CertDNS: Guaranteeing the Integrity of DNS Records Using PKIX Certificates

Hyeonmin Lee*
University of Virginia
Charlottesville, VA, United States
frv9vh@virginia.edu

Sangyoon Seok
Seoul National University
Seoul, Republic of Korea
syseok@mmlab.snu.ac.kr

Taekyoung (Ted) Kwon†
Seoul National University
Seoul, Republic of Korea
tkkwon@snu.ac.kr

***Abstract*—The Domain Name System (DNS) maps the domain name of a host to its resource records, and hence serves as a critical Internet infrastructure. However, ensuring the integrity of DNS messages was not considered in its original design. To address this gap, DNS Security Extensions (DNSSEC) were standardized and refined over the past two decades. Despite this effort, DNSSEC adoption remains low, with only about 7% of second-level domain names adopting DNSSEC, possibly due to its operational complexity, in particular the need to upload DNSSEC-related records to parent zones. To overcome these obstacles, we propose CertDNS, a practical alternative that enables individual DNS zones to guarantee the integrity of their records without relying on other entities in the DNS hierarchy. By leveraging PKIX (Public Key Infrastructure X.509) certificates issued by Certificate Authorities (CAs), CertDNS allows domains to sign their records using their existing keys, thereby granting them greater autonomy and simplifying deployment compared to DNSSEC. Our prototype demonstrates that CertDNS achieves 30% faster resolution time and requires significantly less storage than DNSSEC, showing its practicality for real-world use.**

***Index Terms*—DNS, DNS Security, DNSSEC, PKIX Certificate**

I. Introduction

The Domain Name System (DNS) [23] maps domain names to their resource records like IP addresses (`A` records) and mail servers (`MX` records). However, the original design of DNS did not incorporate security mechanisms, making it vulnerable to attacks such as DNS spoofing and cache poisoning [4], [29]. To address these problems, the DNS Security Extensions (DNSSEC) were introduced to provide authentication and integrity for DNS records [1], [26]–[28]. Since its introduction, DNSSEC adoption has increased, and currently 93% of Top-Level Domains (TLDs) adopt DNSSEC [19].

In contrast, adoption among Second-Level Domains (SLDs) remains very low. Recent statistics show that only 4.2% of `.com`, 4.9% of `.net`, and 5.0% of `.org` domains have deployed DNSSEC. Also, it is reported that mismanagement in DNSSEC is pervasive due to operational complexity [8], [21]. To deploy DNSSEC, a domain has to publish three additional types of DNS records: `DNSKEY`, `RRSIG`, and `DS` (the details of each record are provided in Section II). The domain owner can generate and publish `DNSKEY` and `RRSIG` records locally, but publishing `DS` records requires a much more *complex* procedure. Specifically, `DS` records must be uploaded to the domain's parent DNS zone to establish a *chain of trust*; for example, `example.com` must upload its `DS` record to the `.com` zone. To do so, the domain owner has to pass it to the registrar[1], which is often a *manual* process. In practice, this often means contacting the registrar via chat or email and requesting the upload of the `DS` record. However, many domains fail in this step; nearly 30% of domains that claim to support DNSSEC do not have `DS` records in their parent zones [8].[2] Furthermore, the upload process is error-prone (e.g., manual copy-and-paste) and some registrars do not even support it [9]. As a result, domain owners may upload incorrect `DS` records or cannot upload them at all, even if their `DNSKEY` and `RRSIG` records are properly published. Thus, the difficulty of uploading `DS` records to parent zones remains one of the major obstacles to DNSSEC deployment.

We propose CertDNS as an alternative that enables domain owners to ensure end-to-end DNS integrity without requiring the chain of trust across DNS zones. We leverage the fact that most domains already use digital certificates issued by trusted Certificate Authorities (CAs)[3] to support TLS; for instance, 95% of web traffic on Google Chrome from Windows and macOS is delivered over TLS (e.g., HTTPS) [12].

