Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 6620236, 20 pages
https://doi.org/10.1155/2021/6620236

Research Article

Blockchain-Based DNS Root Zone Management Decentralization for Internet of Things

Yu Zhang, Wenfeng Liu, Zhongda Xia, Zhongze Wang, Lu Liu, Weizhe Zhang, Hongli Zhang, and Binxing Fang

Harbin Institute of Technology, Harbin, China

Correspondence should be addressed to Wenfeng Liu; 15b903031@hit.edu.cn

Received 23 October 2020; Revised 23 December 2020; Accepted 19 January 2021; Published 8 February 2021

Academic Editor: Hongju Cheng

Domain Name System (DNS) is a widely used infrastructure for remote control and batch management of IoT devices. As a critical Internet infrastructure, DNS is structured as a tree-like hierarchy with single root zone authority at the top, which puts the operation of DNS at risk from single point of failure. The current root zone management is lack of transparency and accountability, since only the root zone file is published as the final outcome of operations inside the root zone authority. Towards distributed root zone operation in DNS, this paper presents a blockchain-based root operation architecture—RootChain, composed of multiple root servers. On the basis of maintaining the single root authority for top-level domain (TLD), RootChain decentralizes TLD data publication by empowering delegated TLD authorities to publish authenticated data directly. The transparency and accountability of root zone operation are attained by smart-contracting the whole life cycle of TLD operation and logging all operations on the chain. RootChain is transparent to recursive/stub resolver and DNS/DNSSEC-compatible. A proof-of-concept prototype of RootChain has been implemented with Hyperledger Fabric and evaluated by experiments.

1. Introduction

The rapidly developing Internet of Things devices have touched every corner of our life. Nowadays, remotely controlling IoT devices and acquiring data from IoT devices are the popular requirements. In order to efficiently manage a large number of IoT devices across platforms and scenarios, it is currently a feasible method to allocate a DNS name to each device as a globally unique identifier.

Users resolve the domain name of the IoT device through the DNS infrastructure to obtain the IP address required for further communication. Throughout the whole interaction process with IoT devices, the reliability and availability of the DNS domain name, especially the root domain name as the name resolution starting point, is the prerequisites.

DNS is structured as a tree-like hierarchy with a single root at the top. Currently, under the functions of Internet Assigned Numbers Authority (IANA), the operation and maintenance of root zone are performed by Public Technical Identifiers (PTI) as the root zone operator and Verisign as the root zone maintainer, respectively, on behalf of the IANA function operator—Internet Corporation for Assigned Names and Numbers (ICANN). Although there are 13 root servers with hundreds of root server mirrors, the centralization of root zone management puts the DNS at risk from single point of failure (SPOF). The root zone management is lack of transparency and accountability, since only the root zone file is released as the result of the management process, the management process is like a black-box, implemented through the collaboration of several inner root zone management partners.

In order to reduce the single point of failure of DNS root, there have been some typical and widely used blockchain-based DNS decentralization solutions. Unfortunately, they all have compatibility issues with the current DNS, which makes it difficult to be further widespread used. Namecoin

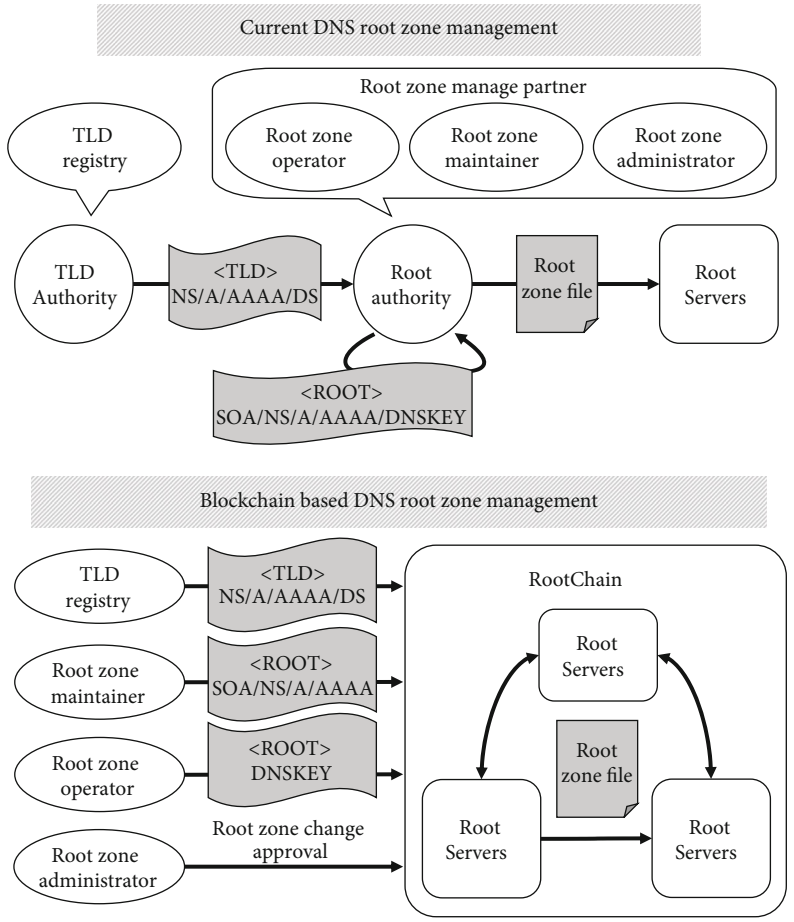


Figure 1: Comparison of the root zone data publication process between DNS and RootChain.

[1] and its successor Blockstack [2] are two notable solutions that have been deployed, which provided decentralized name registration and resolution service based on blockchain. Blockstack creates a new namespace independent from the current DNS, leading to namespace split. In Blockstack, domain names are registered in a first-come-first-serve manner, which is vulnerable to cybersquatting.

To help DNS improve operation transparency and reduce the risk of single point of failure, we propose RootChain, a systematical design for distributed DNS root zone management based on blockchain, which achieves the following goals:

(i) *Goal 1: Uniform Global Namespace*. Retaining a uniform global namespace with the single root authority of current DNS, so to avoid namespace split and cybersquatting.

(ii) *Goal 2: Anti-SPOF Risk*. Distributing root zone operation cross multiple physical nodes under different operators, so to avoid the single point of failure.

(iii) *Goal 3: Transparency and Accountability*. Supporting the operation of the entire TLD life cycle management, so to provide the transparency and accountability of root zone management process.

(iv) *Goal 4: Compatibility*. Be transparent to recursive resolver and compatible with DNS/DNSSEC at protocol level.

RootChain builds a permissioned blockchain that run by current root server operators. To reduce SPOF risk, RootChain decentralizes root zone management while still retaining a single root authority to maintain uniform global namespace; the key idea of RootChain is to separate delegation and data publication from the root zone management process; the root authority is responsible for TLD delegation, and each TLD authority publishes data for its own TLD through RootChain. The root authority and TLD authorities contribute data to the root zone file together via RootChain. Such cooperation among multiple stakeholders effectively mitigates the SPOF risk in current root zone management. With the operations recorded and publicized on RootChain, the inherent consensus and tamper-proof characteristics of blockchain significantly improve the transparency and accountability of root zone management. A comparison between current root zone management and RootChain (blockchain-based) root zone management is illustrated in Figure 1.

The key contributions are as follows:

(i) Propose a blockchain-based architecture for distributed root zone management, realizing a cooperation

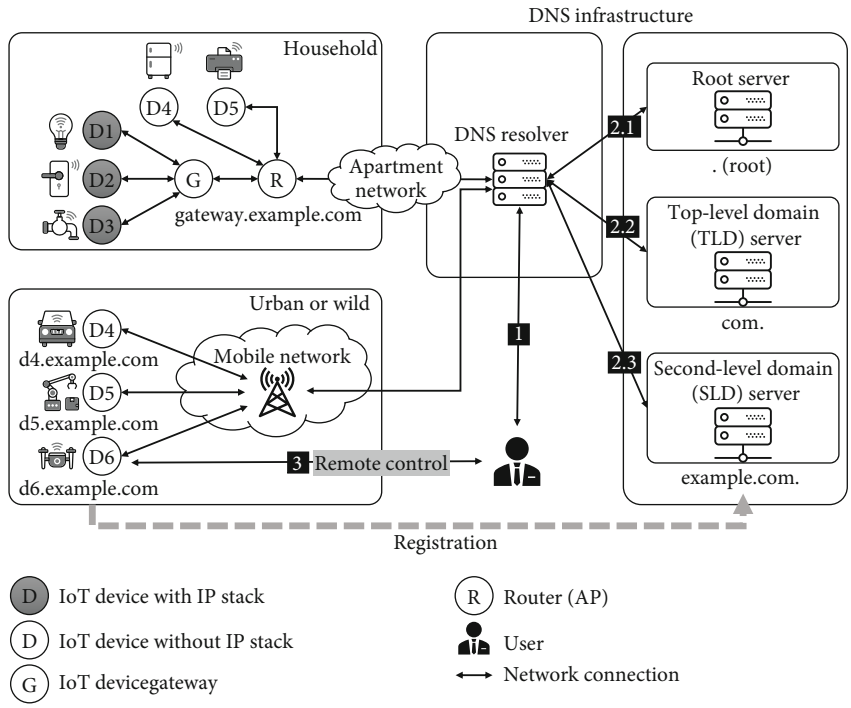


Figure 2: Example scenes of the relationship between DNS and IoT devices. Users identify the network location of the IoT devices by using its domain name of through DNS.

framework for multiple root zone management partners, which is compatible with the current DNS

(ii) Design smart contracts for supporting the operation of the entire TLD life cycle management. We propose a flexible and configurable TLD delegation strategy based on a state machine for different types of TLDs and different business models

(iii) Design, implement, and evaluate a proof-of-concept prototype of RootChain based on Hyperledger Fabric [3]. To the best of our knowledge, RootChain is the first distributed and DNS-compatible system for root zone management

The remainder of this paper is structured as follows. Section 2 introduces the background, including the interaction logic between DNS and IoT as well as the current DNS root zone management model. Section 3 explains the potential risks in the current root zone management model. Section 4 presents the RootChain architecture. Section 5 designs smart contracts. Section 6 discusses the properties of RootChain. Section 7 explains the implementation details. Section 8 evaluates RootChain. Section 9 introduces related work. Section 10 concludes the paper.

2. Background

2.1. The Interactions between IoT and DNS. As shown in Figure 2, we define the model of IoT as a combination of devices that sense and act on the physical environment and provide remote services based on the Internet. There are usually big differences between IoT devices, including differences in hardware architecture, operating systems, and network functions.

From the perspective of the development trend of IoT devices, IoT devices have at least one thing in common, that is, they need to store the current state of the device to a remote server and accept control command from remote users via the Internet. In other words, practical development experiences show that IoT devices usually rely on some centralized cloud services to better interact with users, and users usually interact with IoT devices through cloud servers. The interaction between IoT devices and DNS infrastructure is mainly reflected in the following two typical scenes:

Scene 1. IoT devices collect environmental data through the carried sensors and upload the data to the data server for users to pull and use. In this scenario, IoT devices initiate DNS queries to resolve the IP address of the remote data server.

