The Sad Story of DNSSEC

Exploring the Real-World Deployment of Domain Name System Security Extensions

Alexander Elliott*
aelliott41@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

John Moxley*
jmoxley@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

ABSTRACT

This paper studies the current deployment status of DNSSEC (Domain Name System Security Extensions [5]), a security extension to the Domain Name System (DNS).

DNSSEC is a suite of extension specifications with the goal of providing cryptographic authentication of DNS records. It has a goal of providing a method of validating that a record retrieved from an authoritative name server is unmodified. This is useful for preventing malicious actors changing records in transit or in compromised DNS servers to direct recipients to an alternate domain where they may steal users' information – or worse. DNSSEC has unfortunately suffered from a low adoption rate with few DNS resolvers and domains implementing the extension.

With this in mind, we explore the following research question: What is the current deployment level of DNSSEC, specifically on the side of the DNS resolver? Previous studies, such as in [10], have shown the level of DNSSEC adoption by sites; our study will add more context to these statistics.

In this study, we query a sample of popular websites to a large dataset of DNS resolvers and examine their responses. we find that the end-to-end deployment of DNSSEC from domains, to top-level-domains, to DNS resolvers remains quite minimal. Only a handful of popular domains implement DNSSEC, and most resolvers do not return valid for these domains.

CCS CONCEPTS

• **Networks** → **Network measurement**.

KEYWORDS

DNS, DNSSEC

ACM Reference Format:
Alexander Elliott and John Moxley. 2023. The Sad Story of DNSSEC: Exploring the Real-World Deployment of Domain Name System Security Extensions. In *Proceedings of (Securing the Internet Infrastructure).* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

*Both authors contributed equally to this research.

Securing the Internet Infrastructure, April 2023, Atlanta, GA, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The DNS serves as a necessary tool for converting domain names into Internet Protocol (IP) addresses. With such a crucial role to play in today's Internet infrastructure, it has unfortunately become the target of many coordinated attacks. One such attack is DNS Hijacking. This can be achieved through several methods including gaining access to a DNS server to update records to point to a malicious IP address, or through overriding a user's machine's chosen DNS server to point to a rogue resolver.

According to the IDC 2021 Global DNS Threat Report [1], 87% of companies have been the victim of at least one DNS-based attack with damages averaging $950k per attack. The popularity of using DNS to launch phishing attacks has exploded in recent years, especially with the significant increase of remote workers. Downtime, loss of business, and theft of information represent only a small fraction of possible damages malicious actors can inflict on both enterprises and users alike.

With the impacts of DNS attacks being so severe, and the security protections offered by DNS being so limited, users need a way to ensure that the IP address they receive from a DNS resolver is for the domain they requested. DNSSEC is an extension built on top of the existing DNS infrastructure that provides users a means to validate potentially-malicious DNS responses.

DNSSEC has suffered from low adoption do to several factors. One reason is that, unlike the RPKI [11] approach to BGP hijacking prevention, the extension must be implemented not only by the domain itself, but by the DNS resolver and all domains in its chain of trust. A top-level domain (TLD), e.g. *.xyz*, must implement DNSSEC for a user to be able to use DNSSEC to validate DNS records for *any* domain under that TLD. Any *one* node in the chain may single-handedly prevent DNS record validation through failing to implement DNSSEC.

In this paper, we explore what the current state of DNSSEC deployment resembles from the perspective of a user by examining responses from a pool of DNS resolvers and organizing resolvers into response-based categories.

2 DNSSEC

DNSSEC was developed in the 1990s following the growing concern regarding the lack of security measures present within core Internet protocols [8]. The original design of the DNS does not include security measures; a DNS resolver cannot verify the response it receives from an authoritative name server to be authentic. Thus, an attacker can trivially spoof a DNS response [8]. This can be used to execute a cache poisoning attack, where a DNS resolver unknowingly accepts a forged DNS reply, caches it, and serves the malicious response for the duration of the cache period. However, despite

being published in 2005, DNSSEC is still not widely adopted [4]. We will investigate the extent to which DNS resolvers implement DNSSEC.

