Computers, Materials & Continua CMC, vol.65, no.2, pp.1531-1557, 2020

Blockzone: A Decentralized and Trustworthy Data Plane for DNS

Ning Hu[1], Shi Yin[1], Shen Su[1, *], Xudong Jia[1], Qiao Xiang[2] and Hao Liu[3]

Abstract: The domain name system (DNS) provides a mapping service between memorable names and numerical internet protocol addresses, and it is a critical infrastructure of the Internet. The authenticity of DNS resolution results is crucial for ensuring the accessibility of Internet services. Hundreds of supplementary specifications of protocols have been proposed to compensate for the security flaws of DNS. However, DNS security incidents still occur frequently. Although DNS is a distributed system, for a specified domain name, only authorized authoritative servers can resolve it. Other servers must obtain the resolution result through a recursive or iterative resolving procedure, which renders DNS vulnerable to various attacks, such as DNS cache poisoning and distributed denial of service (DDoS) attacks. This paper proposes a novel decentralized architecture for a DNS data plane, which is called Blockzone. First, Blockzone utilizes novel mechanisms, which include on-chain authorization and off-chain storage, to implement a decentralized and trustworthy DNS data plane. Second, in contrast to the hierarchical authentication and recursive query of traditional DNS, Blockzone implements a decentralized operation model. This model significantly increases the efficiency of domain name resolution and verification and enhances the security of DNS against DDoS and cache poisoning attacks. In addition, Blockzone is fully compatible with the traditional DNS implementation and can be incrementally deployed as a plug-in service of DNS without changing the DNS protocol or system architecture. The Blockzone scheme can also be generalized to address security issues in other areas, such as the Internet of things and edge computing.

Keywords: Network security, DNS security, DNS decentralization, blockchain.

1 Introduction

As a critical infrastructure of the Internet, the domain name system (DNS) provides Internet applications with a mapping service between memorable names and numerical internet protocol (IP) addresses. Many network functions, such as load balancing, domain keys identified mail and service blocking, also rely heavily on the DNS service.

[1] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, China.

[2] Department of Computer Science, Yale University, New Haven, 06520-8241, USA.

[3] Qianxin Technology Group Co., Ltd., Beijing, China.

[*] Corresponding Author: Shen Su. Email: sushen@gzhu.edu.cn.

Received: 09 April 2020; Accepted: 12 June 2020.

Therefore, the availability and security of DNS are vital to the Internet.

However, DNS is vulnerable to many malicious attacks, such as the 2016 Dyn cyber-attack [Zou, Zhang, Pei et al. (2016)]. In addition to directly sabotaging DNS, attackers manipulate DNS to exfiltrate data or launch distributed denial of service (DDoS) attacks. According to the International Data Corporation (IDC) 2019 Global DNS threat report, 76% of organizations have been subjected to DNS attacks in the past year [Efficientip (2019)].

The vulnerabilities of DNS can be categorized into protocol vulnerability, implementation vulnerability, and architecture vulnerability. Many systems and mechanisms have been proposed for addressing the first two types of vulnerabilities. For example, more than 200 internet engineering task force (IETF) request for comments (RFC) documents have been published to improve the protocol design, implementation and operation of DNS. Despite this large body of work, new attacks on DNS continue to arise, which pose significant threats to the Internet.

To effectively resist emerging new attacks, in this paper, we argue that DNS should be improved from an architectural perspective. To facilitate new security solutions, we advocate for the adoption of a decentralized architecture. The current architecture is vulnerable to cyber-attacks, and its architectural vulnerability has been ignored in past research. DNS lookup is a recursive procedure through the DNS hierarchy. During the resolution process, the results of domain name resolution depend completely on the root server and the authoritative server, and no other servers can determine the authenticity of the results. The vulnerability of the DNS architecture is caused by this hierarchical domain name management and fixed-point domain name resolution mechanism. In this article, we refer to this vulnerability as the centralization feature. This vulnerability is the root cause of DNS cache poisoning [Trostle, Van and Pujari (2010)]. Although domain name system security extensions (DNSSEC), which binds a digital signature with each name record and enables the end user to authenticate the resolution result, was proposed as a security enhancement for DNS, it is only deployed at top-level domain (TLD) name servers and has not replaced legacy DNS worldwide [The Internet Corporation for Assigned Names and Numbers (2020)] since the security mechanism of DNSSEC is based on public key infrastructure(PKI), which requires a unified trust anchor. If the trust anchor is managed by single organization, the single organization is a single point of failure and a dictator. For example, an organization that has control over a trust anchor may maliciously produce incorrect authentication results or prevent requests from being resolved, which can cause applications to fail to resolve domain names. Since the Internet is a worldwide infrastructure, the autonomy of domain names has attracted widespread international attention. A centralized domain name authentication solution such as DNSSEC is difficult to deploy worldwide.

In contrast, as a decentralized technology, Blockchain is widely used in multiple applications, such as intelligent data analysis [Wang, Kong, Guan et al. (2019); Zhang, Li, Wang et al. (2018); Zhou and Luo (2018)], network security [Jia, Hu, Su et al. (2020); Jiang, Liu, Yang et al. (2019); Li, Sun, Lu et al. (2020); Muhammad and Wang (2020);Tian, Luo, Qiu et al. (2019)], and the Internet of things [Tian, Gao, Su et al. (2020); Tian, Shi, Wang et al. (2019); Yin, Luo, Zhu et al. (2019)]. Researchers have also examined the leveraging of Blockchain to enable DNS decentralization. The most influential approaches include Namecoin [Kalodner, Carlsten, Ellenbogen et al. (2015)], BlockStack [Ali, Nelson, Shea et al. (2016)] and Ethereum Name Service (ENS) [Bouquet and Molinari (2013)]. These designs attempt to rebuild a new decentralized DNS that is based on a Blockchain platform. Their common features include the binding of domain names and digital currencies, the purchase of domain names through free transactions and mining, and the decentralization of domain name management via the decentralization of digital currencies. Although they realize decentralized domain name management and privacy preservation, the suppression effect for domain name spoofing attacks is not significant. First, DNS that is based on Blockchain is only deployed on the root domain name server. For example, Namecoin provides top-level domain.bit, Blockstack provides top-level domain .id, and Ethereum name services provides top-level domain.eth. Since available application systems already have fixed domain names, it is unrealistic to ask global applications to switch to new domain names. Second, the current ecological environment is not sufficiently mature for supporting these new DNSs in fully replacing the legacy DNS. Each domain name should be associated with an Internet IP address, but the current allocation mechanism of Internet IP address is far from decentralized.

In summary, we posit that the traditional DNS has a structural vulnerability that is not conducive to the collaborative monitoring and authentication verification of domain names, and it has difficulty resisting domain name spoofing attacks. Moreover, the available decentralized DNSs focus on the decentralized management and privacy protection of domain names. They are not designed to improve the security of DNS and do not fully support incremental deployment. Since the traditional DNS is closely bound to many Internet applications, it is impossible to replace it with a new DNS in a short period of time. As such, the fundamental challenge in designing a decentralized DNS architecture is the decentralization of the architecture of DNS to increase its processing efficiency and robustness while maintaining compatibility with the traditional DNS.

To address this challenge, we design Blockzone, which is a novel scheme for DNS data plane decentralization. The basic strategy of Blockzone is to change the traditional recursive resolution process of DNS by introducing new mechanisms of domain name data storage and retrieval. Blockzone stores the DNS zone files in a distributed file system, such as the Interplanetary File System (IPFS) [Benet (2014)], and saves the metadata of the DNS zone files, which include a list of the domain names in the zone file, storage addresses, and verification information, on the blockchain. When the resolution server receives a domain name resolution request, it no longer must conduct a complex recursive process. Instead, it locates the metadata of the zone file through the Blockchain client, accesses the DNS zone file according to the address in the metadata, and obtains the resource record of the target domain name. Compared with the traditional DNS resolution process, our scheme not only realizes higher retrieval efficiency but also guarantees the authenticity of the resolution results. Since it is unnecessary to modify the DNS protocol, Blockzone is fully compatible with traditional DNSs. Since any server that deploys Blockzone can quickly identify false routing information, our scheme can improve the overall security protection capabilities of the DNS.

The main contributions of this paper include the following:

- This paper proposes Blockzone, which is a decentralized and trustworthy data plane for

DNS. Through the decentralization of the DNS zone storage, retrieval, and authentication mechanisms, the processing efficiencies of domain name resolution and domain name verification are improved. Due to the elimination of a single point of failure, the deployment of Blockzone renders DNS more robust against DDoS attacks.

- We implement an improved practical byzantine fault tolerance (PBFT) consensus algorithm, which is used to implement the DNS zone operation in Blockchain. This algorithm has the advantages of fast consensus and low traffic.
- Blockchain supports incremental deployments. As such, it can be deployed on root servers, top-level domain name servers, and authoritative servers, and it can collaboratively identify erroneous resolution results of domain names.

