学号 2019302180017

密级\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

武汉大学本科毕业论文

多用户场景下支持Join查询的

对称可搜索加密

院（系）名 称：国家网络安全学院

专 业 名 称 ：信息安全

学 生 姓 名 ：罗之龙

指 导 教 师 ：欧长海 教授

二○二三年四月

**郑 重 声 明**

本人呈交的学位论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本学位论文的知识产权归属于培养单位。



本人签名： 日期： 2023年4月24日

摘 要

随着云计算的发展，越来越多的用户将数据存储工作外包给云服务器。为了解决云存储的数据安全问题，研究者提出对称可搜索加密，此技术旨在不损坏云端数据隐私的前提下，为用户提供密态数据检索服务。然而目前大多数对称可搜索加密检索功能较为单一，不能满足实际需求。首先，大多现有方案仅支持单用户场景，数据提供者无法将云上的数据开放给多个用户进行检索。其次，大多现有方案仅支持简单数据库查询，所支持查询的匮乏性阻碍了对称可搜索加密技术在实际场景中的应用。针对上述问题，本文深入研究多用户场景下支持复杂查询的对称搜索加密，文章主要工作如下：

（1）针对多用户对称可搜索加密支持查询丰富度较低的现状，本文设计提出了首个多用户场景下支持Join查询的对称可搜索加密协议。方案在单用户对称可搜索加密协议JXT（Join Cross-Tags）的基础上设计而来，采取其将Join查询拆分为多个非Join子查询的思路，在高效支持Join查询的同时，实现用户在经过授权许可后，访问其他用户存储在云服务器上的机密数据。

（2）针对多用户对称可搜索加密方案设计中搜索令牌分发和服务端搜索令牌验证这两个核心问题，本文设计出子令牌拆分方案和同态签名方案分别解决了上述问题。在数据提供者接收检索者请求并给检索者分发搜索令牌时，协议通过子令牌拆分技术，将令牌的一部分交给检索者本地生成，降低了搜索令牌分发的通信开销。同时协议设计了一个基于一次性盲化因子的同态签名方案，使得服务端可以验证检索者发送的搜索令牌是否由数据提供者提供。

（3）本文对所设计的协议进行了正确性、安全性和性能开销的理论分析。安全性分析表明本协议满足恶意客户端、诚实且好奇服务端条件下的基于模拟的安全性；性能开销分析列出了协议在存储开销、计算开销和通信开销三方面的理论数据。此外，本文设计实验测试了协议的实际应用性能表现同理论性能分析的契合性，并证明了协议在查询规模较大的情况下有着较好的查询性能。

关键词：云存储；密态数据检索；对称可搜索加密；多用户场景；Join查询

**ABSTRACT**

With the development of cloud computing, more and more users are outsourcing data storage work to cloud servers. In order to solve the data security problem of cloud storage, researchers proposed symmetric searchable encryption, which aims to provide users with secret data retrieval services without damaging the privacy of cloud data. However, most of the current symmetric searchable encryption implementations only support single retrieval functionality and cannot meet actual needs. First of all, most existing solutions only support single-user scenarios, and data providers cannot open cloud data to multiple users for retrieval. Secondly, most existing solutions only support simple database queries, and the scarcity of supported query types hinders the application of symmetric searchable encryption technology in actual databases. To solve these problems, this article studies symmetric searchable encryption that supports complex queries under multi-user scenarios deeply, and the main work of this article is as follows:

(1) In response to the low richness of queries supported by multi-user symmetric searchable encryption, this article designs and proposes the first symmetric searchable encryption protocol that supports Join queries under multi-user scenarios. The scheme is designed based on the single-user symmetric searchable encryption protocol JXT (Join Cross-Tags), adopts its idea of splitting Join queries into multiple non-Join sub-queries, and efficiently supports Join queries while allowing users to access confidential data stored on cloud servers after authorization.

(2) In response to the two core problems of search token distribution and server search token verification in multi-user symmetric searchable encryption study, this article designs sub-token splitting scheme and homomorphic signature scheme respectively to solve these problems. When data providers receive search requests from searchers and distribute search tokens to them, the protocol uses sub-token splitting technology to let part of the token be generated locally by searchers, reducing communication overhead for distributing search tokens. At the same time, the protocol designs a homomorphic signature scheme based on a one-time blinding factor so that servers can verify whether the search tokens sent by searchers are provided by data providers.

(3) This article conducts theoretical analysis on correctness, security and performance overhead of the designed protocol. Security analysis shows that this protocol satisfies simulation-based security under malicious clients and honest but curious servers; performance overhead analysis lists theoretical data in terms of storage overhead, computation overhead and communication overhead for this protocol. In addition, this article designs experiments to test the actual application performance of the protocol and proves its consistency with theoretical performance analysis, demonstrating that the protocol has good query performance when query scale is large.

**Keywords:** cloud storage; secret data retrieval; symmetric searchable encryption；multi-user scenario; Join queries

目 录

[1 绪论 1](#_Toc133453969)

[1.1 研究背景 1](#_Toc133453970)

[1.2 国内外研究现状 2](#_Toc133453971)

[1.2.1 单用户对称可搜索加密 2](#_Toc133453972)

[1.2.2 多用户对称可搜索加密 3](#_Toc133453973)

[1.3 研究内容 3](#_Toc133453974)

[1.4 章节安排 4](#_Toc133453975)

[2 相关工作 5](#_Toc133453976)

[2.1 关系型数据库与Join查询 5](#_Toc133453977)

[2.1.1 关系型数据库 5](#_Toc133453978)

[2.1.2 Join查询定义 6](#_Toc133453979)

[2.2 对称可搜索加密 7](#_Toc133453980)

[2.2.1 对称可搜索加密定义 7](#_Toc133453981)

[2.2.2 对称可搜索加密正确性 8](#_Toc133453982)

[2.2.3 对称可搜索加密安全性 8](#_Toc133453983)

[2.2.4数据结构 8](#_Toc133453984)

[2.3 JXT协议 9](#_Toc133453985)

[2.3.1 JXT实现Join查询的思路 9](#_Toc133453986)

[2.3.2 两个子查询的实现 10](#_Toc133453987)

[2.3.3 协议具体流程 11](#_Toc133453988)

[3 MC-JXT协议设计 14](#_Toc133453989)

[3.1 多用户对称可搜索加密定义 14](#_Toc133453990)

[3.2 MC-JXT协议构造 15](#_Toc133453991)

[3.2.1 算法 15](#_Toc133453992)

[3.2.2 算法 18](#_Toc133453993)

[3.2.3算法 21](#_Toc133453994)

[3.3 MC-JXT协议正确性分析 24](#_Toc133453995)

[3.4 MC-JXT协议安全性分析 25](#_Toc133453996)

[3.4.1 攻击者为服务端 时的安全性 25](#_Toc133453997)

[3.4.2 攻击者为客户端 时的安全性 25](#_Toc133453998)

[3.5 MC-JXT协议理论开销分析 31](#_Toc133453999)

[3.5.1 存储开销 32](#_Toc133454000)

[3.5.2 计算开销 32](#_Toc133454001)

[3.5.3 通信开销 33](#_Toc133454002)

[4 协议实现以及性能测试 34](#_Toc133454003)

[4.1 协议实现 34](#_Toc133454004)

[4.2 性能测试 34](#_Toc133454005)

[4.2.1 数据集 34](#_Toc133454006)

[4.2.2 实验设计以及实验结果 35](#_Toc133454007)

[4.2.3 实验结论 39](#_Toc133454008)

[5 总结与展望 41](#_Toc133454009)

[5.1 本文总结 41](#_Toc133454010)

[5.2 工作展望 41](#_Toc133454011)

[参考文献 43](#_Toc133454012)

[致谢 46](#_Toc133454013)

# 绪论

## 研究背景

随着云计算技术的快速发展，越来越多的个人和组织将大规模数据的存储和处理工作外包给第三方服务器，以降低本地信息的数据管理开销以及系统维护成本。然而，数据安全威胁也随之而来，恶意的云端服务器管理员或者不具有访问权限的服务器用户可能会在执行正常存储和查询功能之外，试图获取数据库信息，从而导致用户隐私数据的泄露。近年来数据安全问题日益突出，Google、Sony、Yahoo等公司都发生过大规模的用户数据泄露事件[1]。

为了解决云端数据隐私安全问题，数据提供者必须要对上传到云端上的数据进行加密，阻止不可信的服务器获取数据明文信息。如果使用传统的加密算法进行加密，由于数据对服务器是不可见的，服务器无法向客户端提供数据检索服务。一种保持数据可检索性的方案是客户端将整个加密的数据库下载下来，本地将整个数据库进行解密，然后查询所需要的信息（i），但此方案开销太大。另一种方案是数据提供者将密钥发送给服务端，让服务端解密数据并进行查询，但显然此方法会将用户明文数据直接暴露给服务端，与安全要求相悖。总而言之，决绝上述问题需要一种能直接在密态数据上提供完整数据检索服务的技术。

能实现此种需求的技术有很多，包括对称可搜索加密[2]（Searchable Symmetric Encryption，SSE），不经意随机存取[3]（Oblivious Random Access Memory，ORAM）等，其中目前得到广泛研究和应用的是对称可搜索加密技术。对称可搜索加密是一种可以实现在不泄露明文信息的前提下，对密文数据进行匹配和检索的特殊加密技术。在其场景中，数据提供者首先用特殊加密算法将自己的明文数据库（或者数据库索引）加密并发送给服务端，服务端接收并将密态数据库存储下来；当客户端要发起查询的时候，其根据自己要查询的关键字生成一个对应的搜索令牌，并发送给服务端；服务端接收到搜索令牌，将其和密态数据库中的记录进行匹配，如果达到预期匹配结果，就将此密态记录返回给客户端。客户端接收密态查询结果，并解密得到最后的查询结果。

对称可搜索加密分为单用户和多用户两种场景。前者支持数据提供者在向云端上传密态数据后让自己进行数据检索，后者则支持数据提供者将服务器上的密态数据开放给其他多个用户进行检索，﻿满足云环境下多用户共享加密数据的需求。 Join查询是关系型数据库中的一类基础查询，在数据库管理系统中，大多数复杂的查询都涉及Join查询。目前学术界已有许多关于多用户对称可搜索加密的研究，但仍局限于简单类型的查询（如单关键字查询），关于支持Join查询的多用户对称可搜索加密方案的研究比较匮乏，给对称可搜索加密的实际应用带来了阻碍。

## 国内外研究现状

### 单用户对称可搜索加密

对称可搜索加密早期的研究主要是针对单用户场景的。Song等人[2]在2000年提出了首个单用户对称可搜索加密方案实现。其方案使用确定性加密对文档中的每一个关键字进行加密，同时在密文中附加一个特殊形式的哈希值，服务端可以通过对特殊哈希值进行计算确定文档匹配与否。Goh等人[4]在2003年提出了安全索引的概念，其方案对明文数据库中的每个文档建立基于布隆滤波器的索引，服务端通过验证待检索关键字是否存在于文档的索引中来确定其在文档中的存在性。此方案相比传统方案大大降低了查询复杂度，后续许多对称可搜索加密方案均采用类似建立密态索引的思想。﻿Chang 等人[8]通过构建伪随机位来掩盖每个文件的基于字典的关键字索引，从而游有效解决关键字搜索的隐私问题。Curtmola等人[5]延续了Goh方案中安全索引的思想，提出了CGK方案，此方案建立基于关键字的倒序索引，为每个关键字维护了一个加密链表，链表的每个节点元素包含一个含有此关键字的文档ID以及解密下一个节点的密钥。CGK实现了非线性（最优的）的查询时间开销。后续﻿学者们对对称可搜索加密的各个问题进行了深入研究，其中包括密文增加、删除、修改的动态更新问题[10-12]，根据查询模式优化搜索效率的问题[19-21]以及如何提高方案安全性的问题[22-23]。

以上介绍的方案仅支持单关键字查询，后来支持更加复杂数据库查询的单用户对称可搜索加密方案陆续出现。Golle等人[6]提出了第一个支持联合查询的对称可搜索加密方案，其核心思想是为文档设定一个关键字域，在查询时，客户端根据自己要查询的关键字连同其域生成一组搜索令牌并发送给服务端，服务端将搜索令牌和密态文档进行对比，从而确定查询结果。Ballard等人[7]提出了一个定义在标准模型下并支持联合查询的对称可搜索加密方案，该方案基于基于Shamir秘密分享以及关键字域技术。Wang等人[8]使用双线性映射替代了关键字域方案，首次提出了不需要关键字域的，支持联合查询的对称可搜索加密方案。Park等人提出了基于汉明距离的近似串匹配技术的，支持模糊查询的对称可搜索加密方案；后续Adjedj等人又提出了基于局部敏感哈希技术的，支持模糊查询的对称可搜索加密方案。Cash等人[9]里程碑式地提出了首个支持非线性查询时间复杂度，以及联合查询、布尔查询的对称可搜索加密方案，该方案在大规模数据场景中表现优秀，且支持任何结构的数据。Seny Kamara[10]、David Cash[11]陆续提出了支持Join查询的对称可搜索加密方案；近日，Jutla等人[12]提出了高效支持Join查询的对称可搜索加密方案（JXT, Join Cross-tag Protocol），大大降低了传统方案的空间开销。

