说明书摘要

本发明公开了一种。本发明通过。

摘要附图

权利要求书

1. 一种客户端执行的可验证的加密搜索方法，其特征在于：包括以下步骤：U1，客户端对文件集合计算倒排索引，对倒排索引中的每一个关键字计算一组键值对，其中，所述键为关键字对应的令牌，所述值为包含该关键字的文件的增量哈希和；将各组键值对插入MPT树结构中，得到初始化的MPT树；计算并存储各组键值对构建的MPT树结构的根哈希；U2，客户端上传所述初始化的MPT树至服务器，并在文件集合有更新时，上传更新的键值对至服务器；U3，客户端向服务器发送挑战令牌，接收服务器返回的搜索结果和验证结果，所述验证结果为服务器根据挑战令牌对应的MPT树结构中的叶子节点存在与否提取的MPT树结构中节点的键和键值对；U4，客户端将发送的挑战令牌和所述验证结果进行匹配，根据匹配结果重构出MPT树结构的根哈希；U5，将重构出的根哈希与客户端存储的根哈希进行比对，如果相同，表明服务器无篡改行为，搜索结果准确；如果不相同，表明服务器有篡改行为，搜索结果不准确。
2. 根据权利要求1所述的客户端执行的可验证的加密搜索方法，其特征在于：步骤U1中，通过对关键字进行加密处理得到关键字对应的令牌；通过伪随机函数实现加密处理过程，生成所述关键字对应的令牌。
3. 根据权利要求1所述的客户端执行的可验证的加密搜索方法，其特征在于：步骤U2中，当文件集合中有新增或者删除文件f时，对该文件f，得到该文件包含的关键字集合，对每一个属于的关键字，生成其对应的令牌，将令牌和文件f对应的增量哈希和作为更新的键值对上传给服务器。
4. 根据权利要求1所述的客户端执行的可验证的加密搜索方法，其特征在于：步骤U3中，所述验证结果按照如下方法提取得到：当所述挑战令牌对应的MPT树结构中的叶子节点存在时，服务器从叶子节点的上一层节点开始，提取位于搜索路径上的节点的键作为所述验证结果；对于搜索路径上的分支节点，还提取其不在搜索路径上的键值对一起作为所述验证结果；当所述挑战令牌对应的MPT树结构中的叶子节点不存在时，服务器从搜索路径上终结的叶子节点开始，提取终结的叶子节点处的键值对以及搜索路径上的节点的键作为所述验证结果；对于搜索路径上的分支节点，还提取其不在搜索路径上的键值对一起作为所述验证结果。
5. 根据权利要求1或4所述的客户端执行的可验证的加密搜索方法，其特征在于：步骤U4中，如果验证结果中的键是所述挑战令牌的前缀，则设置为所述挑战令牌中剩余的键，根据所述和所述搜索结果重新生成MPT树结构中的叶子节点处的键值对，完善验证结果，根据完善后的验证结果重构MPT树结构的根哈希；如果验证结果中的键无法与所述挑战令牌匹配，则设置为空集，直接根据所述验证结果重构MPT树结构的根哈希。
6. 一种服务器执行的可验证的加密搜索方法，其特征在于：S1，服务器接收客户端上传的初始化的MPT树，并接收更新的键值对，根据所述更新的键值对更新初始的MPT树结构，得到最新的MPT树结构；S2，服务器根据客户端发送的挑战令牌，进行搜索，得到搜索结果；S3，服务器根据客户端发送的挑战令牌，判断所述挑战令牌对应的MPT树结构中的叶子节点存在与否，根据判断结果提取的MPT树结构中节点的键和键值对作为验证结果；S4，服务器将所述搜索结果和验证结果发送给客户端。
7. 根据权利要求6所述的服务器执行的可验证的加密搜索方法，其特征在于：步骤S3中，根据如下步骤提取得到验证结果：当所述挑战令牌对应的MPT树结构中的叶子节点存在时，服务器从叶子节点的上一层节点开始，提取位于搜索路径上的节点的键作为所述验证结果；对于搜索路径上的分支节点，还提取其不在搜索路径上的键值对一起作为验证结果；当所述挑战令牌对应的MPT树结构中的叶子节点不存在时，服务器从搜索路径上终结的叶子节点开始，提取终结的叶子节点处的键值对以及搜索路径上的节点的键作为所述验证结果；对于搜索路径上的分支节点，还提取其不在搜索路径上的键值对一起作为所述验证结果。
8. 根据权利要求6所述的服务器执行的可验证的加密搜索方法，其特征在于：步骤S1中，对于文件集合中有新增文件的更新操作，服务器接收更新的键值对后，当所述键值对中的令牌对应的叶子节点已存在时，将所述键值对中的增量哈希和与该叶子节点上的值相加；当所述键值对中的令牌对应的叶子节点不存在时，创建一个新的叶子节点，将所述键值对中的增量哈希作为它的节点值。
9. 根据权利要求6所述的服务器执行的可验证的加密搜索方法，其特征在于：步骤S1中，对于文件集合中有删除文件的更新操作，服务器接收更新的键值对后，根据所述键值对中的令牌找到对应的叶子节点，将所述键值对中的增量哈希和与该叶子节点上的值相减。
10. 一种可验证的加密搜索方法，涉及客户端和服务器；其特征在于：A1，客户端对文件集合计算倒排索引，对倒排索引中的每一个关键字计算一组键值对，其中，所述键为关键字对应的令牌，所述值为包含该关键字的文件的增量哈希和；将各组键值对插入MPT树结构中，得到初始化的MPT树；计算并存储各组键值对构建的MPT树结构的根哈希；A2，客户端上传所述初始化的MPT树至服务器，并在文件集合有更新时，上传更新的键值对至服务器；A3，服务器接收客户端上传的初始化的MPT树，并接收更新的键值对，根据所述更新的键值对更新初始的MPT树结构，得到最新的MPT树结构；A3，客户端向服务器发送挑战令牌；服务器根据客户端发送的挑战令牌，进行搜索，得到搜索结果；服务器根据客户端发送的挑战令牌，判断所述挑战令牌对应的MPT树结构中的叶子节点存在与否，根据判断结果提取的MPT树结构中节点的键和键值对作为验证结果；服务器将所述搜索结果和验证结果发送给客户端；客户端接收服务器返回的搜索结果和验证结果；A4，客户端将发送的挑战令牌和所述验证结果中的键进行匹配，根据匹配结果重构出MPT树结构的根哈希；A5，将重构出的根哈希与客户端存储的根哈希进行比对，如果相同，表明服务器无篡改行为，搜索结果准确；如果不相同，表明服务器有篡改行为，搜索结果不准确。

