doi:10.3969/j.issn.1673-9833.2025.02.006

基于区块链和压缩前缀树的去中心化域名系统

陈大鹏，肖满生

（湖南工业大学 计算机学院，湖南 株洲 412007）

摘 要：现存的基于区块链的去中心化域名方案多存在去中心化不彻底、时间复杂度高、区块链存储膨胀等问题，为此提出了一个兼顾时空复杂度和去中心化程度的域名方案。首先，借助区块链实现去中心化，使用对等节点充当域名服务器，不另设特殊节点，实现彻底的去中心化；其次，设计了一种基于压缩前缀树的数据结构（DNT）以改善区块链的存储膨胀问题和降低系统的时间复杂度，并针对 DNT 提出了一种高效检索算法；最后，提出了一种基于非对称密钥的零集中管理机制用于系统在去中心环境下运营。实验结果表明：DNT 在改善存储膨胀问题上有着良好的效果，总体空间占用约为每千万条 0.87 GB，仅为同类方案的 10%；DNT 对节点总数和深度都有出色的抑制效果，节点总数比前缀树（Trie）少了 2 个数量级；吞吐率几乎不会随着区块高度的增加而衰减。

关键词：域名系统；区块链；去中心；时空复杂度；零集中管理

中图分类号：TP393.4 **文献标志码**：A **文章编号**：1673-9833(2025)02-0034-08

引文格式：陈大鹏，肖满生．基于区块链和压缩前缀树的去中心化域名系统 [J]. 湖南工业大学学报，2025，39(2)：34-41.

A Decentralized Domain Name System Based on Blockchain and Compressed Prefix Trees

CHEN Dapeng，XIAO Mansheng

（College of Computer Science，Hunan University of Technology，Zhuzhou Hunan 412007，China）

Abstract：Due to the fact that the current decentralized domain name solutions based on blockchain are often characterized with such flaws as incomplete decentralization, high-time complexity, and blockchain storage expansion, a domain name solution, which takes into consideration the spatiotemporal complexity and the decentralization level, has thus been proposed. Firstly, with the help of blockchain, a complete decentralization can be achieved by using peer nodes as domain name servers without setting up special nodes. Secondly, a data structure based on compressed prefix tree (DNT) is designed to cope with the storage inflation of blockchain and reduce the time complexity of the system, with an efficient retrieval algorithm proposed for DNT. Finally, a zero centralized management mechanism, which is based on asymmetric keys, is proposed for the operation of the system in a decentralized environment. The experiment shows that DNT is characterized with a good effect on storage expansion improvement, with an overall space occupation of about 0.87 GB per million, which is only 10% of similar solutions. DNT exhibits an excellent suppression effect on both the total number and depth of nodes, with a reduction of 2 orders of magnitude in the total number of nodes compared to prefix trees (Trie), while the throughput rate hardly decreases with the increase of block height.

Keywords：domain name system (DNS)；blockchain；decentralization；spatiotemporal complexity；decentralized management

收稿日期：2023-10-10

基金项目：湖南省自然科学基金资助项目（2024JJ8055）

作者简介：陈大鹏，男，湖南工业大学硕士生，主要研究方向为区块链，网络安全，计算机图像处理，E-mail：M21077500005@stu.hut.edu.cn

通信作者：肖满生，男，湖南工业大学教授，主要研究方向为智能计算，大数据处理与区块链，E-mail：349407041@qq.com

0　引言

域名系统（domain name system，DNS）负责把人类可读的域名翻译成机器可读的 IP 地址，是一个典型的集中树形结构。树形结构带来一定的问题，如单点故障（single point of failure，SPOF）、低容错性、分布式拒绝服务攻击 [1−2]、网络主权不平等 [3−4] 等。为应对这些挑战，研究者提出了一系列去中心化方案。然而，这些方案很难兼顾去中心化程度和系统效率之间的关系。Namecoin[5]、Bitcoin name service（BNS）[6]、Ethereum name service（ENS）[7]、B-DNS[8]、BlockZone[9] 等都专注于构建一个彻底的去中心化系统而忽视了时空复杂度带来的影响。Namecoin 是比特币 [10] 网络的一个分叉，它直接把信息记录在每一笔交易中，虽实现了彻底的去中心化，消除了 SPOF 风险，但是每次检索都要遍历所有区块，吞吐率只有 7 tps 左右。BNS 是对 Namecoin 的一种改进，它分离了控制平面和数据平面，提高了系统的灵活性，但还是没有重视效率的提升。ENS 是一个基于以太坊的域名系统，通过以太坊的智能合约实现人类可读标识的绑定，具有非常强的抗审查能力，但是以太坊的吞吐率只有 25 tps 左右，且以太坊归档节点的空间占用已经到达 TB 级别。B-DNS 没有使用现有的区块链系统，而是构建了专属的区块链网络。在 B-DNS 的设计中，只有轻节点和完全节点，是一种彻底的去中心化设计。但在时空复杂度方面，B-DNS 没有充分兼顾。它直接将域名取哈希值，然后利用所有哈希值构建一颗二分查找树，树节点中存放相应的值。二分查找在一定程度上能降低时间复杂度，但是也存在一些问题：1）二分查找树不能控制树的深度，有可能出现长单链结构，大幅影响效率；2）二分查找树的节点数量和域名条目数同步增长，达到上亿个，树的规模过于庞大，最终还是会影响效率。BlockZone 把各级名称服务器组织在一起形成一个对等网络，每个名称服务器都把自己的记录写入区块链，同时消除了树形结构，但查询时需要往前遍历整个区块链。