Scene 2. Users send a control command to the IoT device through the control server to make the device take a series of actions. In this scenario, IoT devices register the domain name used for identifying itself in the authoritative server in advance.

Typical IoT applications, such as indoor smart locks, smart water and electricity equipment, outdoor smart cars, and mechanical equipment, need to uniquely name the device object and then use name resolution system to uniquely identify the device on the network.

Figure 2 shows that the user identifies the network location (IP address) of the IoT device by using the domain name of the IoT device through DNS. The user needs to control the drone through the network for cruise and photography task. The domain name of the drone is identified as http://d6

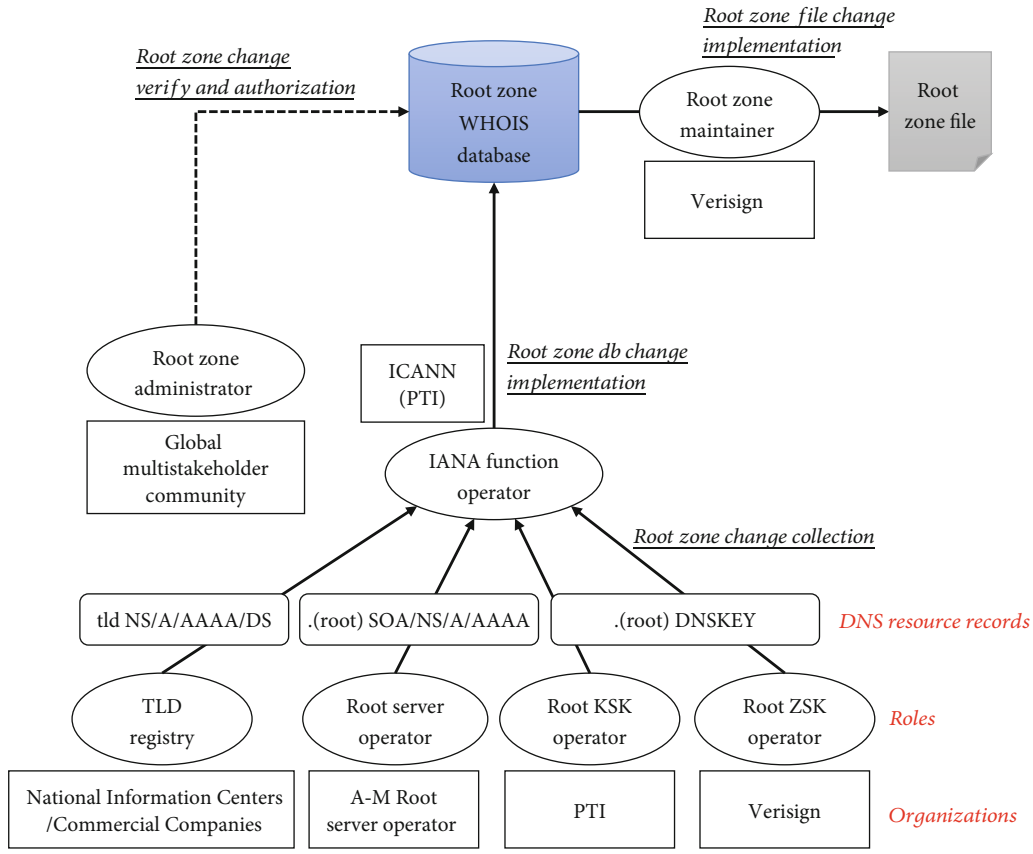


Figure 3: DNS root zone management model—roles and organizations.

.example.com/, and the name is registered in the service application and DNS server in advance.

When remotely controlling the drone to perform a task, the user needs to acquire the IP address of the drone named http://d6.example.com/ through hierarchical structured DNS. DNS root server is the starting point for domain name resolution. The user first queries the root server for the IP address of the top-level domain (TLD) authoritative server of the domain name—com—and then queries the TLD authoritative server for the IP address of the second-level domain (SLD) authoritative server of the domain name—http://example.com/, which is managed by the application service provider. Finally, query the current IP address of the drone from the SLD authoritative server. After the user obtains the IP address of the drone device, they can transmit task commands to the drone device via the Internet.

2.2. DNS Root Zone Management Model. Root zone management involves the processes by which changes are made to the root zone resource records. There are three main roles involved in managing the root zone (Figure 3): IANA function operator, root zone maintainer, and root zone administrator. The IANA function operator is responsible for collecting and processing requests for changes to the root zone data from all participating parties and implementing changes to the root zone database (WHOIS database [4]). The root zone maintainer is responsible for implementing changes to the root zone files based on the root zone database. The root zone administrator verifies whether the above two roles follow the established procedures and policies during the operation process and authorizes the implementation of root zone database changes and implementation of root zone file changes.

The three roles of IANA function operator, root zone maintainer, and root zone manager are performed by different organizations. The ICANN organization (performed by its affiliate PTI) plays the role of the IANA function operator, and ICANN legally obtains the right to exercise the function by signing the IANA function contract [5]. IANA has several key functions [6], and we mainly focus on its DNS root zone management functions in this paper. Verisign performs the role of root zone maintainer by signing a root zone maintenance service agreement (RZMA) [7] with ICANN. The role of root zone manager has been performed by the US Commerce Department's National Telecommunications and Information Administration (NTIA) since 2000. In 2016, NTIA signed a withdrawal contract [8] and announced that this role would be subsequently transferred to the ICANN board of directors composed of global multistakeholders.

The IANA function operator is responsible for collecting root zone changes from the downstream roles that provide root zone data and submitting them to the root zone database after summarizing them. From the perspective of the data provider, the root zone data includes three types: (1) top-level domain data (tld NS-Records, A/AAAA-Records, and DS-Records), (2) root server data (root NS-Records,

A/AAAA-Records, and SOA-Records), and (3) root key data (root DNSKEY-Records). The role of providing top-level domain data is TLD registry, which is usually played by national information centers (ccTLDs) or commercial companies (gTLDs). The role of providing root server data is the root server operator, played by current 12 root server operators [9] around the world. The roles that provide root key data are root KSK operator and root ZSK operator, played by PTI and Verisign, respectively.

3. Problem Description

3.1. Threat Modeling for DNS Root Zone Management. By analyzing the current DNS root zone management model introduced above (Section 2.2), we can find that there are two main potential threats in the root zone management process.

Single point of failure risk. Related roles (TLD registry, root server operator, root KSK/ZSK operator), respectively, submit part of the root zone data that they are responsible for, but all submitted data need to be aggregated to the role of IANA function operator before it can be written to the root zone database according to predetermined procedures and strategies. This means that IANA function operator is the bottleneck of the entire root zone management process. When IANA function operator has an availability failure or misconfiguration, it will cause subsequent name resolution failure of hundreds of root servers that rely on this piece of root zone data file.

Lack of transparency and accountability. Root zone management details inside the root zone management partners are mostly hidden to the public. Only the root zone file is published as the final management outcome. The management process includes the interaction between multiple roles. For example, the IANA function operator needs to verify the authenticity of its identity before receiving the root zone changes submitted by the TLD registry, all root zone data changes need to be reviewed by the root zone administrator before they can be updated to the root zone database. The root zone maintainer can only use the data reviewed by administrator to generate root zone files. However, the root zone file does not contain the records of the above interaction process, which means when the root zone data has misconfiguration, and the public cannot understand at which step the root zone data problem occurs. The lack of transparency has further led to a lack of accountability of root zone management. When there is a problem with the root zone data, you can only rely on the self-correction and self-examination between root zone management partners, which creates a space for shirking responsibility to a certain extent.

3.2. Assumption. This paper assumes that only the IANA function operator role can authorize top-level domains to ensure a globally unique DNS namespace, and all root server operators and all users admit the TLD delegation decisions made by IANA function operator.

This paper assumes that the root management partners may provide service in an arbitrary and unpredictable way. This may be a result of compromised server, misoperation/-misconfiguration, software/hardware failure, etc.

4. System Design

4.1. Architecture of RootChain. RootChain is a permissioned blockchain system comprised of root servers running blockchain nodes. By separating the delegation of TLDs from the data publication of TLDs, RootChain distributes DNS root zone operation while retaining a single root authority. The transactions of root zone operation are submitted to RootChain by the root authority and TLD authorities and are recorded in the blockchain ledger cross all root servers. The root zone file can be derived from the blockchain ledger consisting of transactions of root zone operation. As shown in Figure 4, there are four major roles in RootChain:

Root authority. There is only one single root authority in RootChain. The root authority here covers the 3 roles of IANA function operator, root zone maintainer, and root zone administrator in current DNS. Besides publishing root domain data, the root authority delegates a TLD to a TLD authority and transfers and revokes existing TLD delegations according to contracts signed among the root authority and TLD authorities.

TLD authority. The TLD authority in RootChain refers to the role of TLD registry in DNS. A TLD is delegated to a TLD authority by the root authority through submitting TLD delegation transaction to RootChain. TLD authority publishes the data of authoritative servers for the delegated TLD, renews the delegated TLD before the delegation expires, redeems the delegated TLD after the delegation expires, and performs operations according to the delegation contract signed with the root authority.

Root server operator. A root server operator is responsible for managing root servers. Besides providing name resolution service as authoritative name servers, root servers in RootChain are also responsible for the root zone management according to the blockchain transactions. Root server operators guarantee the physical security and constant availability of root servers, ensure that root servers continue to participate in RootChain, and provide nondiscriminatory name resolution service to users.

User. A user of RootChain corresponds to a recursive resolver and stub resolver in the current DNS, which sends a standard DNS query to any root server to obtain the NS and A/AAAA records of TLDs.

4.2. Root Zone Operation for TLDs. There are three types of root zone operations for TLDs in RootChain, as shown in Figure 5.

Delegation operations. The root authority delegates TLD to TLD authorities with *delegation publication* operations, by binding TLDs to the public keys of TLD authorities. The root authority may transition or revoke TLDs with *delegation transition* operations or *delegation revocation* operations; TLD authorities may renew and redeem the delegation of TLDs with *delegation renewal* operations and *delegation redemption* operations.