### 多用户对称可搜索加密

前一节介绍了单用户场景对称可搜索加密，后来为了解决多用户数据共享的需求，多用户对称可搜索加密应运而生。在多用户场景中，用户被分离为数据提供者和数据检索者两种角色，数据提供者可以将其加密数据上传至服务端，数据检索者在发起查询时需要先从数据提供者获取相应搜索令牌，这样才可以进行和服务端进行交互查询。同时服务端在收到搜索令牌之后会验证其合法性，即其是否由数据提供者生成，以阻止未经数据提供者认证的查询。在验证通过后，服务端就像单用户场景那样执行检索任务并返回检索者查询结果。

Curtmola等人[5]在2006年提出了基于广播加密技术的多用户对称可搜索加密方案。在此方案中，数据提供者将密钥分发给准许检索群组中的用户，以便这些用户生产搜索令牌并进行数据检索；。Raykova等人[13]在2009年提出了基于重路由加密技术的多用户对称可搜索加密方案。此方案引入一个第三方实体查询路由来进行查询的认证，检索者将加密后的查询发送给查询路由，若认证通过，查询路由将查询进行转换并发送给服务端进行查询，以此实现检索者匿名地进行数据检索。﻿Bao 等人[48]利用引入用户管理器管理用户，实现数据库上的的关键字搜索与数据共享。Zhao 等人[50]利用属性基加密和属性基签名实现用户管理，满足了细粒度的数据共享。﻿Miao 等人[60]利用基于身份的加密技术，提出了匿名可撤销的多用户搜索方案。Jarecki等人[14]在2013年提出了支持联合查询和布尔查询的多用户对称可搜索加密方案，大大扩展了多用户对称可搜索加密的查询丰富性。﻿

## 研究内容

由上述分析可知，学术界目前对支持复杂查询的多用户对称可搜索加密方案的研究比较匮乏，给对称可搜索加密在数据共享场景中的实际落地造成了阻碍。本文深入研究多用户场景下支持复杂查询的对称搜索加密，文章研究内容如下：

（1）设计提出了首个多用户场景下支持Join查询的对称可搜索加密协议，并称之为MC-JXT（Multi-Client Join Cross-Tags）方案在单用户对称可搜索加密协议JXT的基础上设计而来，采取其将Join查询拆分为多个非Join子查询的思路，在高效支持Join查询的同时，实现用户在经过授权许可后，访问其他用户存储在云服务器上的机密数据。

（2）针对多用户对称可搜索加密协议设计中搜索令牌分发和服务端搜索令牌验证这两个核心问题，本文提出子令牌拆分方案和同态签名方案解决上述问题。在数据提供者接收检索者请求并给检索者分发搜索令牌时，协议通过子令牌拆分技术，将令牌的一部分交给检索者本地生成，降低了搜索令牌分发的通信开销。同时协议设计了一个基于一次性盲化因子的同态签名方案，使得服务端可以验证检索者发送的搜索令牌是否由数据提供者提供。

（3）本文对所设计的协议进行了正确性、安全性和性能开销的理论分析。安全性分析表明本协议满足恶意客户端、诚实且好奇服务端条件下的基于模拟的安全性；性能开销分析列出了协议在存储开销、计算开销和通信开销三方面的理论数据。此外，本文设计实验测试了协议的实际应用性能表现同理论性能分析的契合性，并证明了协议在查询规模较大的情况下有着较好的查询性能。

## 章节安排

本文共分为五个章节，具体内容安排如下：

第一章绪论。本章主要介绍本文的研究背景、意义和内容。本章大致梳理了当前对称可搜索加密领域的研究进展和国内外研究现状，最后对本文的研究内容进行总结陈述。

第二章相关工作。本章主要介绍和本文研究问题相关的背景知识。首先介绍对称可搜索加密的语义、正确性、安全性定义。接着介绍Join查询的相关概念以及Join查询在关系型数据库的重要性。最后介绍JXT协议的构造。

第三章MC-JXT协议设计。本章主要阐述本文设计多用户场景下支持Join查询的对称可搜索加密方案的思路、方案的先进性以及所设计的协议的具体算法。此外我们还对协议的可行性，正确性、安全性以及协议的理论性能进行了分析。

第四章协议实现和性能测试。本章将介绍MC-JXT协议的代码实现方式和其在选定数据集上测试协议的性能表现，并论证实验数据和理论分析之间的一致性。

第五章总结与展望。本章对本文前面章节的内容进行总结，阐释协议的优势以及目前存在的局限性。同时还会提出未来可以优化改进的方向和开放性问题。

# 相关工作

本章首先介绍对称可搜索加密技术的概念、正确性定义和安全性定义，其次介绍一种在本文和其他相关工作用到的，称为的数据结构，其本质是一个单用户单关键字对称可搜索加密实现。之后本章介绍Join查询的概念和形式化定义，以及在数据库查询中的重要性。最后本章介绍高效支持Join查询的协议JXT，本文MC-JXT协议就是在其基础上设计提出的。

## 关系型数据库与Join查询

### 关系型数据库

（1）数据库及表的形式定义

一个关系型数据库是一个表的集合，本文将其表示为。每张表由若干记录组成，本文将关键词定义为关系型数据库中的属性以及对应值组成的键值对，那么每条记录就是一个关键词集合。而在表中存储记录时，除了记录关键词集合之外，通常还要额外为每个记录设置标号来标识每个记录，故本文将数据库中的表表示为以下形式：。此处是一个字符串，代表着第张表中第条记录的标号；代表着第张表中第条记录的关键词集合。此外，我们用指代表中所有关键词组成的空间。

上述的记录标号是一个在数据提供者在建立数据库时，标识各个记录的值，其主要作用是为了方便数据库查询，减小通信开销。服务端在执行查询之后，得到满足查询的记录标号并返回查询者，而不是直接返回符合查询的记录。在对称可搜索加密的场景下，通常是原标号置换加密后的值。本文假定记录标号同时与记录和记录所在的表相关，即指不同的表之间没有重合的记录标号。

（2）倒序索引

值得一提的是，在数据库上执行查询时，数据库通常不会直接在原始表数据上查询满足条件的记录，因为这样做开销太大，而是会先在表上建立索引结构，然后在索引上执行查询，以提升查询效率。数据库索引有很多种，其中一种常用的，查询效率较高的索引是倒序索引。令表示第张表低第张中，包含关键词的记录标号集合，即，则倒序索引中的每一项由一个数据库关键词索引，其值为包含此关键词的记录标识符集合，即：

### Join查询定义

（1）Join查询的概念

在数据库管理系统中，Join运算是指根据两个或多个表之间指定的共有属性进行组合，并生成一个新表的过程。其作用是在表之间建立一种关系，反馈给用户多个表中的不同实体的信息，提升数据库查询的灵活性。Join查询就是指要需要查询来自多个表的字段的数据库查询，在数据库管理系统中，大多数复杂的查询都涉及Join查询。要实现查询不同表字段的功能，通常我们需要先计算出多个表之间的Join表，然后进一步在Join表进行字段的匹配和查询。

（2）Join属性与Join关键词集

本文称Join运算中所涉及的，表之间的共同属性为Join属性。具体地，本文将和其他表共有属性表示为，为Join属性的个数。这个属性集合时建立数据库伊始就确定的。同时，本文将记录其Join属性集上的关键词集合称为其Join关键词集，记为：

（3）Join查询的形式化定义

对于一个涉及和的Join查询，其形式化定义为:

其中, 是要查询的，分别来自和的关键词，而 则是要查询的，和 之间的Join属性集合。本文用指代Join查询的结果，实际上是一个记录标识符对的集合，满足：

, 且满足:

上述表达式的含义为：,分别为和中包含的两条记录，且这两条记录之间在Join属性集合上有相同的关键词。

## 对称可搜索加密

### 对称可搜索加密定义

本节给出对称可搜索加密的形式化定义。一个对称可搜索加密方案主要包含涉及服务端和客户端的两个算法: 和，方案构造如图 2‑1所示，算法具体语法如下：

（1）：此算法由客户端进行，其输入为明文数据库，输出为 对应的加密数据库和其对应的私钥。值得注意的是，此处客户端在单用户场景下同时充当数据提供者和检索者两个角色，而在多用户场景下这两个角色分别由两个不同实体充当。

（2）：此算法由客户端和服务端共同执行，客户端输入为私钥 以及要发起的查询，服务端输入为中输出的密文数据库。最终，在客户端和服务端进行交互之后，客户端输出满足查询的记录标识符集合 ，而服务端没有输出。

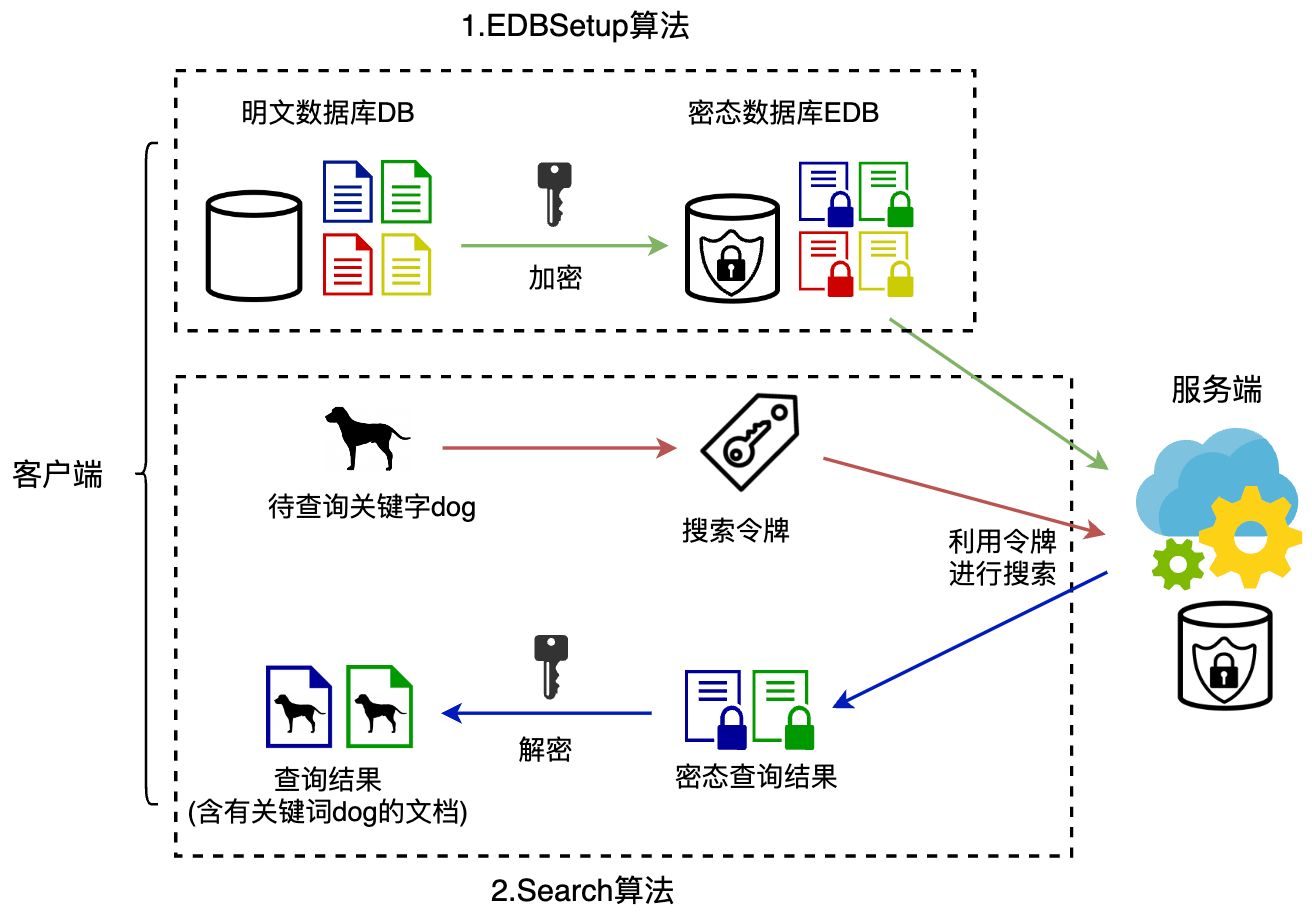


图 2‑1 对称可搜索加密流程图

### 对称可搜索加密正确性

对称可搜索加密方案的正确性定义如下：对于所有输入的数据库和查询 ，在协议执行完上述的和算法之后，如果客户端的输出等于 的话，那么我们就称此方案满足正确性定义，反之，则不满足。

