Computers, Materials & Continua CMC, vol.63, no.1, pp.521-536, 2020

Self-Certificating Root: A Root Zone Security Enhancement Mechanism for DNS

Wenfeng Liu[1, *], Yu Zhang[1], Wenjia Zhang[1], Lu Liu[1], Hongli Zhang[1] and Binxing Fang[1]

Abstract: As a critical Internet infrastructure, domain name system (DNS) protects the authenticity and integrity of domain resource records with the introduction of security extensions (DNSSEC). DNSSEC builds a single-center and hierarchical resource authentication architecture, which brings management convenience but places the DNS at risk from a single point of failure. When the root key suffers a leak or misconfiguration, top level domain (TLD) authority cannot independently protect the authenticity of TLD data in the root zone. In this paper, we propose self-certificating root, a lightweight security enhancement mechanism of root zone compatible with DNS/DNSSEC protocol. By adding the TLD public key and signature of the glue records to the root zone, this mechanism enables the TLD authority to certify the self-submitted data in the root zone and protects the TLD authority from the risk of root key failure. This mechanism is implemented on an open-source software, namely, Berkeley Internet Name Domain (BIND), and evaluated in terms of performance, compatibility, and effectiveness. Evaluation results show that the proposed mechanism enables the resolver that only supports DNS/DNSSEC to authenticate the root zone TLD data effectively with minimal performance difference.

Keywords: Domain name system, root zone security, single point of failure.

1 Introduction

DNS namespace is a tree-like structure with a single root at the top. Correspondingly, a single-center hierarchical tree structure management and authentication architecture are formed in DNS. The Internet Assigned Numbers Authority (IANA) manages root data (root zone file) in DNS. The institution performs this function at present is the Internet Corporation for Assigned Names and Numbers (ICANN). As an affiliate of ICANN, PTI reviews the change request of TLD data submitted by the TLD authority and releases TLD data with the approved change in the form of root zone file after signing by using the private key of root authority. The public key for the initial certification of DNS is called trust anchor, which is configured as the public key of the key signing key (KSK) of the root by the recursive resolver by default and as the starting point for data source

[1] School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China.

[*] Corresponding Author: Wenfeng Liu. Email: 15b903031@hit.edu.cn.

Received: 17 July 2019; Accepted: 02 August 2019.

certification. Managing this trust anchor is now also the function of IANA.

Single-center root zone data management mode and domain name resource authentication structure bring about convenience while also cause a single point of failure (SPOF) risk. As root KSK in the root zone is selected as the only trust anchor, when the private key of the root KSK is leaked, the recursive resolver cannot identify the TLD data forged by the attacker who can then successfully perform DNS spoofing attack. During the attack, TLD authority cannot implement any effective preventive countermeasure because the authenticity of all TLD data is only validated by the root KSK of the root authority.

To help the TLD authority gain the capability to resist the single point of failure risk of the root key, we propose self-certificating root, a security enhancement mechanism of root zone resource certification compatible with DNSSEC. The mechanism can realize the two following goals:

- TLD authority can gain the capability to resist the single point of failure risk. When the private key of the trust anchor is leaked, the TLD authority can make the recursive resolver authenticate TLD data submitted to the root zone.
- Self-certificating root scheme is backward compatible with the DNS/DNSSEC protocol used by the current domain name system. Authoritative servers that apply self-certificating root scheme will not impede the recursive resolver, which only uses DNS and DNSSEC protocols, to query the domain name data and verify data authenticity.

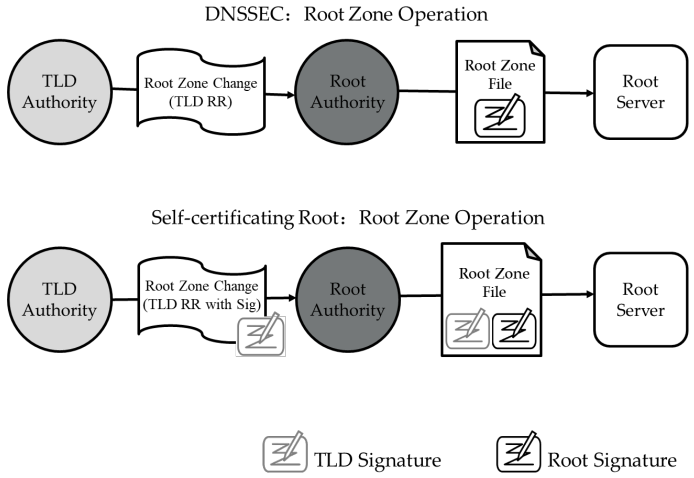

Figure 1: Current root zone operation (DNSSEC) *vs.* self-certificating root zone operation

As shown in Fig. 1, to realize the abovementioned two goals, under the premise that the data and public key in the current root zone are unchanged, the self-certification scheme adds the signature of the TLD glue records (including TLD NS RRs and corresponding A/AAAA RRs, defined in Hoffman et al. [Hoffman, Sullivan and Fujiwara (2019)]) to the root zone. By protecting the authenticity of the TLD authoritative server's IP address (included in TLD glue records), the recursive resolver is guaranteed to obtain TLD data from the correct data source. The main contributions of this study are as follows:

- The self-certificating root, a novel security enhancement mechanism of root zone

data certification, is proposed. This mechanism endows the root authority with the capability to resist the single point of failure risk. This scheme is compatible with existing DNS and DNSSEC protocols.