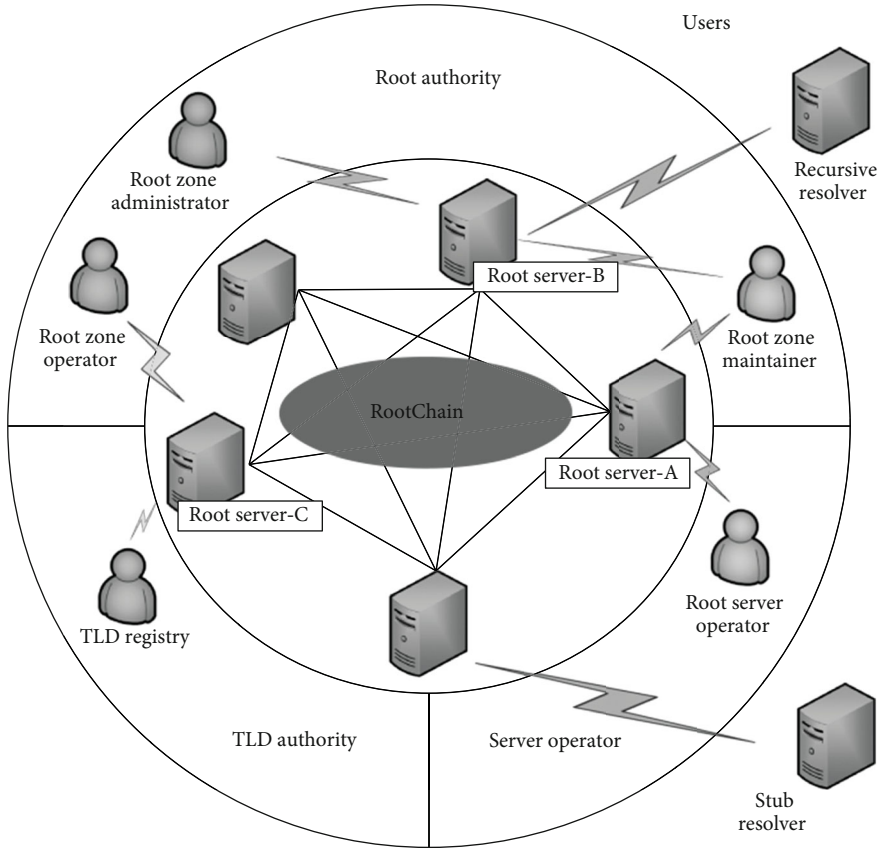


Figure 4: RootChain architecture.

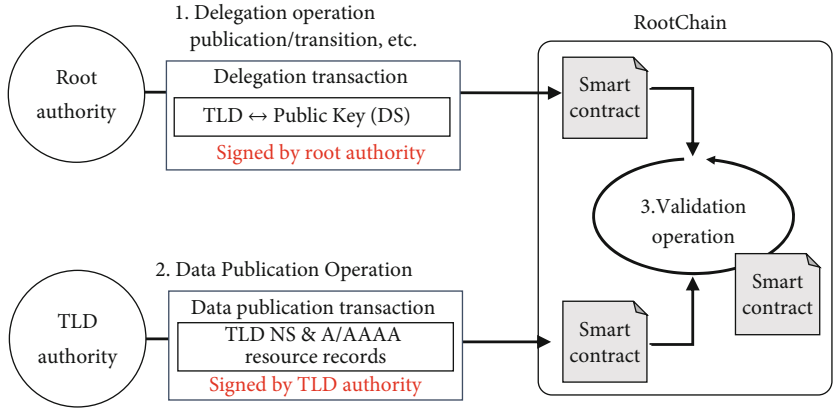


Figure 5: Root zone operation for TLDs.

The smart contracts that determine the *delegation policy* are specified in the corresponding delegation operation. It should be noted that a delegation operation may not always take effect immediately, and certain operations need to wait for confirmations from other authorities to take effect, which we will be explained further in Section 4.3.

Data publication operations. A TLD authority publishes the NS/A/AAAA resource records for the delegated TLD by sending signed data publication operations to RootChain. A data publication operation is valid and will be used as part of the root zone file only if it carries the proper signature signed by the corresponding TLD authority. Note that a new data publication operation for a TLD will overwrite the previously published data.

Validation operations. Root servers validate the effectiveness of the delegation operation and the data publication operation by executing smart contracts in the validation operation. Smart contracts enforce the root authority and all TLD authorities to only act in accordance with established policies. More details on smart contracts will be presented in Section 5.

4.3. Delegation Policies. A delegation policy is specified in a TLD delegation publication operation and determines how

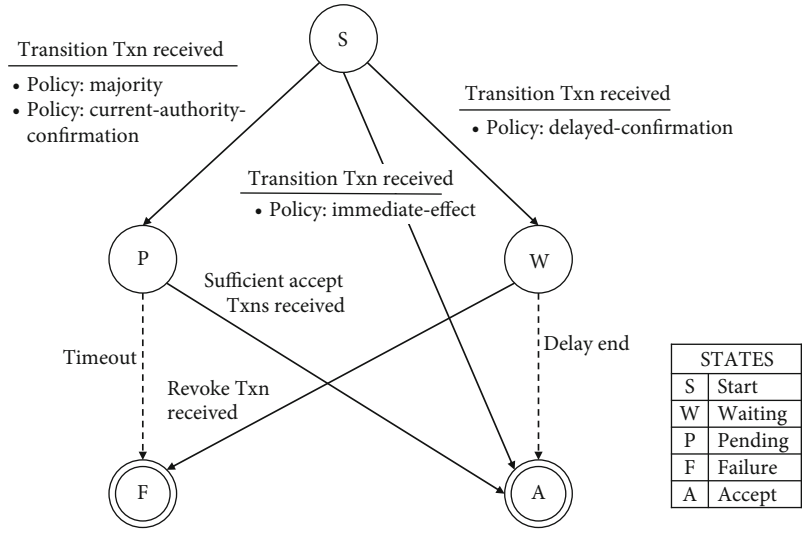


Figure 6: State transition diagram of TLD delegation operations.

the delegation of the TLD may be later operated. We have defined four delegation policies:

Delayed-confirmation policy: an operation will take effect only if no objections are received within a grace period. This policy is usually used in scenarios where disputes and frictions occur between root and TLD authorities; delayed acknowledgement brings a grace period to the delegation operation; during the grace period, the dispute may be resolved through offline negotiations among the authorities.

Majority policy: an operation takes effect only if confirmations are received from more than half of the TLD authorities in RootChain. This policy applies to scenarios that require more democratic and collective decision-making.

Current-authority-confirmation policy: when a TLD has been delegated to an TLD authority, subsequent operations such as delegation transition and revocations must be approved by the current TLD authority. This policy focuses on protecting the rights of the delegated TLD authority.

Immediate-effect policy: an operation submitted by an authority may immediately take effect. This policy applies to delegation operations accepted by default; for example, the former authority of a TLD is generally considered to have the right to redeem the TLD preferentially during a grace period.

The above four delegation strategies are the expression of different degrees of trust relationship between members in different situations. The majority policy applies to situations that require collective decision-making and need to be resolved when the authorization of a top-level domain name is disputed, but a single entity's interests may be damaged by collective decision-making. Delayed-confirmation policy is used in situations of high mutual trust among members. Delegation operations that have been initiated take effect automatically for a certain period. This strategy reduces the process of interaction, but when there are a few of destroyer in the environment, the management process would be confused. The current-authority-confirmation and immediate-effect policy are applied to situations that require a high degree of protection for delegated entity, who have full control over the top-level domains delegated to them. This strategy makes root domain management more distributed and weakens the root authority's control over the top-level domain authority.

RootChain implements the above policies through smart contracts. Different from the current DNS without the concept of delegation policy in the DNS protocol specifications, RootChain can flexibly express more rich and finer-grained delegation policies according to the real-world scenarios. This is one significant advantage of RootChain over the current DNS.

Figure 6 shows the state transition of a TLD in RootChain resulting from the execution of TLD delegation operations under different policies. Take the delegation transition operation as an example, the TLD is in the S state before a coming delegation transition operation, indicating that the current TLD delegation information is in effect. When RootChain receives the delegation transition operation, the state of the TLD begins to change:

(i) The TLD enters the W state, if a TLD uses the delayed-confirmation policy in the corresponding delegation publication operation. If RootChain does not receive any objection to the delegation transition during the grace period, the delegation transition will automatically enter the "accept" state after the grace time expires, and the delegation transition operation takes effect. If the RootChain receives the root authority's objection during the grace period, the TLD delegation transition operation immediately enters the "failure" state, and the delegation status of the TLD remains unchanged

(ii) The TLD enters the P state, if the TLD uses the majority policy. If RootChain receives confirmations from more than half of all TLD authorities during the delegation transition window, the delegation transition operation takes effect. If RootChain fails to receive enough confirmations within the window

period, the delegation transition operation enters the "failure" state, and the delegation status of the TLD remains unchanged

(iii) If the TLD uses the current-authority-confirmation policy, then same as the majority policy, the TLD name also enters the P state. If RootChain receives a confirmation from the current authority of the TLD during the timeout period, the operation takes effect. Conversely, if the RootChain fails to receive such a confirmation within the timeout period, the delegation transition operation enters the "failure" state, and the delegation status of the TLD remains unchanged

(iv) If the TLD uses the immediate-effect policy, the delegation change will take effect immediately without the need for any confirmation after the root authority initiates the delegation transition

4.4. Transactions. Delegation operations and data publication operations are stored in RootChain in the form of *transactions*. There are two types of transactions.

Delegation transaction: the delegation transaction is stored in the "DelegationTxn" data structure (Table 1). The "Previous Txn" field stores the index of the transaction containing the preceding operation associated with this delegation operation. The type of the delegation operation is stored in the "Operation Type" field. There are five types of delegation operations: publication, revocation, transition, renewal, and redemption. The publication operation binds a TLD "Auth Name" with a public key "Auth Key." According to different business scenarios, different TLD delegation transition, revocation, and redemption policies may be specified, and the specific policies are all stored as fields of "DelegationTxn," with each field named with the "Policy" suffix. RootChain currently supports four delegation policies, as described in Section 4.3. The validity period of the publication operation starts from "Valid From" and ends at "Valid To." If the delegated TLD exceeds the validity period, the authority will need to extend the validity period of the TLD by sending a renewal transaction. The renewal window is within "Renewal Grace Period" days since the "Valid To" date. If the renewal fails or is absent, the TLD will enter the redemption state, and the authority needs to redeem the TLD by following the redemption policy within "Redemption Grace Period". Beyond the redemption window, the root authority will send a revocation transaction, and the transaction is received and verified by RootChain, eventually marking the TLD to be revoked. "Signature" is the signature of the entire delegation transaction, being signed using the private key of the appropriate authority to provide the authenticity and integrity of the delegation transaction.

Data publication transaction: the data publication transaction is stored in the "DomainTxn" data structure (Table 2), and the "Previous Txn" field stores the index of the transaction containing the preceding delegation operation associated with this data operation. The type of data operation is stored in the "Operation Type" field, and there are two types including data publication and data deletion. A resource record "(NAME, TTL, CLASS, TYPE, RDATA)" is a 5-tuple representing domain name data, which is consistent with the DNS protocol. The "RRSIG" field is a digital signature for the resource record, which is consistent with the DNSSEC protocol. "Signature" is the signature of the entire data publication transaction, using the private key of the appropriate authority.

Table 1: Definitions of TLD delegation transaction.

DelegationTxn	
PrevTxn	Associated preceding transaction
OperationType	Type of delegation operation
DomainName	Delegated domain name
DelegatedKey	Delegated public key
ValidFrom	Start of validity period
ValidTo	End of validity period
TransitionPolicy	Delegation transition policy
TransitionWindow	Delegation transition window
RenewalGracePeriod	Delegation renewal grace period
RevocationPolicy	Delegation revocation policy
RedemptionPolicy	Delegation redemption policy
RedemptionGracePeriod	Delegation redemption grace period
TimeStamp	Timestamp of transaction creation
TxnSignature	Signature on the transaction

Table 2: Definitions of TLD data publication transaction.