The remainder of this paper is organized as follows: Section 2 discusses related studies and analyses their limitations. Section 3 presents our motivation and objectives. Section 4 describes the architecture and algorithms of Blockzone in detail. Section 5 presents the results of experiments and evaluations. Section 6 presents the conclusions of the paper.

2 Related work

To reduce the architectural vulnerability of DNS, decentralization technology has attracted widespread attention, and many solutions have been proposed. In this section, we divide these solutions into three categories according to our own understanding: peer to peer (P2P)-based data plane decentralization, Blockchain-based control plane decentralization, and alliance-based management plane decentralization.

2.1 Data plane decentralization

Since there is no central node in the P2P network, a flat architecture for domain name storage can be realized by utilizing the P2P network. In this flat structure, domain name resource records (RRs) can be located via a single hash operation. The query path length is shorter than that of recursive or iterative query.

Cox et al. [Cox, Muthitacharoen and Morris (2002)] proposed a distributed DNS (DDNS), which is based on the P2P network. In DDNS, the resource records of each domain name are stored on a node of P2P network and are located using a distributed hash table, namely, Chord. DDNS inherits Chord's fault-tolerance and load balance properties and eliminates many administrative problems that are encountered with the current DNS. To realize load balancing, DDNS uses consistent hash tables to evenly distribute the keys at each stage and caches the query path while each node is retrieved. The time complexity of this query method is O ($logN$). To increase the robustness, DDNS automatically transfers data among server nodes by using distributed hash tables when a node joins or exits. Thus, the data are always stored on a fixed number of servers. Since these servers are selected pseudo-randomly, data are inaccessible only when all servers are down simultaneously.

Danielis et al. [Danielis, Altmann, Skodzik et al. (2015)] proposed P-DONAS, which is another implementation of decentralized DNS that is based on a P2P network. In P-DONAS, the access nodes of an Internet Service Provider (ISP) are organized into a P2P network based on distributed hash table Kademlia, and these nodes provide name resolution service instead of the DNS name server. In the process of domain name resolution, P-DONAS utilizes a look-up process of the P2P network. When a name resolution request is received, the access node initially searches its own cache. Then, it searches the P2P network to determine whether a local cache has been missed, and it forwards the request to an external DNS name server if no result is returned.

By combining a structured P2P network and proactive caching, Cooperative Domain Name System (CoDoNS) realizes high lookup performance and resilience against denial of service attacks [Ramasubramanian and Sirer (2004)]. In CoDoNS, all the name servers are organized into a flat P2P network rather than a fixed hierarchical relationship. Additionally, nodes will actively conduct cache synchronization when new domain name resolution results are generated or the topology of the network changes.

Although the Distributed Hash Table (DHT)-based DNS is more robust to Dos/DDos attacks, it has significantly higher latencies in comparison with traditional DNS. Song et al. [Song and Koyanagi (2011)] propose HDNS, which combines the hierarchical tree structure and the flat P2P structure. In HDNS, domain names are divided into two parts: The top-level domain name and the second-level domain name belong to the public zone, and the remaining domain names belong to the internal zone. The nodes in the public zone are organized using a P2P network, and the nodes in the internal zone are organized using a traditional DNS tree structure.

Many similar approaches are available, which we do not discuss here. The DNS architectures that are constructed based on P2P networks have the following disadvantages:

- Significantly high latency in the worst case. P2P networks have differed in terms of their processing delays due to differences in the underlying implementations; however, the worst-case query latency is not acceptable.
- Inconsistent domain name update. P2P networks allow any node to modify data. When a node is disconnected without modifying the data before the data are broadcast, the network node status will be inconsistent.
- Data spoofing. P2P networks have no data write rate limit or access control mechanism. Attackers can flood the entire P2P network with a large amount of junk data, or they can forge fake domain name information to spread to the entire network.

2.2 Control plane decentralization

When blockchain was introduced in 2008 as the fundamental technology of Bitcoin, it attracted widespread attention. Blockchain can be regarded as a decentralized and distributed database that consists of a series of blocks. These blocks can be used to maintain a continuously growing list of records. Each block contains a cryptographic hash of the prior block and a link to a previous block. In Blockchain, a record cannot be altered retroactively without the alteration of all subsequent blocks and the consensus of the network. The emergence of Blockchain has provided new research avenues for realizing the trustworthy storage and access of data in a distributed environment [Tsai, Yu, Wang et al. (2017)]. Since only authoritative servers can validate the results of domain name resolution, Dos/DDos attacks and DNS hijacking are inevitable in traditional DNS. DNSSEC provides a signature verification mechanism, but all verification must go through the root server and the verification efficiency is low. The consensus mechanism of the Blockchain implements a decentralized trust mechanism. Any node can verify the authenticity of a transaction without a trusted third party. To utilize this feature, Blockchain-based DNS was proposed.

Kalodner et al. [Kalodner, Carlsten, Ellenbogen et al. (2015)] proposed Namecoin, which is a name-value resolution system that provides the virtual '.bit' top-level domain name. In Namecoin, domain names are tradable resources and are traded through Bitcoin. Since the authenticity of a transaction can be verified by all users, the censorship of domain names does not require a trusted third party. Namecoin implements a decentralized management mechanism for Internet domain names, which has technical advantages in terms of privacy preservation and anti-spoofing. Ali et al. [Ali, Nelson, Shea et al. (2016)] proposed Blockstack, which is another outstanding approach that builds a decentralized naming system on top of an underlying Blockchain. The architecture of Blockstack has three layers: a Blockchain layer, a peer network and a data-storage layer. This architecture decouples the DNS logic from the underlying Blockchain and is more extensible than Namecoin. By introducing an independent data plane, the disadvantages of Blockchain data storage in terms of efficiency and capacity are overcome. This outcome enables BlockStack to realize a processing performance that is comparable to that of legacy DNS.

In addition, there are many other Bitcoin-derived systems that are based on Blockchain technology. These systems also provide name resolution services, such as Ethereum-based ENS [Bouquet and Molinari (2013)] and EMCDNS [Karaarslan and Adiguzel (2018)].

The current Blockchain-based domain name systems have the following limitations:

- Due to incompatibility with traditional DNSs, client browsers must install plug-ins to access the domain name system; hence, Blockchain-based domain name systems are difficult to deploy on a large scale.
- 51% attack: Both Namecoin and Blockstack are based on Bitcoin. If any organization maliciously controls 51% of the computing power of the entire system, which is referred to as a 51% attack, it will cause severe damage. Even if 25% of the computing power is maliciously controlled, such an attack can threaten the data security of the entire system [Eyal and Sirer (2014)].

2.3 Management plane decentralization

The current DNS has only 13 root servers, and these servers are deployed in a few countries. If these root servers are attacked or maliciously manipulated, top-level domains and their subordinate domains could become unresolvable.

Fang [Fang (2018)] presents a new DNS root architecture that can be used to replace the root server of DNS. To avoid a unilateral control problem, all root servers form a federation. All the members are operated by different countries and jointly provide root domain name resolution services based on peer-to-peer protocols. This solution can realize the decentralization of the DNS management plane, but it requires the support of various countries.

He et al. [He, Su, Gao et al. (2020)] presents a trustworthy decentralized DNS root management architecture, namely, T-Root, which is based on permissioned Blockchain.

The root server maintains the consistency of the zone file via a consensus algorithm and can tolerate up to one-third of the malicious servers behaving arbitrarily.

3 Motivation and objectives

Although the DNS of the Internet is a distributed system, it has significant centralization features on the data plane, control plane, and management plane. In the data plane, DNS adopts a hierarchical structure. The storage and partition of each domain name are determined by the root server. In the control plane, recursive and iterative domain name resolution are provided; in either approach, the root node must participate. If the root node fails, the entire parsing process will fail. In the management plane, the allocation and management of domain names are centrally managed by the Internet Corporation for Assigned Names and Numbers (ICANN). If this management is abused, services and resources will disappear from the Internet, which will cause significant losses.

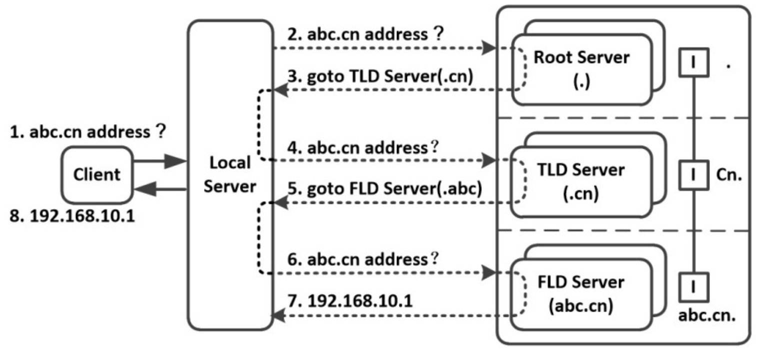


Figure 1: Traditional procedure of DNS name resolution

Fig. 1 illustrates a name resolution procedure in traditional DNS. This process has two main disadvantages: First, the resolution process must be with the root server. If the root server cannot provide the address of the top level domain (TLD) server, the resolution process fails. This failure increases the length of the resolving path and increases the risk of a single point of failure. Second, except for the final authoritative server, intermediate nodes can only forward requests and cannot determine the authenticity of the parsing results. If a malicious node masquerades as an authoritative server and returns false information, other nodes cannot recognize it [Schomp, Callahan, Rabinovich et al. (2014)]. Due to this flaw, the collaborative monitoring capability of current DNS is low.