- The self-certificating root is implemented by modifying the open-source DNS server software-Berkeley Internet Name Domain (BIND). A domain name resolution tool-Domain Information Groper (DIG) which only supports the DNS/DNSSEC protocol, is used to prove that this scheme can protect the authenticity of TLD data in the root zone and it is backward compatible. We also performed a performance evaluation of the implemented solution, comparing the performance differences between the DNSSEC and the self-certified root applied to the root server.

The remainder of this paper is organized as follows: In Section 2, we introduce the background and related work. The background includes an introduction to the domain name system and its usage protocols, focusing on how the DNSSEC protocol protects data in DNS. The related work focuses on PKP, which is one of the core ideas of this work, and other schemes to improve the domain name system from the perspectives of performance, robustness, and security. Section 3 introduces the design of the self-certification scheme. Section 4 describes how to implement a self-certification scheme by modifying the BIND source. Section 5 evaluates the self-certification scheme of implementation from three aspects: performance, compatibility, and effectiveness. We conclude in Section 6.

2 Background and related work

DNS provides the mapping from domain name to value. According to different mapping types, the value can be an IP address, a hostname, or any text character. The security of DNS is the basis of multiple service security, and it also affects the security of cyberspace on the mobile side [Cui, Zhang, Cai et al. (2018)]. The DNS protocol used by DNS does not contain a cryptographic mechanism. Thus, the authenticity and integrity of resource records (RRs) in response message cannot be protected. The subsequently proposed DNSSEC protocol [Arends, Austein, Larson et al. (2005)] provides a solution for these defects.

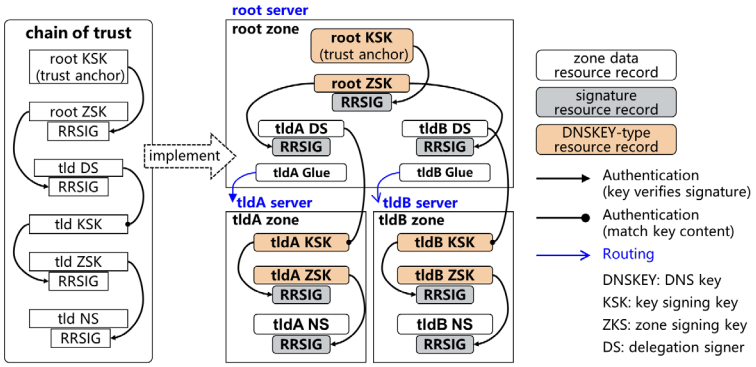

Figure 2: DNSSEC authentication process: build a "chain of trust" from root ksk (trust anchor) down the DNSSEC hierarchy to any zone's resource record

Fig. 2 displays how the DNSSEC protocol protects the authenticity of resource records through digital signatures of resource records. To protect the authenticity and integrity of the domain name RR in the response message, DNSSEC allows authorities of different levels (root, top-level, second-level, etc.) creates a public/private key pair named by their zone and then signs the data in the zone by using the private key. In the zone file, the public key is saved by the resource records of DNSKEY type, which is divided into KSK and zone signing key (ZSK). The signature is saved by the RR of RRSIG (resource record set signature) type. When the recursive resolver acquires the resource records of a domain name, the resolver must construct a chain of trust that follows the DNS hierarchy from a trusted root zone key down to the key of the zone in question.

The construction process of the chain of trust is represented by the black arrow in Fig. 2. The right half of Fig. 2 shows how DNSSEC builds a chain of trust from trust anchors to arbitrary resource records across zones. To certificate the authenticity of the TLD A NS resource record in the zone of TLD A, the recursive resolver first looks for a trust anchor from the root zone consistent with the local configuration, which is usually configured implicitly as the root KSK. Then verify the authenticity of root ZSK by verifying its signature resource records using the public key of root KSK and verifying the authenticity of TLD DS by using the public key of root ZSK. TLD DS is the hash of TLD KSK, so the authenticity of TLD A KSK is also verified, which means that the authenticity of all resource records in the TLD A zone can be verified.

In DNSSEC protocol, as TLD data (TLD DS in Fig. 2) in the root zone will be only signed by the key in the root zone, the TLD authority cannot protect the authenticity of TLD data in the root zone when the trust anchor becomes invalid. The trust anchor may experience misconfiguration [Dai, Shulman, Waidner (2016)] [Pappas, Fältström, Massey et al. (2004)], and this problem has been extensively investigated. The private key of the trust anchor may also be leaked, and the attacker can implement a spoofing attack on the recursive resolver by manipulating the root server once the private key of the trust anchor is acquired; multiple manipulation cases have been found in related measurement works [Kreibich, Weaver, Nechaev et al. (2010)] [Weaver, Kreibich and Paxson (2011)].

Public key pinning (PKP) Evans et al. [Evans, Palmer and Sleevi (2015)], a scheme designed for HTTPS protocol, aims to greatly reduce the risks of man-in-the-middle attack and other fake identity verification problems. PKP scheme allows the browser to transfer trust from the public key of the root CA (certificate authority) to other trusted public keys, such as local public key white-list or public keys acquired using the trust-on-first-use (TOFU) mechanism. The idea of the PKP scheme is applied to DNS in this study to establish a candidate trust anchor from the TLD authority for the local recursive resolver. So that the TLD authority protects the authenticity of TLD data in the root zone through the candidate trust anchor when the root trust anchor cannot be used.