说明书

一种可验证的加密搜索方法

**【 技术领域 】**

本发明涉及加密搜索领域，特别是涉及一种可验证的加密搜索方法。

**【 背景技术 】**

云存储使得用户可以随时随地地存取数据，并且极大地方便了用户之间的数据共享。但与此同时，云存储带来了许多安全性问题，总体来说可以分为以下两类：

（1）可用性（availability）。要求云服务器保证数据不丢失，用户可以将云端作为数据中枢进行数据备份和同步。目前，一般的云服务提供商都采用了多副本的方式保障数据的可用性，即将数据的多个副本分别写入其他的存储节点，当一个节点发生故障时，其他节点上的数据继续提供服务，同时通过其他节点中的数据副本，快速恢复故障节点上丢失的数据。目前，针对数据可用性的相关学术研究包括数据拥有证明（Proof of Data Possession, PDP）以及数据可恢复性证明（Proof of Retrievability, PoR）。

（2）隐私性（privacy）。要求云服务器保证数据的隐私并且不泄露数据。目前，云服务提供商一般采用数据加密方式对隐私数据进行保护，但数据加密往往会导致数据可用性的降低，例如数据失去可搜索性，因此加密搜索（Searchable Encryption）应运而生。

加密搜索技术主要分为两类，一是对称加密搜索（Searchable Symmetric Encryption, SSE），二是非对称加密搜索（Searchable Asymmetric Encryption，SAE）。由于非对称加密搜索的效率问题，目前的加密搜索领域对对称加密搜索关注较多。

