LEA-DNS: DNS RESOLUTION VALIDITY AND TIMELINESS GUARANTEE LOCAL AUTHENTICATION EXTENSION WITH PUBLIC BLOCKCHAIN

Ting Xiong[1], Shaojin Fu[1], Xiaochun Luo[2] and Tao Xie[1]

[1]National University of Defense Technology, Hunan, China
[2]PLA News Media Center, Beijing, China

ABSTRACT

While the Domain Name System (DNS) is an infrastructure of the current network, it still faces the problem of centralization and data authentication according to its concept and practice. Decentralized storage of domain names and user local verification using blockchain may be effective solutions. However, since the blockchain is an add-only type database, domain name changes will cause out of date records to still be correct when using the Simplified Payment Verification (SPV) mechanism locally. This paper mainly introduces Local Enhanced Authentication DNS (LEA-DNS), which allows domain names to be stored in public blockchain database to provide decentralization feature and is compatible with the existing DNS. It achieves the validity and timeliness of local domain name resolution results to ensure correct and up to date with the Merkle Mountain Range and RSA accumulator technologies. Experiments show that less than 3.052Kb is needed for each DNS request to be validated, while the validation time is negligible, and only 9.44Kb of data need to be stored locally by the web client. Its compatibility with the existing DNS system and the lightness of the validation protocols indicate that this is a system suitable for deployment widely.

KEYWORDS

Domain name system, Blockchain, RSA accumulator, Merkle Mountain Range.

1. INTRODUCTION

DNS is a distributed database with a centralized data governance model that maps the names to values online, primarily controlled by the Internet Corporation for Assigned Names and Numbers (ICANN[1]). In this regard, ICANN manages the top-level domain (TLD) and therefore controls the root name server. In practice, if a client wants to contact a host with a specific name, it must first send a query to the DNS server to obtain the host's IP address. In order to improve efficiency, the DNS server may maintain a replica of this information in its cache, based on how often the domain name is requested. In the case that the DNS server does not hold the requested knowledge, the query will be propagated to the root name server. Next, the basic name server will find the server of the corresponding TLD, and then forward the query to the corresponding authoritative name server, which may return the requested IP [13].

[1]https://www.icann.org/resources/pages/governance/bylaws-en

Due to its centralized management architecture, DNS root is vulnerable to many attacks. The article [14] divides the current issues facing DNS into two categories: **centralization problem and data authenticity problem**. The centralization problem is that because users default to all DNS root servers being trusted, there will be malicious servers to attack [10]. The failure of this trust anchor is far more than a theoretical threat. The controversy surrounding the closure of *wikileaks.org* shows that the trust anchor failure occurs in the real world[2]. Distributed Denial of Service (DDoS) attacks are another threat[3]. Cache poisoning [16] and renumbering issues[9] are also related to centralization problem.

Using Blockchain technology in DNS is an effective solution to both of the above problems. Because of the decentralization nature of the blockchain and the untamperability nature of the data, the DNS resolver only needs to provide proof of the existence of the information in the blockchain, and the local browser can effectively solve the problem of DNS centralization and data authority by running the verification program. However, the blockchain is an add-only database, and an attacker may provide an outdated proof of existence to spoof the client to achieve the attack. As shown in Figure 1, an "old Tx" transaction can have all the block headers stored by the light node, then the SPV can be used to prove its existence. However, if a new transaction "new Tx" is based on "old Tx" with modifications, "old Tx" is still correct. But we cannot prove that the "old Tx" is the latest unless we download the whole chain to verify that there are no further transactions. Blockchain has the natural advantage of storing unmodified data, but this correspondence may be adjusted if the pair of <name, value> the DNS is stored, such as in the case of marketplace transactions for domain names. Attackers may take stale transactions(may store the old pair of <name, value>) to trick users, but users don't perceive them. However, compared with the traditional network, the performance of the public chain is very low, and it is difficult to directly resolve the domain name on the blockchain in the production environment. It is feasible to expand the security of the original domain name system, rather than pushing it back. In this paper, we try to solve these problems and design a DNS extension called LEA-DNS.

Contributions:

(1) Drawing on the Namecoin's architecture, we designed a UTXO-based structure for storing and transacting DNS <name, value> pairs to fit our system design.
(2) We draw on the block structure design of miniChain[3] and boneh[1] to simplify the blockchain design applied to stateless blocks as a public blockchain database for storing key-value pairs.
(3) We design a system called LEA-DNS, that allows users to perform enhanced verification of domain name resolution result locally. The system not only verifies the validity of the data, but also the timeliness of the result.
(4) Simulated experimental results show that each DNS response with verification does not exceed 3.052 Kb in size, and only a small amount of data needs to be stored locally (less than 9.44 Kb). Local validation time takes less than 10ms.

Organization of the Paper:

The rest of this paper is organized as follows. The related works are presented in Section 2. Then we give a brief introduction to our LEA-DNS system in Section 3. In Section 4, we present the details of the design of our system. In Section 5, we theoretically and experimentally evaluate our prototype implementation of LEA-DNS. Finally, we conclude our paper in Section 6.

[2]https://news.netcraft.com/archives/2010/12/03/wikileaks-org-taken-down-by-us-dns-provider.html
[3]https://en.wikipedia.org/wiki/Distributed_denial-of-service_attacks_on_root_nameservers

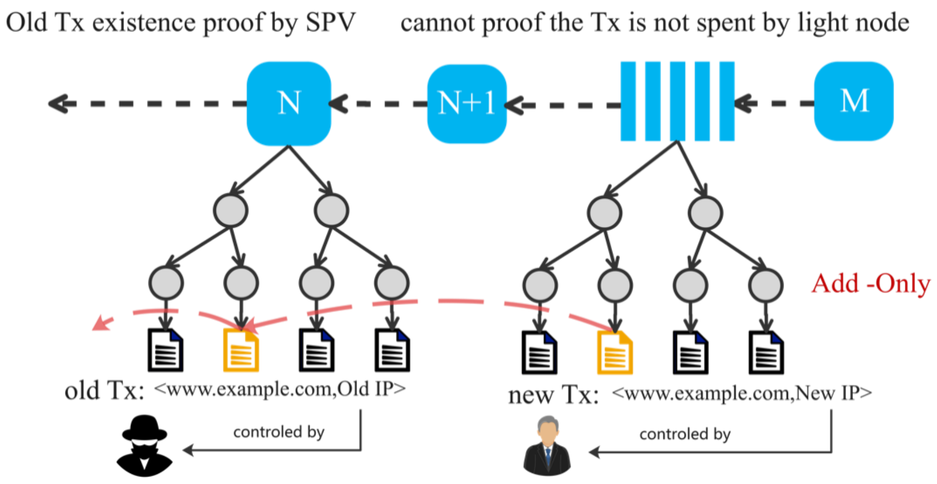

Old Tx existence proof by SPV
cannot proof the Tx is not spent by light node
N
N+1
M
Add -Only
old Tx: <www.example.com,Old IP>
new Tx: <www.example.com,New IP>
controled by
controled by

Figure 1. The Unspent Proof Problem of Transactions in Blockchain by Light Node. Old Tx existence proof can be proved by SPV and old Tx spent proof cannot be proved only by light node, so only give a SPV, the light node may be cheat by the invalid old Tx.