与此同时，一部分研究人员开始牺牲去中心化特性来换取效率。TD-ROOT[11]、TLDChain[12] 和 DNSLedger[13] 都是基于联盟链的模式，或多或少设置了一些特殊功能节点，以此提升性能。TD-ROOT 在根服务器之间建立了一个区块链联盟链网络，并可以容忍 1/3 的恶意根。服务器节点。TD-ROOT 作用只限于根服务器，下游仍然保持中心结构。在效率得到提升的同时，也保留了单点故障等待风险；TLDChain 把所有的顶级服务器（top level domain，TLD）组成一个联盟链，消除了根服务器，并使用数据仓库来加快查询速度。TLDChain 加入了 DSC（domain name service center）和 TSC（TLD service centers）等特殊节点，DSC 负责区块链节点的权限服务，防止域名冲突，TSC 负责对域名注册信息进行审核。这些特殊节点的加入，违背了去中心化原则。DNSLedger 提出了双链结构，所有 TLD 组成一个唯一的根链，而每个 TLD 链可根据需求自定义，这种自定义 TLD 链的设计在一定程度上增加了灵活性和吞吐量。但不论是根链还是 TLD 链，都有联盟链残留的特殊节点，违背去中心化原则的同时，还存在单点故障、DDoS 等问题。

综上所述，现存方案往往不能兼顾去中心化和效率两个方面。本文就此提出一种彻底去中心化且高效的域名系统（flat DNS，F-DNS）。F-DNS 通过区块链实现了域名系统的去中心化，解决了传统域名系统中依赖中心化带来的安全及信任问题。这一系统的设计为未来网络基础设施的优化提供了新思路，推动了更加公平、透明和高效的互联网架构的构建。F-DNS 的实施不仅有助于提升域名解析的可靠性与可扩展性，还可能在广泛的应用场景中促进去中心化技术的发展与应用，为互联网治理模式带来积极意义。

1　F-DNS 去中心化架构

F-DNS 的目标是实现一个彻底去中心的、高效的域名解析系统。F-DNS 利用区块链作为基础设施，实现一个完全扁平的网络结构，网络中不再设任何其他特殊角色。所有对等节点之间通过交流维护一套公共数据库，对等节点充当了去中心场景下的域名服务器。在存储方面，F-DNS 没有过度依赖区块链本身的存储功能，而是使用了 DNT（domain name tree）数据结构用于高效存储和检索域名条目，区块链的区块里只记录一些少量且关键的信息，并通过 DNT 根节点的内容标识符（content identifier，CID）与实际存储数据相联系。整体架构如图 1 所示，F-DNS 可划分为 3 个层次：网络层、区块层、数据层。

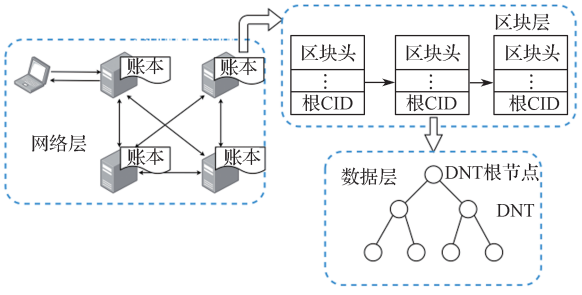


图 1　F-DNS 整体架构

Fig. 1　F-DNS overall architecture of

1.1 网络层

网络层由世界各地的对等节点和客户端组成，客户端是发起域名解析请求的终端，对等节点则是F-DNS 提供服务的实体。F-DNS 在保持功能完整性的同时，没有保留诸如权限审查等特殊角色单位[12]。网络拓扑是扁平、松散、动态变化的。网络层中的对等节点的实际运营方可以是企业、学校、政府机构，它的所在地并不能影响它的工作方式。用户可以选择一个离自己较近的对等节点作为 DNS 接入点，解析请求无需经过多次迭代，在接入点即可完成。“一步到位”的服务方式缩短了解析路径，提升了应答速度，减少了请求被劫持或遭遇缓存中毒的风险。

网络层依托区块层的实用拜占庭容错（practical byzantine fault tolerance，PBFT）共识算法维持数据层数据的一致性。通过该机制，F-DNS 实现了去中心情况下公共账本的强一致性。PBFT 算法能确保在网络中存在一定数量恶意节点的情况下达成一致，这意味着 F-DNS 可在面对恶意行为，如尝试破坏解析结果一致性、中断通信或延长关键消息等，仍能保持域名解析的准确性和可靠性。通过 PBFT 算法，F-DNS 可容忍最多 f 个恶意节点（其中 $f < n/3$，n 为参与网络总节点数），确保系统的可靠性和安全性。

