A Longitudinal, End-to-End View of the DNSSEC Ecosystem

Taejoong Chung
Northeastern University

Roland van Rijswijk-Deij
University of Twente and SURFnet

Balakrishnan Chandrasekaran
TU Berlin

David Choffnes
Northeastern University

Dave Levin
University of Maryland

Bruce M. Maggs
Duke University and Akamai Technologies

Alan Mislove
Northeastern University

Christo Wilson
Northeastern University

Abstract

The Domain Name System's Security Extensions (DNSSEC) allow clients and resolvers to verify that DNS responses have not been forged or modified in-flight. DNSSEC uses a public key infrastructure (PKI) to achieve this integrity, without which users can be subject to a wide range of attacks. However, DNSSEC can operate only if each of the principals in its PKI properly performs its management tasks: authoritative name servers must generate and publish their keys and signatures correctly, child zones that support DNSSEC must be correctly signed with their parent's keys, and resolvers must actually validate the chain of signatures.

This paper performs the first large-scale, longitudinal measurement study into how well DNSSEC's PKI is managed. We use data from *all* DNSSEC-enabled subdomains under the `.com`, `.org`, and `.net` TLDs over a period of 21 months to analyze DNSSEC deployment and management by domains; we supplement this with active measurements of more than 59K DNS resolvers worldwide to evaluate resolver-side validation.

Our investigation reveals pervasive mismanagement of the DNSSEC infrastructure. For example, we found that 31% of domains that support DNSSEC fail to publish all relevant records required for validation; 39% of the domains use insufficiently strong key-signing keys; and although 82% of resolvers in our study request DNSSEC records, only 12% of them actually attempt to validate them. These results highlight systemic problems, which motivate improved automation and auditing of DNSSEC management.

1 Introduction

The Domain Name System (DNS) [36] provides a scalable, flexible name resolution service. Unfortunately, DNS has long been fraught with security issues such as DNS spoofing and cache poisoning [3, 28, 46] To address these problems, DNS Security Extensions (DNSSEC) [20] were introduced nearly two decades ago. At its core, DNSSEC is a hierarchical public key infrastructure (PKI) that largely mirrors the DNS hierarchy and is rooted at the root DNS zone. To enable DNSSEC, the owner of a domain signs its DNS records and publishes the signatures along with its public key; this public key is then signed by its parent domain, and so on up to the root DNS zone. A resolver validates a signed DNS record by recursively checking the associated signatures until it reaches the well-known root zone trusted key.

Largely in response to powerful attacks such as the Kaminsky Attack [28], DNSSEC adoption has increased recently. As of early 2017, more than 90% of top-level domains (TLDs) and 47% of country-code TLDs (ccTLDs) are DNSSEC-enabled [26, 47]. Widely-used DNS resolvers now attempt DNSSEC validation by default, e.g., as of January 2012 Comcast (one of the largest ISPs in the US) requests and validates DNSSEC records for all queries [32], and Google (which operates the largest public DNS resolver) did the same in March 2013 [22].[1]

But like any PKI, DNSSEC can only function correctly when all principals—every signatory from root to leaf, and the resolver validating the signatures—fulfill their respective responsibilities. Unfortunately, DNSSEC is complex, creating many opportunities for mismanagement. *On the server side*, a single error such as a weak key, an expired signature, or a broken signature chain can weaken or totally compromise the integrity of a large number of domains. *On the client side*, mismanaged or buggy DNS resolvers can obviate all server-side efforts by simply failing to catch invalid or missing signatures.

Surprisingly little is known about how well the DNSSEC PKI ecosystem is managed. While there have

[1] It is important to note that these resolvers still accept responses without DNSSEC records, as the vast majority of domain administrators have yet to deploy DNSSEC.

been many studies of DNSSEC, we find that no prior efforts had the data to allow them to study the DNSSEC PKI *at scale*—across many domains and resolvers—and *longitudinally*—by monitoring their behavior over time. For example, server-side studies have shown instances of mismanagement, but only for samples of domain names [12–14]. Likewise, prior studies of DNS resolvers have used ad campaigns, which do not permit repeated, controlled measurements of resolvers over time [7, 33, 47]. What has made large-scale, longitudinal studies of DNSSEC so challenging is a dearth of DNSSEC record datasets and a lack of vantage points from which to repeatedly measure resolver behavior.

In this paper, we present a comprehensive study of the entire DNSSEC ecosystem—encompassing signers, authoritative name servers, and validating DNS resolvers—to understand how DNSSEC is (mis)managed today. To study server-side behavior, our work relies on 21 months of daily snapshots, and three months of hourly snapshots, of DNSSEC records for *all* signed `.com`, `.net`, and `.org` second-level domains. To study client-side behavior, we leverage the Luminati HTTP/S proxy service [35], which allows us to perform repeated,

Our analysis reveals troubling, persistent mismanagement in the DNSSEC PKI:

- *First*, we find that nearly *one-third* of DNSSEC-enabled domains produce records that *cannot be validated* due to missing or incorrect records. 1.7% of signed domains fail to provide `RRSIG`s for `SOA` records, while 30% of signed domains do not have `DS` records. The vast majority of these missing records are due to large hosting providers that fail to publish the correct records for domains they manage. Additionally, we find that 0.6% of signed domains provide incorrect `RRSIG`s for both `SOA` and `DNSKEY` records.
- *Second*, we identify four large providers that use the same keys to sign all of their managed domains. This unnecessary key reuse makes all of the domains vulnerable to the compromise of a single shared key. For example, we find that a single key is shared by over 132K domains.
- *Third*, we observe widespread use of 1024-bit RSA keys, which are now considered "weak" (smaller than the NIST-recommended minimum size of 2048 bits [1]). 39% of domains use weak Key Signing Keys (KSKs), and over 90% of domains use weak Zone Signing Keys (ZSKs). DNSSEC is designed to be resilient against weak and stolen keys via frequent key rotation, but we find that 70% of domains never rotated their KSK during our 21-month study.
- *Fourth*, we find that although 83% of observed resolvers request DNSSEC records during their queries, only 12% of them actually validate the records (defeating the purpose of DNSSEC). This finding motivates the need to reexamine approaches using query logs from authoritative name servers as a lens to measure DNSSEC adoption by resolvers [21, 23].

In summary, our results paint a distressing picture of widespread mismanagement of keys and DNSSEC records that violate best practices in some cases, and completely defeat the security guarantees of DNSSEC in others. On a more positive note, our findings demonstrate several areas of improvement where management of the DNSSEC PKI can be automated and audited. To this end, we publicly release all of our analysis code and data (where possible[2]) to the research community at

`https://securepki.org`

thereby allowing other researchers and administrators to reproduce and extend our work.

2 Background

We begin by presenting an overview of both DNS and DNSSEC.

DNS and DNSSEC DNS uses *records* to map *domain names* to *values* (e.g., an `A` record maps a domain name to an IPv4 address; an `NS` record maps a domain name to the authoritative name server for a domain). DNS is designed to encourage caching, and every DNS record contains a time-to-live (TTL), specifying how long the records can be cached for. The original DNS protocol did not include security, allowing an adversary to forge DNS responses to carry out attacks. DNS Security Extensions (commonly referred to as DNSSEC) are designed to address this vulnerability [4–6, 19]. DNSSEC provides integrity for DNS records using three primary record types[3]:

`DNSKEY` records, which are public keys used in DNSSEC. Typically, each zone uses two `DNSKEY` records to sign DNS records, as discussed below.

`RRSIG` (Resource Record Signature) records, which are cryptographic signatures of other records. Each `RRSIG` is a signature over all records of a given type for a certain name; this set is called an RRSet. For example, all `A` records for `example.org` will be authenticated by a single `RRSIG` (i.e., the

[2]Our `.com`, `.org`, and `.net` zone files are collected under agreement with the zone operators; while we are not permitted to release this data, we provide links where other researchers can obtain access themselves.

[3]There are other record types for expressing the non-existence of records (`NSEC` and `NSEC3` records) and for a child zone to request an update to their `DS` record (`CDNSKEY` and `CDS` records). As these are not integral to our study, we do not discuss them in detail.

example.org A RRSIG). Each RRSIG is created using the private key that matches a public key in DNSKEY records.

DS (Delegation Signer) records, which are essentially hashes of DNSKEYs. These records are uploaded to the parent zone, which establishes the chain of trust reaching up to the root zone. The DS records in the parent zone are authenticated using RRSIGs, just like any other record type.

Most Internet hosts do not do iterative DNS lookups themselves, but instead are configured to use a local DNS *resolver*. When a host wishes to look up a domain name, it sends a query to its resolver; the resolver then iteratively determines the authoritative name server for that domain and obtains the record. If the resolver supports DNSSEC, it will also fetch all DNSSEC records (DNSKEYs and RRSIGs) necessary to validate the record. Finally, the resolver returns the (validated) record back to the requesting host. It is important to note that resolvers make heavy use of caching, and will typically avoid re-requesting any unexpired records that have already been obtained.