对称加密搜索的模型如图1所示。用户自行对数据进行加密并上传到云端，与此同时，用户还需额外上传一个加密索引（index）使得云可以通过该索引来搜索数据。当用户需要搜索数据时，生成一个陷门（trapdoor），该陷门与关键字相关，使得用户可以在不暴露关键字内容的情况下进行内容搜索。

加密搜索使得用户在保护数据隐私的同时，满足了其搜索需求，但加密搜索并不能保证搜索结果的正确性。也就是说，加密搜索的前提是云服务器是诚实的，即服务器会遵守与用户的协议来正确的执行搜索操作，然而实际应用中，云服务器往往是不可信的，例如，云服务器有可能为了节省计算开销和通信开销而返回少量搜索结果给用户，甚至有可能不返回搜索结果给用户。为了防止云服务器的不诚信行为，学术界又提出了可验证的对称加密搜索机制（Verifiable Searchable Symmetric Encryption, VSSE）。可验证的加密搜索允许用户对搜索结果进行验证，来检测服务器的不诚信行为，保障了加密搜索的正确性。

在可验证加密搜索中，由于服务器不诚信导致的安全性攻击主要可以分为以下两种：

数据新鲜性攻击（Data Freshness Attack）：在加密搜索中，数据新鲜性攻击是指服务器（攻击者）试图返回旧的搜索结果，而不是最新的搜索结果。例如，用来表示旧版本的数据集，用来表示最新的数据集，则服务器返回的搜索结果是数据集的搜索结果，其中。

数据完整性攻击（Data Integrity Attack）：在加密搜索中，数据完整性攻击是指服务器（攻击者）试图不让用户获取完整的搜索结果。例如，用来表示加密搜索中用户的搜索陷门，用户应该得到的搜索结果为，而服务器返回的搜索结果为，其中并且可能为。



数据新鲜性攻击仅存在于动态的加密搜索方案中，在数据库静态的情况下不存在。但现实中，动态数据库较为常见，因此防范数据新鲜性攻击是可验证加密搜索必须要解决的问题。数据完整性攻击不仅包括服务器少返回搜索结果的情况，还包括了服务器不返回搜索结果来规避结果验证的情况。该问题是很严重的一个问题，但目前很少有研究考虑到该问题。

以上背景技术内容的公开仅用于辅助理解本发明的发明构思及技术方案，其并不必然属于本专利申请的现有技术，在没有明确的证据表明上述内容在本专利申请的申请日已经公开的情况下，上述背景技术不应当用于评价本申请的新颖性和创造性。

**【 发明内容 】**

本发明所要解决的技术问题是：弥补上述现有技术的不足，提出一种可验证的加密搜索方法，可同时验证服务器不诚信导致的新鲜性攻击和完整性攻击，适用范围较广泛，且开销小，。

本发明的技术问题通过以下的技术方案予以解决：

一种，

优选的技术方案中，

进一步优选的技术方案中，

本发明与现有技术对比的有益效果是：

本发明的可验证的加密搜索方法，利用新型的MPT树数据结构并设计合理的机制来完善对数据新鲜性和数据完整性攻击的防御，尤其是可防御服务器返回空结果规避结果验证的情况。通过构建MPT，将用于结果验证的索引与加密搜索的索引进行分离，使得验证过程可与任何加密搜索方案得到的搜索结果相结合，可在广泛范围内提供结果验证服务。通过挑战令牌与验证结果中的键进行匹配、由匹配结果重构出MPT树结构的根哈希，由根哈希与客户端存储的根哈希进行比对验证，本发明完善地解决了云存储中可能存在的数据完整性攻击和数据新鲜性攻击，尤其可以检测到服务器恶意返回空结果来规避结果验证的情况。本发明的方法，服务器与客户端进行一轮通讯即可完成验证，通讯轮次少，开销小。综上所述，本发明不仅是一种通用的可验证加密搜索框架，而且以较低成本完善地解决了加密搜索中的结果验证问题。

