A Dive into the Realm of DNSSEC: Understanding and Analyzing the Domain Name System Security Extensions

Chester Davison
Brigham Young University

Warda Usman
Brigham Young University

1 Introduction

The Domain Name System (DNS) is one of the oldest and most fundamental components of the modern Internet. DNS works as the mechanism to map domain names to Internet Protocol (IP) addresses, very much like a phonebook directory. This works to provides a human-readable layer to navigate the millions of machines and devices on the Internet.

When DNS was initially designed in the early 1980s, there was no consideration for strong security mechanisms in the protocol; as the notion was that internet security with DNS is not a dire need. Computers used at that time were not powerful enough to easily disrupt the DNS, and the network was much smaller, with fewer participants who were relatively well known and trusted. As the network grew and evolved, DNS remained unchanged as an insecure and unauthenticated protocol.

In order to add a layer of security to the DNS and to prevent attacks on DNS integrity, the Internet Engineering Task Force (IETF) has developed DNSSEC. DNSSEC (Domain Name System Security Extensions) is a set of extension mechanisms on the DNS which allows DNS records to be digitally signed, thus preventing (or at least detecting) tampering and attacks on the DNS records.

The scope of this work is to understand the working of DNSSEC, explore the different components that the DNSSEC is comprised of, and analyze how DNSSEC is implemented in the wild. We analyze the adoption of DNSSEC in the top websites on the internet. We also analyze which algorithms are currently being used most commonly in the DNSSEC implementation. Further, we look into the prevalence of using multiple key-signing keys in DNSSEC. We also find that there are multiple domains using a common key-signing key.

2 Background

DNSSEC allows authoritative nameservers to use public key cryptography to digitally sign RRsets (Resource Record Sets) in order to protect against DNS spoofing attacks. As the DNS is organized in a tree-like structure with the labels as nodes, where the top most node, (the root) has an empty label and has the highest rank in the hierarchy. With DNSSEC, a chain of trust is established, beginning with the root as a “trust anchor." To validate, the chain of certification is traced from the root zone downwards to the record being validated. It is made sure that every DNSKEY and DS record from the root zone downwards to the bottom-most label is valid and trustable (correctly signed by the immediate parent’s public key). By following this process, when an RRset is received by a stub resolver, it can be sure that the RRset was vetted by the holder of a certain private key, and the established chain of trust is a guarantee that a trusted key is used to validate the signatures.

For our analysis, we studied three different components of DNSSEC: DNSKEY, RRSIG, and DS.

DNSKEY: Every zone in the DNS has one or more public/private key pairs. The private keys are used to sign the resource record sets and the public keys are published in the DNSKEY resource records. To validate which key was used to sign a certain record (or key), we look at its DNSKEY record, as it holds the public key. The key provided by the DNSKEY record is the public key for which the corresponding private key in the pair signed a certain record.

RRSIG: A resource record signature, or RRSIG, is created for each RRset-public key pair whenever a zone is signed. An RRSIG record holds the digital signature, name, and type of the RRset being signed, along with the validity period of the key. The issue here is that the signature of the key is checked with the key itself, which is potential point of entry for an attacker who could tamper with the public keys and impersonate a legitimate source. To ensure that the public keys are not changed, a hash of the DNSKEY record is stored at the registry.

DS: The Delegation Signer, or DS, resource record type has the DNSKEY record. If the hash matches the public keys in the DNS zone, it ensures that the public keys are not tampered with, and the signatures associated with them can be trusted. The DS essentially links the signed zones to establish the chain of trust. Every DS record has the information about the

DNSKEY records of the zones that lie under it.

For analysis, we looked into the two types of keys used in DNSSEC: Zone Signing Key (ZSK) and Key Signing Key (KSK)

ZSK: A Zone Signing Key is a public/private key pair, where the ZSK private key is used to generate the RRSIG, for each of the resource record sets in a zone. The public key stored in the DNSKEY record to authenticate an RRSIG is the ZSK. A value of 256 in the key record indicates that the DNSKEY record holds the ZSK. Figure 1 shows the purpose of a zone-signing key.

