基于平衡Merkle树的工业OT网络访问授权溯源方法

谢鹏寿[1]，冉玉翔[1]，冯涛[1]，康永平[2]，杨兴慧[1]，杨超[1]
（1.兰州理工大学计算机与通信学院，甘肃 兰州 730050；2.兰州理工大学机电工程学院，甘肃 兰州 730050）

摘　要：为解决工业OT网络中访问授权溯源效率较低的问题，提出了一种基于平衡Merkle树的访问授权溯源方法。该方法在传统布隆过滤器和Merkle树的基础上，对原始结构进行重构，构造双层布隆过滤器与平衡Merkle树，使之更加契合工业OT网络中数据的处理。在构造过程中，为解决构造平衡Merkle树所带来的时间开销，使用分段构造的方法。同时，引入星际文件系统（IPFS）存储来确保每一次访问操作和原始数据的可靠性和安全性。仿真结果表明，在不同数据量下，所提方法显著提升了访问授权溯源效率，减少了系统响应时间。

关键词：工业OT网络；平衡Merkle树；信息溯源；布隆过滤器；星际文件系统
中图分类号：TP309
文献标志码：A
DOI：10.11959/j.issn.1000−436x.2025063

Industrial OT network access authorization traceability method based on balanced Merkle tree

XIE Pengshou[1], RAN Yuxiang[1], FENG Tao[1], KANG Yongping[2], YANG Xinghui[1], YANG Chao[1]
1. School of Computer and Communications, Lanzhou University of Technology, Lanzhou 730050, China
2. School of Mechanical and Electrical Engineering, Lanzhou University of Technology, Lanzhou 730050, China

Abstract: To solve the problem of low efficiency of access authorization traceback in industrial operational technology (OT) network, an access authorization traceback method based on balanced Merkle tree was proposed. Based on the traditional Bloom filter and Merkle tree, the original structure was reconstructed to construct a double-layer Bloom filter and a balanced Merkle tree, which was more suitable for the data processing in industrial OT network. In order to solve the time cost caused by the construction of a balanced Merkle tree, a segmenting construction method was used in the construction process. At the same time, interplanetary file system (IPFS) storage was introduced to ensure the reliability and security of every access operation and raw data. Experimental results show that under different data sizes, the proposed method significantly improves the efficiency of access authorization traceability and reduces the system response time.

Keywords: industrial OT network, balance Merkle tree, information traceability, Bloom filter, IPFS

0　引言

随着工业物联网技术的迅猛发展，工业控制系统也日益普及和应用于各行各业的工业领域。然而，随之而来的是对工业OT（operational technology）网络安全性和数据完整性的日益关注。在一个复杂的工业OT网络中，多个设备、传感器和控制器相互连接，形成了一个庞大的网络生态系统。为了确保该网络生态系统的稳定性和可靠性，对网络访问授权[1]信息的溯源变得至关重要。Merkle树作为一种高效的数据验证结构，被广泛应用于区块

收稿日期：2024−11−07；修回日期：2025−03−23
通信作者：冉玉翔，417914748@qq.com
基金项目：国家自然科学基金资助项目（No.61862040, No.62162039）
Foundation Items：The National Natural Science Foundation of China (No.661862040, No.62162039)

链[2]、供应链管理、数据完整性验证等领域。Merkle 树通过对数据进行逐层的哈希运算，形成一个唯一的根哈希值，使得任何对数据的改动都能够被快速检测到。然而，尽管 Merkle 树在数据完整性验证方面具有显著优势，但其在实际应用中仍面临以下问题。

1) 查询效率低。在传统 Merkle 树中，查询特定数据需要遍历整个树结构。对于工业 OT 网络中大规模数据的场景，这种查询效率难以满足实时性要求。

2) 开销大。在数据量较小的情况下，这种开销尚可接受。但在工业 OT 网络中，产生的数据量呈指数级增长，Merkle 树的存储开销也随之增加，这对资源受限的工业设备来说是一个巨大的挑战。

3) 构造时间长。Merkle 树的构造过程需要对数据逐层计算哈希值，当数据量较大时，构造时间会显著增加，影响系统的整体响应速度。

在现有的溯源研究中，研究人员更加侧重于对供应链的溯源，陈飞等[3]、Centobelli 等[4]和 Lai 等[5]仅使用区块链或 Merkle 树实现供应链的溯源，并未涉及溯源过程中效率的提升，这也是目前大多数研究人员研究的内容所在。Yao 等[6]和 Liu 等[7]在实现溯源的过程中更加注重安全问题，所以在效率方面并没有做出相应的提升。Wu 等[8]虽然在溯源过程中对查询过程提升了效率，但同时增加了空间开销，使存储数据时的时间开销增大，从整个溯源过程来看，整体溯源效率并没有较大的提升。为了解决上述这些问题，基于 Merkle 树实现访问授权信息溯源成为一种备受关注的解决方案。

Merkle 树[9-10]是一种二叉树结构，它通过对数据进行逐层的哈希运算，形成一个唯一的根哈希值。这种树状结构使得任何对数据的改动都能够被快速检测到，而且根哈希值的验证结果可以被广播到整个网络中进行验证。基于 Merkle 树的工业 OT 网络访问授权信息溯源利用了这种结构的特性，通过将授权信息存储在 Merkle 树中，并将根哈希值广播到整个网络，实现授权信息的溯源和验证。在数据完整性方面，Merkle 树可以帮助验证数据的完整性[11]，这对防止数据被篡改或损坏至关重要。在基于 Merkle 树的授权追踪[12]中，数据被组织成树状结构，每个节点的哈希值都由其子节点的哈希值计算得出，从根节点到叶子节点的哈希值链构成了数据的唯一指纹。通过比较数据的哈希值，可以迅速检测到数据是否被篡改。在数据安全性方面，哈希值被用于验证数据，而不是直接暴露原始数据。这有助于保护数据的隐私[13-14]，因为只有拥有正确的哈希路径的参与者才能访问特定数据，其他人无法获取实际数据内容。在溯源效率方面，Merkle 树通过递归组合数据块的哈希值生成单一根哈希值，从而使验证特定数据是否属于某集合的过程变得高效且简便。这一特性在区块链、供应链管理[15-17]和云存储等领域得到了广泛应用，能够提供快速且可靠的数据验证机制，显著降低计算开销并提升系统透明度。

在溯源过程中，为使效率进一步提升，引入布隆过滤器[18-22]是当前研究的焦点之一。布隆过滤器是一种空间效率极高的概率型数据结构，用于快速判断一个元素是否属于某个集合。其核心思想是通过多个哈希函数将元素映射到一个位数组中，并通过位数组中的值来判断元素是否存在。研究者们致力于优化布隆过滤器的结构和算法，降低假阳性率[23-24]，并探索其在动态调整大小[25]、并行处理和分布式环境[26]下的应用。这些努力不仅提升了溯源系统的处理速度和资源利用率，还为大规模数据管理提供了有效的解决方案。布隆过滤器的引入为溯源技术的进一步发展和实际应用奠定了坚实的基础，但对构造过程中所带来的时间开销并没有考虑。