By adding TLD keys in the root zone, TLD data are protected from the risk brought by a single point of failure of the root key. Other schemes have also tried to optimize DNS from angles of performance, robustness, and safety, similar to the present study. Cox et al. [Cox, Muthitacharoen and Morris (2002)] firstly proposed applying P2P idea to the DNS resolution service to implement a distributed domain name resolution. Freedman et al. [Freedman, Freudenthal and Mazieres (2004)] designed a P2P-based DNS resolution

service to support P2P-based CDN. Ramasubramanian et al. [Ramasubramanian and Sirer (2004)] suggested a P2P-based distributed name resolution system, namely, CoDoNS, to save domain name data by using a distributed hash table, thereby improving the robustness of domain name data storage and resolution rate. Park et al. [Park, Pai, Peterson et al. (2004)] proposed a DNS query plan, namely, CoDNS, under the collaboration of recursive resolver to improve the resolution performance and reliability of the resolution result. Cachin et al. [Cachin and Samar (2004)] presented a type of distributed DNS authoritative server architecture, used a redundant state machine to cope with Byzantine fault, and utilized threshold cryptography to manage the leakage of DNSSEC private key.

3 Design of self-certificating root

3.1 Overview

To endow the TLD authority with the capability to resist a single point of failure risk, the self-certificating root scheme must enable the TLD authority to verify the authenticity of TLD data saved in the root zone independently. Here, "independently" means that the recursive resolver can verify the TLD data in the root zone without dependence on the root key.

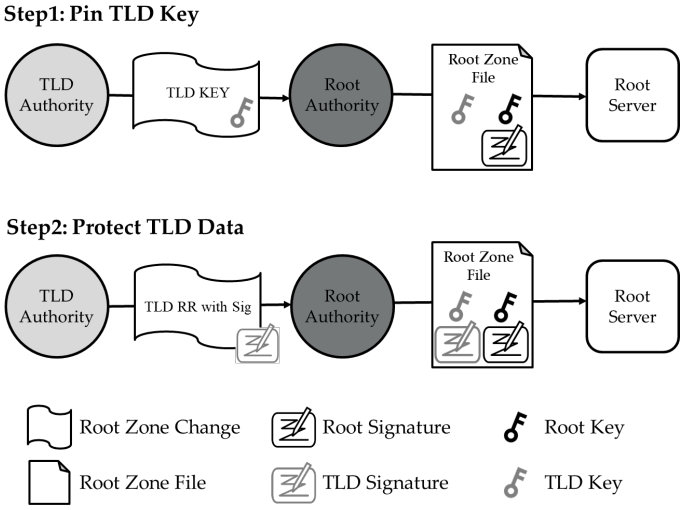

Figure 3: High-level overview of data and key publishing procedure in self-certificating root

Fig. 3 displays the operation flow of the TLD authority issuing keys and data to the root zone in the self-certificating root scheme, which is divided into two stages: 1) In Stage 1, the TLD authority anchors its public key in the root zone. TLD authority submits the TLD public key to the root authority, which writes the TLD key into the root zone file after judging the TLD public key as credible and signing it by using the root private key; 2) In Stage 2, TLD authority uses its own managed keys to protect the data about to submit to the root zone. The TLD authority generates the RRSIG record corresponding to the resource record by signing the TLD resource record with the private key corresponding to the TLD public key in Step 1 and submits the TLD resource record together with the corresponding RRSIG record to the root authority. After verifying the signature, the root authority writes the content submitted by the TLD authority into the root zone file.

In the two stages of the self-certificating root scheme, Stage 1 is the bootstrap stage in which the TLD authority needs the certification of the root authority such that it can anchor the TLD public key into the root zone. After the root authority has completed Step 1 once, the TLD authority can independently realize the certification of TLD data and the update of TLD key in the root zone by executing Step 2. The design idea "whoever submits the data will be responsible for protecting such data" used by the self-certificating root scheme enables the TLD authority to verify the authenticity of TLD data in the root zone without dependence on the root authority to resist SPOF risk of the root key.

3.2 Enable TLD authority to protect authenticity of TLD RRs in root zone

The key to protecting TLD data in the root zone is that the key managed by the TLD authority is used to sign the TLD data in the root zone and enable the recursive resolver for obtaining the TLD public key, TLD data, and signature of TLD data from the root zone. To meet these requirements, the three following elements are added to the root zone in the self-certificating root scheme, as shown in Fig. 4.

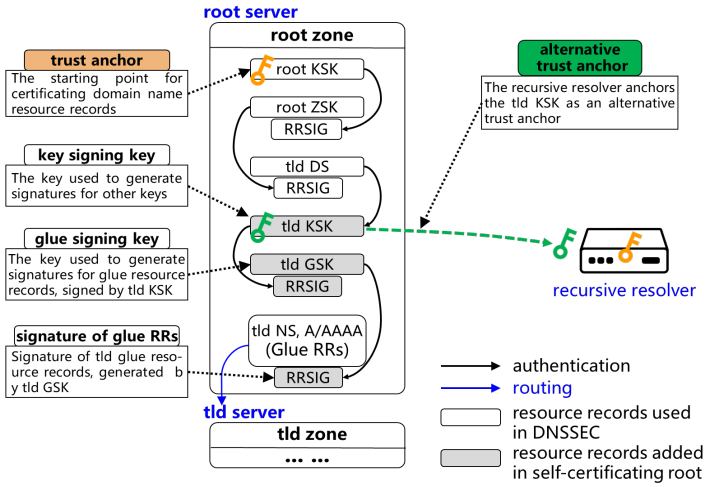

Figure 4: Presentation of the root zone and the public key chain in the self-certificating root design

