"DNSSEC and deSEC"

Implementation of DNSSEC, challenges and practical comparison of deSEC, platform implementing DNSSEC

A Thesis submitted in partial fulfillment for the requirement for the degree of

Masters' of Science

in

Computer Science: Focus on Cyber Security

on the Fourth Semester (Summer 2020)

SUBMITTED BY:-

AKASH KARAR

Student id: 3103123

Email: karar.akash@gmail.com, 3103123@stud.srh-campus-berlin.de

Under the guidance of

Primary Supervisor

Prof. Dr. Vladimir Stantchev

Professor of Information Systems

SRH University of Applied Sciences

Secondary Supervisor

Dr. Peter Thomassen

Senior Security Expert

SSE – Secure Systems Engineering GmbH, deSEC e.V.

Table of Contents

Acknowledgements / Foreword

I would like to thank my secondary supervisor Dr Peter Thomassen who was also my mentor during my internship in SSE – Secure Systems Engineering GmbH, for his guidance and constructive comments while I was writing the thesis. Without his help this would not have been feasible.

Furthermore, I would like to extend my thanks to Herzberg, Amir & Shulman, Haya {amir.herzberg,haya.shulman}@gmail.com for their presenting in the IEEE 2013 Conference on Communications and Network Security (CNS) with their article named as “DNSSEC: Security and availability challenges. 365-366.10.1109/CNS.2013.6682730.”

Last but not the Least I would like to thank the owners of http://stats.dnssec-tools.org/ namely Wes Hardeker { hardaker@isi.edu} and Viktor Dukhovni { viktor@dukhovni.org} for givng me the licence to use the statistics data of their website regarding DNSSEC trends.

List of Tables and Figures1

Abstract

Domain Name System is a publicly accessible infrastructure designed by the IETF to facilitate the lookup of human friendly and human rememberable names of websites aka web service providers to their respective machines in the Internet which are identifiable only by their numbers. DNSSEC is a security extension of DNS that helps in securing these lookups against spoofing, tampering and malicious intenders in the internet, to maintain a public, secure, authentic, and universally compatible system of information transfer between the web service providers and their clients. In this document we have worked on the general overview of working of DNSSEC, the need for it, current trends and operational problems faced to implement such an infrastructure. Further research has been made upon a DNSSEC implementing platform known as deSEC from the grounds of Berlin with an open source community, including a comparative study made between some domains using the tool and its contribution in the field of DNSSEC with security and usability in mind.

Chapter 1: DNS

DNS or Domain Name System is a hierarchical tree of servers and a distributed data base of Internet mappings of domain names and different values, i.e., IP addresses. In the internet IP addresses are the actual numeric addresses of computers, network items and other devices.[2] The Domain Name System delegates the responsibility of assigning domain names and mapping those names to Internet resources. They make it human readable and friendly to surf through the web without remembering numeric identities of websites. [3]

1.1 Domain name syntax, internationalization

Domain names consist of one or more parts called labels that are separated by dots. The right-most label gives us the top-level domain; as for instance, the top-level domain of domain name example.com is com. [2][3][4] Since DNS is a hierarchical system, traversing from right to left takes us further down the tree and specifies subdivisions, or subdomains of the domain to the right. Thus, the label example specifies a subdomain of the com domain. The maximum allowed sub division is 127 levels. [5] A label can have zero to 63 characters. The root zone has a null label, i.e. of length zero. A full domain name cannot exceed the length of 253 characters in its textual representation. [2] The characters in the labels belong to the ASCII character set, including characters from a through z, A through Z, digits 0 through 9, and the hyphen (LDH rule - letters, digits, hyphen). A Domain name is case- independent, but the labels cannot be starting or ending with a hyphen and top-level domain names should not be an all-numeric. ICANN also specifies some rules for nomenclature [3].

1.2 Background

1.2.1 Working: A brief description of how DNS works:

In order to have a short however brief summary, let us take into account a case when a user enters the name of a web website name during a browser on their device. A Stub resolver, a really straightforward DNS client that is inbuilt with the device is used by the browser to construct an application's request for DNS data to a recursive resolver to start the process of address resolution of the domain name into an Internet Protocol (IP) address. The recursive resolver then talks to the Root Server that contains the address of all the top-level domains like .com, .org etc. The top-level name server then directs to

the second level domain name server. For Example, in the case of example.com, the level ends within the above three steps and the website query is served. In case of further more levels the process is repeated until the domain name is traversed completely from right to left. This needs to be done for every query, but doing such a mechanism for all devices every time would put a huge traffic queue on root servers. In order to curb this queue, caching is used to store the recently queried domain names and corresponding addresses for a certain amount of time.[3]

1.2.2 Name servers:

The Domain Name System is a distributed database system, that uses the client–server model, whose nodes are the name servers. Each domain normally has at least one authoritative DNS server, publishing information about that particular domain and name servers of the domains subordinate to it. The top of the hierarchy tree has the root name servers, which are queried while looking up TLDs.