为使数据存储更加安全，引入星际文件系统（IPFS, interplanetary file system）存储[27-30]，研究主要聚焦于数据完整性、隐私保护和防篡改等领域。IPFS 存储采用内容寻址和加密哈希技术，能够确保数据在传输和存储过程中的完整性与防篡改能力。此外，通过分布式存储模式，IPFS 有效降低了单点故障和集中化攻击的风险。目前的研究还涵盖了访问控制机制、匿名性和抗审查能力，以期增强其在去中心化应用和区块链环境中的安全性能。

针对上述内容，在确保数据完整性和安全性的前提下，本文改善了 Merkle 树和布隆过滤器的结构，以此提高系统整体的溯源效率。本文主要的研究工作如下。

1) 采用平衡 Merkle（BM, balance Merkle）树，相较于 Merkle 树，计算是从叶子节点两两结合计算 Hash 值，直到得出最后的 Merkle 根值，而平衡 Merkle 树将原本的叶子节点这一层删除，将信息存

储在每个节点上面，将原本计算叶子节点Hash值所消耗的时间用来构造平衡Merkle树。

2) 采用双层布隆过滤器（DBF, double Bloom filter），相较于传统的布隆过滤器，在使用相同空间或较少空间的情况下，计算Hash的次数会大幅度减少，同时误判率也会优于的传统布隆过滤器。

3) 通过结合IPFS实现内容可寻址和高效传输，同时采用这种分布式的文件存储机制确保数据的冗余性和可靠性，提高数据的可用性。

1　系统模型

1.1　方案阐述

本文旨在解决工业OT网络中访问授权信息溯源效率较低的问题，为此提出了一种结合Merkle树、IPFS和布隆过滤器的方法。如图1所示，本文方案共分为2个过程，分别为存储过程与查询过程。存储过程为图中①~④，其中，第1步将工业OT网络中产生的访问授权信息存储在IPFS中，第2步将得到的内容标识符（CID, content identifier）存储在平衡Merkle树的节点中，第3步将查找因子存储在BM树的节点中并与CID绑定在一起，第4步用查找因子构造双层布隆过滤器。查询过程为图中⑤~⑨，其中，第5步在DBF中快速查询是否具有所需数据，第6~7步是在第5步中证明存在相关数据时，在其BM树中查询相关信息，第8~9步是在上述步骤中得到相关索引后，对其查询详细信息。

本文提出了基于BM树、DBF和IPFS存储优化的访问授权溯源方法。具体而言，BM树用于高效存储和验证授权信息，DBF用于快速过滤和查询数据，而IPFS存储优化则确保数据的安全性和可扩展性。详细介绍如下。

1) BM树作为核心数据结构。BM树优化了传统Merkle树的存储效率和查询效率，减少了哈希计算的时间开销。BM树的每个节点不仅存储授权信息的哈希值，还包含IPFS中对应数据的CID，从而将数据存储与验证过程解耦。

2) DBF作为前置过滤层。在BM树查询之前，DBF用于快速判断目标数据是否存在于系统中。DBF的查询结果直接决定了是否需要进一步在BM树中进行详细查询，从而减少了不必要的计算开销。

3) IPFS存储优化作为底层支持。IPFS通过分布式存储和内容寻址技术，确保了数据的高可用性和完整性。本文在IPFS的基础上引入了数据分片、加密存储和缓存优化机制，进一步提升了数据的安全性和访问效率。IPFS的CID与BM树的节点直接关联，确保了数据在存储过程和验证过程中的一致性。

1.2　BM树构建

本文方案是在Merkle树的基础上进行改进，采用BM树，即在构造Merkle树时，通过平衡二叉树的构造方法构造BM树。本文方案是基于以下3点考虑的。1) 保证数据的不可篡改性。2) 节省大量的存储空间，减小空间开销。3) 采用BM树的构造方法，减少查询过程的时间开销。针对第1点，在IPFS存储时采用哈希算法，利用哈希算法的单向性保证链下数据的不可篡改，同时将地址信息存储在Merkle树中，如果链上数据或IPFS中存储的内容发生改变，则其得到的哈希结果是不一样的。针对第2点，Merkle树存储结构如图2所示，该树的

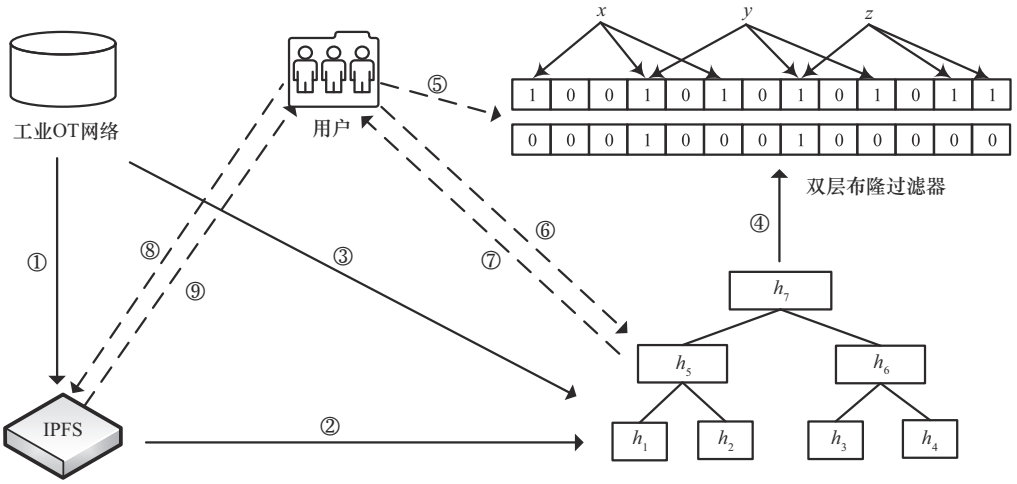


图1　基于BM树的工业OT网络访问授权溯源方法

总存储节点数量与树高的关系为

$$n = 2^{T-1} \tag{1}$$

其中，n表示总存储节点数量，T表示树高。

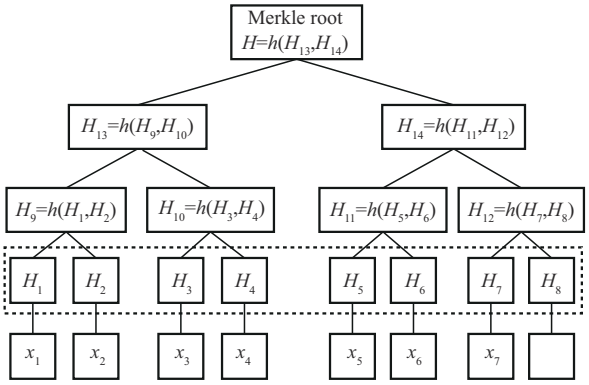


图2　Merkle树存储结构

本文采用BM树存储结构如图3所示，树的总存储节点数量与树高的关系为

$$n = 2^{T} - 1 \tag{2}$$

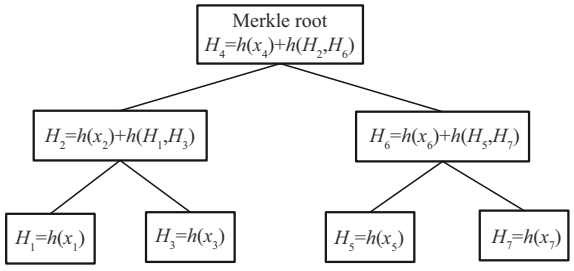


图3　BM树存储结构

如图2中虚线框所示，与本文方案对比，Merkle树多了叶子节点，致使存储相同数量的数据时，存储空间多了一个量级。

针对第3点，本文采用BM树存储结构，可大幅度提高查找授权信息效率，而Merkle树存储结构若要查找则需遍历每一个节点，使时间开销增大。

本文采用BM树存储结构，将信息存储在节点中，通过每次插入哈希值的大小把节点插入合适的位置。存储信息后若BM树未达到满二叉树的标准，本文采用与Merkle树相同的处理方法，使用空节点代替空缺的位置，达到满二叉树的标准。具体实现算法如算法1所示。