TLD key singing key (KSK) resource record is the same as the TLD KSK stored in the TLD zone file. It is a resource record of the DNSKEY-257 (defined in Section 2.2 of Austein et al. [Austein, Larson, Massey et al. (2005)]) type currently used in the DNSSEC protocol. The self-certification scheme stores a copy of the TLD KSK resource record stored in the TLD zone to the root zone so that the recursive resolver obtains the public key for verifying the TLD data from the root server and uses it as an alternative trust anchor. In DNSSEC, only the public key of the root authority is saved in the root zone. All data (e.g., data submitted from TLD: TLD DS resource record) in the root zone are only signed by the private key of the root authority. Therefore, the recursive resolver can only select the public key of the root authority (root KSK) as the starting point of the certificating process. Consequently, when the root KSK is leaked or cracked, all data in the root zone will be unsafe. TLD public key (TLD KSK) is added into the root zone in

the self-certificating root scheme so that a private key, which is managed by the TLD itself, is added to the root zone. The recursive resolver can save TLD KSK locally as an alternative trust anchor in the way of TOFU. When the trust anchor undergoes a fault, the recursive resolver rapidly switches the trust anchor to the alternative trust anchor to protect the safety of TLD data in the root zone.

TLD glue singing key (GSK) in the root zone is a key used to sign TLD glue record. Similar to the role of ZSK only for signing zone data, which is defined in DNSSEC. TLD GSK is also only used to sign the data in the zone. The difference between GSK and ZSK is that GSK only signs glue records, while ZSK is used to sign other data in a zone other than glue records and DNSKEY records. The DNSSEC protocol does not distinguish between different types of DNSKEYs. The motivation for setting different types of DNSKEYs is simply to distinguish them from the operation and maintenance aspects. The resolver does not distinguish between the types of DNSKEYs. To ensure optimal compatibility, GSK uses the DNSKEY-256 type of resource record storage, which is defined in the DNSSEC protocol. This design is also consistent with the DNSSEC operation and maintenance practices (defined in Section 3.1 of Kolkman et al. [Kolkman, Mekking and Gieben (2012)]).

KSK and GSK separated design is used in the self-certificating root scheme. TLD KSK is utilized to certify TLD GSK that is used to protect the safety of TLD glue records in the root zone. This separated design aims to guarantee the safety of TLD KSK as an alternative trust anchor as much as possible. TLD KSK must only sign in the key rollover process such that the number of use times of TLD KSK is reduced, which reduces the possibility that KSK may be violently cracked. This idea is identical to the idea that of separating KSK and ZSK in DNSSEC protocol (Section 3.1 in Kolkman et al. [Kolkman, Mekking and Gieben (2012)]).

Signature of glue RR is generated by GSK which is independently managed by the TLD authority and stored using the RRSIG-type resource record defined by DNSSEC. After the glue record is signed, the TLD authority can protect the authenticity of the glue record. As A or AAAA records in the glue record store the IP address of the TLD authoritative server, the TLD authority can protect the authenticity of this IP address via GSK to ensure that the recursive resolver can access the authentic TLD authoritative server and acquire TLD data from the correct source.

Fig. 4 displays the certification chain of resource records in the root zone in the self-certificating root scheme. The solid black arrow represents certification of RRs, which is realized by generating the digital signature for the RR to be certified. A certification chain from the trust anchor (root KSK) to the TLD private key (TLD KSK) exists in the self-certificating root scheme. Thus, the authenticity of the public key of the TLD authority can be certified by the root authority. Meanwhile, a certification chain from the TLD authoritative private key (TLD KSK) to the TLD data in the root zone (TLD glue RR) is created in the self-certificating root scheme, thereby protecting the authenticity of TLD data in the root zone through the private key managed by the TLD authority.

3.3 Key rollover

For various reasons, keys in DNSSEC must be changed occasionally. The longer a key

has been in use, the greater the probability that it has been compromised through carelessness, accident, espionage, or cryptanalysis. On the basis of DNSSEC operating practice (Section 3.3 in Kolkman et al. [Kolkman, Mekking and Gieben (2012)]), the validity period of a reasonable private key is 13 months, and the administrator of the zone should conduct a key rollover every 12 months.

Key rollover capability is an important factor guaranteeing the safety and robustness of the self-certificating root scheme. In the root zone of self-certificating root scheme, keys of the root and TLD authorities in the root zone must be updated regularly. The compatibility of root KSK is exactly the same as DNSSEC, so the root KSK key rollover scheme is fully compliant with root KSK update plan [Root Zone KSK Rollover Project (2019)] published by ICANN.

Table 1: Stages of TLD KSK double-signature key rollover

DNS RR Type	Initial Stage	New DNSKEY Stage	Removal DNSKEY Stage
SOA	SOA_1 RRSIG(root ZSK_1)	SOA_2 RRSIG(root ZSK_1)	SOA_3 RRSIG(root ZSK_1)
tld DS	tld DS_KSK_1 RRSIG(root ZSK_1)	tld DS_KSK_1 RRSIG(root ZSK_1)	—
	—	tld DS_KSK_2 RRSIG(root ZSK_1)	tld DS_KSK_2 RRSIG(root ZSK_1)
tld KSK	tld KSK_1 — / RRSIG(tld KSK_0)	tld KSK_1 — / RRSIG(tld KSK_0)	—
	—	tld KSK_2 RRSIG(tld KSK_1)	tld KSK_2 RRSIG(tld KSK_1)
tld GSK	tld GKS_1 RRSIG(tld KSK_1)	tld GKS_1 RRSIG(tld KSK_2)	tld GKS_1 RRSIG(tld KSK_2)

The self-certification root scheme needs to additionally design the key updating process for the TLD KSK in the root zone. The key rollover scheme adopts the Double-Signature key rollover method, that is, the old key and the new key are simultaneously released in the root zone, wait until the old key expires in the cache of a recursive resolver and then remove the olds from the root zone. Tab. 1 shows the process of updating the TLD KSK using this method. The process is divided into three stages, namely the initial stage, the new DNSKEY stage, and the removal DNSKEY stage.