在 PBFT 算法中，主要分为 6 个阶段：请求、转发、预准备、准备、提交和响应，如图 2 所示。

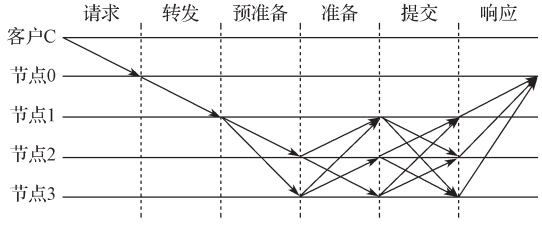


图 2 PBFT 算法的共识过程

Fig. 2 Consensus process of PBFT algorithm

首先，网络层中的客户端 C 向任意一个对等节点 0 发起请求，节点 0 把请求转发给主节点 1，主节点 1 收到请求后进入预准备阶段并请求分配一个递增序列号 n，然后将当前视图版本 v、序号 n 和请求哈希值 d 发送给从节点 2 和 3。这里的主从节点并非系统预设特殊角色，而是根据 PBFT 算法运行机制而定。每个节点都可能成为主节点，当前主节点根据视图版本 v 确定。在从节点收到主节点 1 的预准备消息后，会先验证数字签名、v、n、d 等基本信息，若未通过验证则直接丢弃该消息，若通过验证则进入准备阶段。准备阶段的目的在于确保集群中有足量的节点也收到了该请求。此阶段中，节点既会广播消息，也会接受别人广播的消息。准备阶段的广播消息包括节点的视图版本 v、序列号 n、请求哈希值 d 以及节点编号 i，若收到的准备消息的 v、n、d 和预准备消息的一样则称两者匹配。当收到 $2f$ 个匹配的准备消息后即可进入提交阶段。在提交阶段，节点还不能直接响应该请求，还要明确集群中是否有足量节点让该请求通过了准备阶段。提交消息同样包含 v、n、d 等内容，在收到 $2f$ 个匹配的提交消息后，节点便认为已有足量节点确认了该消息，主从节点便会执行请求，并进入最后响应阶段，把结果给节点 0。

1.2 区块层

区块层位于网络层之下，它表示对等节点间维护的一个公共数据库，即区块链公共账本。形式上，它是一个以区块为单位的单向链表，区块中记录了时间戳、DNT 根节点 CID 和前一区块哈希值，结构见图 3。

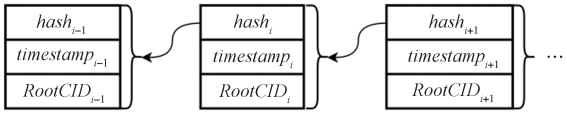


图 3 区块层的区块链表

Fig. 3 Blockchain linked list of the block layer

当前区块中记录着前一个区块的哈希值，而当前区块又被计算出哈希值记录在下一区块之中，哈希计算如式（1）所示。这样，每一个区块不止受到前一个区块的影响，而是受到之前所有区块的共同影响，对之前区块的任何修改都不能通过哈希校验。

$$hash_i = sha256(hash_{i-1}+timestamp_{i-1}+RootCID_{i-1}) \quad (1)$$

区块中还记录了 DNT 根节点的 CID，通过根节点 CID 可以索引整棵 DNT。考虑到 DNS 场景下不必保存历史记录，故在区块中不会记录每条交易。这种设计一方面可以进一步优化空间存储，另一方面可以避免区块回溯，链上检索时间复杂度永远是 $O(1)$。

1.3 数据层

数据层位于整个系统最底层，它主要负责 DNT 数据的实际存储。在数据层中，一个 DNT 节点对应一个星际文件系统（inter planetary file system，IFPS）[14]文件。同时，一个 DNT 节点可以供多个域名条目复用，在 DNT 中体现为多个子节点共用一个父节点。DNT 中的父子关系是通过 IPFS 可以利用 CID 寻址的特性建立的，父节点中都会记录子节点的 CID。

由于父节点包含子节点的 CID，而 CID 又是基于散列函数计算出来的，所以当一个节点的内容发生变动时，它本身的 CID 也会随之改变，从而导致它的父节点需更新子节点的 CID，进而也改变了父节点本身的 CID，这种连锁反应会一直延续到根节点。DNT 更新过程，其实是发生一系列连锁反应并产生一个新 DNT 根节点的过程，期间那些因连锁反应而游离的节点最终会被释放，整个过程如图 4 所示。虚

线左边为原 DNT，右边为更新后的 DNT。当 c 节点发生修改时，b 节点需更新相应的子节点 CID，从而导致 b 节点本身的 CID 变化，进而 a 也需更新对应的子节点 CID。最终产生了 3 个新的 DNT 节点 A、B、C。剩余节点没有变动，继续成为新 DNT 的组成部分，而 a、b、c 节点因游离而最终会被释放。