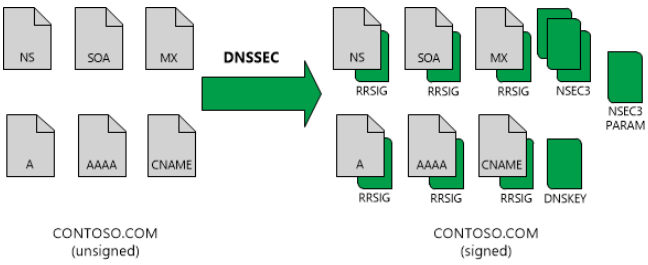


Figure 1: DNSSEC provides additional records, shown in green [3].

DNSSEC is an extension to the existing DNS which provides a mechanism for verifying the response to a DNS query [3]. Additional records are added to allow verification, as is shown in Figure 1. At a high level, DNSSEC verification can be performed by using the DNSKEY record, a public key, to decrypt RRSIG records (which are available for each existing record) and compute hashes which will match for valid records [3]. The public key itself is signed by the parent zone, as is the parent zone's public key by its parent, all the way up to the root zone [8]. This establishes a chain of trust for verifying DNSSEC records, as is shown in Figure 2. There are also other records involved mechanisms for signing child zones and proving the existence of records, but those are outside the scope of our investigation.

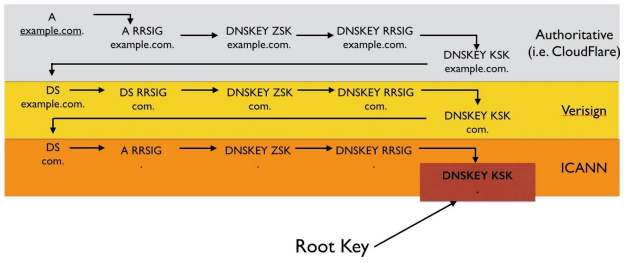


Figure 2: DNSSEC establishes a chain of trust all the way up to the root DNSKEY Key-Signing Key (KSK) [14]

3 DATASETS

In order to run this experiment, we first obtain a source for domains and DNS resolvers to test DNSSEC validity.

For an accurate perspective, we must select a sample of sites that represent a large slice of Internet traffic. To facilitate this, we use the Tranco [13] domain popularity ranking accessed in April, 2023. This is a project that maintains a list of top sites that is resilient against manipulation and fluctuations. The top 100 domains from this list are selected.

Due to the structure of DNSSEC, it is necessary for not only the domain to implement DNSSEC, but also the DNS server, all servers in the chain of trust, and the TLD. With this in mind, we use a large, crowd-sourced pool of 67,177 DNS resolvers from the Trickest DNS resolver dataset[2]. None of these resolvers are guaranteed to be functional or provide responses within a timeout threshold. There exists a separate list within this dataset of trusted resolvers, however, it is currently limited to only 35 resolvers. Taking this limitation into account, we use the large list and eliminate resolvers that fail to provide a query response.

4 METHODOLOGY

To perform our investigation, we develop a multi-process program to rapidly perform queries for top sites to our set of resolvers and store results in a MySQL database.

4.1 Program Details

The crux of our program is a useful tool named `delv` which can be used to validate the DNSSEC chain [7]. It is developed by the Internet Systems Consortum (ISC) and is a successor to the well-known tool `dig`. One benefit of using `delv` is that the mechanisms it uses to validate DNSSEC closely mirror the BIND9 DNS server [7]. Thus, our queries via `delv` will closely mirror how a downstream resolver would verify DNSSEC.

We instrument `delv` by running it against one DNS resolver and site at a time, and parse the output of the tool. Based on the output, we draw conclusions which are categorized in the following way:

- **Valid**. The site has correct DNSSEC results. This resolver correctly implements DNSSEC.
- **Unsigned**. Response does not include an RRSIG. This site does not implement DNSSEC.
- **Broken Chain**. "DNSKEY records don't correspond with the DS record in the parent zone, records are signed with a different key than expected or the DNSKEY is missing entirely [9]." The DNS resolver likely does not properly implement DNSSEC.
- **Timeout** The resolver failed to provide an answer in the provided time frame.
- **Error** The test threw an exception or returned a result other than those listed above.

We make a few key assumptions to facilitate our research. First of all, we assume that the sites that we test will **never** be DNSSEC invalid, which would cause DNSSEC-aware resolvers to return an error. This is reasonable because we are testing the top 100 visited sites, which are highly unlikely to have severe configuration errors which would greatly limit their audience. Note, we do explicitly test `www.dnssec-failed.org`, which is not on the top sites list. This is discussed in detail in Section 5.2.