In CertDNS, a domain can use a public/private key pair from its certificate issued by a CA to sign its DNS records. Note that CertDNS reuses DNS record types already defined in IETF standards. Specifically, the domain generates signatures for its records with its private key and publishes them as `RRSIG` records. The corresponding public key is published as a `DNSKEY` record, allowing clients to verify the signatures. In addition, the domain publishes its certificate (containing the public key) and the certificates of its CA chain using `CERT` records [18]. In this design, the signature ensures the integrity of the DNS record, while the certificate ensures that the signature was generated by the domain's private key;

*This work was conducted while the first author was a postdoctoral researcher at Seoul National University.

†Corresponding author.

[1]A registrar provides domain registration services to end users and mediates operational tasks such as `DS` record management.

[2]The `.com` zone operator must also upload its own `DS` record to the root servers; however, the coordination between root servers and TLD name servers is relatively straightforward and outside the scope of this paper.

[3]Here, a trusted CA refers to a certificate authority that is publicly trusted and included in the root stores maintained by major operating systems [2], [22], [24], as well as the CCADB [30].

the CertDNS design will be detailed in Section IV-B. A client can check the integrity of the domain's DNS record by fetching and verifying the corresponding signature record (i.e., `RRSIG` record), domain's public key (i.e., `DNSKEY` record), and the domain's certificate (i.e., `CERT` record). In this way, CertDNS guarantees the integrity of DNS records without requiring cooperation from other DNS entities (e.g., uploading `DS` records to parent zones), thereby granting domain owners greater autonomy and control over DNS integrity and making deployment easier than DNSSEC.

This paper makes the following contributions:

- We design CertDNS, a certificate-based mechanism that ensures the integrity and authenticity of DNS records. By leveraging digital certificates to generate signatures, CertDNS avoids cross-zone dependencies, giving domain owners more autonomy and flexibility (than DNSSEC) in securing their DNS records.
- We implement and evaluate a prototype of CertDNS, demonstrating lower query resolution delay and significantly reduced storage overhead compared to DNSSEC.

II. Background

We provide an overview of DNSSEC and PKIX certificates.

DNS and DNSSEC. The DNS maps domain names to resources (e.g., hostnames to IP addresses) through a hierarchical namespace, where authoritative servers manage records within their zones. Clients query a recursive resolver, which follows the hierarchy to obtain records and returns the response. Because DNS was not designed with authentication, clients cannot verify the authenticity of responses, making DNS vulnerable to spoofing and cache poisoning [4], [29].

DNSSEC [1], [26]–[28] is a major attempt to address the above-mentioned vulnerabilities by providing the integrity and authentication for DNS records. It introduces several new DNS record types; the most relevant ones are as follows:

- A `DNSKEY` record contains the public key used to validate an `RRSIG` record for a given DNS record. DNSSEC uses two public/private key pairs for each zone: a Key Signing Key (KSK) and a Zone Signing Key (ZSK). The KSK signs `DNSKEY`, while the ZSK signs all other records in the zone.
- An `RRSIG` record contains the cryptographic signature of a DNS record. If the `RRSIG` record passes the validation, the integrity of the corresponding DNS record is ensured.
- A `DS` record contains the hash of the KSK, which is uploaded to the parent DNS zone to establish a chain of trust across zones.

To establish a DNSSEC chain up to the root, these records must be published correctly. Figure 1 illustrates how a chain of trust is built from a name server's DNS record to the root zone's `DNSKEY` record. Validation starts from the root according to the DNS hierarchy: Using the pre-installed KSK of a root name server, a (client-side) resolver validates the root zone's `DNSKEY` record, and thus it can validate the root zone's ZSK. Here, validating a DNS record (or a key in it) means validating its `RRSIG` for a given record. Next, the ZSK

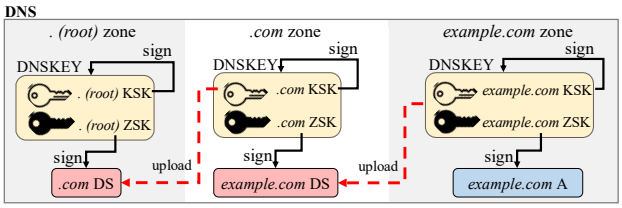

Fig. 1. A DNSSEC trust chain is illustrated. The DNS records (i.e., `A` records) of `example.com` are signed by its ZSK (in the `DNSKEY` record), and the `DNSKEY` record is signed by its KSK. The KSK of `example.com` is uploaded to `.com` as a `DS` record to form a trust chain.

is then used to validate the `DS` record of a child zone (e.g., in the root zone), which in turn validates the child zone's KSK. This process is repeated for the lower-level name servers until the requested record is verified.