DNSSEC is designed to be backwards-compatible, while enabling resolvers who support DNSSEC to specifically request DNSSEC records. A resolver indicates that it would like DNSSEC records by setting the DO ("DNSSEC OK") bit in its DNS request. If the responding authoritative name server has RRSIGs corresponding to the record type of the request, it is obligated to include them. Should the resolver also need DNSKEYs to validate the record, it may need to request them separately.

DNSSEC keys Unlike other common PKIs (e.g., the SSL/TLS PKI [34]), each zone in DNSSEC typically has *two* public/private key pairs: one called a Key Signing Key (KSK) and another called a Zone Signing Key (ZSK). Typically, the KSK is used only to produce RRSIGs for DNSKEY records (hence the name). In contrast, the ZSK is used to produce the RRSIGs for all other record types.

There is no key revocation (apart from root authorities) in the DNSSEC PKI[4]; rather, to mitigate potential effects of key compromise, ZSKs are intended to be *rolled over* (i.e., replaced) daily or weekly, and the KSKs monthly or yearly (the intention is that the KSK can be stored separately from, and in a safer location than, the ZSK). In fact, RFC 6781, which is the best current practice document for DNSSEC management, recommends rolling over KSKs every 12 months and ZSKs even more frequently [30].

[4]If the current DNSKEYs are suspected of being compromised, a zone administrator can replace existing DNSKEYs by following an emergency key rollover process [30].

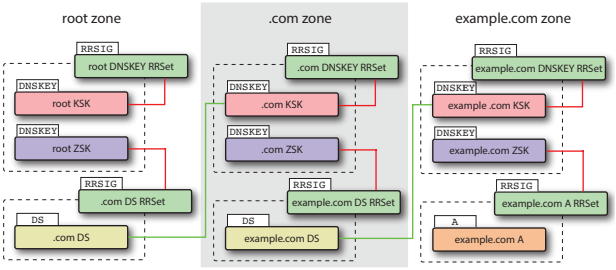

Figure 1: Overview of DNSSEC records necessary to validate example.com's A record. Each RRSIG is the signature of a record set (dashed lines) verified with a DNSKEY (red lines). Each DS record is the hash of a child zone's KSK (green lines).

Validating a DNSSEC record The DNSSEC PKI is rooted at the KSK of the DNS root zone. This KSK is well-known by DNSSEC-aware resolvers. Validating a DNS response starts at the root and continues down the DNS hierarchy: A resolver begins by using the KSK to validate the root DNSKEY RRSIG, which validates the root zone's ZSK. The resolver can then validate the child zone's DS record (and thereby the child zone's KSK) using the RRSIG for the DS records in the root zone, as this is signed with the root zone's ZSK. This process continues until the record in question is authenticated. Figure 1 shows example records and how they are related.

3 Related Work

In this section, we discuss related studies of the DNSSEC ecosystem, covering both server- (DNSSEC domain) and client-side (DNS resolver) studies.

DNSSEC domain deployment As the DNSSEC deployment has grown, researchers and industrial practitioners have examined the DNSSEC ecosystem. The Internet Society publishes periodic DNSSEC deployment reports [47], the most recent of which found that 89% of generic TLDs (gTLDs) and 47% of country-code TLDs (ccTLDs) are signed. They also report that the major authoritative DNS server software and libraries support DNSSEC. Web-based debugging tools such as the DNSSEC Debugger [15] and DNSViz [18] can help administrators verify correct DNSSEC deployment.

Several studies examined the early deployment of DNSSEC by monitoring the availability, verifiability, and validity of domains [40,41,54]. For example, Deccio et al. [12, 13] studied the misconfiguration of DNSSEC domains by surveying approximately 2,000 domains for five months, finding that 20% of zones where RRSIGs had expired at least once would experience another expiration three or more times. Adrichem et al. [49] analyzed a sample of second-level domains on one day and found that 4% exhibited misconfigurations. Similarly, Dai et

al. [14] found that 19.46% of second-level domains from the Alexa Top-1M had an invalid chain of trust from the root.

Our study extends these prior works in three ways. *First*, we examine **all** DNSSEC-enabled domains in three of the largest TLDs, rather than a sample. *Second*, we examine 21 months of DNSSEC behavior, allowing us to investigate temporal trends. *Third*, we examine more types of misconfigurations, including those that require longitudinal data to study (e.g., key rotation behavior).

DNS resolvers Researchers have also studied whether resolvers request and validate DNSSEC records. In general, this is challenging because most resolvers do not allow arbitrary clients to initiate queries. Prior work typically uses one the following techniques to address this limitation:

Passive techniques A number of studies rely on logs from authoritative DNS servers. Guðmundsson et al. [23] measured the deployment of DNSSEC-enabled resolvers using traces of DNS queries made to the `.org` servers. Similarly, Fukuda et al. [21] used snapshots of the `.jp` ccTLD authoritative name server to measure the portion of resolvers requesting `DS` and `DNSKEY` records; they found that 50% of the resolvers requested such records. These studies provide a glimpse into resolvers' DNSSEC queries, but not what they do with the responses. As we show later in this paper, many resolvers do not bother to validate responses.

Active techniques Other approaches issue DNS queries from clients deployed in large numbers of networks, e.g., using dedicated hardware (e.g., RIPE Atlas devices [45]) or software running on web clients (e.g., Java applets [27]). Alternatively, Yi et al. [55] deployed a middlebox that intentionally removed `RRSIG`s from DNS responses to investigate resolver behavior. Unfortunately, these approaches are limited by the coverage of a given deployment platform and user adoption model.

Recent work shows that advertisements embedded in webpages [7, 33, 47] can enable DNS resolver measurements at scale. This approach places ads that cause clients to make HTTP requests to a domain used for testing purposes. Using this methodology, Lian et al. [33] showed that 1% of clients could not resolve DNSSEC-enabled domains at all, while only 3% of clients successfully detected DNSSEC-signed domains with broken signatures. Similarly, APNIC Labs recently reported that the DNSSEC validation rate is increasing, particularly in Africa (16.58%) and Asia (10.17%), due to users' reliance on Google's public DNS service [7].

While the ad-based approach can quickly cover large numbers of resolvers, it gives researchers relatively little control over client selection. This makes it difficult to run multiple experiments using the same client (and their associated resolver) and to understand DNSSEC behavior of resolvers in depth, and makes it difficult to disambiguate lookup failures due to other factors (e.g., loss of connectivity). In Section 5, we address this limitation by using a large-scale platform that enables repeatable measurements of DNSSEC resolvers worldwide.

4 DNSSEC Deployment and Management

We begin our analysis of the DNSSEC PKI by focusing on the deployment and management of DNSSEC records by domains, and how this has changed over time. We perform a longitudinal analysis of nearly 150M domains to answer questions that include: 1) how widely is DNSSEC deployed; 2) when deployed, how often are DNSSEC records correctly published and managed; and 3) how are DNSSEC cryptographic keys managed and maintained, and what is their impact on security? We begin by describing the datasets we use to answer these questions before proceeding with our analysis.

4.1 Datasets

Our goal in this section is to conduct a large-scale, longitudinal, and detailed study of DNSSEC adoption and deployment at authoritative DNS servers. To cover a large number of registered domains, we investigate those listed in zone files for the `.com`, `.net`, and `.org` TLDs. While this does not cover all domains, the approximately 150M domains that we study cover 64% of the Alexa Top-1M and 75% of the Alexa Top-1K websites. To understand how DNSSEC adoption and management changes over time, we use snapshots of DNSSEC records covering nearly two years. Finally, we conduct hourly snapshots of a subset of domains to provide a detailed view of management over shorter timescales.

Taken together, our dataset represents the largest and most comprehensive known set of DNSSEC observations of authoritative DNS servers.

Daily scans Our dataset includes measurements from OpenINTEL [43,52], a project that conducts daily crawls of DNS records for a large number of domains listed in TLD zone files. OpenINTEL first obtains daily snapshots of the `.com`, `.net`, and `.org` TLD zone files, which contain the Name Server (`NS`) and Delegation Signer (`DS`) records for an average of over 147M second-level domains. For each of these, OpenINTEL also collects responses from the authoritative name server for `SOA`, `DNSKEY` records, and the corresponding `RRSIG` records.[5] The daily snapshots span 21 months (between March 1st,

[5]This dataset contains only records for domains whose authoritative name server responded to a query; on average, the name servers for 9% of domains failed to respond to any queries.

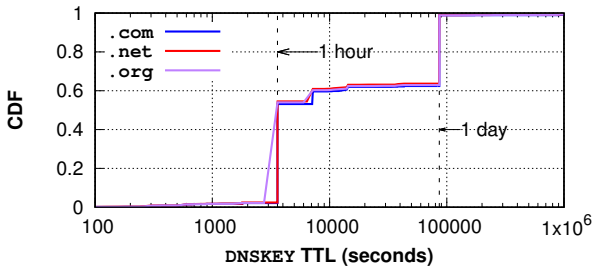

Figure 2: CDF of the TTL of DNSKEY records for .com, .org, and .net second-level domains. Note that 97.9% of TTL values are greater than or equal to one hour and 36.4% of TTL values are greater than or equal to one day (86,400 seconds).

