B-DNS: A Secure and Efficient DNS Based on the Blockchain Technology

Zecheng Li, Shang Gao, Zhe Peng, Songtao Guo, *Senior Member, IEEE,* Yuanyuan Yang, *Fellow, IEEE,* and Bin Xiao *Senior Member, IEEE*

Abstract—The Domain Name System (DNS) plays a crucial role in the Internet. However, it is vulnerable to many attacks such as the cache poisoning attack and DDoS attack. Though some countermeasures have been proposed, they still have some limitations. In this paper, we propose B-DNS, a blockchain-based domain name system, which can provide a secure and efficient DNS service. B-DNS fills up two shortcomings of current blockchain-based DNS, namely computation-heavy Proof-of-Work (PoW) protocol and inefficient query, by building a Proof-of-Stake (PoS) consensus protocol and an index of domains. We propose a novel way to quantitatively compare the security of B-DNS and legacy DNS in terms of attack success rate, attack cost, and attack surface. Our experiments show that the probability of a successful attack on B-DNS is 1% of a successful attack on legacy DNS, the attack cost goes up a million times in B-DNS, and the attack surface of B-DNS is far less than that of legacy DNS. The query performance evaluation of B-DNS shows that B-DNS can achieve similar or even less query latency than state-of-the-art commercial DNS implementations.

Index Terms—Blockchain, Domain Name System, Cache Poisoning Attack, DDoS Attack.

1 Introduction

IP addresses are unique identifiers of Internet resource. Anyone who wants to visit some specific resources must know their IP addresses. However, it is hard to remember alphabetized domain names compared with numerical IP addresses. Accordingly, the DNS is designed to provide a domain name to IP address mapping service so that people could approach resources on Internet easily by only remembering their domain names. Unfortunately, researchers have exposed several vulnerabilities in current DNS such as the weak verification mechanism and single point failure of name servers, which causes different attacks.

The weak verification mechanism of current DNS leads to the cache poisoning attack [1], [2]. In cache poisoning attacks, attackers can send well-crafted response packets when a recursive resolver updates cache. Once a forged DNS entry is injected into the cache successfully, clients under the victim recursive resolver will be redirected to a phishing website when they visit the affected domain. The biggest banks of Brazil, Bandesco was reported to be attacked in this way [3].

The single point failure makes legacy DNS vulnerable to DDoS attacks. Currently, DNS domains are structured in a tree and each node stores a bunch of IP addresses of its sub-domians. Once the name server of a critical domain is under the DDoS attack, the service of its sub-domains will be disrupted. In this way, DDoS attacks that targeting key servers can significantly collapse the availability of partial legacy DNS [4]. The more essential the server, the more serious the consequence of the DDoS attack. In 2016, a DNS service provider Dyn was attacked in this way [5].

Several methods have been proposed to address these attacks. Approaches against the cache poisoning attack can be categorized into two types: employing the Domain Name System Security Extensions (DNSSEC) [6] and increasing the entropy of query packets [7], [8], which provides more information for recursive resolvers to distinguish a valid response packet. Methods to mitigate the DDoS attack mainly focus on storing more resource records in the cache [9], [10], which makes domains can still be resolved even its parent domain is under attack.

However, we observe that there are still some limitations in these countermeasures as they aim to defend attacks rather than repair the vulnerabilities. We noticed that DNSSEC suffers from low deployment rate by launch a quick scan over the Alexa top 1000 .com/.net/.org domains and found only 3% domains support DNSSEC. Merely increasing the entropy of query packets can decrease the success rate of attack to some extent. However, as the development of computation power and network bandwidth, this defense becomes weaker and weaker. In DDoS defense, there exist the probability that queried domains are not cached. Moreover, these methods do not work when authoritative servers are under attack. As to the T-DNS, its security depends on the TLS protocol, which employs certificates issued by the certificate authority (CA). However, the centralized CA is not secure since it suffers from the single point failure and unauthorized issued certificates.

Facing these challenges and inspired by the promising features of blockchain, we intend to build a secure do-

- *Z. Li, S. Gao, and B. Xiao are with the Department of Computing, Hong Kong Polytechnic University, Hong Kong.*
  E-mail: {cszcli,csshanggao,csbxiao}@comp.polyu.edu.hk
- *Z. Peng is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong.*
  E-mail: pengzhe@comp.hkbu.edu.hk
- *S. Guo is with the College of Computer Science, Chongqing University, China.*
  E-mail: guosongtao@cqu.edu.cn
- *Y. Yang is with the Department of Computer Science, Stony Brook University, NY, USA.*
  Email: yuanyuan.yang@stonybrook.edu

Manuscript received April XX, 20XX; revised August XX, 20XX.
(Corresponding author: B. Xiao.)

main name system based on the blockchain. In blockchain, blocks are chained sequentially by encapsulating the hash of their previous blocks as the `PrevHash` into headers. Transactions are stored in blocks, and each block calculates a `MerkleRoot` to provide an easy way to verify the integrity of transaction records. By combining `PrevHash` and `MerkleRoot`, blockchain can guarantee the stored data is tamper-proof. Additionally, blockchain's underlying peer-to-peer network provides great resilience to the single point failure and DDoS attack. However, building a blockchain-based DNS is quite tricky. It is not merely storing resource records in the blockchain. Many challenges arise in the way to implement a secure and efficient blockchain-based DNS system.

Stored Data are Immutable. Once a resource record is written into the blockchain, it is hard to modify the content. However, there exists the need to update DNS records because domain owners may change the IP addresses of their domains. Accordingly, some new designs should be adopted to provide flexible record updates.

Blockchain Performance is Poor. The primary search operation in the blockchain is slow while the name service is time-sensitive. Therefore, some schemes should be designed to speed up the search process in a blockchain.

New Vulnerabilities May be Introduced. Though blockchain provides good features such as data tamper-proof and DDoS resilience, it may introduce new vulnerabilities such as inconsistent data across nodes and mining attack, which are inherent problems of the blockchain [11]. Accordingly, how to build a blockchain-based DNS without introducing new security problems of the blockchain is a big challenge.

In this paper, we propose B-DNS, a secure and efficient domain name system based on the blockchain. B-DNS stores DNS records as transactions and leverages an index to accelerate blockchain searches to provide efficient name service. B-DNS is compatible with the legacy DNS system, i.e., recursive resolvers and users can interact directly with B-DNS name servers. Our contributions can be summarized as follows:

We alleviate the computation-heavy PoW consensus protocol utilized in current blockchain-based DNS. By proposing a biased-coin flipping protocol and a distributed random-number generation (DRG) protocol, B-DNS builds a Proof-of-Stake (PoS) consensus protocol. The security of B-DNS PoS consensus protocol will not be affected by the amount of computation power.

We address the problem of inefficient query in current blockchain-based DNS. We build an index tree for B-DNS and propose a search algorithm to increase the query speed. Our experiment results show that B-DNS can provide similar query performance with state-of-the-art commercial DNS implementations.

We propose a novel way to quantitatively compare the security of B-DNS and the legacy DNS in terms of the attack success rate, attack cost, and attack surface. To the best of our knowledge, this is the first time that researchers quantitatively compare the security of blockchain-based systems with traditional systems. Experiments show that B-DNS is much securer than legacy DNS.