2. Related Work

Decentralized systems were in principle used to improve the robustness and availability of domain name resolution tasks as well as enabling the feature of by passing censorship campaigns and tampering [6, 7,17].

In the Byzantine fault-tolerant DNS, a client sends a request to all the replicas that runs the Byzantine fault-tolerant consensus algorithm and waits for enough authenticated replies. It can tolerate one-third malicious servers behaving arbitrarily. However, Byzantine fault-tolerant systems increase the communication overhead squared back as the number of nodes increases. So, the performance of Byzantine fault-tolerant algorithm decreases quadratically as more servers are added. In DHT based DNS schemes, DDNS and Overlook are peer-to-peer name services designed to enhance load balancing and fault tolerance properties. DDNS is a DNS alternative using a peer-to-peer distributed hash table built on top of Chord. The Overlook is based on Pastry. Both DDNS and Overlook have much higher latencies than conventional DNS.

With the birth of bitcoin, blockchain technology is increasingly being used in distributed DNS technology. The Ethereum name service (ENS) uses smart contracts to manage the *.eth* registrar by means of bids and recently added the support for .onion addresses. Namecoin is a cryptocurrency based on Bitcoin, with additional features such as decentralized name system management, mainly for the *.bit* domain. It was the first project to provide an approach to address Zooko's triangle[4] since the system is secure, decentralized and user-chosen names (human meaningful). Nevertheless, contrary to well-established blockchains like Bitcoin, Namecoin's main drawback is its insufficient computing power, which makes it more vulnerable to the 51% attack. Blockstack is a well-known blockchain-based domain name storage system that overcomes the main drawbacks of Namecoin. The architecture of Blockstack separates control and data planes, enabling seamless integration with the underlying blockchain. EmerDNS[5] is a decentralized domain name system that supports all the range of DNS records. Nebulis[6] is a

[4]http://en.wikipedia.org/wiki/ Zooko%27s triangle
[5]https://emercoin.com/en/documentation/blockchain-services/emerdns/emerdns-introduction
[6]https://www.nebulis.io/

globally distributed directory that relies on the Ethereum ecosystem and smart contracts to store, update, and resolve domain records. Moreover, Nebulis proposes the proposal of using off-chain storage (i.e. IPFS) as a replacement for HTTP. OpenNIC[7] is a hybrid method during which a group of peers manage the name space registration, but the name resolution task is completely decentralized. OpenNIC provides DNS resolution and namespace over a collection of domains, including those maintained by blockchain solutions like New Nations[8] and EmerDNS. In addition, the OpenNIC resolver has recently added access to domains managed by ICANN. Additionally, to namespace registrar, users can even create their own TLD upon request [8, 13, 15].

However, none of them consider the data authenticity problem properly. Though the SPV capability is available, clients need to pay too much overheads to verify the resolution results. BlockDNS [14] give a solution, but it still cannot solve the problem showed in Figure 1(explained in Section 1).

3. Overview of LEA-DNS

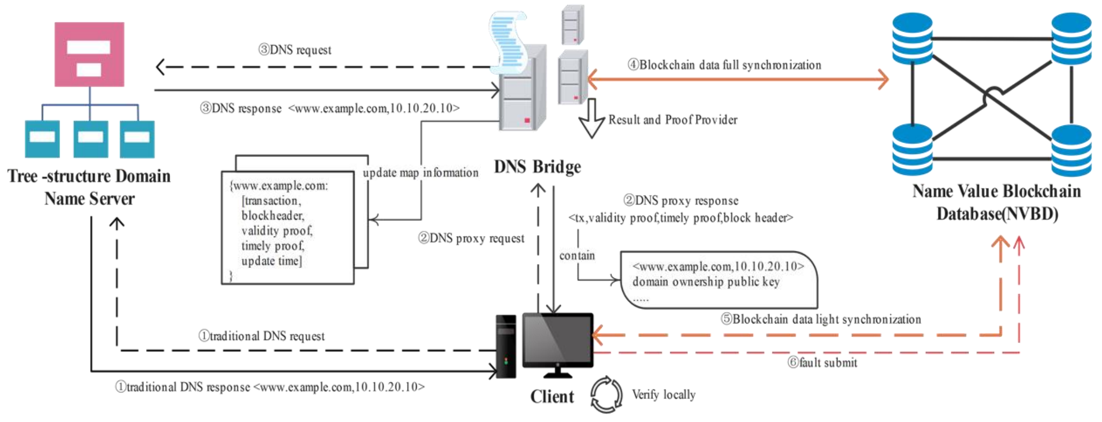

③DNS request
③DNS response <www.example.com,10.10.20.10>
④Blockchain data full synchronization
Result and Proof Provider
Tree -structure Domain Name Server
{www.example.com: [transaction, blockheader, validity proof, timely proof, update time] }
update map information
DNS Bridge
Name Value Blockchain Database(NVBD)
②DNS proxy response <tx,validity proof,timely proof,block header>
②DNS proxy request
contain
<www.example.com,10.10.20.10> domain ownership public key .....
⑤Blockchain data light synchronization
①traditional DNS request
⑥fault submit
①traditional DNS response <www.example.com,10.10.20.10>
Client
Verify locally

Figure 2. System Architecture of LEA-DNS.

As shown in Figure 2, LEA-DNS is composed of four main components:

3.1. Tree-structure DNS

Tree-structure DNS is the traditional tree architecture domain name system. To maintain good compatibility with the LEA-DNS, it can be deployed directly without any modification to the current DNS architecture. In other words, LEA-DNS is transparent to the legacy DNS.

3.2. Name-Value Blockchain Database(NVBD)

Name-Value Blockchain Database (NVBD) is a decentralized way of storing <name, value> pairs that require enhanced validation. The design of NVBD is borrowed from Namecoin, Blockstack and miniChain. The <name, value> is stored directly in the transaction, allowing domain name registration, assignment, and transfer operations. Due to the immutability nature of the blockchain, we consider the data stored by the blockchain itself to be authenticated (the data has been verified by enough honest nodes when confirmed). Since the operation of each

[7]https://www.opennic.org/
[8]http://www.new-nations.net/

transaction in the blockchain is controlled by the public and private keys, we bind <name, value> to the transaction, and the ownership of the domain name is signed with the private key, we can easily verify the data through the public key in the transaction, which confirm the source of authenticated.