算法1　构建BM树

输入　授权记录Hash值

输出　BM树

1) LL_rotate()、RR_rotate()、LR_rotate()和RL_rotate()分别为左左、右右、左右和右左旋转函数
2) LRBF()平衡因子获取函数，利用左右子树高度差计算
3) balance()平衡函数
4)　获取当前节点平衡因子
5)　if BF > 1：当前节点平衡因子大于1，说明左子树过高
6)　　当前节点左子树的平衡因子大于0，则执行左左旋转，否则进行左右旋转
7)　else BF < −1: 当前节点平衡因子小于−1，说明右子树过高
8)　　当前节点右子树的平衡因子大于0，则进行右左旋转，否则进行右右旋转
9)　end if
10)　返回当前节点
11) insert()插入节点函数，新插入的节点都是通过Hash计算的值
12)　if newroot ≤ root: 利用递归在当前节点的左孩子节点插入新节点，并平衡结构
13)　else newroot > root: 利用递归在当前节点的右孩子节点插入新节点，并平衡结构
14)　end if
15) BMerkle_Value()计算BM树根节点Hash值
16)　if node = None: 返回空
17)　end if
18)　利用递归计算相邻叶子节点的Hash值，最终得到BM树根节点的Hash值

在上述算法中，若出现数据量过大的情况时，构造BM树会带来大量的时间开销，因此本文在数据量过大时，采用分段构造的方法构造BM树。该方法首先将整个数据分割成若干较小的部分，然后对每部分数据进行BM树的构造，最后将这些独立的分段BM树整合成完整的BM树。优势在于减少了单次构造的时间开销和降低内存占用，同时，每次只需要处理数据的一部分，因此能够大幅减少内存的使用量，适用于资源受限的环境或设备。分段大小直接影响构造时间和查询效率。较小的分段可以减少单次构造的时间开销，但会增加整合分段BM树的开销，较大的分段则相反。此外，分段数

量需根据总数据量和选定的分段大小来动态调整，确保每一分段的数据量适中，既能有效减少构造时间，也能避免分段过多而导致的整合成本上升。具体实现算法如算法2所示。

算法2　分段构造BM树

输入　授权记录Hash值hash_X，分段后每一段的数据量

输出　BM树

1) 计算分段后每一段的数量
2) while 2^i < part:
3)　　调整分段数量为2的幂
4) for *i* in range(part): 遍历所有分段
5)　　计算并得到分段BM树
6)　　将分段BM树构造为完整BM树
7)　　验证并平衡当前BM树
8) end for

本文提出的BM树的节点初始状态存储了信息，使存储结构与原有Merkle树的结构发生了改变，Merkle树的叶子节点存储信息的Hash值，非叶子节点存储左右孩子的Hash值。在本文提出的BM树中，访问授权信息存储在IPFS中，得到的CID便是节点中存储的内容。在构造BM树后，再次计算左右孩子的Hash值，将得到的结果与节点中存储的内容连接在一起。换言之，BM树节点存储的内容的长度是要长于Merkle树节点存储的内容，但增加的存储内容并不会影响BM树的其他性能。节点存储内容例子如下。

‘QmYqsDzm2op3sfNWdFtLW2XNJPwB2pXkBF5QCDSnLuszp8’ + ‘Hash（查找因子）’ + ‘Hash（左孩子节点、右孩子节点）’。

在该例子中，CID采用V0版本，且长度一定，方便后续查询。而对查找因子与孩子节点的Hash计算结果长度也是一定的，即使节点存储内容增多，也不影响系统整体的性能和效率。

1.3　布隆过滤器构建

本文所提DBF在原有布隆过滤器的基础上再加一层布隆过滤器，相较于传统布隆过滤器，2种方案在使用同等空间的情况下，DBF将此空间分为2个大小相等的空间，而DBF因两层空间大小相等，使用相同的Hash函数。同时，因单层布隆过滤器空间使用较小，故在布隆过滤器中使误判率达到最小时，所使用Hash函数的个数也会相应减少，这在数据量较大的情况下会节省大量时间开销。双层布隆过滤器如图4所示，只有在一层布隆过滤器中误判才会启用二层布隆过滤器，而DBF的误判率是两层布隆过滤器误判率的乘积。

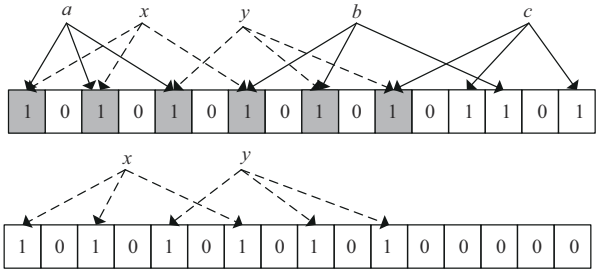


图4　双层布隆过滤器

在布隆过滤器中，当添加一个元素的一次散列时，其中某一比特位的值为1的概率为$\frac{1}{m}$，则任意一位比特位的值为0的概率为$1-\frac{1}{m}$。添加一个元素后，任意一位比特位的值为0的概率为$\left(1-\frac{1}{m}\right)^k$，为1的概率为$1-\left(1-\frac{1}{m}\right)^k$。添加$n$个元素后，任意一位比特位的值为0的概率为$\left(1-\frac{1}{m}\right)^{kn}$，为1的概率为$1-\left(1-\frac{1}{m}\right)^{kn}$。因此，如果将一个新元素添加到已存在$n$个元素的布隆过滤器中，$k$个比特位的值为1的概率为$\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^k$，此概率即为误判率。而在本文提出的DBF中，当插入一个新元素时，在一层布隆过滤器中，通过计算K个Hash值后，判断每一位都存在冲突时，则启用二层布隆过滤器。如此，DBF的误判率为$\left(\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^k\right)^2$。具体实现算法如算法3所示。

算法3　双层布隆过滤器构建

输入　查找因子，哈希函数

输出　双层布隆过滤器位数组

1) 初始化双层布隆过滤器，将所有位置置0
2) for *i* in range(n):n为元素个数
3)　　for *j* in range(k):k为Hash函数个数

4)　　if $BF1_{h_j(x_i)} = 1$:

5)　　统计当前元素在DBF第一层中不同位上被置1的个数，被记作特征值

6)　　end if

7)　end for

8)　for j in range(k):

9)　　if flag ≠ k: 判断特征值与Hash函数个数是否相等。若不相等，则通过k个Hash函数在第一层中映射该元素信息；若相等，则通过k个Hash函数在第二层中映射该元素信息

10)　　end if

11)　end for

12)　计算完成一个元素，特征值置0

13) end for

1.4　链下扩展存储

IPFS通过内容寻址技术来存储和检索文件，文件在IPFS中被拆分为多个小块，每个块都有一个唯一的内容标识符。分布式哈希表（DHT, distributed hash table）用于存储和检索这些块的CID。工业OT网络中的数据量通常非常庞大，且链上的存储能力有限，因此需要链下扩展存储方案。针对本文相关诉求结合IPFS为链下扩展存储提供了理想的解决方案，改进措施如下。