### 对称可搜索加密安全性

对称可搜索加密主要考虑攻击者为服务端时的安全性，其语义安全定义在由泄露函数参数化的理想/现实范式框架[15]下，此处泄露函数指协议交互过程中客户端向攻击者（即服务端）泄露的信息上界。

理想/现实范式定义了现实世界和理想世界两个世界，现实世界中运行着真实的协议，存在一个攻击者与协议进行交互；理想世界中运行着可信第三方可以协助协议参与者进行计算，满足所有安全性要求。在攻击者可以自由选取数据库和查询的前提下，如果对于任意现实世界的攻击者，理想世界都存在一个模拟者可以只通过泄露函数模拟出此攻击者在现实世界的视角，那么我们就说此协议是安全的。

### 数据结构

JXT协议以及MC-JXT协议都用到了一种称为 的数据结构，其本质就是一个单关键字对称可搜索加密的实现。是一种密态倒序索引实现，与明文倒序索引不同的是，中的键和相应的值都是以密态的方式存储的，其安全性和正确性满足对称对称可搜索加密的定义。

假设关系型数据库中 对应的倒序索引为，即有：

则关于有以下三个算法：

（1）。此算法接收明文数据库的倒序索引 作为参数，并输出在此索引上建立的密态倒序索引结构和用到的密钥，输出的是一个数组，其中的元素对应于明文索引。

（2） 。此算法用于生成上的搜索令牌，算法输入要查询的表的标号，关键字和密钥，输出一个相应的搜索令牌 。

（3） 。此算法接受、要查询的表的标号、以及中生成的，最终返回表中包含关键字 的记录标识符集合。

## JXT协议

JXT协议（Join Cross-Tag Protocol）由Jutla等人[12]在2021年提出，是首个无需预运算过程，并且能高效支持Join查询的对称可搜索加密方案。学术界中传统的支持Join查询的可搜索加密方案在建立密态数据库时往往需要预先计算待查询表之间的Join表。而由Join运算的性质可知，这一步可能会造成存储空间的爆炸性增长，从而导致密态数据库增长规模很大。JXT协议的优势主要在于此方案不需要任何上述的预运算，但仍然能实现对称可搜索加密场景下的Join查询同时保持可靠的安全性，这使得JXT协议在大规模数据场景下有很大的存储性能优势。

### JXT实现Join查询的思路

JXT协议实现Join查询的主要思路是将一个Join查询拆分成多个非Join类型的子查询 ，然后分别实现子查询，从而实现在不预先计算表之间的Join表的前提下实现Join查询。下面本节考虑一个服务端在接收数据提供者明文数据库和明文查询的情况下，执行指定Join查询的例子：设表A（tabA）和表B（tabB）之间有Join属性personID，表 A有待查询属性name，表 B有待查询属性city，考虑如下查询：

服务端将此查询拆分为和两个子查询。子查询在表A中查询满足（name, NN）字段的记录的personID值集合:

将子查询的结果集合记为 。而子查询则是在表B中查询满足（city, CC）且personID属性上的值处于查询结果之中的记录：

服务端遍历查询结果中的每个personID值，在表B中寻找personID匹配并且city字段也匹配的记录。最终的查询结果为，是 子查询结果中每个personID对应的记录，是表B中匹配（city, CC），并且匹配记录对应的personID字段的记录。

总结以上示例，本文将一个涉及两个表、两个字段的Join查询拆分成两部分查询。在中查询包含的记录的Join关键词集，记为:

然后在中进行一个联合查询，查询包含同时Join关键词集在查询结果当中的记录。

满足

在执行完之后，每一对匹配的就是查询的结果。其正确性是显然的，和有着相同的Join关键词集，并且各自匹配两个查询字段，所以这两个记录Join运算后得到的新记录必然是匹配查询的。

### 两个子查询的实现

其实就是一个单关键字查询，为了增加查询的效率，数据提供者可以在数据库索引中的每个单元增加一个字段，事先就把每条记录对应的Join关键词与其记录标识共同存储起来，即将倒序索引中每个对应的替换为

与此同时，倒序索引更新为。通过此种构造，服务端仅仅查询表中包含中的记录就可以直接获得所需的Join关键词集，以节省计算开销。

是一个联合查询，第二个字段是查询得到的Join关键词集，服务端要完成查询，只需先将满足的记录查询出来，然后再逐个验证此记录的Join关键词集是否在查询的结果之中。此处服务端需要大量地验证满足的记录标号同查询结果中每一个Join关键词的匹配关系。为了高效地完成验证，数据提供者生成一个叫做的查找表，并交给服务端。在建立密态数据库时，数据提供者为每一张表生成对应的，其每一单元存储着一个记录和其对应的Join关键词集，定义如下:

此处是伪随机函数。这样通过，服务端就可以快速确定一个给定的记录 和一个给定的Join关键词集是否匹配，只需验证是否成立即可。

### 协议具体流程

JXT协议总共包含两个算法，分别是和，下面本节首先介绍一个初级版本的JXT协议，然后再引出最终的JXT协议构造。

（1）初级JXT协议构造

由数据提供者执行，接受明文数据库作为输入，输出密态数据库和一组密钥，后续被存储在服务端上以供查询。中，对于每张表，数据提供者首先建立其对应的，即将每条记录的标识符和其对应的Join关键词集的匹配关系存储下来，正如2.3.2。在JXT中 ，和分别是和连同每个Join属性t经过另一个伪随机函数计算之后的值。然后，数据提供者对每张表建立密态倒序索引，即 ，它存储着每个关键词 和包含该关键词的记录标识符集合。由搜索令牌进行索引，正如2.3.2所提到的那样，不仅是个记录标识符集合，还存储着各个标识符对应的Join关键词集(伪随机函数值形式)，其构造为:

为经过加密的，而和的含义就像前面所提到的那样，用于后续的查询。

过程从更高的层面来看就是一个涉及客户端和服务端的两方计算，客户端的输入是Join查询，服务端的输入是密态数据库。最终客户端输出符合Join查询的记录标识符集合，服务端不输出任何信息。客户端选择Join查询，首先进行查询，客户端计算出 并发送给服务端，服务端用检索得到，得到满足的记录对应的Join属性集。然后就是查询的执行过程，客户端计算出并发送给服务端，服务端用其检索出 ，得到满足 的记录以及其对应的。接着服务端遍历每一组，并验证对于每个 是否成立，也就是验证中的记录和中的是否有相同的Join关键词集，如果满足，则服务端返回给客户端，客户端进一步解密得到一对查询结果。

（2）最终的JXT协议构造

显然初级JXT协议是不安全的。一方面，构造直接向服务端暴露了 和，同时过程中也向服务端暴露中元素。要解决安全性问题，就要对原协议做如下的修改：

首先在中协议给和加上掩码，在中存储掩码后的值，此时中每个单元表示为：

其次在中，我协议使用掩码后的和去计算 并进行验证，这对掩码的设计提出了要求。

改进过后的中每个单元内容为，其中

这里针对和有两组掩码值，分别为和，这两组掩码值都是关于的伪随机函数值，区别是使用了不同的密钥。 在计算中加入了计数值，中每个单元都有一个对应的值，存在的意义是使得中不同单元的掩码值不相同。

在改进后的 中，客户端额外向服务端发送两组辅助查询的令牌：

其中，。服务端利用这两组令牌就可以在和被掩盖的情况下仍然计算出正确的。服务端用两个计数器分别遍历和，并验证这每对记录是否匹配。对于查询中的每一个Join属性，即每个，分别计算

最后计算是否属于，如果全部满足条件，则将对应的一对密态标识符返回给查询者，查询者进行解密得到查询结果。

# MC-JXT协议设计

本章主要介绍MC-JXT的算法流程以及对协议正确性、安全性、和性能开销的理论分析。正确性方面，本章会给出协议的正确性数学定义和证明；安全性方面，本章会给出协议的安全性数学定义，泄露函数以及基于模拟的安全性证明；性能开销方面，本章会给出协议存储开销、计算开销、通信开销三方面的分析。

## 多用户对称可搜索加密定义

不同于单用户场景中同一客户端既提供数据又检索数据，多用户对称可搜索加密方案（以下称MC-SSE）中的客户端被分离为数据提供者和数据检索者 两种角色。数据提供者持有明文数据库以及其自己指定的查询政策，服务端存放着供检索的密态数据库，数据检索者在发起查询时需要先向 请求搜索令牌，拿到搜索令牌后才能同服务端进行交互，获得最终的查询结果。

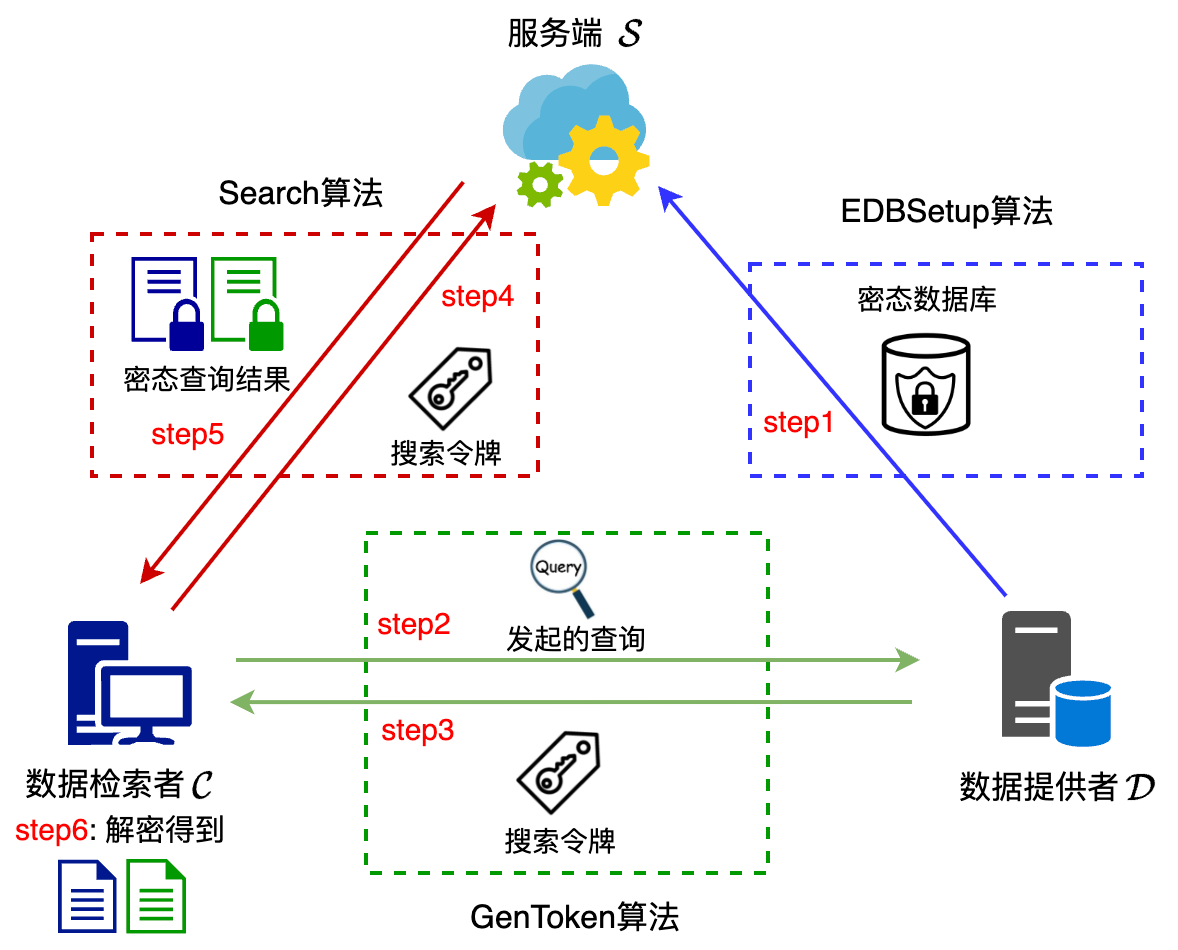


图 3‑1 多用户对称可搜索加密流程图

换言之，多用户场景下把自身数据的查询权外包给了第三方实体，并要求在从自己这里获得搜索令牌之后才能进行查询。多用户场景下，数据提供者 要根据请求将搜索令牌分发给 ，并希望只能用自己签发的搜索令牌才能进行查询。达到这一目标的主要方法是让对搜索令牌进行数字签名，然后在接收到发送的搜索令牌之后对签名进行验证，只有通过签名通过验证，才会用其进行查询，这就保证了对的查询控制。在实际多用户对称可搜索加密方案设计中，研究者主要关注解决以下两个问题：（1）如何高效地给分发搜索令牌；（2）采用何种方案对搜索令牌进行签名。