- Operational difficulties. In practice, forming the chain of trust requires uploading a domain's `DNSKEY` as a `DS` record to its parent zone. This process is often manual (e.g., copy-and-paste through email or web chat) and therefore error-prone, and some registrars do not even support it [9]. As a result, DNSSEC deployment frequently suffers from misconfigurations and failures in establishing the required chain of trust.

PKIX certificate. A digital certificate binds a public key to the name of its holder (e.g., a domain owner). The PKIX (Public Key Infrastructure X.509) [6] certificate format is widely used and is typically issued by Certificate Authorities (CAs) that form a hierarchical trust model. A domain's PKIX certificate can be validated by verifying the certificate chain from a trusted root CA to the domain's leaf certificate.

HTTPS is a common application of PKIX certificates; about 95% of page loads on Google Chrome from Windows and macOS are delivered over HTTPS [12]. Therefore, we can assume that most domains already possess certificates and are familiar with their issuance and management. Thus, if we exploit such certificates (issued by CAs) to sign DNS records, domains can ensure the integrity of their DNS records with a low technical barrier, particularly compared to DNSSEC.

III. Related Work

DNS security. Since DNS packets are sent in cleartext, they are vulnerable to attacks such as eavesdropping and cache poisoning. To protect the confidentiality of DNS messages between clients and resolvers, several solutions have been proposed. DNS-over-TLS (DoT) [15] encrypts DNS queries and responses using TLS, while DNS-over-HTTPS (DoH) [14] encapsulates DNS traffic in HTTPS. However, both DoT and DoH protect DNS messages only between clients and recursive resolvers. Thus, *DNS traffic between resolvers and authoritative name servers remains unprotected and vulnerable*.

To address this limitation and provide end-to-end integrity and authenticity of DNS messages (from clients to authoritative name servers), DNSSEC [1], [26]–[28] was standardized around 2000. Nevertheless, its deployment has been slow, which leads to numerous studies that analyzed DNSSEC adoption and operational challenges. Chung et al. [8] reported

TABLE I
COMPARISON OF DNS SECURITY PROTOCOLS WITH RESPECT TO THEIR SECURITY PROPERTIES (AUTHENTICITY [A], INTEGRITY [I], AND CONFIDENTIALITY [C]), SCOPE OF PROTECTION, AND CROSS-ZONE OPERATIONAL DEPENDENCIES.

Method	Security properties	Protect DNS messages between	Dependency
DoT/DoH	A, I, C	Client – Recursive resolvers	–
DNSSEC	A, I	Client – Auth. name servers	O (parent zone)
CertDNS	A, I	Client – Auth. name servers	X

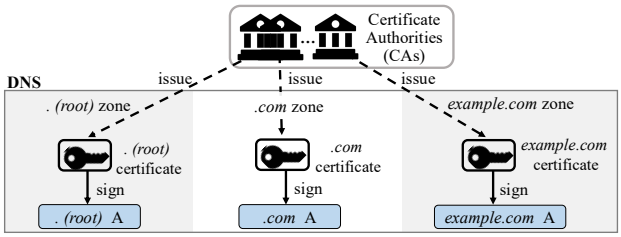

Fig. 2. Trust chain of CertDNS based on PKI is illustrated. Each domain holds a certificate and signs its DNS records (e.g., A records).

that only 1.2% of domains (in .com, .net, and .org) deployed DNSSEC. Moreover, prior work [8], [9], [21] found that many DNSSEC-enabled domains suffer from misconfigurations, most notably failures in uploading DS records.

Unlike DNSSEC, CertDNS avoids operational dependencies on parent zones or registrars. Instead, it relies on the well-established PKI for its operations.