3.3. DNS Bridge

DNS Bridge is a set of proxy servers that handle requests and provide verifiable DNS response. The DNS Bridge is required to synchronize block and transaction information with the NVBD as step 4) in Figure 2. The DNS request is specially processed to obtain a transaction containing a <name, value> pair and proof of validity and timeliness of the result, which is sent to the Client. Note that DNS Bridge is a trusted institution, because it can't forge real proof. If the proof provided by DNS Bridge can be verified locally, it can be trusted by users. In order to prevent DNS Bridge from being attacked by DOS, multiple nodes can provide services.

3.4. Client

Client can firstly request services directly from the traditional DNS. Secondly, it can send a verifiable request to DNS Bridge as step 2). In step 5), the Client needs to maintain communication with the NVBD at the same time but only needs to synchronize the latest block headers and save a small amount of other data to verify the transaction locally about guaranteeing validity and timeliness. If the validation fails, the error-proof message is submitted to the NVBD as step 6) in Figure 2.

4. DESIGN OF THE SYSTEM

In this section, we focus on the main techniques used to design each part of the system. While many of the market mechanisms in Namecoin were examined in the article[11], this section focuses on the technical details, including transaction design and the block structure of NVBD, DNS Bridge and Client.

4.1. Transactions Design

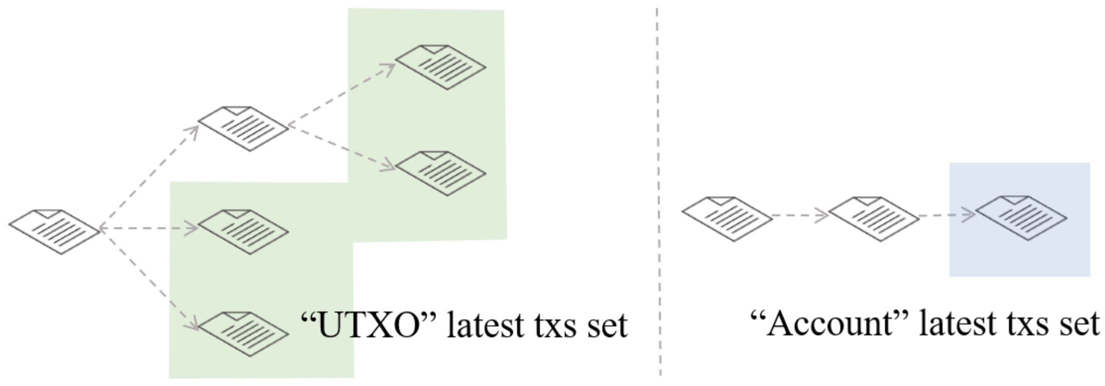
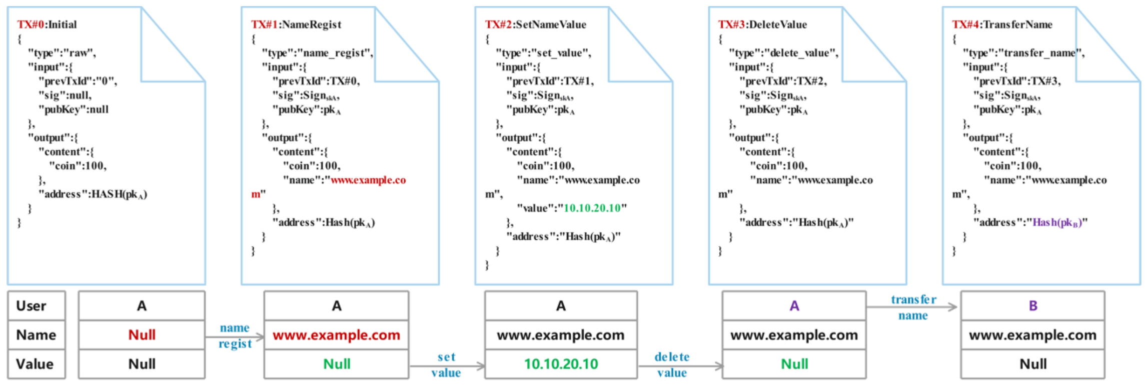


Figure 3. The difference between UTXO model and Account model.

As shown in Figure 3, because of the parallel characteristics of UTXO model, it can process multi transactions at the same time without mutual influence. The set of unspent transactions "UTXO" can be used as the set of the latest transactions. The Account model is based on the account, all transactions are added serially, and the latest transaction set size is only 1. From the perspective of transaction, UTXO model is more suitable for timeless proof, because it contains more transactions than Account model. If the account is taken as the basic unit of proof, the Account model can also be applied. For convenience, this paper uses UTXO model.

Let’s assume that domain name provider A has a public-private key pair (pk_A, sk_A) and its user address is $HASH(pk_A)$. The public-private key pair for domain name provider B is (pk_B, sk_B), and its user address is $HASH(pk_B)$. The key to a domain name store with blockchain as the storage interpretation is that the data is transparent, traceable, and verifiable, but anonymity can be ignored, so our user address is a direct public key hash, and the user can also perform transactions on the same address. To simplify the situation, we only consider the transaction of one-to-one addresses, and do not consider the transaction fee, so we omit the index field of the transaction output, a simple process design is shown in Figure 4.



```
TX#0:Initial
{
  "type":"raw",
  "input":{
    "prevTxId":"0",
    "sig":null,
    "pubKey":null
  },
  "output":{
    "content":{
      "coin":100,
    },
    "address":HASH(pkA)
  }
}
```

```
TX#1:NameRegist
{
  "type":"name_regist",
  "input":{
    "prevTxId":TX#0,
    "sig":SignskA,
    "pubKey":pkA
  },
  "output":{
    "content":{
      "coin":100,
      "name":"www.example.co
m"
    },
    "address":Hash(pkA)
  }
}
```

```
TX#2:SetNameValue
{
  "type":"set_value",
  "input":{
    "prevTxId":TX#1,
    "sig":SignskA,
    "pubKey":pkA
  },
  "output":{
    "content":{
      "coin":100,
      "name":"www.example.co
m",
      "value":"10.10.20.10"
    },
    "address":"Hash(pkA)"
  }
}
```

```
TX#3:DeleteValue
{
  "type":"delete_value",
  "input":{
    "prevTxId":TX#2,
    "sig":SignskA,
    "pubKey":pkA
  },
  "output":{
    "content":{
      "coin":100,
      "name":"www.example.co
m"
    },
    "address":"Hash(pkA)"
  }
}
```

```
TX#4:TransferName
{
  "type":"transfer_name",
  "input":{
    "prevTxId":TX#3,
    "sig":SignskA,
    "pubKey":pkA
  },
  "output":{
    "content":{
      "coin":100,
      "name":"www.example.co
m",
    },
    "address":"Hash(pkB)"
  }
}
```

User	A	A	A	A	B
Name	Null	www.example.com	www.example.com	www.example.com	www.example.com
Value	Null	Null	10.10.20.10	Null	Null

Figure 1. NVBD Transactions Design and the State Change. The conversion from transaction TX#0 to transaction TX#4 represents four operations on a domain name: registration, assignment, deletion and transfer.