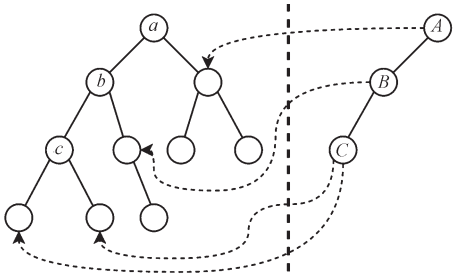


图 4　DNT 数据层更新的连锁反应

Fig. 4　Chain reaction of DNT data layer updates

2　时空复杂度优化

区块链具有公共账本功能以实现全局统一数据库，但直接利用区块记录 DNS 条目会导致时空复杂度异常高。为此本文提出了 DNT 数据结构作为链下保存数据的载体，在时间和空间两方面进行优化。

域名条目数据有两大特点：首先，数据为 key-value 形式，key 为域名，value 为 IP 地址；其次，key 存在一定规律性，即从后向前逐字匹配可找到重叠部分，见图 5，这是 DNT 节点复用的基础。DNT 以域名为 key 组织成树，不同域名之间重叠部分共享存储节点，只有不匹配时树才会分叉。这种设计不但可充分复用存储节点，还可把区块的线性遍历转为树搜索，时间复杂度降至 $O(\log M)$，M 为域名长度。

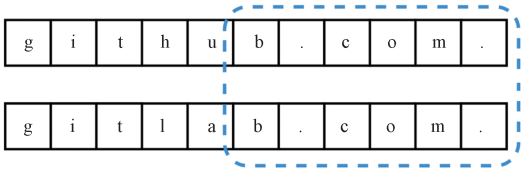


图 5　域名的重叠

Fig. 5　Domain name over lapping

2.1　DNT 设计

一个 DNT 节点由若干个分支（branch string）字段和一个重叠字段（overlapping string）组成。重叠字段代表复用该节点所有域名的重叠部分，DNT 借助了压缩前缀树（patricia tree）的思想，把上下相邻的节点尽可能合并成一个，故重叠部分可能有多个字符。分支字段表示 DNT 节点上的一个分叉，一个节点可以有不同的分叉方向。

因每个节点都只一个父节点，故多个域名复用了一个父节点也会同时复用该父节点的所有祖先节点。域名的 IP 地址最后会被包含进分支字段中，故 IP 地址对应的完整域名体现在树搜索路径中。DNT 示例见图 6，图中共 3 个域名条目：“github.com”“gitlab.com”“google.com”。虚线框为重叠部分，实线框为分支字段，域名对应 IP 记录在对应分支字段中。

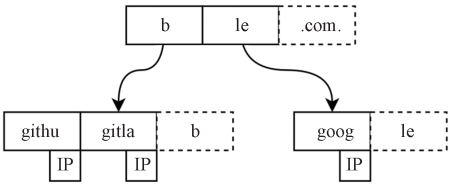


图 6　DNT 示例

Fig. 6　DNT illustration

DNT 分支字段结构如图 7 所示，设置 3 个标志位，分别为 ValueExist、SuccessorsExist、TopDomain。当 ValueExist 置 1 时，表示该分支还包含一个解析结果。SuccessorsExist 置 1 表示该分支有后继，此时分支字段还应包含后继的 CID 以便找到该后继。只有当一个节点的所有分支都没有后继时，它才是一个叶子节点。TopDomain 置 1 表示 Public Key 包含一些与顶级域验证相关的公钥。

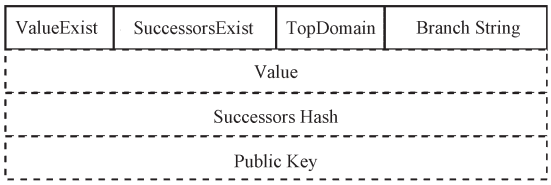


图 7　DNT 分支字段结构

Fig. 7　DNT branch field structure

2.2　DNT 优化检索算法

DNT 检索算法是 DNT 适应去中心环境下的配套检索算法，是一个逐步从上往下迭代的过程，时间复杂度为 $O(\log M)$，待检索域名长度 M 与数据总量无关。迭代遵循以下步骤：对于一给定字符串 s 和节点指针 p，先令 s 减去 p 所指节点重叠部分，即 $s=s-OverlappingString_p$。然后令 s 分别与各分支字段进行匹配并产生匹配结果。匹配结果可分为以下几种：（i 为分支字段字符下标，j 为分支字段序号）

1）完全匹配（ExactMatch）。

$$\begin{cases} s[i]=BranchString_p^j[i], \\ 0\leqslant i<\mathrm{len}(s)=\mathrm{len}(BranchString_p^j)。 \end{cases} \tag{2}$$

分支字段与 s 完全相同。

2）完全不匹配（MisMatch）。

$$s[0]\neq BranchString_p^j[0]\ 。 \tag{3}$$

分支字段与 s 从第一字符就开始分叉。

3）过盈匹配（ExcessMatch）。

$$\begin{cases} s[i]=BranchString_p^j[i], \\ s[i+1]\neq BranchString_p^j[i+1], \\ 0\leqslant i<\mathrm{len}(BranchString_p^j), \\ \mathrm{len}(BranchString_p^j)<\mathrm{len}(s)。 \end{cases} \tag{4}$$