Certificate-based DNS security. Fetzer et al. [11] also proposed incorporating certificates into DNS security. Their scheme suggests that a domain owner use its own certificate to sign DNS records and store the certificate on the name server. However, their design requires a separate certificate for every server (i.e., each physical machine hosting a name server); the more servers are used, the more certificates should be issued and stored as DNS records. In addition, the proposal does not specify how certificates should be delivered to clients or how signatures should be attached to DNS records or responses. As a result, their work remained at a conceptual level without substantial description or evaluation. Rishith et al. [25] proposed a TLS-based approach by authenticating the resolver–authoritative hop via TLS and IP-bound certificates. This design strengthens resolver–authoritative communication even when the DNSSEC chain is broken, but it provides transport-level protection rather than DNSSEC-style, record-level authenticity when DNSSEC signing is absent.

Novelty of our work. First, prior DNS security techniques either (i) focus on encrypting traffic between clients and resolvers (e.g., DoT, DoH) or (ii) rely on DNSSEC, which provides end-to-end integrity guarantees but introduces operational dependencies on parent zones and registrars. In contrast, CertDNS eliminates these dependencies while reusing existing DNS record types to ensure the end-to-end integrity of DNS records with minimal operational overhead. Table I shows the comparison across security properties and operational scope.

Unlike prior work [11], [25], CertDNS utilizes PKIX certificates already widely deployed (for HTTPS) and distributes certificates/keys and signatures via standard DNS record types, enabling record-level verification without introducing an additional TLS channel on the validation path. Through prototype-based experiments, we show that CertDNS achieves lower resolution delay and smaller storage overhead than DNSSEC, making it a practical alternative for real-world deployment.

IV. CERTDNS DESIGN

Here, we present the central idea and design of CertDNS.

A. Motivation and Design Requirements

The main drawback of DNSSEC lies in its low deployment rate and pervasive misconfiguration due to operational complexity. As discussed in Section II, establishing a chain of trust in DNSSEC requires domain owners to complete a complex and (often manual) process. Consequently, even domains that enable DNSSEC (e.g., by publishing DNSKEY and RRSIG records) commonly suffer from misconfigurations such as missing DS records [8], [9], [21].

To provide DNS record integrity in a more practical and deployable manner, a solution should avoid fragile dependencies within the DNS hierarchy (e.g., uploading DS records via registrars to parent zones), while instead exploiting infrastructure that is already mature and widely deployed. From this perspective, we identify two design requirements for CertDNS:

- *Avoid fragile DNS dependencies*: DNSSEC introduces additional dependencies on parent zones and registrars, such as the requirement to upload DS records, which has proven to be complex and error-prone. CertDNS, in contrast, should operate without such hierarchical dependencies.
- *Reuse of current DNS practice*: CertDNS should reuse the current DNS standards, such as DNS record types, to ensure backward compatibility with today's DNS infrastructure.

B. Design Details

The central idea of CertDNS is to guarantee and verify the integrity of DNS records using PKIX certificates issued by CAs. In CertDNS, a domain signs its records with its private key. Then, the domain publishes its signature, public key (corresponding to the private key), and certificate as RRSIG, DNSKEY, and CERT record, respectively. The public key is used by clients to verify the signatures.

Here, (1) the signature (i.e., RRSIG) attached to a DNS record guarantees its integrity, and (2) the public key (i.e., DNSKEY) together with the certificate (i.e., CERT) ensure that the signature was generated by the domain (owner). Thus, the integrity of the DNS records can be verified by validating the corresponding signatures and a certificate (chain). In this way, CertDNS allows domain owners to generate the signatures of their DNS records solely from their certificates *without requiring any cooperation from other DNS entities such as parent zones*. Figure 2 shows how to guarantee the integrity of DNS records in CertDNS (compared to DNSSEC in Figure 1).

Deploying CertDNS. We suggest reusing existing DNS record types:

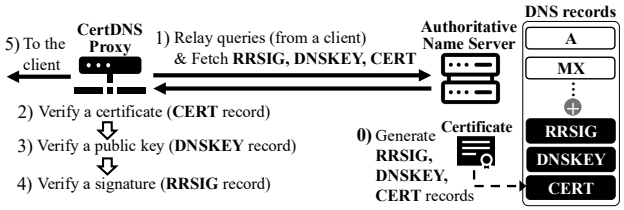

Fig. 3. Overview of CertDNS operation.