The initial Stage indicates that the key is in normal use, and this is the state before the key rollover process. When the authority of the TLD KSK key starts the key rollover, it immediately enters the new DNSKEY Stage in Tab. 1, which contains the following four steps (Fig. 5):

1. The TLD authority generates a new TLD key signing key (tld KSK_2) and uses the old TLD KSK (tld KSK_1) to generate the signing record of the new TLD KSK (tld KSK_2) signature (RRSIG).
2. The TLD authority uses the new TLD KSK (tld KSK_2) from Step 1 to generate the signature record of TLD GSK (RRSIG).

3. The TLD authority uses the new TLD KSK (tld KSK_2) to generate the corresponding TLD delegation of the signing record (tld DS_KSK_2).
4. The TLD authority submits the records generated in Steps 1-3 to the root authority, which then signs the TLD DS record (tld_DS_KSK_2) generated in step 3 by using root ZSK (root ZSK_1) to generate the signing record (RRSIG).

After the root authority adds the new resource record in Steps 1-4 to the root zone file, it indicates that the new DNSKEY stage begins.

The root authority waits for a period until it is speculated that all the old KSK records, the old DS records and related RRSIG records disabled in the cache of the recursive resolver (usually the waiting period is the maximum value of the TTL of all replaced resource records), then root authority can start the removal DNSKEY stage. In the removal DNSKEY stage, the root authority immediately removes the old resource record of the root zone file replaced by Steps 2-4 and completes the stage. At this point, the self-certification scheme completes the key rollover process for the TLD KSK.

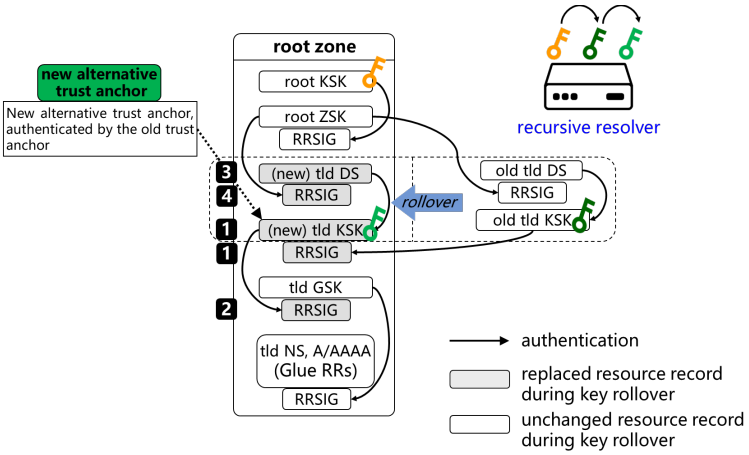

Figure 5: Changes in resource records during key rollover process of the self-certificating root

After the recursive resolver finds that the alternative trust anchor (TLD KSK) is updated in the root zone within the new DNSKEY stage (Stage 2). Only the old alternative trust anchor saved in local storage is needed to verify the digital signature of the new alternative trust anchor (upper right corner in Fig. 5) and save the authenticated new alternative trust anchor in the local storage, thereby realizing trust anchor rollover of the recursive resolver.

4 Implementation

The self-certificating root scheme designed in Section 3 was implemented on the basis of the BIND software. BIND, which was presented in the early stage of the 1980s, is a DNS software with the most extensive used [DNS server survey (2015)] on the Internet. BIND has implemented DNS and DNSSEC protocols. Newly added elements and mechanisms in the self-certificating root scheme were implemented in BIND by modifying the source code to enable the resolver to obtain TLD data in the root zone and verify its authenticity.

4.1 Data model in BIND

Fig. 6 shows the domain name data storage model maintained by BIND. BIND utilizes the view→zone→name→type four-tier structure to save domain name data. For each view, BIND maintains a red-black tree (structure *dns_rbt_t* in tier 2), which saves domain name RRs under this view, and each red-black tree saves data of one zone. In one BIND operation instance, BIND simultaneously provides authoritative responses for multiple zones by maintaining multiple red-black trees. Each red-black tree saves pointer directing at multiple nodes (structure *dns_rbtnode_t* in tier 3), and a node saves RRs with the same name in this zone. Each node saves a pointer directing at multiple headers (structure *rdataheader_t* in tier 4), and each header saves RRs with the same type in the same name.

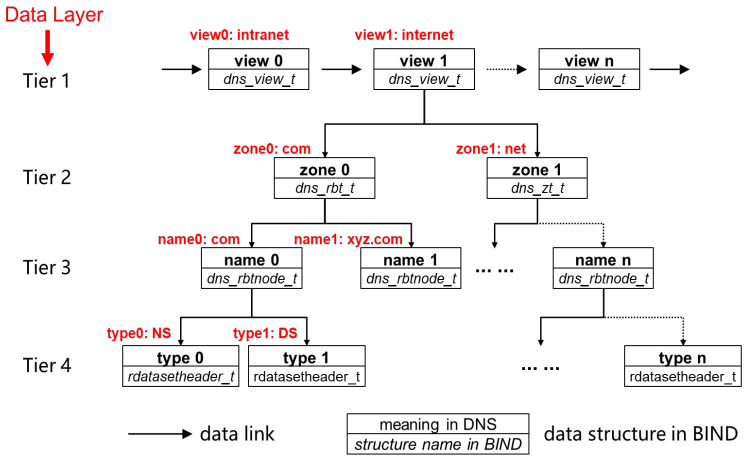

Figure 6: Storage model of domain resource record data in memory when BIND authoritative server is running

4.2 Enabling BIND to respond with the TLD data of the root zone

After receiving the DNS query request from the resolver, the BIND authoritative server software retrieves the zone file data loaded to the memory. BIND filters the retrieved results in accordance with matching rules and uses filtered data to construct the DNS response message to respond to the resolver.