1.2.3 Authoritative name servers:

An authoritative name server is one that only provides answers to the DNS queries about domain names configured by the administrator. They can be of two types, either a primary or a secondary server. The DNS data stored on an authoritative name server for every domain name is called as the zone.[2]

1.2.4 Recursive and caching name server:

It is possible for Authoritative name servers to single handedly operate the DNS, but then every DNS query would have to start with recursive queries at the root zone. In order to improve efficiency and to reduce DNS traffic, DNS cache servers store DNS query results but only for a certain period of time known as the Time to Live or TTL. Such caching DNS servers contain the recursive algorithm and can also resolve a given name from DNS root to the authoritative name server.

1.2.5 DNS resolvers:

DNS resolver, the client side of the DNS, is responsible for address resolution of the queries i.e. translating a domain name into an IP address. Different DNS resolvers are present according to their query methods, such as recursive, non- recursive, and iterative.

In a non-recursive query, a DNS resolver queries an authoritative server or one which provides a partial result without querying others. A caching DNS resolver delivers a time cached result and reduces the load on the DNS servers that handle upstreaming.

A recursive query, is one for which the DNS server answers the query entirely by querying other name servers as needed.

An iterative query procedure by which a DNS resolver queries a chain of one or more DNS server(s) until the request is fully resolved.

1.2.6 Circular dependencies and glue records:

Name servers in delegations are identified by name, instead of IP addresses. Circular dependency is when a name given within a delegation maybe a subdomain of the domain for which the delegation is being provided, i.e. name server is hosted at the same domain as the name itself. For example, let's say that example.com has name server with the name of ns1.example.com. Thus, when requests are sent to example.com, they will be pointing to ns1.example.com which will again point to example.com and so on.

Therefore, the name server providing the delegation must also provide at least one IP addresses for the authoritative name server mentioned in the delegation. This is called a glue information. Glue records are used to prevent circular dependencies. A glue record is an additional A record served by a parent DNS server that is not authoritative for the zone in order to locate the name server without any problem of dependency. It is like a static record held by a parent domain so that the resolver does not get lost in a circular loop in case of circular dependencies.

1.2.7 Other applications:

The Domain Name System includes several other features: One to one relationship between host names and IP addresses are not always necessary, Multiple hostnames can be a part of a single IP. This is useful in virtual hosting and serving many websites from a single host. Similarly, a single hostname may resolve to many IP addresses. This is done to have higher fault tolerance and load distribution to multiple servers across the globe.

As an MX record maps between a domain and a mail exchanger, it can be efficiently used for storage and sharing of IP addresses of blacklisted email hosts. Such blacklists are either subscription-based or free of cost and are available for use by email administrators and anti-spam software.

Anycast addressing is used to provide backup and continual service in the event of computer or network failure. In anycast, one IP address is applied to multiple servers, thus, a number of DNS servers responds to DNS queries; typically, the one that is geographically closest. This reduces latency and also improves uptime for the DNS resolving service.

Dynamic DNS (DDNS) updates DNS records dynamically, i.e. while moving between ISPs or mobile towers IP addresses changes dynamically.

1.3 DNS message formatting

The DNS protocol has two types of messages, queries and replies. They both have the same format consisting of a header and four sections.[2] The four sections are: Question, Answer, Authority, and an Additional space. The header section is always present and consists of the Identification (ID), Flags(discussed in table 1), Number of entries in question section (QDCOUNT), Number of entries in answer section (ANCOUNT), Number of authority resource records (NSCOUNT), and Number of additional RRs flags to control the content of the above four sections (ARCOUNT). The header flags have the following format as explained in Table 1.

Table 1: DNS Header Flag Format

QR	A 1-bit field – 0 for when the message is query and 1 for when it's a response
OPCODE	A 4-bit field specifying the kind of queries 0 for a standard query (QUERY) 1 for an inverse query (IQUERY) 2 for a server status request (STATUS) 3-15 is reserved for future use.
AA	Authoritative answer
TC	Truncation bit set when messages get truncated for greater than permissible length
RD	Recursion Desired bit is set in a query and can be copied to be a response
RA	Recursion Available bit set whether recursive query support is available
Z	Reserved for future use
RCODE	A 4-bit field with values ranging from 0 -15. 0 for no error condition

	1 for format error 2 for server failure 3 for name error 4 for not implemented 5 for refusal to perform a specific operation 6 -15 being reserved for future use.

1.3.1 DNS Question section:

The question section contains the following fields:

Table 2:DNS Question format

QNAME	Contains a sequence of labels of the domain name, with the length variable to the length of the label.
QTYPE	Consists of the type of RR requested
QCLASS	Consists of the class code of the query

1.3.2 CLASS Values:

Table 3: DNS CLASS values for queries and responses

IN	1 the Internet
CS	2 CSNET class (obsolete)
CH	3 CHAOS class
HS	4 Hesiod

1.3.3 DNS protocol transport mechanism:

DNS usually uses the User Datagram Protocol (UDP) [3] which runs on port 53 to serve requests of 512 bytes with queries consisting of a single UDP request packet from the client to the server and then again, a single UDP reply back to the client. Larger UDP packets greater than 512 bytes are used, when support for EDNS [6] is present. Or else, the query is resent using the Transmission Control Protocol (TCP). TCP is also used for zone transfers.

1.3.4 Resource records:

The Domain Name System stores many types of information elements regarding a particular domain known as Resource records (RRs). Each RR has a type which is a name or number, a TTL, a class, and type-specific data. Resource records of the same type are described as a resource record set (RRset) which is returned upon query. All records use the common format specified in RFC 1035 [3].