- An RRSIG record [28] stores the cryptographic signature of a DNS record. As in DNSSEC, a DNS operator can generate a single signature for a set of RRs. For simplicity, we assume one record per signature here. The private key of the domain is used to generate the signature.
- A DNSKEY record [28] stores the public key corresponding to the private key used for signing. For CertDNS, the *Flags* field in the DNSKEY record type is extended: we propose using a non-reserved bit to indicate CertDNS usage. If this bit is set, a proxy (or client) must verify the signature using the certificates in the CERT records.
- A CERT record [18] stores the domain's certificate, and, if applicable, its intermediate CA certificates so that clients can validate the complete chain.[4]

With these record types, a domain can ensure DNS integrity by: (1) generating a public/private key pair, (2) obtaining a certificate (and its public key) from a CA, (3) signing its DNS records with the private key, and finally, (4) publishing the signatures, public key, and certificate (chain) as RRSIG, DNSKEY, and CERT records, respectively. Certificate issuance and renewal can be automated via ACME (e.g., Let's Encrypt), reducing the manual effort required of domain operators.

Verifying DNS records in CertDNS. The integrity of DNS records can be verified as shown in Figure 3, while preserving compatibility with legacy clients and recursive resolvers. For clarity, we describe the case where a CertDNS proxy is placed on a client machine, though it may also be co-located with a recursive resolver. [5]

1) When a CertDNS proxy receives a DNS query (e.g., for an A record) from its client, it relays the query to the authoritative name server (assuming the root and TLD servers have been contacted).[6] The proxy also issues additional queries for the record's signature (RRSIG), the corresponding public key (DNSKEY), and the certificate chain (CERT).
2) Upon receiving the DNS responses (i.e., A, RRSIG, DNSKEY, and CERT records), the proxy first verifies the certificates (chain) in the CERT record. If the chain cannot be validated, the verification fails[7].

[4]We assume that resolvers already maintain the certificates of root CAs.

[5]In practice, such a proxy can initially be deployed as a standalone component, but with wider adoption, it may be integrated into recursive resolvers as part of their standard functionality.

[6]Every query from the proxy server to the authoritative server is delivered via the recursive resolver, which we omit in Figure 3 for brevity.

[7]The revocation status of the certificate should be checked as well, which is skipped for the sake of brevity

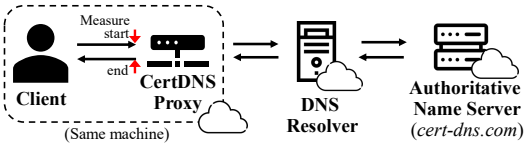

Fig. 4. Experiment settings for evaluating CertDNS.

3) The proxy extracts the public key from both the leaf certificate and the DNSKEY record. If the two keys do not match, verification fails.
4) Using the verified public key, the proxy checks the signature of the requested record.
5) If the signature is valid, the proxy returns the DNS record to the client.

In this way, CertDNS offers domain owners a practical means of providing DNS integrity without requiring cooperation from parent zones or registrars. While clients or resolvers should be extended to support CertDNS, existing root and TLD servers remain unaffected, as further evaluated in Section V-C.

Satisfying the design requirements. CertDNS meets the requirements identified in Section IV-A as follows:

- *Avoid fragile DNS dependencies*: CertDNS eliminates the need for domains to rely on parent zones for DS record uploading. Instead, it leverages CA-issued certificates, which are already widely deployed and straightforward to obtain.
- *Reuse of current DNS practice*: CertDNS relies solely on existing DNS record types, ensuring seamless coexistence with the current DNS infrastructure.

V. Performance Evaluation

In this section, we evaluate the performance of CertDNS through a prototype implementation.

A. Prototype Implementation and Experiment Settings

Figure 4 illustrates our experimental setup.

Client. We set up a Linux client on an Amazon EC2 instance. The client is unaware of the proxy and issues DNS queries to its configured resolver as usual.

CertDNS proxy. We implement CertDNS as a client-side proxy in Python, running on the same EC2 instance as the client. The proxy intercepts and relays queries from the client to the resolver, while also generating additional queries for RRSIG, DNSKEY, and CERT records. It then verifies the signature of the DNS response and, if successful, forwards the validated response to the client.

Recursive resolver. We install a recursive resolver on a separate EC2 instance using BIND 9 [17]. To eliminate caching effects, we flush the resolver's DNS cache after each DNS response is delivered to the client. Thus, every query is handled recursively, from the root server, through the TLD server, down to the authoritative server for our test domain.