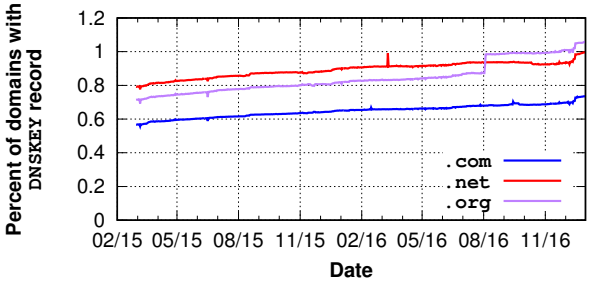

Figure 3: The percentage of all .com, .org, and .net second-level domains that have a DNSKEY record, from the Daily dataset. Between 0.75% and 1.0% of all domains publish a DNSKEY record at the time of writing.

2015 and December 31st, 2016); we refer to this as the Daily dataset.

Hourly scans The Daily dataset is sufficient for studying DNSSEC behavior at coarse granularity, but cannot capture dynamics at timescales shorter than one day. For example, consider the case of replacing DNSKEY records. Figure 2 depicts the cumulative distribution of TTL values for DNSKEY records across the entire Daily dataset. The figure shows that more than 63% of records have a TTL of less than one day, meaning the daily scans can potentially miss a large fraction of key replacement operations.

To address this limitation, we collect a second dataset using *hourly* queries, based on the observation that 97% of observed domains have a TTL of one hour or more. For efficiency, we focus only on domains that have a DS record in the TLD zone (i.e., domains that *may* have correctly deployed DNSSEC, as a DS record is a necessary but not sufficient condition for validity). Specifically, once per hour we collected the DNSKEY and corresponding RRSIG records for all second-level domains (an average of 708K domains) in the .com and .org TLDs, between September 29th and December 31st, 2016.[6] We refer to this dataset as the Hourly dataset.

4.2 DNSSEC Prevalence

We begin by examining how support for DNSSEC has evolved over time. Specifically, we focus on the number of second-level domains that publish at least one DNSKEY record according to the Daily dataset. Note that having a DNSKEY record published does not by itself imply that the domain has correctly deployed DNSSEC; there could be other missing records or invalid signatures. We examine the prevalence of *correct* DNSSEC deployment later in the paper.

[6]We did not collect hourly scans of .net domains as we did not have access to the hourly snapshots of the .net zone file.

Figure 3 plots the fraction of .com, .net, and .org second-level domains that publish at least one DNSKEY record. One key observation is that DNSSEC deployment is rare: between 0.6% (.com) and 1.0% (.org) of domains have DNSKEY records published in our latest snapshot. The fraction of domains that have DNSKEYs is, however, steadily growing. For example, for .org, the fraction rose from 0.75% in March 2015 to over 1.0% in December 2016, even though the number of second-level domains in these TLDs is growing as well (e.g., the .com TLD grew from 116M domains to 125M domains during the same time period).

We observe that large portions of the growth in DNSSEC deployment are due to a small number of steep increases in domains with DNSKEY records. Investigating this trend further, we found that these "spikes" were due to actions by a few authoritative name servers. For example, the authoritative server hyp.net enabled DNSSEC for 11,026 domains in the .org TLD between July 21st and August 5th, 2016, which explains the "spike" in the .org line in Figure 3. In addition, starting on December 16th, 2016, a significant number of new domains enabled DNSSEC, all of which used domainnameshop.com as their authoritative name server.

This observation suggests that a small number of authoritative name servers are responsible for most of the growth in DNSSEC deployment. Thus, incentivizing authoritative name server operators to deploy DNSSEC may end up having a large impact on future growth. For example, the .nl and .se ccTLDs incentivize second-level domains to deploy DNSSEC by offering lower registration costs; these second-level domains are tested every day by the registry to ensure they have correct DNSKEYs, RRSIGs, and DS records [37]. Both TLDs show significantly higher levels of DNSSEC deployment than the TLDs we study (47% [39] and 14% [2], respectively).

Next, we explore whether popular domains are more likely to have deployed DNSSEC. Figure 4 shows the

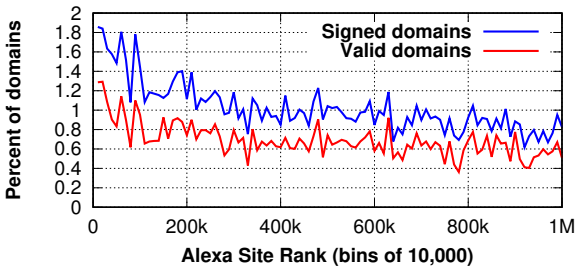

Figure 4: Percentage of domains publishing DNSKEYs as a function of website popularity. Even among the most popular domains, deployment is no more that 1.85% of domains.

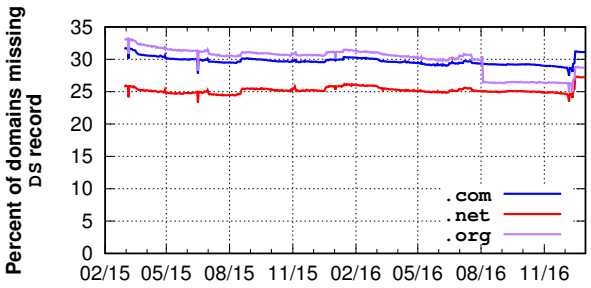

Figure 5: The percentage of signed domains that fail to publish a DS record in the parent zone. Approximately 30% of signed domains fail to do so, meaning they cannot be validated.

percentage of the Alexa Top-1M domains in .com, .org, and .net that publish DNSKEYs, as of December 31st, 2016. We observe that popular websites are more likely to sign their domains, but the overall deployment remains low even among the most popular domains (e.g., the Top-10K sites have a DNSSEC deployment of only 1.85%).

Figure 4 also shows that not all of these domains have correctly deployed DNSSEC; surprisingly, almost 33% of domains that publish DNSKEYs cannot be validated. Next, we explore why so many domains fail to properly deploy DNSSEC. We focus only on the domains that *attempt* to deploy DNSSEC by publishing a DNSKEY record; consistent with prior work [33, 49], we refer to these domains as *signed domains*.

4.3 Missing Records

We now examine whether domains are publishing all necessary DNSSEC records. For this section, we use the Daily dataset, as the Hourly dataset does not cover domains missing DS records. Recall that properly deploying DNSSEC for a domain means that it must have a DS record in the parent zone, DNSKEY records, and RRSIG records for every published record type. We ask what fraction of signed domains properly publish all such records.

DS records Recall from Section 2 that the Delegation Signer (DS) record, which contains a hash of the domain's KSK, is essential to establish a chain of trust from a parent to a child zone. Unlike other DNSSEC record types, DS records are published in the *parent* zone (e.g., .com), along with the domain's NS records. Thus, correctly installing a DS record is often a manual process, where the administrator contacts its registrar and requests that the registrar add a DS record.[7] Domains that fail to upload a correct DS record are not signed by the parent and therefore cannot be validated, even if they provide correct RRSIGs for all of their records.

We begin by examining the percentage of signed domains that fail to upload a DS record using the Daily dataset (Figure 5). We observe that 28%–32% of signed domains do not have a DS record, meaning they cannot be validated. This observation is in line with previous studies [49, 51]; however, prior work has not explored *why* such a large fraction of domains are missing DS records.

To shed light on why, we focus on domains' authoritative name servers. Specifically, we identify the name servers that are authoritative for the largest number of signed domains from our latest snapshot (December 31st, 2016), and calculate the fraction of their domains that are missing a DS record (a name server can be authoritative for multiple domains if they are managed by the same organization). Table 1 shows the results for the top 15 authoritative name servers, which cover 83% of the signed domains we study. We find a highly skewed distribution, with most of the name servers publishing DS

Name servers	Number of domains Signed	w/ DS	DS Publishing Ratio
***.ovh.net**	316,960	315,204	99.45%
***.loopia.se**	131,726	1	0.00%
***.hyp.net**	94,084	93,946	99.85%
***.transip.net**	91,103	91,009	99.90%
***.domainmonster.com**	60,425	4	0.01%
***.anycast.me**	52,381	51,403	98.13%
***.transip.nl**	47,007	46,971	99.92%
***.binero.se**	44,650	17,099	38.30%
***.ns.cloudflare.com**	28,938	17,483	60.42%
***.is.nl**	15,738	11	0.07%
***.pcextreme.nl**	14,967	14,801	98.89%
***.webhostingserver.nl**	14,806	10,655	71.96%
***.registrar-servers.com**	13,115	11,463	87.40%
***.nl**	12,738	12,674	99.50%
***.citynetwork.se**	11,660	13	0.11%

Table 1: Table showing the most popular 15 authoritative name servers, the number of domains with a DS record, and the total number of signed domains for our latest snapshot (December 31st, 2016). The shaded rows represent registrars that fail to publish DS records for nearly all of their domains.