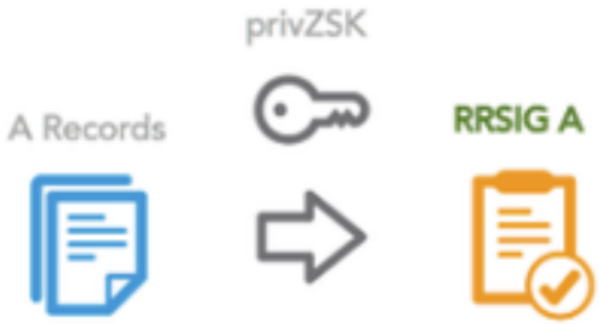

Figure 1: Working of ZSK

KSK: A Key Signing Key is also a public/private key pair, where the KSK private key is used to generate a signature for the ZSK (in the DNSKEY record). Similar to ZSK, the KSK public key is also stored in the DNSKEY record to be used to authenticate the ZSK. A KSK vets a ZSK very similarly to how the ZSK signs the RRset records. A value of 257 indicates that it contains the KSK. 257 basically represents that the last bit in its binary representation (`100000001`) is on. When that bit is off (`100000000`), the number is 256, which represents a ZSK. Figure 2 represents the working of a key-signing key. [1]

Figure 2: Working of KSK

[1] Figure 1 and 2 courtesy of https://blogs.akamai.com/

3 Methodology

We began our study by choosing a representative dataset of the DNSSEC situation today. We decided to use the Alexa Top 1 Million domain names for our analysis as we believed it would give us a good representation of the currently used domains and their DNSSEC implementation. We also used the top 500 domains from *mov.com*. Because the measurements vary when the top-lists are updated, as stated by Scheitle et al., the date when the top-lists are downloaded hold value. [2]. We downloaded the Alexa list on March 17, 2021. This file had a little over 700,000 domains instead of a full one million. After analyzing the the Alexa Top 1 Million we compared those values with the domains obtained from *mov.com*.

One of the first challenges we faced was developing an understanding of how DNSSEC is implemented in theory and in practice. We noticed that DNSSEC can be configured in various different ways and forms, and that it is the prerogative of the programmers to choose how they decide to implement it for their zones. We noticed that one variant of the implementations had domains that were self-signing both their Key Signing Keys (KSKs) as well as their Zone Signing Keys (ZSKs). Self-signing a ZSK essentially uses the ZSK as a KSK to sign itself, which was challenging for us to understand in the beginning. We learned that, in theory, a domain can have unlimited number of KSKs. Parsing these unlimited combinations of responses took us the most time, but also contributed towards developing our understanding of the DNSSEC. One example of domains with a complicated DNSSEC architecture is *nih.gov*. This domain has two different ZSKs, which were both self-signing, and two different KSKs, which are also self-signing. It took us some time to understand that the DNSSEC implementation on nih.gov was working by using four self-signing keys to validate its rdata types. Additionally, we saw that nih.gov was using TCP on top of this complicated implementation. Figure 3 visualizes the keys and the entities they sign for the nih.gov domain. This image is taken from DNSViz [1]. This image is just a single representation of one of the ways DNSSEC can be configured on a domain.

Another challenge in the analysis was understanding the various responses we got to our queries. We noticed that in response to our queries, we sometimes got a truncated response from the server where the answer section of the response was empty. This was very confusing as this happened for domains we were sure had DNSSEC enabled. As we used python for all our programming along with the DNS library, all the information we had was the query response. Understanding why querying domains with DNSSEC enabled returned empty answers was confusing to us. With the assistance of Professor Deccio we were able to figure out that some domains had TCP enabled as well. We then understood how DNSSEC works with TCP, and the importance of the TC flag. Afterwards, we re-queried the domains using TCP instead of UDP.