1) 数据分片与加密存储。IPFS的CID确保文件内容不可篡改，任何对文件的修改都会导致CID发生变化，从而容易检测到数据是否被篡改。同时，通过将数据分片存储在不同的IPFS节点上，并结合加密算法，确保数据在存储和传输过程中的安全性。即使某个节点被攻击或数据被窃取，攻击者也无法获取完整的原始数据，从而有效防止数据泄露。

2) 动态访问控制与高可用性。针对授权后的数据，结合动态访问控制机制与IPFS，每次数据访问请求都会经过验证，确保只有授权用户才能获取相应的CID并访问数据。同时，所有访问授权操作会被记录在区块链上，便于对违法操作的溯源。

3) 高效存储和访问。提出了一种缓存优化机制，通过分析访问频率，动态调整缓存策略，减少数据检索时延，提高系统响应速度。这种机制适用于工业OT网络中频繁访问的授权信息，能够显著提升系统的整体性能。

通过将IPFS与BM树结合使用，以实现更高效和更安全的数据管理和存储。具体流程如下。

1) 存储访问授权记录信息。首先将访问授权记录信息存储在IPFS中，生成对应的CID，并将该CID传递给BM树。

2) 生成Merkle树。以CID作为叶子节点，将查找因子计算的哈希值与相应的CID存储在同一节点中，逐层计算父节点的哈希值，最终生成Merkle树根节点。

3) 存储根节点。将最终生成的BM树根节点的哈希值存储在IPFS中。

4) 根节点验证。通过存储Merkle树根节点的哈希值，可以通过根节点的哈希值验证整个数据集的完整性，用于后续数据是否被篡改的验证。

2　溯源查询阶段

2.1　布隆过滤器查询

在布隆过滤器查询阶段，由于在构造DBF时，只有在第一层产生误判的情况下才会启用第二层布隆过滤器，若在第一层中发生误判，说明该数据在第二层中，即只需对第一层布隆过滤器进行查询即可，若第二层中仍然发生误判，这也是在DBF的误判率的计算之内。双层布隆过滤器的查询算法如算法4所示。

算法4　双层布隆过滤器查询

输入　目标数据

输出　查询结果（是否存在）

1) for i in range(k):k为Hash函数个数

2)　　通过不同的Hash函数计算目标数据在布隆过滤器位数组中的位置

3)　　if BF1[index] == 0: 判断当前位置是否为0

4)　　若是，则返回False，证明目标数据不存在；若不是，则用下一个Hash函数计算

5)　　end if

6) end for

7) 若在步骤4)中未返回False，则返回True，证明目标数据存在

2.2　BM树查询

DBF的主要作用是快速判断目标数据是否存在，从而减少不必要的BM树查询。DBF通过双层结构减少了哈希计算的次数。当用户发起查询请求

时，首先在DBF中进行快速过滤，如果DBF返回“不存在”，则直接结束查询，不需要进入BM树查询阶段。如果DBF返回“可能存在”，则进一步在BM树中进行详细查询。这种分层查询的设计使BM树的查询负载显著降低，尤其是在大规模数据场景下，能够有效减少查询时间。DBF与BM树的结合特别适用于大规模数据的实时查询需求，能够有效应对数据量大和查询频繁的挑战。

在BM树查询阶段，与传统Merkle树查询不同，由于通过DBF已确定该棵树中存在需要的信息，所以在查询时只需对比每一层的节点值，便可快速查找到所需信息。BM树查询算法如算法5所示。

算法5 BM树查询

输入 查找值

输出 CID（版本：V0）

1) search(root,x): 递归函数名及参数

2) if root is Null: 判断当前节点是否为空，若为空，则返回递归的上一步；若不为空，则进行下一步

3) end if

4) if root.key[l:end_index] == x: 比较当前节点所存的值与目标值是否相等，若是，则返回完整的CID；若不是，则进行下一步

5) end if

6) if x < root.key[l:end_index]:

7) 若满足，则递归搜索左子树；若不满足，则递归搜索右子树

8) end if

2.3 IPFS查询及下载

IPFS使用Merkle树管理数据，每个数据块都有一个唯一的内容标识符CID。通过BM树查询得到的CID，用户可以在IPFS中查询并下载相关数据。BM树的存储结构确保数据的完整性，因为任何数据的修改都会导致CID的变化，从而使数据篡改容易被检测到。因此，用户只需通过CID就能快速、准确地在IPFS中找到并获取所需的数据。

3 BM树数据验证

在传统Merkle树中，数据溯源过程需对树中叶子节点存储的数据进行安全性验证，以此确保数据未被篡改。本文提出的BM树在溯源时也需对数据进行安全性验证，原始值为Merkle Root值，对溯源数据进行计算验证，与原始值相对比进行判断。若结果相同，则表明溯源数据未被篡改且可信。若结果不同，则表明该溯源数据已被恶意篡改，且当前溯源数据不可信。验证过程具体步骤如下。

步骤1 从验证数据节点开始，向上层节点查找。

步骤2 查找到该棵树中参加运算的所有节点。

步骤3 将查找出的所有节点构成一个证明集合，即BM树证明集合。

上述阶段算法实现如算法6所示。

算法6 BM树证明算法第一阶段

输入 BM树，需证明数据索引

输出 证明集合

1) for index in tree:

2) if index%2 == 0: 确定兄弟节点索引，判断当前节点的索引是偶数还是奇数，偶数索引的兄弟节点在其右边（索引+1），奇数索引的兄弟节点在其左边（索引−1）

3) end if

4) if sibling_index < len(level): 检查兄弟节点是否存在，若存在，则添加到证明集合；若不存在，则直接跳转下一步

5) index = $\frac{\text{index}}{2}$ 更新索引，使其索引指向父节点

6) end if

7) end for

8) 返回构建好的证明集合

步骤4 将该证明集合进行Hash运算得到的值与原始Merkle Root值比较是否一致，以此完成验证过程。

如图5所示，以节点H_{12}为例BM树的证明过程要证明h_{12}节点，就得需要图中虚线圈住节点的Hash值。详细流程如下。

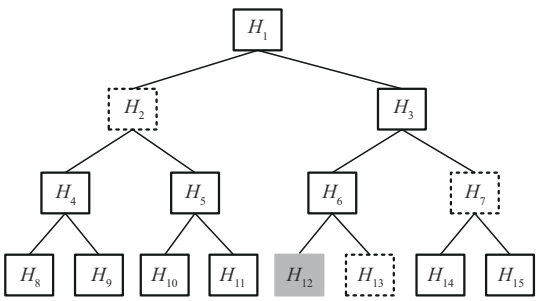


图5 BM树证明

获取 H_{13} 并计算 $h(H_{12},H_{13})$，获取 H_7 并计算 $h(h(H_{12},H_{13}),H_7)$，获取 H_2 并计算 $h(H_2, h(h(H_{12},H_{13}),H_7))$，之后将 $h(H_2, h(h(H_{12},H_{13}),H_7))$ 与Merkle Root值进行比较是否一致，若一致，则证明节点 H_{12} 并没有被恶意篡改，反之，则被篡改。该阶段算法实现如算法7所示。