DNSSEC, which is an important supplement to DNS security, realizes domain name verification. The verification process is the reverse process of domain name resolution, as illustrated in Fig. 2. DNSSEC uses digital signature technology that is based on the public key infrastructure (PKI) to ensure the authenticity of domain name resolution results. A unique trust anchor is required in PKI, and this requirement negatively impacts the efficiency and deployment of DNSSEC. First, as DNS is a hierarchical system, in addition to verifying the digital signature that is returned by the authoritative server, it also must verify the delegation from the upper-level server. This recursive verification process involves multiple nodes, is inefficient, and has the risk of a single point of failure. Second, the certificate of the root server belongs to a single management organization, and there is a unilateral control issue. Additionally, DNSSEC cannot be compatible with nodes that do not support the DNSSEC protocol. Therefore, it is not a highly deployable solution. This deficiency causes DNSSEC to be deployed only at the top-level domain name server, and the secondary server still has no ability to judge the authenticity of the domain name resolution results [DNSSEC (2020)].

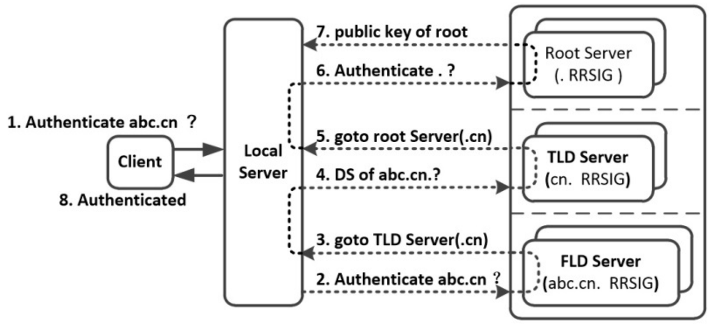


Figure 2: Traditional procedure of DNSSEC name authentication

According to the above analysis, to further improve the security protection capability of the DNS, it is necessary to change its architecture. Since the traditional DNS is closely bound to many Internet applications, it is impossible to replace the traditional DNS with a new DNS in a short period of time. We hope to decentralize the DNS while maintaining its compatibility with available systems, thereby increasing its processing efficiency and robustness.

The scheme that is proposed in this paper has the following design objectives:

- Increase the efficiency of domain name resolution and verification.
- Improve the collaborative monitoring performances of single DNS servers.
- Enable progressive deployment. Our solution does not require a complete replacement of the traditional DNS. It can gradually expand the scope of deployment, and the overall effect can be improved with the expansion of the scope of deployment.

4 Overview of Blockzone

4.1 Description of the basic strategy

DNS uses a namespace to define all Internet domain names, and the namespace includes top-level domains (such as ".cn"), second-level domains, (such as "edu.com") and lower-level domains, which ae. also called subdomains (such as "gzhu.edu.cn"). For ease of management, the domain name space is divided into many zones. A DNS zone refers to a portion of the namespace, and each DNS zone represents a boundary of authority that is subject to management by specified entities. The collection of all DNS zones, which are organized in a hierarchical tree-like order of cascading lower-level domains, constitutes the DNS namespace. At each hierarchical level of the DNS, there is a name server that contains a zone file, which holds the trusted, correct DNS records for that zone.

When an Internet application must find the IP address for a domain name, such as "www.gzhu.edu.cn", it conducts a DNS lookup by submitting a request to the DNS server that manages the DNS zone for that domain name. This server is called the authoritative name server for the domain. The authoritative name server resolves the DNS lookup by providing the IP address and other data. All the information for a zone is stored in a plain text file, which is called a DNS zone file. A zone file contains mappings among domain names, IP addresses and other resources, which are organized in the form of Resource Records (RRs). Since the storage of zone files is organized according to the hierarchical relationship of the DNS domain name space, the authoritative server that contains the specified domain name must be located recursively during the domain name resolution process.

The traditional DNS resolves domain names according to a hierarchical structure, which has three disadvantages: First, it must pass through the root node, and there is a risk of a single point of failure. Second, the recursive parsing path is too long, and the efficiency is low. Third, the authenticity of the parsing result is partially visible. Only an authoritative server can determine its authenticity, and it is difficult to defend against domain name hijacking attacks.

To eliminate the single point of failure, we use a distributed file system, namely, IPFS, to store replicas of the DNS zone files. Through the distributed file system, each server can access the contents of the file if it knows the address. To preserve the management authority of the DNS management organization, modifications to the zone file by the distributed file system are invalid, which is guaranteed via digital signature technology. To eliminate the unilateral control of the root node, all the DNS servers are organized into a consortium Blockchain. The index information and basic description information of the DNS zone file, which is called the metadata of the zone, are stored on the Blockchain. Due to the immutability and global visibility of the data on the Blockchain, every domain name resolution server can verify the authenticity of the domain name resolution results through a local client. Therefore, our scheme can effectively improve the ability to monitor malicious parsing behaviour. For convenience, we refer to our scheme as Blockzone, and its operational principle is illustrated in Fig. 3.

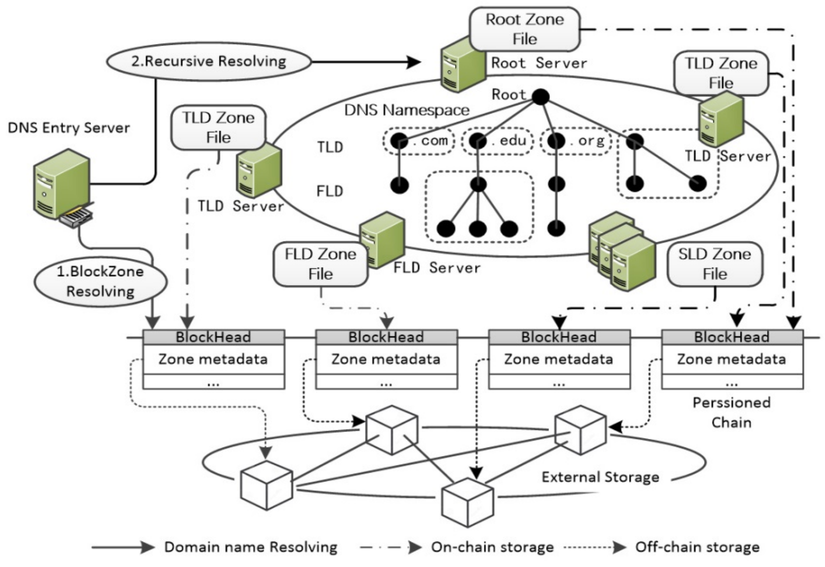


Figure 3: Operational principle diagram of Blockzone

As illustrated in Fig. 3, when a DNS entry server begins a DNS recursive query, it initially queries the Blockchain to directly locate the zone file that corresponds to the domain name that must be resolved. After obtaining the address information, it returns the resolution result. Only if the resolution fails will the traditional recursive query process be initiated.

Blockzone differs from BlockStack and NameCoin as follows:

- Blockzone does not change the traditional DNS operation management model, namely, the domain name is still managed and authorized by a specified organization instead of everyone being free to register and declare; therefore, we chose the consortium Blockchain as the implementation method of the Blockchain. Compared with the public Blockchain, it has advantages in resisting the Sybil attack and the Byzantine attack.
- Blockzone is progressive deployable and compatible with available systems. Without changing the protocol interface, we embed Blockchain services into the process of accessing the underlying data. The entire process is transparent to higher-level protocols and does not change the implementation of the original protocol. When Blockchain services are missing or failing, the resolving server can seamlessly switch to the traditional recursive parsing process.

4.2 Architecture

Based on the implementation architecture of the traditional DNS resolution server, Berkeley Internet Name Domain (BIND), we expand the lookup and storage procedures for DNS zone files [Liu and Albitz (2006)]. In the process of zone file local lookup, we insert a lookup procedure that is based on the Blockchain platform, which can not only increase the efficiency of domain name resolution but also ensure the credibility of the resolution result. The architecture of Blockzone is illustrated in Fig. 4, which includes an application layer, a protocol layer, a service layer and a data layer. Each layer is defined in detail as follows.

- Application layer

The application layer mainly includes two types of users: a DNS management application and DNS clients. The DNS management application is responsible for the registration, assignment, cancellation, and modification of domain names. When the DNS management application changes any resource record of a domain name, it accesses the Blockchain via a smart contract.

- Protocol layer

The protocol layer implements a DNS protocol session. When a DNS resolution server receives a name resolution request, it creates a session and processes the request.