The rest of this paper is organized as follows: In Section 2, we introduce the background of legacy DNS and the blockchain technology. Then, we bridge the gap between the legacy DNS and B-DNS in Section 3. In Section 4, we give the detailed design information of B-DNS. Section 5 presents experiment results related to the security and performance of B-DNS. In Section 6, we discuss how B-DNS handles some other potential attacks. Section 7 summarizes the related work and we conclude our work in Section 8.

2 Background

2.1 The Overview of Legacy DNS

DNS nameservers are organized as a tree, and the namespace is separated into layers. In each layer, the namespace is partitioned into non-overlapping regions called domains. A domain owner formulates the domain policy and keeps track of its sub-domains. The root node of DNS tree is called the *root zone*, which stores the delegation information of its leaf nodes. The leaf node domains of the DNS tree are called the top-level domain (TLD). There exist different kinds of TLDs, of which the most widely-used ones are country-code TLD (ccTLD) and generic TLD (gTLD). The third level of the DNS tree is usually represented as the second-level domain.

Each domain operates several authoritative servers, which stores DNS records in the form of *resource record*. There are many types of resource record such as `A/AAAA`, `NS`, and `SOA`. The `A/AAAA` resource record is responsible for IPv4 and IPv6 address resolution. The `NS` record stores the name of a authoritative server. Since DNS is hierarchical, a query packet should traverse from the root zone to the target authoritative server layer-by-layer, which is called *recursive resolution*.

2.2 The Problems of Legacy DNS

A DNS query packet utilizes the transaction ID (TXID) to distinguish a valid response packet from forged ones. However, the recursive resolver typically increments TXID from zero, which makes it easier to guess. An attacker can first initiate a query to the recursive resolver and then forge response packets, which try all possible TXIDs, to deceive the recursive resolver that it is a valid response packet [2]. Once the recursive resolver accepts the forged packet, the attacker succeeds in poisoning the cache. Dan Kaminsky proposed an improvement to the cache poisoning attack and made it more effective [12]. Kaminsky's attack adds a non-existent sub-domain name to the victim domain. For example, it sends a query for the *ns1.example.com* when it wants to poison the cache of *example.com*. Accordingly, if the first trial failed, a Kaminsky attacker can immediately launch another attack, which queries for *ns2.example.com*. After evolving to the Kaminsky's attack, the danger of the cache poisoning attack increases significantly.

Additionally, the DDoS attack that targets legacy DNS often happens. Once a higher-level domain is under the DDoS attack, the availability of its sub-domains could be greatly degraded. In history, the root server and some TLD authoritative servers have been attacked by DDoS attack several times [4]. Some attacks did succeed in disabling the victim DNS servers and caused parts of the Internet experiencing severe domain resolution problems.

2.3 Blockchain

The blockchain technology is derived from the Bitcoin, which was proposed in 2009 by Satoshi Nakamoto [13]. A blockchain system is intrinsically a distributed ledger. Participants collaborate with each other to maintain the system operation and periodically elect node to write new content into the ledger through Byzantine fault tolerant consensus protocol [14], [15]. Tamper-proof is one of the main features provided by the blockchain. Specifically, all transactions are put in the leaf nodes of a Merkle Tree and an iterative calculation process will proceed until the `MerkleRoot` is calculated, which is encapsulated in the block header so that the user can easily verify the integrity of the transaction data. Moreover, blocks are chained sequentially by calculating the hash value of the previous block's header, which is called the `prevHash`. It is very tough to calculate a valid `PrevHash` value so that once a transaction is confirmed (i.e., has 6 sequential blocks), subverting it can be exponentially hard. Accordingly, the combination of `MerkleRoot` and `prevHash` constitutes the cornerstone of the tamper-proof feature of the blockchain.

3 From Legacy DNS to B-DNS

In this section, we describe the changes from legacy DNS to B-DNS to provide a smooth transition. We also discuss some considerations when designing B-DNS and give a formal definition of the B-DNS blockchain.

3.1 What's the Differences?

The difference between legacy DNS and B-DNS is mainly reflected in three aspects: the management of DNS records, the way that name servers structured, and the domain resolution of domain names.

In legacy DNS, DNS records are managed by domain owners, who operate authoritative name servers. In this case, domain owners can update, add, or delete DNS records by changing the resource records in authoritative name servers. In B-DNS, as DNS records are stored in the blockchain, the management of DNS records are conducted by different types of transactions.

In legacy DNS, name servers are structured in a tree. The name servers in each layer store the IP addresses of their sub-domains. By contrast, B-DNS name servers are structured in a peer-to-peer network. Each name server either stores a full copy or metadata of the blockchain.

In legacy DNS, the domain resolution is conducted by the recursive resolver, who maintains a cache to store frequently-queried DNS records. Once a queried domain name is not cached, the resolver will conduct recursive resolution to acquire the asked DNS record. In B-DNS, as DNS records are stored in blockchain, end-users can directly query the name servers with complete blockchain data.

3.2 Considerations of B-DNS's Consensus Protocol

Current mainstream blockchain consensus protocols includes PoW, PoS, and Practical Byzantine Fault Tolerance (PBFT).

PoW has been widely studied and verified for its security and performance. However, an empirical study on the Namecoin has exposed some problems of using PoW in blockchain-based naming system [16]. A PoW system is always exposed to 51% attack, which can only be prevented by enlarging the system's overall computation power. More computation power brings in extra resource consumption, which introduces additional system maintenance cost. Bitcoin's security is guaranteed by its huge computing power, whose annual electricity consumption is around 61.4 TWh. Moreover, for smaller or newer PoW blockchain systems, it may only require 5% of Bitcoin's computing power to reach 51% of its computing power, which is easy and affordable. Though adopting merged mining with Bitcoin provides a viable solution to the above problem, the high maintenance cost it introduces will become the burden on system maintenance.

PBFT consensus protocol can enable high-throughput transaction processing, low-latency confirmation, and good security properties. However, its excessive usage of the network to transmit consensus packets makes it difficult to scale to larger sizes. Experiments have demonstrated that the throughput of PBFT consensus protocol drops exponentially after the number of nodes exceeds 64.

The PoS consensus protocol assigns the blockchain generation right based on the amount of stake controlled by each node. The proportion of stake determines the probability of being selected. In addition, compared with the PBFT protocol, PoS is more scalable because once the leader is selected, consensus is reached. Based on the above considerations, we chose PoS as the B-DNS consensus protocol.

3.3 Formal Definition of B-DNS Blockchain

We give the formal definition of B-DNS blockchain in this section. In B-DNS, the leader election is conducted according to discrete-time units.

Definition 1. (Slot). In B-DNS, time is divided into discrete units called *slot*, which is represented as $sl_j; j \in \mathbb{Z}^+$.

All the registries are equipped with roughly synchronized clocks that could indicate the current slot. Each slot owns a slot leader L_j, who is responsible for issuing a new block. However, limited by the network latency, the leader election process cannot be executed slot-by-slot. Accordingly, a larger time unit *epoch* is also defined.

Definition 2. (Epoch). The *epoch* is defined as a set of adjacent slots. Each epoch consists of R slots and is denoted as $e_x; x \in \{1; 2; \ldots\}$. Specifically, $e_x = \{sl_{xR+1}; sl_{xR+2}; \ldots; sl_{(x+1)R}\}$.