DomainTxn	
PrevTxn	Associated preceding transaction
OperationType	Type of delegation operation
NAME	TLD name
TTL	Time-to-live (DNS protocol)
CLASS	Record class (DNS protocol)
TYPE	Record type (DNS protocol)
RDATA	Record data sets (DNS protocol)
RRSIG	Signature of the resource record (DNSSEC protocol))
TimeStamp	Timestamp of transaction creation
TxnSignature	Signature on the transaction

5. Smart Contracts

5.1. Overview. As shown in Table 3, a smart contract describes operations involved in the three phases described in Section 4.2, including delegation operations (publication, transition, revocation, renewal, and redemption), data publication operations, and validation operation. Delegation operation means the binding of a top-level domain to an entity; after the binding operation, the entity becomes the registry of that top-level domain, also known as the authority of that top-level domain. The data publication operation binds the delegated top-level domain to its authoritative server. The binding information includes the name and IP address of

Table 3: Semantics of RootChain smart contract.

RootChain operation	
Smart contract	Example Initiator → Output
DelegationPublication	RA → <TLD,TAPubKey>RA
DelegationTransition	TA → <TLD,TAPubKey>TA
DelegationRevocation	RA → <TLD,∅>RA
DelegationRenewal	TA → <TLD,ValidTo>TA
DelegationRedemption	TA → <TLD,TAPubKey>TA
DataPublication	TA/RA → <ZoneData>(TA/RA)
DelegationValidation	RootChain → true/false
DataValidation	RootChain → true/false
RevokeOP	RA/TA → <TLD,OP-ID>(RA/TA)
ConfirmOP	RA/TA → <TLD,OP-ID>(RA/TA)

the authoritative server. The verification operations are primarily used to verify the legitimacy of delegation operations and data publication operations.

There are two additional operations involved in the execution of smart contracts, namely, the *revoke* operation (RevokeOP) and confirmation operation (ConfirmOP). A revoke operation is submitted by the initiator of another operation to revoke the previously initiated operation, and such a revoke operation is submitted as a result of objections from other authorities to the submitted operation. A confirmation operation stands for the confirmation on a submitted operation, and an authority conducts a signed confirmation operation for a submitted operation that requires confirmation from other authorities to take effect.

5.2. Delegation Smart Contract. Delegation smart contracts are implemented by the function *DelegationOperation()* as described by Algorithm 1. The delegation publication operation is initiated by the root authority (RA in Table 3) specifying the TLD ("AuthName") to be delegated and the authoritative public key ("AuthKey") bound to the TLD. The delegation publication operation also needs to specify the validity period (from "ValidFrom" to "ValidTo"), the authoritative transition policy ("TransitionPolicy"), and the policies for subsequent delegation operations that may be submitted after the current delegation of the TLD expires. The delegation transition operation is initiated by the current TLD authority ("TA" in Table 3) and is implemented by changing the authoritative public key bound to the TLD ("AuthKey"). The delegation revocation operation resets the authoritative public key bound to the TLD ("AuthKey" = ∅). The delegation renewal operation extends the authoritative TLD valid period by resetting the end of valid period date of the delegated TLD ("ValidTo"). Delegation redemption operation rebinds the TLD with the previous public key ("AuthKey"), if the TLD's authoritative public key has been reset within a certain period ("RedemptionPolicy").

5.3. Data Publication Smart Contract. Data publication smart contract is implemented by the function *DataPublicationOperation()* as described by Algorithm 3. The data publishing operation publishes various types of domain name data including NS records and A/AAAA records. Domain data is stored in NAME, TTL, CLASS, TYPE, and RDATA fields, and the signature is stored in the RRSIG field, corresponding to the fields with the same names in DNS/DNSSEC protocol specifications.

5.4. Validation Smart Contract. Validation smart contracts check the validity of submitted transactions. There are currently two types: DelegationValidation and DataValidation smart contract. Two forms of checks are performed to determine the validity of a transaction: transaction format check and transaction effectiveness check. To pass the transaction format check, the value filled in each field needs to conform to the specifications. For example, the timestamp of a transaction must be within the valid period, and the digital signature of the transaction should match the transaction content. To pass the transaction effectiveness check, i.e., to determine whether the operation in this transaction should take effect, the preceding transaction must be valid, and the current transaction must satisfy the policy set in the preceding transaction. A preceding transaction is the transaction whose outcome directly influences the operation in this transaction. For example, the delegation publication transaction of a TLD is the preceding transaction of a delegation transition transaction of the TLD if no other delegation transactions are submitted in-between, and in this case, only when the delegation publication transaction of the TLD is valid and the TLD is in the validity period, it is possible for the subsequent delegation transition transaction to pass the effectiveness check. Generally speaking, the transaction effectiveness check performs association checks on the validity of all transactions in the partial order set formed by transaction dependencies.

The smart contract that verifies the validity of delegation transactions is implemented by the function *DelegateVerification()* (Algorithm 2), and the smart contract that verifies the validity of a data publication transaction is implemented by the function *DataValidation()* (Algorithm 4). For the format check on data publication transactions, in addition to meeting the requirements already mentioned, all data fields inherited from the DNS protocol and the DNSSEC protocol must also conform to the corresponding protocol specifications. Also, note that in both functions, the effectiveness check is implemented by invoking the smart contract in the preceding delegation transaction.

5.5. A Detailed Example: Life Cycle of One TLD. The life cycle of a TLD includes three major phases: delegation phase, data publication phase, and name resolution phase. Figure 7 shows how the full life cycle of a TLD is managed using smart contracts. In this example, ICANN is the root authority, the TLD registry represents a TLD authority, and the user is a user of the RootChain name resolution service.

In delegation phase (step 1), the root authority delegates the TLD to a certain TLD registry. To initiate this phase, the TLD authority sends a delegation publication request to RootChain (step 1.1), and the root server that receives this request executes the smart contract (Algorithm 1) to validate

```
Input:
    op_type, auth_name, auth_key, valid_from, valid to,
    trans_policy, redempt_policy, revoke_policy, prev_txn
Output:
    DelegationTxn
1: DelegationTxn.OperationType == op_type
2: if op_type != publication then
3:     DelegationTxn.PrevTxn = prev_txn
4: end if
5: if op_type == publication then
6:     DelegationTxn.AuthName = auth name
7:     DelegationTxn.AuthKey = auth key
8:     DelegationTxn.ValidFrom = valid from
9:     DelegationTxn.ValidTo = valid to
10:    DelegationTxn.TransitionPolicy = trans policy
11:    DelegationTxn.RevocationPolicy = revoke policy
12:    DelegationTxn.RedemptionPolicy = redempt policy
13: end if
14: if op_type == transformation then
15:    if DelegationTxn.ValidTo < CurrentTime() then
16:        DelegationTxn.AuthKey = auth_key.
17:    end if
18: end if
19: if op_type == Revocation then
20:    DelegationTxn.AuthKey =∅
21: end if
22: if op_type == Renewal then
23:    DelegationTxn.ValidTo = Valid_to
24: end if
25: if op type == Redemption then
26:    if auth_key == DelegationTxn.PrevTxn.AuthKey then
27:        DelegationTxn.AuthKey = auth_key
28:        Auth.ValidFrom = valid_from
29:        Auth.ValidTo = valid_to
30:    end if
31: end if
32: DelegationTxn.TimeStamp = CurrentTime()
33: DelegationTxn.Signature = GenSig()
34: return DelegationTxn
```

Algorithm 1. DelegateOperation().

the contents of the request and then generate the delegation publication transaction (DelegationTxn in step 1.2). The transaction is then handed over to the ordering server to reach a global consensus (steps 1.3 and 1.4) and packaged into data blocks (step 1.5). The consensus server sends the already wrapped data block containing the delegation publication transaction to root servers who store the block in the ledger database (step 1.6).

In data publication phase (step 2), a TLD authority publishes the domain name data for its TLD. Since only the resource record of a delegated TLD published by the TLD authority may take effect, step 2 must take place after step 1 for the same TLD. In this phase, the TLD authority sends a data publication request to the RootChain (step 2.1), and the root server that receives the request executes a data publication smart contract (Algorithm 3) to validate the contents of the request and generate a data publication transaction DomainTxn (step 2.2) which is then passed to ordering nodes where consensus is reached (steps 2.3 and 2.4). Subsequent operations (step 2.5 and step 2.6) are the same as those in step 1 and will not be described here. For compatibility reasons, each root server operator periodically checks the validity of the committed domain name data (step 2.7) by executing the validation smart contract (Algorithm 4). Root server operators read verified data publication transactions from RootChain that contain root zone data (including root DNSKEY resource records, NS resource records, and A, AAAA resource records associated with NS) and TLD data (TLD DS resource records, NS resource records, and A, AAAA resource records related to NS records). Then, root server operators write the validated root domain data and the TLD data into the root zone file (step 2.8). The format of the zone file is compatible with the current mainstream authoritative server software.

In name resolution phase (step 3), a user resolves a published TLD name by sending a DNS query. The user obtains

```
Input:
    CurDelegationTxn
Output:
    bool
1: if CurDelegationTxn has invalid field value then
2:     return false
3: end if
4: if prevtx !=; then
5:     verify_key = prevtx.AuthKey
6:     sig = CurDelegationTxn.Signature
7:     if VeriSig(verify_key, sig) != true then
8:         return false
9:     end if
10:    /* validity period of prevtx needs to cover the validity period of CurDelegationTxn */
11:    if prevtx.ValidFrom < CurDelegationTxn.ValidFrom or prevtx.ValidTo > CurDelegationTxn.ValidTo then
12:        return false
13:    end if
14:    if op_type == Transition or Revocation then
15:        if ConfirmOPs of CurDelegationTxn do not satisfy prevtx.TransitionPolicy/RevocationPolicy then
16:        return false
17:        end if
18:    end if
19:    if op_type == RenewalPolicy then
20:        if CurDelegationTxn.TimeStamp - prevtx.TimeStamp > prevtx.RenewalGracePeriod then
21:            return false
22: end if
23: end if
24: if op_type == RedemptionPolicy then
25:            if (ConfirmOPs of CurDelegationTxn do not satisfy prevtx.RedemptionPolicy) or (CurDelegationTxn.TimeStamp - pre-
vtx.TimeStamp > prevtx.RedemptionGracePeriod) then
26:            return false
27:        end if
28:    end if
29:    DelegationVerification(prevtx)
30: end if
31: return true
```

Algorithm 2. DelegateVerification().

```
Input:
    op_type; name, ttl, type, rdata_sets; rrsig, prev_txn
Output:
    DomainTxn
1: DomainTxn.OperationType == op_type
2: DomainTxn.PrevTxn == prev_txn
3: if op_type == publication then
4:     DomainTxn.NAME = name
5:     DomainTxn.TTL = ttl
6:     DomainTxn.TYPE = type
7: end if
8: DomainTxn.RDATA = rdata_sets
9: DomainTxn.RRSIG = rrsig
10: if op_type == deletion then
11:     DomainTxn.NAME =∅
12: end if
13: DomainTxn.TimeStamp = CurrentTime()
14: DomainTxn.Signature = GenSig()
15: return DomainTxn
```

Algorithm 3. DataPublicationOperation().

the TLD name resource record that has been committed on the RootChain by initiating a standard DNS query to the root server. In Figure 7, the user queries the NS record of the TLD (step 3.1). After the root server receives the query, the data validity check (Algorithm 4) is performed according to the smart contract. Only the resource record in transactions that pass the validity check is returned to the user in the form of DNS reply (step 3.2).