**【 附图说明 】**

图1是现有技术的对称加密搜索模型的示意图；

图2a是本发明具体实施方式的搜索方法中将键值对插入MPT树结构的分支节点的一种情形时的示意图；

图2b是本发明具体实施方式的搜索方法中将键值对插入MPT树结构的分支节点的另一种情形时的示意图；

图2c是本发明具体实施方式的搜索方法中将键值对插入MPT树结构的叶子节点的一种情形时的示意图；

图2d是本发明具体实施方式的搜索方法中将键值对插入MPT树结构的叶子节点的另一种情形时的示意图；

图3是本发明具体实施方式的搜索方法中的倒排索引以及键值对的示意图；

图4是本发明具体实施方式的搜索方法中构建的MPT树结构的示意图；

图5a是本发明具体实施方式的搜索方法中用户搜索的令牌存在时的结果证明的示意图；

图5b是本发明具体实施方式的搜索方法中用户搜索的令牌不存在时的结果证明的示意图。

**【 具体实施方式 】**

下面结合具体实施方式并对照附图对本发明做进一步详细说明。

可验证加密搜索方案的系统框架包括client和server两个实体，其中client将与其数据对应的验证索引（index）上传给云；云服务器（server）为client提供存储、搜索和验证服务；client搜索后对云服务器返回的搜索结果进行验证。如下，在描述系统的工作流程之前，先给出本具体实施方式将会用到的标记以及部分概念的含义。

|  |  |
| --- | --- |
| 记号 | 含义 |
|  | 表示关键字集合 |
|  | 表示关键字的总数 |
|  | 表示一个特定的关键字， |
|  | 关键字对应的令牌 |
|  | 表示文件集合 |
|  | 表示包含关键字的明文文件集合 |
|  | 表示包含关键字的密文文件集合 |
|  | 表示一个文件的明文 |
|  | 表示一个文件的密文 |
|  | 表示一个文件中包含的所有关键字集合 |
|  | 表示验证索引 |
|  | 表示结果证明（即验证结果） |

系统的工作流程定义如下：

:是由Client执行的概率计算操作。它的输入为一个安全参数，输出为对称密钥集合。

:是由Client执行的初始化操作。它的输入为对称密钥集合以及文件结合，它的输出包括一个验证索引。Client将验证索引上传给服务器。



:是由Client执行的预更新操作。它的输入为对称密钥集合以及待更新文件，输出为更新令牌集合。Client将更新令牌集合上传给云。



:是由Server执行的更新操作。它的输入为验证索引和更新令牌集合，输出为更新后的验证索引和更新路径。Server将更新路径发送给Client。



:是由Client执行的更新操作。它的输入为验证索引根哈希和更新路径，输出为更新后的根哈希值。

:是由Client发起的挑战操作。它的输入是对称密钥集合与关键字，输出为该关键字对应的挑战令牌。Client将该令牌提交给云。



:是由Server执行的证明操作。它的输入是验证索引和一个挑战令牌，输出是结果证明。服务器将该结果证明发送给发起挑战的Client。



：是由Client执行的验证操作。它的输入为对称秘钥集合，服务器返回的搜索结果，结果证明以及挑战令牌，最终算法输出或者。



MPT表示默克尔帕特里夏树（Merkle Patricia Tree，MPT）。最早在以太坊（Ethereum）中提出，它将传统的Trie Tree和Merkle Tree结合，使得该树同时具有查找和验证的功能。MPT具有四种类型的节点，分别为空节点（Blank Node，BN），叶子节点（Leaf Node，LN），分支节点（Branch Node，BN）和扩展节点（Extension Node，EN）。其中空节点只是一个不存任何信息的节点，叶子节点存储了键值对（key-value pair），扩展节点也存储了键值对，但扩展节点的键值分别为其子节点的公共前缀和子节点的哈希值。分支节点有17个元素，其中前16个元素代表了该节点上有可能的分支，即16个十六进制数字，第17个元素为值域，当某一个关键字在该分支节点匹配完成时，该关键字对应的值就存储该元素中。MPT中的每一个节点都通过RLP进行了编码并对编码值再进行了哈希，在数据库中存储了每个节点的键值对，其中键为该节点RLP编码的哈希，值为该节点的RLP编码。这样每个节点可以通过他的哈希值被引用，就同时保证了MPT的可搜索性和可验证性。通过这种方式，MPT的根哈希成为了整棵树的指纹信息，根哈希的值由所有下层节点的哈希值所决定，任何节点的微小改变都会导致根哈希的值发生变化。