MC-SSE包含三个算法。首先运行生成和密钥；接着以发送来的查询和密钥作为参数运行算法，并产生签名后的搜索令牌；然后和共同运行协议，输入，输入签名后的搜索令牌，验证搜索令牌是否由签发，如果是则继续进行查询，最终输出查询结果，不输出任何信息，否则协议终止，具体流程见图 3‑1。

## MC-JXT协议构造

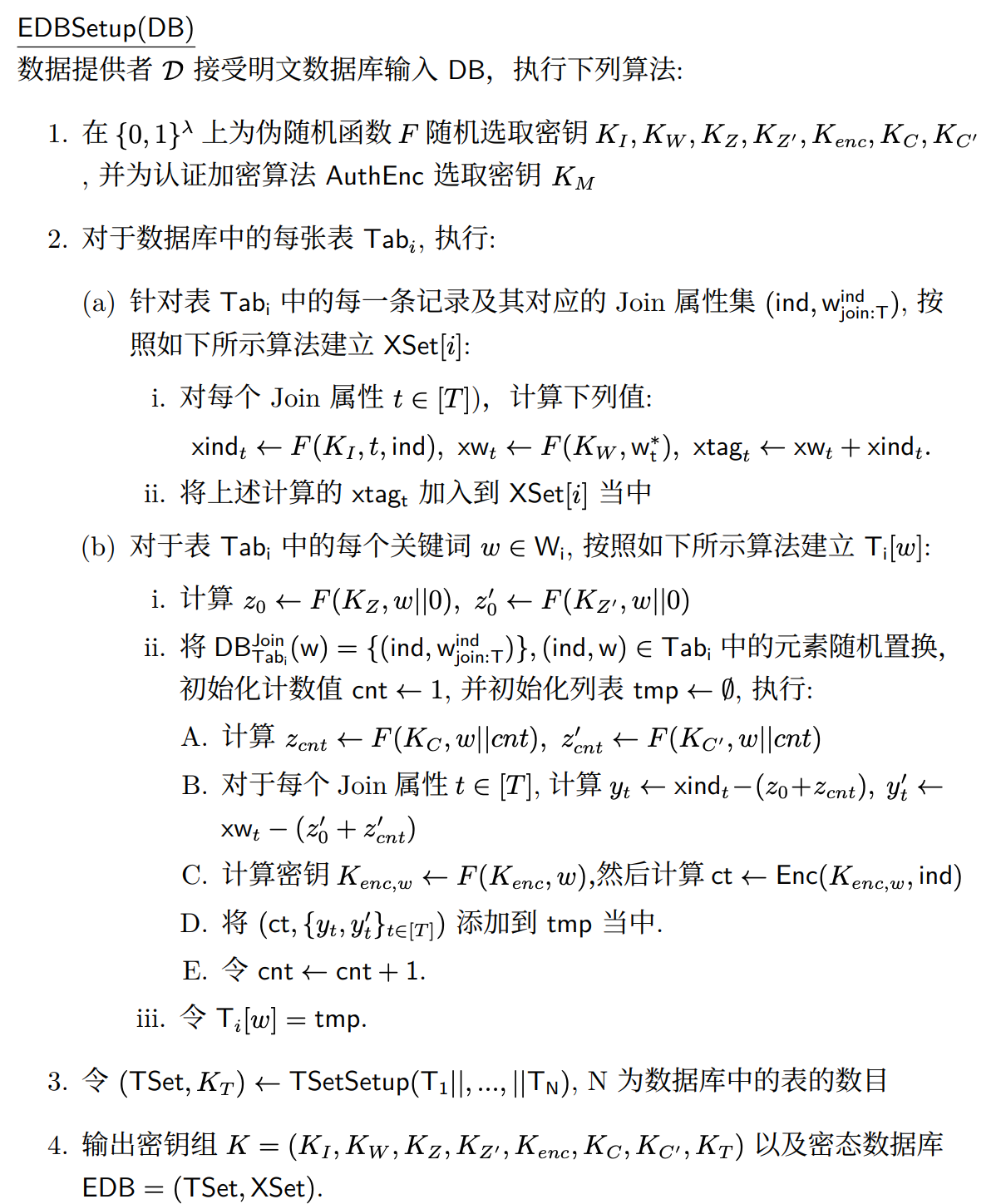
本节介绍MC-JXT协议的构造。MC-JXT协议同JXT协议一样支持高效的Join查询，同时满足上述MC-SSE的定义。MC-JXT协议同样由三部分算法构成，具体结构见图 3‑2。

此处列举协议中用到的一些数学定义和密码学原语，本节将表记为，其关键词空间为，表和表之间的Join查询如公式（2.4）定义为

。本节将两表之间的Join属性数目记为, 标识符为的记录对应的Join关键词集记为，其定义如公式（2.3）所示。本节用表示协议中使用到的伪随机函数，表示认证加密算法，相应地， 表示认证解密算法。

### 算法

算法由数据提供者执行，接受明文数据库作为输入，接算法对要用到的密钥组进行采样，并对中的每张表建立的结构和 结构，最后算法输出密态数据库，以及密钥组。值得强调的是会将密钥共享给，用于让验证发送过来的搜索令牌上的签名，具体算法如算法 1所示。本小节将会从和的生成两部分介绍此算法。



算法 1 MC-JXT-EDBSetup

（1）生成过程

存储着每条记录和其对应的Join关键词集的匹配关系，MC-JXT协议的构造同JXT协议基本相同。对于表中的记录，针对每个Join属性，计算下列值:

是在上随机选取的密钥。接着算法计算上面描述的“匹配关系”标识：

最后，算法将加入到。

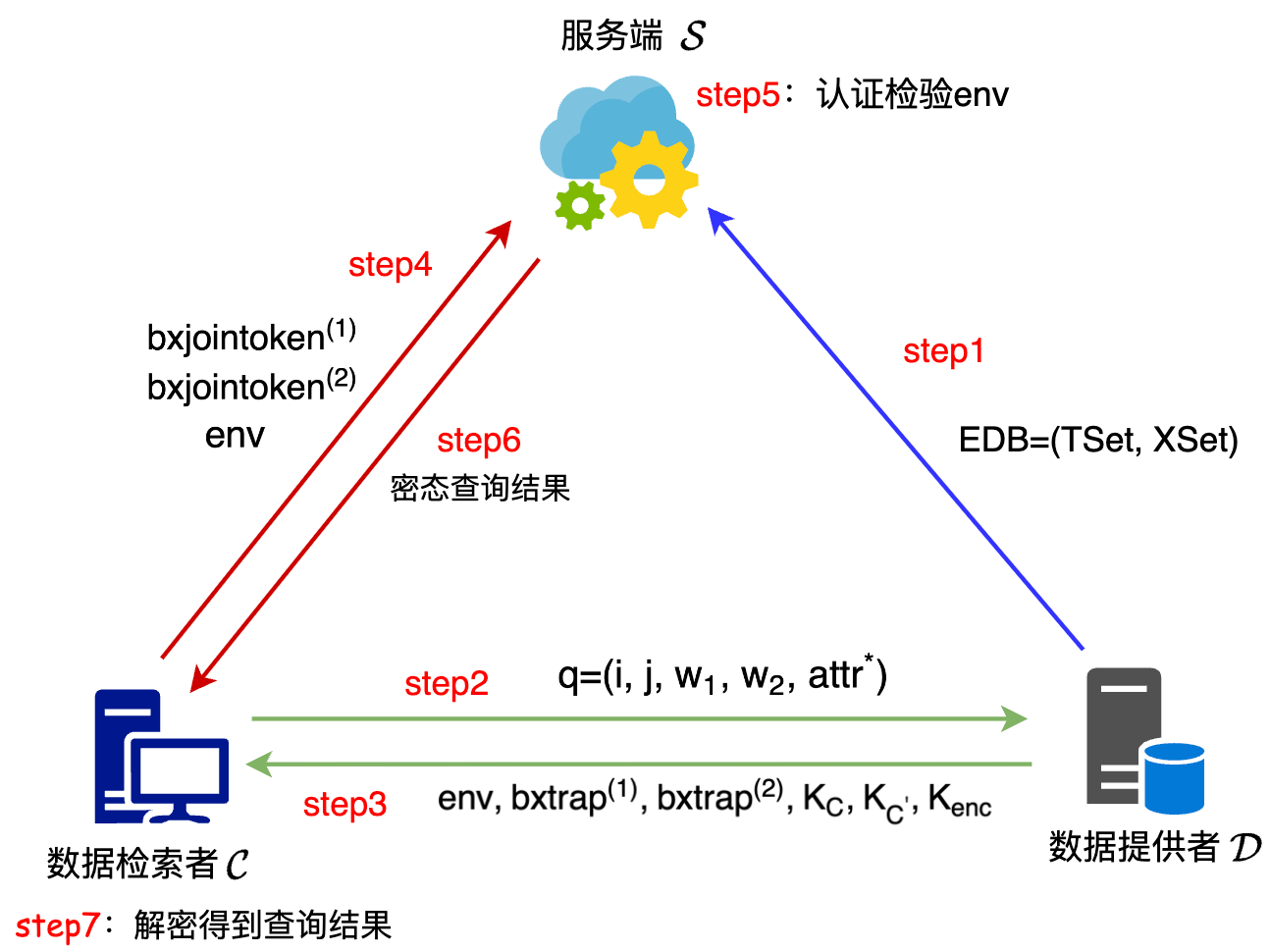


图 3‑2 MC-JXT协议流程图

（2）生成过程

对于每一个关键词，存储着对中包含关键词的记录标识符集合加密后的值，以及相应的支持Join查询功能的额外字段。 算法建立的步骤如下:

1）生成如下一对掩码值:

此处是在上随机选取的密钥，是选定的伪随机函数，代表将 和0值拼接起来作为伪随机函数的输入。

2）对中的每一个元素，设为，都赋予一个不同的计数值，并针对计算另一对额外的掩码值:

此处 是在上随机选取的密钥。

3）接着对每个Join属性，算法计算：

4）然后计算对应的加密记录标识符:

此处如同前面的密钥，在上随机选取。

5）最后算法构造出元组加入中。

这里计算掩码过程不同于JXT的地方在于，JXT中计算额外掩码值的方法时使用的密钥和计算时的相同，都是，而MC-JXT计算额外掩码值采用了另一组不同的密钥。这样做的目的是为了降低后续 时和 之间的通信开销，具体原因见3.2.2。

以上就是和的建立过程，如本节开头所解释的那样，最终算法还要随机选取一个密钥，和、以及其他密钥一同作为算法输出。

### 算法

算法是MC-JXT相比于JXT协议独有的算法。算法由数据提供者 和检索者交互进行，输入一个Join查询，输入查询审查政策和密钥组。算法执行过程中，将发送给进行审查，如果判断 满足，则将与对应的搜索令牌进行签名并分发给，否则终止协议。

如3.1所述，研究者主要关注如何解决以下两个问题：（1）如何高效地给 分发搜索令牌；（2）采用何种方案对搜索令牌进行签名。

对于问题一，本文协议的解决方式是采用子令牌拆分方案，来降低令牌分发的通信开销。协议将原本的搜索令牌拆分成两个子搜索令牌，让其中一部分由 生成并分发给，另一部分则由自己生成，并由其组装成完整的令牌，然后和进行查询交互。

对于问题二，本文协议的解决方式是设计一种同态签名方案来实现令牌的有效签名。对要分发给的子搜索令牌进行签名，即使用一次性盲化因子做掩盖运算，接着由于掩盖运算的同态性，导致使用签名后的子令牌组装成的完整令牌仍然保持着的签名，这样就在保持低令牌分发开销的同时，完成对令牌的签名。然后使用认证加密算法将这些一次性盲化因子封装成数字信封，并连同签名子令牌一同发送给，在完成令牌组装之后将得到的已签名搜索令牌连同数字信封发送给。由于与共享认证加密算法密钥（上一节提到的），所以 可以解密并恢复出盲化因子，从而验证收到的已签名搜索令牌的合法性，如果合法，则可以通过逆掩盖运算恢复出原始令牌，否则，则得不到有效的搜索令牌。这两部分的具体方案设计如下所示。完整如算法 2所示。

（1）高效搜索令牌分发方案

2.3.3中提到对于Join查询，其对应的搜索令牌构造为：

其中，。为 中各个元素对应的掩码值，同理。一种简单的分发方式是 接收到发送来的查询并审查通过后，直接将搜索令牌全部发送给。但是这样做通信开销显然比较大，仅仅对于一个查询，就需要发送单元的数据。

MC-JXT采用子令牌拆分的方法降低分发过程中的通信开销。注意到 和数组中所有项有共同的部分，即和，这两个值不随改变而改变，因此协议可以将,拆分成两个子令牌，分别是和：