Regist Name:

Domain name provider A first needs a “raw” transaction as its initial transaction, as shown in Figure 4, TX#0. When A needs to register a domain name,it takes the “raw” transaction as input and outputs a transaction of type “name regist”. The output contains the domain name field. If the domain name is duplicated, the transaction will not be packaged into the block.

Set Name Value:

Assigning a value to a registered domain name or a transferred domain name is known as an IPv4 address in LEA-DNS. When A needs to specify an IP address for the domain name, it takes the transaction “name regist” as input and generates a transaction of type “set value”.

Delete Value:

When A needs to reassign a value to a domain name or to transfer a domain name, A needs to enter a transaction of type “set value” and generate a “delete value” type of transaction, the original assignment is deleted.

Transfer Name:

Transfer of a registered domain name. When A needs to transfer its domain name to B, A needs to enter a transaction of type “name regist” or “delete value” and generate a transfer of type “transfer name” transaction, the output address of which is the hash of the public key of B. At this point, the public key of B is not public and is anonymous to a certain extent.

Update Value:

The value update operation is similar to the domain assignment operation. Its input and output are both a transaction of type “set value”.

All transactions are verified similarly to Bitcoin, but with the addition of a determination of domain duplicity.

4.2. Block Design

First, we’d like to understand about accumulators. Basically, a cryptographic accumulator[5] is an algorithm to mix an outsized set of values into one short commitment, and enables to compute a brief membership witness (or nonmembership witness) of any element that has (or not) been accumulated. RSA accumulator is predicated on modular exponentiation under the strong RSA assumption[12]. In decentralized public blockchains where no single trusted accumulator manager exists, the essential RSA accumulator doesn’t satisfy the need, anyone who knows the secret keys p and q can use the Euclidean theorem $\varphi(N) = (p-1)(q-1)$ to calculate the order of RSA group, which may further forge any membership and nonmembership witness. Boneh[1] built a stateless blockchain[4] supported UTXO commitment by using the RSA accumulator, which needs plenty of deletion operations. Since the complexity of deletion operation is $O(n^2)$, the efficiency of the accumulator updates would drop rapidly when the amount of deletion operations increases. We use the Chen’s work[3],which divides the UTXO to STXO(i.e., spent transactions outputs) and TXO(i.e., all transactions outputs).A transaction in UTXO indicates its validity like a transaction in TXO but not in STXO.

Since LEA-DNS allows user validation of <name, value> to be done locally, the simplest idea is to store the full blockchain in the NVBD to validate transactions of the “set value” type. However, storing the full blockchain data would significantly increase the storage cost for the user. A more lightweight approach is similar to Bitcoin’s light wallet, where only the block header data is stored locally, and the full node sends the transaction’s SPV proof to verify the transaction’s validity in the blockchain. But the SPV scheme doesn’t solve the problem of whether the transaction is the most recent, or to determine if the transaction is a UTXO except all UTXOs data needs to be synchronized locally. But UTXO grows at a rate that the average user can’t afford. These solutions are not feasible. We modified the structure of Chen’s work[3] to allow users to verify the validity and timeliness of transactions by storing only a small amount of data.

The design architecture of the block is shown in Figure 5.

STXO Commitment:

STXO_C is an append-only data structure which contains all spent transaction outputs, removing the time-consuming deletion operations needed by UTXO commitment. Specifically, each block header contains an accumulator which represents the current STXO set. A transaction can be provided a nonmembership witness which specifies that the transaction was not spent before.

STXO_C is essentially an RSA accumulator, and when a transaction is added to a block, we simply add the UTXO spent on that transaction to the accumulator. The initialization accumulator is generated by a security parameter λ and returns an accumulator A_0 .

The whole process of handling transactions is similar to Bitcoin, except that it needs to update the accumulator by marking the transaction input as spent when a blockchain mining node receives a transaction. The update algorithm accepts an old accumulator A_t and a transaction TX, and updates A_t to A_{t+1} by performing a $HashToPrime$(i.e., a function that transfer hash to prime) of the input transaction in TX. The number of transactions in a block means the times an accumulator needs to be updated from Pre_STXO_C to STXO_C.

TXO Commitment:

TXO_C is a commitment for all transactions. Traditional verification can check the Merkle path from transaction to TMR directly, but this requires the verifier to store all block headers. To reduce the verifier's storage overhead and speed up this verification approach, we use the MMR approach. The user only needs to store all the MMR Peaks in Figure 5(a), provided with the Merkle path from the transaction to the TMR and the Merkle path from the TMR to the MMR root which is constructed by MMR Peaks. The number of MMR Peaks increases logarithmically with the length of the blockchain, so this overhead is small.

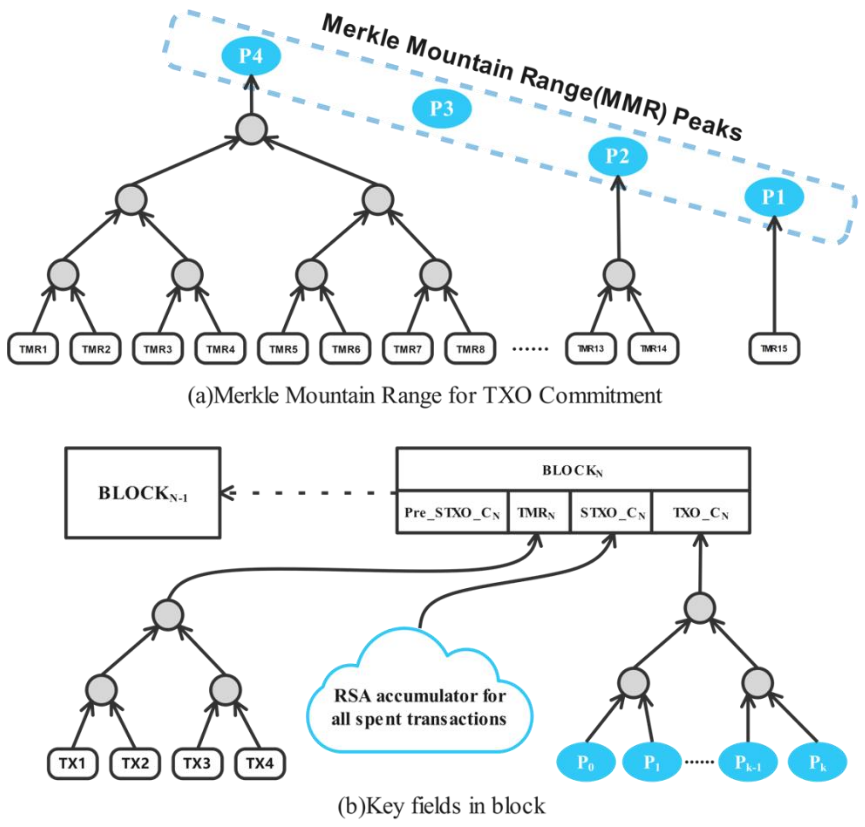


Figure 2. NVBD block architecture. (a) represents the Merkle Mountain Range (MMR)[2] which connects all the block headers. (b) represents the main fields in block header containing 4 parts:(1)TMR organizes all transactions within a block through a merkle structure, (2) STXO_C represents a commitment for all spent transactions, (3)TXO_C represents a commitment for all MMR Peaks through a normal merkle structure. (4) Pre_STXO_C represents the previous block STXO_C.