Definition 3. (Block). A *block* B_j issued at slot sl_j contains the current state $st_j \in \{0; 1\}^*$, data $d \in \{0; 1\}^*$, the slot number sl_j and a signature $= Sign_{sk_i}(st_j; d; sl_j)$ signed using the private key sk_i of the slot leader R_i.

Definition 4. (Genesis Block). The *genesis block* B_0 contains the list of registries identified by their public keys and stakes

$S^0 = \{(vk_1; s_1^0); \ldots; (vk_n; s_n^0)\}$ and initial randomness ^{0}, which is used to seed the leader election function.

Definition 5. *(Blockchain).* A *blockchain* relative to the genesis block B_0 is a sequence of blocks $B_1; \ldots; B_n$ associated with a strictly increasing sequence of slots. The length of a chain $len(C) = n$ is its number of blocks. The block B_n is the head of the chain, denoted $head(C)$. B-DNS treats the empty string $''$ as a legal chain and by convention set $head('') = ''$.

Definition 6. *(State).* The *state* is defined as a string $st \in \{0; 1\}$ that represents the balance of each account. Specifically, the state st_j of B_j is equal to $H(B_{j-1})$, where H is a prescribed collision-resistant hash function.

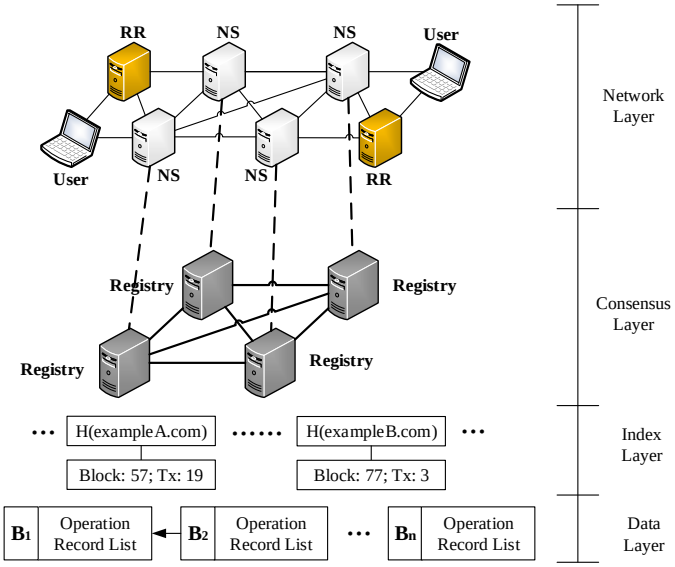


Fig. 1. The 4-layer architecture of B-DNS. The data layer stores DNS records. The index layer maintains a domain index. The consensus layer ensures data consistency. The network layer provides peer-to-peer connections.

4 System Design

In this section, we introduce the detailed design information of B-DNS according to the 4-layer architecture, which is depicted in Fig. 1. B-DNS's 4-layer architecture provides good extensibility, and the loosely-coupled relationship between different layers makes it possible to upgrade a layer solely in the future without changing the operating logic of other layers.

4.1 Data Layer

In the data layer, B-DNS stores DNS records as transactions in the blockchain. Since blockchain transactions are immutable once confirmed, we propose a new format called *operation records* to store resource records. There are three types of operation records in B-DNS: registration, update, and revocation.

Registration. In legacy DNS, the right of domain registration is controlled by official registries (e.g., Verisign) and registrars (e.g., GoDaddy). In B-DNS, to be compatible with legacy DNS, new domain names still should be registered with the corresponding registries. After successfully registered, the new domain record with its valid period are signed and encapsulated into a registration record by the registry. Additionally, the address (i.e., hash of public key) that is controlled by the domain owner is also added to the registration record. The registry signature enables the slot leaders to verify whether the record is legitimate, while the address of the domain owner is left for further update.

Update. Dynamic update is one of the major concerns when designing the B-DNS. Considering a scenario where the IP address changes, the corresponding registration record needs to be updated to map to new address. The update operation is defined to meet these requirements. Similar to the registration record, the update record is signed and broadcasted by the registry. However, an update record needs to redeem its corresponding registration record first, which can only be conducted by its domain owner. Otherwise, it cannot pass the verification and will not be included in the blockchain.

Revocation. As the name suggests, the revocation record is used to terminate the ownership of a domain name. An expired domain will be revoked automatically. The registry will issue a revocation record to terminate its ownership. Similar to the update record, the revocation record should first redeem its registration or update record.

The idea of operation record is inspired by Bitcoin's scripting system. Bitcoin scripting system only allows the change of coins' ownership in a transaction. Operation record not only enables the change of domain ownership but also can change the transaction content. B-DNS blockchain also stores the stake of each registry as well as their public keys in the block header. The stake information updates every epoch, which makes the PoS consensus protocol in accordance with the latest state.

4.2 Index Layer

In the index layer, B-DNS maintains an index to increase the search speed. Searching DNS records in a blockchain is time-consuming as data are structured in a linked list. However, DNS service is time-sensitive. If the target record is located in the latest block, a DNS query needs to take a long time. In this case, we build an index tree to map domain names to their IP addresses, where keys are hashes of domains and values are corresponding IP addresses. B-DNS also encapsulates a `IndexHash` into the block header, which stores the hash value of the index of current block. The `IndexHash` enables B-DNS name servers to verify the correctness of generated indexes.

We design an update algorithm and a search algorithm for our constructed index, which are presented in Algorithm 1 and Algorithm 2. The update algorithm can recursively insert new records into the tree. It first compares the target value with the root node value. Then, the tree can route to the target node so that we could insert new value. The worst-case complexity of the update algorithm is $O(\log_2 n)$, where n is the number of nodes in the tree. The search algorithm works on an updated tree. It first checks the root node. If the root node is empty then returns an error. Otherwise, it goes down according to the node's value. Searching

Algorithm 1: The Update Algorithm

```
Data: domain name: key; IP address: val
Result: An updated index tree
initialization;
while receiving a new block do
    validate the received block;
    key = hash(domain name);
    val = IP address;
    if node.root = null then
        return a new tree with node(key, val);
    endif
    if key < node.key then
        return node = node.left;
    else if key > node.key then
        return node = node.right;
    else
        node.val = val;
    endif
end
```

Algorithm 2: The Search Algorithm

```
Data: A DNS query
Result: updated index tree
initialization;
key = domain name;
while receiving a new query do
    parse the DNS query;
    if node.root == null then
        return null;
    endif
    if key < node.key then
        return node.left.key);
    else if key > node.key then
        return node.right.key);
    else
        return node.val;
    endif
end
```

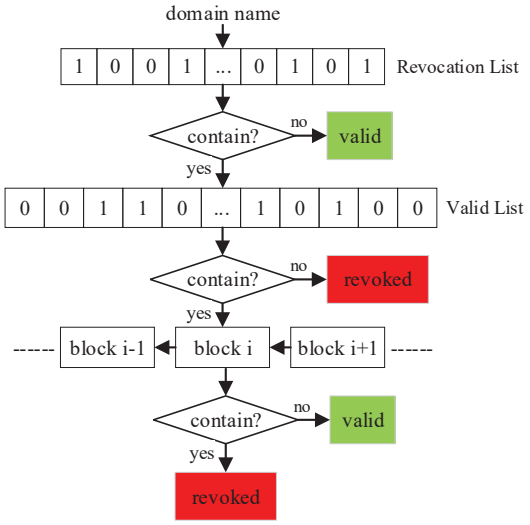


Fig. 2. B-DNS's revocation checking workflow.