6. Discussions

6.1. Architecture Features. RootChain improves the operation transparency and accountability of the root zone by smart-contracting the whole life cycle of TLD operation and logging all operations on the RootChain blockchain ledger. RootChain reduces the risk of single point of failure in the root zone data publication and distribution process by allowing delegated TLD authorities to publish the authoritative data for delegated TLDs directly through the RootChain. Meanwhile, RootChain retains the single root authority (ICANN) as the single trust anchor, ensuring compatibility with the current

```
Input:
    CurDomainTxn
Output:
    bool
1: if A certain field in CurDomainTxn has an invalid value
then
2:      return false
3: end if
4: /* check if the values conform to DNS/DNSSEC protocol */
5: if CurDomainTxn.[NAME/CLASS/TTL/TYPE/RDATA] do not conform to DNS protocol specification then
6:      return false
7: end if
8: if CurDomainTxn.RRSIG do not conform to DNSSEC protocol specification then
9:      return false
10: end if
11: DelegationTxn = CurDomainTxn.PrevTxn
12: /* check signature of CurDomainTxn */
13: if DelegationTxn!=∅; then
14:      verify_key = DelegationTxn.AuthKey
15:      sig = CurDomainTxn.TxnSignature
16:      if VeriSig(verify_key, sig) != true then
17:          return false
18:      end if
19:      if DelegateVerification(DelegationTxn) == false then
20:          return false
21:      end if
22: end if
23: return true
```

Algorithm 4. DataValidation().

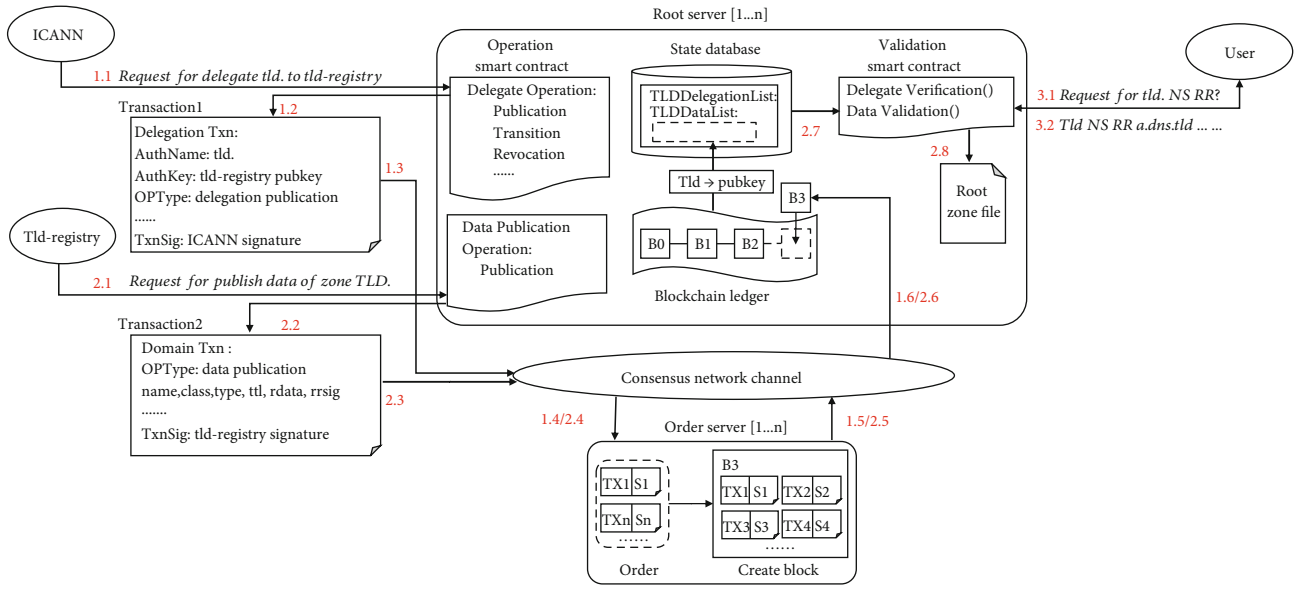


Figure 7: A demonstration of the life cycle of a TLD managed with operations that depend on smart contracts. (1) The root authority delegates a TLD. (2) A TLD authority publishes TLD domain data. (3) A user resolves a delegated TLD.

DNS. Corresponding to the four goals introduced in Section 1, the RootChain system has reached the following four features:

One global namespace with a single root authority. RootChain keeps ICANN as the *single* root authority, which is crucial for the compatibility with the current DNS. In RootChain, all TLD delegation transactions must be initiated by the root authority to guarantee the uniformity of namespace. This feature of RootChain avoids namespace split and cybersquatting.

Distributed root zone operation. RootChain realizes root zone operation decentralization in two aspects: (1) root zone data publication decentralization: in RootChain, root zone data is published through a blockchain network. The data for each TLD is published directly by the corresponding TLD authority, instead of through the root authority in the current DNS. (2) Root zone data distribution decentralization: each node with access to the RootChain blockchain ledger may obtain a complete and accurate copy of the root zone file by examining the operations and data stored in RootChain, instead of by retrieving the root zone file published by the single point of source—the root authority in the current DNS.

Management of the entire TLD life cycle. RootChain supports the management of the entire life cycle of a TLD through the use of smart contracts. The life cycle of a TLD mainly includes delegation phase, data publication phase, and name resolution phase. In delegation phase, the TLD is delegated to a TLD authority, and the policies for subsequent operations are determined; in data publication phase, the domain name data of a TLD is published by and only by the TLD authority; in name resolution phase, the latest effective domain name data is returned to resolvers as DNS reply.

Be transparent to recursive/stub resolvers. RootChain is compatible with DNS/DNSSEC in terms of protocol specifications. RootChain stores domain name data according to DNS/DNSSEC protocol specifications (i.e., in the form of NS/A/AAAA/DS/RRSIG records etc.), from which a complete and accurate root zone file can be composed, and users can receive name resolution service in a DNS/DNSSEC-compliant way.

6.2. Potential Risks. RootChain is a permissioned blockchain architecture. The characteristic of the permissioned blockchain is that all nodes know each other's identity (by their certificates), so there are some certain amount of underlying trusts between nodes. All nodes need to be confirmed by the access control mechanism before they can enter the blockchain network and invoke operations. Therefore, RootChain is insufficient to guard against clients that have already permitted to the blockchain network from maliciously disrupting the system, e.g., clients write large amounts of garbage data to blockchain network. For such a malicious client node, RootChain can ensure that the client node can only have a limited malicious impact on the blockchain network by limiting the invoke rate of each node. RootChain can also identify abnormal behavior through transaction listening mechanisms and then deny access to invoke operations within the blockchain network through access control mechanisms.

6.3. Interoperability Influence. DNS, as Internet infrastructure, interacts with many other systems, such as CDN. The RootChain only deals with the management process of the DNS root zone and does not affect the operation of the DNS system components. Therefore, other protocols or services that interact with DNS components are not affected by RootChain. Besides, DNS has some related security mechanisms, such as DNSSEC and DNS encryption (DoT and DoH). For DNSSEC, RootChain can meet DNSSEC requirements by adding DNSSEC-related resource records in the zone management process, such as DS resource records and DNSKEY resource records to data publication operations. For DNS encryption, DNS encryption is worked for the communication link between the DNS stub resolver and the DNS recursive resolver. RootChain works for the management process of the root zone, which is used for the DNS authoritative server. Thus, RootChain does not affect DNS encryption mechanisms.

7. System Implementation

We implemented RootChain based on the open source project Hyperledger Fabric (version 1.4) [3]. Fabric is a widely used open source blockchain infrastructure project led by IBM that provides common components for blockchain-based application development, such as peer-to-peer (P2P) networking overlay and consensus algorithms. Figure 8 shows the implementation architecture of RootChain. Three main modules provide the basic functionalities:

Blockchain ledger module. The blockchain ledger module provides storage and access to all the transactions issued by authorities, including transactions that contain approved policies, and built-in smart contracts for performing delegation and data validity verification. Each transaction stored on the RootChain is signed by its publisher's private key; hence, any attempt to tamper with stored data can be detected.

Consensus module. The consensus module builds a decentralized network composed of root servers that facilitate the execution of consensus algorithms among all root servers. The consensus network nodes, i.e., root servers, communicate with each other over a peer-to-peer (P2P) network. All nodes execute the same consensus algorithm to ensure global consistency on a public transaction sequence, i.e., a public blockchain ledger.

Identity module. The identity module manages the identity information of all the root servers in the consensus network. This module registers and issues X.509 certificates to valid root servers (also operators) and root zone management partners. The identity module also provides access control service to the consensus network, granting access only to root servers with valid certificates. The access control policy ensures the authenticity and integrity of the data exchanged in RootChain-related internode communications.

On top of the main modules, the modules of root chain provide services related to root zone management. <Delegation Module> and <Data Publication Module> provide root zone operation services to the root authority and TLD authorities. <Domain Resolution Module> provides name resolution service to users. <Validation Module> provides delegation and domain data validation service to the Domain Resolution Module.

8. Evaluation

The root zone management service provided by RootChain is mainly implemented by combining four basic smart

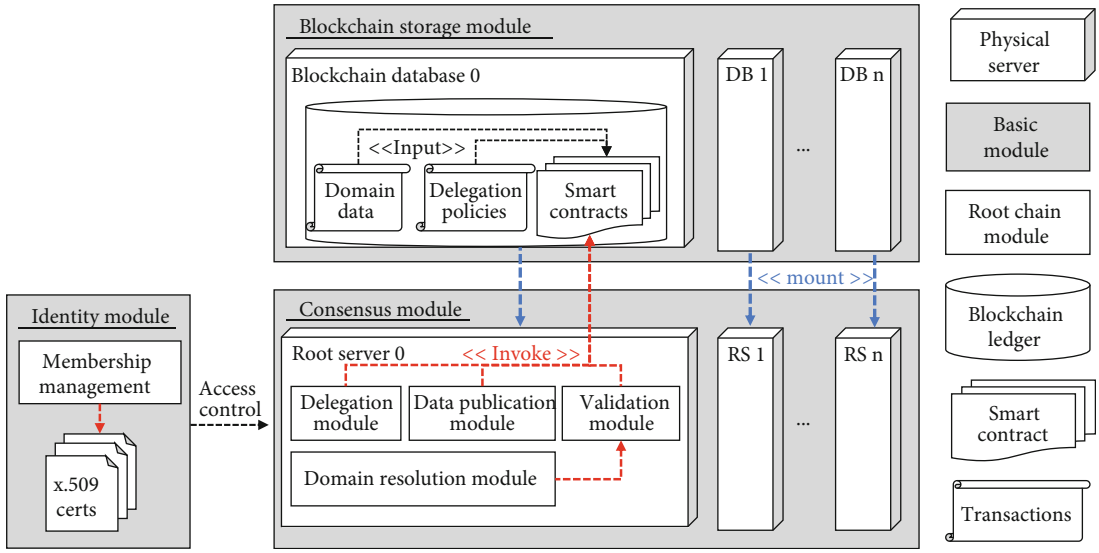


Figure 8: Architecture of system implementation.

contracts (designed in Section 5). Four basic smart contracts are as follows: write delegation (publish delegation), read delegation, write data (publish domain data), and read data. In this section, we will evaluate the execution efficiency and resource consumption of the four basic smart contracts during the operation process of the RootChain system.