4.3. DNS Bridge Design

DNS Bridge is required to synchronize all of NVBD's block information and update the DNS and transaction map information based on the transactions in the block. It forwards the user's DNS request, and finally form a verifiable DNS response back to the user. In order to let user verify the validity and timeliness of the returned <name, value> pair, it is necessary to generate the validity and timeliness proof of the "set value" type of transactions corresponding to <name, value>.

Validity Proof:

A validity proof of a transaction is a proof of the existence of a transaction in the blockchain. The proof of existence of a transaction is divided into two parts. 1) Merkle path of the transaction to TMR . 2) The Merkle path from TMR to MMR root(i.e., TXO_C).

Timely proof:

A timely proof of a transaction is a proof of the non-existence (i.e., unspent) of the transaction from the beginning of the generated block n to the specified block $m(m > n)$. In Li's paper[12], this is called a nonmembership witness. The situation described above is full timely proof. If n is not the block where the transaction is located inside, we take Δ to represent the difference in height from the located block height to the height of the $block_m$, then call it a Δ-timely proof. We use $tf_m(x_n)$ to represent the timely proof of the transaction x_n to $block_m$.$\Delta tf_m(x_n)$ means in the latest Δ blocks, the x_n is not been spent. The algorithm for generating $tf_m(x_n)$ is shown in Algorithm 1. Assuming $x_n \in UTXO$ and$x_n \notin STXO$, the algorithm first obtains the set of all unspent transactions $STXO_{n:m}$ from the $block_n$ to $block_m$, and simultaneously performs a $HashToPrime$. Then calculate the product of all primes as p. Since x_n and p are different prime numbers, it is easy to find $Bezout$ coefficients a and b such that $ax + bp = 1$ by using extended Euler's theorem. The final calculation $d = A_n^a$, returns (d, b) as $tf_m(x_n)$.

Algorithm 1 Timely Proof

```
Input:
$block_n$ previous accumulator $A_{n-1}$(Pre_STXO_C$_n$);
$block_m$ accumulator $A_m$(STXO_C$_m$);
    proof transaction $x_n$;
    all spent transactions from $block_n$ to $block_m$ presented by STXO$_{n:m}$ .
Output:
    timely proof $tf_m(x_n) \leftarrow \{d, b\}$.
1: $p \leftarrow 1$
2: $x_n \leftarrow HashToPrime(x_n)$
3: for $stx$ in STXO$_{n:m}$ do
4:   $p \leftarrow p \cdot HashToPrime(stx)$
5: end for
6: $a, b \leftarrow Bezout(x_n, p)$
7: $d \leftarrow A_n^a$
8: return $tf_m(x_n) \leftarrow \{d, b\}$
```

Algorithm 2 Timely Proof Update

```
Input:
$block_m$ accumulator $A_m$;
    old proof $tf_m(x_n)$;
    transaction $x_n$;
    all spent transactions from $block_m$ to $block_{m'}$ presented by $\mathrm{STXO}_{\mathrm{m:m}'}$ .
Output:
    new timely proof $tf_{m'}(x_n)$.
1:  $p \leftarrow 1$
2:  $d, b \leftarrow tf_m\ (x_n\ )$
3:  for $stx$ in $\mathrm{STXO}_{\mathrm{m:m}'}$ do
4:      $p \leftarrow p \cdot HashToPrime(stx)$
5:  end for
6:  $a', b' \leftarrow Bezout(HashToPrime(x_n), p)$
7:  $r\ \leftarrow a'b$
8:  return $tf_{m'}(x_n) \leftarrow \{dA_m^r, b'b\}$
```

Timely Proof Update:

$tf_m\ (x_n\ )$ with the growth of the blockchain will become out of date. Assume the current block height is m', in order to update the proof, we can recalculate the $tf_{m'}(x_n)$, and the new $tf_{m'}(x_n)$ can be computed based on $tf_m\ (x_n\ )$.The specifific update algorithm is shown in Algorithm 2. Here we give the proof:

Suppose there is a Timely Proof $tf_m(x_n) \leftarrow \{d, b\}$. it satisfies the following conditions (1) through Algorithm3:

$$d^x A_m^b = A_n; x = HashToPrime(x_n) \tag{1}$$

when add some TXs to A_m,the prime product of TXs is p, from Algorithm2, we can get the new timely proof $tf_{m'}(x_n) \leftarrow \{dA_m^r, b'b\}$ and new accumulator $A_{m'} = A_m^p$,the conditions (2) provided:

$$\begin{gathered} a'x + b'p = 1 \\ r = a'b \\ \hat{d} = dA_m^r \\ \hat{b} = b'b \end{gathered} \tag{2}$$

If the proof update Algorithm2 is true, then equation (3) should be satisfied.

$$\hat{d}^x A_{m'}^{\hat{b}} = A_n \tag{3}$$

By procedure (3), we can verify that equation (4) always holds.

$$\begin{aligned} \hat{d}^x A_{m'}^{\hat{b}} &= (dA_m^r)^x A_{m'}^{\hat{b}} \\ &= d^x A_m^{a'bx} A_m^{pb'b} \\ &= d^x A_m^{b(a'x+b'p)} \\ &= d^x A_m^b = A_n \end{aligned} \tag{4}$$

Therefore, DNS Bridge returns a message in a format similar to <tx, block header, validity proof, timely proof>, where tx is a "set value" type of transaction that contains the <name, value> pair and the domain name provider's public key information.

4.4. Client Design

The client receives the messages returned by DNS Bridge and can verify the validity and timeliness of the message content. The prerequisite for the client to be able to perform validation is that 1) only some latest block header information needs to be synchronized 2) all MMR Peaks are saved.

Validity Verify:

Initially, users are required to download all MMR Peaks collections. After that, the MMR Peaks collection can be updated by itself each time a new block header is synchronized. The validity verify is divided into two steps: 1) Check if the Merkle Root of MMR Peaks is equal to the TXO C of the latest synchronized block header, if it is equal, proceed to the second step, otherwise resynchronize the block and check again. 2) Check Merkle path form transaction to TMR and Merkle path from TMR to MMR Peaks, if so, the verification is passes or succeeds, otherwise the verification fails.