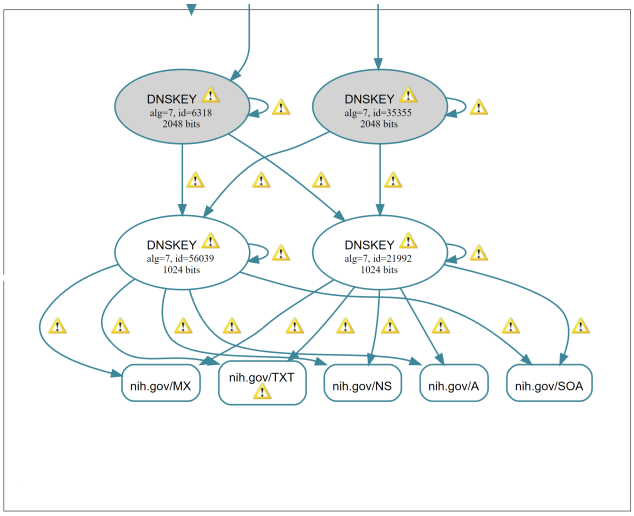

Figure 3: Key Visualization for nih.gov

After developing a working understanding of how DNSSEC works and the various ways it can be implemented, we queried using the domain names provided by Alexa Top 1 Million, asking for the NS rdata type. Then we worked on extracting the NS name from the response and executed another query by using the NS name and rdata type of A. This allowed us to capture the NS address. Finally, we constructed a request using the domain, rdata type of DNSKEY, and setting `want_dnssec` to 'True', we were then able to execute the query using the constructed request with the NS address from before. Upon getting the query responses, we parsed and analyzed the results. The query responses were first analyzed to confirm that the KSK was self-signed. Once we were sure that we had validated the domains using DNSSEC we collected the data we needed, such as: algorithm number, KSK ID, ZSK ID, was TCP enabled, and if the domain used multiple keys.

The time it took to query a domain and get a response back was very crucial to our analysis. We set the timeout to 3 seconds for every query. While none of the queries ever hit timeout, some took a little over 1 second each. As we wanted to analyze all the Alexa one million domains, even at 100 milliseconds per request, it would have taken almost 20 hours, and an average of 500 milliseconds would have been over four days! As this time was not practical, we decided to reduce the scope a little to be able to get a representative number of responses for our analysis that would not take overwhelmingly long. We decided that an analysis of the first 100,000 domains from the Alexa list would be reasonable. We ran our code using the Alexa's first 100,000 domains but after 4 hours of runtime, one of the domains threw an unreachable error and the code halted at 92,289 queries. We decided that an analysis of over 92,000 queries was sufficient for this project.

4 Analysis

A major motivation behind this project was to develop our understanding of how DNSSEC works and what are the important components that run this system. We also wanted to take a glimpse of what the DNSSEC implementation for the top-lists looks like. To achieve this, we decided to analyze all the aspects of DNSSEC that we were curious about.

DNSSEC Adoption: Our first research question was about the prevalence of DNSSEC in the top domains. We were interested in finding out how widely adopted DNSSEC is. To answer this, we looked at the query responses from the 92,289 domains from the Alexa list and separated them on the basis of whether or not they had DNSSEC enabled. In this list, we saw that only 3272 domains had DNSSEC enabled whereas all others were not using it at all. We also looked at the top 500 domains obtained from mov.com and found that 32 of them had DNSSEC enabled, about 6.4%, which was still almost twice the Alexa list (3.5%)

Algorithms: Since DNSSEC makes use of various cryptographic algorithms, we wanted to see which of them were the most prevalent. We were also interested in knowing if there were certain algorithms still being used which had been rendered obsolete by IETF. We found that SHA-256 was the most prevalent algorithm. SHA-256 is currently recommended-to-use by IANA and it was reasonable to see that it was the most commonly implemented algorithm. We also saw that SHA-512 was being used as well. Even though it was not very common, we found it interesting that there were domains using this updated implementation. M. Wander's work also shows that most second-level domains use RSA as signing algorithm [3]. Figure 4 shows the distribution of the top six algorithms in the domains we queried. The results from this particular analysis were rather reassuring. While there were some domains using RSA-SHA1, which DNSSEC specifications do not recommend to be used for signing, the number was rather low. We found it encouraging that SHA-256, the algorithm recommended by DNSSEC specifications could be seen in the highest frequency.