协议让部分由生成，而部分只能由生成，并且将其在审查通过之后分发给，然后将两部分组合成完整的和，再进行后续交互查询。此时对于一个查询，分发通信开销只有两个单元，比之前大大减小。为了实现上面的方案，协议需要让和用不同的密钥组生成，而不是像JXT中都由生成。协议让仍由生成，而 用另一个密钥组 生成，这样分发过程就是本地用计算出，然后将连同发送给，后续用密钥组计算出。

（2）同态签名方案

一种令牌签名思路是让对搜索令牌直接进行数字签名，然后服务端验证签名。但是此种方案显然无法实现，因为按照上一节中的方案，令牌是由组合生成的而不是，其对是不可见的。本节设计了一个同态签名方案来解决上述的问题。虽然无法直接对令牌签名，但是 可以对子令牌进行签名，如果所设计的签名具有加法同态性，那么由已签名的子令牌和未签名的子令牌就可以计算出已签名搜索令牌，即：

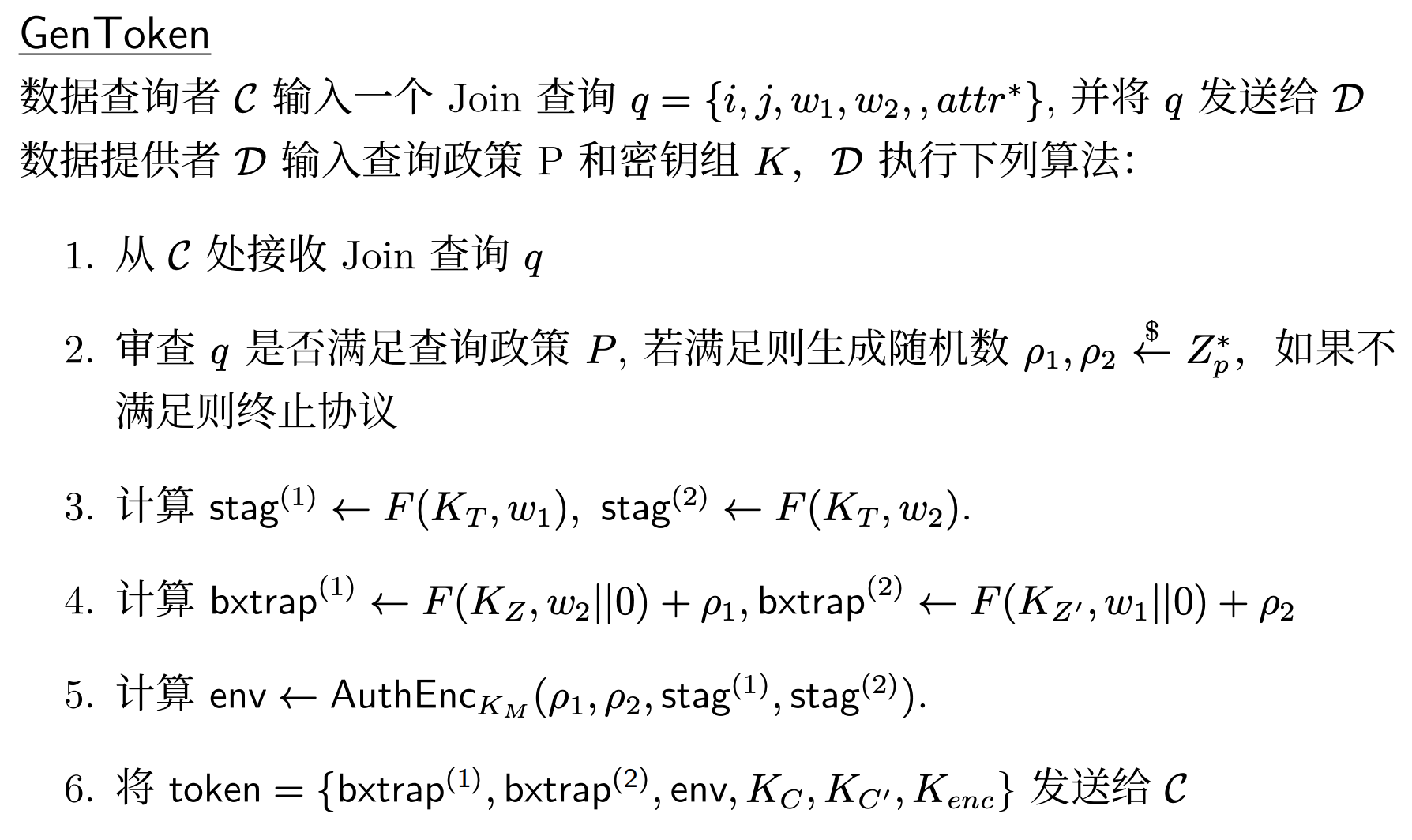
这样便可以实现在不破坏（1）中令牌分发算法的情况下，完成对令牌的签名。我们称中的两个元素分别为,，计算出之后，再随机选取一次性盲化因子，然后计算：

并将 连同发送给，然后利用本地计算出, 然后计算出：

, 。此时就等于在 的基础上加上一次性盲化因子，也就是已签名搜索令牌。之后将发送给。

要使得能够验证签名，需要把由传递给，将从接收到的 减去从接收到的，只有是由生成的，这一步才会产生正确有效的。此还需将查询所需要的 直接发送给，这两个值不能披露给。协议这里使用认证加密算法来进行上述值从到的传递。和共享密钥，然后使用认证加密算法 对上述值进行加密：

然后将发送给，将其连同发送给。



算法 2 MC-JXT-GenToken

### 算法

Search算法由和交互进行，由三个轮次组成，具体流程见算法 3。

轮次1：使用从接收的已签名子搜索令牌组装生成已签名搜索令牌，并发送给。

轮次2： 验证从接收到的已签名搜索令牌的合法性，并恢复出原始令牌，接着用原始令牌在密态数据库中检索匹配的数据，作为初步结果并返回给。

轮次3： 本地对从接收到的初步结果进行解密，获取到最终查询结果。

（1）轮次1:

首先接受发送过来的数据:

接着利用中的按照上一节的描述的方法生成并发送直到收到来发送的停止信号。此外还会将发送给。

（2）轮次2:

首先从中取出，利用查询出对应的中 对应的索引，其具体形式为，接着对查询中的Join属性集合中的每个Join属性计算：

遍历完中所有的元素之后，向发送。

类似地，从中取出，利用查询出对应的中 对应的索引，其具体形式为，接着对查询 中的Join属性集合中的每个Join属性计算：

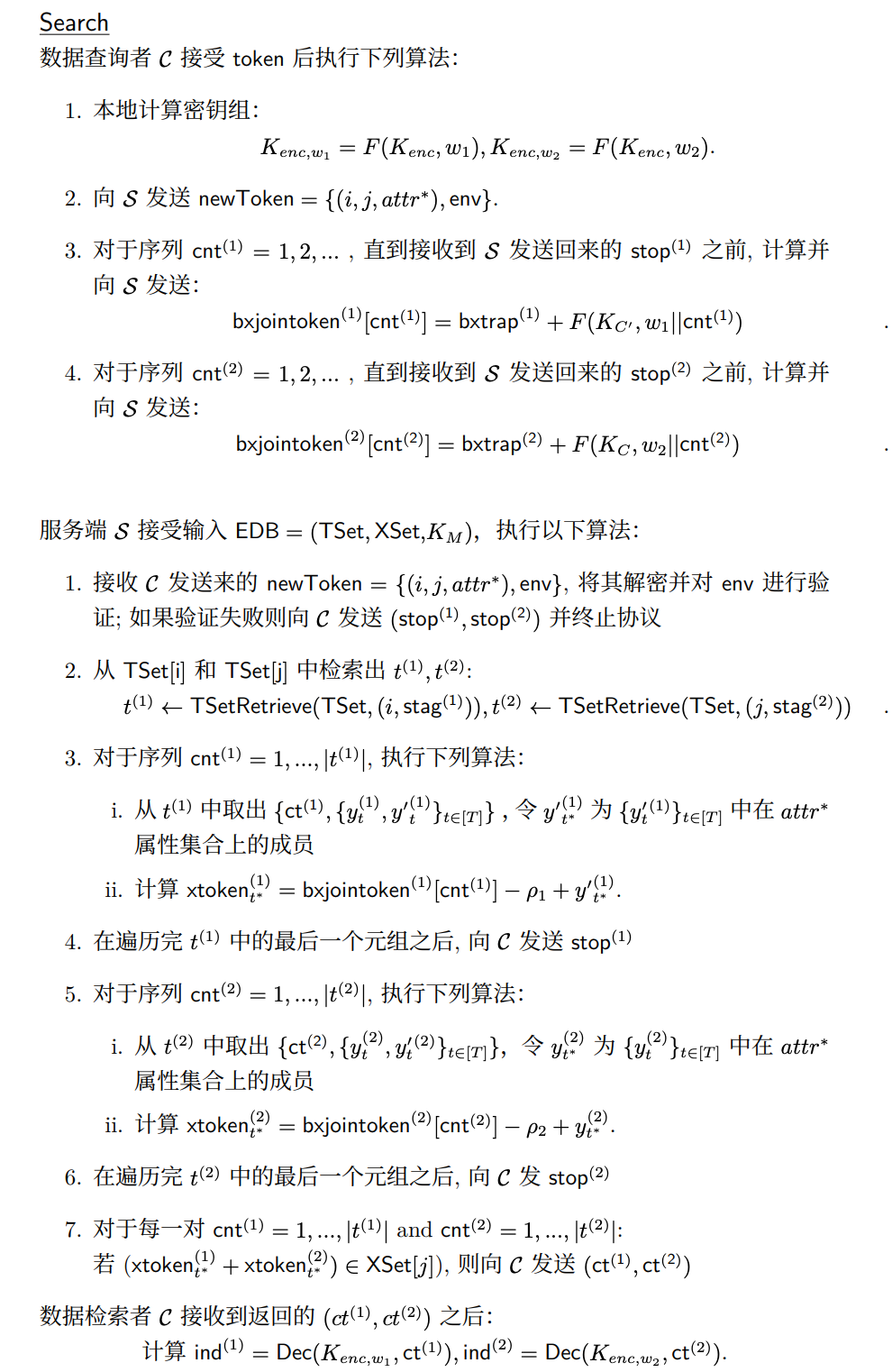
遍历完中所有的元素之后，向发送 。

然后针对每一组，计算

是否成立。如果所有的，上式均成立，则将给。（3）轮次3: 本地解密

从中取出密钥, 像在中那样，计算相应的：

然后对接收到的进行解密得到最终的查询结果：



算法 3 MC-JXT-Search

## MC-JXT协议正确性分析

本节对MC-JXT的正确性进行形式化分析。MC-JXT协议正确性的定义指，发起任意查询如果其运行完MC-JXT协议之后其得到的查询结果同 一致，那么此协议就是正确的。本节下面就会对此MC-JXT正确性定义进行证明，证明脉络是证明对于任意查询, 中的任意元素都包含于MC-JXT实例最终的查询结果当中，相应的，不属于的元素一定不会出现在MC-JXT实例最终的查询结果中。

考虑Join查询，假设存在一对记录，也即满足：

下面本节论证此协议的正确性，也就是论证处于 之后 所获得的查询结果之中。

由上一节可知，协议中首先获取中对应的项 ,又知对于每个 ：

而在轮次一中发送给的搜索令牌为：

其中，，所以在轮次二中对于每个，计算的值为:

类似地有：

所以最后计算的为：

由上面对的相关定义可知，相应的有对于每个，有，所以有

而由可知，，故可知成立，所以会将对应的 返回给， 最终获得的查询结果中会包含。

同理可以证明的情况下，运行完整个协议之后得到的查询结果一定不包括。由此本节便证明了运行完MC-JXT协议之后，得到的查询结果同是相同的，也就证明了协议的正确性。

## MC-JXT协议安全性分析

在一般的单用户对称可搜索加密方案中（包括JXT），攻击者只包括服务端 ，而多用户场景下还要额外考虑客户端作为攻击者的情况。MC-JXT在假设客户端和服务端之间不会勾结的前提下，考虑如下情况下的安全性：（1）诚实且好奇的服务端；（2）恶意的客户端。MC-JXT所有的安全定义都定义在由泄露函数 参数化的理想/现实范式框架[15]下。

### 攻击者为服务端 时的安全性

MC-JXT首先考虑攻击者为诚实且好奇的服务端的情况，这也是大部分对称可搜索加密方案安全定义中的攻击者配置，包括JXT。MC-JXT在此情况下的安全性定义同JXT安全性定义[12]类似，都满足抗自适应攻击的-语义安全，此处本文不再赘述。