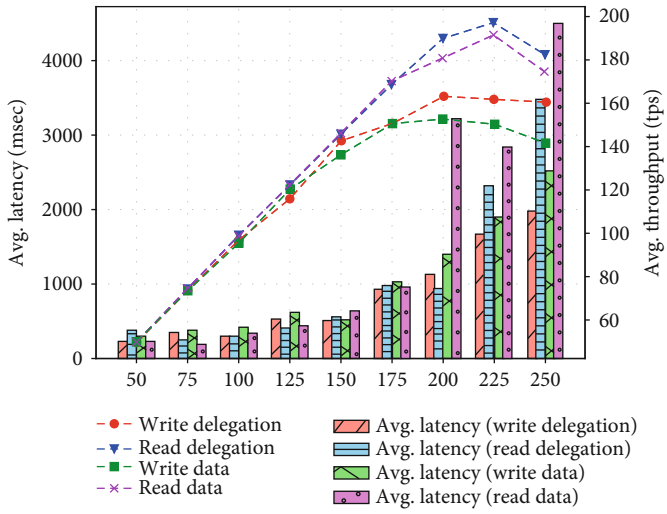
8.1. Test Tools and Test Environment. We use the open source blockchain benchmark tool Caliper [10] to evaluate RootChain prototype system. Caliper allows testers to customize test cases to measure the performance of various blockchain implementations. RootChain runs in a small network composed of 5 physical machines, uses an additional machine to run the Caliper client to initiate smart contract invoke request to the RootChain network, and collects benchmark test results. Each machine that runs RootChain is allocated 32 vCPUs of Intel Xeon(R) CPU E5-2620 v4 at 2.10 GHz and 32 GB of memory. The client machines used to generate requests are allocated with 12 vCPUs and 32 GB of memory. All nodes are connected to a local network with 1000 Mbps bandwidth.

8.2. Latency and Throughput. Figure 9(a) comprehensively shows the change trend of the average delay and throughput of the 4 basic smart contracts under different transaction arrival rates. The results show that the type of smart contract does not affect the changing trend of latency and throughput. Therefore, all subsequent tests use a single smart contract to display the evaluation results.

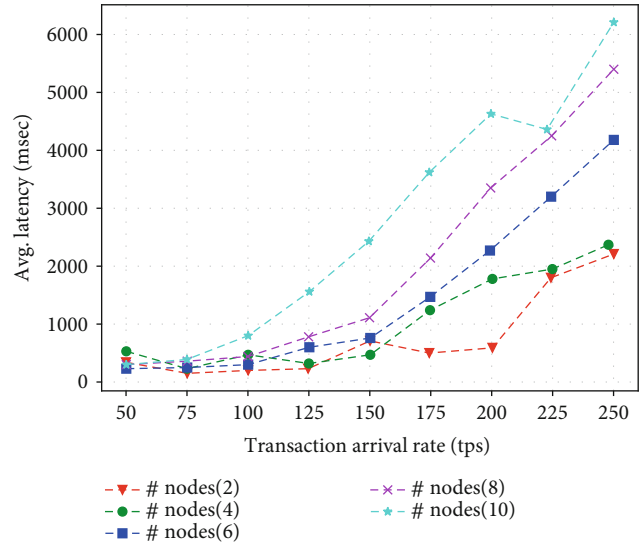
Figure 9(a) shows that the latency increases with the transaction arrival rate, and after passing the critical point, the latency starts to increase rapidly. Combining Figures 9(a)–9(c), it can be found that the arrival of the critical point is affected by two factors, transaction arrival rate (TAR) and network scale. The result of Figure 9(b) shows that when the TAR gradually increases, the larger the network scale, the easier it is to usher in the critical point. For example, when the number of network nodes is 6, the critical point is at 150 tps, and when the number of network nodes is 10, the critical point is at 75 tps. Similarly, the results of Figure 9(c) show that as the number of networks increases, the value of TAR that reaches critical becomes smaller. In addition, the delay fluctuation is affected by the type of smart contract, and the measurement results in Figure 9(d) show that the delay fluctuation range of the write type smart contract is larger than that of the read-type smart contract.

Figure 9(a) shows that throughput gradually increases with the growth of TAR, beginning to show a linear growth trend, but the throughput of write-type smart contract stops growing when the transaction arrival rate hits the range of 175-200 tps. Beyond this range, the transaction queue begins to block, and the average waiting time per transaction becomes longer, causing the average throughput to decrease slowly. Throughput results for read-type smart contract show the same trend as those for write type. Since read operations require relatively fewer computational resources, the average throughput reaches the critical point when the transaction arrival rate reaches a higher rate of about 225 tps. In combination with Figures 9(e) and 9(f), the critical interval that leads to a decrease in transaction throughput is also affected by TAR and network scale. Figure 9(e) shows that the larger the network scale, the earlier the critical interval will come. For example, when the number of nodes is 4, the critical interval is 175-200 tps, but when the number of nodes is 10, the critical interval is 125-150 tps. What is more, Figure 9(f) shows that when the TAR is below the critical point, the transaction throughput is largely independent of the number of nodes. Beyond the critical point, as the number of nodes increases, transaction throughput decreases, and the rate of decline decreases as the number of nodes grows bigger.

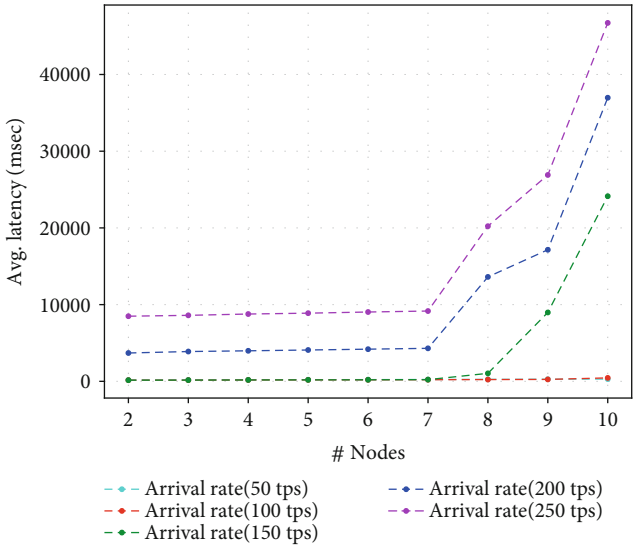
Practicality analysis. The evaluation results show that RootChain's prototype system has a minimum of 150 tps and a maximum of 200 tps at a scale of 10 nodes. This means that RootChain can process at least 150 secondary root zone data change requests per second. Currently, there are 1504



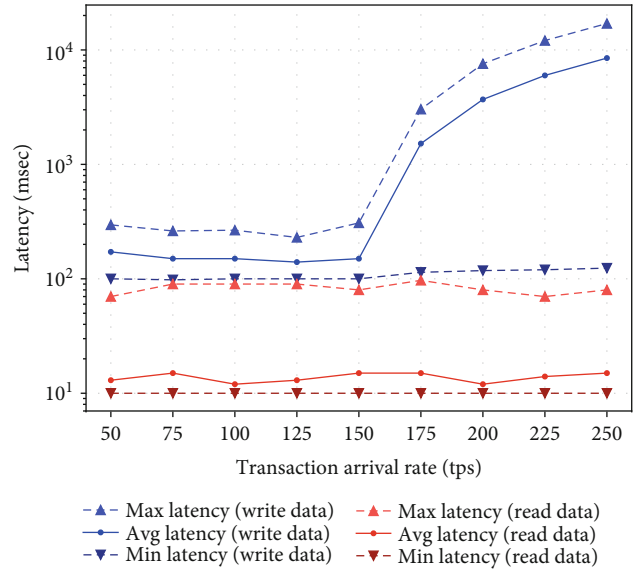
(a) Average latency/throughput vs. transaction arrival rate



(b) Average latency vs. transaction arrival rate in different number of nodes

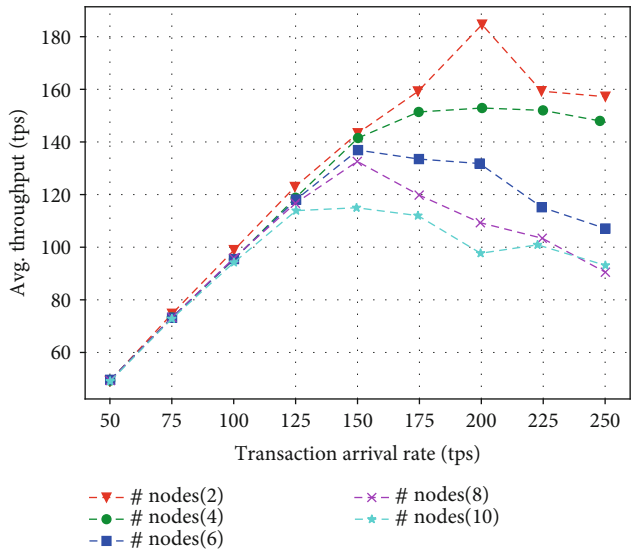


(c) Average latency vs. number of nodes in different transaction arrival rates

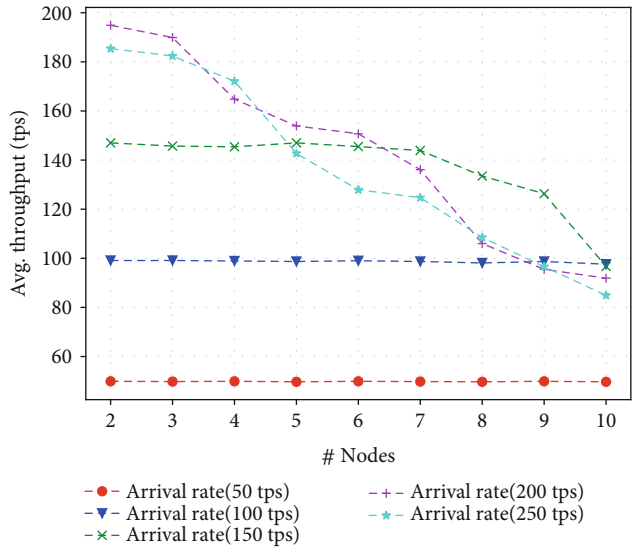


(d) Max/Min/Avg. latency vs. transaction arrival rate

Figure 9: Continued.



(e) Average throughput vs. transaction arrival rate in different number of nodes



(f) Average throughput vs. number of nodes in different transaction arrival rate

Figure 9: Evaluation results of the execution performance of the RootChain prototype system.

top-level domains registered in the root zone database published by IANA [11]. Therefore, our system supports each top-level domain to submit 8,617 (150*3600*24/1504) change requests to the RootChain system every day. This value far exceeds the daily update frequency of the top-level domain registry. From the perspective of delay, when tps reaches 150, the processing delay of each transaction is between 2 s and 4 s, so the second-level delay is acceptable to users.

