constructions offered the ability to search keywords over encrypted files. But the index grows bigger with the increasing of files, and is difficult to be stored in memory. Based on the keyword Red-Black tree, a new construction is designed and by using graphic database can support persistence and other features. Our construction has new building and searching method, and by using cache and slow-update, handles the updating efficiency. Through the performance test experiment, compared with the traditional memory index to verify the effectiveness and usability of the system design, the time of the initial building index increased by 40%, but the search and update efficiency can be maintained approximately with memory index.

Keywords

Searchable Symmetric Encryption, Graphic Database, Data Security, Cryptography, Cloud Storage

基于图数据库的可搜索加密

田　琛，范　磊

上海交通大学电子信息与电气工程学院，上海
Email: s.holmestian@hotmail.com, fanlei@sjtu.edu.cn

收稿日期：2016年12月1日；录用日期：2016年12月14日；发布日期：2016年12月22日

摘　要

可搜索加密技术提供了对加密文件的关键词搜索能力，但是无法有效对云环境中海量文件形成的巨大索

文章引用: 田琛, 范磊. 基于图数据库的可搜索加密[J]. 计算机科学与应用, 2016, 6(12): 778-785.
http://dx.doi.org/10.12677/csa.2016.612093

引进行操作。在树形索引的基础上，提出基于图形数据库构建可持久化索引的方法。设计了新型索引的构建和检索算法，并实现了索引的并行查找优化检索效率，通过使用内存缓存和慢更新的方法应对索引更新慢的问题。通过性能测试实验，对比传统内存索引，验证系统设计的有效性和可用性，索引初次构建时间增加40%，但是搜索和更新效率可以维持与内存索引近似。

关键词

可搜索加密，图数据库，数据安全，密码学，云存储

1. 引言

近年来，云计算得到了迅猛的发展，提出了“软件即服务”、“平台即服务”、“服务即服务”和“应用即服务”等众多的解决方案[1]。基于云的各种存储产品也走进了大众的生活，例如 DropBox、百度云盘等。但是由于黑客入侵等原因，发生了多起云安全事故，例如 2011 年黑客入侵 Sony 公司造成数亿用户个人信息泄露，同年 Google 公司旗下产品 Gmail 用户信息也遭到恶意泄露[2]。

常用的隐私保护方案是，在数据文件离开用户控制前将其加密，并将密文存储在云端。这种解决方案可以实现端到端的安全，但是却不实用。因为数据被加密后，云端无法对它进行任何的有效计算，这不适用高吞吐和高容量的云存储场景。

为了同时实现数据加密和关键词搜索，可以采用完全同态加密[3]或者不经意随机存取(ORAM) [4]或者可搜索加密(SE)。前两者理论上拥有更多的优点，如更具有通用性和更强的安全性，但是由于计算复杂度高、系统设计难度大，距离实践尚远。可搜索加密实现了安全性和效率、可用性之间的平衡，更具有实践意义，得到了广泛的研究。

可搜索加密(SE)按加密方式可分为基于对称密钥的可搜索加密(SSE)和非对称密钥的可搜索加密(PSE)。目前主要有两种实现可搜索加密的方式，一种由 Goh [5]提出并在文献[6]中使用，为每一个文档构造加密的数据结构，这种结构支持对关键词的检查能力，时间复杂度为 $O(n)$，其中 n 为文档的个数；第二种由 Curtmola [7]提出，为整个文档集合构建倒排索引，通过搜索关键词找到对应的文档位置，这种方式的时间复杂度是 $O(k)$其中 k 为关键词的个数。由于第二种方案是时间复杂度上的最优解，所以有许多构建算法采用[8] [9] [10] [11]。

倒排索引的构建方式存在以下几个问题，索引的构建是静态的，无法应对动态的扩张；索引的搜索过程是串行化的，因为索引结构的唯一性导致无法很好的实现并行搜索；索引主要是构建在内存中，无法应对大量文件的索引同时不具备容灾性。Kemara 在文献[9]中首次提出了动态索引的概念和构造方法，在文献[12]中通过使用关键词红黑树(KRB)索引的构造方法，较好的实现了索引的动态性和可并发性。