TABLE 1
Summary of notations

Notation	Description
R_i	The i-th registry
vk_i	The verification key of registry R_i
sk_i	The secret key of registry R_i
s_i	The stake held by registry R_i
sl_j	The basic time unit, called *slot*
L_j	The slot leader in slot sl_j
e_x	A set of continuous time slots, called *epoch*
B_i	The block issued in slot sl_i
st_i	The state of the blockchain in slot sl_i
	The signature calculated by the slot leader
$\mathcal{C}$	The current blockchain, used in PoS
C	A set of candidate blockchain, used in PoS
S^x	The stake distribution $\{(vk_1; s_1^x); \ldots; (vk_n; s_n^x)\}$
$_x$	The randomness used to select slot leaders

a specific domain takes time $O(\log_2 n)$ in the worst case, where n is the number of domains in the updated tree. In this way, B-DNS can find the target value in the index promptly.

We also establish two bloom filters, which consist of a revocation list and a valid list, to enable fast domain revocation checking. The workflow of B-DNS revocation checking algorithm is depicted in Fig. 2. B-DNS first checks whether the domain name is in the revocation list. If not, the domain name must be valid. Otherwise, we check whether the domain name is in the valid list. If not, the domain name must be revoked. If it is in the valid list, we need to find this domain in the blockchain for a certainty.

4.3 Consensus Layer

In the consensus layer, B-DNS implements a PoS consensus protocol to ensure the consistency of DNS records. A leader election function is executed every epoch to select block generators. The B-DNS PoS consensus protocol ensures that a registry R_i holds the probability proportional to its stake s_i to be elected as the block generator:

$$P(R_i) = \frac{s_i}{\sum_{m=1}^{n} s_m}$$

In B-DNS, the stake of a registry is defined as the number of domains registered with it. However, this number is zero before the system initialization. In this case, all participants' stakes are set as $s_i = \frac{1}{n}$ in the genesis block so that all the registries have equal probability to be elected as the slot leader in the first epoch. In B-DNS, the stake updates every epoch since the distribution of registered domains is continually changing.

The key point is to construct a progressive protocol that can select leaders according to the pre-defined probability. The B-DNS PoS consensus protocol flips a p_i-biased coin to achieve this goal where

$$p_i = \frac{s_i}{\sum_{m=i}^{n} s_m}$$

If the output of this biased coin is 1, then the i-th registry is selected as the slot leader. Otherwise, the protocol proceeds and it flips a p_{i+1}-biased coin where

$$p_{i+1} = \frac{s_{i+1}}{\sum_{m=i+1}^{n} s_m}$$

Note that $p_n = s_n / s_n = 1$ so that this protocol always outputs a deterministic slot leader.

In B-DNS, the randomness seeds are generated distributionally. All the registries follow a distributed random-number generation (DRG) protocol to generate a distributed number, which acts as the seed in the p_i-biased coin flipping. The DRG protocol contains three phases:

Commitment Phase. When epoch e_x starts, each registry R_i samples a uniformly random string u_i and randomness r_i for the underlying commitment scheme, generates shares $\{s_1^i, s_2^i, \ldots, s_N^i\} \leftarrow Deal(N, u_i)$ and encrypts these shares under the public key of registry $R_1, R_2, \ldots, R_N$. Finally, R_i posts the encrypted shares and commitments $Com(r_i, u_i)$ onto the blockchain.

Reveal Phase. In the reveal phase, the registry R_i distributes the key to open its commitment by posting $Open(r_i, u_i)$ onto the blockchain.

Recovery Phase. When all shares $\{s_1^i, s_2^i, \ldots, s_N^i\}$ distributed by R_i are available, the other registries can compute $Rec(s_1^i, s_2^i, \ldots, s_N^i)$ to reconstruct u_i. Then, the randomness for the next epoch is calculated by $u_1 \oplus u_2 \oplus \cdots \oplus u_N$.

By leveraging the proposed DRG protocol and p_i-biased coin flipping protocol, the B-DNS PoS consensus protocol can select leaders proportional to their stakes. Then, we introduce the detailed PoS consensus protocol in Fig. 3, which defines the operations each registry should follow as well as the corresponding encryption mechanisms that ensure the data consistency.

Protocol Π_{PoS}

Π_{PoS} is operated by a set of registries $\{R_1, \ldots, R_n\}$. It proceeds as follows:

1.Initialization. At the very begining, each registry R_i generates its secret key and verification key (sk_i, vk_i) and puts it into the genesis block. Then, the registry R_i sets the local blockchain $\mathcal{C} = B_0 = (S^0, \rho^0)$ and the initial state $st_0 = H(B^0)$.

2.Chain Extension. In a typical slot sl_j, each online registry R_i performs the following jobs:

1) **Update Stake.** In epoch e_x, each registry updates its stake using the data drawn from latest block. The stake distribution is updated as $S^{x+1} = \{(vk_1, s_1^{x+1}), \ldots, (vk_n, s_n^{x+1})\}$.
2) **Update Randomness.** In epoch e_x, a registry R_i needs to update the randomness ρ^x for the upcoming epoch. They will communicate with each other and come up with a random number ρ^{x+1} via the DRG protocol, which is used to seed the leader election function for epoch e_{x+1}.
3) **Collecting Valid Chains.** Once a registry R_i is selected as the slot leader in slot sl_j, it needs to collect all valid chains and put them into a candidate chain set $\mathbb{C}$. Then, the registry R_i verify whether all the candidate chains are valid. In this case, R_i computes $\mathcal{C} = \max\text{valid}(\mathcal{C}, \mathbb{C})$, sets $\mathcal{C}$ as the current chain $\mathcal{C} = \mathcal{C}$ and sets state $st = H(head(\mathcal{C}))$.
4) **Issuing New Block.** If R_i is the slot leader in the slot sl_{xR+j} of epoch e_x, it generates a new block $B_{xR+j} = (st, d, sl, \sigma)$, where st is the state of the former block (i.e., $st = H(B_{xR+j-1})$), $d \in \{0, 1\}$ is the stored operation records data and $\sigma = Sign_{sk_i}(st, d, sl)$ is a signature on (st, d, sl). R_i appends the newly-generated block to the current chain $\mathcal{C} = \mathcal{C}|B$, broadcasts block B_{xR+j}, sets $\mathcal{C}$ as the new current chain and sets state $st = H(B_{xR+j})$.

3. Broadcasting Operation Records. Once a client registers a domain with registry R_i, R_i will create an operation record *op* and broadcast it.

Fig. 3. The B-DNS PoS consensus protocol. Each registry should follow it to extend the blockchain and broadcast valid operation records.

4.4 Network Layer

In the network layer, B-DNS provides three interfaces to enable different types of communication with different entities. In this case, B-DNS not only communicate with peers, but also provides domain name service to resolvers and end-users.

The interface between B-DNS name servers is similar to current blockchain systems, i.e., peer-to-peer communication. To be compatible with legacy DNS, B-DNS's network layer is designed to be able to respond to queries from recursive resolvers and end-users. In this case, B-DNS name servers can generate response packets according to the format of resource record and send back to the recursive resolvers and end-users. A B-DNS name server can respond to the query directly by fetching the required DNS record from the blockchain.