### 攻击者为客户端 时的安全性

（1）安全性定义

首先介绍此攻击场景下的泄露函数。由3.2中的协议算法可知，协议中服务端会泄露给客户端满足和的记录数目，即和。泄露的原因是过程中，服务端分别在遍历完和之后会向客户端发送停止信号，因此客户端可以由自己发送的数目来确定上述两个值。在实际的应用中，我们通常会使用增加冗余的方式来对上述泄露信息做掩码。我们设置一对掩码值，。然后我们在 阶段将和建立时使用的计数值的范围扩展为和，这样客户端在查询过程中就要生成掩码值数目的，原先的值就被掩盖起来了。故泄露函数定义为：

此场景下的安全性定义针对的是在协议运行过程中，除了查询结果 之外，客户端获取到的其他额外信息。遵循Lindell 等人[16]对恶意的攻击者的定义，MC-JXT安全定义中的理想世界中存在可信第三方运行一个理想算法并同模拟者进行交互。此算法是对MC-JXT协议功能的一种理想模拟，其接受查询作为参数，然后验证是否满足政策，如果满足，则该算法返回，如果不满足，则返回空值。MC-JXT针对恶意的客户端的完整安全性定义如下：

**定义1**：令 为MC-JXT协议的一个实现。给定泄露函数 ，以及模拟者，定义如下两个实验：

(1) : 选择，接着实验运行。攻击者可以自由地调用和实例，在前者中接受和并与数据提供者进行交互；在后者中与接受的服务端进行交互。在这些交互过程中可以任意表现，最终输出一个比特，且就是实验最终的输出。

(2) :同上述实验选择，实验首先运行以初始化模拟者算法。当调用时，它将于运行着的实验进行交互；当调用时，将于运行着的实验进行交互。此处和在与进行交互时，都可以更新全局模拟者状态变量。两者也都可以向理想算法发起查询并得到上述的查询结果。在这些交互过程中可以任意表现，最终输出一个比特，且就是实验最终的输出。

如果对于攻击者所能进行的任何多项式时间算法，总是存在一个运行算法 的模拟者，使得:

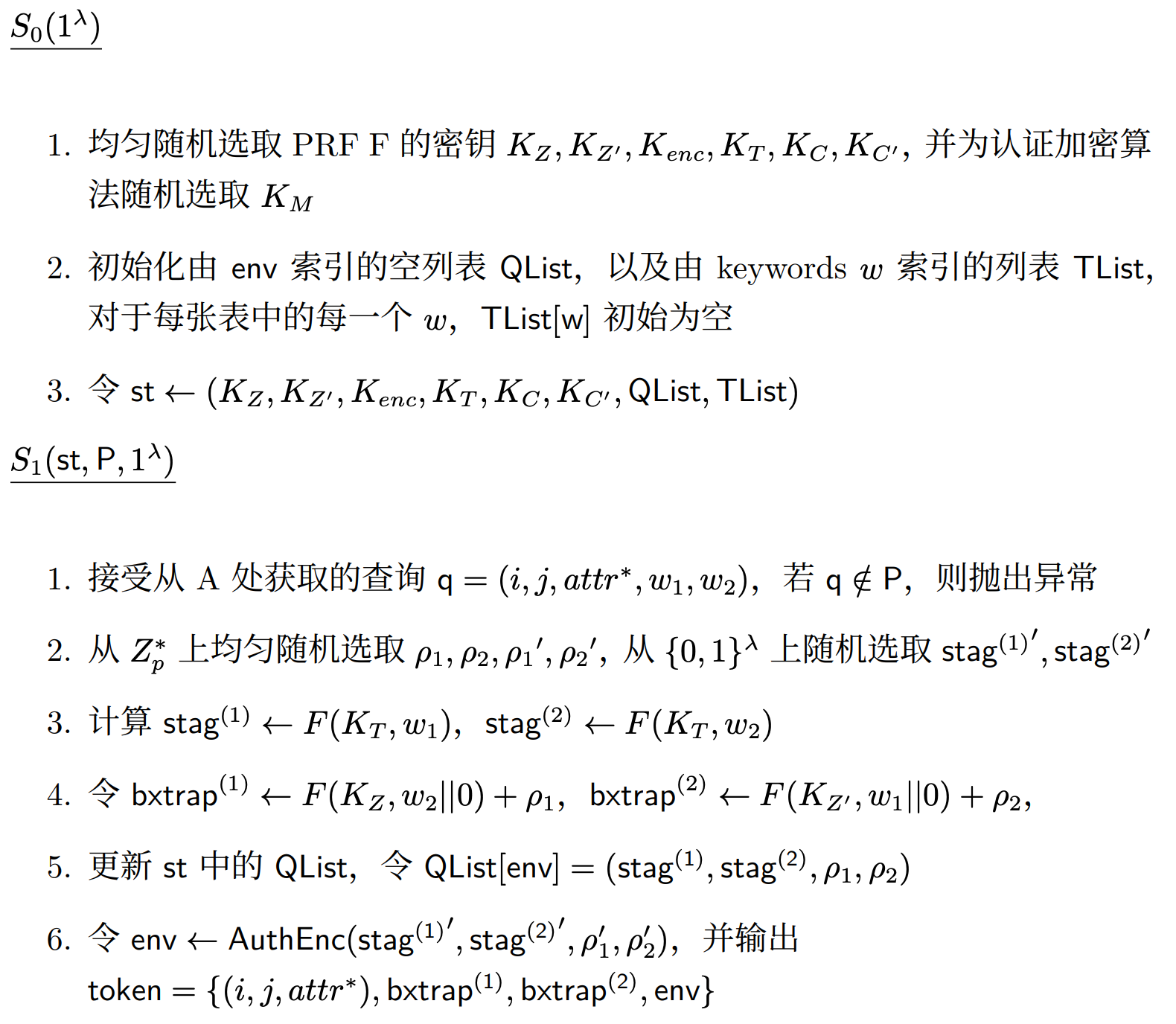
即使得两个实验输出结果的统计学差异为安全参数的可忽略函数，我们就称协议满足抗恶意客户端攻击的-语义安全。

（2）MC-JXT实现的安全性及其证明

以上是攻击者为恶意客户端的情况下MC-JXT协议的安全性定义，对于一个实际的MC-JXT实现而言，当满足下列条件时，我们称此实现在攻击者为恶意服务端的情况下是安全的：

**定理1**：令泄露函数为，当满足以下条件时，一个MC-JXT实现满足抗恶意客户端攻击的 -语义安全:（1）是一个安全的伪随机函数。（2） 满足计算正确性。（3）认证加解密算法（）满足IND-CPA以及Strongly-UF-CMA定义。

下面是定理1的证明。此证明的模拟者算法 如算法 4，算法 5所示。我们构造游戏来指代定义一中的实验，在下列的安全性证明当中，我们一步步地对做修改和放缩，使得最终获得的与定义一中的 实验相同。我们设为是游戏输出1的概率。如果我们能证明和的差异在安全参数下是可忽略不计的，也就证明了定理一，具体证明如下所示。



算法 4 模拟者算法S（a）

Game ：相比于，去除了以下不安全的情况：在协议中，存在没有由在中生成并签名的密文（即伪造的）通过服务端认证。若认证加密算法满足Strong-UF-CMA不可伪造定义，则有。

Game ：相比去除了以下不安全的情况：存在两个不同的实例产生相同的密文。由于协议中是上随机生成的，因此对应的明文发生碰撞的概率可忽略不计；同时由Strong-UF-CMA定义可知，此认证加密算法出现不同明文对应密文相同的概率也是可忽略的，因此有。

Game ：去除了以下不安全的情况：存在两个不同的关键字，，满足或者。此种情况代表着协议中或者发生了碰撞，由于是一个伪随机函数，由其定义可知碰撞发生的概率可忽略不计，即有。

Game ：去除了。在中，我们不再调用来通过明文索引建立，而是仅仅保留; 在中用查询 时，我们也不用来得到结果，而是直接令。只要保证的正确性，对于一个合法的，我们使用上述方式得到的和使用得到的一定是相等的，因此有。

Game ：在中计算是加密 ，将其替换为加密一组独立随机值，其中从上均匀随机选取，从上随均匀随机选取。为了保证协议功能性，维护一个由密文作为索引的数组，在调用时，真正的被存储在中。相对应的，在中，模拟者不再解密以获得，而是直接查询。如果认证加密算法满足IND-CCA定义，则有。

Game ：将PRF 替换为上的随机函数。由PRF的定义，有。

Game ：去除了，修改后的过程如下：

（1）服务端接收客户端发送的, 并从中取出。

（2）对于和，检验是否满足以下条件：

（3）继续检验是否满足：。这里对应于 的第个元组，同理。若仍然满足此条件，则服务端返回相应的，否则停止协议。

对于任意的，中的查询结果一定也在的查询结果中。因为如果满足，那么保证了和 是合法的，按照协议后续步骤一定可以计算出；而则进一步说明了。因此，和 之间唯一的区别就是有以下特例发生：不满足，即是不合法的，但是由计算出的，即在不满足公式（3.21）的情况下有：

令，若不满足 ，则有，其中

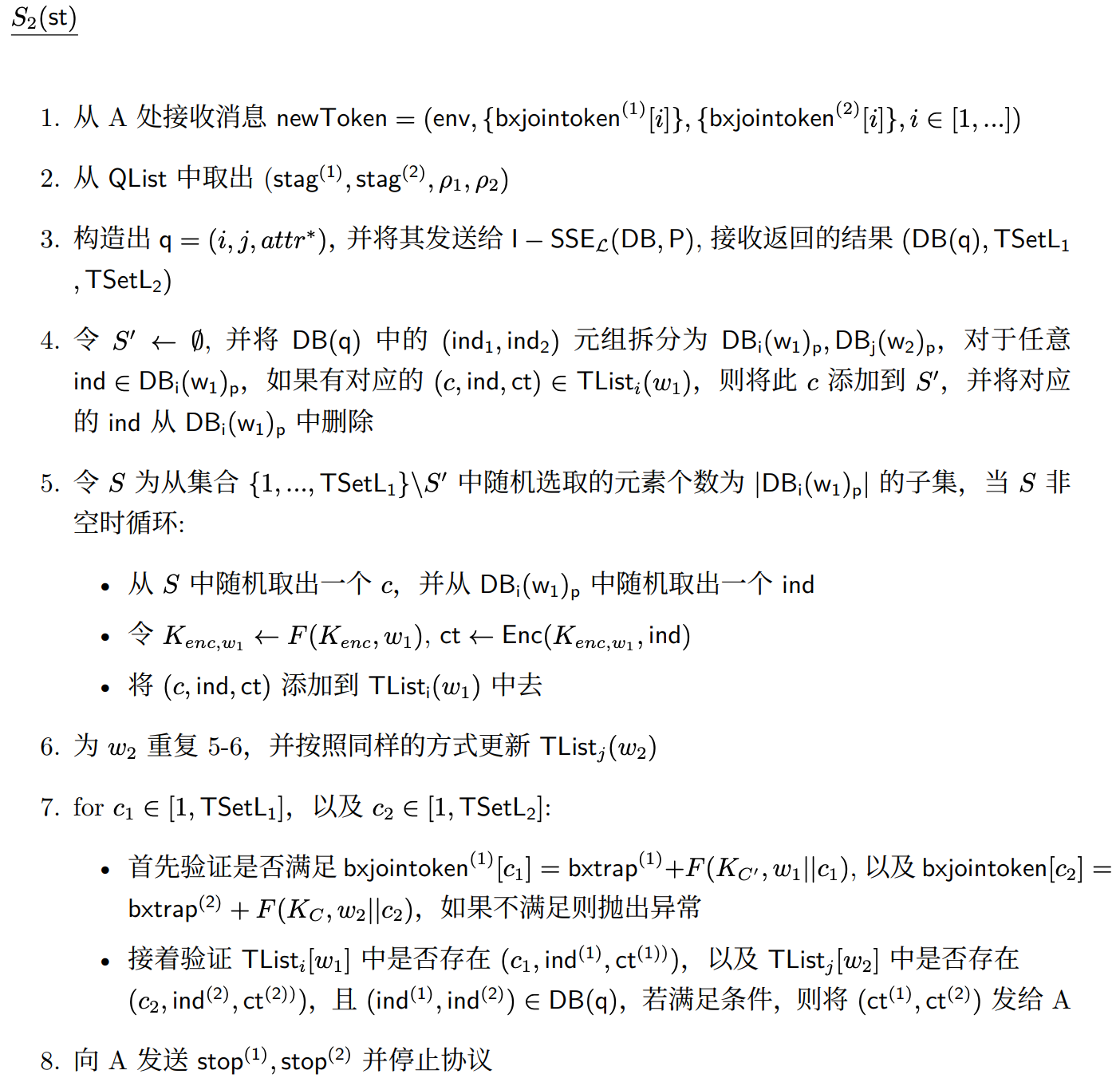
而可以说明，使得，即为：

由 是PRF可知，上式成立的概率可以忽略不计，故有。

Game ：考虑了复用的安全性问题，去除了以下不安全的情况: 对于同一个，存在两对元组, 有，使得如下等式成立：

对于同理。此种情况发生于攻击者将上一次实例的保存下来，在这一次实例中重复使用。我们只考虑 ，此问题就等价于恶意的客户端在有效的情况下，猜测的值等于的成功概率问题。由于是PRF，并且是均匀随机选取的，故此处猜测成功的概率可忽略不计，即有。