We also assume that resolvers in our dataset are not implementing any content filtering which would prevent accessing certain sites. However, we are able to identify resolvers which are misbehaving and exclude them from our list. For example, we noticed one resolver returning a valid response for a site which did not implement DNSSEC, because the redirect page it returned instead *was* implementing DNSSEC. We are able to identify a small number of cases like this and remove them from our results.

4.2 Environment

We run our program simultaneously in two separate cloud computing environments. One on an AWS t3.2xlarge VM (8 vCPU, 32 GiB memory, 5 Gib network), and another on an Azure Standard D2s v3 VM (2 vCPU, 8 GiB memory). The machines perform scans on different sites. We do not observe any significant differences in terms of timeouts between the different cloud providers.

4.3 Ethical Considerations

We carefully considered the ethical considerations of our scan. First of all, we perform our scans in a site-iterative manner. That is, we scan all resolvers in our dataset for one site before proceeding to the next. Thus, we are never overloading a single resolver with many concurrent requests at one time. Resolvers are chosen in a random order from our dataset, which prevents us from overloading a single AS with a lot of traffic at one time. We also track how many times a resolver times out, and remove it from our dataset if it times out more than one time. Ultimately, we find that 24,572 resolvers in the dataset meet our standards and are included in the final results.

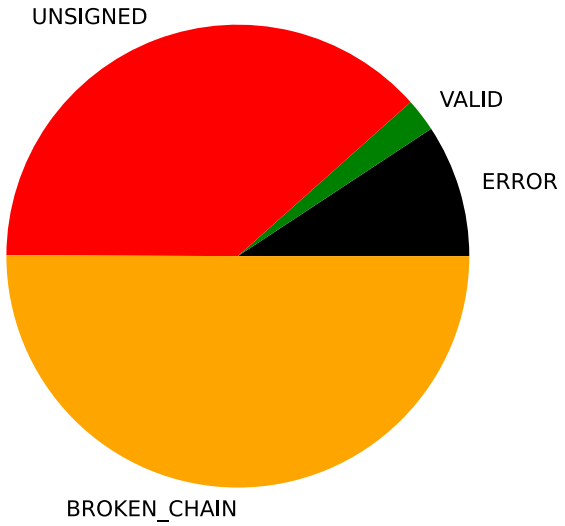
We assume that the DNS resolvers do not mind us performing our measurements on their servers. The IP addresses for all of these DNS servers are available in a public dataset [2]. Additionally, we perform only DNS queries against these servers, which is the behavior that the operators of these servers should expect; we do not perform any other scans. Finally, each server should only ever see at most about 100 queries from our project, during our collection period lasting several weeks.

5 RESULTS

The DNSSEC responses for each of the 100 domains against each of the 24,572 DNS resolvers are aggregated in Figure 3. We note that our of all 2,457,200 responses, only 2% represent a DNSSEC valid response.

Each of the DNS resolvers tested falls into one of three categories as shown in Figure 4: Those that only return valid or unsigned, those that only return broken chain, and those that return all three. For a DNS resolver properly implementing DNSSEC, it is expected that a either a valid or unsigned answer is returned; either an RRSIG record is found and validated through the chain of trust of the queried domain, or it is not. The next category of DNS resolver is comprised of those which return broken chain as a response to every query – even for those for domains (and trust chains) that properly implement DNSSEC.

For each site, we aggregate the total number of each class of response for queries made to resolvers. In Figure 6, we compare the responses received for queries to two domains that implement DNSSEC (`cloudflare.com` and `nih.gov`) with two domains that do not (`google.com` and `facebook.com`). Only responses from DNS resolvers that did not time out for *any* domain are included; all responses from DNS resolvers that timed out are removed from our data. For 6a and 6b, only 39% of DNS resolvers tested correctly return valid while 53% of resolvers incorrectly return a broken chain response. For 6c and 6d, only 41% of resolvers correctly return unsigned while 49% and 56% of resolvers respectively incorrectly return a broken chain response. While we acknowledge that it is great to see no unsigned records for valid domains and no valid records for invalid domains were returned, the amount of resolvers that return neither is troublingly significant.



Error	227862	9%
Valid	56921	2%
Unsigned	942271	38%
Broken Chain	1230146	50%

Figure 3: Aggregation of all results from each <site, resolver> pair using the top 100 sites and the 24,572 DNS resolvers that returned a result for every domain.