B-DNS Name Servers to B-DNS Name Servers: The interactions between B-DNS name servers are mainly responsible for data transmission.

inv. It allows a node to advertise its knowledge about the blockchain.

getblock. A name server sends this message with its highest block number to get a list of unstored blocks from its peers.

getdata. This is used to respond to *inv* message. After receiving an *inv* message, a B-DNS name server checks if there are any unstored DNS records. If so, it sends a *getdata* message to require the lost records.

getmerklepath. When a light node wants to verify whether a record is valid, it needs to query a random full node the corresponding Merkle path.

B-DNS Name Servers to Recursive Resolvers: In this interface, B-DNS acts as an authoritative name server that

replies to DNS queries. On receiving a DNS query, a B-DNS name server directly fetches the queried DNS record from the stored blockchain and responds to it. In addition, if a recursive resolver wants to use the B-DNS name service, it simply sends a DNS query to a B-DNS name server directly.

B-DNS Name Servers to Users: B-DNS is designed to be compatible with the legacy DNS. Accordingly, clients can interact with B-DNS name servers directly by adding their IP addresses to its resolv.conf le. If the queried B-DNS name server is a full node, it could easily fetch the required DNS record from the blockchain. By contrast, if the queried B-DNS name server is a light node, which only stores the headers of the blockchain, the query will be redirected to a full node.

Fig. 4. The query operations in B-DNS. The light node will randomly choose a full node to obtain the queried record and merkle proof.

We nally discuss the difference of query operation between legacy DNS and B-DNS. In legacy DNS, name servers need to traverse the DNS tree from root servers to authoritative name servers. By contrast, as B-DNS name servers are structured in a peer-to-peer way, the query process is different as shown in Fig. 4. As a full node, the B-DNS name server can respond to the client directly by fetching the required DNS record from the blockchain. As to the light node, the B-DNS name server needs to check whether the queried DNS records are stored locally. If not, the light node needs to query a full node to acquire the requested DNS record.

5 Experiment

We implement a prototype of B-DNS in Golang according to our 4-layer architecture. We also establish a testbed for B-DNS on an i9-9900k server. We set up eight B-DNS nodes and each acts as a registry that stores a full copy of the blockchain. Each node is a 2 GB memory, 2 CPU, Ubuntu 18.04 virtual machine and the hypervisor is Vmware Workstation 15.0.4. We also set up a commercial DNS implementation PowerDNS Recursor 4.1.10 as a comparison. As to the DNS record dataset, we use publicly accessible DNS traces provided by the CAIDA to generate transactions [17]. Our blockchain consists of around 100,000 DNS entries. In our experimental blockchain, all operation transactions are signed using the ECDSA scheme and encapsulated in the form of registration records. We also craft a query dataset, which consists of 20,000 domain names, to test the lookup performance and resilience of B-DNS name servers.

We design two sets of experiments to evaluate the security and performance of B-DNS. In the security evaluation, we compare the security properties between legacy DNS and B-DNS from three dimensions: the probability of a successful attack, the attack cost, and the attack surface. In the performance evaluation, we test whether B-DNS provides acceptable performance.

5.1 Security Evaluation

In security evaluation, we conduct three experiments to compare the security between legacy DNS and B-DNS in terms of the probability of a successful attack, the attack cost, and the attack surface.

5.1.1 The Probability of A Successful Attack

In this experiment, we compare the probability of a successful attack against legacy DNS and B-DNS. In legacy DNS, a successful attack means that the attacker has generated a response packet with identical transaction ID and port number as the query packet. Its probability can be calculated as:

$$P_{success} = \frac{b \quad t}{(\quad)\quad s}$$

We list the meaning of these notations in Table 2. In B-DNS, since we adopt a different architecture, the attack methods against B-DNS is different. As all records are stored in the blockchain, some nodes maintain a cache to facilitate the query service. In this case, if an attacker wants to poison the cache, it needs to tamper with corresponding data stored in the blockchain. This requires the attacker to rewrite the blocks after the one that stores the queried record. In this case, we argue that a successful poisoning attack requires the attacker to catch up with the latest block. Therefore, when the attacker's stake is s_i and there are n nodes in B-DNS, the probability of a successful attack is:

$$P_{success} = \left(P_{\frac{s_i}{\sum_{m=1}^{i-1} s_m + \sum_{m=i+1}^{n} s_m}} \right)^z$$

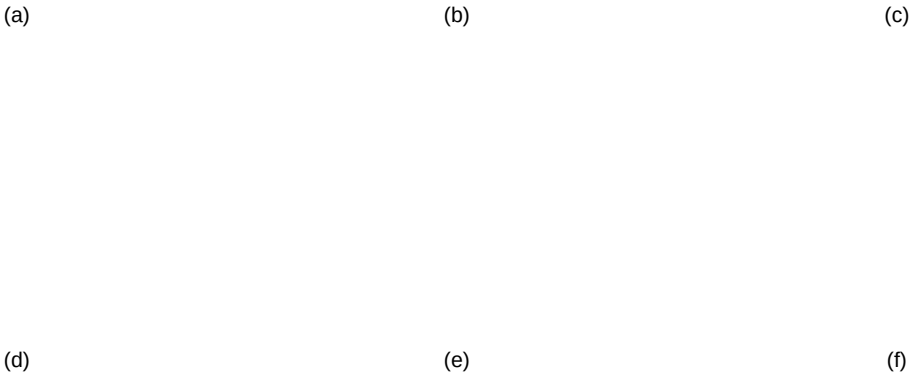
In Fig. 5, we draw the probability of cache poisoning attacks against current DNS and B-DNS, respectively. We can see that the probability of attacks against current DNS increases linearly with the number of sent packets and decreases with the number of authorities. We can also see that the highest probability of an attack against current DNS reaches 100%, which means an attacker will always succeed as long as it sends enough forged packets.

On the contrary, the probability of attacks against B-DNS increases with the number of controlled stake and decreases with the depth of target block. Usually, the attacker's controlled stake is a constant. Even a small increase in the stake is costly. In this case, the probability of a successful attack against B-DNS can hardly exceed 0.6% in current setting.

We also conduct real-case experiments to evaluate the probability. The results are shown in Fig. 6. We simulate an environment with 100 B-DNS nodes, each with 1% stake. Then we adjust the portion of attackers from 10% to 30%, which means the number of malicious nodes is from 10 to 30. We also assume that the honest nodes' chain is 6 blocks longer than attackers', which is the optimistic assumption for the attacker. Then we start the system to generate 100 blocks. If the attackers' chain catches up with honest nodes' chain, we record that attackers succeed. Otherwise, honest nodes win. We conduct the experiment 10000 times to

(a) Legacy DNS (b) B-DNS

Fig. 5. The calculated success probability of cache poisoning attacks against legacy DNS and B-DNS. In legacy DNS, as the number of attack packets increases, the probability of success increases and will eventually succeed. In B-DNS, only when the attacker has a large number of stakes and a small mining gap, there is a small probability of success.



(a) (b) (c)

(d) (e) (f)

Fig. 6. The experimental success probability of cache poisoning attacks against B-DNS. Although the probability of success increases with the attacker's stake, it is still negligible compared to the success probability of traditional DNS.

calculate the probability according to the times that attackers win. We get 100 possible success rates for each stake distribution scenario. As we can see, even with 30% stake, the success rate of cache poisoning attack against B-DNS is tremendously small.