8.3. Resource Consumption. Figure 10(a) shows the relationship between CPU usage and transaction arrival rate. Observation shows that CPU usage first increases linearly with TAR and then reaches a certain critical point, and the growth trend slows down until it stops increasing. This is because the number of transactions that the network can process per second is limited. When the transaction rate exceeds the critical point, the excess transactions will be discarded. After that, the number of transactions processed by the network per second remains unchanged, so the CPU usage rate tends to a certain fixed value. Combined with the analysis of Figure 10(d), it can be found that the higher the transaction arrival rate, the smaller the network scale when the CPU usage reaches the upper limit. When the transaction arrival rate is 250 tps, the critical number of network node is 7, and when the transaction arrival rate is 150 tps, the critical number of network node is 8.

Figures 10(b) and 10(e) show the relationship between memory usage and transaction arrival rate. From the test results, memory usage increases linearly with transaction arrival rate. Although there is an upper limit on the number of transactions processed per second in the RootChain network, unprocessed transactions will be cached in the pending transaction queue of the RootChain network, so the memory usage increases linearly with the transaction arrival rate, unlike the CPU usage, which has a critical point in the trend of change.

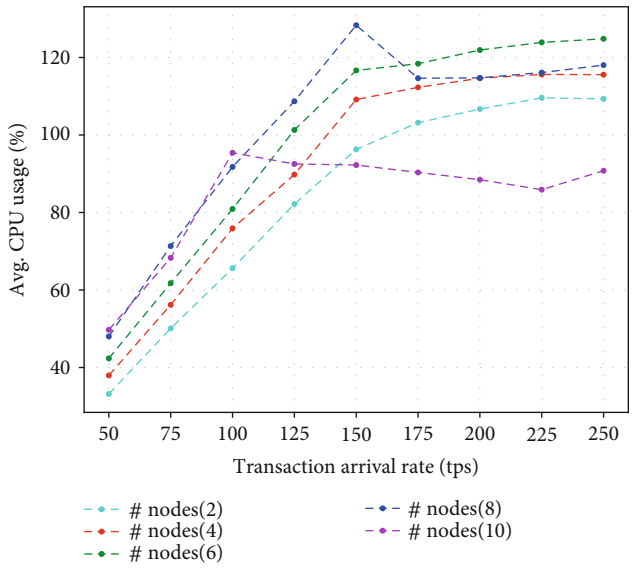
Figures 10(c) and 10(f) show the relationship between the total network traffic and the transaction arrival rate. Since the duration of the test process is fixed, the total transaction volume tested has a linear relationship with the transaction arrival rate. The total transaction volume determines the communication traffic within the network, so there is also a linear relationship between communication traffic and transaction arrival rate.

9. Related Work

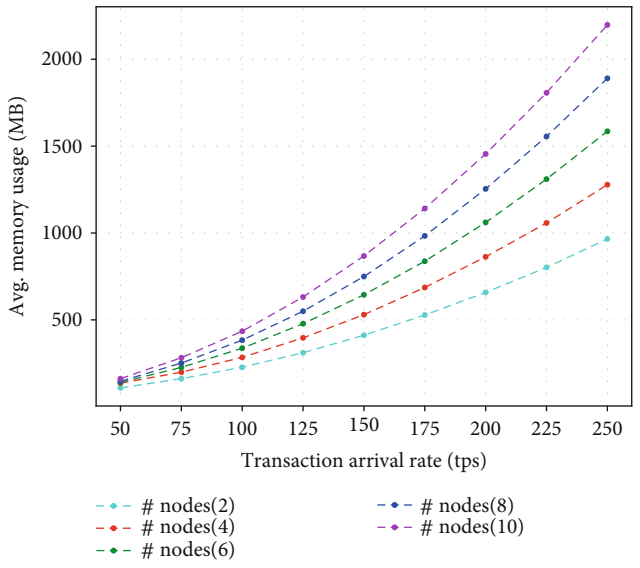
Distributed security is an important research direction in the field of IoT [12].

9.1. Traditional DNS Decentralization Solutions. In 2000, Kangasharju and Ross proposed a new decentralized DNS data management scheme [13], which changed the storage of DNS data from a hierarchical structure to a flat structure. The scheme proposed to replace all the secondary authoritative servers with the new authoritative servers—replicated name servers (RNSs). RNSs are interconnected by using multicast and synchronizing their database with each other. RootChain does not change the storage of root zone data inside root servers, but decentralizes root zone operation for TLDs.

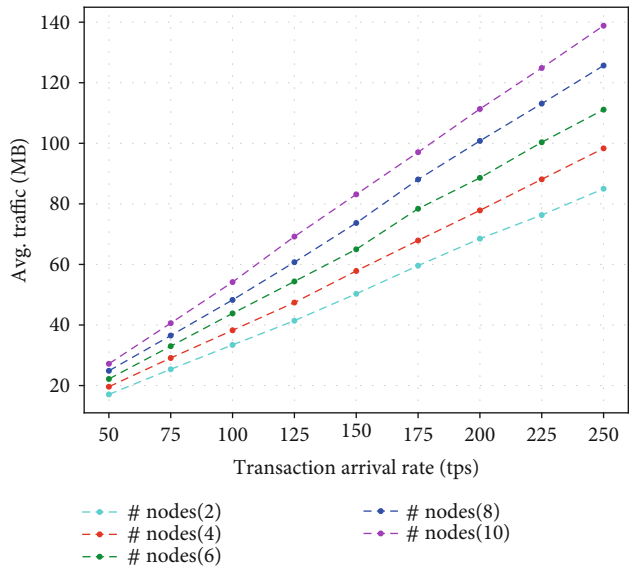
The ORSN project [14] maintains a set of independent open root servers to prevent users from being monitored and controlled by government agencies when resolving domain names. The scheme implements root server operation decentralization in the form of community spontaneous organization, but the dependency on the root zone file provided by ICANN is the limitation of the scheme. RootChain realizes decentralization of root zone operation, removing this limitation.



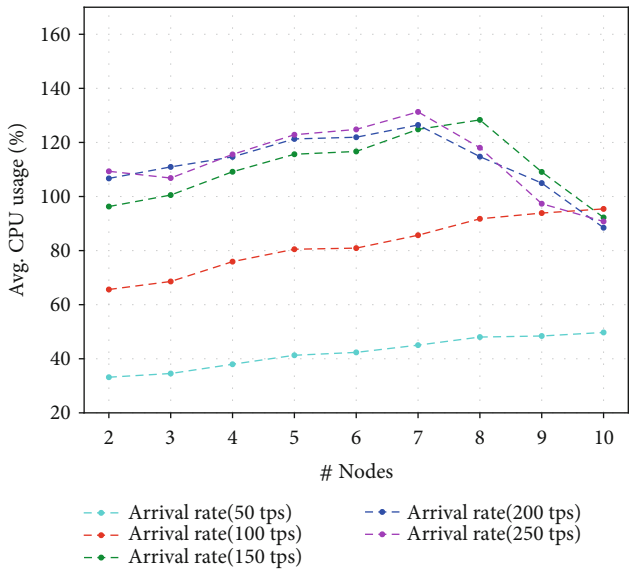
(a) Average CPU usage vs. transaction arrival rate in different number of nodes



(b) Average memory usage vs. transaction arrival rate in different number of nodes

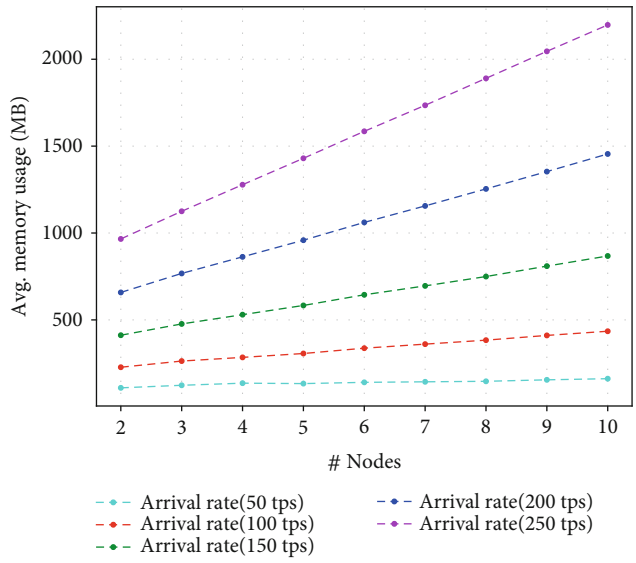


(c) Average Traffic vs. transaction arrival rate in different number of nodes

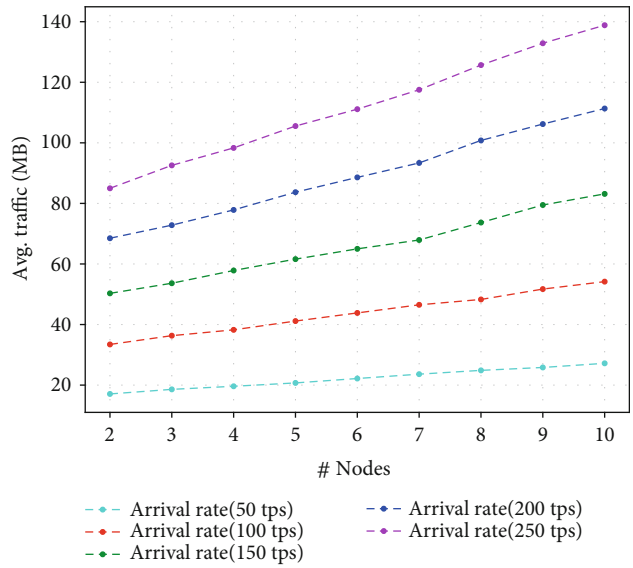


(d) Average CPU usage vs. number of nodes in different transaction arrival rate

Figure 10: Continued.



(e) Average memory usage vs. number of nodes in different transaction arrival rate



(f) Average Traffic vs. number of nodes in different transaction arrival rate

Figure 10: Evaluation results of the resource consumption of the RootChain prototype system.

The UnifiedRoot [15] project and the Public-Root [16] project have established a new DNS that is completely independent of the current DNS, which uses a new namespace parallel to the current one, leading to namespace split. Those solutions are completely isolate from the current DNS ecosystem and essentially only transfer the control from one center to another. RootChain retains the current DNS namespace and does not change the root zone authority for TLD delegation.

9.2. Blockchain-Based DNS Decentralization Solutions. Namecoin [1] is a distributed naming system based on blockchain, without any name registration authority. Instead, Namecoin adopts the "first come, first served" manner for obtaining domain names under the ".bit" domain. As Namecoin provides name resolution service by sharing domain name data over a peer-to-peer network, there is no single-point-of-failure issue. But the following issues remain unsolved with Namecoin: (1) there is no root zone; thus, no TLD authority, which is not compatible with the current DNS ecosystem. (2) Proof-of-Work (PoW) [17] is adopted for global consensus over name data, which means that Namecoin is vulnerable to attacks from major mining pools of Bitcoin. In fact, there has been cases when a single mining pool controlled over 50% of the overall computing power [18].