[7] CDNSKEY and CDS can partially reduce the burden of doing manual secure delegation [31] by allowing a domain owner to directly provide the DS record to the registry; unfortunately, we know of no TLDs that currently support CDS or CDNSKEY.

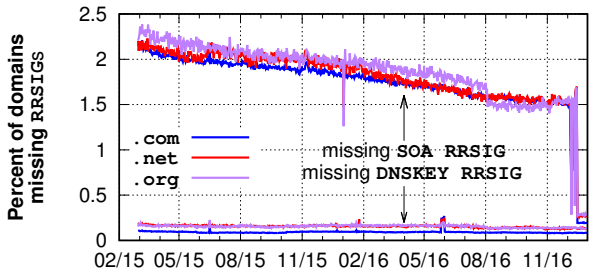

Figure 6: The percentage of signed domains that do not have RRSIGs for SOA and DNSKEY.

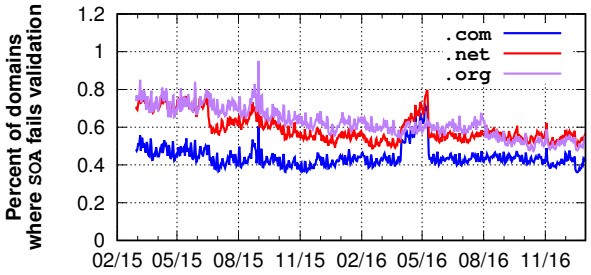

Figure 7: The percent of signed domains for which the RRSIG signatures for the SOA records could not be validated.

records for almost all signed domains, but with four failing to upload a DS record for nearly all of their domains.[8] For example, Loopia (a Swedish hosting provider) is authoritative for more than 131,000 domains that publish DNSKEYs, but only *one* of these domains actually uploads a DS record, which is invalid.[9] Yet again, large hosting providers and outsourced name servers play a significant role in (im)properly maintaining chains of trust.

Returning to Figure 5, we also observe a few dips and spikes in the fraction of domains missing DS records. For example, the drop in the percentage of .org domains with missing DS records in August 2016 was due to a single registrar (hyp.net) publishing 11,026 new signed domains, all with proper DS records (the same set that was observed in Section 4.2). However, the spike in all three TLDs in December 2016 was caused by one hosting provider, Domain Monster, bulk-signing over 37,000 new domains *without* placing the proper DS records.

RRSIG records We next examine the percentage of signed domains that fail to provide RRSIGs for SOA and DNSKEYs using the Daily dataset. Figure 6 presents these results. We find a surprisingly high fraction of missing SOA RRSIGs (1.7%, on average), and a lower fraction of missing DNSKEY RRSIGs (0.2%, on average). We also observe a decreasing trend of missing SOA RRSIGs, and find sudden drops occur in all three TLDs in December 2016. These were caused by the same hosting provider, Domain Monster, which not only provided DNSKEYs for over 37,000 domains *without* corresponding DS records, *but also did not sign the SOA*, indicating thorough mismanagement. Domain Monster finally started publishing SOA RRSIGs in December 2016.

[8]Interestingly, three of these hosting providers (loopia.se, citynetwork.se, and domainmonster.com) do not even upload a DS record for their *own* (signed) domains.

[9]We contacted all four of these operators to ask the reason behind this behavior. One administrator said "*Most people do not understand DNS, so imagine the white faces when I mention DNSSEC ... I don't think DNSSEC has a high priority anymore currently in our organization or our customer base.*" *[48]*

4.4 Incorrect Records

Despite substantial mismanagement, a large fraction of domains publish all the DNSSEC records required for validation. However, this alone is not sufficient to properly deploy DNSSEC; the signatures (and timestamps) in those records must be correct (and not expired).

RRSIG signatures We begin by examining the correctness and freshness of RRSIGs records for SOA and DNSKEY records, using only domains in the Daily dataset that provide RRSIG records. As all RRSets except DNSKEY records are signed by the same ZSK, we verify SOA records with ZSKs, and DNSKEY records with KSKs. Figure 7 plots the fraction of domains where RRSIG validation for SOA records fails. We find that nearly 99.5% of them are valid. Similarly, we observe that most DNSKEYs are also valid (omitted from the figure for clarity), indicating a common, correct process for generating the records.

Interestingly, the fraction of domains with valid records in Figure 7 fluctuates substantially over the course of days or weeks. To investigate the root causes, we determine the reason for validation failure using a customized dnspython library, and assign them to one of three categories: *Expired* RRSIGs (i.e., signatures beyond their expiration date), records with *Signature Invalid* RRSIGs (i.e., signatures that do not match the cor-

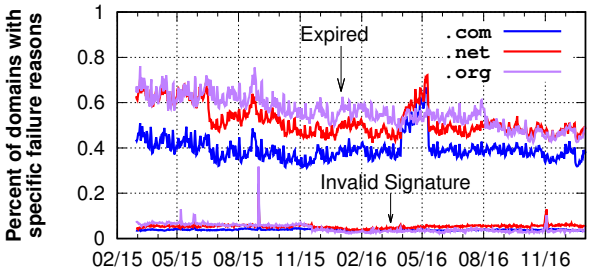

Figure 8: The percent of signed domains with each validation failure type for SOA records.

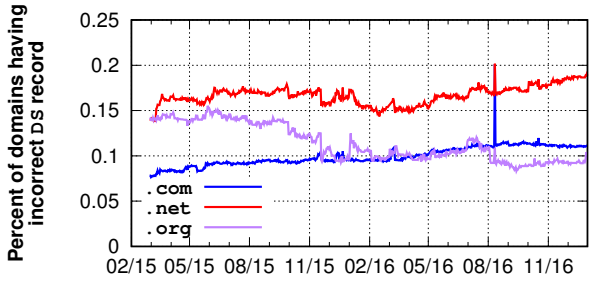

Figure 9: The percent of signed domains having DS records that do not match their KSKs.

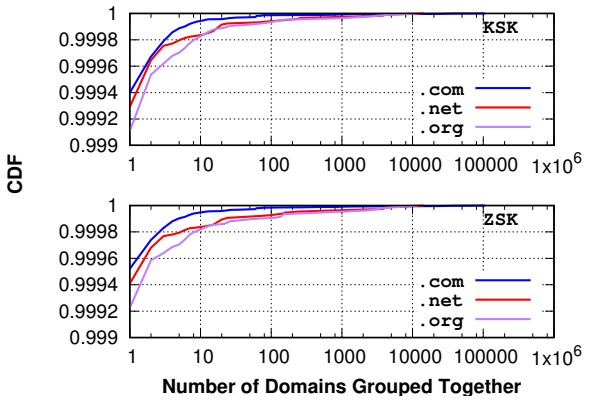

Figure 10: Cumulative distribution of the number of domains grouped by DNSKEYs. The *y* axis starts at 0.999 and both long tails extend to 106,640 domains (.com).

responding DNSKEY), and *Other* reasons (e.g., malformed RRSIGs). We show the fraction of signed domains with the first two failure types for SOA RRSIGs in Figure 8.[10] We find that expired RRSIG records are the primary reason for validation failure. This indicates the need for better automation and auditing of processes for refreshing RRSIG records in DNSSEC.

As one example of this problem, consider the rise in expired signatures in May 2016 for .com and .net. This rise is due to a single registrar: 1,938 .com domains and 254 .net domains, all served by registrar-servers.com, became invalid over this period. This registrar fixed the issue on May 10th, 2016.

Finally, we observe a few intermittent spikes indicated short-lived correlated failures. For example, in September 2015 a total of 1,493 domains with the authoritative name server transip.net published incorrect RRSIGs, a problem that was corrected the following day.

DS records We now examine the correctness of DS records using the Daily dataset. Recall that DS records are basically hashes of KSKs, signed by the parent zone.

Figure 9 shows the results of this analysis. For domains with a DS record, 99.9% of those records are valid (i.e., match the KSK). The spike that occurred in .com and .net in August 2016 was caused by one name server, transip.net, that published incorrect record RRSIGs in September 2016. This name server suddenly changed ZSKs and KSKs for their 381 .com domains and 25 .net domains without switching the DS record, and the problem was corrected the following day.

4.5 Key Management

The previous sections focused on the necessary records for providing valid responses to DNSSEC queries; however, even the best record management practices can result in an insecure system if the cryptographic keys that they rely on are mismanaged. In the next two sections we focus on how administrators manage these keys. In particular, we investigate how often keys are shared across domains (thus increasing the attack surface), how often private keys are weak (e.g., using short keys that can potentially be brute-forced), and whether administrators take the correct steps when rolling over to new keys.

Shared keys In principle, each domain's KSK and ZSK should be unique, as the DS record binds an identity (e.g., a domain) to a KSK, and the ZSK produces RRSIGs for integrity. Otherwise, if the same private key is used for multiple domains, an attacker who steals this key can forge valid DNSSEC records for any of those domains. However, recent work demonstrates that key sharing is common for operational reasons in the SSL/TLS PKI [9]. We thus conducted a study to determine if similar practices occur with DNSSEC keys.