算法7　BM树证明算法第二阶段

输入　证明数据索引，证明集合proof

输出　与BM树根值是否相等

1) for sibling_hash in proof: 遍历证明集合

2)　　if index%2 == 0: 根据索引确定哈希的组合顺序并更新哈希值

3)　　end if

4)　　更新索引指向父节点

5) end for

6) 返回验证结果

4　验证分析

4.1　性能分析

本文提出的BM树在存储数据量方面相较于传统Merkle树具有显著优势，对比如图6所示。在相同树高的情况下，BM树的存储数据量约为传统方案的2倍，从而有效地节省了存储空间。

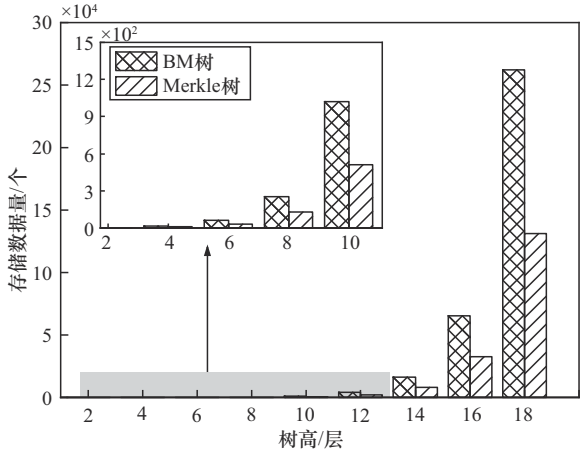


图6　存储数据量对比

在相同数据量（如5 000个数据）的情况下，根据式(1)和式(2)可知，BM树需要13层，而传统Merkle树则需要14层。在计算Merkle根值时，BM树需要计算4 095次Hash值，而传统Merkle树需要计算8 191次Hash值。因此，在存储数据量相同的情况下，通过Hash值计算BM树的Merkle根值时，所需时间仅为传统Merkle树的一半，从而显著降低了时间开销，而减少的时间开销可用来构造BM树的结构。

由前文可知，布隆过滤器的误判率计算式为

$$p=\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^{k}\approx\left(1-\mathrm{e}^{-\frac{kn}{m}}\right)^{k} \tag{3}$$

令 $a=\mathrm{e}^{-\frac{n}{m}}$，可将式(3)转换为

$$\ln p=k\ln\left(1-a^{k}\right) \tag{4}$$

将式(4)求导可得

$$\frac{1}{p}p'=\ln\left(1-a^{k}\right)-\frac{ka^{k}\ln a}{1-a^{k}} \tag{5}$$

得 p,m,n 之间的关系如式(6)所示。

$$\ln p=-\frac{m}{n}(\ln 2)^{2} \tag{6}$$

由式(6)可知，误判率最终的取值与 $\frac{m}{n}$ 比值有关。为了更加直观地证明本文所提DBF，需取最佳 k 值。如表1所示，通过对 $\frac{m}{n}$ 的不同取值，由式(6)与式(3)可计算得出最佳 k 值（第2列数值），同时表1还给出不同 k 值时的误判率。

表1　　布隆过滤器误判率

$\frac{m}{n}$	最佳 k 值	k=2	k=3	k=4	k=5	k=6	k=7	k=8
2	1.39	0.400 0	—	—	—	—	—	—
3	2.08	**0.237 0**	0.253 0	—	—	—	—	—
4	2.77	0.155 0	**0.147 0**	0.160 0	—	—	—	—
5	3.46	0.109 0	**0.092 0**	0.092 0	0.101 0	—	—	—
6	4.16	0.080 4	0.060 9	**0.056 1**	0.057 8	0.063 8	—	—
7	4.85	0.061 8	0.042 3	0.035 9	**0.034 7**	0.036 4	—	—
8	5.55	0.048 9	0.030 6	0.024 0	0.021 7	**0.021 6**	0.022 9	—
9	6.24	0.039 7	0.022 8	0.016 6	0.014 1	**0.013 3**	0.013 5	0.014 5
10	6.93	0.032 9	0.017 4	0.011 8	0.009 4	0.008 4	**0.008 1**	0.008 4

假设使用布隆过滤器对5 000个数据进行判断存在与否。若传统布隆过滤器在采用 $\frac{m}{n}$=10的空间时，即布隆过滤器空间大小为50 000个，从表1中可以看到，只有在使用7个Hash函数时，误判率才会达到最小值0.008 1，同时也需要计算35 000次Hash值。若采用DBF，空间采用 $\frac{m}{n}$=5，即每一层布隆过滤器空间大小为25 000，共使用空间50 000个，

误判率达到最小时只需使用3个Hash函数，且只需计算15 000次Hash值，远小于传统的布隆过滤器，大大减少了构建布隆过滤器的时间开销，此时DBF的误判率为0.092×0.092≈0.008 1，达到了在使用同等空间和误判率相同的情况下效率最高。

4.2　效率分析

模拟环境是在一台配置为英特尔（R）酷睿（TM）i7-8500U CPU、12 GB内存、64位Windows 10专业版操作系统的主机上进行模拟。SCATC索引结构使用Python语言编写和实现。由于区块链要求每个完整节点维护一个完整的账本，因此数据检索将在本地进行。

4.2.1　验证时间开销对比

为讨论BM树在验证数据的效率对溯源效率的影响，本文对完美二叉树、文献[31]和本文方案结构进行验证时间开销对比，如图7所示。由于文献[31]所提出的Merkle山脉容易生成非完美二叉树，因此本文提取文献[31]数据是完美二叉树缺一个叶子节点的情况。当树高分别为14、15和16层时，对数据进行单次验证，本文方案在验证时间开销方面远低于文献[31]和完美二叉树，由于验证过程也是在查询过程基础之上进行的，而BM树在查询时降低了时间开销，因此本文方案的验证时间开销低于文献[31]。

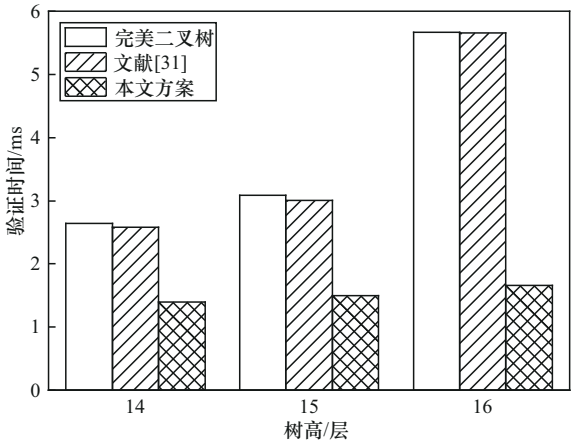
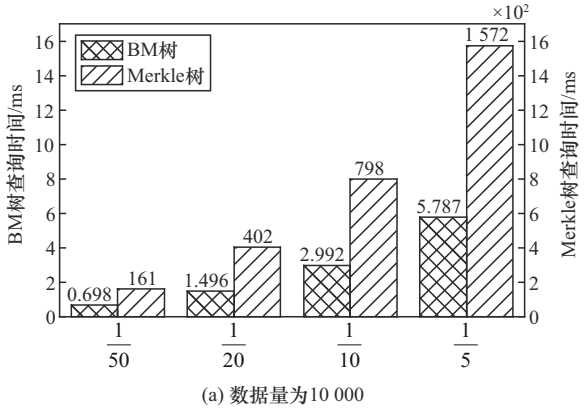