1.3.5 Resource record (RR) fields:

Table 4: DNS Resource records field

Field	Contents	Length in octets
NAME	Name of the node of record	Variable
TYPE	Type of RR in numeric form	2
CLASS	Class code	2
TTL	Count of seconds that the RR stays valid (max $2^{31}-1$ = 68 years)	4
RDLENGTH	Length of RDATA field	
RDATA	Additional RR-specific data	Variable, as per RDLENGTH

NAME is the fully qualified domain name of the node in the tree

TYPE is the record type showing the format of the data hints its intended use. For instance, the A record is used to translate from a domain name to IPv4 address and the MX record specifies the mail server for a domain specified in an e-mail address.

RDATA is data of type-specific relevance, like the IP address for address records.

The CLASS of a record is usually set to IN (for Internet) for common Internet hostnames, servers, or IP addresses.

1.4 DNS Resource Records

Following are the common DNS resource records defined by the IETF [2][3] with examples. We have taken posteo.de as a common example and used the tool dig to find the records.

1.4.1 A (Host address)

The A-record is a basic and most commonly used DNS record type. It is used to translate human friendly domain names such as "www.example.com" into IP-addresses that are machine friendly numbers.[3] For Example we can see that the A record for posteo.de is:

```
posteo.de.          60      IN      A       185.67.36.145
```

1.4.2 AAAA (IPv6 host address)

An AAAA-record is used to specify the IPv6 address for a host and is equivalent of the A-record type for IPv4.(IPv6), e.g.,

```
ns01.posteo-dns.de.         86400   IN      AAAA    2a05:bc0:1000::40:1
```

1.4.3 ALIAS (Auto resolved alias)

They are virtual alias records, configured on the server side, that are of multipurpose uses including solution to problems with CNAME-records at the domain apex (like the one cause by “the naked domain”, i.e. “www.posteo.de” is called an internet domain, but “posteo.de” is called a naked domain as it misses the front “www”).

1.4.4 CNAME (Canonical name for an alias)

CNAME-records are domain name aliases that can be used to identify single device, e.g., the device "device.xyz.de" can be a web-server and an ftp-server. Thus, two CNAME-records defined as: "www.xyz.de" = "device1.xyz.de" and "ftp.xyz.de" = "device1.xyz.de". [3]

1.4.5 MX (Mail exchange)

MX-records are used to specify the e-mail server(s) by pointing to the name of an e-mail server and holds a preference number for that server, in case of handling by multiple e-mail servers.[4]

1.4.6 NS (Name Server)

NS-records identify the DNS authoritative servers for a zone. It is mandatory for a zone to contain one NS-record for each of its own DNS servers (primary and secondaries). Delegation is an important function of the NS-record. A domain is delegated to other sub-domains, e.g., all ".de" sub-names ("example.de") are delegated from the "de" zone, and the "de" zone has NS-records for all the ".de" sub-names.[3]

For example:

posteo.de.	300	IN	NS	ns01.posteo-dns.de.
posteo.de.	300	IN	NS	ns21.posteo-dns.ch.
posteo.de.	300	IN	NS	ns31.posteo-dns.eu.

1.4.7 PTR (Pointer)

They are used reverse A-records and AAAA-records for reverse IPv4 and IPv6 mapping respectively. The PTR-record is the IP address with reversed segments and "in-addr.arpa" at end. Generally, PTR records are used to point to host names. [3]

1.4.8 SOA (Start of Authority)

A SOA record has the following:

Name of the primary DNS server - host name of primary DNS server that also has the same NS-record.

E-mail address of responsible person – standardized as “hostmaster.example.com” for “hostmaster@example.com”

Serial number - In order to check if the zone has been changed. Zone transfer is initiated when the number is higher than that with the secondary server. When it is close to overflowing, very low numbers also indicate an update. [2]

Refresh Interval Intervals for checking for changes

Retry Interval Intervals that the secondary server should retry checking for changes when first refresh fails.

Expire Interval Validity interval of zone after refresh, after which servers will discard the zone upon no refresh made within said time.

Minimum (default) TTL Time specified for storing negative responses, e.g., non-existence of records. [3][2]

For example:

```
posteo.de.          300     IN      SOA     (ns01.posteo-dns.de. hostmaster.sys4.de.
                                            1596416451      ;serial
                                            3600            ;refresh
                                            900             ;retry
                                            604800          ;expire
                                            300                     ;minimum
        )
```

This shows that for the domain posteo.de, the SOA record has the name server ns01.posteo-dns.de. and the host email address as hostmaster.sys4.de.

1.4.9 TXT (Descriptive text)

Holds descriptive general information about a domain name like owner name and contact information. RFC1035.

1.4.10 SRV

SRV-records specify location of a service, for advanced load balancing and to specify ports for services. They are mostly redundant. The SRV record name is made up of 3 parts: Service, Protocol (TCP or UDP) and Domain name. [8]

1.4.11 ANY

Clients can ask for an ANY record and servers may or may not answer it. If answered, this generally provides the client with a list of several records known by the server.