To do so, we extract each domain's DNSKEY record from our latest snapshot (December 31, 2016), and group domains by their KSKs and ZSKs respectively. Figure 10 shows the cumulative distribution of the number of the domains using each ZSK and KSK. We find that 99.95% of keys are used by only one domain. However, this common behavior masks a long tail for key sharing: 384 KSKs (0.04%) and 587 ZSKs (0.05%) are shared by more than one domain, and one KSK and ZSK is shared by over 132,000 domains! Further, we find that ZSK and KSK sharing rates are similar, suggesting that domains sharing ZSKs are highly likely to share KSKs as well.

To understand the key sharing phenomena in more detail, we investigate whether key sharing is mostly explained by a policy from a small set of hosting providers for the affected domains. We first group domains by their authoritative name server, and then group them again by the DNSKEYs in our latest snapshot.

[10]The results for DNSKEY RRSIGs are similar, and omitted for brevity. Furthermore, less than 0.0006% of domains fail to validate for *Other* reasons, and are similarly omitted.

Name servers	KSK		ZSK	
	Domains	Keys	Domains	Keys
***.others**	151,733	157,533	152,144	188,482
***.ovh.net**	316,888	318,036	316,887	326,011
***.loopia.se**	133,258	199	133,258	217
***.hyp.net**	94,888	119,150	94,885	119,161
***.transip.net**	93,819	93,774	93,818	187,129
***.domainmonster.com**	60,984	60,991	60,984	121,939
***.anycast.me**	55,936	56,075	55,936	58,296
***.transip.nl**	45,676	45,648	45,675	91,161
***.binero.se**	44,963	49	44,963	54
***.ns.cloudflare.com**	28,469	239	28,469	214
***.nl**	12,837	12,834	12,836	25,512
***.pcextreme.nl**	15,210	15,192	15,210	28,654
***.webhostingserver.nl**	15,023	15,019	15,023	22,741
***.registrar-servers.com**	13,183	13,043	13,181	12,998
***.is.nl**	11,945	11,978	11,945	23,790
***.citynetwork.se**	11,702	21	11,702	28

Table 2: Table showing the most popular 15 authoritative name servers, the number of domains they manage, and the number of unique DNSKEYs for these domains. The shaded rows represent registrars that share the same DNSKEY across most of their domains.

Table 2 shows the most popular 15 authoritative name servers, their total number of domains, and their total number of DNSKEYs. Similar to the previous section, we find highly bimodal behavior, with most name servers having a low prevalence of shared DNSKEYs, but with a few popular name servers using shared DNSKEYs for nearly all of the domains for which they are authoritative. Of course, key sharing may make sense from an operational perspective (easier management) and from a domain ownership perspective (multiple domains owned by the same company). However, key sharing across domains belonging to different companies for efficiency can substantially increase security risks, e.g., when a single shared key is compromised or cracked this affects all domains that share that key.

Weak keys Next, we examine how often weak keys are used in DNSSEC. The National Institute of Standards and Technology (NIST) recommended against the use of 1024-bit keys after December 31, 2013, as rapid advances in computational power and cloud computing make it easier to break 1024-bit keys [1]. Correspondingly, the Certificate Authority/Browser Forum [11] announced that 1024-bit RSA keys should no longer be supported for SSL certificates or code signing [38]. Further, a recent study showed that 66% of DNSKEYs obtained from the Alexa Top-1M domains can be factored due to their short length [56].

While there is no standard minimum key length for DNSSEC, we adopt the NIST recommendations, and thus define *weak* keys as ones meeting any of the following conditions: (1) RSA keys with a length less than or equal to 1024 bits, (2) DSA keys with a length less than or equal to 2048 bits, or (3) elliptic curve keys with a length less than or equal to 160 bits [1].

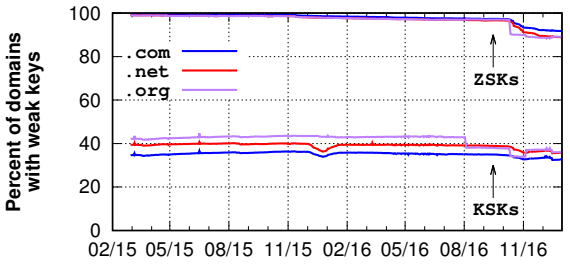

Figure 11: The percentage of domains with weak ZSKs and KSKs. Most keys are weak by NIST standards, even today.

Figure 11 shows the percentage of weak ZSKs and KSKs each day, using the Daily dataset. First, we observe that *the vast majority (91.7%) of ZSKs are weak*, with most being 1024-bit RSA keys. Interestingly, the same trend holds true, but to a lesser extent, for KSKs: over one-third of KSKs are weak keys. Second, there is a small trend towards using stronger keys over time; for example, the fraction of weak ZSK keys began to decline in October 2016. Regardless, the large fraction of weak keys—coupled with the key sharing that we observed in the previous section—further underscores mismanagement of private key material, which can severely weaken security in the DNSSEC PKI.

4.6 Key Rollover

As with any PKI, DNSSEC provides a way for entities to change their public/private key pairs. This process, called key *rollover*, is a recommended best practice in the DNSSEC RFCs [29, 30], and the use of two-key pairs (KSKs and ZSKs) is designed to facilitate frequent rollover.

KSK rollover Rolling over a KSK involves publishing a new DNSKEY and updating the DS in the parent zone.

Unlike many other PKIs, DNSSEC must address issues raised by DNS record caching when considering key rollovers. Recall that all DNS responses contain a TTL field indicating how long a given record can be cached; for efficiency, these TTL values are often on the order of hours to days (see Figure 2). Thus, a domain must carefully manage records during key rollover: if a domain conducts an *Abrupt* rollover (simply publishing a new KSK and DS record), old cached RRSIGs and DS records can cause record validation to fail for clients.

The DNSSEC RFC specifies two schemes by which a domain can roll over their KSK to mitigate this problem: *Double Signature* and *Double DS*. As we do not observe *any* domains using *Double DS*, we focus only on the *Double Signature* scheme. To roll over a KSK using the *Double Signature* scheme, a domain first publishes a new KSK alongside the old KSK, and uses the new

Scheme	.com	.net	.org
No KSK rollovers	621,213	93,558	65,704
Abrupt	17,724	3,183	1,710
Double Signature	219,547	46,092	32,206

Table 3: Distribution of KSK rollover schemes for all domains for each TLD. We do not observe any KSK rollovers for roughly 70% of domains; for the 320,462 domains where we do see a rollover, we observe that 7.0% conduct rollovers that may cause their domains to fail validation for some clients.

KSK to sign additional DNSKEY RRSIGs. At this point, there are *two* KSKs and DNSKEY RRSIGs published. The domain then uploads the new DS record to the parent zone. The domain removes the old DNSKEY and DNSKEY RRSIGs only after the DNSKEY record TTL has expired. By doing so, the domain ensures that all clients will be able to validate the domain, regardless of whether they have cached records.

Table 3 shows the inferred KSK rollover schemes for all domains we measured. We observe that over 70% of domains do *zero* KSK rollovers during our 21 month study period.[11] For those that perform a rollover their KSK, we observe that over 7% of the domains do *Abrupt* rollovers (i.e., simply switching out their keys and DS records without regard for caching effects). These domains may become unavailable during rollover due to failed validation. We also find that between 46% and 50% of the domains that did not rollover their keys have weak keys, underscoring the urgent need for them to quickly perform a rollover their keys to stronger versions.

ZSK rollover We now turn to examine rollovers of ZSKs. Unlike KSK rollovers, a domain need not involve its registrar; the ZSK rollover can be done unilaterally by the domain itself. However, conducting an *Abrupt* rollover can still lead to validation failures, so the DNSSEC RFC defines two schemes for domains to safely roll over their ZSK:

Pre-Publish Under the *Pre-Publish* scheme, a domain publishes a new ZSK DNSKEY, but still uses the old key to sign the RRSIGs (e.g., for A records). After waiting until the TTL of the old DNSKEY expires, the domain then uses the new ZSK to sign the RRSIGs, but continues to publish the old DNSKEY. In this way, cached RRSIGs created with the old key can still be verified. After the maximum TTL of any record in the zone elapses, the old key is no longer published.

Double Signature The *Double Signature* scheme works similarly to the KSK scheme: a new ZSK DNSKEY is introduced, and is used to sign additional RRSIGs immediately. As a result, there are two RRSIGs for each record type: one is signed by the old key, and the other is signed by the new key. After the maximum TTL of any record in the zone, the old key and its RRSIGs are removed.

[11] These results align with a recent report [53] that showed 55% *of TLDs* had not rolled over their KSKs for 22 months.