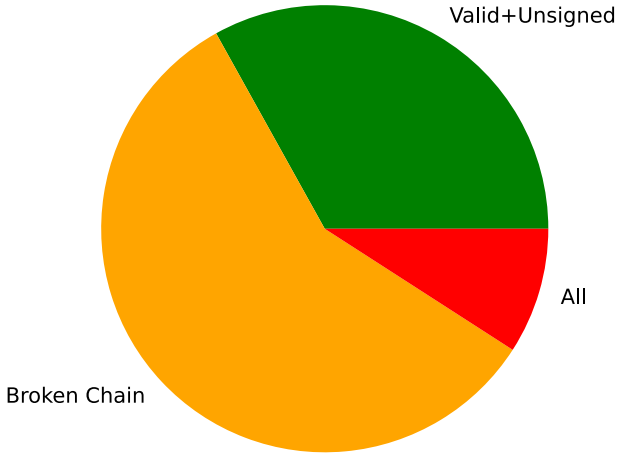
5.1 Resolver Consistency

One issue faced during this study is that of consistency of responses from DNS resolvers. While it may seem that a DNS resolver should return the same response for a given domain query, we found that this is not always the case. Take for example Figure 5. The DNS records for the `bilibili.com` domain were queried against this DNS resolver three times within several seconds. Each time, the DNS resolver returned a different response.

Results for each <site, resolver> pair were obtained with *one* query to `delv`. Thus, it is impossible to make an accurate assessment on the reliability of each resolver tested in this study or on the reliability of the resolver pool as a whole. We find through manual testing that the occurrence of inconsistent responses is insignificant, however. Future work may involve assigning consistency scores to resolvers and only considering those that meet certain thresholds.

5.2 DNSSEC Invalid Testing

Another important metric for our experiment involves testing a site where DNSSEC is invalid, instead of just not implemented. That is, there are existing RRSIG and DNSKEY records, but the hashes do not match. We expect DNS resolvers implementing DNSSEC to return an error. We use `www.dnssec-failed.org`, which is a



Valid+Unsigned	8076	33%
Broken Chain	14116	57%
All	2224	9%

Figure 4: Category of DNSSEC status that responses from each DNS resolver fall into (excluding error responses). Each group represents those resolvers which *only* returned responses of that group. Note, *All* represents an ambiguous category of resolvers which return a mix of these responses. See 6.2.2 for further discussion.

```
$ delv @99.48.38.153 bilibili.com
;; shut down hung fetch while resolving 'bilibili.com/A'
;; resolution failed: operation canceled

$ delv @99.48.38.153 bilibili.com
;; broken trust chain resolving 'bilibili.com/A/IN':
    99.48.38.153#53
;; resolution failed: broken trust chain

$ delv @99.48.38.153 bilibili.com
; unsigned answer
bilibili.com.    33  IN  A   8.134.50.24
bilibili.com.    33  IN  A   47.103.24.173
bilibili.com.    33  IN  A   119.3.70.188
bilibili.com.    33  IN  A   120.92.78.97
bilibili.com.    33  IN  A   139.159.241.37
```

Figure 5: Three queries to the DNS resolver at 99.48.38.153 for the records to `bilibili.com` made in quick succession.

domain maintained by Comcast, specifically for the purpose testing DNSSEC [6].

The results are evenly split, with almost exactly half of resolvers returning an error, while the other half return broken chain. See Figure 7. This demonstrates that DNSSEC is working as intended on the servers that implement it.

6 DISCUSSION

6.1 Categorizing Results

We use results from `delv` to determine whether a DNS resolver is properly implementing DNSSEC. Figure 3 demonstrates that the same share of resolvers, about 33%, will exclusively return either *valid* or *unsigned*. This is consistent with the expected behavior of DNSSEC. This leads us to conclude that about 33% of publicly available DNS resolvers properly implement DNSSEC and are able to validate site being queried. On the other hand, we see about 57% of resolvers returning *broken chain* for the same sites, which leads us to conclude that these resolvers do not properly implement DNSSEC. Note, the client we're using, `delv`, and the configuration our testing environment is consistent across all of our queries; thus, we maintain that the resolver is the point of failure for those results.