- Service layer

We deployed the BlockZone plug-in service in the service layer, which obtains domain name resolution records through the Blockchain. If the Blockzone plugin service fails to resolve the domain name, the local resolution service will forward the request to the external server to complete the recursive resolution process of traditional DNS. Since the DNS protocol is not changed and services are allowed to fail, progressive deployment is realized.

- Data layer

Considering the limited storage capacity of the Blockchain, we adopt a combined off-chain and on-chain data storage mechanism. The resource records of each domain name are stored off-chain in a distributed file system, and the ownership information of each domain name and the index information that is used to locate the complete information are stored on-chain.

The data layer includes a consortium Blockchain and a distributed storage system. The consortium Blockchain mainly provides three functions: First, it stores the authorization information, index information and signature information of each domain name. Second, it queries and operates the complete resource records of domain names through index information, which is stored in the distributed storage system. Finally, it processes each request from the management application and synchronizes the information via a smart contract.

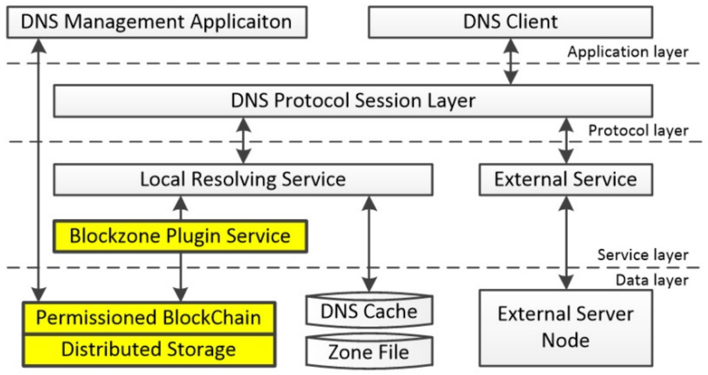


Figure 4: Architecture of Blockzone

4.3 On-chain data storage

4.3.1 Metadata of the DNS Zone

The metadata of the DNS zone refers to the description of the DNS zone file, which mainly includes the ID, header information of the DNS zone file, name list, signature, zoneFileHash, and fragmentation information. The structure of the metadata is illustrated in Fig. 5.

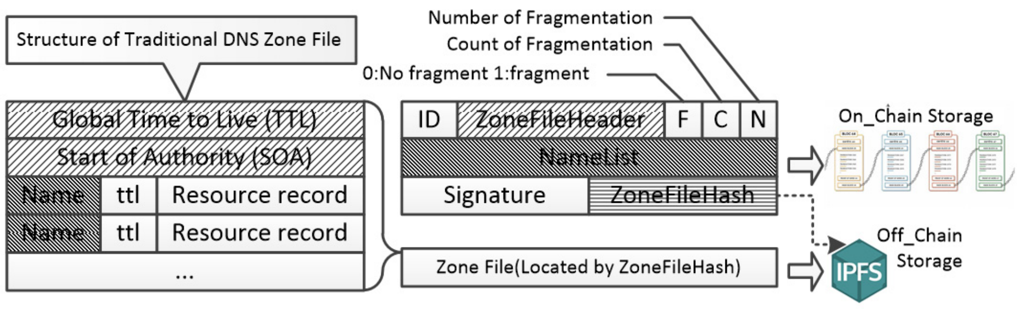


Figure 5: Metadata of the DNS zone file

The ID is a unique identifier of the metadata record. The ZoneFileHeader contains the header information of the DNS zone. It mainly includes Start of Authority (SOA) and Time to Live (TTL). The NameList records all the domain names contained in the zone. The ZoneHash is used to determine whether the data in the external storage are damaged. The external address is the storage address of the off-chain data. Considering the block size of Blockchain, Blockzone supports the fragmentation and storage on the chain of the original DNS zone file.

4.3.2 Assumption for consensus

If a management agency of the DNS zone must register, modify, or deregister a new domain name, it will notify other nodes. After reaching a consensus, the new record is accessed by the block and is globally visible. Consensus algorithms strongly affect the performance of the Blockchain. In view of the characteristics of the DNS, we designed the Blockzone consensus protocol based on the following assumptions:

- Availability Assumption

As an important network facility, the DNS server typically has a satisfactory operating environment and a stable network connection. Therefore, we assume that the DNS-server-deployed Blockzone is always online and accessible. Hence, most servers can receive or return synchronization messages within an expected maximum time.

- Security Assumption

To ensure the security of the system and reduce the likelihood of malicious nodes, we assume that all DNS nodes that deploy Blockzone are trustworthy and are organized into an alliance. Each node must register its public keys and accept authentication prior to writing to the block. In the traditional DNS, all nodes that are responsible for managing the DNS domain name space are authorities, not ordinary Internet users. Therefore, these assumptions are reasonable.

4.3.3 Consensus algorithm

Various types of consensus algorithms are available, which include practical Byzantine fault tolerance (PBFT), Raft, Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), and Ripple, among others. The PBFT algorithm is the most commonly used consensus algorithm in private and consortium blockchains, which are collectively referred to as the permissioned blockchain. The communication complexity of the standard PBFT algorithm is $O(N^2)$ [Castro and Liskov (1999)], and the lightweight quorum-based protocol can reduce the communication complexity to $O(N)$ in the case of contention absence [Cowling, Myers, Liskov et al. (2006)].

Although many optimization algorithms were developed based on PBFT in recent years, most improve the performance by optimizing node selection, properly relaxing algorithm constraints and implementing other methods, and there is no essential difference between them and the PBFT algorithm. Therefore, Blockzone implements a consensus mechanism that is based on the PBFT algorithm. The PBFT algorithm offers both liveness and safety if at most $(n-1)/3$ out of a total of n replicas are simultaneously faulty. A PBFT-based distributed file system that supports the Network

File System (NFS) protocol has been implemented, and it is only 3% slower than the standard NFS. Thus, the performance is acceptable.

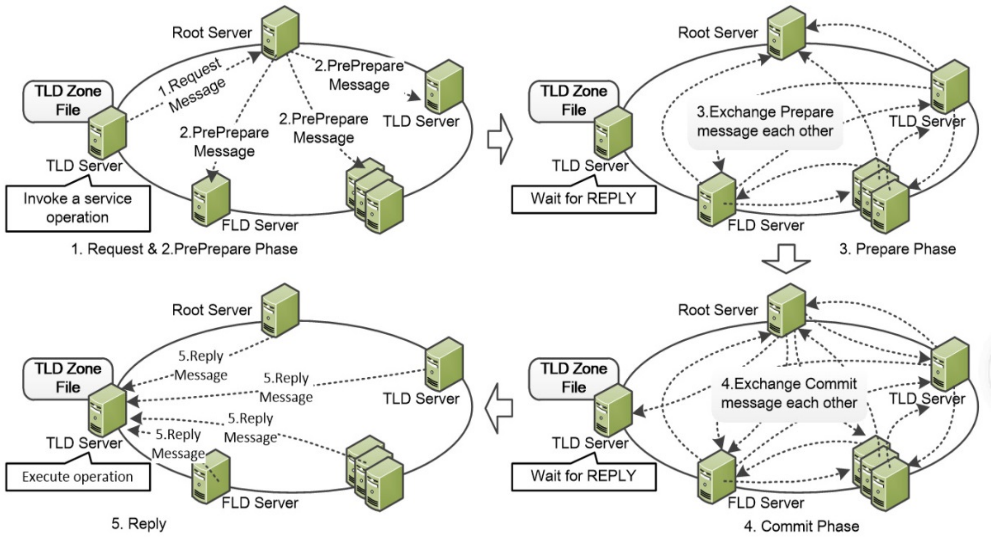


Figure 6: The basic process of the PBFT consensus algorithm

The basic process of the PBFT consensus algorithm is illustrated in Fig. 6. When a DNS node must publish new domain name information, it will write the information into the Blockchain as a transaction. Prior to this process, the node selects the higher-level node as the master node of this consensus process and sends a request message to the primary node. After receiving the request message, the primary node sends a *PREPREPARE* message to other registered DNS nodes, which are backup nodes during this procedure, and asks for confirmation. Regardless of any network delay or packet loss, when other nodes receive the *PREPREPARE* message, they verify the message and broadcast the *PREPARE* message. When the primary node and backup nodes have received sufficiently many *PREPARE* messages, all the nodes are ready. Therefore, the primary node and backup nodes execute the request and send a *REPLY* message to the requesting node. When the requesting node receives a sufficiently many *REPLY* messages, the network-wide synchronization is complete. Finally, the requesting node completes the writing of the information and, thus, completes the consensus process.