Scheme	.com	.org
No ZSK rollovers	279,935	27,166
Abrupt	5,527	66
Double Signature	58,807	9,615
Pre-Publish	259,327	33,518

Table 4: Distribution of ZSK rollover schemes for all domains for each TLD. We do not observe any ZSK rollovers for roughly 45% of domains; for the 366,718 domains where we do see roll over, we observe that 1.5% conduct rollovers that may cause their domains to fail validation for some clients.

When detecting the different ZSK rollover schemes, we face a significant challenge: the Daily scans have a resolution of only 24 hours. Thus, we may not observe the rollover behavior of domains that use TTL values of less than 24 hours.[12] Instead, we use the Hourly dataset, which covers nearly all domains (only 2.1% of domains were observed using TTL values smaller than 1 hour).

Table 4 presents the results of our analysis for each TLD. We first observe that both the *Double Signature* and *Pre-Publish* schemes are used, but that the *Pre-Publish* scheme is more popular by a significant margin. However, as with the KSK rollovers, we observe a non-trivial fraction (1.5%) of domains abruptly changing their keys, even though this may lead to validation failures for clients. The lower frequency of *Abrupt* ZSK rollovers may be due to the fact that ZSK rollovers are done entirely by the domain itself, whereas KSK rollovers require coordination with the parent zone.

4.7 Superfluous Signatures

Each DNSKEY RRSIG must be verified by the domain's KSK; but, we find that a large fraction of domains (676,104, or 61% of domains in the December 31, 2016 snapshot) sign their DNSKEY record *twice*: once with the KSK (as expected), and once with the ZSK (which is not used in validation). When focusing only on domains having a corresponding DS record, we find that 644,797 domains (83.6%) exhibit this behavior. While this does not inhibit validation (assuming a valid KSK signature), it does increase the size of DNSKEY packets significantly. When using strong keys (e.g., 2048-bit RSA), this behavior can lead to avoidable DNSKEY packet fragmentation, which not only makes domain resolution inefficient [50], but also makes DNSSEC vulnerable to poisoning attacks when resolvers do not validate responses [24]. As we show in the next section, the vast majority of the resolvers we studied request DNSSEC records but do not

[12] This was not a problem for detecting KSK rollover schemes, as all TLDs we study use TTL values of at least one day.

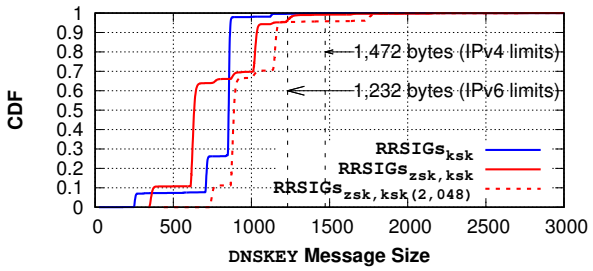

Figure 12: DNSKEY message size for all domains with a DS record. Packets are fragmented when the message size exceeds 1,472 and 1,232 bytes for IPv4 and IPv6, respectively (assuming an MTU of 1,500 bytes).

validate such responses.

Fragmentation does not impact a majority of domains today as the main reason is that relatively short (1024-bit RSA) keys are used. Ironically, if operators were to improve the security by using longer keys, substantial numbers of domains may become vulnerable to poisoning. Figure 12 shows the cumulative distribution of estimated DNSKEY packet size per domain when using 1024-bit and 2048-bit keys, and when using only KSK signatures or using both KSK and the unnecessary ZSK signatures in the DNSKEY record.

The figure shows that records today rarely incur IPv4 fragmentation; only 403 (0.01%) of the domains that sign their DNSKEY with just the KSK, and 5,568 (0.8%) of the domains that use both the KSK and ZSK cause fragmentation. Of these latter domains, 3,380 (60.7%) could have avoided fragmentation by using only a KSK signature in the DNSKEY record. Increasing the key size to 2048 bits does not substantially increase fragmentation for those that use only KSK signatures (only 10 additional domains are affected); however, for those that use superfluous signatures, 30,914 (4.6%) of their responses will be fragmented—more than five times as many cases as today.

This behavior is likely due to misconfigured DNS software. For example BIND [17] and Windows Server 2012 [16] both generate DNSKEY RRSIGs using both the KSK and ZSK by default. PowerDNS [44], and OpenDNSSEC [42] correctly generate DNSKEY RRSIGs only with the KSK.

4.8 Summary

We found that DNSSEC deployment is rare but increasing, and nearly a third of DNSSEC-enabled domains are misconfigured in ways that defeat security by providing records that cannot be validated. The latter is primarily caused by a small number of popular hosting providers and registrars that fail to provide DS records, use expired RRSIGs, etc. We also found that almost all ZSKs and one-third of KSKs are weak by NIST standards, that a few hosting providers use the same DNSKEYs for almost all of the domains for which they are authoritative, and many domains exhibit poor rollover hygiene. These issues undermine the security of DNSSEC regardless of resolver behavior, and highlight the need for improved auditing and automation in DNSSEC management.

5 DNS Resolver Support

Even if domains properly manage their DNSSEC records, a client is not protected unless its resolver requests and validates them properly. We now examine the DNSSEC behavior of resolvers.

5.1 Data Collection Methodology

A challenge when studying the behavior of resolvers is that most will respond only to local clients (i.e., they are not open resolvers). To address this limitation, we use the Luminati proxy network [10] to issue DNS requests.

Hola Unblocker [25] is a system that allows users to route traffic via a large number of proxies, often to evade geofencing of content. The Hola software is available on multiple platforms (e.g., as a stand-alone application on Windows, as cross-platform web browser extensions, and on Android) and has been installed more than 91 million times. *Luminati* [35] is a paid HTTP/S proxy service that enables clients to route traffic via Hola users' machines.

To route HTTP/S traffic via Luminati, a client first connects to a Luminati server (called the *super proxy*). The super proxy then checks that the destination domain is valid (via Google's DNS service), and then forwards the request to a Hola client (called the *exit node*). The exit node then makes a DNS request for the destination domain, makes the HTTP/S request, and returns the response back via the super proxy. The super proxy annotates the response with a unique identifier for the exit node that made the request, called the *zID*. An overview is shown in Figure 13; more details about using Luminati for network measurement experiments are provided by Chung et al. [10].

Luminati allows clients to choose the exit node that will forward traffic via two mechanisms. *First*, the client is allowed to select the country where the exit node is located. *Second*, the client can repeatedly send requests via the same exit node by specifying a *session number*; Luminati will continue to use the same exit node as long as the node remains alive and no errors are encountered. This functionality allows us to conduct multiple experiments using the same exit node.

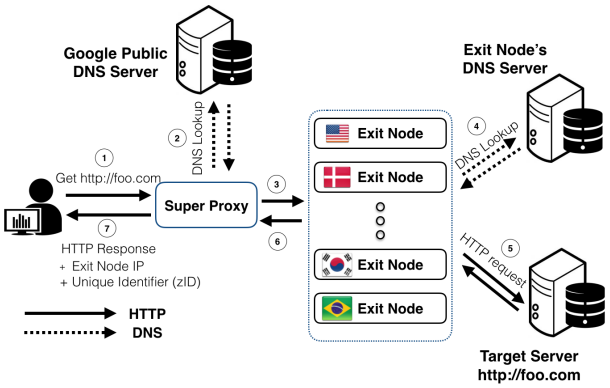

Figure 13: Timeline of a request in Luminati: the client connects to the super proxy and makes a request ①; the super proxy makes a DNS request ② and forwards the request to the exit node ③; the exit node makes a DNS request ④, then requests the HTTP content ⑤. The response is then returned to the super proxy ⑥, then to the client ⑦.

Ethics To conduct these experiments, we paid the operators of Luminati for access, and we abided by their terms of service. The owners of exit nodes agreed to route Luminati traffic through their hosts in exchange for free service.[13] Users can opt out by subscribing to Hola (for a fee) or uninstalling the software. We took great care to make sure that our experiments would not harm users, by sending only a small amount of traffic and by not visiting any potentially sensitive domains. For the latter, we mitigated any potential harm to operators of exit nodes by generating traffic only toward domains that we own, which are hosted in our university testbed and serve empty web pages.

5.2 Experimental Setup

Our broad goal in this section is to understand the DNSSEC behavior of resolvers. For these experiments, we built an authoritative DNS server and web server for a testbed domain under our control. Our testbed domain (a second-level domain) fully supports DNSSEC functionality with a chain of trust by uploading its `DS` record to the `.com` zone.

Domain configuration One of our goals is to examine whether DNSSEC resolvers properly validate DNSSEC records. To do so, we configured our DNS server with 10 different subdomains, each of which simulates a different kind of DNSSEC misconfiguration, along with a single *valid* zone. These misconfigurations include missing, incorrect, and expired `RRSIG`s, missing `DNSKEY`s, incorrect `DS` records, *etc*.

For each exit node we test, we generate a unique identifier for that node's DNS requests (e.g., `http://`id_1`.invalid-rr-sig.example.com`). This approach allows us to easily map incoming DNS and HTTP requests to specific exit nodes, and to avoid any potential caching issues at intermediate resolvers. To implement this, we created a custom DNS server that generated `DNSKEY`s, `DS` records, and `RRSIG`s on-the-fly.