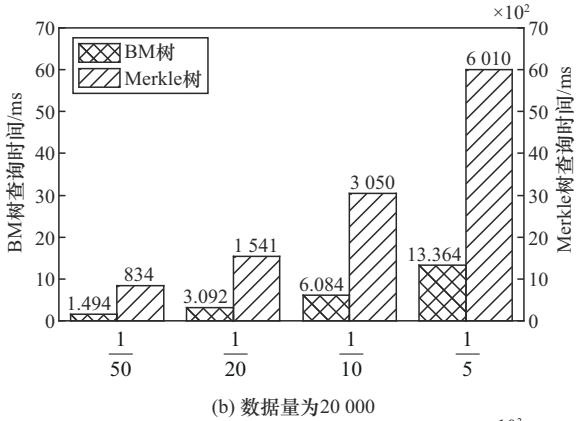
图7　验证时间开销对比

4.2.2　BM树查询时间开销对比

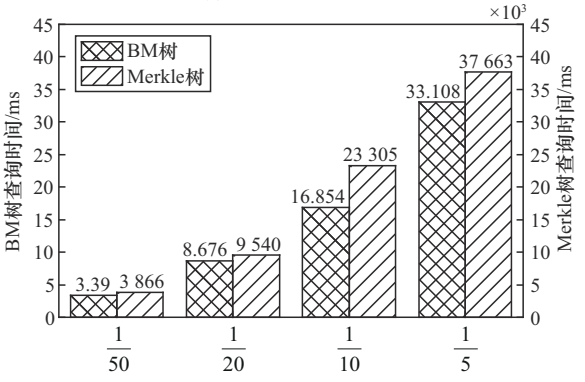
本文采用公开数据集（工业和专业职业数据集（IPOD）-数据集-OpenDataLab），将其转换为本文所需的查找因子，对不同数据量的数据构造BM树与传统Merkle树，并对其进行不同次数的查询。其中，图8(a)~图8(d)分别是对10 000、



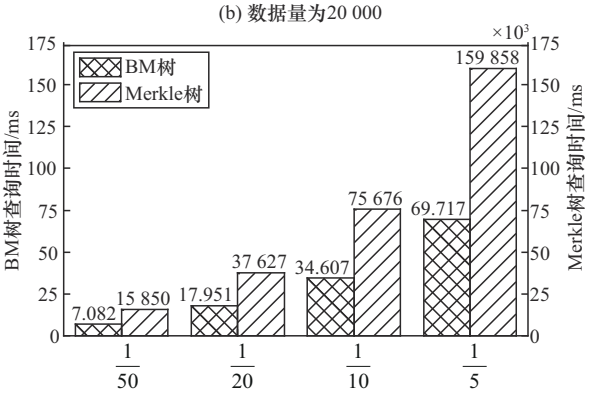
(a) 数据量为10 000



(b) 数据量为20 000



(c) 数据量为50 000



(d) 数据量为100 000

图8　BM树与Merkle树查询时间开销对比

20 000、50 000和100 000个数据进行查询的时间开销对比，图中横坐标分别是对当前数据总量的$\frac{1}{50}$、$\frac{1}{20}$、$\frac{1}{10}$和$\frac{1}{5}$数据量（因为查询一次时间过小，且查询一次时间不稳定，所以针对不同数据量都选取相同规模的查询次数，且得出的数据都是经过多次查询所求得的均值）进行查找，纵坐标为查询时间。因查询时间差距较大，图中采用双Y轴来表示。由图8可知，本文方案在查询时的时间开销远低于传统方案，并且随着总体数据量查询数据次数的增多，优势也随之增大。这是因为本文采用BM树查询，只需经过等于树高的次数便可以查询到想要的数据，而传统方案是需要遍历所有数据，直到查询到所需数据，所以查询数据时间或长或短，而随着数据量及查询次数的增大，两者的差距也越来越大。

为突出本文方案优势，对文献[8]的并行搜索算法方案进行实现并与本文方案进行对比分析。由于文献[8]与本文方案查询单一数据时间开销较小，故在查询时都设置为100次，且最终的数据都是经过100次实验求得的平均值。不同数据量的查询时间开销对比如表2所示。在不同数据量下，本文方案的查询时间开销远低于文献[8]。

表2　　查询时间开销对比

数据量/个	文献[8]/ms	本文方案/ms
1 000	140.227 9	1.093 6
2 000	190.170 6	1.296 4
5 000	326.063 5	1.492 7
10 000	432.537 3	1.600 4

4.2.3　DBF构造时间开销对比

本文构造DBF所采用的Hash函数为DJBX33A（Daniel J. Bernstein, times 33 with addition）算法，DBF与BF构造时间开销对比如图9所示。最后一列数据量过大导致时间开销过大，为使其他数据也有较好的可视性，对图中数据条进行截断处理。图中数据都是在BF采用$\frac{m}{n}$=10的空间，DBF采用$\frac{m}{n}$=5时，且达到同样的误判率下（4.1节已说明），在不同数据量下构造布隆过滤器所消耗的时间开销可以看出，DBF在构造时所带来的时间开销接近构造BF所带来时间开销的一半。

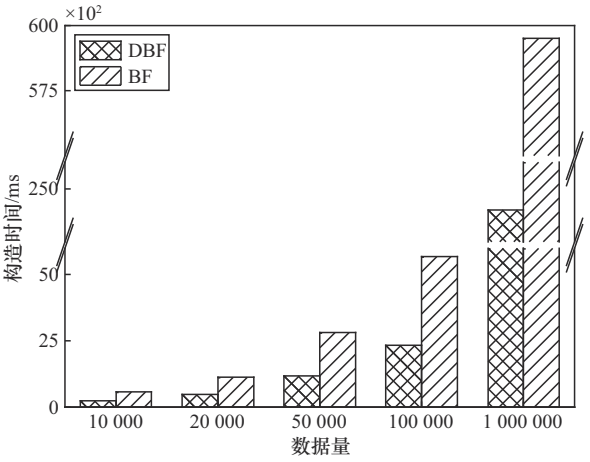


图9　DBF与BF构造时间开销对比

在查询时，BF与DBF在时间开销方面并没有明显的差距，故本文不再对比BF与DBF查询时间开销。

4.2.4　总时间开销对比

本文在构造BM树时，由于需要构造平衡树，故带来大量时间开销，为减少构造时间开销，采用分段式方法构造BM树，具体构造时间对比如图10所示。

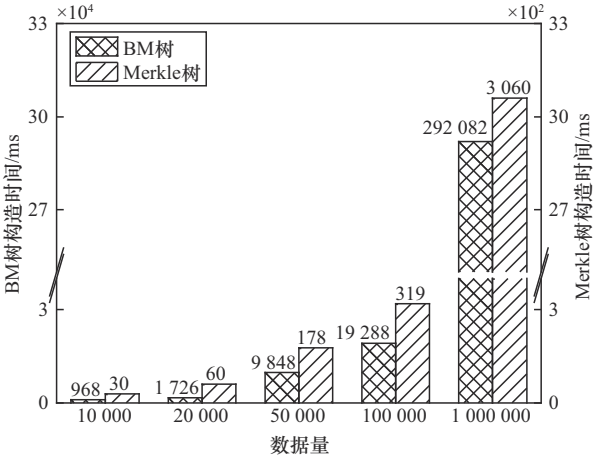


图10　BM树与Merkle树构造时间开销对比

虽在前期消耗了较多的时间开销，但与后续多次查询所带来的时间开销相比，在可接受范围内。

本文方案与传统方案的总时间开销对比（总时间开销是在不同数据量下BM树与Merkle树的构造时间与查询时间之和）如图11所示。图11(a)是在10 000个数据的情况下对不同数据量的总时间开销对比，在两线交点之前，本文方案总时间开销高于传统方案是因为在构造BM树时带来的时间开销