5.1.2 Attack Cost

In this experiment, we compare the attack cost for attackers to launch attacks against legacy DNS and B-DNS. We think a fair way to compare the attack cost in different systems is when their success rates are equal. In this case, we consider the case that the probability of a successful attack is 1%. How much should an attacker pay to attack legacy DNS and B-DNS?

For legacy DNS with 3 authority name servers, an attacker needs to continuously send 126,835,750 packets to reach the probability of 1%, which requires 12216.9 MB traf c. For a network with 10 Mbps bandwidth, this attack lasts for 9780 seconds, less than three hours. The cost to use 10 Mbps for 2.7 hours is just several dollars.

For B-DNS, if an attacker wants to succeed with 1% probability, the stake it should own is shown in Table 3. If B-DNS possesses 1,000,000 domains and registering one domain requires 10 dollars, the attack cost ranges from 3,160,000 dollars to 4,320,000 dollars, which is far more than that of attacking legacy DNS.

TABLE 2
The notations used to represent the probability

Notation	Description
	The range of transaction IDs (universally 2^{16}, or 65536 values)
	The range of source ports (conceptually 2^{16})
	The number of reserved ports (usually 2^{10})
	The number of authority name servers. Many domain operate serveral authority servers with independent public IP address. A recursive server normally queries the closest one. Accordingly, is the product of all public facing addresses used by recursive resolver and authority servers.
b	the bandwidth between the attacker and the victim recursive resolver
t	The time that the attacker is able to send forged response packets
s	The size of a response packet
p	The probability that an honest node nds the next block
q	The probability that an attacker nds the next block
p_z	The probability that an attacker will never catch up from z blocks behind
q_z	The probability that an attacker will catch up from z blocks behind

TABLE 3
The amount of stake that attackers should hold to launch an attack with 1% success rate

Stake	Depth	Stake	Depth	Stake	Depth
31.6%	6	38.6%	10	41.8%	14
34.1%	7	39.6%	11	42.4%	15
35.9%	8	40.5%	12	42.8%	16
37.4%	9	41.2%	13	43.2%	17

5.1.3 Attack Surface

In this experiment, we compare the security of legacy DNS and B-DNS with respect to their attack surfaces. We de ne the attack surface as the system's actions that are externally visible to its users and the system's resources that each action accesses or modi es. We rst identify all resources of the system that are potential targets of attacks. For current DNS and B-DNS, the stored records and provided name service are vulnerable to different kinds of attacks. Then, we de ne the attack class as a set of attacks that employ similar attack methods. In our experiment, we categorize the common attacks against DNS as spoo ng, denial-of-service, hijacking, injection, and poisoning. Finally, we counte the number of instances of each attack classes for DNS and B-DNS, respectively. The results are concluded in Table. 4.

We can conclude that legacy DNS has more vulnerabilities in all types of attack classes. In spoo ng class, current DNS has exposed 48 vulnerabilities while B-DNS only has one. The closest attack class between current DNS and B-DNS is the denial-of-service attack, where current DNS has detected 24 vulnerabilities, double of B-DNS's vulnerabilities. As to the hijacking and poisoning classes, B-DNS does not have such kind of vulnerabilities. Even the only one injection vulnerability, the affected component of B-DNS is its debug log, which does not affect the core parts of B-DNS such as the name service. In a word, we can see the attack of B-DNS is much smaller than that of current DNS, which makes attackers more dif cult to attack B-DNS.

5.2 Performance Evaluation

In performance evaluation, we conduct four experiments to evaluate the performance of B-DNS.

5.2.1 Search Speed

In this experiment, we examine to what extent can index speed up searching in the blockchain and whether adding an index affects the overall performance. Speci cally, we test the search time in the blockchain with and without an index, respectively. We tested the search speed of our index with different record sets. The experiment results is illustrated in Table 5. As we can see, the search time is very limited in our constructed index. We also notice that the distribution of search speed without an index is approximately linear, which is because the search in a single chain needs to travel from the very beginning to the target block. Therefore, as the blockchain grows, the search time increases linearly. However, in B-DNS, the search time grows logarithmically.

5.2.2 Space Cost

In this experiment, we investigate the space cost of the B-DNS full node and light node, respectively. We use 31,535,998 DNS entries provided by CAIDA. We rst test the volume of a full node. The result is 887.41GB, which is quite an affordable result. Considering the price of disks nowadays, a registry can easily afford hundreds of drives. Then, we test the space cost of a light node, which only keeps the block header of a blockchain. The result shows that it only needs 4.12 GB. Apparently, it is feasible for most current DNS servers to operate a B-DNS name server.

5.2.3 Query Latency

In this experiment, we examine the query latency of B-DNS and compare it with PowerDNS. Both of them are set up in the lab without any cache warming up. B-DNS name server is equipped with a full blockchain and an empty cache. Correspondingly, the PowerDNS server is initialized with an empty cache as well. We continuously query two servers using pre-generated query packets and measure the corresponding latency. The results are illustrated in Fig. 7a. Explicitly, we nd that B-DNS could achieve approaching or even better lookup performance than PowerDNS. We remark that this is because the PowerDNS server obtains the queried IP address by recursive resolution while B-DNS can fetch records from the locally-stored blockchain directly. In recursive resolution, PowerDNS may suffer from the network congestion and packet loss. By contrast, B-DNS can provide more stable and ef cient name service as long as the record has been stored in the blockchain.

5.2.4 Flash-crowd Effect

In this experiment, we test the resilience of B-DNS when it faces the ash-crowd effect. Speci cally, the ash-crowd effect in DNS refers to sudden upheavals in the frequency of queried domain names. The server setting in this experiment is the same as the former one. We start by continuously sending DNS query packets to B-DNS server and

TABLE 4
The comparison of attack surface between legacy DNS and B-DNS

Attack Class	DNS			B-DNS		
	Number	Example	Description	Number	Example	Description
Spoo ng	48	CVE-2020-6412 CVE-2018-6175 CVE-2017-5106	insuf cient validation incorrect handling of URL characters insuf cient policy enforcement	1	CVE-2018-10831	incorrect veri er accepts spoof mining shares
Denial-of-Service	24	CVE-2020-6079 CVE-2018-8304 CVE-2018-19118	resource allocation vulnerability DNS response stack over ow	12	CVE-2018-17144 CVE-2016-10724	duplicate inputs memory exhaustion
Hijacking	2	CVE-2015-4020	insuf cient validation on SRV records	-	-	-
Injection	15	CVE-2019-5168 CVE-2011-5276	iochecked service vulnerability SQL injection vulnerability	1	CVE-2018-20586	injection to debug log
Poisoning	4	CVE-2018-5532 CVE-2015-4641	cached BIG F5 IP addresses directory traversal vulnerability	-	-	-

(a) Legacy DNS (b) B-DNS

Fig. 7. The performance of B-DNS. In gure (a), almost all B-DNS query latency is less than 20ms, while some PowerDNS's latency exceeds 100ms. In gure (b), we can see that B-DNS is not affected by the ash-crowd effect while PowerDNS takes around thirty seconds to recover.