原有的可搜索加密方案无法进行分布式拓展，无法应对高并发等实际生产场景。通过改进基于关键词红黑树的可搜索加密方案，将其应用在图数据库上，获得更好的并发搜索特性和分布式部署能力。最后通过微基准测试，比较原方案和图数据库方案的各项性能，证明图数据库方案的可行性。

2. 相关工作

2.1. 动态可搜索加密模型

在构建可搜索加密模型中，使用了一些基础的密码学原语。其中私钥加密模型是一个多项式时间算法的三元组，SKE = (Gen, Enc, Dec)，Gen 是一个概率算法，输入安全参数 k，输出密钥 K；Enc 是一个概率算法，输入密钥 K 和消息 m，输出密文 c；Dec 是一个确定性算法，输入密钥 K 和密文 c，输出 m，

其中 K 是用来生成 c 的密钥。如果一个私钥加密模型，即使攻击者可以自由查询加密串，输出 c 仍然不泄漏任何明文的信息，则称这个模型是 CPA 安全的。本文仍使用到了其他密码学原语如，伪随机函数(PRF)、排列(PRP)，文献[13]给出了 PRF、PRP、CPA 的正式定义。

定义 1 (动态可搜索加密模型)动态可搜索加密模型是一个多项式时间算法的八元组：(Gen, Enc, SrchToken, Search, UpdHelper, UpdToken, Update, Dec)，其中：

$K \leftarrow \mathrm{Gen}\left(1^k\right)$：概率算法，输入安全参数 k，输出密钥 K。

$(\gamma, c) \leftarrow \mathrm{Enc}(K, \delta, f)$：概率算法，输入密钥 K，索引 δ，文件集 f，输出加密后的索引 γ 和密文集 c。

$\tau_s \leftarrow \mathrm{SrchToken}(K, w)$ 概率算法，输入是密钥 K 和一个关键词 w，输出搜索令牌 τ_s。

$i_w \leftarrow \mathrm{Search}(\gamma, c, \tau_s)$：确定性算法，输入加密后的索引 γ，密文集 c 和搜索令牌 τ_s，输出是一组标记 $i_w \subseteq i$。

$\inf o_{i,u} \leftarrow \mathrm{UpdHelper}(i, u, \gamma, c)$：确定性算法，其中 i 表示文件标示符，u 表示更新的类型{add, delete}，γ 表示加密后的索引，c 表示密文集。输出 $\inf o_{i,u}$ 包含了此次更新的信息。

$f \leftarrow \mathrm{Dec}(K, c)$：确定性算法，输入密钥 K 和密文 c，输出明文 f。

2.2. 关键词红黑树

关键词红黑树(Key Red Black Tree)是对红黑树结构的改进，类似倒排索引，可以支持对多关键词的查询。设有 KRB Tree δ，是由文件集合 $f = \{f_{i_1}, \cdots, f_{i_n}\}$ (其中 $\{i_1, \cdots, i_n\}$ 为文件的标识符)和全局单词集合 W 构建，这里假设某种自然语言的单词集是固定的、有限的。则 δ 的每个节点 node 存储文件标识符 i 和长度为#W 的 bit 列表 $\mathscr{L}$，其中 $\mathscr{L}$ 的第 m 个元素代表文件 f_i 是否含有关键词 $W[m]$，δ 的叶子节点 leave 存储指向对应文件的标识符。

2.3. 图数据库 Neo4J

随着数据间关系的复杂，人们认识到虽然关系型数据库已经成为事实上的数据存储标准，但并不适用于解决面对数据复杂性日益剧增的所有问题。于是 NoSQL 数据存储技术应运而生，图数据库便是其中一种代表。在应用程序开发中，有时数据建模非常像一种图形状结构，如果仍然使用表状结构存储，那么在查询相关联项时必须使用多表联合，极大的影响效率。