Authoritative name server. For experimental purposes, we registered the domain cert-dns.com. An authoritative name server for cert-dns.com is deployed on another EC2 instance running BIND 9. We generate and publish

TABLE II
THE QUERY RESOLUTION TIMES ARE COMPARED AMONG VANILLA DNS (UNPROTECTED), DNSSEC, AND CERTDNS (UNIT: SECONDS).

Time	DNS (Unprotected)	DNSSEC	CertDNS
Query resolution	0.2s	0.35s	0.27s

TABLE III
STORAGE OVERHEADS OF DNSSEC AND CERTDNS AT A TLD WITH 6.9 MILLION SIGNED SLDS. EACH: SIZE OF A SINGLE RECORD, COUNT: NUMBER OF RECORDS, TOTAL: OVERALL STORAGE.

Record type	DNSSEC			CertDNS		
	Each	Count	Total	Each	Count	Total
DS	68 B	6.9 M	452 MB	-	-	-
CERT	-	-	-	1.8 KB	2	3.6 KB
DNSKEY, RRSIG	*Same for both DNSSEC and CertDNS (omitted)*					

the necessary DNS records for CertDNS (RRSIG, DNSKEY, and CERT). For CERT records, we obtained a certificate for cert-dns.com from Let's Encrypt and use it for our experiments.

Using this setup, we conduct experiments in three modes: (1) vanilla DNS, (2) DNSSEC, and (3) CertDNS. Vanilla DNS serves as a baseline without any security guarantees. When testing DNSSEC, we enable DNSSEC for our domain. In both vanilla DNS and DNSSEC, the client communicates directly with the resolver without a proxy. We evaluate the three modes in terms of query time and storage overhead.

B. Time Overhead

We first measure the query resolution time. The delay is measured at the client side, from the moment a DNS query is issued to the moment the proxy returns the verified response (as shown in Figure 4). Each measurement involves the client querying an A record using the Linux dig command [16]. We report the average over 100 runs for each mode.

Table II presents the query resolution times for vanilla DNS, DNSSEC, and CertDNS. Without any security extensions, vanilla DNS resolves queries in about 0.2 seconds. With DNSSEC, the resolution time increases to approximately 0.35 seconds. CertDNS achieves about 0.27 seconds, which is roughly 30% faster than DNSSEC.

DNSSEC incurs additional delay because the resolver retrieves and validates records (e.g., DNSKEY, RRSIG, DS) at each level of the hierarchy, including the root, .com, and cert-dns.com zones. By contrast, CertDNS requires the resolver to fetch DNSKEY, RRSIG, and CERT records only from the cert-dns.com zone. Therefore, CertDNS achieves a *shorter resolution time compared to DNSSEC*.[8]

C. Storage Overhead

Both DNSSEC and CertDNS require domains to publish additional DNS records, which imposes storage overhead on name servers. Since TLD name servers bear the largest burden, we focus on them in this analysis. To compare the storage overhead of DNSSEC and CertDNS at TLDs, we measure the storage requirements as the number of SLDs increases. We exclude RRSIG and DNSKEY records since DNSSEC and CertDNS are assumed to have the same method of signatures.

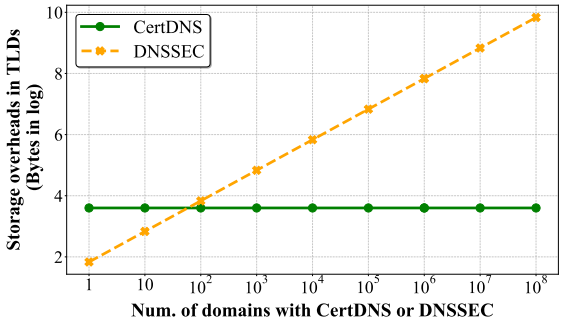

Fig. 5. The storage overheads of DNSSEC and CertDNS are plotted as the number of domains that sign their DNS records increases.