分支字段为 s 的真子串。

4）过短匹配（DeficientMatch）。

$$\begin{cases} s[i]=BranchString_p^j[i], \\ s[i+1]\neq BranchString_p^j[i+1], \\ 0\leqslant i<\mathrm{len}(s), \\ \mathrm{len}(s)<\mathrm{len}(BranchString_p^j)。 \end{cases} \tag{5}$$

与过盈匹配正好相反，s 为分支字段的真子串。

5）部分匹配（PartialMatch）。

$$\begin{cases} s[i]=BranchString_p^j[i], \\ s[i+1]\neq BranchString_p^j[i+1], \\ 0\leqslant i<\min\left\{\mathrm{len}(s),\mathrm{len}(BranchString_p^j)\right\}-1。 \end{cases} \tag{6}$$

分支字段与 s 之间只有后面部分重叠。

最后，根据匹配结果集决定下一步是继续迭代还是返回。在一次迭代过程中，所有分支的匹配结果集满足：1）所有匹配结果均在以上 5 种情况内；2）部分匹配、恰好匹配、过盈匹配、过短匹配之间互斥，即结果集中不能同时出现这 4 者的 2 个及以上（包括重复）；3）完全不匹配与所有结果相兼容，包括本身。

结合上述，一个完整流程的伪代码如算法 1 所示。

算法 1 DNT 域名检索算法。

输入 待检索域名 s，DNT 的根节点 p。

输出 s 在 DNT 中的检索结果。

```
while true do
    s = s − OverlappingString_p
    num = GetBranchNum(p)
for i = 0; i< num; i ++
    r = Match(Branch_i, s)
 if r not in(ExcessMatch, ExactMatch)
        continue
      else if r ==ExcessMatch
        if Branch_i has successor
            p = p → next
            recursive call
        else
          contiune
        end
      else if r ==ExactMatch
        if Branch_i has value
          return value
        else
          contiune
  return −1
```

3 基于密钥的子域名托管机制

以上讨论的 DNT 结构及其优化算法主要解决了系统存储和检索的效率问题，而一个去中心的域名系统还需要配套的零集中运维和管理机制，如域名持有者身份识别、子域名托管等。在现行 DNS 中，一个域名管理者要新增一个子域名，可以添加一条 NS 记录把这个子域名和一个主机或者 IP 地址绑定，用这条记录来表示托管关系。在去中心环境下，对等节点都是动态变化的，没有固定的 IP 地址，这就需要一种机制能将子域名托管关系与 IP 地址解耦的同时还不能打破零集中管理这一原则。

3.1 基于密钥的子域名托管机制设计

基于密钥的子域名托管机制提出是为了适应去中心环境下域名系统的零集中管理需求。在此机制中，每一层级的域名都有一对身份密钥以证明对该级域名的所有权，身份密钥需经过上一层级身份密钥的签名批准。在计划新建一个子域名并将其托管出去时，父级域名需为子域名构建一个托管证书。托管证书内容为子域名名称、子域身份密钥的公钥以及过期时间等信息，这些内容由父级的身份密钥签名，如图 8 所示，com. 域计划新建一个子域名“example.com”，“com.”需把 3 个材料一起用自己的私钥签名后封装成托管证书。托管证书证明了域的身份密钥的合法性，从而使得该域可以顺利履行各项职能。

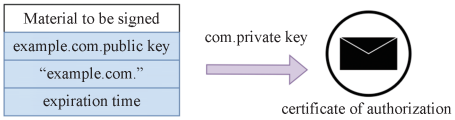


图 8 委托证书的生成

Fig. 8 Generation of delegation certificate

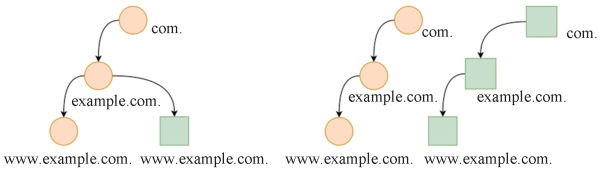
托管证书和身份密钥的生成由域组织内部自行协商管理，不用记录在区块上，也不用记录在 DNT 上，除了身份密钥的私钥以外都是可以公开的。当一个域准备履行自己的职能时，应当用自己的身份密钥对交易进行签名，然后准备一些证明材料以证明自己的合法身份，其中包括自己和上面各级域的公钥和托管证书，直到二级域为止。顶级域没有托管证书，它身份密钥的合法性需要使用 DNT 来验证，它的身份密钥被记录在 DNT 分支中的 Public Key 字段中。

身份的验证也是该机制中的重要一环，当对等节点收到交易后，需要验证其合法性。验证主要分两部分：第一部分是验证交易的合法性，用交易发起域提供的公钥即可验证；第二部分是验证发起域身份密钥的合法性，这一部分由发起域的托管证书验证。由于发起域的托管证书是被递归授权的，所以也应该递归

向上验证各个层级的托管证书合法性，直到二级域为止。二级域的托管证书验证需要用到顶级域的公钥，而顶级域的公钥记录了在 DNT 中，当验证完二级域的托管证书后，便完成了验证链条的所有验证工作。