Before presenting the consensus algorithm, we briefly introduce several concepts that will be used in the following algorithms. When a consensus process is in progress, it may be interrupted due to the failure of nodes or the network. When timeout occurs, a new consensus process must be initiated. To ensure that similar messages that are generated in different processes do not cause conflicts or confusion, PBFT uses View to distinguish messages of the same type that are generated in different processes. Each interactive message must be associated with a view number. If the view number of the message does not match the view number that is used in the current consensus process, it will be discarded. In addition, to ensure that the messages that are received by all participating nodes are consistent, the messages must be sorted. Thus, each message must carry a timestamp and is stored in the log file of each node according to the time sequence. The use of timestamps also protects against replay attacks. The last important concept is a signature. When a DNS node must generate a message during the consensus process, in addition to setting the view number and the timestamp of the message, it must use its own private key to calculate a digital signature for the message to ensure the integrity of the message. We denote a message *m* that is signed by node *i* as m_{δ_i}.

We implement a consensus algorithm based on the basic principles of PBFT in Blockzone. The principle codes are presented as follows.

Algorithm 1. Consensus Algorithm that is based on PBFT

Input: REQUEST *message*, *viewID*, *timestamp, cert of the node, handle of log file*
Output: *result of operation, log record*

```
/* When a Blockzone node receives a request message from any node, it executes the
following code. The request message has the form
$< \text{REQUEST}, \boldsymbol{opeation}, \boldsymbol{timeStamp}, \boldsymbol{client} >_{\delta_c}$*/
enum status {IDLE, PREPREPARE_PHASE, PREPARE_PHASE, COMMIT_PHASE};
status = IDLE;
while (status == IDLE) do {
  msg = listenMSG(anyNode);
  switch msg.type
  {
    case REQUEST:
        if valid_Request_MSG(msg) && isPrimaryNode() && status == IDLE {
            pp_msg create_Pre_Prepare_MSG(PRE_PREPARE, view, number, digst) ;
            /*$<< \text{PRE}_{\text{PREPARE}}, \boldsymbol{view}, \boldsymbol{number}, \boldsymbol{digst} >_{\delta_p}, \boldsymbol{m} >$ */
            multicast_MSG（pp_msg）;
            status = PREPREPARE_PHASE;
            log(msg); log(pp_msg);
        } /* Primary Node's Code */
    break;
    case PRE_PREPARE:
        if valid_PrePrepare_MSG(msg) && isBackupNode() && status==IDLE {
            status = PREPREPARE_PHASE;
            if msg_Acceptable(msg) {
                p_msg = createPrepareMSG(PREPARE, view, number, digst, i);
                multicastMSG(p_msg);
                status = PREPREPARE_PHASE
                log(msg);
                log(p_msg);
            }
        }
    break;
    case PREPARE:
        if valid_Prepare_MSG (msg) && status= PREPARE_PHASE {
            log(msg);
            if prepared(m, v, n, i) {
                commitMSG = reateCommitMSG(COMMIT,view,number,digst(m),i);
                /* $<<\text{COMMIT}, \mathbf{v}\text{iew}, \mathbf{n}\text{umber}, \mathbf{d}\text{igst(m)}, \text{i} >_{\delta_i}, \boldsymbol{m} >$*/
                multicastMSG(commitMSG);
                status = COMMIT_PHASE;
                log(commitMSG);}
            }
    break;
    case COMMIT:
        if valid_Commit_MSG(msg) && status= COMMIT_PHASE {
            log(msg);
            if committed-local(m,v,n) {
                result = execute(m.operation);
                replyMSG = createReplyMSG(REPLY, view, timeStamp, client, i, result);
               /*$< \text{REPLY}, \mathbf{v}\text{iew}, \boldsymbol{timeStamp}, \boldsymbol{client}, i, \boldsymbol{result} >_{\delta_c}$ */
               Status = IDLE; /*finished*/}
        }
    break;
    default:
        DiscardMSG(msg);
        break;
    } /*end of switch*/
} /* end of while*/
```

If the primary node fails and does not broadcast the consensus request message within the specified time *t* or the backup node broadcast view update message does not obtain *2f* node confirmations or the number of node consensus confirmations is less than *2f*, the view update operation is conducted. The variable *f* is the maximum number of replicas that may be faulty.

The process of view updating is described as follows:

1. Increase the view $v = v + 1$;
2. The backup node sends a view update message $< ViewChange, h, v, s, v', m_{\delta_i} >$, where *ViewChange* specifies the message type as view update, *h* is the current block height, *v* is the current view number, *s* is the child node number, v' is new view number, and m_{δ_i} is the signature of the message;
3. If the consensus node accepts the number of view update broadcast messages, the view is updated to v', the master node is updated to $m + 1$, and a new consensus process begins;
4. If the number of received view update messages does not reach 2*f*, return to Step 1 to continue.

If the primary node network is unstable or the consensus node network fluctuates, frequent view changes may occur, which will consume network resources. To avoid frequent triggering of view replacement due to network fluctuations, the operating time of the primary node should increase with the index of view updating. If view updating occurs frequently, it indicates that network fluctuation is occurring, and the time *t* should be increased. The setting of time t is shown in Formula 1.

$$\text{T(k)} = 2^{\text{k}} * \text{t}, 0 \leq \text{k} \leq 10 \qquad (1)$$

Function *T* increases exponentially with the number of view updates, which can avoid frequent view changes that are caused by network fluctuations, which waste network resources. To avoid unlimited time growth, the number of consecutive view changes is less than 10.

4.4 Off-chain data storage

In Blockzone, on-chain data are mainly used for authorization and authentication, and off-chain data are used to support domain name resolution. Hence, off-chain storage is an important element of DNS decentralization. The management of domain names includes links such as registration, update, and cancellation. These operations are designed for modifying zone files; hence, it is not efficient to complete all on-chain. In addition, the Blockchain's data storage capacity is limited, and external systems are needed to compensate for its lack of storage capacity. According to the latest data from Verisign, there were 359.8 million registered domain names at the close of 2019's third quarter [WebSiteHub (2020)]. A zone file that contains only a single domain name is approximately 1K bytes in size. Therefore, the zone file space that is occupied by all domain names is approximately 360 G. This space is larger than that for the Bitcoin Blockchain. Therefore, we attempted to implement off-chain data storage based on IPFS, and we realized the linkage between Blockchain and IPFS through intelligent reduction.

The IPFS is a distributed file system that uses content-based addressing instead of using location-based addressing (such as the domain name, IP address, and the path to the file). When a file is added to an IPFS repository, it can be referenced by its cryptographic hash. By using IPFS, zone files that are scattered among various DNS servers can be organized for efficient retrieval while maintaining data autonomy and privacy.

In Blockzone, when the DNS server resolves a domain name, it obtains the value of the ZoneFileHash via on-chain retrieval, as illustrated in Fig. 5, and uses this value as an address to acquire the zone file through the IPFS client. If the zone file is not saved locally, an IPFS file is created. The hash search function, because it is a hash search, can quickly locate and obtain zone files and cache them locally. The whole process is illustrated in Fig. 7.

In the process that is illustrated in Figure 7, when the resource records of multiple domain names are saved in the same Zone file, the resolving server can obtain the entire ZoneFile content through a single resolution. Due to privacy concerns, the owner of the ZoneFile may not accept it. For overcoming this problem, Blockzone provides two solutions:

- Via Smart Contract

The owner of the ZoneFile encrypts the content of the resource record before uploading the ZoneFile to IPFS. After obtaining the encrypted field, the domain name resolver decrypts the field via a smart contract on the Blockchain. Since the owner of the ZoneFile can specify the decryption rules for the data in the smart contract, the decryption scope can be controlled. In Blockzone, the registration, update and cancellation of domain names are realized through smart contracts; see Section 4.5;

- Via File Fragmentation

The ZoneFile could be decomposed into multiple small files that contain only a small number of domain names. Not only can this decomposition improve transmission efficiency, but it can also realize hierarchical protection. Three fragment fields, as illustrated in Fig. 5, are designed to support such methods.

The above two mechanisms can also be used in combination.

For protecting the offline data, the main mechanisms are as follows: First, IPFS locates files based on the index of the data file content, and Blockzone can ensure the uniqueness of the ZoneFile when generating the ZoneFile; hence, file conflicts can be avoided. Second, all ZoneFiles will carry digital signatures; thus, if the data are damaged during storage, they can be verified by digital signatures. Additionally, ZoneFiles can be encrypted in Blockzone. During the process of resolving domain names, the decryption can be conducted through a smart contract. Due to the trustworthiness of the smart contract, the content of the ZoneFile can only be accessed by authorized users. Finally, IPFS has strong data reliability storage capabilities, which can ensure the reliability of ZoneFiles in storage.