增量哈希函数（Incremental Hash，IH）最早由Bellare等人提出，并且被CS2方案所采用。IH函数是抗碰撞的，其定义为 在IH函数中两个随机字符串的相加或相减不会产生碰撞。

如下将具体阐述本具体实施方式的加密搜索方法的流程。首先，将描述如何建立并更新验证索引，然后将给出生成结果证明的过程，并详细解释如何利用结果证明进行验证，以确保搜索结果的正确性。

1. 建立验证索引

首先Client根据文件集合计算出倒排索引，其中倒排索引是指关键字与包含该关键字的文件组成的索引。对倒排索引中的每一个关键字，计算它的键值对，其中键是每一个关键字通过加密处理（例如伪随机函数）生成的令牌，而值是包含该关键字的所有文件的增量哈希和。通过将这些键值对插入MPT中来形成验证索引。

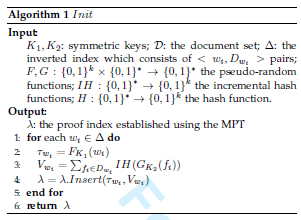


将键值对插入MPT树中时，包括插入键值对到一个分支节点或者叶子节点。

将键值对键入分支节点包括两种可能情况：第一种情况，键为空，此时直接将值存储到分支节点的第17个位置即可。如图2a所示，为插入【key，value】=【“”，dog】到分支节点的示意图。第二种情况，键值不为空，此时生成一个新的叶子节点，将未匹配的键和值存储在该叶子节点中，并通过原始的分支节点指向该叶子节点。如图2b所示，为插入【key，value】=【“345”，dog】到分支节点的示意图。

将键值对插入到一个叶子节点也包括两种情况：第一种情况带插入的键和叶子节点的键完全匹配，这时，只需要修改原有叶子结点的值为新的值即可。如图2c所示，为插入【key，value】=【“123”，dog】到叶子节点的示意图。第二种情况插入的键和叶子节点的键不匹配，则需要利用能匹配的公共前缀生成扩展节点，然后利用分支节点的多分支特性，指向多个叶子节点来存储新的键值对【1200，dog】和原有叶子节点中的键值对【123，cat】。如图2d所示，为插入【key，value】=【“1200”，dog】到叶子节点的示意图。

如下为实现上述建立验证索引的代码：



对验证索引的更新操作支持三种方式，即插入、删除和编辑文件，其中编辑文件相当于删除一个文件后再新增一个文件。对于插入新文件操作，首先解析该文件，得到该文件包含的关键字集合，对每一个关键字，通过伪随机函数生成它的令牌，并将文件的伪随机结果同时上传给云。Server收到后通过更新令牌找到对应的叶子节点，并将 （包含关键字K的文件f进行增量哈希运算得到的值）与原有的叶子节点的值相加。删除操作是同样的过程，只是将原有的叶子节点的值减去。



需说明的是，“键”的称呼是对应于放在MPT树结点的称呼；令牌（或者叫陷门trapdoor）的称呼是对应于在传输过程中。一个加密后的关键字是一个令牌，存储到MPT中后，每一条从根节点到叶子节点的路径组成了一个令牌。也即，MPT从根到叶子的路径=令牌=加密后的关键字。

具体地，如图3所示，以包含文件f1~f4的文件集合为例，计算倒排索引后，包含关键字w1、w2、w3和w4。图3中的第一栏与第二栏即为由关键字与包含该关键字的文件组成的倒排索引。第三栏为键，即各关键字对应的令牌（token）。第四栏为值，也即包含该关键字的文件的增量哈希和。将图3所示的键值对插入MPT树结构中，得到的MPT树结构即如图4所示。

客户端一般是将MPT树结构的初始结构上传给服务器，服务器存储该初始结构的MPT树结构后，当有更新时，接收更新的键值对，根据更新的键值对更新初始的MPT树结构，得到最新的MPT树结构。例如，当文件集合D中新增文件f5时，得到该文件包含的关键字集合为w2和w5。对每一个关键字，通过伪随机函数生成它的令牌，并计算文件f5的增量哈希值。对于每一个关键字，将键值对（令牌和文件f5的增量哈希值 ）同时上传给云。Server收到后，对于已经存在的关键字，通过令牌找到对应的叶子节点，并将与原有的叶子节点的值相加。对于不存在的关键字，则创建一个新的叶子节点，并将作为它的节点值。图3和图4中用浅色的示意了该更新过程。注意，服务器在更新完MPT后，需要将更新路径发回给客户端，以便用户进行验证并更新根哈希值。