Experimental configuration At first glance, measuring whether a resolver supports DNSSEC seems trivial. We configure our server to respond to queries with misconfigured DNSSEC `RRSIG`s, which should be dropped by validating resolvers. If the exit node successfully retrieves the web page, then we know that the exit node's resolver did not provide DNSSEC security.

In practice, implementing this experiment correctly is not so simple. First, Luminati's super proxy checks that the requested domain name is valid using a Google resolver (which does DNSSEC validation) *before* forwarding the request to the exit node. Thus, a simple request for a misconfigured record would be rejected by the super proxy and not forwarded to an exit node. Second, if an exit node's resolver correctly rejects a misconfigured DNSSEC response, it will respond to the exit node with a `SERVFAIL` message. In this case, the super proxy will return an error message to our measurement client and invalidate our session identifier (i.e., we can no longer send requests via that exit node using the identifier). Third, a request may fail for reasons other than DNSSEC validation (e.g., due to network failure at the exit node), so we must develop techniques to disambiguate such cases.

We address these issues as follows:

1. We configure our DNS server to always return a valid response if the request comes from Google's DNS service, to ensure that the super proxy forwards the request to the exit node.[14]
2. Each exit node first fetches a *valid* record for a name with a unique identifier. We record the identifier, the *zID* in the super proxy's response, the IP address of the exit node's resolver (from the incoming DNS request), and the IP address of the exit node (from the incoming web request).
3. If the incoming DNS request from the exit node's resolver does not set the `DO` bit, the resolver does not support DNSSEC and we continue to test a different exit node.
4. Otherwise, we iteratively request each of our 9 misconfigured records from the same exit node. If we receive a response from the super proxy with the same *zID*, the exit node's resolver did not validate the DNSSEC record.

[13] `https://hola.org/legal/sla`

[14] Note that we test Google's DNS resolver, as well as other open resolvers, outside of Luminati.

Country	Hosting ISP	DNS Resolvers	Exit Nodes
Indonesia	PT Telekomunikasi	1,319	2,695
U.S.	Level 3 Communications	522	79,303
U.S.	Time Warner Cable Internet	148	1,133
Germany	Deutsche Telekom AG	104	2,682
Canada	Bell Canada	89	1,120
U.K.	TalkTalk Communications	76	878
U.K.	Sky UK Limited	74	1,535
U.S.	Frontier Communications	63	241
China	China Telecom	56	344
Canada	Rogers Cable Communications	49	1,250
Spain	Telefonica de Espana	48	1,982
U.S.	Charter Communications	46	355
Austria	Liberty Global Operations	40	10,554
U.S.	SoftLayer Technologies	37	2,559
Czech	AVAST Software s.r.o.	33	2,731

Table 5: The top 15 ISPs in terms of the number of DNS resolvers that do not validate our DNSSEC response. Level 3 (shaded) has 522 resolvers that do not validate the DNSSEC response, while six do (not shown).

5. If the measurement client receives an error (or a response from the super proxy with a different *zID*), it means that the exit node's resolver *may* successfully validate DNSSEC responses (but the error could have been for other reasons). To rule out transient failures unrelated to DNSSEC validation, we repeatedly test each resolver (by finding more exit nodes that use it), and only consider those we test at least 10 times.

During our experiments, we sometimes observed multiple DNS requests (and even multiple HTTP requests) arriving at our servers for the same unique identifier, sometimes hours after we had concluded our experiment. This behavior is likely due to malware, spyware, or intrusion detection systems [10]. To prevent these from biasing our results, we only consider the DNS request that comes before the first HTTP request that arrives at our web servers.

5.3 Results

We use this methodology to measure a total of 403,355 exit nodes—from 177 countries and 8,842 ASes—over a period of 13 days in early 2017. These exit nodes use a total of 59,513 unique resolvers. We observe that 49,424 of the resolvers (83.0% of resolvers, covering 65.9% of the exit nodes) send requests with the `DO` bit set, suggesting that a majority of resolvers support DNSSEC.

Next, we study whether these resolvers actually *validate* the DNSSEC responses they receive. To do so, we need to filter the data to (a) focus only on exit nodes that are configured with a single resolver (exit nodes that use multiple resolvers make it difficult to identify how the different resolvers behave) and (b) only consider resolvers that we were able to measure 10 times or more.

After filtering, we arrive at 4,427 resolvers whose DNSSEC validation policies we can test. We refer to this set of resolvers that request DNSSEC records as *DNSSEC-aware resolvers*. We classify these resolvers into ones that incorrectly validate DNSSEC records (i.e., more than 90% of the exit nodes received a response when the resolver is given an incorrect `RRSIG`), ones that correctly validate DNSSEC records (i.e., more than 90% of the exit nodes received an error), and ones whose policies are ambiguous (all other cases).

Incorrectly validating resolvers We found that 3,635 resolvers (82.1% of the DNSSEC-aware resolvers) from 146 ASes fail to validate the DNSSEC responses, even though they issue the DNS requests with the `DO` bit set;[15] these resolvers cover 149,373 (78.0%) of the exit nodes covered by DNSSEC-aware resolvers. These resolvers all pay the overhead for DNSSEC responses, but do not bother to validate the results they receive.

Table 5 shows the top 15 ASes whose resolvers do not validate DNSSEC responses. Interestingly, we found that even though six resolvers from Level 3 *do* validate DNSSEC responses, another 522 *do not*, indicating that DNSSEC validation can be different between resolvers in the same AS. Most likely, this variance is due to third-party DNS resolvers that are hosted in the Level 3's network, as we are only classifying resolvers by the AS they lie in.[16]

Correctly validating resolvers Only 543 resolvers (12.2% of the DNSSEC-aware resolvers) from 196 ASes correctly validate DNSSEC responses; these resolvers cover 31,811 (16.6%) of the exit nodes covered by DNSSEC-aware resolvers. We found surprisingly few large ASes that validate DNSSEC responses; the largest ones include Comcast (US), Orange (Poland), Bahnhof Internet AB (Sweden), Free SAS (France), and Earthlink (Iraq). Interestingly, we found that all validating resolvers successfully validate all scenarios; we did not find any resolvers that failed some of our misconfiguration tests but passed others. This is in contrast to client behavior for other PKIs, such as the web [34], where browsers pass different subsets of validation tests.

Validation efficiency A concern for DNSSEC is the overhead it places on resolvers, both to fetch DNSSEC records and to validate signatures. For instance, if an `RRSIG` is invalid due to expiration then a resolver can save time and traffic by withholding requests for the corresponding `DNSKEY` or `DS` record. By investigating

[15] We further verified this behavior by looking for requests for `DNSKEY` and `DS` records that are necessary for validation; in all cases, we did not observe any lookups for these records.

[16] For example, we found similar cases of inconsistent validation in ARNES (Slovenia), Rostelecom (Russia), KDDI (Japan), Stofa (Denmark), Sprint (U.S.), and `hd.net.nz` (New Zealand) as well. Personal communication with the ARNES operators indicated that resolvers with different behavior are managed by different entities (ARNES and Univ. of Ljubljana) [8].

Provider	DO bit	Requested DS	Requested DNSKEY	Validated?
Verisign	✓	✓	✓	✓
Google	✓	✓	✓	✓
DNS.WATCH	✓	✓	✓	✓
DNS Advantage	✓	✓	✓	✓
Norton ConnectSafe	✓	✓	✓	✓
Level3	✓	✗	✗	✗
Comodo Secure DNS	✓	✗	✗	✗
SafeDNS	✓	✗	✗	✗
Dyn	✓	✗	✗	✗
GreenTeamDNS	✓ and ✗	✓	✓	✗
OpenDNS Home	✗	✗	✗	✗
OpenNIC	✗	✗	✗	✗
FreeDNS	✗	✗	✗	✗
Alternate DNS	✗	✗	✗	✗
Yandex.DNS	✗	✗	✗	✗

Table 6: Public DNS services that we tested for DNSSEC validation. Five services (shaded) do not validate DNSSEC responses even though they request the DNSSEC records.

DNSKEY and DS requests arriving at our DNS server, we found that all but four ISPs (Comcast, Orange Polska, O2 Czech Republic, and The Communication Authority of Thailand) make these unnecessary requests when the RRSIG for A is missing.

5.4 Open Resolvers

We investigate the DNSSEC validation behavior for public DNS resolvers using clients outside of Luminati. Table 6 shows 15 public resolvers and their DNSSEC policies. We found five do not request DNSSEC records at all (DO bit not set), and that half of the resolvers that do request DNSSEC records fail to validate the responses. Strangely, when we send a DNS request to *GreenTeamDNS*, our DNS server observes two queries from different resolver IPs: one from GreenTeamDNS without the DO bit, and the other Google with the DO bit (suggesting that they outsource lookups to Google). However, even though Google is known to return a SERVFAIL for the domains with invalid DNSSEC records, the request ultimately succeeds and we (incorrectly) receive a response.