TABLE 5
The search performance in B-DNS with the constructed index tree

Records Number	Route Times	Search Time (ms)
136780	17.9	63.7
1467032	19.7	74.9
1803284	20.3	80.3
23190410	21.9	89.6
160403846	23.4	91.3

PowerDNS server for three hours. Then, we ipped the popularity of queried domain names. Speci cally, the most popular domain name becomes the least popular and the second popular domain name becomes the second least popular, and so on. We measure the corresponding query latency and use the median in each minute to illustrate the trend as shown in Fig. 7b. We notice that the latency of PowerDNS is much higher than that of B-DNS at the very beginning and between 180-th to 210-th mins. This is because of the uncached query, which requires PowerDNS to launch recursive resolution. Speci cally, when facing the ash-crowd effect, the query latency of PowerDNS increases substantially while that of B-DNS remains stable.

6 Security Analysis

In this section, we discuss how B-DNS handles the DDoS attack and two other potential attacks: the Sybil attack and the index attack.

6.1 DDoS Attack

B-DNS can provide great resistance against the DDoS attack. The underlying blockchain distributes content to a large number of nodes. In addition, the peer-to-peer structure of B-DNS makes it hard to attack all the B-DNS name servers. Though some name servers may be compromised by the DDoS attack, it will not affect the overall name service.

6.2 Sybil Attack

Sybil Attack is a type of attack seen in peer-to-peer networks in which a node in the network operates multiple identities actively at the same time and undermines the authority/power in this system. In B-DNS, the Sybil attack can be launched by one registry mispresenting the number of domain names registered by it and so pretending to control a huge amount of stake. We argue that in B-DNS the cost to launch Sybil attack is tremendously high. A registry cannot arbitrarily claim the amount of its registered domains. The other nodes can easily check its real stake

by quickly traversing the whole blockchian. In this case, the only way to increase your stake is to register as many domain as possible, which is costly and tardy. Additionally, once a registry is caught to lie in its stake, B-DNS can eliminate its domain registration power in a soft-fork, which in turn warns the other registries to behave honestly.

6.3 Index Attack

The index attack is conducted by forging an incorrect attack and send it to honest peers. This stems from the fact that an adversary may try to create a fork when it generates a new block. However, the same as Bitcoin, B-DNS sets a security parameter 6, which represents that a block is stable when it is 6-block deep in the blockchain. B-DNS can ensure the correctness of the index by only converting the stable portion of the blockchain. In addition, B-DNS encapsulates the parameter IndexHash into the header. The parameter IndexHash is the hash value of the latest index. It allows each node to verify whether their index has been updated to the latest state of the blockchain.

7 Related Work

In this section, we introduce the related work on enhancing the security and performance of legacy DNS, and efforts that have been devoted to implementing blockchain-based domain name systems.

7.1 DNS Security and Performance

Several methods have been proposed to defend against the cache poisoning attack. Dagon et al. proposed to mix the upper and lower case spelling of the domain name in the query packet so that the adversary can hardly guess the right combination of upper and lower case letters [7]. Perdisci et al. utilized wildcard domain names (e.g., :example:com) to prevent attackers from guessing correct domain names [8]. In this way, a recursive resolver can prepend random strings to the queried domain name to distinguish valid response packets. Klein et al. proposed a user tracking technique to track user behaviors even if they use “privacy mode” browsing on multiple browsers [18].

DNSSEC creates a trust chain from the root server to the authoritative name servers, by which a recursive resolver can check the query route of the response packet by verifying the signatures [19]. However, though DNSSEC has been proposed for decades, its deployment rate is still meager nowadays [20]. Recent survey reveals that only 1% of .com, .net, and .org domains enabled DNSSEC [21], [22]. Several reasons account for this phenomenon: the sophisticated deployment procedure of DNSSEC, additional cost [21], and political reasons that some countries may be hostile to the country where the root servers are located [8] (e.g., Cuba may deny DNS packets originated from the USA). In addition, Shulman et al. conducted an Internet study of the cryptographic security of DNSSEC-signed domains [23]. They collected 2.1M DNSSEC keys and found that 35% are singed with RSA keys that share their moduli with some other domain, and 66% use keys that are too short. They concluded that this problem arises from the poor key generation practices. Additionally, researchers proposed T-DNS, which employs the transport-layer security (TLS) to establish secure DNS channels [24].

Facing the DDoS attack, Pappas et al. increased the TTL of NS records to ensure the availability of some crucial domains, especially when their father domains are under DDoS attacks [10]. Similarly, Ballani et al. proposed to build a separate “stale cache” in the recursive resolver to store the expired records [9]. In this way, if a recursive resolver does not receive the response from the authoritative server, it could use the stored records in the stale cache to complete the query process. Besides, some other efforts were devoted to evaluating the performance of the root servers under DDoS attacks [4], which demonstrated that massive attacks could overwhelm some root servers. In addition, Alieyan et al. [25] proposed a DNS-based schema to detect botnet by analyzing the query and response behaviours. Gao et al. proposed several detection and mitigation methods for the DDoS attack [26].

There are also some work on improving the performance of DNS. Park et al. proposed CoDNS [27], a lightweight, cooperative DNS lookup service that can be independently and incrementally deployed to augment existing name-servers. CoDNS is demonstrated to reduce the lookup latency by 28-82%. Gao et al. focused on DNS's update performance, which consumes dozens of seconds to complete, and proposed feasible improvement techniques [28]. Alouf et al. introduced an analytical model to study expiration-based caching systems based on renewal arguments and found that no distribution maximizes the hit probability anywhere in a network of caches [29]. Liu et al. proposed ContainerDNS [30], a scalable high-performance DNS for large-scale container cloud platforms, which maximizes DNS's performance and scalability by optimizing packet processing and using ef cient memory and cache management.

7.2 Blockchain-based Name Service

Namecoin was proposed to build a blockchain-based namespace [31]. It was forked from Bitcoin so that they share lots of similarities such as the block size, mining interval, and scripting system (with a few additions). Namecoin adopts the merged mining to ensure its data consistency. However, an empirical study on Namecoin shows that most registered domains are inactive and squatted [16], which is of great danger to a naming system [32]. Moreover, the Bitcoin-like system is shown to be vulnerable to mining attacks, which requires improvements [33]. Blockchain-DNS [34] provided a browser-side name resolution service for Namecoin. However, as Namecoin has many intrinsic problems, the usage of Blockchain-DNS is limited.

Ali et al. proposed Blockstack [35], a blockchain-based naming and storage system. Blockstack introduced the virtual chain so that it can introduce new functionalities without forking the underlying blockchain. Blockstack also has some advantages such as the cross-chain migration ability and fast bootstrapping. These properties make it easier to deploy the blockstack system. Yao et al. proposed to introduce cloud computing to mitigate the low computational load of the blockchain, exploring ways to of oad computationally intensive work to cloud services so that

low-computation devices in the DNS can participate in consensus [36]. Stergiou et al. aim to mitigate the security and privacy issues by presenting an IoT-based cloud system [37], which offers a secure infrastructure for establishing B-DNS.

8 Conclusion

In this paper, we propose B-DNS, a secure and efficient blockchain-based DNS. B-DNS is compatible with current DNS and can provide better defense against the cache poisoning attack and the DDoS attack. We propose a novel way to quantitatively compare the security of B-DNS and legacy DNS according to attack success rate, attack cost, and attack surface and our experiments demonstrate the good security of B-DNS. B-DNS can also provide efficient name service compared with legacy DNS. Our work actively explored the construction of the next-generation DNS infrastructure and provides a potential solution for building domain name systems for a wide area network, local area network, or intranet.