Chapter 2: DNSSEC

The Domain Name System (DNS) was initially simply intended to be a versatile conveyed framework and did not include any security detail, that are currently required in modern Internet world. The Domain Name System Security Extensions (DNSSEC) is a lot of augmentations to DNS from the Internet Engineering Task Force (IETF) which gives DNS resolvers cryptographic verification of DNS information, confirmed disavowal of presence, and information uprightness, yet not accessibility nor privacy. [9][11]. Consequently, it endeavors to include security, while likewise keeping up backward compatibility. DNSSEC was made to secure applications and caching resolvers serving them from utilizing manufactured or controlled DNS information, for example, that which might be made by DNS reserve harming. Each answer from DNSSEC ensured zones are carefully marked. A DNS resolver can check the data genuineness by checking the computerized mark to the data distributed by the zone proprietor and served on a definitive DNS worker. For instance, DNSSEC can secure any information distributed in the DNS, including text records (TXT) and mail exchange records (MX)[10].

DNS was developed in the 1980s when the Internet was very much much smaller. Security was also not a primary concern in its design. A recursive resolver sent DNS queries to an authoritative name server but the problem was that the resolver had no way to verify the authenticity of the response. It can have only one metadata information i.e. the response is from same IP address where the resolver sent the original query. But trusting on one source is not enough as it can also be easily spoofed. Thus, an attacker can easily masquerade himself as an authoritative server by spoofing a response that a resolver originally queried, and thus redirect users to a malicious website.

Recursive resolvers often cache the DNS data they have received from authoritative name servers in order to speed up the resolution process for next resolutions of queries. This caching has a downside, as if an on-path attacker sends a forged DNS response that has been accepted by a recursive resolver then the attacker has poisoned the cache of the recursive resolver. Thus, henceforth the recursive resolver would then proceed to return the fraudulent DNS data to other devices that query for it. So, threat posed by a cache-poisoning attack [13], if considered the case when a user visits their bank's website and

is redirected to a fraudulent website that impersonates the bank website, unknowingly users would enter their name and password, as usual and thus the user would have provided its banking credentials to the attacker, who could misuse them as intention. Using DNSSEC as an extension to DNS has proved the right way to solve this cache poisoning attack [14][15], as the resolvers can verify the responses through signatures of parent name servers and also check for authenticity. DNSSEC has also been proposed to solve the Kaminsky Cache poisoning attack [12] and has been recommended as the long-term solution by ICANN.

2.1 Background:

The following are important terms needed to know before engaging deeper into DNSSEC. Each sub-section describes some words that are used in DNSSEC.

2.1.1 EDNS:

DNSSEC adds some new Resource records to the legacy DNS responses. These mostly include the cryptographic keys and hashes. Extension mechanisms for DNS (EDNS) is used for expanding the size of different parameters of the Domain Name System (DNS). This is due to the fact that the old DNS protocol had size restrictions of 512 bytes of UDP packets that the Internet engineering community designed in the legacy DNS format. EDNS is essential for the implementation of DNS Security Extensions (DNSSEC) and is used for sending general information from resolvers to name servers. EDNS0 [5] is the first set of extensions published by the IETF (Internet Engineering Task Force) in RFC 2871 [6], but updated in RFC 6891[7]. EDNS has added the OPT pseudo-record that adds accommodation for upto 16 flags and also extends the space for the response code, thus allowing DNSSEC to implement its new RRs. [9]

As seen from the tool Domain Information Groper (dig) here is an example of an OPT pseudo record from the website posteo.de:

;; EDNS version 0

;; flags: 8000

;; rcode: NOERROR

;; size: 4096

;; option:

2.1.2 Authentication Chain:

The alternating sequence of DNS public key (DNSKEY) RRsets and Delegation Signer (DS) RRsets structures a chain of signed information, with each next connection in the chain vouching for the following. The DNSKEY RR is utilized to verify signature covering the DS RR and permits the DS RR to be validated. The DS RR contains a hash of another DNSKEY RR and this new DNSKEY RR is verified by coordinating the hash in the DS RR. This new DNSKEY RR thus validates another DNSKEY RRset and, thus, some DNSKEY RR in this set might be utilized to verify another DS RR, so forth, until the chain at long last finishes with a DNSKEY RR whose private key signs the ideal DNS information. For instance, the root DNSKEY RRset can be utilized to verify the DS RRset for "example." The "example." DS RRset contains a hash that coordinates some "example." DNSKEY, and this present DNSKEY's corresponding private key signs the "example." DNSKEY RRset. Private key parts of the "example." DNSKEY RRset sign data records, like, "www.example." and DS RRs for assignments, like, "subzone.example."

2.1.3 Authentication Key:

An Authentication key is a public key that a security-aware resolver has confirmed and can in this manner utilize to authenticate information. The security-aware resolver can get confirmation keys in three ways: [9] The resolver is by and large arranged to know minimum one public key; this information is as a rule either the public key itself or a hash of the public key as found within the DS RR (see "trust anchor").The resolver may utilize the authenticated public key to verify a DS RR and the DNSKEY RR to which the DS RR alludes. The resolver may be able to decide that a new public key has been signed by the private key comparing to another public key that the resolver has verified. Thus, the resolver must continuously be guided by a local policy when choosing whether to authenticate a new public key, also in the event, the local policy is basically to confirm any unused public key for which the resolver is able confirm the signature.