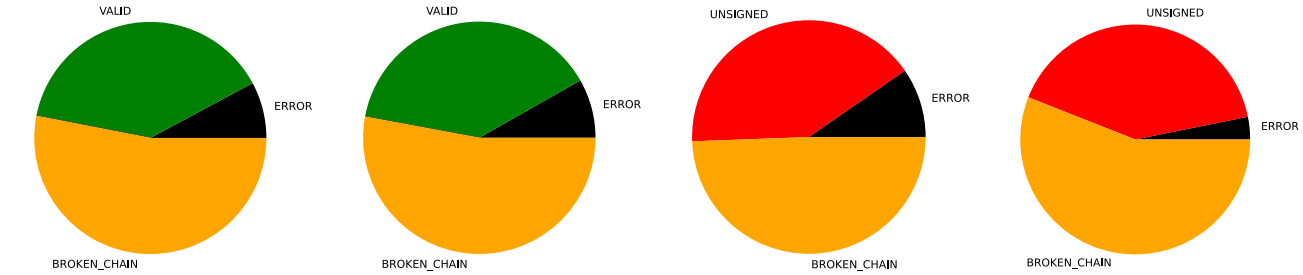
6.2 Ambiguous Results

6.2.1 Errors. As shown in Figure 6, a significant share of resolvers (10% for `google.com`) return an error status even for exceedingly popular websites. This could be caused by a number of factors. First of all, we are not certain whether all of the DNS servers in the dataset [2] that we found are truly available to the public Internet. Some of them could be filtering out requests which are outside of their delegated zone of service, which may manifest as errors in our results. Furthermore, we are not certain of these resolvers' policies. They may in fact be performing content filtering as directed by a governmental or private entity. Further testing is needed to confirm the locations and statuses of these resolvers. Alternatively, a more robust dataset which includes some sort of classification with each entry would allow us to draw conclusions about large-scale failures. For example: Do DNS servers returning errors for a specific site commonly reside in a particular country?

6.2.2 Outlying Resolvers. After categorizing the 24,572 DNS resolvers based on DNS responses, we find that a small but significant portion (9%) of DNS resolvers return a combination of valid, unsigned, and broken chain responses. These DNS resolvers do not fit neatly into our dichotomy of resolvers that implement DNSSEC and those that do not. Ones that fit into this third category may be those like in Figure 5 which return inconsistent responses.

6.3 Implications for DNSSEC

Prior studies have shown 31.65% of *sites* implement DNSSEC [10]. Additionally, we find that about 33% of *resolvers* implement DNSSEC. This translates to remarkably limited protection for the common user. With only one third of resolvers implementing DNSSEC as shown in Figure 4, it is a tough sell for users to only accept DNSSEC valid responses from DNS resolvers and filter out unsigned and broken chain responses. In the top 100 sites alone, *only 6* would be reachable. To this end, it does not provide an extremely high value to users who will simply ignore invalid responses. As it stands, DNSSEC does not provide a substantial benefit to an average user. This is the fault of site operators as well as DNS resolver operators.



Error	1924	8%
Valid	9610	39%
Unsigned	0	0%
Broken Chain	13038	53%

(a) cloudflare.com

Error	2020	8%
Valid	9550	39%
Unsigned	0	0%
Broken Chain	13002	53%

(b) nih.gov

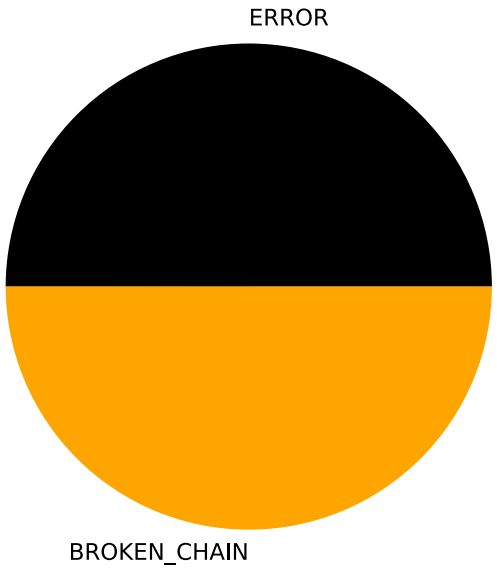
Error	2365	10%
Valid	0	0%
Unsigned	10065	41%
Broken Chain	12142	49%

(c) google.com

Error	777	3%
Valid	0	0%
Unsigned	10040	41%
Broken Chain	13755	56%

(d) facebook.com

Figure 6: DNSSEC query responses from four domains each tested against the same set of DNS resolvers. Domains 6a and 6b implement DNSSEC whereas 6c and 6d do not.