Acknowledgments

This work was partially sponsored by Hong Kong Research Grant Council (RGC) under grants GRF PolyU 15216220 and 152124/19E. This work was partially supported by Guangdong Basic and Applied Basic Research Foundation 2020A1515111070.

References

[1] A. Klein, H. Shulman, and M. Waidner, "Internet-Wide Study of DNS Cache Injections," in IEEE INFOCOM 2017-IEEE Conference on Computer Communications. IEEE, 2017, pp. 1–9.

[2] S. Son and V. Shmatikov, "The Hitchhiker's Guide to DNS cache poisoning," in International Conference on Security and Privacy in Communication Systems. Springer, 2010, pp. 466–483.

[3] L. Constantin, "DNS Poisoning Attack Against Major Brazilian ISP," 2009. [Online]. Available: https://news.softpedia.com/news/DNS-Poisoning-Attack-Against-Major-Brazilian-ISP-110226.shtml

[4] G. C. Moura, R. d. O. Schmidt, J. Heidemann, W. B. de Vries, M. Muller, L. Wei, and C. Hesselman, "Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event," in Proceedings of the 2016 Internet Measurement Conference, 2016, pp. 255–270.

[5] D. Lewis, "The DDoS Attack Against Dyn One Year Later," 2017. [Online]. Available: https://www.forbes.com/sites/davelewis/2017/10/23/the-ddos-attack-against-dyn-one-year-later

[6] R. van Rijswijk-Deij, A. Sperotto, and A. Pras, "DNSSEC and its Potential for DDoS attacks: A Comprehensive Measurement Study," in Proceedings of the 2014 Conference on Internet Measurement Conference, 2014, pp. 449–460.

[7] D. Dagon, M. Antonakakis, P. Vixie, T. Jinmei, and W. Lee, "Increased DNS Forgery Resistance through 0x20-bit Encoding: Security via Leet Queries," in Proceedings of the 15th ACM Conference on Computer and Communications Security, 2008, pp. 211–222.

[8] R. Perdisci, M. Antonakakis, X. Luo, and W. Lee, "WSEC DNS: Protecting Recursive DNS Resolvers from Poisoning Attacks," in 2009 IEEE/IFIP International Conference on Dependable Systems & Networks. IEEE, 2009, pp. 3–12.

[9] H. Ballani and P. Francis, "Mitigating DNS DoS Attacks," in Proceedings of the 15th ACM Conference on Computer and Communications Security, 2008, pp. 189–198.

[10] V. Pappas, D. Massey, and L. Zhang, "Enhancing DNS Resilience against Denial of Service Attacks," in 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. IEEE, 2007, pp. 450–459.

[11] L. Lao, Z. Li, S. Hou, B. Xiao, S. Guo, and Y. Yang, "A Survey of IoT Applications in Blockchain Systems: Architecture, Consensus, and Traffic Modeling," ACM Computing Surveys (CSUR), vol. 53, no. 1, pp. 1–32, 2020.

[12] D. Kaminsky, "Black Ops 2008: It's the End of the Cache as We Know It," Black Hat USA, Tech. Rep., 2008.

[13] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Tech. Rep., 2008.

[14] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies," in 2015 IEEE Symposium on Security and Privacy. IEEE, 2015, pp. 104–121.

[15] F. Tschorsch and B. Scheuermann, "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies," IEEE Communications Surveys & Tutorials, vol. 18, no. 3, pp. 2084–2123, 2016.

[16] H. A. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan, "An Empirical Study of Namecoin and Lessons for Decentralized Namespace Design," in WEIS. Citeseer, 2015, pp. 1–21.

[17] CAIDA, "DNS Names for IPv4 Routed Topology Dataset," 2017. [Online]. Available: http://www.caida.org/data/active/ipv4_dnsnames_dataset.xml

[18] A. Klein and B. Pinkas, "DNS Cache-Based User Tracking," in 26th Annual Network and Distributed System Security Symposium. The Internet Society, 2019, pp. 1–15.

[19] S. Goldberg, M. Naor, D. Papadopoulos, L. Reyzin, S. Vasant, and A. Ziv, "NSEC5: Provably Preventing DNSSEC Zone Enumeration," in 22nd Annual Network and Distributed System Security Symposium. The Internet Society, 2015, pp. 1–15.

[20] W. Lian, E. Rescorla, H. Shacham, and S. Savage, "Measuring the Practical Impact of DNSSEC Deployment," in 22nd USENIX Security Symposium. USENIX Association, 2013, pp. 573–588.

[21] T. Chung, R. van Rijswijk-Deij, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "Understanding the Role of Registrars in DNSSEC Deployment," in Proceedings of the 2017 Internet Measurement Conference, 2017, pp. 369–383.

[22] J. Bau and J. C. Mitchell, "A Security Evaluation of DNSSEC with NSEC3," in Proceedings of the Network and Distributed System Security Symposium, 2010, pp. 1–18.

[23] H. Shulman and M. Waidner, "One Key to Sign Them All Considered Vulnerable: Evaluation of DNSSEC in the Internet," in 14th USENIX Symposium on Networked Systems Design and Implementation. USENIX Association, 2017, pp. 131–144.

[24] L. Zhu, Z. Hu, J. Heidemann, D. Wessels, A. Mankin, and N. Somaiya, "Connection-Oriented DNS to Improve Privacy and Security," in 2015 IEEE Symposium on Security and Privacy. IEEE, 2015, pp. 171–186.

[25] K. Alieyan, A. Almomani, M. Anbar, M. Alauthman, R. Abdullah, and B. Gupta, "DNS Rule-based Schema to Botnet Detection," Enterprise Information Systems, vol. 15, pp. 1–20, 2019.

[26] S. Gao, Z. Peng, B. Xiao, A. Hu, Y. Song, and K. Ren, "Detection and Mitigation of DoS Attacks in Software Defined Networks," IEEE/ACM Transactions on Networking, vol. 28, no. 3, pp. 1419–1433, 2020.

[27] K. Park, V. S. Pai, L. L. Peterson, and Z. Wang, "CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups," in 13th USENIX Symposium on Operating Systems Design and Implementation, 2004, pp. 199–214.

[28] Z. Gao and A. Venkataramani, "Measuring Update Performance and Consistency Anomalies in Managed DNS Services," in 2019-IEEE Conference on Computer Communications (INFOCOM'19). IEEE, 2019, pp. 2206–2214.

[29] S. Alouf, N. C. Fofack, and N. Nedkov, "Performance Models for Hierarchy of Caches: Application to Modern DNS Caches," Performance Evaluation, vol. 97, pp. 57–82, 2016.

[30] H. Liu, S. Chen, Y. Bao, W. Yang, Y. Chen, W. Ding, and H. Shan, "A High Performance, Scalable DNS Service for Very Large Scale Container Cloud Platforms," in Proceedings of the 19th International Middleware Conference Industry, 2018, pp. 39–45.

[31] Namecoin, "Namecoin," 2014. [Online]. Available: https://namecoin.org

[32] S. Alrwais, K. Yuan, E. Alowaisheq, Z. Li, and X. Wang, "Understanding the Dark Side of Domain Parking," in 23rd USENIX Security Symposium. USENIX Association, 2014, pp. 207–222.

[33] S. Gao, Z. Li, Z. Peng, and B. Xiao, "Power Adjusting and Bribery Racing: Novel Mining Attacks in the Bitcoin System," in