3.2　私钥泄露处理

在基于密钥的子域名委托机制的使用过程中，由于无法假设私钥不会泄露，而数字签名又具有不可撤销性的特性，所以已签发的托管证书只有在过期的情况下才会失效，故有必要妥善处理泄露的私钥。当某一个域的私钥泄露时，不能只更新验证链条中的部分身份密钥，而是需要递归地更新整个验证链条上的身份密钥。因为攻击者可以通过收集旧的证书和公钥继续形成一条完整的验证链条，如图 9a 所示，圆形节点表示旧的身份密钥，矩形节点表示新的身份密钥，箭头表示密钥之间的授权签名关系。当“www.example.com”私钥泄露时，在仅更新验证链条上的“www.example.com”的身份密钥时，该域原先的身份密钥只要没有过期就依然可以通过最终验证。只有如图 9b 把整个验证链条的身份密钥都更新了，泄露的密钥才不能通过最终验证。对于那些不在验证链条上，但父级域又更新了身份密钥的域，只需用父级域用新的身份密钥重新构建托管证书即可。上述操作只有顶级域更新密钥时涉及上链操作，其余的密钥更新、托管授权等操作均在组织内部自行解决，故私钥泄露处理的速度基本不会受到区块链系统的限制。为了提高安全性，即使私钥没有泄露，组织内部也可定期更新密钥。



a）仅更新部分身份密钥　　b）更新验证链条全部密钥

图 9　私钥泄露处理

Fig. 9　Private key disclosure handling

4　实验

为了验证 F-DNS 的性能，针对时空复杂度和吞吐率做了定量分析及对比实验，其中包括 DNT 深度、空间占用、节点的平均复用次数、节点吞吐率等。数据集为一个 CAIDA 公开数据集 [15] 的子集，用于模拟真实的数据环境，并在 Linux 环境下用 Hyperleder Fabric 框架搭建了一个 20 个对等节点的区块链网络。具体的实验环境如下：处理器为 i5-11300H @ 3.10 GHz，内存为 16 GB，磁盘为 1 TB，系统版本为 Ubuntu 22.04，Hyperleder Fabric 版本为 2.2，Golang 版本为 18.0，docker 版本为 20.10.18，IPFS 版本为 v0.18.0，Protobuf 版本为 v1.29.0。

4.1　存储节点复用次数分析

节点复用是 F-DNS 能降低空间复杂度的基础，这节对 F-DNS 的节点复用能力做了一个定量分析。节点复用定义为多个域名共用一个 DNT 节点，而节点复用次数在这里定义为一个 DNT 节点被复用的域名总数。实验逐步增加域名条目数 1×10^8，并记录了 DNT 前 5 个层级的平均复用次数，结果见图 10。

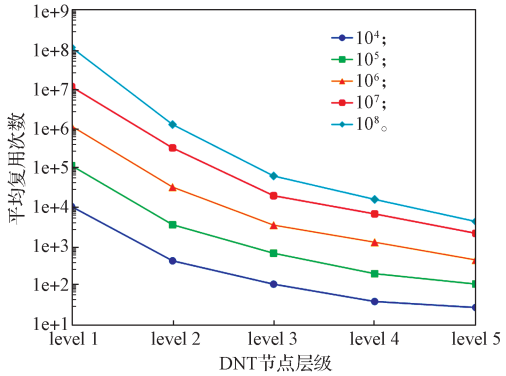


图 10　DNT 平均节点复用次数

Fig. 10　DNT average node reuse times

从图 10 中可发现，当条目数不断增加时，5 个层级的平均节点复用次数均稳定提升，这表明大数据量的场景下更有利于 DNT 复用性能的发挥。在同一数据量情况下，节点深度越浅，复用次数越高。一级节点复用次数达到了条目总数，也即所有域名条目都复用了一级节点。虽然深度会降低复用次数，但是深层节点的复用次数也维持了一个较高水平，在 1×10^8 条目的情况下，5 级深度的节点平均复用次数也到达了 10^4 的量级以上。除此之外，DNT 在深度抑制方面也有很出色的性能，这一点可以缓解深度增加导致的复用次数下降问题，所以总体来说 DNT 的复用次数处于一个很高的水平。

4.2　树深度和节点总数分析

树的深度可以影响检索的时间复杂度，节点总数直接反映了树的规模以及存储空间占用。本节测试了域名条目数对深度和节点总数的影响，并以其他几个方案作为对照，结果如图 11 和 12 所示。由图可知，在域名条目数达 1×10^7 时，Trie[16]、梅克尔－帕特里夏树（Merkel Patricia tree，MPT）[17] 和 B-DNS 的深度分别约为 DNT 的 6 倍、4 倍、2 倍，只有 F-DNS 和 B-DNS 深度增长曲线相对平缓，F-DNS 深度上限可以保持在 30 以下。

从图 12 可以看出，DNT 对节点的压缩效果非常明显。DNT 的最大节点数约为 4.5×10^6，相比之下 Trie 的最大值达到了约 1×10^8，两者差了两个数量级，