In the matching process, BIND checks whether different nodes meet the matching rules. A critical rule is that the authoritative server in each zone can only respond to an RR in this zone and a glue record in its sub-zone, wherein the signing records of private key record (TLD KSK/GSK) and glue record in the sub-zone do not meet the matching rules. To implement the design of self-certificating root, the matching mechanism should be remodified (logic in algorithm 1) such that the authoritative server can respond to newly added elements (TLD KSK, TLD GSK, and Glue RRSIG). The recursive resolver can acquire all data needed to construct the trust chain and their signatures from the root server to verify the authenticity of TLD glue RR in the root zone.

Algorithm 1 Modify the filtering rules so that the BIND could respond with the resource records added in self-certification root

Input：

Retrived zone data tree, *zone_rbt*;

Output：

Effective resource records, *node_list*;

```
1:  global node_list = [∅]
2:  zonename = TreeName (zone_rbt)
3:  valid_nameserver = [∅]
4:  if zonename != root
5:     return node_list
6:  currentnode = zone_rbt→root
7:  while(currentnode != null){
8:     nodename = NodeName(current)
9:     if Labelcount(zonename) != 1
10:       continue
11:    currentheader = currentnode→rdatasetheader
12:    while(currentheader != null){
13:       if (Type(currentheader) == DNSKEY)
14:          ListAppend(node_list, currentnode)
15:       if (Type(currentheader) == NS)
16:          ListAppend(node_list, currentnode)
17:          ListAppend(valid_nameserver, RDATA(currentheader))
18:       if (Type(currentheader) == A or AAAA)
19:          if NodeName(currentnode) in valid_nameserver
20:             ListAppend(node_list, currentnode)
21:       if (Type(currentheader) == RRSIG)
22:          if SigType(currentheader) == NS or A or DNSKEY
23:             ListAppend(node_list, currentnode)
24:    currentheader = Next(currentheader)}
25: currentnode = Next(currentnode)}
26: return node_list
```

5 Evaluation

The self-certificating root scheme is implemented by modifying the open-source DNS software BIND (version 9.12.3). The implemented scheme is evaluated from three aspects-performance, compatibility, and effectiveness: 1) Performance evaluation focuses on the performance impact of modifying source codes to BIND. 2) The effectiveness evaluation focuses on whether the self-certification scheme has the ability to enable the

resolver to verify the authenticity of the root zone TLD data, which corresponds to the first design goal mentioned in Section 1. 3) Compatibility evaluation focuses on whether the self-certificating root scheme is backward compatible with the DNS/DNSSEC protocols used by the existing domain name system, that is, the second design goal mentioned in Section 1.

The system testing environment is configured with two physical hosts: an authoritative server machine and a resolver machine. The physical machines used are allocated with 12 vCPUs and 32 GB of memory. All machines are connected to a local network with 1000 Mbps. The physical machine used as an authoritative server is deployed with Docker containers (Nos. A and B) of two BIND operation cases. A runs BIND software without any modification, and B runs BIND software which modifies BIND source code and implements the design of self-certificating root. The physical machine used as the resolver operates the DIG software, which is an open-source domain name resolution tool that can send query message that supports the DNSSEC protocol and verify whether the digital signature of RR is valid.

5.1 Evaluation of performance

Based on the BIND software with unchanged source code, the resolution performances after the self-certificating root scheme was implemented by modifying the source code were comparatively tested. Resolver tool-DIG was used to send the query request of resolving the same domain name and type simultaneously to docker containers A (DNSSEC) and B (self-certificating root) (command: *dig @<dns-server addr>-p <port>name type+sigchase+trusted-key=<trust-anchor>*). Time delays needed to resolve the domain name successfully from two containers were recorded and set as one group of test data. Fig. 7 shows 5000 groups of tests of the DNSSEC and self-certificating root schemes. Relative to unmodified BIND, the performance difference brought by the self-certificating root scheme was minimal. In the 5000 groups of comparative tests, the average resolution time delay of DNSSEC is 11.34 ms, that of the self-certificating root scheme is 11.09 ms. The difference is no more than 2.2%. From the perspective of delay composition (Fig. 8), implementing a self-certification root will bring a 0.3 ms delay (from 0.80 ms to 1.18 ms) to the authoritative server but will reduce the resolver authentication delay (from 10.63 ms to 10.13 ms). In total, the self-certificating root scheme brought small performance impact to the root server.