Error	18667	50%
Broken Chain	18670	50%

Figure 7: Query results from `www.dnssec-failed.org`

7 CONCLUSION

DNSSEC is an important tool for users to be able to validate DNS responses, but must be implemented by several nodes outside the control of a given domain. Regardless of whether a domain correctly implements DNSSEC, if its chain of trust or the DNS resolver does not completely implement it, then validation will fail. In this work, we show the extent to which DNS resolvers implement DNSSEC. We investigate the behavior of DNS resolvers and demonstrate the long road DNSSEC has to go before it can be considered widely adopted.

7.1 Future Work

There are several aspects of this study that could be improved in future work. For instance, instead of removing DNS resolvers that time out, they could be re-tested at a later time. In addition, it would be interesting to measure how responses from each resolver change as time progresses (as was done by a previous study in 2008 [12]). This would be useful to understand DNS resolvers which do not return consistent results, such as the one shown in Figure 5. Additional testing which takes DNS resolver response consistency into account would be of interest. Furthermore, investigating the causes of inconsistent results from specific DNS resolvers would provide further context into the deployment of DNSSEC and the challenges operators face in its implementation.

This work could also be extended by attempting to classify resolvers based on organization and software used. This could lend insight into whether publicly listed resolvers (such as those advertised by Cloudflare or Google) implement DNSSEC at a higher rate than those provided by Internet service providers. Additionally, it could identify software suites that are widely used, but do not implement DNSSEC.

7.2 Outlook

Based on this study, with such a limited fraction of the DNS resolver pool implementing DNSSEC, it remains challenging to make a case for DNSSEC filtering at this time. As more sites and resolvers embrace the adoption of DNSSEC, its value to users and enterprises will expand. We hope that with time, this void in DNS security eventually fills because the current state of DNSSEC is **bleak**.

REFERENCES

[1] [n. d.]. IDC 2021 Global DNS Threat Report. https://www.efficientip.com/resources/idc-dns-threat-report-2021/. Accessed: 2023-04-30.

[2] [n. d.]. The most exhaustive list of reliable DNS resolvers. https://github.com/trickest/resolvers. Accessed: 2023-04-28.
[3] [n. d.]. Overview of DNSSEC. https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/jj200221(v=ws.11). Accessed: 2023-04-28.
[4] [n. d.]. What is DNS cache poisoning? https://www.cloudflare.com/learning/dns/dns-cache-poisoning/. Accessed: 2023-04-28.
[5] 2005. DNS Security Introduction and Requirements. https://datatracker.ietf.org/doc/html/rfc4033. Accessed: 2023-04-28.
[6] 2013. DNSSEC Test Sites. https://www.internetsociety.org/resources/deploy360/2013/dnssec-test-sites/. Accessed: 2023-04-28.
[7] 2018. Dig and Delv. https://kb.isc.org/docs/aa-01152. Accessed: 2023-04-28.
[8] 2019. DNSSEC – What Is It and Why Is It Important? https://www.icann.org/resources/pages/dnssec-what-is-it-why-important-2019-03-05-en. Accessed: 2023-04-28.
[9] 2020. DNSSEC validation and BIND 9 cache. https://kb.isc.org/docs/aa-00912. Accessed: 2023-04-28.
[10] 2023. DNSSEC Validation Rate by country. https://stats.labs.apnic.net/dnssec. Accessed: 2023-04-28.
[11] Randy Bush and Rob Austein. 2013. The Resource Public Key Infrastructure (RPKI) to Router Protocol. RFC 6810. https://doi.org/10.17487/RFC6810
[12] Eric Osterweil, Michael Ryan, Dan Massey, and Lixia Zhang. 2008. Quantifying the Operational Status of the DNSSEC Deployment. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement* (Vouliagmeni, Greece) *(IMC '08)*. Association for Computing Machinery, New York, NY, USA, 231–242. https://doi.org/10.1145/1452520.1452548
[13] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society. https://doi.org/10.14722/ndss.2019.23386
[14] Nick Sullivan. 2014. DNSSEC: An Introduction. https://blog.cloudflare.com/dnssec-an-introduction/. Accessed: 2023-04-28.