Neo4J [14]基于图形遍历的局部性，实现了一个性能可预期的图遍历模型。例如社交网络关系，使用 Neo4J 来表达和建模会非常的自然，Neo4J 同时提供了丰富的 API 接口帮助遍历和搜索节点的关联项。所以，Neo4J 可以用来为大容量倒排索引建模和实现持久化。

3. 基于图数据库的 SSE 构造

目前提出的主要可搜索加密方法中，其倒排索引都是构造在内存中，对应的实现与真正可用之间还有较大差距。内存索引的搜索速度快，但是容错性差，也无法适应大容量多用户索引场景，同时无法分布式部署和计算。针对内存索引的这些缺点，提出基于图数据库的索引构建和搜索方法，实现了并行搜索和分布式存储。

本构造是一个动态可搜索加密模型，借鉴关键词红黑树的数据结构，在图数据库 Neo4J 上的一个完整实现。算法的设计包括三个部分：索引的构建、关键词搜索和索引的动态更新。

3.1. 索引构建

倒排索引是实现可搜索加密的核心数据结构，也是可搜索加密模型的基础。根据第 2 节中动态可搜索加密模型的定义，详细描述索引的构建过程。在本方案中，假设全局不同关键词个数是固定不变的，

设为 m。

首先构建关键词红黑树 δ，设有文件集合 $f=\left(f_{i_1},\cdots,f_{i_n}\right)$ 其中 $i1$ 到 in 表示文件的 id 号，W 表示文件中的关键词向量。根据文件的 id，构建一个红黑树，在树的每一个非叶子节点 u 上存储一个 bit 向量 data_u，其中若 f_u 含有关键词 $W[i]$则 $\text{data}_u[i]=1$。树的叶子节点储存指向文件的指针。在树结构形成后，从叶子节点到根节点递归的对 data 向量做与运算。算法的伪代码如下表所示：

Algorithm 1. Build the key red-black tree
算法 1. 构建关键词红黑树

```
Input: D, the list of document
Initialization: δ ← initialIndex (D[0])
1     δ.root ← newNode(D[0])
2δ.size ← 1
3     for each d_i in D:
4   addNode(δ, d_i)
5     endfor
AddNode: δ' ← addNode(δ, d), d means a document
1Node t ← δ.root
2Node parent
3do:
4parent ← t
5 if d.getId < t.id:
6t ← t.right
7          else:
8                t ← t.left
9while t is not null and is not leaf
10Node e ← new Node(d, parent)
11if d.getId < t.id:
12e.left ← parent.left
13          parent.left ← e
14else:
15          e.left ← parent.right
16          parent.right ← e
17     e.right ← new LeafNode(d, e)
18     fixAfterInsertion(e)
19     updateAfterFix(δ.root)
20     δ.size ++
```

完成关键词红黑树 δ 的构建后，需要对 δ 和文件集合加密保护。主要的算法步骤为：

1) $K\leftarrow \text{Gen}\left(1^k\right)$：产生三个 k-bit 的随机字符串 $K1$，$K2$，$K3$。

2) 对每一个关键词产生一个密钥：$SK_i=\mathcal{R}.\text{Gen}\left(1^k;G_{K_2}\left(W_i\right)\right)$ 其中，$\mathcal{R}$ 为第 2 节中提到的一种私钥加密模型，$G_{K_2}\left(W_i\right)$ 为一个伪随机函数，随机数种子为 $K2$。

3) 对文件集合中的每一个文件做加密处理：$c_{i_j}\leftarrow \varepsilon.\text{Enc}\left(K_3,f_{i_j}\right)$ 其中 ε 为另一种私钥加密模型。

4) 对关键词红黑树做加密处理：(a)在每个节点 v 中存入两个 HashTable，λ_{0v}，λ_{1v} (b)对节点 v 中的关键词向量 data_v 做加密处理 $\lambda_{bv}\left[P_{K1}\left(W_i\right)\right]\leftarrow\mathcal{R}.\text{Enc}\left(SK_i,\text{data}_v[i]\right)$ 其中 $b=H\left(P_{K1}\left(W_i\right),id\left(v\right)\right)$ 是一个根据关键词和节点号计算出的随机 bit，或为 0 或为 1，起到混乱作用。(c)对两个 HashTable 中的剩余位置填充随机 bit。