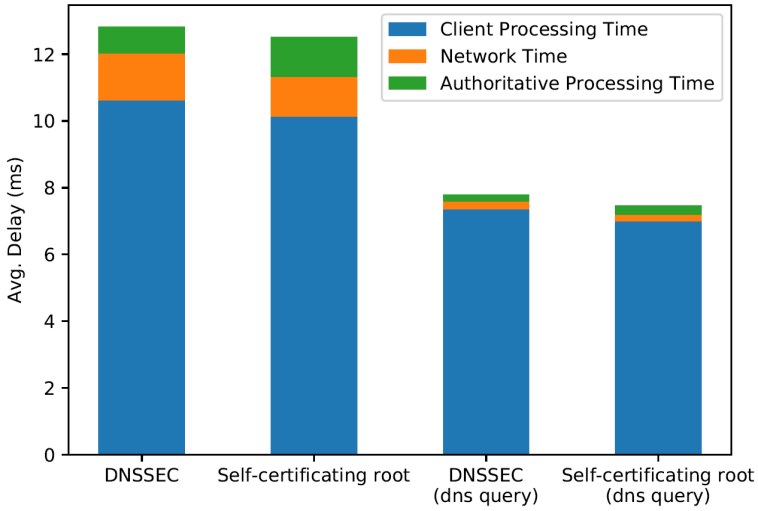

Figure 8: Three components of the domain name resolution delay of the DNSSEC and self-certificating scheme: server response delay, network communication delay, parser verification delay

5.2 Evaluation of compatibility

In Section 1, the design goals of self-certificating root scheme with compatibility were proposed, namely, the self-certificating root scheme should be backward compatible with the DNS and DNSSEC protocols used by the existing DNS. The use of the self-certificating root scheme must not impede the recursive resolver that only supports DNS and DNSSEC protocols to query the domain name data and verify data authenticity.

The compatibility of the self-certificating root scheme was evaluated and proven from two aspects: 1) The changed RR in the root zone file. As shown in Tab. 2, the first column is the resource record in the root zone file of the DNSSEC protocol, and the second column is the resource record in the root zone file of the self-certification root scheme. By comparing the two columns, it can be found that the self-certificating root scheme neither deleted any RRs used by DNSSEC protocol nor changed the use of existing RRs in the current root zone. Instead, only the necessary TLD RRs was added to protect the authenticity of TLD glue record. The resolver that only supports the DNSSEC protocol only needs to ignore the newly added TLD DNSKEY RR in the root zone and verify in accordance with the original logic. 2) The authentication results of DIG resolver. The DIG resolver that is unmodified and only supports DNS/DNSSEC protocols were utilized to acquire and verify the digital signatures of TLD data successfully from the authoritative server. This process proved that the self-certificating root scheme is backward compatible with resolvers that only support DNS and DNSSEC.

Table 2: Compare the resource records in root zone between the DNSSEC scheme and the self-certification root scheme (NSEC resource record has been omitted)

Root Zone of DNSSEC	Root Zone of Self-certificating root
. IN TTL SOA RDATA(...)	. IN TTL SOA RDATA(...)
. IN TTL RRSIG RDATA(SOA ...)	. IN TTL RRSIG RDATA(SOA ...)
. IN TTL NS RDATA(root-server)	. IN TTL NS RDATA(root-server)
. IN TTL RRSIG RDATA(NS ...)	. IN TTL RRSIG RDATA(NS ...)
. IN TTL DNSKEY RDATA(256 ...) //root ZSK	. IN TTL DNSKEY RDATA(256 ...) //root ZSK
. IN TTL DNSKEY RDATA(257 ...) //root KSK	. IN TTL DNSKEY RDATA(257 ...) //root KSK
. IN TTL RRSIG RDATA(DNSKEY ... zskID ...)	. IN TTL RRSIG RDATA(DNSKEY ... zskID ...)
. IN TTL RRSIG RDATA(DNSKEY ... kskID ...)	. IN TTL RRSIG RDATA(DNSKEY ... kskID ...)
tld IN TTL DS RDATA(...)	tld IN TTL DS RDATA(...)
tld IN TTL RRSIG RDATA(DS ...)	tld IN TTL RRSIG RDATA(DS ...)
tld IN TTL NS RDATA(tld-server)	tld IN TTL NS RDATA(tld-server)
——	tld IN TTL RRSIG RDATA(NS ...)
tld-server IN TTL A RDATA(ip_address)	tld-server IN TTL A RDATA(ip_address)
——	tld IN TTL RRSIG RDATA(A ...)
——	tld IN DNSKEY RDATA(256 ...) //used as tld GSK
——	tld IN DNSKEY RDATA(257 ...) //used as tld KSK
——	tld IN TTL RRSIG RDATA(DNSKEY ... gskID ...)
——	tld IN TTL RRSIG RDATA(DNSKEY ... kskID ...)
root-server IN TTL A RDATA(ip_address)	root-server IN TTL A RDATA(ip_address)

5.3 Evaluation of effectiveness

In Section 1, the goal that the self-certificating root scheme should enable the TLD authority to gain the capability to resist the single point of failure risk was proposed. When the root key was leaked, the TLD authority could independently and effectively protect the authenticity of TLD data in the root zone.

As shown in Fig. 4, regardless of whether the root trust anchor (root KSK, orange) or alternative trust anchor of the TLD authority (TLD KSK, green) is used, a certification chain from the trust anchor to the signature of TLD glue record exists in the self-certificating root scheme. Thus, authenticity can be proven by verifying the signature of the TLD glue record in the root zone to ensure that the self-certificating root scheme can effectively protect TLD data in the root zone. Meanwhile, as the alternative trust anchor is independently managed by the TLD authority, the TLD authority can realize the goal of independently protecting the authenticity of glue records in the root zone only if the resolver is switched to the alternative trust anchor.

By measuring all TLDs in the domain name system, the authenticity of the glue records of some TLDs is still not protected. Among them, there are 20.58% of the TLDs, the A-type records in the glue records do not have corresponding digital signatures. There are 15.50% of the TLDs, the AAAA-type records in the glue records do not have corresponding digital signatures. In all TLDs, only 22.29% of them, their glue records are

all protected with signature (including both NS, A, AAAA type resource records). One of the reasons for the phenomenon is that DNSSEC is not deployed in the second-level domain where these domain names are located. Although the top-level domain supports DNSSEC, the second-level domain authority that store the A-type and AAAA-type does not support DNSSEC, which still makes the authenticity of glue records cannot be protected. These TLD authorities who want to protect the authenticity can reach this goal by applying self-certificating root schemes.