2.1.4 Authoritative RRset:

Inside the context of a specific zone, a RRset is "authoritative" if and just if the proprietor name of the RRset exists in the subset of the name space that is at or beneath the zone apex and also at or over the cuts that differs the zone from its kids, assuming any. All RRsets at the zone apex are authoritative, with the exception of certain RRsets

at this domain name that, if there, have a place with this current zone's parent. These RRset could incorporate a DS RRset, the NSEC RRset (defined in 2.3.3) referencing this DS RRset (the "parental NSEC"), and RRSIG RRs related with these RRsets, which are all legitimate in the parent zone. Thus, if this zone contains any delegation points, just the parental NSEC RRset, DS RRsets, and all RRSIG RRs related with these RRsets are legitimate for this zone.

2.1.5 Delegation Point:

Is the term used to depict the name at the parental side of a zone cut, i.e. the delegation points for, for example "foo.example" would be the foo.example node in the "example" zone (rather than the zone apex of the "foo.example" zone)

2.1.6 Island of Security:

When a signed, delegated zone that does not have an authentication chain from its delegating parent, i.e., there is no DS RR containing a hash of a DNSKEY RR for the island in its delegating parent zone (see [16]), an island of security is used to define such a condition. It is served by security-mindful name server and may give validation chains to any appointed kid zones.

2.1.7 Non-Validating Security-Aware Stub Resolver:

A security-aware stub resolver confides one or more security-aware recursive name servers to perform the vast majority of the tasks examined in this thesis for its benefit. Thus, a non-validating security-aware stub resolver sends DNS queries, receives DNS responses, and establishes a secured channel to a security-aware recursive name server providing these services on behalf of a security-aware stub resolver.

2.1.8 Non-Validating Stub Resolver:

A less dreary term for a non-approving security-mindful stub resolver.

2.1.9 Security-Aware Name Server:

An entity acting in the role of a name server that comprehends the DNS security expansions. Specifically, a security-aware name server is an entity that gets DNS queries, sends DNS responses, supports the EDNS0 [2.1.1] message size extension, the DO bit [17] , and also supports the RR types and message header bits.

2.1.10 Security-Aware Recursive Name Server:

An element that demonstrations in both the security-aware name server and security-aware resolver roles. An increasingly bulky however identical expression would be "a security-aware name server that offers recursive service".

2.1.11 Security-Aware Resolver:

An element acting in the job of a resolver (characterized in segment 2.4 of [RFC1034]) that understands the DNS security extensions. Specifically, a security-aware resolver is an element that sends DNS inquiries, gets DNS reactions, underpins the EDNS0 (2.1.1) message size extension and the DO bit [17], and is fit for utilizing the RR types and message header bits characterized in this record set to give DNSSEC administrations.

2.1.12 Security-Aware Stub Resolver:

A security-aware entity behaving as a stub resolver (defined in section 5.3.1 of [RFC1034]) that has an understanding of DNS security extensions set to provide additional services that are not possible to get from a security-oblivious stub resolver. Security-aware stub resolver can be either "validating" or "non-validating". This nomenclature depends whether the stub resolver attempts to verify DNSSEC signatures on its own or trusts a friendly security-aware name server to do so.

2.1.13 Security-Oblivious:

An entity that is not "security-aware".

2.1.14 Signed Zone:

A zone whose RRsets are signed and has properly made DNSKEY, Resource Record Signature (RRSIG), Next Secure (NSEC), and (optionally) DS records.

2.1.15 Trust Anchor:

It is a configured DNSKEY RR or DS RR hash of a DNSKEY RR. This public key is used by a validating security-aware resolver for building the authentication chain to a signed response. A validating resolver has to get the initial values of its trust anchors via some secure or trusted means outside the DNS protocol.

2.1.16 Validating Security-Aware Stub Resolver:

The validating security-aware resolver is one that sends queries in recursive mode but, performs signature validation on its own rather than just blindly trusting any other security-aware recursive name server.

2.1.17 Validating Stub Resolver:

Another term for a validating security-aware stub resolver.

2.1.18 Zone Apex:

It is the term that describes the name at the sub domain's side of a zone cut. A Zone apex is where the SOA, NS and MX records are placed.

2.1.19 Zone Signing Key (ZSK):

A zone signing key is an authentication key pair (private and public). The private key is used to digitally sign each RRset in the zone, while the Public key is used to verify the signature. It is a part of the same DNSKEY RRset as the key signing key and used for a slightly different purpose from the key signing key also has a different validity lifetime. DNSSEC validation does not differentiate between key signing keys and other DNSSEC authentication keys, and it is possible to use a single key as both a key signing key and a zone signing key, thus designating an authentication key as KSK is totally an operational problem. [18]

Digital signatures are created by the zone operator for each RRset using private ZSK and stored in their name server as RRSIG records. The resolvers need the public key in order to verify these signatures, they get it from the name server of the zone from the DNSKEY record. Thus, when a DNSSEC resolver asks for a particular record type, the name server returns the corresponding RRSIG and the public ZSK with the record.