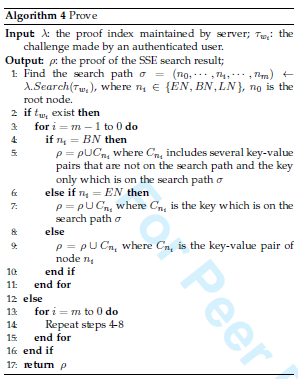
搜索时，客户端向服务器发送挑战令牌。服务器根据客户端发送的挑战令牌，进行搜索，得到搜索结果。此外，服务器根据客户端发送的挑战令牌，在MPT树结构中寻找令牌对应的搜索路径，提取搜索路径上的键值对作为验证结果。

2、生成结果证明（即验证结果）

Server根据用户提交的挑战令牌和验证索引来生成结果证明。首先Server根据挑战令牌来寻找搜索路径。如果对应的叶子节点存在，即用户查询的关键字存在，则服务器从叶子节点上一层节点开始，返回搜索路径上的‘‘键’’作为结果证明。对于分支节点，服务器还返回不在搜索路径上的键值对，从而便于后续重构出MPT树结构的根哈希。如果对应的叶子节点不存在，即用户查询的关键字不存在，则服务器需要从搜索终结的节点开始，向上返回搜索路径中的‘‘键’’作为结果证明，而对于搜索的终结节点，服务器返回完整的键值对。



如下为实现上述生成结果证明的代码：



客户端收到上述结果证明值后，即可执行验证操作。

3、进行结果验证

当Client收到了搜索结果及其对应的结果证明时，就可以开始验证数据的新鲜性和完整性。

首先，Client通过在搜索的时上传的关键字的令牌与结果证明中的键进行匹配。

如果结果证明中的键是挑战令牌的前缀，则设置存储挑战令牌中剩余的键。从根节点到叶子节点的路径事实上是一个完整的key，但前述结果证明中返回的是从根节点到叶子结点的上一层节点的路径，是不完整的，因此定义的remain\_key用于指代令牌与结果证明中匹配完成后剩余的key。例如，以匹配图3中的令牌a5432时，服务器返回的结果证明包括BN2，EN1，BN1，这条路径只包含了a54，则与a5432匹配后，remain\_key就是32。

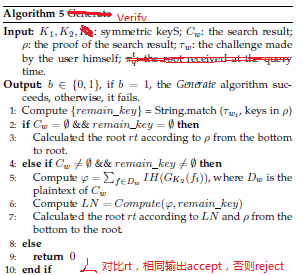
如果结果证明中的键不是挑战令牌的前缀，那么设置就为。



其次，重建MPT树结构的根哈希值。如果搜索结果和都为空集，则通过结果证明直接计算出根哈希值。如果二者都不为空，则首先通过搜索结果和生成叶子节点的哈希值，得到完整的结果证明值，然后通过结果证明值重建出根哈希值。除了这两种情况以外，就认为服务器故意返回了空结果，或服务器篡改了结果证明的内容。

最后，用户通过对比重建得到的根哈希和客户端存储保留的根哈希是否相等来判断数据新鲜性和完整性。如果二者相等，则验证通过，如果二者不相等，则说明服务器少返回了搜索结果或者服务器篡改了结果证明。