通过以上操作，得到加密后的倒排索引和加密后的密文集合，发送给服务器端，服务器端将索引转译为图数据库的节点和关系集合，实现分布式存储索引的目的。可以证明，以上的加密操作可以抵抗搜索路径攻击和访问路径攻击。

3.2. 关键词搜索

客户端需要生成搜索令牌发送给服务器端，服务器端根据下面的算法进行搜索。由于图数据库的分布式部署和内存缓存，所以对于缓存命中的搜索，其速度接近内存索引的查询速度。

算法 2 (令牌产生) $\tau_s \leftarrow \mathrm{SrchToken}(K, w_i)$

调用 $\mathcal{R}.\mathrm{Gen}\left(1^k; G_{K2}(w_i)\right)$ 产生密钥 SKi，然后产生搜索令牌 $\tau_s \boxminus \left(P_{K_1}(w_i), SK_i\right)$

算法 3 (搜索算法) $i_w \leftarrow \mathrm{Search}(\gamma, c, \tau_s)$

把 τ_s 作为一个二元组 (τ_1, τ_2)，调用 search(r)，其中 r 为索引的根节点。设 v 和 z 为节点 u 的左右儿子节点，则 search(u)可以按一下算法递归的进行：

1) 输出一个 bit $b = H\left(\tau_1, \mathrm{id}(u)\right)$ 然后计算 $a = \mathcal{R}.\mathrm{Dec}\left(\tau_2, \lambda_{bu}[\tau_1]\right)$；
2) 如果 $a = 0$，返回；
3) 如果 u 是叶子节点，返回 $c_w \boxminus c_w \cup c_u$；
4) 否则：search(v) and search(z)

由于客户端产生的搜索令牌中所有信息都是加密后的，所以服务器端是无法得知客户的搜索意图也无法得知搜索结果的明文。服务器端根据密文集合的 id 将对应密文上传给客户端，客户端使用储存的密钥解密得到明文文件。

3.3. 索引动态更新

和文件 f_i 有关的索引更新 u，其实就是一次插入操作和 / 或一次删除操作。为了计算更新的信息 $\mathrm{info}_{i,u}$，服务器需要更新索引的结构。但是由于树的特点，服务器并不需要更新整棵树，而只需要调整子树 T(u)。更新信息 $\mathrm{info}_{i,u}$ 含有这个子树，已经足够用于更新。

如果更新操作 u 是一次插入操作，服务器端会采用 lazy-load。服务器端会根据当前索引的热度来判断是否立刻执行插入操作，如果有线程在读写索引，则将子树暂存在根节点下。设一个低优先级的线程会缓慢的扫描每个索引并执行合并操作。

3.4. 系统设计

系统采用 *C/S* 架构，所有的涉密操作均在客户端完成，服务器端不需要任何的客户密钥，客户端和服务器端通过加密索引和令牌交流，系统的主要结构如图 1 所示。

客户端首先将需要索引的文件输入到物料处理器，该处理器使用了 Lucene 框架中的分词工具，可以屏蔽文件的类型从中提取全文关键词，并根据预设的词典，将关键词列表映射为 Bit 的数组。然后通过调用对称加密对文本进行加密处理并生成全局唯一 ID。物料处理器的输出为一个称作 Document 的数据结构列表。

索引构建器接受物料处理器的 Document 列表为输入，通过调用伪随机函数混淆关键词列表，并构建基于关键词红黑树的倒排索引，最后生成构建令牌发送到服务器端。服务器端接收到构建令牌，根据令牌的索引同步图数据库，生成用户和图节点。索引的更新过程与构建类似，但是会产生更新令牌，服务器收到更新令牌后会在图数据库中构建子索引，并依据算法实现懒合并。