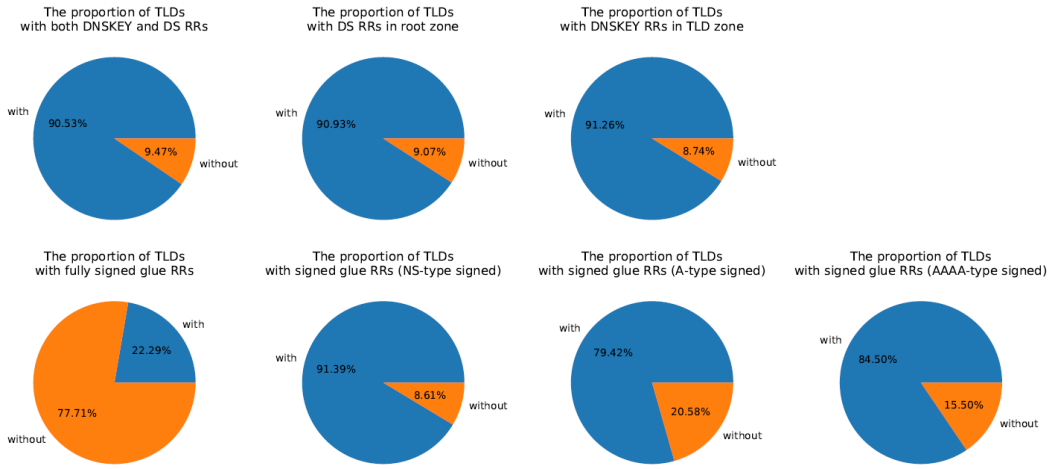

Figure 9: Percentage of the authenticity of glue records protected in all TLD of the current domain name system

6 Conclusion

A lightweight security enhancement mechanism, namely, self-certificating root, which protects TLD data in the root zone, is proposed in this study for the single point of failure risk brought by the central governance structure in the DNS. The self-certificating root scheme enables the TLD authority to resist the single point of failure risk, and it is compatible with DNS and DNSSEC protocols used by the existing DNS. This mechanism is implemented by modifying the open-source software BIND, and its performance, compatibility, and effectiveness are evaluated. The evaluation results indicate that the self-certificating root scheme endows the resolver that only supports DNS and DNSSEC protocols with the capability to verify TLD data in the root zone only with minimal performance influence.

Acknowledgement: This work is partially supported by the National Key Research and Development Program (2018YFB1800702).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Arends, R.; Austein, R.; Larson, M.; Massey, D.; Rose, S. (2005): Protocol modifications for the DNS security extensions, RFC 4035. *Internet Society*.

Austein, R.; Larson, M.; Massey, D.; Rose, S. (2005): Resource records for the DNS security extensions. RFC 4034. *Internet Engineering Task Force.*

Cachin, C.; Samar, A. (2004): Secure distributed DNS. *International Conference on Dependable Systems and Networks*, pp. 423-432.

Cox, R.; Muthitacharoen, A.; Morris, R. T. (2002): Serving DNS using a peer-to-peer lookup service. *International Workshop on Peer-To-Peer Systems*, pp. 155-165.

Cui, J.; Zhang, Y.; Cai, Z.; Liu, A.; Li, Y. (2018): Securing display path for security-sensitive applications on mobile devices. *Computers, Materials and Continua*, vol. 55, no. 1, pp. 17.

Dai, T.; Shulman, H.; Waidner, M. (2016): DNSSEC misconfigurations in popular domains. *International Conference on Cryptology and Network Security*, pp. 651-660.

DNS Server Survey (2015): http://mydns.bboy.net/survey.

Evans, C.; Palmer, C.; Sleevi, R. (2015): Public key pinning extension for HTTP. RFC 7469. *Internet Engineering Task Force.*

Freedman, M. J.; Freudenthal, E.; Mazieres, D. (2004): Democratizing content publication with coral. *USENIX Symposium on Networked Systems Design and Implementation*, vol. 4, pp. 18-18.

Hoffman, P.; Sullivan, A.; Fujiwara, K. (2019): DNS terminology. RFC 8499. *Internet Engineering Task Force.*

Kolkman, O.; Mekking, W.; Gieben, R. (2012): DNSSEC operational practices version 2. RFC 6781. *Internet Engineering Task Force.*

Kreibich, C.; Weaver, N.; Nechaev, B.; Paxson, V. (2010): Netalyzr: illuminating the edge network. *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, pp. 246-259.

Pappas, V.; Fältström, P.; Massey, D.; Zhang, L. (2004): Distributed DNS troubleshooting. *Proceedings of the ACM SIGCOMM Workshop on Network Troubleshooting: Research, Theory and Operations Practice Meet Malfunctioning Reality*, pp. 265-270.

Park, K.; Pai, V. S.; Peterson, L. L.; Wang, Z. (2004): CoDNS: improving DNS performance and reliability via cooperative lookups. *USENIX Symposium on Operating Systems Design and Implementation*, vol. 4, pp. 14.

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

Root Zone KSK Rollover Project (2019): https://www.icann.org/resources/pages/ksk-rollover.

Weaver, N.; Kreibich, C.; Paxson, V. (2011): Redirecting DNS for Ads and profit. *FOCI*, no. 2, pp. 2-3.