6 Conclusion

This paper presents a longitudinal, end-to-end study of DNSSEC ecosystem—encompassing more than 147M second-level domains and 59K DNS resolvers—to understand the security implications of how DNSSEC is managed. We found that DNSSEC deployment by domain owners is rare but growing, and that nearly one third of all DNSSEC-supporting domains publish records in ways that prevent validation and thus provides no practical security. Further, we found widespread use of weak, shared keys combined with poor rollover hygiene (mostly due to a small number of hosting providers), undermining the protection DNSSEC provides against stolen or factored keys. We used Luminati to measure resolver behavior in 8.8K ASes in 177 countries, and found that while DNSSEC-aware resolvers are common (83%), only 12% of them actually validate responses to provide any practical security benefits. In summary, our study paints a bleak picture of the security provided by the DNSSEC ecosystem, one that has not improved substantially over time. Our findings highlight the need for continuous auditing of DNSSEC deployments and automated processes for correctly and securely managing DNSSEC material.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This research was supported in part by NSF grants CNS-1421444 and CNS-1563320.

References

[1] Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf.

[2] .se (The Internet Infrastructure Foundation). .se Health Status — DNS and DNSSEC. https://www.iis.se/docs/Health-Status-DNS-and-DNSSEC-20120321.pdf.

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

[4] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033, IETF, 2005. http://www.ietf.org/rfc/rfc4033.txt.

[5] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035, IETF, 2005. http://www.ietf.org/rfc/rfc4035.txt.

[6] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034, IETF, 2005. http://www.ietf.org/rfc/rfc4034.txt.

[7] APNIC DNSSEC validation rate. https://stats.labs.apnic.net/dnssec.

[8] Alex Mihićinac, ARNES. Personal Communication.

[9] F. Cangialosi, T. Chung, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. Measurement and Analysis of Private Key Sharing in the HTTPS Ecosystem. *CCS*, 2016.

[10] T. Chung, D. Choffnes, and A. Mislove. Tunneling for Transparency: A Large-Scale Analysis of End-to-End Violations in the Internet. *IMC*, 2016.

[11] Certificate Authority/Browser Forum. `https://cabforum.org`.

[12] C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra. A case for comprehensive DNSSEC monitoring and analysis tools. *SATIN*, 2011.

[13] C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra. Quantifying and improving DNSSEC availability. *ICCCN*, 2011.

[14] T. Dai, H. Shulman, and M. Waidner. DNSSEC Misconfigurations in Popular Domains. *CANS*, 2016.

[15] DNSSEC Debugger. `http://dnssec-debugger.verisignlabs.com`.

[16] DNSSEC in Windows Server 2012. `https://technet.microsoft.com/en-us/library/dn593642(v=ws.11).aspx`.

[17] DNSSEC signzone manual pages. `https://ftp.isc.org/isc/bind9/cur/9.9/doc/arm/man.dnssec-signzone.html`.

[18] DNSViz. `http://dnsviz.net`.

[19] D. Eastlake. Domain Name System Security Extensions. IETF RFC 2535, IETF, 1999.

[20] D. Eastlake and C. Kaufman. Domain Name System Security Extensions. RFC 2065, IETF, 1997.

[21] K. Fukuda, S. Sato, and T. Mitamura. A Technique for Counting DNSSEC Validators. *INFOCOM*, 2013.

[22] Y. Gu. Google Security Blog: Google Public DNS Now Supports DNSSEC Validation. `https://security.googleblog.com/2013/03/google-public-dns-now-supports-dnssec.html`, 2013.

[23] O. Gudmundsson and S. D. Crocker. Observing DNSSEC validation in the wild. *SATIN*, 2011.

[24] A. Herzberg and H. Shulman. Fragmentation Considered Poisonous, or: One-domain-to-rule-them-all.org. *CNS*, 2013.

[25] Hola VPN. `http://hola.org/`.

[26] ICANN TLD DNSSEC Report. `http://stats.research.icann.org/dns/tld_report`.

[27] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating the Edge Network. *IMC*, 2010.

[28] D. Kaminsky. It's the End of the Cache as We Know It. Black Hat, 2008. `https://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-Kaminsky/BlackHat-Japan-08-Kaminsky-DNS08-BlackOps.pdf`.

[29] O. Kolkman, R. Gieben, and N. Labs. DNSSEC Operational Practices. RFC 4641, IETF, 2006. `http://www.ietf.org/rfc/rfc4641.txt`.

[30] O. Kolkman, W. Mekking, N. Labs, R. Gieben, and S. Labs. DNSSEC Operational Practices, Version 2. RFC 6781, IETF, 2012.

[31] W. Kumari, O. Gudmundsson, and G. Barwood. Automating DNSSEC Delegation Trust Maintenance. RFC 7344, IETF, 2014.

[32] J. Livingood. Comcast Voices: Comcast Completes DNSSEC Deployment. `http://corporate.comcast.com/comcast-voices/comcast-completes-dnssec-deployment`, 2012.

[33] W. Lian, E. Rescorla, H. Shacham, and Stefan. Measuring the Practical Impact of DNSSEC Deploymenta. *USENIX Security*, 2013.

[34] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, A. Schulman, and C. Wilson. An End-to-End Measurement of Certificate Revocation in the Web's PKI. *IMC*, 2015.

[35] Luminati. `https://luminati.io/`.

[36] P. Mockapetris. Domain Names - Concepts and Facilities. RFC 1034, IETF, 1987.

[37] R. Mohan. Slowly cracking the DNSSEC code at ICANN 43. `https://afilias.info/blogs/ram-mohan/slowly-cracking-dnssec-code-icann-43`.

[38] Phasing out Certificates with 1024-bit RSA Keys. `https://blog.mozilla.org/security/2014/09/08/phasing-out-certificates-with-1024-bit-rsa-keys/`.

[39] Netherlands Knowledge platform DNSSEC. https://www.dnssec.nl/home.html.

[40] E. Osterweil, D. Massey, and L. Zhang. Deploying and monitoring DNS security (DNSSEC). *ACSAC*, IEEE Computer Society, 2009.

[41] E. Osterweil, M. Ryan, D. Massey, and L. Zhang. Quantifying the operational status of the DNSSEC deployment. *IMC*, 2008.

[42] OpenDNSSEC - making DNSSEC easy for DNS administrators. http://manpages.ubuntu.com/manpages/precise/man7/opendnssec.7.html.

[43] OpenINTEL. https://www.openintel.nl/.

[44] PowerDNS: Serving authoritative DNSSEC data. https://doc.powerdns.com/md/authoritative/dnssec/.

[45] K. Ranjbar. Measuring DNSSEC using RIPE Atlas. CNN, 2014. https://la51.icann.org/en/schedule/wed-dnssec/presentation-dnssec-ripe-atlas-15oct14-en.pdf.

[46] S. Son and V. Shmatikov. The hitchhiker's guide to DNS cache poisoning. *Security and Privacy in Communication Networks*, Springer, 2010.

[47] State of DNSSEC Deployment 2016. https://www.internetsociety.org/sites/default/files/ISOC-State-of-DNSSEC-Deployment-2016-v1.pdf.

[48] Thijs Stuurman, KPN Interned Services. Personal Communication.

[49] N. L. M. van Adrichem, N. Blenn, A. R. Lua, X. Wang, M. Wasif, F. Fatturrahman, and F. A. Kuipers. A measurement study of DNSSEC misconfigurations. *Sec. Info.*, 4(8), 2015.

[50] G. van den Broek, R. van Rijswijk-Deij, A. Sperotto, and A. Pras. DNSSEC meets real world: dealing with unreachability caused by fragmentation. *IEEE Communications*, 52(4), 2014.

[51] R. van Rijswijk-Deij, M. Jonker, and A. Sperotto. On the Adoption of the Elliptic Curve Digital Signature Algorithm (ECDSA) in DNSSEC. *CNSM*, 2016.

[52] R. van Rijswijk-Deij, M. Jonker, A. Sperotto, and A. Pras. A High-Performance, Scalable Infrastructure for Large-Scale Active DNS Measurements. *IEEE Journal on Selected Areas in Communications*, 34(6), 2016.

[53] M. Wander. Measurement Survey of Server-Side DNSSEC Adoption. ICANN, 2016.

[54] H. Yang, E. Osterweil, D. Massey, S. Lu, and L. Zhang. Deploying cryptography in Internet-scale systems: A case study on DNSSEC. *IEEE Transactions on Dependable and Secure Computing*, 8(5), 2011.

[55] Y. Yu, D. Wessels, M. Larson, and L. Zhang. Check-Repeat: A New Method of Measuring DNSSEC Validating Resolvers. *TMA*, 2013.

[56] M. B. Yosef, H. Shulman, M. Waidner, and G. Beniamini. Factoring DNSSEC: Evaluation of Vulnerabilities in Signed Domains. *NSDI*, 2017.