客户端的 Searcher 根据用户指定的关键词生成搜索令牌，发送到服务器端。服务器端根据搜索令牌查找含有此关键词的 Document 列表，并将列表返回。

4. 性能分析

基于第三章的系统设计和算法描述，使用 java 语言开发实现了客户端，使用 spring 框架和 Neo4j 的

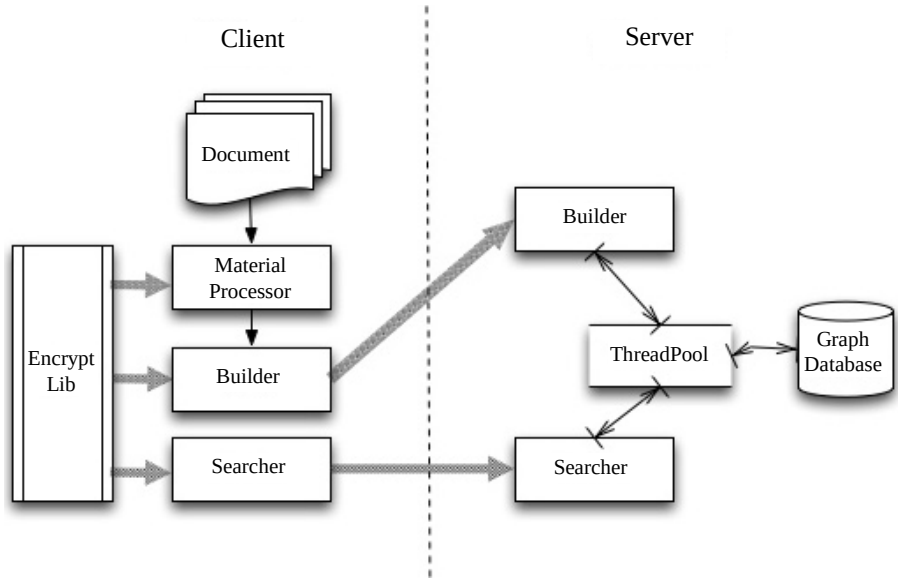

Figure 1. System construction design
图 1. 系统结构设计图

jar 包实现了服务器端，客户端和服务器端的通信使用 fastJson 进行对象的序列化和反序列化。其中，客户端运行在 OS X 10.11 Mac 机器上，内存 8 G，CPU 型号为 Intel Corei7，服务器端运行在三台内存 16 G，CPU 4 × 2.3 GHz 的组装机上。系统实现时基于一个假设：关键词列表是定长的。通过查找相关文献确定，英语中常用单词数在 1 万到 3 万之间，采用 10240 作为索引中的关键词列表长度。

通过使用树状结构和图形数据库的结合，可以实现并行搜索。假设现在搜索关键词 *Wi*，在节点 *v* 上若 datav[*i*] = 1，则启动两个线程搜索 *v* 的左子节点和右子节点，直到耗尽图数据库的可用资源。若图数据库提供的可并发线程为 *p* 则一次关键词搜索的时间复杂度为 $O\left(\frac{r}{p}\log n\right)$，其中 *r* 为含有关键词 *Wi* 的文章数。由于图数据库的分布式特性，所以他的可使用线程数远大于单机，带来了更好的搜索速度。

借鉴文献[9]中提到的 micro-benchmark 的测试方法，通过使用较小的文件集梯度测试来评估系统的性能。使用不带附件的 Email 文件作为测试文件库，将文件拆分为等单词数的小文件。通过对比在内存中和分布式数据库中每种操作的耗时，来分析引入持久化的代价。所有数据点都是重复 5 次取平均值得到。

由图 2 可以看出，在使用图数据库后，由于存在将内存中的树状结构向图数据库中存储的过程，对于新建索引引入了约一倍的时间消耗。但是，通常的场景是，用户在第一次创建索引的时候，并不会提供太多的文件，而是通过后续多次的添加操作，使索引膨胀。

图 3 对比了在插入一个文件的操作中，两种不同索引的表现。因为在图数据库索引中采用了 lazy-load 的模式，所以极大的加快了更新的效率，获得了更好的响应能力。实现了重任务的延后执行。