如下为实现上述结果验证的代码：



如下结合图3和图4的具体结构说明上述生成结果证明和验证步骤的具体过程。

一种情形：

当客户端想要搜索的关键字为时，根据图3所示的对应关系，则提交的对应该关键字的挑战令牌为“a5432”。由于该关键字令牌在图4所示的MPT树（也即验证索引）中已经存在，服务器可以找到与该令牌对应的搜索路径为。根据上述生成结果证明的过程，服务器则返回路径上除LN3以外的节点（从根节点BN1开始到叶子节点的上一层节点BN2）的键值对以及不在路径上的分支节点的键值对作为结果证明。得到的结果证明如图5a中的所示。其中，结果证明对应分支节点BN2上的提取内容，也即键“4”，其值为空（因为未提取LN3的内容），键“c”及其值LN4，键“f”及其值LN5。结果证明对应扩展节点EN1上的提取内容，也即键“5”及其值，其值即为。结果证明对应分支节点BN1上的提取内容，也即键“4”及其值LN1，键“a”及其值，键“f”及其值LN2。

提取得到上述结果证明值后，可以根据该证明和搜索结果重新构建MPT树结构的根哈希。首先用户将令牌“a5432”与结果证明中的键进行匹配，发现“a54”为令牌的前缀，为“32”。用户根据“32”以及搜索结果重新生成节点LN3，则可完善结果证明。这样，通过完善的结果证明中的内容，自底向上构建出根哈希的值。最后用户通过比较重构得到的根哈希和用户端保留的根哈希，来判断数据是否完整。假如服务器只返回了文件，那么重构得到的根哈希将与正确的根哈希不匹配。

另一种情形：

当用户想要搜索的关键字对应的令牌为“a5433”时，根据图3和图4可知，该令牌在MPT树（也即验证索引）中不存在，但其搜索路径与“a5432”相同，不同的是，该令牌在LN3处发生了不匹配。此时，服务器提取从LN3节点开始自底向上的路径上的节点的键值对来生成结果证明，如图5b中的所示。相对于图5a所示的情形，多提取了叶子节点LN3的键“32”以及值“H2”作为。需说明的是，如为增加文件f5之前，则叶子节点的值为H2。如为更新增加文件f5之后，则叶子节点的值为H2+。

用户在收到该结果证明以后，由于发现令牌“a5433”与结果证明中的键a5432无法匹配，因此被置空。则用户将直接根据结果证明重构根哈希。同样，通过与正确的根哈希进行对比，如果不相同，则说明服务器篡改了结果证明，产生了恶意行为。

综上，本具体实施方式通过Merkle Patricia Tree（MPT）构建索引，并以该索引为基础，提出了一种完善的结果验证机制。验证时，通过MPT树的根哈希进行验证。由于根哈希的第一次生成是由用户执行的，因此可以保证根哈希的正确性。随后的根哈希更新也由用户验证并更新，因此可以保证根哈希的正确性和新鲜性。结果验证过程中，是基于用户保留的根哈希来进行最终的比对判断，任何节点的改动，都会引起重构出的根哈希与用户保留根哈希无法匹配。因此可以保证验证环节的参照对象准确可靠，从而验证后确保用户拿到的搜索结果是新鲜并且完整的。

本具体实施方式的搜索方法，可以防止数据完整性攻击和数据新鲜性攻击。特别地，本方法可以在用户不保留关键字集合的情况下，检测出服务器故意返回空结果的情况。该方法是一种通用的结果验证方案，它将传统加密搜索方案作为黑盒，可以为多种加密搜索方案提供结果验证功能。此外，通过一次通讯轮次即可实现验证搜索，具有开销小的优势。综上，本方法具有开销小、实现简单以及不依赖于原有加密搜索方案的特性。

以上内容是结合具体的优选实施方式对本发明所作的进一步详细说明，不能认定本发明的具体实施只局限于这些说明。对于本发明所属技术领域的普通技术人员来说，在不脱离本发明构思的前提下做出若干替代或明显变型，而且性能或用途相同，都应当视为属于本发明的保护范围。