Game ：在上述游戏中，模拟者是知晓攻击者选取的Join查询的，因为协议中客户端会将直接发给模拟者进行审查。所以我们在中，可直接将 发送给理想算法，并得到正确的查询结果，游戏具体流程如下: （1）在过程中抛弃了和，仅仅选取密钥。（2）过程同相同。（3）在中，将发送给 并得到。接着将中的 元组进行拆分，拆分为和，分别含有所有元组的第一维和第二维。然后赋给中元素1到的非重复随机索引标号，对也同样这样做。为了跟踪此索引标号，维护列表 ，其作用是模拟原先游戏中的。以为例，中存储着元组集合 ，是上述过程给此的索引标号；是对应的密文。在协议的开始，将所有的初始为空，每次在调用完 之后，将中的不在的按上述方式赋索引值，并同相应 加入 。后续如。



算法 5 模拟者算法S（b）

由此，本节构造了，两个游戏，使得与定义一种的相同，与定义一中的实验相同。并且本节还一步步证明了在满足定理一所要求条件的情况下，和的差异在安全参数下是可忽略不计的（即），故定理一得证。

## MC-JXT协议理论开销分析

本节对MC-JXT的性能开销进行理论分析，后续第四章我们将对协议进行代码实现并对理论分析的结论进行实验验证。我们将从三个方面分析MC-JXT协议的性能，分别是存储开销，计算开销，和通信开销。为了便于分析，我们假设进行Join查询的两个表是大小相同的，这里的大小相同指两表有相同的记录数和属性数。我们设两表的记录数为，属性数为，两表Join属性的数目为。

### 存储开销

本节我们讨论MC-JXT中每张表对应的密态数据所造成的存储开销，此开销主要分为两部分，分别是和。由3.2.1可知，对于表中所有的关键词 ，中都有一个对应的列表，列表中元素的形式为。显然每一个元素占用个存储单位，分别对应于和。这样以来 占用的空间为个单位，而这张表整体占用的空间为单位。又由，故。

接下来我们再考虑的存储开销。由3.1.1可知，算法会对表中每条记录的每个Join属性计算一个值，将其存储在中，故知表的 由个构成，每个由两个伪随机函数值相加得到，占用一个存储单元，所以。

综上所述，MC-JXT中一张表的存储开销为：，而如果按照传统方案那样，预先计算两表之间的Join表，那么则存储一张表的存储开销至少为。一般而言，在数据库规模比较大的场景下，的值远大于的值，显然MC-JXT有着明显的存储优势。

### 计算开销

的计算开销：数据提供者首先运行算法建立密态数据库，首先考虑和的建立计算开销。正如3.4.1所提到的那样，的建立需要计算个，每个需要若干次计算，故其建立计算开销为；建立的计算开销也就是生成中所有单元的计算开销，由3.4.1可知，其为。此外，还需要承担的计算任务有：生成、和，这些运算的计算复杂度都是所以整体计算开销为。

的计算开销：客户端在协议中根据发送而来的和密钥，计算出搜索令牌和。根据3.1可知，此过程需要计算的令牌数量为|，所以其计算开销为。

的计算开销：服务端在协议中的任务分成两部分，分别是恢复出和检索密态数据库。在第一部分中服务端首先将进行解密，得到 ，计算复杂度为。接着 将收到的和分别减去以恢复出，此过程的计算复杂度为 。在第二部分中，服务端首先使用检索出和，接着其会遍历这两个列表笛卡尔积的每一个元组对，并针对每一个Join属性进行匹配运算，计算复杂度为。

### 通信开销

由上面描述的协议流程，和之间的通信开销显然是的，而和 之间没有通信。对于和而言，从接受查询结果，大小为 个单位，而从获得的数据有个单位。总而言之，协议整体的通信复杂度为。

# 协议实现以及性能测试

本章首先会介绍MC-JXT的代码实现方案。然后会介绍实验用到的数据集以及相应的性能测试实验结果（性能测试的评价指标是运行时间），并讨论实验结果同3.5中理论分析的契合度以及MC-JXT协议的性能优势。

## 协议实现

我们使用了Java语言实现了MC-JXT协议，该实现由两部分组成，分别是数据预处理阶段和数据检索阶段。数据预处理阶段将读取数据集，并在其基础上执行获得密态数据库；查询执行阶段主要包括生成所需的查询令牌以及查询的执行。这两阶段程序上的密码学原语都是依赖包JPBC-2.0.0（Java Pairing-Based Cryptography Library）实现的。具体而言，伪随机函数的具体实现为:，为JPBC库中对称质数双线性群TypeA（建立在圆锥曲线 之上）；加密算法实现为AES的ECB模式；认证加密算法AuthEnc实现采用Encrypt-then-MAC算法，其中加密算法和MAC算法分别选取AES-ECB算法和HMAC-SHA256算法。

数据预处理阶段生成和结构，并存储在内存里。具体的实现是一个由作为键的哈希表，每个键对应的值是一个固定大小的元组列表。此处我们使用布隆过滤器[17]实现，布隆过滤器的插入和查询时间均为常数，此种实现方式可以大大提升判断是否在中的判定效率。然而布隆过滤器会造成“误判”（假阳性）的可能，我们此处将布隆过滤器的误判率设置为 ，以达到准确性同占用存储空间之间的平衡。

## 性能测试

### 数据集

本文采用人造的美国人口普查数据[18]（US-cencus data）作为测试数据集。美国人口普查局提供有关国家人口的数据，此数据公开在相关网站上。该数据集只有人口数据的统计信息，没有个人的具体信息，因此本文按照公开的US-cencus data中统计数据的比例，构造了一个大小为1000000条记录的美国公民个人信息数据库，包含（personID, firstname, lastname, sex, age, state, race）七个属性，数据库具体包含以下两个表：

（1）state\_census(personID, firstname, lastname, sex, age, state)

（2）race\_census(personID, firstname, lastname, sex, age, race)

表state\_census 和表race\_census分别存储着相同集合公民的住址信息（来自哪个州，包含50个州）以及种族信息（是什么种族，包含白人，拉丁裔，亚裔等七个种族），两表中公民的顺序是被随机打乱的，并且两表都有两个主键，分别是(personID)和(firstname,lastname)。我们执行以下J查询来测试协议的性能表现：

### 实验设计以及实验结果

本文实验环境信息如下:

表 4‑1实验环境介绍

|  |  |
| --- | --- |
| 处理器 | Apple M1 8 cores |
| 内存 | LPDDR4 16 GB |
| 操作系统 | macOS Ventura 13.1 |
| 硬盘 | APPLE SSD AP0256Q |

实验中会根据需求随机从上述构造的仿真的美国人口普查数据集中获取记录并构建实验数据集，然后对每一组选取的实验数据集建立密态数据库并执行相应的查询，整个实验阶段使用到的记录数从200到30000不等。本文实验中的性能评价指标是协议的运行时间，以下是具体的实验结果:

（1）MC-JXT在实际应用场景中的运行时间

本实验测量在不同查询规模下协议整体的运行时间。由于Join运算的特点（即两表的记录集合做笛卡尔积），此处的查询规模指满足查询中 和 的记录数量之积，即为 。本实验主要反映了MC-JXT协议在实际应用场景中的性能表现，本实验所选取的查询规模从19840到7279584不等，实验结果如图 4‑1所示。

由图 4‑1可知，MC-JXT协议运行时间在查询规模较大时与查询规模呈现线性关系，具体地，在查询规模为19840条记录时，协议运行时间仅为0.20秒；在查询规模为455281条记录时，协议运行时间仅为1.093秒；在查询规模为4167792条记录时，协议运行时间仅为5.389秒。

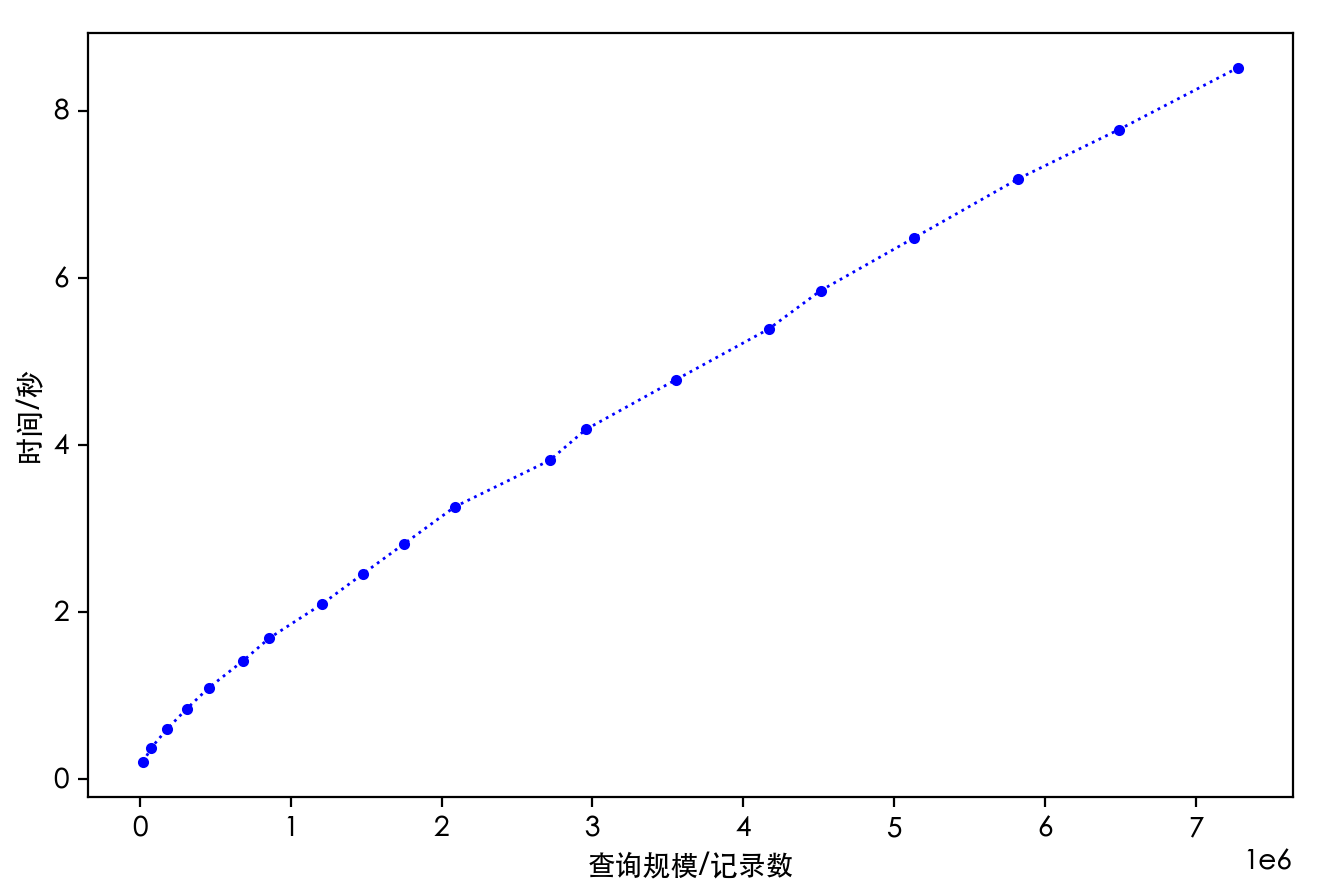


图 4‑1 MC-JXT运行时间随查询规模的变化

（2）阶段性能的影响因素

本段实验主要探究性能的影响因素，由两个子实验组成。

1. 运行时间与数据库规模之间的关系



图 4‑2 EDBSetup运行时间与数据库规模的关系

此处我们改变数据库规模大小的方式是控制数据库中的记录数目，不改变数据库中表的属性，故数据库规模就是指表中的记录数目，图 4‑2为实验结果。

实验结果显示 运行时间与数据库规模成正比，与理论分析相符。

2. 运行时间与表之间Join属性数目的关系

我们在4.2.1中提到，实验的两表的主键有两组，分别为（personID）和（firstname, lastname），所以实验构造三种不同的Join属性集合，如表 4‑2所示。

表 4‑2 T1,T2,T3取值

|  |  |
| --- | --- |
| T1 | （personID） |
| T2 | （firstname, lastname） |
| T3 | （personID, firstname, lastname） |

实验测量三种场景各自的性能表现，实验结果如图 4‑3所示。



图 4‑3 EDBSetup运行时间与Join属性数目的关系

由图 4‑3可知，随着的增大，运行时间也在增大。为了进一步探究具体的运行时间的增大和T变化之间的关系，我们计算每一个数据库规模下，T=3比T=2增加的时延同T=2比T=1增加的时延的关系，结果如实验结果如图 4‑4所示。