DNSSEC. Each SLD generates a DNSKEY record and uploads its hash to the parent zone as a DS record. While the overhead of DNSKEY is trivial for individual SLD name servers (e.g., cert-dns.com), TLD name servers must store all corresponding DS records, which can be substantial. As of September 2025, the .com zone contains about 6.9 million DS records (out of 156 million domains) [7]. As shown in Table III, if 6.9 million domains publish DS records using the SHA-256 algorithm, the total amounts to about 452 MB.

CertDNS. CertDNS eliminates the need to upload DS records to parent zones. An SLD only (e.g., cert-dns.com) publishes two CERT records (a leaf and an intermediate CA certificate), totaling about 3.6 KB[9]. Thus, even if 6.9 million domains adopt CertDNS, the storage overhead at the .com TLD remains negligible: around 0.0008% of DNSSEC's.

Moreover, while DNSSEC's storage overhead grows with the number of SLDs (since each requires a DS record at the parent TLD), CertDNS imposes no such growth. As shown in Figure 5, the .com TLD needs to store only its own CERT records, regardless of how many SLDs deploy CertDNS. Therefore, CertDNS has a *fixed and minimal storage requirement, independent of the number of SLDs*.

D. Summary

Our results show that CertDNS strikes a practical balance between security and efficiency. It resolves queries about 30% faster than DNSSEC (0.27s vs. 0.35s on average) while adding only modest delay compared to vanilla DNS. For storage, DNSSEC forces TLDs to maintain millions of DS records (e.g., 452 MB for 6.9 M domains in .com), whereas CertDNS requires only a domain's own certificates (about 3.6 KB), yielding a negligible footprint regardless of scale.

VI. SECURITY CONSIDERATIONS

Challenges with CAs. CertDNS relies on certificates issued by CAs to generate signatures for DNS records. However,

[8]If DNSSEC or CertDNS records are cached by a resolver, their query response times may converge.

[9]We assume that root CA certificates are already trusted and stored by clients or resolvers.

certificate issuance is not constrained by DNS namespace boundaries, and any trusted CA may issue a certificate for any domain. Compromised or misbehaving CAs have issued fraudulent certificates in the past [5], [10], enabling domain impersonation. In CertDNS, mis-issuance can directly undermine DNS integrity because an attacker with a mis-issued certificate can generate valid-looking signatures over forged DNS records. This trust model differs from DNSSEC, where record authenticity is anchored in the DNS hierarchy via DS-based delegation; CertDNS instead shifts trust to the CA ecosystem. Domains can reduce exposure by restricting authorized issuers via CAA [13] and leveraging Certificate Transparency (CT) to detect mis-issuance and trigger a timely response [20]. These mechanisms improve accountability and detectability, but do not eliminate CA-compromise risks. Thus, CertDNS benefits from continuous monitoring and prompt revocation/rotation upon mis-issuance detection. Moreover, reusing the same certificate for TLS and CertDNS increases the blast radius of compromise, potentially enabling attacks on both TLS and DNS; we thus recommend using a dedicated certificate (and key pair) for CertDNS signing.

Authenticated denial of existence. We recommend using the `NSEC` [28] or `NSEC3` [3] mechanisms. They allow a server to prove to a client the absence of a record type by pointing to the next valid record.

VII. Conclusion

We presented CertDNS, a system that ensures the integrity of DNS records by leveraging PKIX certificates. In CertDNS, a domain signs its DNS records with a private key corresponding to the public key in its CA-issued certificate. This approach allows domains to guarantee the integrity of their records without relying on parent zones or registrars, addressing one of the major operational challenges of DNSSEC (e.g., the `DS` record uploading). We implemented a prototype with existing DNS record types and showed that CertDNS achieves lower query resolution time and greatly reduces storage overhead compared to DNSSEC, demonstrating its practicality for deployment.

Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [NO.RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)], by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2022R1A2C2011221), and by Institute of Information & communications Technology Planning & Evaluation (IITP) under Next-generation Cloud-native Cellular Network Leadership Program (IITP-2025-RS-2024-00418784) grant funded by the Korea government (MSIT).

References

[1] D. E. E. 3rd. Domain Name System Security Extensions. RFC 2535, Mar. 1999.

[2] Apple. Available trusted root certificates for Apple operating systems, 2025. https://support.apple.com/en-us/103272 (Accessed: Sep 21, 2025).

[3] R. Arends, G. Sisson, D. Blacka, and B. Laurie. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. RFC 5155, Mar. 2008.