说明书附图

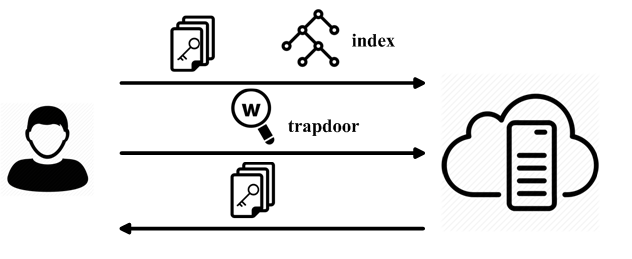


图1

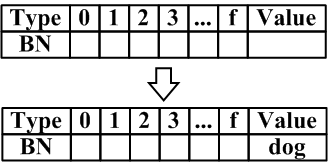


图2a

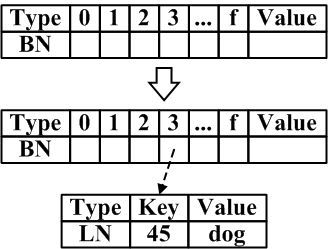


图2b



图2c

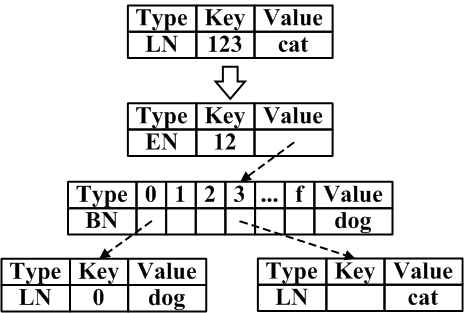


图2d

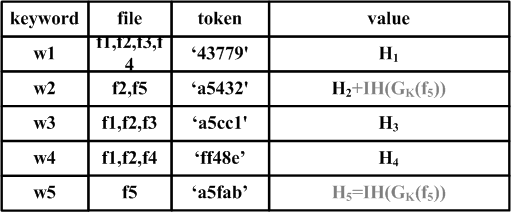


图3

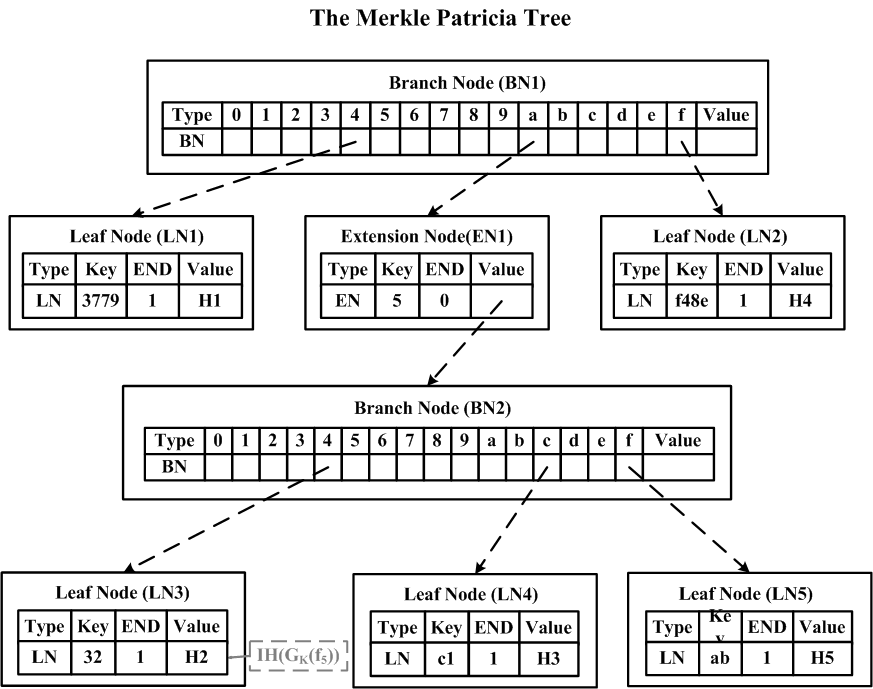


图4

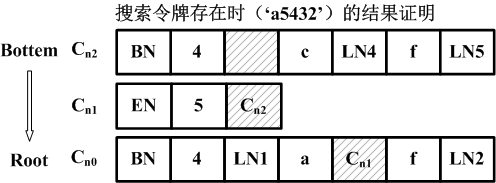


图5a

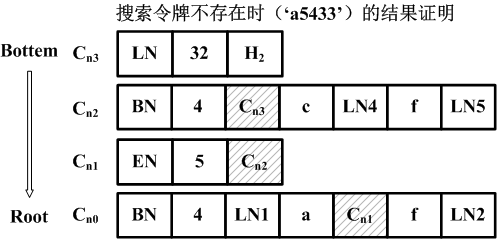


图5b