Algorithm 3 Timely Proof Verify

```
Input:
$block_n$ accumulator $A_n$;
$block_m$ accumulator $A_m$;
    timely proof $tf_m(x_n)$;
    transaction $x_n$.
Output:
Verify result $true\ or\ false$.
1: $x_n \leftarrow HashToPrime(x_n)$
2: $a, b \leftarrow tf_m(x_n)$
3: return $d^x A_m^b == A_n$
```

Timely Verify:

The user receives the timely proof $tf_m(x_n)$ and the block header where x_n is located, and the existence proof of x_n has been verified by validity verify. We extract the accumulator field STXO_C A_n from the block header, the latest block STXO_C A_m, timely proof $tf_m(x_n)$and the transaction x_n that needs to be verified as parameter inputs, as shown in Algorithm 3. The $\Delta tf_m(x_n)$ can be verified similarly, but the input block header is $block_{n-\Delta}$ rather than $block_n$.

5. Cost Analysis and Evaluation

5.1. Experiment Settings and Parameters

Based on the source code of RSA-Accumulator[9], Merkle Tree[10], and Merkle Mountain Range, we implemented a prototype of NVBD, DNS Bridge, and Clien[11]t with Python language. We use the

[9]https://github.com/oleiba/RSA-accumulator
[10]https://github.com/Tierion/pymerkletools
[11]https://github.com/jjyr/mmr.py/blob/master/mmr/mmr.py

RSA accumulator with 3072 bit-modulus, 128 bits prime representative, and the Merkle root is set to 32 bytes. All these parameters are considered to be safe enough in the field of cryptography. We run all our experiments on our desktop computer equipped with one 3.7 GHz Intel Core i9 processor, 64 GB RAM, and perform 10 runs and report their average for each data point of running time. We test the performance of the accumulator update, proof generation and update, proof verification. We also find the problem that timely proof generation time is a little high and we give our solution. It should be noted that our LEA-DNS is an extension based on DNS. We don't care about the specific performance or implementation of the public blockchain, such as throughput, confirmation time, network structure, etc. We only consider the additional consumption when the public blockchain supports timeliness verification, which may be a limitation of this paper.

We denote that the interval between block generation is T, the size of the block header is S_h, the average size of transactions is S_t. For convenience, we assume that each transaction will consume one input (UTXO) and generate two outputs (UTXOs). Denote m as the average number of transactions per block. The average number of UTXOs consumed per block will be $m/2$, and the average number of UTXOs generated per block will be m. Suppose n is total the number of UTXOs and L is the length of the current blockchain state.

5.2. NVBD Extra Cost

NVBD's full node requires additional work to add to the original Bitcoin node to update the accumulator STXO_C and update TXO_C. The accumulator needs to be updated for each transaction in the block, and the time complexity of the update is *O(m)*. The update of the accumulator in a block can be divided into the process of calculating the product of all transactions using the *HashToPrime* function and the process of product modular exponentiation. The experimental results are shown in Figure 6, where the time grows linearly with the number of transactions in the block.

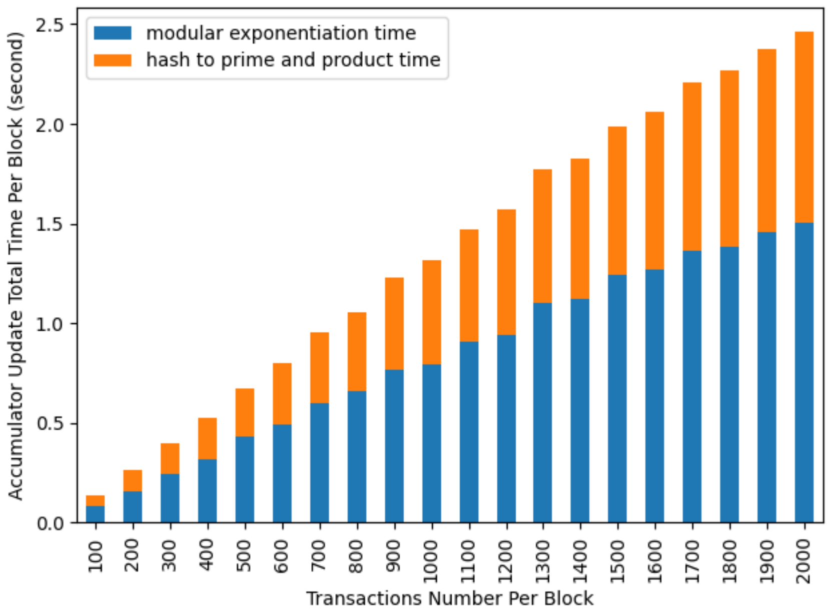


Figure 3. Performance of Accumulator Update Per Block.

The update of TXO_C is in two steps: the first step is to insert the new TMR to the MMR with a time complexity of $O(log(L))$. The second step is to construct the MMR Peaks into TXO_C via the Merkle structure. Since the number of MMR Peaks is $\lceil log(L) \rceil$, the time complexity of constructing TXO_C from MMR Peaks is $O(log(L))$. Thus the time complexity of TXO_C update will be $O(log(L))$. By the blue curve in Figure 6, even when the block height L grows to 2^{24}, the time to insert the TMR into the MMR is still less than 1ms.

5.3. DNS Bridge Cost

Firstly, DNS Bridge needs to synchronize all the data of NVDB with $S_t m + S_h$ scale each block generation, so the synchronization bandwidth between DNS Bridge and NVBD must be at least: $(S_t m + S_h)/T$. if we use the Bitcoin parameters, it only needs about 2Kb/s bandwidth. However, it needs to provide service for clients with high bandwidth.

Validity Proof Generation:

After data synchronization, DNS Bridge needs to construct a validity proof of transaction, construct a Merkle tree of new transactions to the TMR, and insert the TMR into the MMR. After the MMR tree is updated, the Bridge only needs to provide the Merkle path from the updated MMR each time clients request validity proof, and the time complexity is $O(1)$. Therefore, the time complexity for DNS Bridge to update the validity proof is $O(m) + O(log(L))$. By the orange curve in Figure 6, it takes only about 0.06ms to generate a TMR to TXO_C proof even when the block height grows to 2^{24}.

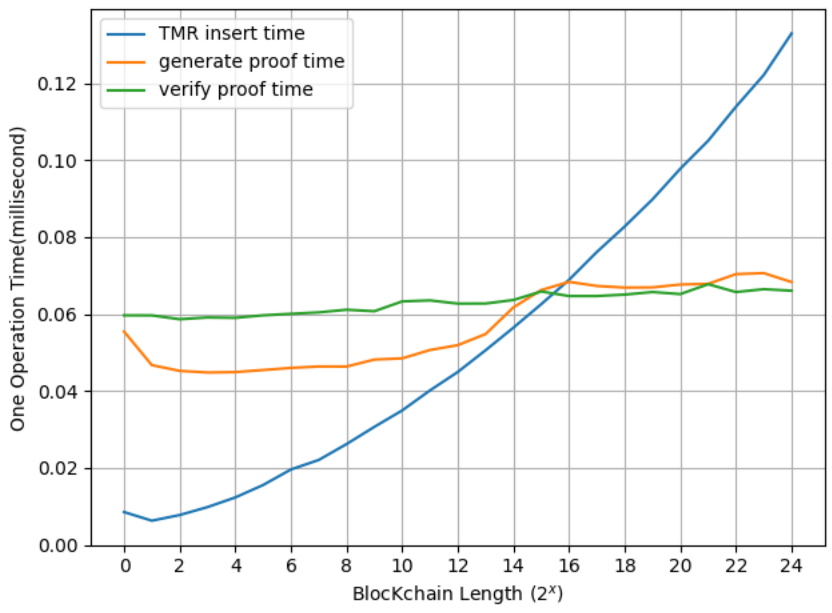


Figure 4. Perfomance of the MMR Operation With the Blockchain Growing.(Based on the first TMR).