MPT 和 B-DNS 也远高于 F-DNS。

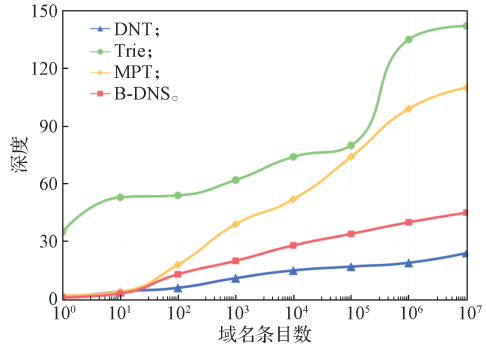


图 11 DNT、Trie、MPT 以及 B-DNS 的深度对比

Fig. 11 Deep comparison of DNT,Trie,MPT and B-DNS

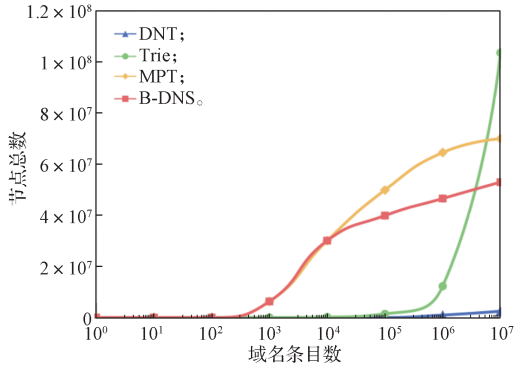


图 12 DNT、Trie、MPT 以及 B-DNS 节点总数对比

Fig. 12 Node total number comparison of DNT,Trie,MPT and B-DNS

4.3 空间占用分析

这节做了 DNT 持久化实验，把内存中构建好的 DNT 写入磁盘，持久化时使用 Protocol Buffer 把节点序列化成二进制格式。

F-DNS 空间占用与域名数量的关系见表 1。

表 1 F-DNS 占用空间

Table 1 F-DNS space occupancy

域名条目数	占用空间	域名条目数	占用空间
1×10^3	105.37 kB	1×10^6	92.05 MB
1×10^4	0.98 MB	1×10^7	0.87 GB
1×10^5	9.48 MB		

由表 1 可知，在 1×10^7 条域名的情况下，空间占用约为 0.87 GB，平均每个节点占用 204 字节，每个域名条目占用 93 字节。

表 2 反映了流行的去中心域名系统在拥有一千万条域名条目后的空间占用情况。

表 2 不同方案下的空间占用数据

Table 2 Space occupancy of different schemes

方案名称	空间占用（每千万条）/GB	方案名称	空间占用（每千万条）/GB
F-DNS	0.87	BlockZone[9]	10.00
B-DNS	281.39	BlockZone[18]	12.88

由表 2 可知，其中 B-DNS 最多，达到了 281.39 GB，其余两个也达到了 10 GB 量级，F-DNS 仅用了 0.87 GB 的硬盘空间。

4.4 吞吐率分析

在本实验条件下，对每个对等节点都做了吞吐率极限测试和吞吐率衰减测试，最后取所有节点的平均值。吞吐率分为检索吞吐率和注册吞吐率，检索吞吐率表示单位时间内能对外响应的检索请求的数量，注册吞吐率表示单位时间内能处理的注册请求数量。注册相较于检索多了共识过程，因此注册吞吐率通常较低于检索吞吐率。在测试环节，使用了一个外部脚本充当客户端来模拟大量并发请求，并手动设置了并发量。如图 13 所示，横轴表示不同的并发量，纵轴表示吞吐率。在较低并发量时，检索吞吐率和注册吞吐率都能接近并发量。但是达到一定并发量后，吞吐率增长速度逐渐减缓，最终达到吞吐率的极限值，检索和注册吞吐率分别为 1 200 tps 和 1 700 tps。

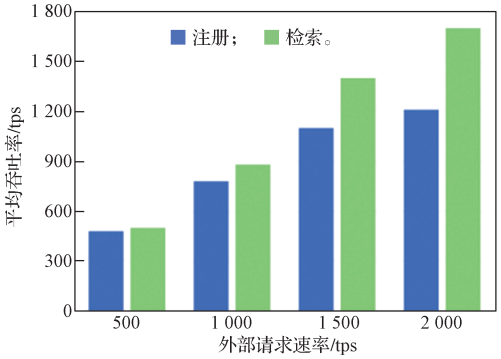


图 13 吞吐率随外部并发量的变化

Fig. 13 Throughput changes with external concurrency

在吞吐率衰减测试中，主要测试了在一定请求并发量的情况下，吞吐率随着区块高度的变化关系。如图所示 14，在请求并发量为 1 000 时，吞吐率并不会随着区块数量的增加而衰减，分别达到了 780 tps 和 900 tps 左右。