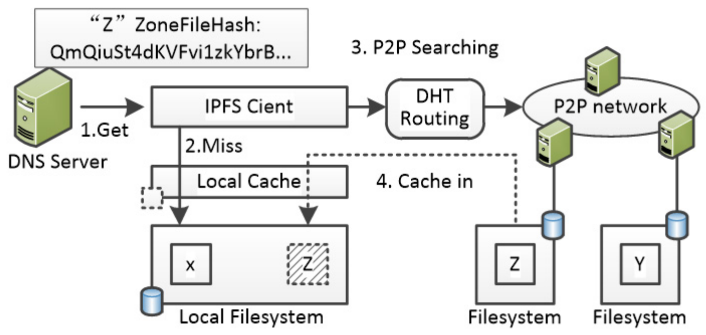


Figure 7: Procedure for obtaining Off_chain data

4.5 DNS Zone operation

In contrast to BlockStack and NameCoin, Blockzone does not change the management or operation of domain names. All the operations on domain names are completed by the management agency. Blockzone mainly provides access services for data that are stored on-chain and off-chain, and these services include registry, lookup, modification and revocation.

4.5.1 Zone registration

The domain name server uses the zone registration service to publish a zone file that contains the new domain name information on Blockzone. Since all necessary checks are completed by the smart contract during the registration process, malicious domain name registration will be difficult to implement; hence, the domain name registration is credible. The main process is as follows:

1. The DNS management application (illustrated in Fig. 4) submits a registration request to the Zone Registration service through the smart contract access interface.
2. The Zone Registration service queries the Blockchain to check if the Zone File is already registered. If so, it rejects the user's request by returning a failure message.
3. If the DNS Zone file is available for registration, the DNS management application connects to the IPFS network and adds the DNS zone file onto the IPFS network. After this, IPFS will return a unique hash value that is based on the contents of the

zone file. Through this hash, any node can access the file in the IPFS network.

4. The DNS management application constructs and submits Zone metadata (illustrated in Fig. 5) to the Zone Registration service. The smart contract that is deployed on the Blockchain will invoke a consensus procedure (as discussed in Section 4.3). Since Blockzone is designed as a plug-in for the traditional DNS, we have not considered the charging mechanism of smart contracts during the execution.
5. Once the consensus is realized successfully, the DNS Zone metadata will be written into the Blockchain.
6. The smart contract of registration returns a confirmation to the DNS management application, and the registration is complete.

4.5.2 Zone Lookup

As illustrated in Fig. 2, the local name resolution service will send a lookup request to Blockzone before it initiates a recursive querying procedure. The main process is as follows:

1. The local name resolution service submits a lookup request for a desired domain name to the lookup service.
2. Upon receiving the request, the smart contract checks if the name is a valid name and if there are Zone metadata that contain the desired domain name. If yes, it returns the zone file hash; otherwise, it returns an error.
3. If the local name resolution service receives an error message, it begins a recursive querying procedure.
4. If the local name resolution service receives a DNS zone file hash, it connects to the IPFS network to access the zone file with the hash value.
5. When the name resolution service receives the zone file contents, it parses the zone file and obtains the corresponding resource records. In some cases, the owner of the Zone File may encrypt resource records for privacy preservation. At this time, it can return to the application after completing the decryption through the smart contract.

4.5.3 Zone revocation

When the owner of a DNS zone file seeks to modify the content of the Zone file, it must initiate a revocation transaction. Only the owner of the domain name can initiate the revocation transaction. This checking process is conducted using smart contracts. The main process is as follows:

1. The DNS management application submits a revocation request to the Zone Revocation service through the smart contract access interface.
2. The Zone revocation service queries the Blockchain to check if the Zone File is already registered and if the requester is the original owner. If not, it rejects the user's request by returning a failure message.
3. If the Zone revocation service returns *yes*, the DNS management application connects to the IPFS network and adds an empty DNS zone file, which contains no domain name, on the IPFS network. After this, IPFS will return a unique hash value that is based on the contents of the zone file. Since IPFS cannot temporarily delete

files that have already been uploaded, we must compensate for this deficiency by uploading new files. Blockzone suffers from the same deficiency.

4. The DNS management application constructs and submits new Zone metadata to the Zone Registration service. The smart contract that is deployed on the Blockchain will invoke a consensus procedure again. Once the consensus is realized successfully, the new DNS Zone Metadata will be written into the Blockchain.
5. The smart contract of registration returns a confirmation to the management application, and the registration is complete. After that, if the domain name file is accessed again, an invalid resolution result will be returned. Thus, we implement the replacement of the Zone.

4.5.4 Zone update

When the management agency of DNS adds or changes domain name information, it must modify the zone file. Once the zone file has been saved to the Blockchain and IPFS, it cannot be deleted. Therefore, before updating a zone file, we must abolish the old zone file and register the new one. To avoid the frequent changes in the zone file, we recommend using the fragmentation mechanism that is provided by BlockZone to finely divide the zone so that each file contains only one or a few domain name resource records. When a single domain name changes, it will not affect other domain names.

5 Evaluation

5.1 Efficiency

5.1.1 Experimental Setup

The simulation environment is illustrated in Fig. 8.

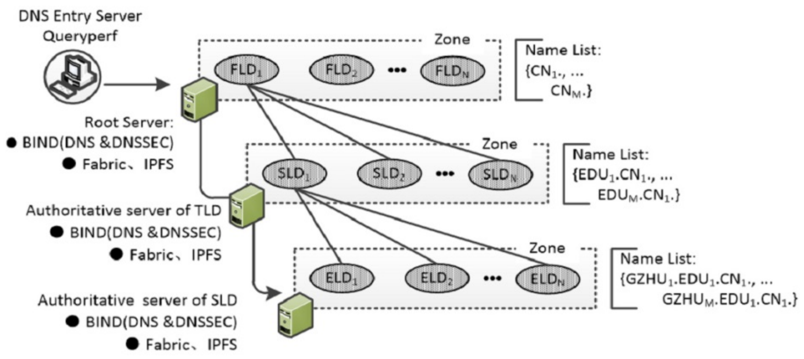


Figure 8: Experimental setup for efficiency

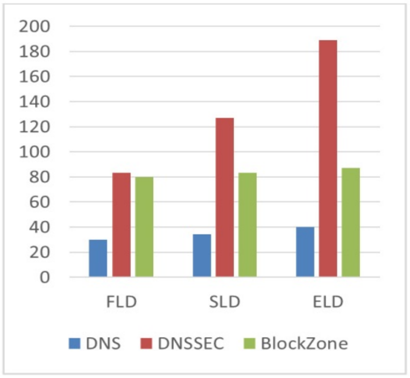
We implement the prototype of Blockzone that is based on BIND9 and Hyperledger Fabric [Androulaki, Barger, Bortnikov et al. (2018)]. To evaluate the availability of Blockzone, we set up a three-tier DNS instead of using Internet DNS so that we can control the path of query requests by adjusting the location and content of the zone file. All servers are deployed on Aliyun Cloud and run as virtual machines. In the simulation DNS, the namespace is divided into the first-level domain (FLD), second-level domain

(SLD) and third-level domain (ELD). Each level of Zone contains 50,000 domain names. For example, the zone file on the root server contains domain names from 'CN1.' to 'CN50000.'. We installed a stress testing software, namely, QueryPerf, in the DNS entry server and used it to send domain name resolution requests. QueryPerf sends resolution requests for the domain names in FLD, SLD, and ELD, in turn. We compare traditional DNSSEC and Blockzone in terms of the response latency and queries per second.

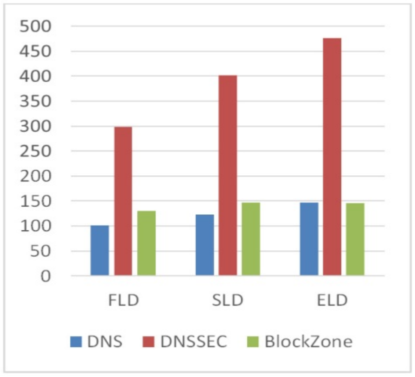
To examine the effect on the query efficiency, which depends on the length of the recursive query path, we evaluate the latency and throughput in the cases of cache hit and cache miss, respectively. In the case of a cache hit, the server locally caches the most recent query results and returns them directly to the requester. Without cache, the server must obtain the query results through the server where the zone file is located. We implement cache hits by resolving the same set of domain names repeatedly, and we query various domain names to implement cache misses.

5.1.2 Results of the efficiency experiment

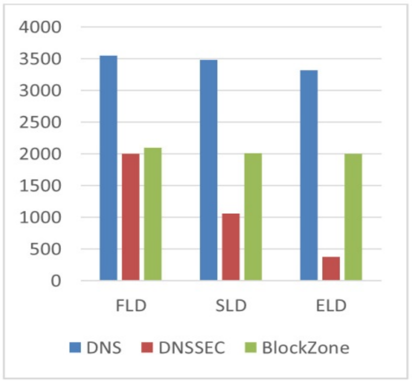
The results of the availability experiment are presented in Fig. 9.