Timely Proof Generation:

When the Client requests a timely proof of a DNS response, the timely proof needs to be generated if the transaction is newly generated. From Algorithm 1, the time complexity of timely proof generation is $O(m)$ for a block. And the time complexity of generating a $\Delta t f_m(x_n)$ is $\Delta O(m)$. If the transaction's timely proof already exists, then it needs to be updated or reconstructed. For each update from $\Delta tf_m(x_n)$ to $\Delta tf_{m+1}(x_n)$, the time complexity is $(\Delta + 1)O(m)$, which is the same as reconstructing a timely proof. However, we don't need to cache the previous STXO's prime products when using update operation. As we can see from the Figure 8, when the number of blocks (transactions) grows, the time it takes for timely proof to be generated will gradually increase. To keep the time within an acceptable range, we propose the idea of **phase validation**: the DNS Bridge provides only $\Delta tf_m(x_n)$, rather than full timely proof, where Δ has a limitation. A timely proof $\Delta tf_m(x_n)$ will be validated by many Clients and submitted to NVBD if an error is found, so outdated proofs($tx \notin STXO_{n:m-\Delta}$) can be omitted and we just need to keep the recent Δ range of proofs reliable if the error can be advertised to all the Clients during Δ blocks time. From the Figure 8, we can see that the timely proof generation will cost much time(second level), so DNS Bridge can also provide the $tf_{m-1}(x_n)$proof to ease the pressure of computing. The proof generation can also be easily accelerated by parallel computation.

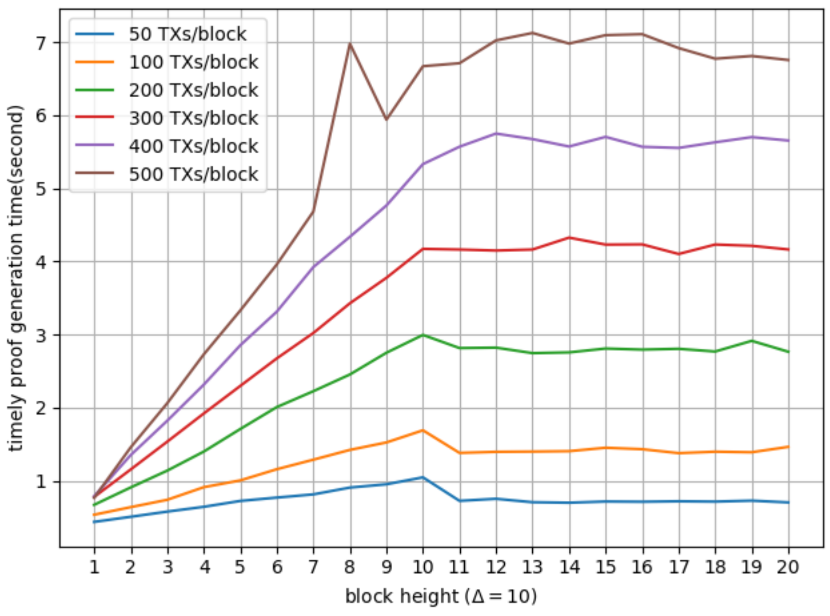


Figure 5. Performance of the Timely Proof Generation ($\Delta tf_m(x_n), \Delta = 10$).

5.4. Cilent Cost

Client will get <tx, blockheader$_n$, valid proof, timely proof> from DNS Bridge and blockheader$_m$ from NVBD as a response. Suppose $m < 2^{10}, L < 2^{20}, \Delta = 10$ and each transaction is assumed to be 300 bytes. Client also need to get the latest MMR Peaks when you first start, they are less than $32 * 20 = 640(bytes)$. The blockheader adds two accumulators (STXO_C and Pre_STXO_C) and TXO_C compared with the original Bitcoin, and its size is assumed to be 80(Bitcoin blockheader)+2*384(RSA accumulator)+32(TXO_C) = 880(bytes).The spatial complexity of validity proof is $O(log(m)) + O(log(L))$, so the size of validity proof size is at most 32 * (10 + 20) = 960(bytes).The size of timely proof is fixed to two constants, and the total size is 416 bytes. The blockheader$_m$ size is 496 bytes. Therefore, each time you interact with DNS Bridge, the size of the validation data is less than:

$$300 + 880 + 960 + 416 + 496 = 3052(\text{bytes})$$

The local data that needs to be maintained is all the MMR Peaks, with its spatial capacity scale being $log(L)$ and the latest blockheaders from $\mathrm{blockheader}_{m-\Delta}$ to $\mathrm{blockheader}_{m}$. At the first time when the client starts, the data size for synchronization will be less than:

$$640 + 10 * 880 = 9440(bytes)$$

Validity Verify Time:

Firstly, client need to insert TMR_{m} into MMR to update the local MMR Peaks, the time complexity is $O(log(L))$. Next, generate Merkle root from all MMR Peaks, and compare whether it is equal to $\mathrm{TXO\_C}_{m}$, whose time complexity is also $O(log(L))$. Verifying the path from transaction tx to TMR_{n}has a time complexity of $O(m)$, followed by verifying the path from TMR_{n} to MMR root, which has a time complexity of O(log(L)). Therefore, the time complexity of each validity verify is $O(m) + 3O(log(L))$ which includes TMR insert time and verify proof time in Figure 7. The insert time is less than 0.14ms by the blue curve and the verify proof time is about 0.06ms through the green curve. So, the total time is less than 0.2ms when L grows to 2^{24}.