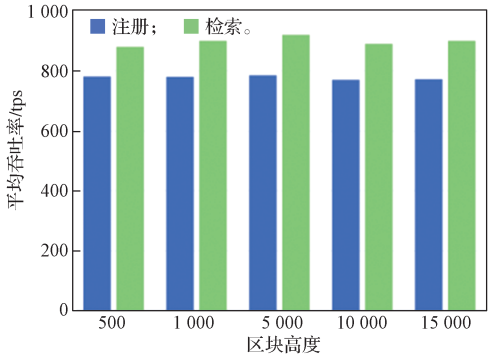


图 14 吞吐率随区块高度的变化

Fig. 14 Throughput rate changes with block height

5　结论

本文提出了一种基于区块链的、高效的去中心化域名解析方案。首先，提出了基于网络层、区块层和数据层的 3 层架构。在网络层中只有对等节点，不设特殊功能节点，实现了域名系统的彻底自治。其次，在数据层优化了数据的存储结构，利用键值的规律性充分压缩节点空间，抑制了树的节点总数和深度的增长。同时，设计了一套高效的检索算法，规避了区块链的回溯，在区块高度增加的情况下，仍可以保持极低的性能损耗。最后，设计了去中心环境下的管理机制，实现了零集中管理。实验结果表明，F-DNS 在存储效率、性能保持、检索复杂度等方面都有优异表现。不足之处是，使用区块链的域名系统对于服务发现或者企业内部网络解析等场景不是很友好。

参考文献：

[1] BURTON R. Unsupervised Learning Techniques for Malware Characterization[J]. Digital Threats：Research and Practice，2020，1(3)：1−26.

[2] FANG L，WU H B，QIAN K X，et al. A Comprehensive Analysis of DDoS Attacks Based on DNS[J]. Journal of Physics：Conference Series，2021，2024(1)：012027.

[3] 方滨兴 . 从“国家网络主权”谈基于国家联盟的自治根域名解析体系 [J]. 信息安全与通信保密，2014，12(12)：35−38.
FANG Binxing. On the Autonomous Root Domain Name Resolution System Based on National Alliance from the Perspective of “National Network Sovereignty”[J]. Information Security and Communications Privacy，2014，12(12)：35−38.

[4] 张　宇，夏重达，方滨兴，等 . 一个自主开放的互联网根域名解析体系 [J]. 信息安全学报，2017，2(4)：57−69.
ZHANG Yu，XIA Zhongda，FANG Binxing，et al. An Autonomous Open Root Resolution Architecture for Domain Name System in the Internet[J]. Journal of Cyber Security，2017，2(4)：57−69.

[5] KALODNER H A，CARLSTEN M，ELLENBOGEN P，et al. An Empirical Study of Namecoin and Lessons for Decentralized Namespace Design[J]. Workshop on the Economics of Information Security，2015：1−23.

[6] ALI M，NELSON J，SHEA R，et al. Blockstack：A Global Naming and Storage System Secured by Blockchains[C]//2016 USENIX Annual Technical Conference (USENIX ATC 16). Denver，CO：VSENIX，2016：181−194.

[7] XIA P，WANG H，YU Z，et al. Challenges in Decentralized Name Management[J/OL]. [2023−04−13]. Proceedings of the 22nd ACM Internet Measurement Conference，2022. DOI:https://doi.org/10.1145/3517745.3561469.

[8] LI Z C，GAO S，PENG Z，et al. B-DNS：A Secure and Efficient DNS Based on the Blockchain Technology[J]. IEEE Transactions on Network Science and Engineering，2021，8(2)：1674−1686.

[9] HU N，YIN S，SU S，et al. Blockzone：A Decentralized and Trustworthy Data Plane for DNS[J]. Computers，Materials & Continua，2020，65(2)：1531−1557.

[10] NAKAMOTO S. Bitcoin：A Peer-to-Peer Electronic Cash System[EB/OL]. [2023−04−03].https://bitcoin.org/bitcoin.pdf.

[11] HE G B，SU W，GAO S，et al. TD-Root：A Trustworthy Decentralized DNS Root Management Architecture Based on Permissioned Blockchain[J]. Future Generation Computer Systems，2020，102：912−924.

[12] SHEN Y T，LU Y，WANG Z L，et al. DNS Service Model Based on Permissioned Blockchain[J]. Intelligent Automation & Soft Computing，2021，27(1)：259−268.

[13] DUAN X N，YAN Z W，GENG G G，et al. DNSLedger：Decentralized and Distributed Name Resolution for Ubiquitous IoT[C]//2018 IEEE International Conference on Consumer Electronics (ICCE). Las Vegas，NV：IEEE，2018：1−3.

[14] BENET J. IPFS-Content Addressed，Versioned，P2P File System[EB/OL]. [2023−03−11]. https://github.com/ipfs/papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf.

[15] CAIDA. IPv4 Routed /24 DNS Names Dataset[EB/OL]. [2023−04−13]. https://www.caida.org/catalog/datasets/ipv4_dnsnames_dataset/.

[16] MORRISON D R. PATRICIA：Practical Algorithm To Retrieve Information Coded in Alphanumeric[J]. Journal of the ACM，1968，15(4)：514−534.

[17] HUANG Q. Ethereum：Introduction，Expectation，and Implementation[J]. Highlights in Science，Engineering and Technology，2023，41：175−182.

[18] WANG W T，HU N，LIU X. BlockZone：A Blockchain-Based DNS Storage and Retrieval Scheme[M]// Cham：Springer International Publishing，2019：155−166.

（责任编辑：姜利民）