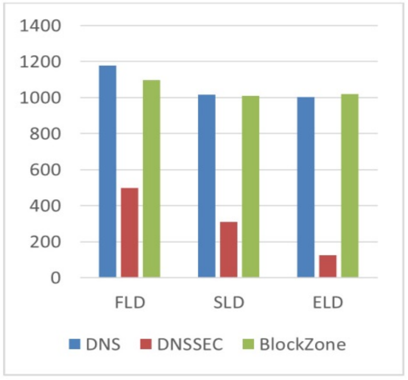
(a) Response time (ms) in case of cached



(b) Response time (ms) in case of uncached



(c) Query per second in case of cached



(d) Query per second in case of un-cached

Figure 9: Results of the availability experiment

From Fig. 9, observe the following:

Observation 1: In the case of a cache hit, the efficiency of traditional DNS resolution is the highest, and it is not affected by the length of the domain name. However, the efficiency of DNSSEC has decreased significantly with the increase of the domain level. This is due to the signature verification from the root level to the end level. Since Blockzone adopts a decentralized design and need only verify one level of digital signature, its resolution efficiency is not affected.

Observation 2: In the case of a cache miss, the resolution efficiencies of DNS and DNSSEC are closely related to the length of the resolution path. If multiple servers must be queried, the resolution efficiency will be significantly reduced (see the analysis process in Fig. 1 and Fig. 2). Blockzone locates the result via the client of Blockchain instead of recursive resolution.

Observation 3: Since most applications use third-level domain names, Blockzone's response time is close to that of DNS when resolving third-level domain names, and its throughput is higher than that of DNSSEC, it can be deployed in the second-level DNS name server instead of DNSSEC.

5.2 Performance

5.2.1 Experimental setup

Although Blockzone can be deployed on any Blockchain platform, to ensure security, the entire DNS should not be randomly joined by ordinary Internet users but should be composed of authenticated nodes. In addition, the processing performance of the public chain is insufficient for satisfying the performance requirements of domain name resolution. Based on these considerations, Blockzone utilized permissioned Blockchain as the fundamental platform, which is highly suitable for a DNS that is composed of many authoritative servers.

Hyperledger Fabric is an enterprise-grade open-source permissioned Blockchain that allows components, such as consensus and membership services, to be plug-and-play. Fabric is now being used in many use cases. We implement Blockzone via a smart contract rapid development platform, namely, XML Linking Language (XLINK), which is an optimized implementation version of permissioned BlockChain that is based on Hyperledger Fabric.

Since all the domain name operations are executed via a smart contract in Blockzone, we used the throughput and latency of transaction as the primary performance metrics for Blockzone. The throughput refers to the rate at which transactions are committed to the BlockChain, and the latency is the time from request sending to transaction commit.

We construct a Fabric network as illustrated in Figure 10, which has four peer nodes that belong to different organizations, which correspond to the root server, DNS top level domain (TLD) server, second level domain (SLD) server and extended level domain (ELD) server. On each server, we deploy BIND9 and Blockzone. In addition, we use an independent server to act as a client and generate many transaction requests, which include domain name registration and domain name query. The sorting service is provided by another server. As with the experiment on efficiency that was presented in

Section 5.1, all servers are deployed on Aliyun Cloud and run as virtual machines. Each VM is configured with 8 vCPU, 16 G memory and up to 2 Gbps network throughput.

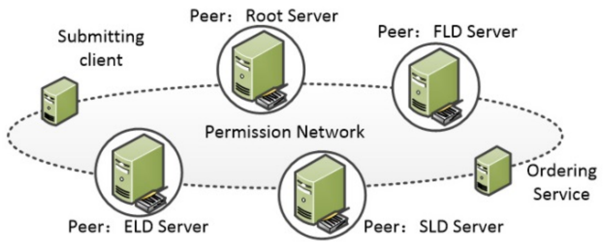


Figure 10: Experimental setup for performance

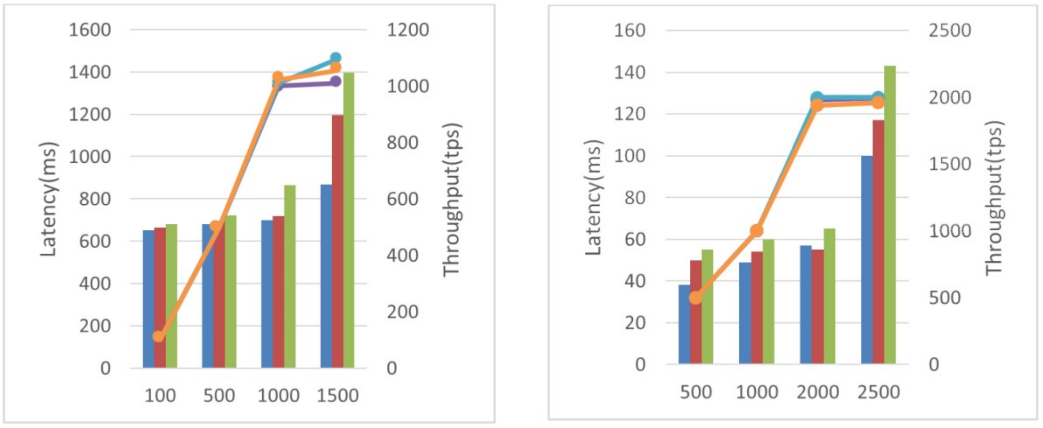
To eliminate the impact of the network environment, our experiment lasted 60 minutes. During the experiment, we evaluated the performances of two types of operations: zone lookup and zone registration (see Section 4.3). Zone registration will trigger the PBFT consensus process. We also analysed the impacts of the block size and node load on the experimental results. Tab. 1 lists the main experimental parameters.

Table 1: Parameters of the performance experiment

Parameter	Value
Batch Size	100, 200, 500 (tx)
Batch Timeout	2 seconds
Request rate of Client (TPS)	100, 500, 1000, 1500

5.2.2 Results of the performance experiment

Fig. 11 presents the average throughput and latency for various block sizes at various rates of request. The line graph represents the change in throughput, and the histogram represents the processing latency. The horizontal axis represents the change in the request rate.



(a) Zone registration (b) Zone lookup

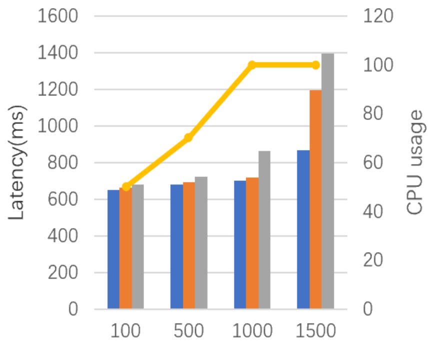
Figure 11: Performance results

From Fig. 11, we observe the following:

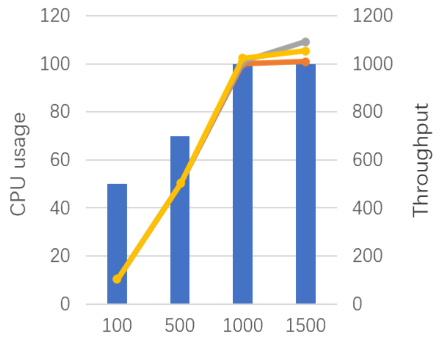
Observation 1: It is possible to realize the registration of approximately 1,000 domain names per second under the configuration conditions of this experiment, which proves that the permissioned Blockchain is highly suitable for the implementation of DNS decentralization. In addition, the query performance of Blockzone can reach almost 2000 Transactions Per Second (TPS), which does not include the signature verification time for off-chain data. Under the same configuration, the query performance of DNSSC is approximately 1000 TPS; hence, Blockzone outperforms DNSSEC in processing. This is due to the decentralized design, which shortens the length of the verification path and reduces the number of times digital signatures are calculated.

Observation 2: Until the processing bottleneck is reached, the system throughput increases linearly with the request rate. When the request rate exceeds the processing bottleneck, the processing delay begins to increase significantly. This is mainly due to the accumulation of many transaction requests in the transaction queue of the processing node, thereby resulting in longer waiting times. To solve this problem, we can add more processing nodes and introduce a load balancing mechanism. There is far more than one server at the same domain name level.

Observation 3: Until the processing bottleneck is reached, the value of the batch size has little effect on the processing performance of the system. Therefore, to avoid the re-registration of the entire zone due to a domain name change, the number of domain name records that are contained in a single block should be reduced. In addition, if a block contains too many records, this will also lead to too long of a wait for consensus, thereby resulting in an increased processing delay.



(a) Latency & CPU



(b) Throughout & CPU

Figure 12: CPU usage and processing performance

Fig. 12 presents the relationship between the CPU usage and the system processing performance. From Fig. 12, we observe the following:

Observation 1: In the case of a node that is running at full capacity, further increasing the request rate will significantly increase the processing delay, thereby resulting in a decrease in the availability of the entire system. In the case of limited resources, suitably increasing the batch size facilitates performance. This is because by reducing the number of blocks, the scheduling overhead and network communication times can be reduced, thereby reducing the load on the nodes.