[4] D. Atkins and R. Austein. Threat Analysis of the Domain Name System (DNS). RFC 3833, Aug. 2004.

[5] BBC. Trustwave to escape 'death penalty' for SSL skeleton key, 2012. https://www.theregister.com/2012/02/14/trustwave_analysis/ (Accessed: Sep 20, 2025).

[6] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, May 2008.

[7] F. Cambus. statdns: DNS and Domain Name statistics and tools. https://www.statdns.com/ (Accessed: Sep 21, 2025).

[8] T. Chung, R. van Rijswijk-Deij, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. A Longitudinal, End-to-End View of the DNSSEC Ecosystem. In 26th USENIX Security Symposium (USENIX Security 17), pages 1307–1322, 2017.

[9] 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, pages 369–383, 2017.

[10] Comodo. Comodo-Fraud-Incident-2011-03-23, 2011. https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html (Accessed: Sep 20, 2025).

[11] C. Fetzer, G. Pfeifer, and T. Jim. Enhancing DNS security using the SSL trust infrastructure. In 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, pages 21–27, 2005.

[12] Google Transparency Report. HTTPS encryption on the web. https://transparencyreport.google.com/https/overview (Accessed: Sep 20, 2025).

[13] P. Hallam-Baker and R. Stradling. DNS Certification Authority Authorization (CAA) Resource Record. RFC 6844, Jan. 2013.

[14] P. E. Hoffman and P. McManus. DNS Queries over HTTPS (DoH). RFC 8484, Oct. 2018.

[15] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. E. Hoffman. Specification for DNS over Transport Layer Security (TLS). RFC 7858, May 2016.

[16] IBM. dig command, 2025. https://www.ibm.com/docs/en/aix/7.3.0?topic=d-dig-command (Accessed: Sep 22, 2025).

[17] Internet Systems Consortium, Inc. BIND 9. https://www.isc.org/bind/ (Accessed: Sep 21, 2025).

[18] S. Josefsson. Storing Certificates in the Domain Name System (DNS). RFC 4398, Mar. 2006.

[19] R. Lamb. DNSSEC Deployment Report. http://rick.eng.br/dnssecstat/ (Accessed: Sep 21, 2025).

[20] B. Laurie, E. Messeri, and R. Stradling. Certificate Transparency Version 2.0. RFC 9162, Dec. 2021.

[21] H. Lee, A. Gireesh, R. van Rijswijk-Deij, T. T. Kwon, and T. Chung. A Longitudinal and Comprehensive Study of the DANE Ecosystem in Email. In 29th USENIX Security Symposium (USENIX Security 20), 2020.

[22] Microsoft. Release notes - Microsoft Trusted Root Certificate Program, 2025. https://learn.microsoft.com/en-us/security/trusted-root/release-notes (Accessed: Sep 21, 2025).

[23] P. Mockapetris. Domain names - concepts and facilities. RFC 1034, Nov. 1987.

[24] Mozilla Wiki. Mozilla's CA Certificate Program, 2025. https://wiki.mozilla.org/CA (Accessed: Sep 21, 2025).

[25] A. Rishith, A. Kulkarni, T. Das, and V. Balachandran. Overcoming DNSSEC Islands of Security: A TLS and IP-Based Certificate Solution. In 2024 IEEE Conference on Engineering Informatics (ICEI), pages 1–7. IEEE, 2024.

[26] S. Rose, M. Larson, D. Massey, R. Austein, and R. Arends. DNS Security Introduction and Requirements. RFC 4033, Mar. 2005.

[27] S. Rose, M. Larson, D. Massey, R. Austein, and R. Arends. Protocol Modifications for the DNS Security Extensions. RFC 4035, Mar. 2005.

[28] S. Rose, M. Larson, D. Massey, R. Austein, and R. Arends. Resource Records for the DNS Security Extensions. RFC 4034, Mar. 2005.

[29] S. Son and V. Shmatikov. The hitchhiker's guide to DNS cache poisoning. In International Conference on Security and Privacy in Communication Systems, pages 466–483. Springer, 2010.

[30] The Linux Foundation. Common CA Database (CCADB), 2025. https://www.ccadb.org/ (Accessed: Sep 21, 2025).