The BlockStack Naming Service (BNS) [2] is a global naming and storage system based on Bitcoin, which binds names to off-chain state. Blockstack extends the functionality of Bitcoin by adding a middle layer "virtualchain" on top of Bitcoin, enabling any data storage with a name as the key. Blockstack has the following problems: (1) TLD names are still acquired by bidding, and as with Namecoin, malicious squatting behavior cannot be avoided. (2) Blockstack is developed on top of Bitcoin, and its security and efficiency are inevitably determined by Bitcoin. The PoW consensus mechanism adopted by Bitcoin has already shown security vulnerabilities [19]. Bitcoin's slow transaction committing speed and long confirmation time have also be inherited and not been properly resolved in Blockstack.

Compared to Namecoin and Blockstack, RootChain retains the current DNS root zone authority for TLD delegation, so there is no risk of namespace split nor malicious squatting. Within the system, participants make an agreement in a "one person, one vote" fashion, gathering all root server operators into the root zone data operation, making sure that the system will not be compromised by a single participant. The PoW consensus algorithm has a lot of computational waste and is proved to be not efficient. RootChain adopts a PBFT-like [20] consensus scheme to eliminate the unnecessary resource waste [21–25]. Use blockchain technique to design a trust enhancement mechanism or security data management scheme for DNS.

9.3. Other IoT-Related Decentralized Security Solutions. Yang et al. [12] propose a decentralized and adaptive flocking algorithm for autonomous mobile robots. Shahzad et al. [26] propose a real-time transmission encryption security protocol for industrial IoT devices. Lu et al. [27] propose an IoT security solution based on authentication mechanism. Zhang et al. [28] propose multimodel incident prediction and risk assessment methods for dynamic network security in industrial IoT control devices. Huang et al. [29] propose a network intrusion detection method for sensor devices. Wu et al. [30] propose an energy optimization method for wireless sensor networks composed of IoT devices. [31–35] propose enhanced solutions for Internet of Things security.

DePET [36] is a decentralized privacy-preserving energy trading scheme for vehicular energy network via blockchain. CertCoin [37] is a blockchain-based identity authentication system alternative to PKI. EthIKS [38] is a blockchain-based key verification service system on the Ethereum

platform, eliminating the need for a trusted third party. Wang et al. [39] and Yakubov et al. [40] also introduced blockchain-based PKI certificate management systems. BGP Coin [41] is a blockchain-based Internet resource management solution for IP address prefixes and AS numbers through smart contracts. Those solutions share the similar spirit with RootChain in distributing infrastructures through blockchain, but for different systems.

In summary, different from previous solutions, RootChain distributes DNS root zone operation through blockchain while retaining a single root authority, by separating the delegation of TLDs from the data publication of TLDs.

10. Conclusion

In this paper, we have proposed a blockchain-based root operation architecture—RootChain. By decoupling TLD data publication from TLD delegation, RootChain distributes root zone operation across multiple root servers while maintaining the single root authority. The root zone data is authenticated and published by delegated TLD authorities directly into the ledger of RootChain. To improve the transparency and accountability of root zone operation, we designed smart contracts for the whole lifetime of TLD from delegation to revocation, including some abuse-proof mechanisms for TLD transition. The prototype of RootChain has been implemented with Hyperledger Fabric and evaluated by experiments. We take RootChain as the first step towards distributed root zone operation in DNS. Our future work includes (1) the quantitive comparison between RootChain and other proposals for DNS decentralization, (2) the design of distributed consensus mechanism for RootChain under the assumption without a single trust anchor, and (3) extending the design of RootChain to other Internet infrastructures with single root authority, such as RPKI.

Data Availability

The data used to support the finding of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is partially supported by the National Key Research and Development Program of China (Grant No. 2018YFB1800702).

References

[1] A. Loibl and J. Naab, *Namecoin*, 2014, http://namecoin.info.

[2] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: a global naming and storage system secured by blockchains," in *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pp. 181–194, Denver, CO, 2016.

[3] *The Hyperledger Fabric project*March 2019, https://www.hyperledger.org/projects/fabric.

[4] *Root Zone Database*October 2020, https://www.iana.org/domains/root/db.

[5] *Iana naming function contract*October 2020, https://www.icann.org/en/system/files/files/iana-namingfunction-agreement-revised-to-address-comments-15sep16-en.pdf.

[6] *An introduction to the internet assigned numbers authority (iana) functions*October 2020, https://www.icann.org/en/system/files/files/ianafunctions-15jun15-en.pdf.

[7] *Root zone maintainer service agreement*October 2020, https://www.icann.org/ianaimpdocs/129-root-zonemaintainer-service-agreement-v-28sep16.

[8] *National telecommunications and information administration (ntia) close-out contract*October 2020, https://www.ntia.doc.gov/files/ntia/publications/sa1301-12-cn-0035001-10212016.pdf.

[9] *List of root server operators*October 2020, https://www.iana.org/domains/root/servers.

[10] *The Hyperledger Caliper project*March 2019, https://www.hyperledger.org/projects/caliper.

[11] *IANA Root Zone Database*October 2020, https://www.iana.org/domains/root/db.

[12] Y. Yang, N. Xiong, N. Y. Chong, and X. Defago, "A decentralized and adaptive flocking algorithm for autonomous mobile robots," in *International Conference on Grid Pervasive Computing Workshops*, Kunming, 2008.

[13] J. Kangasharju and K. W. Ross, "A replicated architecture for the domain name system," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*, vol. 2, pp. 660–669, Tel Aviv, Israel, Israel, 2000.

[14] *The Open Root Server Network (ORSN) project*March 2019, https://www.orsn.org.

[15] *The UnifiedRoot project*March 2019, http://www.unifiedroot.com.

[16] *The Public-Root project*March 2019, http://public-root.com/.

[17] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols," *Secure Information Networks*, pp. 258–272, 1999.

[18] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan, "An empirical study of Namecoin and lessons for decentralized namespace design," *WEIS*, 2015.

[19] S. Feng, W. Wang, Z. Xiong, D. Niyato, P. Wang, and S. S. Wang, "On cyber risk management of blockchain networks: a game theoretic approach," *IEEE Transactions on Services Computing*, p. 1, 2018.

[20] M. Castro and B. Liskov, "Practical byzantine fault tolerance," *OSDI*, vol. 99, pp. 173–186, 1999.

[21] Z. Yu, D. Xue, J. Fan, and C. Guo, "Dnstsm: DNS cache resources trusted sharing model based on consortium blockchain," *IEEE Access*, vol. 8, pp. 13640–13650, 2020.

[22] J. Zhang, J. Zhai, R. Yang, and S. Liu, "Research on enterprise DNS security scheme based on blockchain technology," in *International Conference on Blockchain and Trustworthy Systems*, pp. 690–701, Singapore, 2019.

[23] W. Wang, N. Hu, and X. Liu, "Blockzone: a blockchain-based dns storage and retrieval scheme," in *International Conference on Artificial Intelligence and Security*, pp. 155–166, Cham, 2019.

[24] S. Gourley and H. Tewari, "Blockchain backed dnssec," in *International Conference on Business Information Systems*, pp. 173–184, Cham, 2018.

[25] G. He, W. Su, S. Gao, and J. Yue, "TD-Root: a trustworthy decentralized DNS root management architecture based on permissioned blockchain," *Future Generation Computer Systems*, vol. 102, pp. 912–924, 2020.

[26] S. Aamir, L. Malrey, L. Young-Keun et al., "Real time MODBUS transmissions and cryptography security designs and enhancements of protocol sensitive information," *Symmetry*, vol. 7, no. 3, pp. 1176–1210, 2015.

[27] Y. Lu, S. Wu, Z. Fang, N. Xiong, S. Yoon, and D. S. Park, "Exploring finger vein based personal authentication for secure IoT," *Future Generation Computer Systems*, vol. 77, pp. 149–160, 2017.

[28] Q. Zhang, C. Zhou, N. Xiong, Y. Qin, X. Li, and S. Huang, "Multimodel-based incident prediction and risk assessment in dynamic cybersecurity protection for industrial control systems," *IEEE Transactions on Systems Man Cybernetics Systems*, vol. 46, no. 10, pp. 1429–1444, 2017.

[29] H. Kaixing, Z. Qi, Z. Chunjie, X. Naixue, and Q. Yuanqing, "An efficient intrusion detection approach for visual sensor networks based on traffic pattern learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 10, pp. 2704–2713, 2017.

[30] W. Wu, N. Xiong, and C. Wu, "Improved clustering algorithm based on energy consumption in wireless sensor networks," *IET Networks*, vol. 6, no. 3, pp. 47–53, 2017.

[31] Y. Liu, M. Ma, X. Liu, N. Xiong, A. Liu, and Y. Zhu, "Design and analysis of probing route to defense sink-hole attacks for Internet of Things Security," *IEEE Transactions on Network Science and Engineering*, 2018.

[32] Z. Pan, C.-N. Yang, V. S. Sheng, N. Xiong, and W. Meng, "Machine learning for wireless multimedia data security," *Security and Communication Networks*, vol. 2019, 2 pages, 2019.

[33] A. Shahzad, M. Lee, C. Lee et al., "The protocol design and new approach for scada security enhancement during sensors broadcasting system," *Multimedia Tools and Applications*, vol. 75, no. 22, pp. 14641–14668, 2016.

[34] A. Shahzad, R. Landry, M. Lee, N. Xiong, J. Lee, and C. Lee, "A new cellular architecture for information retrieval from sensor networks through embedded service and security protocols," *Sensors*, vol. 16, no. 6, p. 821, 2016.

[35] N. Xiong, J. He, J. H. Park, D. H. Cooley, and Y. Li, "A neural network based vehicle classification system for pervasive smart road security," *The Journal of Universal Computer Science*, vol. 15, no. 5, pp. 1119–1142, 2009.

[36] Y. Long, Y. Chen, W. Ren, H. Dou, and N. N. Xiong, "Depet: a decentralized privacy-preserving energy trading scheme for vehicular energy network via blockchain and k-anonymity," *IEEE Access*, vol. 8, pp. 192587–192596, 2020.

[37] C. Fromknecht, D. Velicanu, and S. Yakoubov, *Certcoin: A Namecoin Based Decentralized Authentication System*, vol. 6, Massachusetts Institute of Technology, Cambridge, MA, USA, 2014.

[38] M. S. Melara, A. Blankstein, J. Bonneau, E. W. Felten, and M. J. Freedman, "Coniks: bringing key transparency to end users," *24th USENIX Security Symposium (USENIX Security 15)*, pp. 383–398, 2015.

[39] Z. Wang, J. Lin, Q. Cai, Q. Wang, D. Zha, and J. Jing, "Blockchain-based certificate transparency and revocation transparency," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[40] A. Yakubov, W. Shbair, A. Wallbom, and D. Sanda, "A blockchainbased pki management framework," in *The First IEEE/IFIP InternationalWorkshop on Managing and Managed by Blockchain (Man2Block) colocated with IEEE/IFIP NOMS 2018*, Tapei, Tawain, 2018.

[41] Q. Xing, B. Wang, and X. Wang, "Poster: Bgpcoin: a trustworthy blockchain-based resource management solution for bgp security," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2591–2593, Dallas, TX, USA, 2017.