图 4 对比了不同索引在搜索高命中概率关键词时的表现，例如英文单词“the”几乎出现在每一个文件中，所以这近似体现了遍历索引的性能。由于图数据库的缓存机制，使得在多次查询中缓存命中概率较高，提升了多次查询的效率，所以在引入图数据库后对查询性能的影响是可控的。同时，由于采用树状结构，使得多线程查询非常容易实现，进一步的修正了对效率的影响。

由此可见，通过使用图数据库存储大容量索引会对构建和更新过程造成可控的影响，但是换来了更高的稳定性和搜索速度。

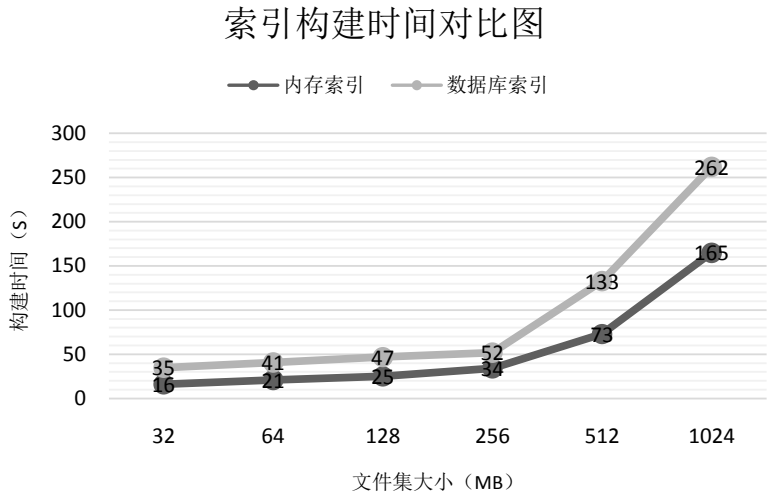

Figure 2. Time comparison of old index and new index
图 2. 不同索引构建时间对比图

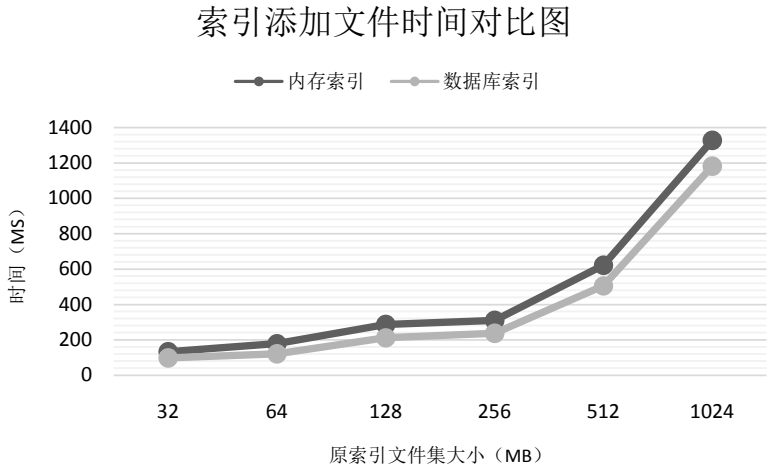

Figure 3. Comparison of adding new file
图 3. 添加文件时间对比图

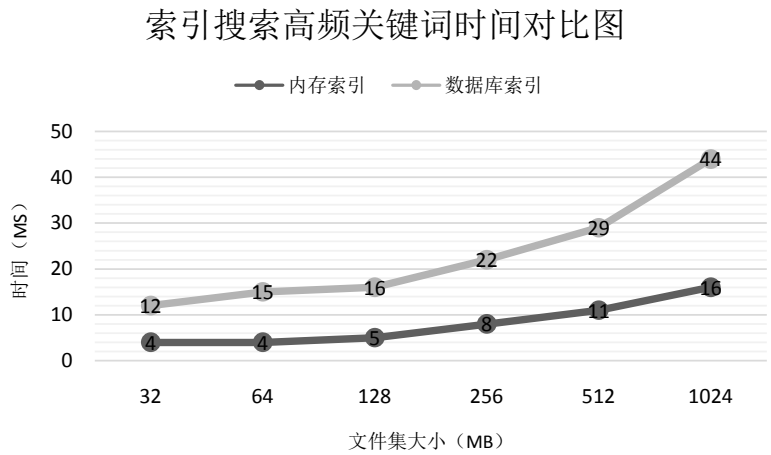

Figure 4. Effective comparison during keyword searching
图 4. 搜索高频关键词时间对比图

5. 结束语

研究了基于倒排索引的可搜索加密实现方式，针对大容量索引的持久化问题和并行搜索问题提出改进，通过使用图形数据库，为 KRB 树建模，并使用 lazy-load 的方法极大的提升了索引更新效率。通过性能测试实验，证明持久化的引入对性能的影响是可控的。为推进可搜索加密技术在工程实践中应用，做了尝试。

致　谢

感谢研究生导师范磊先生对本研究的悉心指导和帮助。

基金项目

上海市科委基础研究重点项目(No.13JC1403501)。

参考文献 (References)

[1] 冯登国, 张敏, 张妍, 等. 云计算安全研究[J]. 软件学报, 2011, 22(1): 71-83.

[2] Weber, T. (2011) Cloud Computing after Amazon and Sony: Ready for Primetime? http://www.bbc.co.uk/news/business-13451990

[3] Gentry, C. (2009) Fully Homomorphic Encryption Using Ideal Lattices. *Proceedings of the Annual ACM Symposium on Theory of Computing*, **9**, 169-178. https://doi.org/10.1145/1536414.1536440

[4] Goldreich, O. and Ostrovsky, R. (1992) Software Protection and Simulation on Oblivious RAMs. Massachusetts Institute of Technology, Cambridge.

[5] Goh, E.J. (2003) Secure Indexes. Cryptology ePrint Archive.

[6] Chang, Y.C. and Mitzenmacher, M. (2015) Privacy Preserving Keyword Searches on Remote Encrypted Data. Applied Cryptography and Network Security. Springer Berlin Heidelberg, 442-455.