2.1.20 Key Signing Key (KSK):

In order to mitigate the probability of having a compromised ZSK, in addition to ZSK, DNSSEC name servers also have a key-signing key (KSK). The KSK validates the DNSKEY record in the same way as ZSK secured the RRsets. The private KSK signs the public ZSK creating a RRSIG for the DNSKEY. Also like the public ZSK, name server publishes the public KSK in another DNSKEY record. KSK is an authentication

key that relates to a private key used to sign at least one other authentication keys for a particular zone. Typically, the private key of the KSK will sign a Zone Signing Key (ZSK), which in turn will sign other zone data with its private key. Local policy may require that the zone signing key be changed more frequently than the key signing key which may have a longer validity period to have a better stable and secure entry point into the zone.

2.1.21 CSK:

CSK (combined signing key) is an implementation in some DNS provides that provide DNS support. One such example is PowerDNS which provides the software for the tool we discuss in later chapter 3. CSK or Combined Signing Key is a combination of KSK and ZSK. It is used instead of having two different keys to sign the zone. This provides flexibility and robustness in key rollovers for DNSSEC, and also helps in decreasing the rollover complexity discussed later in chapter.
Although a reverse argument specified in 4641 says that the advantages of having KSK and ZSK separated are:

1. Ads security as KSK can be made extra strong by adding more bits than ZSK.
2. KSK is only used to sign the ZSK so, less rotation of KSK can be done and made more secure by storing them in a safer location than ZSK. [19]

2.2 DNS Security (DNSSEC) Working:

DNS requests and responses are vulnerable to on-path attacker (Man in the Middle - MitM attacks), like a malicious wireless client can eavesdrop to the communication of other clients and respond to their DNS requests with a malevolently made DNS response, containing a spoofed IP address, e.g., redirecting the clients to a phishing website. This above process is known as DNS cache poisoning. To mitigate this cache poisoning vulnerability, Domain Name System Security Extensions (DNSSEC) standard [9][16][20] was designed, by integrating data integrity and origin authenticity through cryptographic digital signatures over DNS resource records. The digital signatures enable the client, e.g., a resolver, that supports DNSSEC validation, to verify the data in a DNS response to be the same as published within the target zone.

DNSSEC has new resource records (RRs) to store signatures and keys for authentication of DNS responses. The RRSIG record contains a signature authenticating an RR-set, thus signing is done on RR-sets, and not responses. Thus, DNSSEC allows

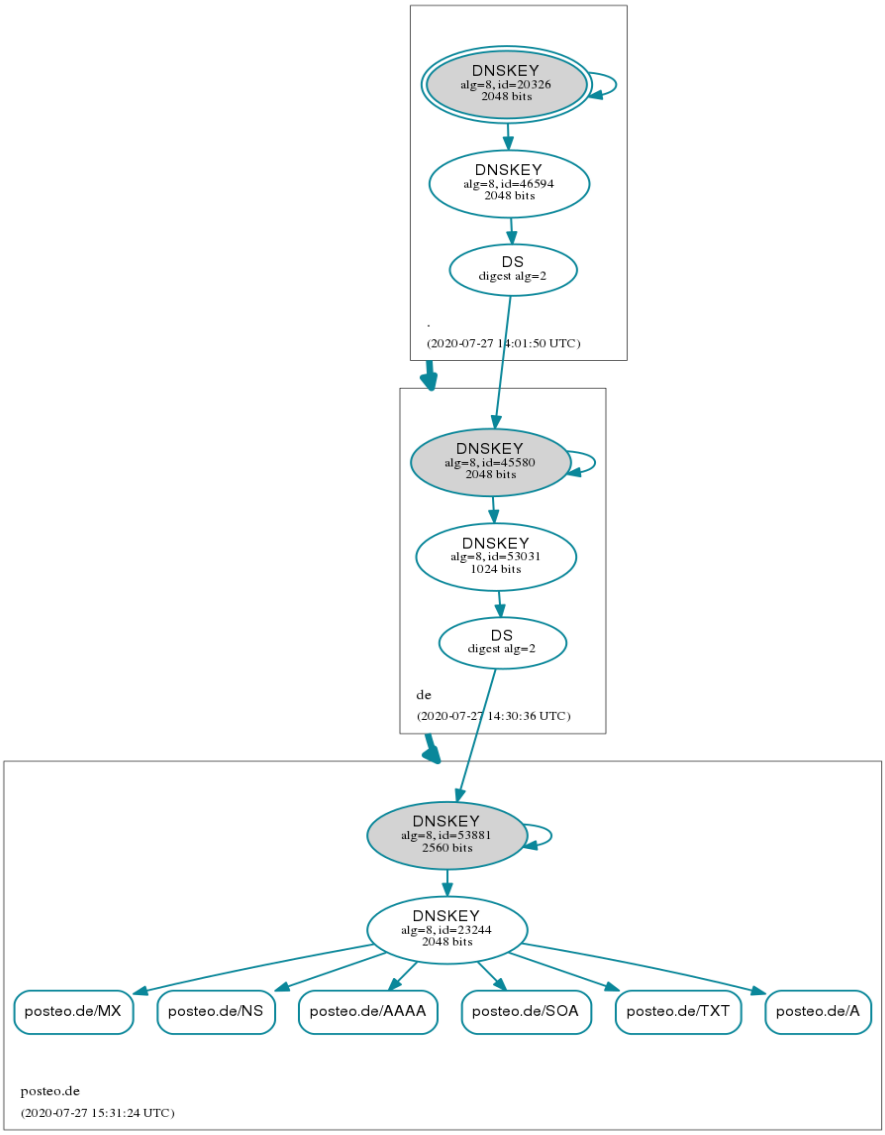
signatures to be computed off-line, and not upon request. This is important, both for performance and security. In order to allow clients to authenticate DNS data, each zone creates a signing and verification key pair, (private key and public key). The signing key is the private key and is used to sign the zone data, and should be kept secret and offline. DNSSEC supporting name servers return the requested RRs, along with the corresponding signatures (in a RRSIG RRs).