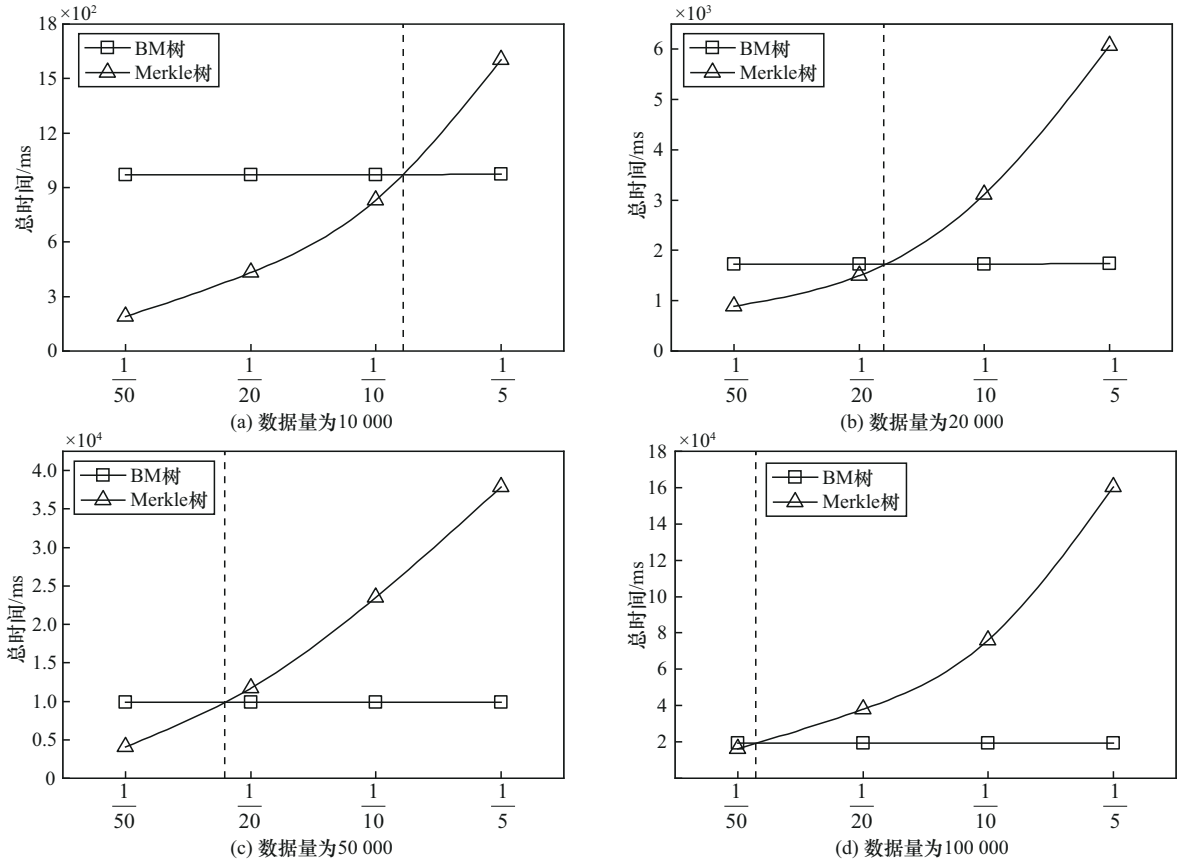


图11　总时间开销对比

相较构造 Merkle 树较大，且 Merkle 树针对数据量较小及查询次数较小的情况下，查询所带来的时间开销还是较小的，而两线交点之后则是由于随着查询次数变多的情况下，Merkle 树的查询时间迅速扩大，甚至远远大于 BM 树的构造时间，故时间开销增长得越来越快。图 11(b)、图 11(c)和图 11(d)分别为20 000个、50 000个和100 000万个数据的情况下对不同数据量的总时间开销对比。可知，图 11(b)~图 11(d)的趋势与图 11(a)的趋势保持一致，但交点随着数据量的变大向前移动。在图 11 中，本文方案总时间开销趋于稳定是因为查询过程的时间开销较小，所以总时间开销趋于稳定，而传统方案时间开销增长得越来越快则是因为随着数据量的增大，查找时间随之增大。若在溯源过程中是针对某一数据的查询，在只针对查询的情况下，无论数据量多与少（由 4.2.2 节可知），本文方案的优势大大优于传统方案。

5　结束语

本文提出了一种基于 Merkle 树的溯源方法，其中包括 BM 树的构建和查询方法。该方法能够高效地在 BM 树中查询所需数据，并对查询结果进行验证，以确保数据未被篡改。针对在查询前判断 BM 树中是否存在结果的布隆过滤器，本文提出了一种 DBF。在保持与传统 BF 相同空间占用和误判率不高于传统 BF 的前提下，该结构显著提升了构造效率，从而提高了整体溯源过程的效率。本文对 BM 树及 DBF 的实现进行了详细描述，并在不同数据量的情况下分别测试了其构造时间和查询时间。实验结果表明，该方法在提高溯源效率方面具有显著优势。进一步优化 IPFS 的数据分片与加密存储机制，结合更高效的加密算法，确保数据在存储和传输过程中的安全性，同时探索动态分片策略，深入研究基于访问频率的缓存优化机制，动态调整 IPFS 节点的缓存分布，减少数据检索时延，提升系统响应速度，以应对工业 OT 网络中不断变化的数据量和安全需求。同时，完善相关工作比较，并通过实验验证本文方案在实际工业 OT 网络中的性能表现，是下一步研究的重点。

参考文献：

[1] TIAN F, WU Z Q, GUI X L, et al. Fine-grained query authorization with integrity verification over encrypted spatial data in cloud storage[J]. IEEE Transactions on Cloud Computing, 2022, 10(3): 1831-1847.

[2] ZHUANG C X, DAI Q Y, ZHANG Y. BCPPT: a blockchain-based privacy-preserving and traceability identity management scheme for intellectual property[J]. Peer-to-Peer Networking and Applications, 2022, 15(1): 724-738.

[3] 陈飞, 叶春明, 陈涛. 基于区块链的食品溯源系统设计[J]. 计算机工程与应用, 2021, 57(2): 60-69.
CHEN F, YE C M, CHEN T. Design of food traceability system based on blockchain[J]. Computer Engineering and Applications, 2021, 57(2): 60-69.

[4] CENTOBELLI P, CERCHIONE R, VECCHIO P D, et al. Blockchain technology for bridging trust, traceability and transparency in circular supply chain[J]. Information & Management, 2022, 59(7): 103508.

[5] LAI C Z, WANG Y Z. Achieving efficient and secure query in blockchain-based traceability systems[C]//Proceedings of the 2022 19th Annual International Conference on Privacy, Security & Trust (PST). Piscataway: IEEE Press, 2022: 1-5.