Discussion: According to the results of the above experiments, when a single server reaches the performance bottleneck, the efficiency of domain name resolution cannot be further improved. This problem is severe in the traditional DNS. Therefore, in the traditional DNS, when a single authoritative server cannot meet the performance requirements, we must set up redundant authoritative servers to improve the processing performance, but the number of redundantly configured authoritative servers remains far less than the total number of servers of the entire DNS. However, setting up too many redundant servers will increase the difficulty of data synchronization and increase the likelihood of DNS poisoning attacks. Compared with traditional DNSs, Blockzone is more scalable. Blockzone can well utilize the scalability of the blockchain system. Any authoritative server can obtain a global domain name view through the deployment of Blockzone and provide domain name resolution services locally. Therefore, when the domain name server reaches the performance bottleneck, it can forward the domain name request to any other authoritative servers to complete the domain name resolution. The more authoritative servers that deploy BlockZone, the higher the overall throughput rate is.

The construction of a decentralized DNS based on permissioned blockchains is practical. In addition to satisfying user performance requirements, it can increase the credibility of the domain name resolution results.

6 Conclusions

This paper presents an implementation scheme for a DNS data plane, which is called Blockzone. Based on the combination of on-chain authorization and off-chain storage, Blockzone implements a decentralized and trustworthy DNS data plane, which renders the DNS more robust in the face of Dos / DDos attacks and buffer poisoning attacks. The experimental results demonstrate that Blockzone is more efficient than traditional domain name resolution systems and has lower storage overhead. Since Blockzone does not modify the DNS protocol, it can be integrated into the traditional DNS as a plug-in for the data plane; hence, it has satisfactory compatibility and progressive deployment capabilities. The core strategies of Blockzone can also be used in applications such as anomaly detection, vehicle security, and edge computing.

Acknowledgement: Thank the equipment support of Guangzhou University and the support of National Natural Science Fund of China.

Funding Statement: This research was supported by National Natural Science Foundation of China (Grant No. 61976064), Project of National Defense Science and Technology Innovation Zone (Grant No. 18-H863-01-ZT-005-027-02), Equipment Pre-Research Key Laboratory Fund Project (61421030203), and Zhijiang International Young Talent Scheme (2019).

Conflicts of Interest: We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submit.

References

Ali, M.; Nelson, J.; Shea, R.; Freedman, M. J. (2016): Block stack: a global naming and storage system secured by block chains. *Proceedings of the USENIX Annual Technical Conference*, vol. 1, no. 1, pp. 181-194.

Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K. et al. (2018): Hyperledger fabric: a distributed operating system for permissioned blockchains. *Proceedings of the Thirteenth EuroSys Conference*, vol. 1, no. 1, pp. 1-15.

Benet, J. (2014): IPFS-content addressed, versioned, p2p file system. arXiv:1407.3561.

Bouquet, P.; Molinari, A. (2013): A global entity name system (ENS) for data ecosystems. *Proceedings of the VLDB Endowment*, vol. 6, no. 11, pp. 1182-1183.

Castro, M.; Liskov, B. (1999): Practical byzantine fault tolerance. *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, vol. 1, no. 1, pp. 1-14.

Cowling, J.; Myers, D. S.; Liskov, B.; Rodrigues, R.; Shrira, L. (2006): Hq replication: a hybrid quorum protocol for byzantine fault tolerance. *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, vol. 1, no.1, pp. 177-190.

Cox, R.; Muthitacharoen, A.; Morris, R. (2002): Serving DNS using a peer-to-peer lookup service. *Proceedings of the International Workshop on Peer-to-Peer Systems*, vol. 1, no.1, pp. 155-165.

Danielis, P.; Altmann, V.; Skodzik, J.; Wegner, T.; Koerner, A. et al. (2015): P-DONAS: a p2p-based domain name system in access networks. *ACM Transactions on Internet Technology*, vol. 15, no. 3, pp. 1-20.

DNSSEC (2020): DNSSEC deployment report. http://rick.eng.br/dnssecstat/.

Efficientip (2019): Understanding the critical role of DNS in network security strategy. https://www.efficientip.com/resources/idc-dns-threat-report-2019/.

Eyal, I.; Sirer, E. G. (2014): Majority is not enough: bitcoin mining is vulnerable. *Financial Cryptography*, vol. 1, no. 1, pp. 436-454.

Fang, B. X. (2018): *Cyberspace Sovereignty: Reflections on Building a Community of Common Future in Cyberspace*. Springer Press.

He, G. B; Su, W.; Gao, S.; Yue, J. (2020): Td-root: a trustworthy decentralized DNS root management architecture based on permissioned blockchain. *Future Generation Computer Systems*, vol. 2020, no. 102, pp. 912-924.

Jia, X. D.; Hu, N.; Su, S.; Yin, S.; Zhao, Y. et al. (2020): IRBA: an identity-based cross-domain authentication scheme for the internet of things. *Electronics*, vol. 9 no. 634, pp. 1-21.

Jiang, X.; Liu, M. Z.; Yang, C.; Liu, Y. H.; Wang, R. L. (2019): A blockchain-based authentication protocol for Wlan mesh security access. *Computers, Materials & Continua*, vol. 58, no. 1, pp. 45-59.

Kalodner, H. A.; Carlsten, M.; Ellenbogen, P.; Bonneau, J.; Narayanan, A. (2015): An empirical study of namecoin and lessons for decentralized namespace design. *Proceedings of the 14th Workshop on the Economics of Information Security*, vol. 1, no. 1, pp. 1-23

Karaarslan, E.; Adiguzel, E. (2018): Blockchain based DNS and PKI solutions. *IEEE Communications Standards Magazine*, vol. 2, no. 3, pp. 52-57.

Li, M. H.; Sun, Y. B.; Lu, H.; Maharjan, S.; Tian, Z. H. (2020): Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems. *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 1-13.

Liu, C.; Albitz, P. (2006): *DNS and Bind*. O'Reilly Media Press.

Muhammad, G.; Wang, J. (2020): Blockchain-enabled distributed security framework for next generation IoT: an edge-cloud and software defined network integrated approach. *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 1.

Ramasubramanian, V.; Sirer, E. G. (2004): The design and implementation of a next generation name service for the internet. *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 331-342.

Schomp, K.; Callahan, T.; Rabinovich, M.; Allman, M. (2014): Assessing DNS vulnerability to record injection. *Proceedings of the International Conference on Passive and Active Network Measurement*, vol. 1, no. 1, pp. 214-223.

Song, Y.; Koyanagi, K. (2011): Study on a hybrid p2p based DNS. *Proceedings of the IEEE International Conference on Computer Science and Automation Engineering*, vol. 1, no. 1, pp. 152-155.

The Internet Corporation for Assigned Names and Numbers (2020): TLD DNSEC report. http://stats.research.icann.org/dns/tld_report/.

Tian, Z. H.; Gao, X. S.; Su, S.; Qiu, J. (2020): Vcash: a novel reputation framework for identifying denial of traffic service in internet of connected vehicles. *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 1-9.

Tian, Z. H.; Luo, C. C.; Qiu, J.; Du, X. J.; Guizani, M. (2019): A distributed deep learning system for web attack detection on edge devices. *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1963-1971.

Tian, Z. H.; Shi, W.; Wang, Y. H.; Zhu, C. S.; Du, X. J. et al. (2019): Real time lateral movement detection based on evidence reasoning network for edge computing environment. *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4285-4294.

Trostle, J.; Van, B. B.; Pujari, A. (2010): Protecting against DNS cache poisoning attacks. *Proceedings of the 6th IEEE Workshop on Secure Network Protocols*, vol. 1, no. 1, pp. 25-30.

Tsai, W. T.; Yu, L.; Wang, R.; Liu, N.; Deng, E. Y. (2017): Blockchain application development techniques. *Journal of Software*, vol. 28, no. 6, pp. 1474-1487

Wang, B. W.; Kong, W. W.; Guan, H.; Xiong, N. N. (2019): Air quality forecasting based on gated recurrent long short-term memory model in internet of things. *IEEE Access*, vol. 7, no. 1, pp. 69524-69534.

WebSiteHub (2020): How many domains are there?–Domain name stats for 2020. https://makeawebsitehub.com/how-many-domains-are-there/.

Yin, L. H.; Luo, X.; Zhu, C. S.; Wang, L. M.; Xu, Z. et al. (2019): Connspoiler: disrupting C&C communication of IoT-based botnet through fast detection of anomalous domain queries. *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1373-1384.

Zhang, Z.; Li, Y. B.; Wang, C.; Wang, M. Y.; Tu, Y. et al. (2018): An ensemble learning method for wireless multimedia device identification. *Security and Communication Networks*, vol. 2018, no. 1, pp. 1-9.

Zhou, Q. Y.; Luo, J. J. (2018): The study on evaluation method of urban network security in the big data era. *Intelligent Automation and Soft Computing*, vol. 24, no. 1, pp. 133-138.

Zou, F. T.; Zhang, S. Y.; Pei, B.; Pan, L.; Li, L. S. et al. (2016): Survey on domain name system security. *Proceedings of the IEEE First International Conference on Data Science in Cyberspace*, vol. 1, no. 1, pp. 602-607.