In order to mitigate replay attacks, each signature contains a fixed expiration date. The clients, i.e., resolvers, also gets the zone's public verification key, stored in a DNSKEY RR, which in turn is used by the clients to authenticate the origin and integrity of the DNS data. Resolvers contain a set of verification keys for specific zones, known as trust anchors. They have the trust anchor for the root zone. They get other verification keys (non trust anchors) by requesting a DNSKEY resource record from the domain. To validate the public keys obtained from DNSKEY, the resolver gets the corresponding DS RR from the parent zone, that has a hash of the public key of the child; the resolver verifies the DNSKEY to be authentic if the corresponding hashed value is same as the value in the DS record, and also that the DS record is properly signed (in the corresponding RRSIG record). As the DS record of the parent is also signed by a DNSKEY belonging to the parent, authenticity is carried forward with backward checking Creating a chain of trust allows the resolver to authenticate the public verification key of a target zone. Any change in the Key Signing Key also requires a change in the parent zone's DS record. But, changing the DS record is a sophisticated multi-step process that need to be performed correctly. The parent is needed to add the new DS record, then wait until the TTL for the original DS record expires before removing it. Finally, we can see that the clients authenticate the public key of the zone by making a chain of trust beginning at the root zone and ending at the end target zone.

For example, considering (Figure 1) showing the chain of trust implemented in the domain posteo.de, we see that, each zone excluding the posteo.de zone has a DS key that is used to sign the DNSKEY of the next sub zone. We see that the DS RR of the root zone "." is used to sign the DNSKEY of the next zone or top-level domain "de" and subsequently the DS of de is used to sign the DNSKEY of "posteo" sub zone.

Figure 1: Chain of trust implementation in posteo.de

Taken from https://dnsviz.net/d/posteo.de/dnssec/ using the tool DNSViz for live DNSSEC graphs on the current date. Accessed on 2020-07-27.



2.2.1 Authenticated Denial of Existence:

NX-Domain (non-existing domain) is the response made by a nameserver when a non-existing record is specified in a DNS request query. In order to authenticate such responses, DNSSEC does not send a signature over the non-existing domain name. This is done because it requires real-time signing, also negating DNSSEC's norms of only signing offline for better security and performance. In turn of signing each NX-Domain response, DNSSEC incorporates alternative authentication systems to permit a name server to prove the non-existence of a resource, thus the NSEC and NSEC3 record types. NSEC, specifies what type of records exist, where it resides and also points to the next domain name in the zone. [16]. It cryptographically allows to demonstrate that a resource record set does not exist, by creating a gap between two domain names in a zone. NSEC's downside is that it allows the discovery of all subdomains, as it points to

the next domain name. In order to fix this subdomain exposure vulnerability, NSEC3 as an alternative was proposed.[17]

NSEC3 is a chain of hashed names that prevent enumerability and also supports opt-out option, where only secure delegations have NSEC3 record. Opt-out NSEC3 permits better performance for large domains also shorter NSEC3 chains with less signatures and smaller files of signed zones.

Thus, the validation works as:

1. Request the desired RRset, which also returns the corresponding RRSIG record.
2. Request the DNSKEY records containing the public ZSK and public KSK, which also returns the RRSIG for the DNSKEY RRset.
3. Verify the RRSIG of the requested RRset with the public ZSK.
4. Verify the RRSIG of the DNSKEY RRset with the public KSK.

2.3 Resource Records types

DNSSEC adds numerous RRs. Below are the data formats with examples. We have taken as an example the resource records of posteo.de (an independent and a very secure email provider based in Berlin) :

2.3.1 DNSKEY Resource Record

DNSSEC public key (DNSKEY-record) holds the public key used to verify DNSSEC signatures in the corresponding RRSIG-records. DNSKEY-records are made of the following data elements:

Flags: "Zone Key" and "Secure Entry Point".

Protocol: Fixed to a value of 3 for backwards compatibility.

Algorithm: Cryptographic algorithm used to generate the public key.