[6] YAO Q, ZHANG H J. Improving agricultural product traceability using blockchain[J]. Sensors, 2022, 22(9): 3388.

[7] LIU C Y, WANG Z H, XIONG A, et al. Research on industrial Internet traceability technology based on blockchain[C]//Proceedings of the 2022 IEEE 14th International Conference on Advanced Infocomm Technology (ICAIT). Piscataway: IEEE Press, 2022: 286-291.

[8] WU H Q, JIANG S, CAO J N. High-efficiency blockchain-based supply chain traceability[J]. IEEE Transactions on Intelligent Transportation Systems, 2023, 24(4): 3748-3758.

[9] DU P T, LIU Y J, LI Y, et al. EthMB+: a tamper-proof data query model based on B+ tree and Merkle tree[M]//Blockchain Technology and Application. Singapore: Springer Nature Singapore, 2022: 49-59.

[10] ZHANG X F, LING L. A review of blockchain solutions in supply chain traceability[J]. Tsinghua Science and Technology, 2023, 28(3): 500-510.

[11] LIU Z P, REN L L, FENG Y J, et al. Data integrity audit scheme based on quad Merkle tree and blockchain[J]. IEEE Access, 2023, 11: 59263-59273.

[12] SONG J Z, LI Y T, LU H R, et al. An IPFS privacy storage sharing scheme based on SM4 algorithm[C]//Proceedings of the 2022 2nd International Conference on Education, Information Management and Service Science (EIMSS 2022). Dordrecht: Atlantis Press International BV, 2022: 641-649.

[13] LIU H J, LUO X B, LIU H R, et al. Merkle tree: a fundamental component of blockchains[C]//Proceedings of the 2021 International Conference on Electronic Information Engineering and Computer Science (EIECS). Piscataway: IEEE Press, 2021: 556-561.

[14] ZHAO Y, TIAN B, NIU Y R, et al. A security management and control solution of smart park based on sensor networks[J]. Sensors, 2021, 21(20): 6815.

[15] KAMBILO E K, ZGHAL H B, GUEGAN C G, et al. A blockchain-based framework for drug traceability: ChainDrugTrac[C]//Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing. New York: ACM Press, 2022: 1900-1907.

[16] LAI C, WANG Y, WANG H, et al. A blockchain-based traceability system with efficient search and query[J]. Peer-to-Peer Networking and Applications, 2023, 16(2): 675-689.

[17] DUAN P F, WANG J Y, ZHANG Y Q, et al. Policy-based chameleon hash with black-box traceability for redactable blockchain in IoT[J]. Electronics, 2023, 12(7): 1646.

[18] REVIRIEGO P, SÁNCHEZ-MACIAN A, WALZER S, et al. On the privacy of counting bloom filters[J]. IEEE Transactions on Dependable and Secure Computing, 2023, 20(2): 1488-1499.

[19] REVIRIEGO P, SÁNCHEZ-MACIÁN A, ROTTENSTREICH O, et al. Adaptive one memory access bloom filters[J]. IEEE Transactions on Network and Service Management, 2022, 19(2): 848-859.

[20] DUONG Q M, NGUYEN K S, NGUYEN H D, et al. FeCBF: a novel sub-optimal cascaded bloom filter structure based on feature extraction[J]. IEEE Access, 2024, 12: 67619-67631.

[21] MIAO Y B, YANG Y T, LI X H, et al. Efficient privacy-preserving spatial range query over outsourced encrypted data[J]. IEEE Transactions on Information Forensics and Security, 2023, 18: 3921-3933.

[22] 徐松松, 过晓冰, 徐恪. 可验证布隆过滤器: 加速区块链中的不存在查询与证明[J]. 中国科学: 信息科学, 2023, 53(12): 2386-2405.
XU S S, GUO X B, XU K. Verifiable bloom filter (VBF): accelerate the query and proof of nonexistent data in a blockchain[J]. Scientia Sinica (Informationis), 2023, 53(12): 2386-2405.

[23] FAN Z C, WEN G, HUANG Z P, et al. On the evolutionary of bloom filter false positives-an information theoretical approach to optimizing bloom filter parameters[J]. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(7): 7316-7327.

[24] ALMEIDA P S. A case for partitioned bloom filters[J]. IEEE Transactions on Computers, 2023, 72(6): 1681-1691.

[25] WU Y H, HE J T, YAN S, et al. Elastic bloom filter: deletable and expandable filter using elastic fingerprints[J]. IEEE Transactions on Computers, 2022, 71(4): 984-991.

[26] LIANG Y R, MA J F, MIAO Y B, et al. Privacy-preserving bloom filter-based keyword search over large encrypted cloud data[J]. IEEE Transactions on Computers, 2023, 72(11): 3086-3098.

[27] DOAN T V, PSARAS Y, OTT J, et al. Toward decentralized cloud storage with IPFS: opportunities, challenges, and future considerations[J]. IEEE Internet Computing, 2022, 26(6): 7-15.

[28] SAVIOUR M A, SAMIAPPAN D. IPFS based file storage access control and authentication model for secure data transfer using block chain technique[J]. Concurrency and Computation: Practice and Experience, 2023, 35(2): e7485.

[29] PAMUNGKAS A R D, HUSNA D, EKADIYANTO F A, et al. Design-

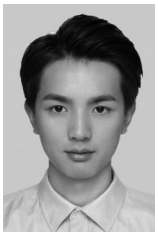
ing a blockchain data storage system using Ethereum architecture and peer-to-peer interplanetary file system (IPFS) [C]//Proceedings of the 2021 the 7th International Conference on Communication and Information Processing (ICCIP). New York: ACM Press, 2021: 152-157.

[30] MA M Y, NAKAZATO H. An application of named data network on interplanetary file system[C]//Proceedings of the 2024 IEEE 21st Consumer Communications & Networking Conference (CCNC). Piscataway: IEEE Press, 2024: 606-607.

[31] 刘炜, 张聪, 佘维, 等. 基于Merkle山脉的数据可信溯源方法[J]. 计算机应用, 2022, 42(9): 2765-2771.
LIU W, ZHANG C, SHE W, et al. Data trusted traceability method based on Merkle mountain range[J]. Journal of Computer Applications, 2022, 42(9): 2765-2771.

[作者简介]

谢鹏寿（1972– ），男，甘肃清水人，兰州理工大学教授、硕士生导师，主要研究方向为隐私保护、车联网安全、工业互联网安全等。

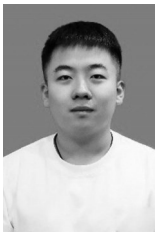


冉玉翔（1999– ），男，甘肃永昌人，兰州理工大学硕士生，主要研究方向为网络与信息安全、工业互联网安全。

冯涛（1970– ），男，甘肃临洮人，兰州理工大学研究员、博士生导师，主要研究方向为网络与信息安全、区块链、工业互联网安全等。

康永平（1970– ），女，甘肃永登人，兰州理工大学副教授，主要研究方向为生产设备故障诊断、工业互联网安全等。

杨兴慧（1998– ），男，甘肃张掖人，兰州理工大学硕士生，主要研究方向为网络与信息安全、工业互联网安全。



杨超（2000– ），男，山西忻州人，兰州理工大学大学硕士生，主要研究方向为网络与信息安全、工业互联网安全。