图 4‑4 不同T下时延增量的比例关系

由图 4‑4 不同下时延增量的比例关系，=3相比 =2增加的时延与=2相比=1增加的时延基本相等，故运行时间同成线性关系。

（3）阶段性能的影响因素

本实验首先探究运行时间同查询规模的关系，结果如图 4‑5。

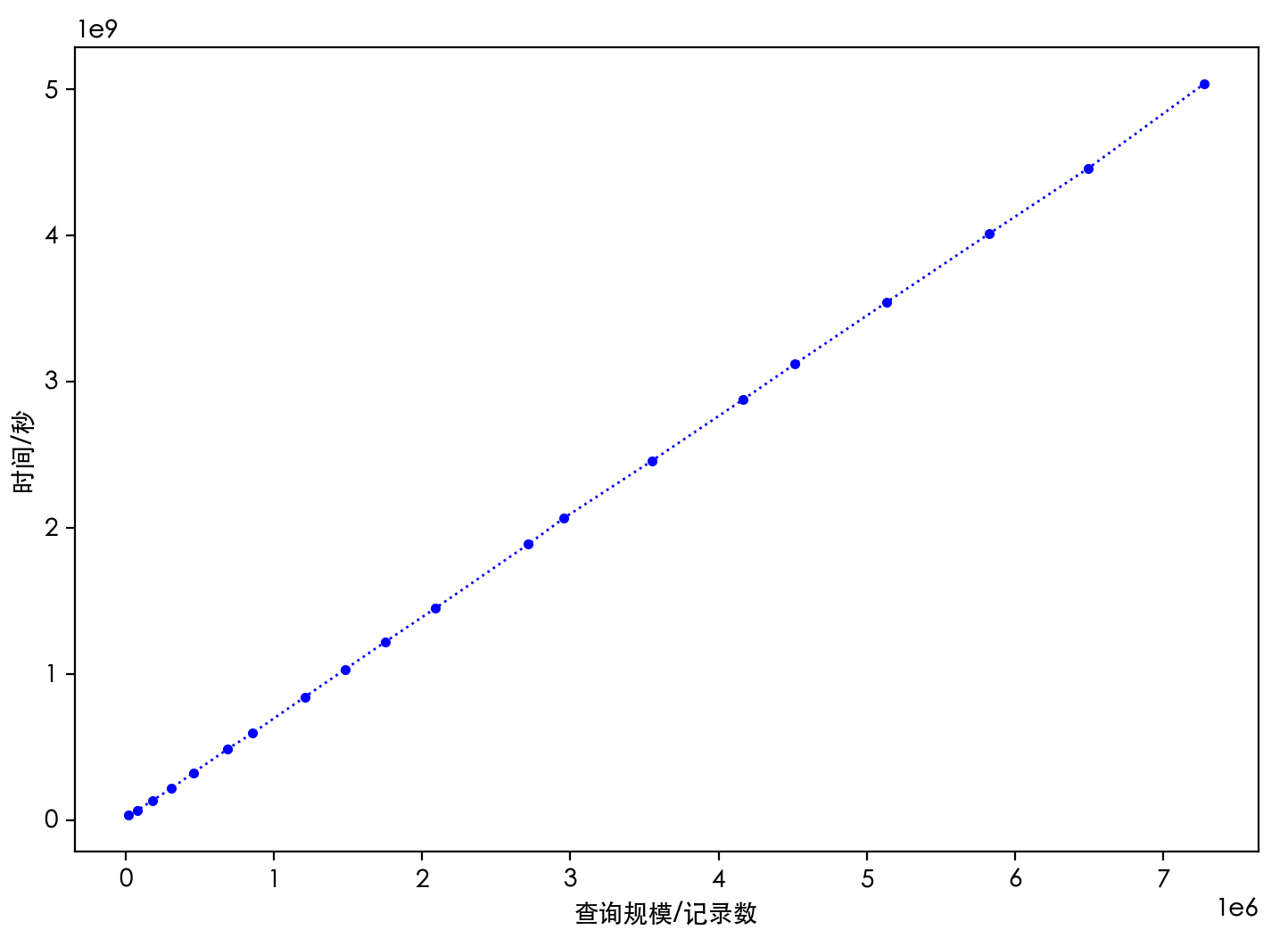


图 4‑5 Search运行时间与查询规模的关系

实验结果显示运行时间同查询规模成正比，这一点与理论分析结论相符。此外本文还设计实验探究了MC-JXT阶段运行时间同数据库规模的关系，我们控制查询规模不变，让数据库规模不断增大（这一点的实现方式是在表中插入特定数量的，和执行的查询无关的冗余记录），然后测量不同数据库规模下的运行时间，实验结果如图 4‑6所示。



图 4‑6 固定查询规模的下Search运行时间与数据库规模的关系

实验结果显示MC-JXT协议阶段的运行时间同数据库规模之间没有直接联系，只和查询规模相关。

### 实验结论

上述实验结果同3.5理论分析结果相契合。实验结果显示协议整体运行时间在查询规模较大时与查询规模呈现线性关系，在大规模数据的场景下依然可以达到较优的性能。同时，协议的数据检索阶段运行时间同数据库规模之间没有直接联系，只与查询规模相关，这说明协议服务的检索者用户的查询时延不会因为数据库规模的增大而产生直接的影响，进一步显示出MC-JXT在大规模数据下的应用价值。此外实验表明，协议阶段运行时间同数据库规模和表之间Join属性数目成正比，协议阶段运行时间同查询规模成正比。

# 总结与展望

## 本文总结

在隐私泄露事件频发，数据安全挑战日益严峻的背景下，如何提升对称可搜索加密技术的实用价值对进一步保护云端用户隐私数据的安全有着重要的意义。目前学术界支持复杂查询的多用户对称可搜索加密研究仍然比较匮乏，给对称可搜索加密在数据共享场景中的实际应用造成了阻碍。

为了进一步提高对称可搜索加密在数据共享场景下的价值，本文对现有方案进行深入研究，设计出了多用户场景下高效支持Join查询的对称可搜索加密方案MC-JXT。协议在支持高效Join查询的同时，采用子令牌拆分方法和同态签名方案实现了搜索令牌的高效分发和合法性验证，很好地实现了多用户间机密数据共享的功能性。协议安全性分析显示MC-JXT满足恶意客户端、诚实且好奇服务端条件下的基于模拟的安全性。

此外，本文对MC-JXT进行了性能开销的理论分析，并设计相关实验测试了协议在实际应用场景的性能表现以及性能主要制约因素。实验结果同理论结果相契合，显示协议整体运行时间在查询规模较大时与查询规模呈现线性关系，在大规模数据的场景下依然可以达到较优的性能。同时实验还显示协议的数据检索阶段运行时间同数据库规模之间没有直接联系，只与查询规模相关，这说明协议服务的检索者用户的查询时延不会因为数据库规模的增大而产生直接的影响，进一步显示出MC-JXT在大规模数据下的应用价值。

## 工作展望

本文在研究过程中发现了一些未来值得研究的新问题，总结如下面几点。

首先是如何扩展MC-JXT协议，使其支持涉及三个及以上表的Join查询。MC-JXT主要考虑两表之间的Join查询，如果能使其支持涉及更多表数目的Join查询，那么协议在大规模复杂关系型数据库中的应用价值就得到进一步的提高。

其次是如何将MC-JXT协议扩展为支持动态增减的多用户对称可搜索加密方案。目前MC-JXT协议建立的密态数据库是在 创立之初确定的，后续不支持对密态数据库添加或者删除记录。

最后是如何改进MC-JXT中的同态签名方案，使得协议开销更小。MC-JXT中数据提供者 要为每一个Join查询都生成一组盲化因子，并将其装入数字信封中，并发送给服务端 。在执行大量的Join查询时，生成和传递盲化因子造成的计算与通信开销不容忽视。如何改进此同态签名方案，进一步降低其性能开销值得进一步研究。

# 参考文献

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

[2] DAWN XIAODING S, WAGNER D, PERRIG A. Practical techniques for searches on encrypted data; proceedings of the Proceeding 2000 IEEE Symposium on Security and Privacy S&P 2000, F 14-17 May 2000, 2000 [C].

[3] GOLDREICH O, OSTROVSKY R. Software protection and simulation on oblivious RAMs [J]. J ACM, 1996, 43(3): 431–73.

[4] GOH E-J. Secure Indexes [J]. 2003.

[5] CURTMOLA R, GARAY J, KAMARA S, et al. Searchable symmetric encryption: improved definitions and efficient constructions [Z]. Proceedings of the 13th ACM conference on Computer and communications security. Alexandria, Virginia, USA; Association for Computing Machinery. 2006: 79–88.10.1145/1180405.1180417

[6] GOLLE P, STADDON J, WATERS B. Secure Conjunctive Keyword Search over Encrypted Data; proceedings of the Applied Cryptography and Network Security, Berlin, Heidelberg, F 2004//, 2004 [C]. Springer Berlin Heidelberg.

[7] BALLARD L, KAMARA S, MONROSE F. Achieving Efficient Conjunctive Keyword Searches over Encrypted Data; proceedings of the Information and Communications Security, Berlin, Heidelberg, F 2005//, 2005 [C]. Springer Berlin Heidelberg.

[8] WANG P, WANG H, PIEPRZYK J. Keyword Field-Free Conjunctive Keyword Searches on Encrypted Data and Extension for Dynamic Groups; proceedings of the Cryptology and Network Security, Berlin, Heidelberg, F 2008//, 2008 [C]. Springer Berlin Heidelberg.

[9] CASH D, JARECKI S, JUTLA C, et al. Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries; proceedings of the Advances in Cryptology – CRYPTO 2013, Berlin, Heidelberg, F 2013//, 2013 [C]. Springer Berlin Heidelberg.

[10] KAMARA S, MOATAZ T. SQL on Structurally-Encrypted Databases [Z]. Advances in Cryptology – ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part I. Brisbane, QLD, Australia; Springer-Verlag. 2018: 149–80.10.1007/978-3-030-03326-2\_6

[11] CASH D, NG R, RIVKIN A. Improved Structured Encryption for SQL Databases via Hybrid Indexing; proceedings of the Applied Cryptography and Network Security, Cham, F 2021//, 2021 [C]. Springer International Publishing.

[12] JUTLA C, PATRANABIS S. Efficient Searchable Symmetric Encryption for Join Queries [Z]. Advances in Cryptology – ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part III. Taipei, Taiwan; Springer-Verlag. 2023: 304–33.10.1007/978-3-031-22969-5\_11

[13] RAYKOVA M, VO B, BELLOVIN S M, et al. Secure anonymous database search [Z]. Proceedings of the 2009 ACM workshop on Cloud computing security. Chicago, Illinois, USA; Association for Computing Machinery. 2009: 115–26.10.1145/1655008.1655025

[14] JARECKI S, JUTLA C, KRAWCZYK H, et al. Outsourced symmetric private information retrieval [Z]. Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. Berlin, Germany; Association for Computing Machinery. 2013: 875–88.10.1145/2508859.2516730

[15] CHASE M, KAMARA S. Structured Encryption and Controlled Disclosure; proceedings of the Advances in Cryptology - ASIACRYPT 2010, Berlin, Heidelberg, F 2010//, 2010 [C]. Springer Berlin Heidelberg.

[16] LINDELL Y. How to Simulate It – A Tutorial on the Simulation Proof Technique [M]//LINDELL Y. Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich. Cham; Springer International Publishing. 2017: 277-346.

[17] BLOOM B H. Space/time trade-offs in hash coding with allowable errors [J]. Commun ACM, 1970, 13(7): 422–6.

[18] United States Census [Z].

# 致谢

首先，我要感谢我的指导老师欧长海老师和孙士锋老师。在此次毕业论文的撰写过程中，两位师给予了我很多的指导和帮助。他们不仅在论文选题、研究方法、实验设计等方面给予了我很多宝贵的意见和建议，还在论文写作过程中对我的语言表达、逻辑思维等方面进行了耐心细致的指导。正是由于他们的支持和鼓励，我才能够顺利地完成这篇毕业论文。

其次，我要感谢本科阶段在武汉大学的授课老师们。在武汉大学国家网络安全学院的学习生活中，我得到了很多其他老师和同学的关心和支持。他们不仅在学习上给予了我很多启发和帮助，还在生活上给予了我很多关心和照顾。正是由于他们的陪伴和支持，我才能够度过这段充实而难忘的大学时光。

最后，我要感谢我的家人和朋友们。他们在我学习期间一直支持着我，给予了我很多的鼓励和帮助，让我能够更加自信地面对未来。

衷心感谢武汉大学，给我提供了这么好的学习、生活设施，让我遇到了那么多优秀而善良的同学、老师，希望以后能够见到更多像你们这样优秀的人。