[7] Curtmola, R., Garay, J., Kamara, S., *et al.* (2011) Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. *Journal of Computer Security*, **19**, 79--88. https://doi.org/10.3233/jcs-2011-0426

[8] Chase, M. and Kamara, S. (2010) Structured Encryption and Controlled Disclosure. Advances in Cryptology—ASIACRYPT 2010. Springer Berlin Heidelberg, 577-594. https://doi.org/10.1007/978-3-642-17373-8_33

[9] Kamara, S., Papamanthou, C. and Roeder, T. (2012) Dynamic Searchable Symmetric Encryption. *ACM Conference on Computer & Communications Security*, October 2012, 965-976. https://doi.org/10.1145/2382196.2382298

[10] Kurosawa, K. and Ohtaki, Y. (2012) UC-Secure Searchable Symmetric Encryption. Financial Cryptography and Data Security. Springer Berlin Heidelberg, 258-274. https://doi.org/10.1007/978-3-642-32946-3_21

[11] Liesdonk, P.V., Sedghi, S., Doumen, J., *et al*. (2010) Computationally Efficient Searchable Symmetric Encryption. *Secure Data Management*, *VLDB Workshop*, *SDM* 2010, Singapore, 17 September 2010, 87-100. https://doi.org/10.1007/978-3-642-15546-8_7

[12] Kamara, S. and Papamanthou, C. (2013) Parallel and Dynamic Searchable Symmetric Encryption. Financial Cryptography and Data Security. Springer Berlin Heidelberg, 258-274. https://doi.org/10.1007/978-3-642-39884-1_22

[13] Katz, J. and Lindell, Y. (2007) Introduction to Modern Cryptography (Chapman & Hall/CRC Cryptography and Network Security Series). Chapman & Hall/CRC, Stanford.

[14] Neo4J Index Introduction (2016). https://neo4j.com