Zone Signing Keys: Another aspect of DNSSEC that seemed interesting to us was to understand the working of ZSKs. We wanted to see if there are any ZSKs which are being reused over the top websites we had. Since DNSSEC uses a chain of trust model, which begins with the trust anchor all the way to the bottom-most zone, and the chain of certification is traced from the rootzone downwards to the record being validated, we understood that there might be some ZSKs being used more than once. The results that we got, however, were very surprising to us. Out of the 3272 domains in the Alexa list that had DNSSEC enabled, over 1200 had the same

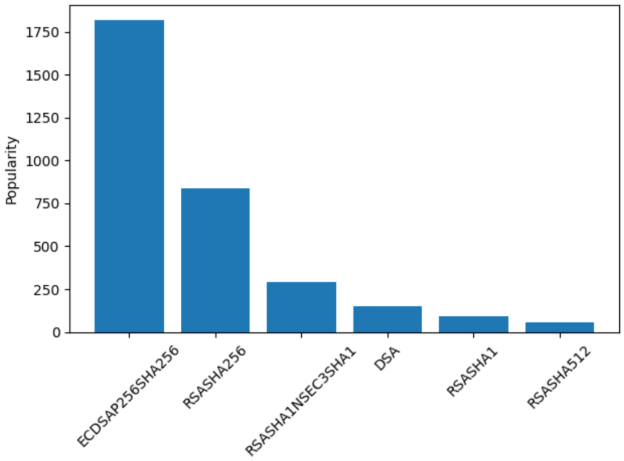

Figure 4: Working of KSK

ZSK, with `id = 34505`. We also saw some small number of repetitions in other keys, such as `id = 33072` and `id = 44688`. While using a repeated key is completely feasible, it puts multiple zones at risk when one key pair is compromised. We were not sure if multiple zones were using the same ZSK or if we were querying the same zone multiple times e.g., amazon.com, and google.com. It is understandable that a single key pair for one zone will be less error-prone to manage and cost-effective, but it is more risky to the security of the DNS if multiple zones are using the same key.

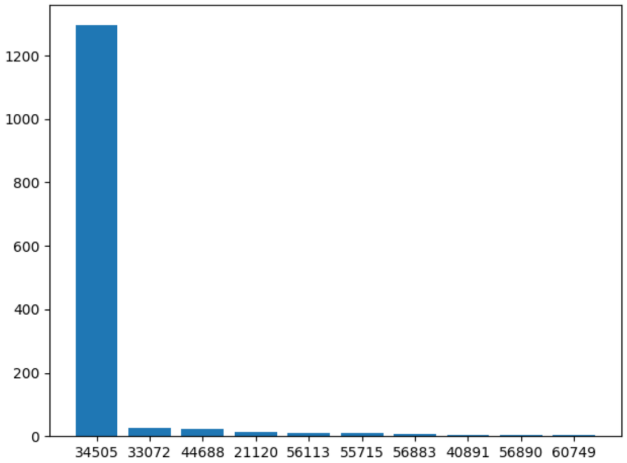

Figure 5: ZSK Distribution

Key Signing Keys: We saw a similar pattern with KSKs. We were interested in checking if there were any KSKs that were being used repeatedly. We saw that similar to the pattern for ZSKs, one KSK (with `id = 2371`) was being used repeatedly across more than 1200 domains. While it is 'allowed' to use the same KSK across multiple domains, we wonder how challenging it would be if there is any issue with the KSK `2371`. Since this key is being used across so many domains, what would happen if it becomes invalid either cryptographically, or if its private key is lost, or if it not renewed before its expiry? The near exclusive use of one key for every domain was astounding. We still do not have a definitive answer to why this is. To avoid issues and to make DNS more secure, we suggest use of unique key pairs for each zone.