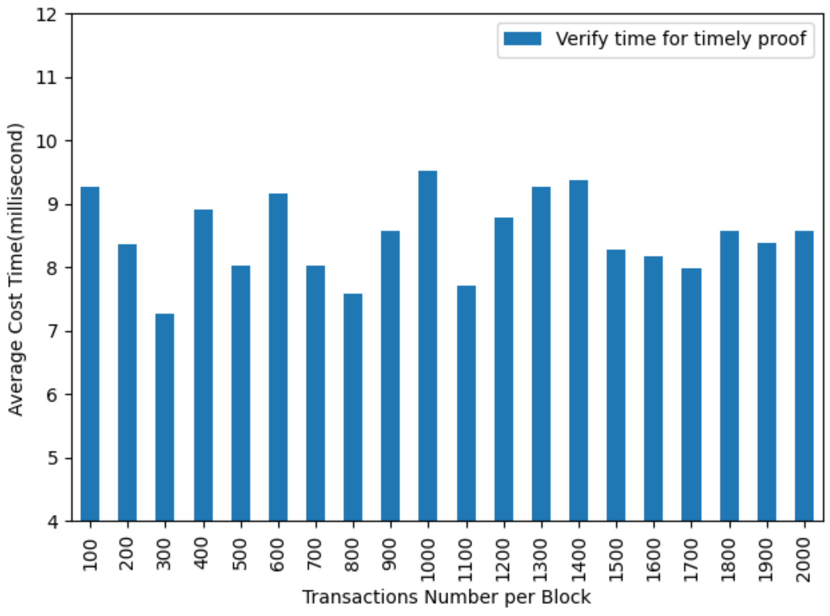


Figure 6. Performance of the Timely Proof Verify.

Timely Verify Time:

Timely proof only needs to perform a $HashToPrime$ and a modular exponentiation operation, thus the time complexity is $O(1)$. From experiments, the timely verify time is less than 10ms through the Figure 9, which is almost no burden to the Client. We conclude with a summary of the temporal complexity of three components and the time cost level, as show in Table 1.

Table 1. Time complexity and cost for NVDB full node, DNS bridge, and Client every block or transaction generation round.

Part		Time Complexity	Cost Time Level
NVBD	Acc Update	O(m)	second
	MMR Insert	O(log(L))	millisecond
DNS Bridge	Validity Proof	O(m)+O(log(L))	millisecond
	Timely Proof	$\Delta O(m)$	second
Client	Validity Verify	O(m)+O(log(n))	millisecond
	Timely Verify	O(1)	10 milliseconds
	Message Size	< 3.052Kbytes	
	Storage Size	< 9.44Kbytes	

6. CONCLUSIONS

This paper proposed a blockchain-based decentralized naming system called LEA-DNS to solve the centralization problem and data authenticity problem. We find the problem of record obsolescence in the blockchain when DNS <name, value> has been changed and propose our solution. LEA-DNS enables name owners to apply domain names and maintain authoritative server information on blockchain in a decentralized manner which mainly consists of NVBD, DNS, and Clients. The UTXO mechanism, RSA accumulator, and Merkle Mountain Range have been used for the blockchain design called NVDB. DNS Bridge will generate the validity proof and timely proof for the verifiable DNS request and the response size is only a few hundred bytes. Clients will verify the response with little time. Our simulated results show that the Clients will only need storage no more than 9.44Kb data locally, the overhead of verification message is less than 3.052Kb and the verification time is below 10ms. LEA-DNS is also compatible with current legacy DNS architectures. In the future work, we will deploy this system in a real network to further test its performance.

ACKNOWLEDGEMENTS

This paper is supported by the national Science Foundation of China, under the project No.61472476.

REFERENCES

[1] Boneh, D., B¨unz, B., Fisch, B.: Batching techniques for accumulators with applications to iops and stateless blockchains. In: Annual International Cryptology Conference. pp. 561–586. Springer (2019).

[2] Bunz, B., Kiffffer, L., Luu, L., Zamani, M.: Flyclient: Super-light clients for cryptocurrencies. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 928–946.

[3] Chen, H., Wang, Y.: Minichain: A lightweight protocol to combat the utxo growth in public blockchain. Journal of Parallel and Distributed Computing 143, 67 – 76(2020).

[4] Chepurnoy, A., Papamanthou, C., Zhang, Y.: Edrax: A cryptocurrency with stateless transaction validation. IACR Cryptol. ePrint Arch. 2018, 968 (2018).

[5] Fazio, N., Nicolosi, A.: Cryptographic accumulators: Defifinitions, constructions and applications. Paper written for course at New York University: www. cs. nyu.edu/nicolosi/papers/accumulators. pdf (2002).

[6] He, G., Su, W., Gao, S., Yue, J.: Td-root: A trustworthy decentralized dns root management architecture based on permissioned blockchain. Future Generation Computer Systems 102, 912 – 924 (2020).

[7] Huynh, T.T., Nguyen, T.D., Tan, H.: A decentralized solution for web hosting. In: 2019 6th NAFOSTED Conference on Information and Computer Science (NICS). pp. 82–87.

[8] Jiang, Y., Bai, H., Yang, H.: The messaging model design based blockchain and edge computing for the internet of things. In: 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS). pp. 604–608.

[9] Jin, Y., Fujikawa, K., Harai, H., Ohta, M.: Secure glue: A cache and zone transfer considering automatic renumbering. In: 2015 IEEE 39th Annual Computer Software and Applications Conference. vol. 2, pp. 393–398.

[10] Jones, B., Feamster, N., Paxson, V., Weaver, N., Allman, M.: Detecting dns root manipulation. In: International Conference on Passive and Active Network Measurement. pp. 276–288. Springer (2016).

[11] Kalodner, H.A., Carlsten, M., Ellenbogen, P., Bonneau, J., Narayanan, A.: An empirical study of namecoin and lessons for decentralized namespace design. In: WEIS. Citeseer (2015).

[12] Li, J., Li, N., Xue, R.: Universal accumulators with efficient nonmembership proofs. In: Katz, J., Yung, M. (eds.) Applied Cryptography and Network Security. pp. 253– 269. Springer Berlin Heidelberg, Berlin, Heidelberg (2007).

[13] Patsakis, C., Casino, F., Lykousas, N., Katos, V.: Unravelling ariadne's thread: Exploring the threats of decentralised dns. IEEE Access 8, 118559–118571 (2020).

[14] Ren, S., Liu, B., Yang, F., Wei, X., Yang, X., Wang, C.: Blockdns: Enhancing domain name ownership and data authenticity with blockchain. In: 2019 IEEE Global Communications Conference (GLOBECOM). pp. 1–6.

[15] Shi, P., Liu, H., Yang, S., Zhang, Y., Zhong, Y.: The inherent mechanism and a case study of the constructional evolution of the jointcloud ecosystem. IEEE Internet of Things Journal 7(3), 1561–1571 (2020).

[16] Trostle, J., Van Besien, B., Pujari, A.: Protecting against dns cache poisoning attacks. In: 2010 6th IEEE Workshop on Secure Network Protocols. pp. 25–30.

[17] Yoon, W., Im, J., Choi, T., Kim, D.: Blockchain-based object name service with tokenized authority. IEEE Transactions on Services Computing 13(2), 329–342 (2020).

AUTHORS

Ting Xiong received the bachelor's degree from National University of Defense Technology, Changsha, China, in 2019. He is currently pursuing the engineering degree with the National University of Defense Technology, Changsha, China. His current research interests include blockchain architecture, consensus algorithm, and application.