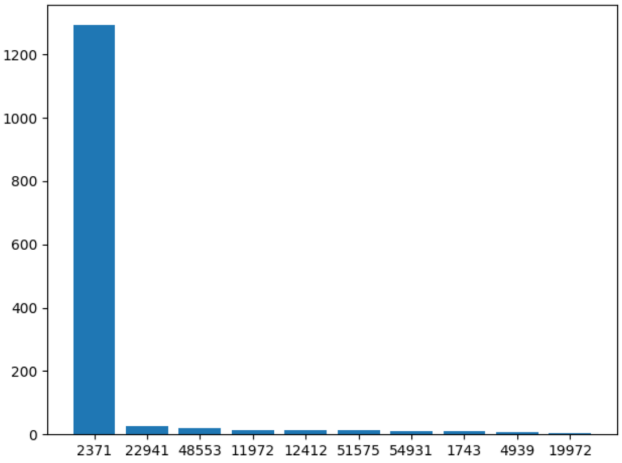

Figure 6: KSK Distribution

Number of KSKs: We noticed that there were certain domains using multiple KSKs. Developing an understanding of how DNSSEC works with multiple KSKs in a single domain was initially very difficult for us. We would send out a query and then get a response back with multiple keys in the answer. Once we understood the details of using multiple KSKs for a domain, we were interested to see if this was a common practice. We separated the domains that had multiple KSKs from the ones that were using a single KSK. We discovered that about 69% of the domains which had DNSSEC enabled were using a single KSK, while the remaining 31% were using multiple KSKs. Figure 7 shows a pie chart for this distribution.

5 Limitations and Future Work

As we began with studying the working of DNSSEC and its various implementations, we were able to analyze some of the most important components of DNSSEC in the top domains. However, due to time constraints, we could not run analyses from some other viewpoints. It took considerable amount of time to learn DNSSEC at a level that allowed us to parse and understand the data that was coming back from our queries. It would be interesting to see if our results can be replicated on other domain sets to give us a clearer image of how widely DNSSEC is enabled in the websites on

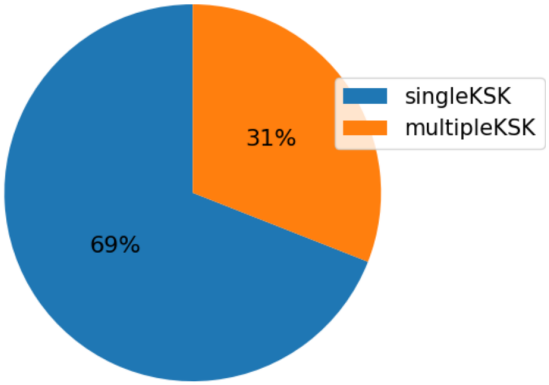

Figure 7: Distribution of Domains Using Single vs Multiple KSKs

the internet. Having access to the top level domains such as .com, .gov etc, we would have liked to analyze the DNSSEC situation in these top level domains, if time permitted. Another interesting analysis would be to check if there is a common quality or pattern to the websites that have DNSSEC enabled versus those which do not. Upon finding such patterns, we could look into the factors that cause a domain to implement DNSSEC and the obstacles to the wide adoption of DNSSEC. This kind of analysis could open ways to facilitating higher adoption of DNSSEC for all domains on the internet. One thing we really wanted to look into was why there are so many keys with the same ID. It was completely unexpected for us to see and at first we thought we had done something wrong. If time permitted we would have liked to explore and flesh out this idea more.

6 Conclusion

We began with a very rudimentary understanding of DNSSEC and tried to explore the different ways in which DNSSEC is implemented. We were able to develop a working understanding of DNSSEC and the major components on which the foundation of its implementation lies. We also gathered the DNSSEC responses by querying the top internet domains and then parsed and analyzed the data to have a deeper look into the DNSSEC world and compare theory with the real-world practices.

One finding that was the most surprising to us was that DNSSEC was not nearly as widely adopted as we had expected. In the Alexa Top 1 Million, there was not a domain with DNSSEC enabled until we got to the 41st domain name. It was unexpected to see most of the heavily used domains without DNSSEC enabled. We also found it interesting to see that a lot of *.gov* domains not only had DNSSEC enabled, but were also using multiple KSKs along with TCP encryption. Another interesting observation was that a large number of institutes, such as *stanford.edu*, have very sophisticated DNSSEC setups. We were also curious about checking the DNSSEC situation at *BYU.edu*. We found that BYU does have DNSSEC enabled, although the use is very limited where some rdata types need to be updated to ensure better security.

References

[1] DNSviz *http://dnsviz.net.*

[2] Quirin Scheitle, Oliver Hohlfeld, Julien Gamba, Jonas Jelten, Torsten Zimmermann, Stephen D. Strowes, and Narseo Vallina-Rodriguez. A long way to the top: Significance, structure, and stability of internet top lists. In *Proceedings of the Internet Measurement Conference 2018*, IMC '18, page 478–493, New York, NY, USA, 2018. Association for Computing Machinery.

[3] M. Wander. Measurement survey of server-side dnssec adoption. In *2017 Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–9, 2017.