Public key: Data of the public key.

```
;; QUESTION SECTION (1 record)
;; posteo.de.              IN          DNSKEY
;; ANSWER SECTION (3 records)
posteo.de.  300         IN          DNSKEY  ( 256 3 8
                AwEAAfF+q6BPXteBHRzPhEI3JRznhsI76cql2vfObU1e/EdbWQazph6zsIH4KVAuBJVhotgJCphE
```

0GAzESwG1RjaqNn3ioO8ceHw2OIfsn8eOOyku12+NfSNa2SOfM7c8y5rBaUD8eAfaqxgadL4sNDB

9oju9ChA10MkBDONnTdvuCgHDuuqGaIOG+FaBxQAx6Y6LTWmmGmtHoZNi30ubEkzd2JpC1rsn486

jTkaf9sb5SLCLRw4vYMdUIMBr0Ldliq/pWWHkkV5I7vOoKj31nQm58165YncHeCG/hPAN+b9iYLq

YkfNYB7ahCb82WhEKpw71agcStR0yhFA5BLcf3ywOVs=) ; Key ID = 23244

posteo.de. 300 IN DNSKEY (257 3 8

AwEAAaf1r7SIqmjxs+kjuXGt/hSwa1IZvMb2ed5zuALoSu80+rAB79lZWSiNCS2Y1MYkIu/OicbR

tfV15HEooaeLfduZyFgwWVJK7IgHdqRAkU/2pCuX4arQ8h9SPQLNTccFlH0aP+oAuZPfeCVlqFh0

fgMtfPdNOKnk49qClPuND+EWoWsowMyZT4yPF20Uv105tm6A5KRMmK7Lv+pbOZbmCpIfLwmAHcez

k+yNq8VF+J0kfzq1NgVt4Pds514x3Q2h8ZI4XkyJll36FwlvGOwZXs477fwsDUT8xMknzCbPY1hC

xkRYC8+X+hx6vezsLoSgVctllyS8U0q7MHRMrDPbEhNRcFYDqvxXyLN2hHYOU43WlDKmcUsxIpdw

GHh+tg4C/YLsNmWbI19Flf9kGbQqmKDnUkBhbdGcR7RKVaFIfvix) ; Key ID = 53881

2.3.2 RRSIG Resource Record

The RRSIG Resource Record - Digital signatures used in the DNSSEC authentication process by a validator on RRsets from a zone are stored in an RRSIG RR. It contains a particular name, class, type and validity interval along with the signature. Since every authoritative RRset must have a digital signature, RRSIG RRs should also have a CNAME RR. Example of an RRSIG record for a DS record for posteo.de is :

posteo.de. 86400 IN RRSIG (DS 8 2 86400 20200817031534 20200803014534 53031

de. abhgphq9YYO3W3kVhHyslP5n/QpCETzpZVQkiv6PJ566Pk7XktMa6L4dlJkuUyUCZDpRYpYGEB6x

s9+o5PNZeFl9IRvqluQe0Pm4x4iQo2aN47WvVTwHFtg+nlQXEhc4WX7W30Wpn5Wofh24NIwKThin

otszzdWcw3j1VD3LhIg=)

2.3.3 NSEC and NSEC3 Resource Record

The NSEC RR has the next owner name and the set of RR types of the NSEC RR's owner. Thus, it links to the next secure record name and also provides a list of types of records existing. An NSEC3 uses cryptographic hashes of record names to prevent zone enumeration. The next owner name (that has authoritative data or a delegation point NS RRset) should be in canonical order of the zone. The NSEC RRs in a zone portrays the authoritative RRsets existing in a zone and makes a chain of the authoritative owners of the said zone. This information can be used for providing authenticated denial of existence as per DNSSEC validation [20].

Furthermore, since NSEC chains have every authoritative name in a zone, NSEC RRs should be present against every CNAME RR. This is different to DNS specification that

state that if a CNAME or alias name is present for a name, it’s the only type allowed at that name. Thus an. See [20][21] for discussion of how a zone signer determines precisely which NSEC RRs it has to include in a zone. The type value for NSEC RR is 47 and is class is always independent. The NSEC RR must also possess the similar TTL value as that specified in a SOA minimum TTL field for purposes of negative caching.

NSEC3-records contains the Hash Algorithm used, Flags (“Opt-out" indicating if delegations are signed or unsigned), Iterations (times of hash algorithm application), Salt for the hash, Next Hashed Owner Name and Record Types.

2.3.4 NSEC3PARAM

Since an RRSIG and NSEC MUST coexist for the same name as does a CNAME resource record for a signed zone, an NSEC3PARAM-record has the hash method used for computing NSEC3 records.

NSEC3PARAM-records have the name Hash Algorithm Flags count of the number of Iterations of the hash algorithm and Salt value for the hash calculation. [21].

2.3.5 DS (Delegation Signer)

DS-records are used to securely reference a DNSKEY-record of a sub-delegated zone. They contain a Key Tag, Cryptographic algorithm of the referenced DNSKEY-record, Cryptographic hash algorithm type used to calculate the hash of the DNSKEY- record and the hash value of the referenced DNSKEY-record RFC4034[16].

2.4 DNSSEC-enabled use cases:

2.4.1 DANE

DNS-based Authentication of Named Entities is an Internet security protocol commonly used for Transport Layer Security (TLS), to be bound to domain names for Domain Name System Security Extensions. [22] In TLS connections Certificate Authorities (CA) issue verification certificates for checking authenticity of a website providers, but since this also requires putting trust on a large number of CA providers, it might cause problems upon breaches in any of the CAs. DANE solves this problem by enabling the administrator of a domain name to vouch for the keys used in that domain's TLS clients or servers by storing them in DNS records (TLSA). It needs the TLSA record to be signed with DNSSEC for its security model to work.

It also allows a domain owner to specify which CA to trust that solves a hard problem in TLS communications and helps to avert rogue CAs into issuing certificates without the consent of the domain holder. (TLS)(problems of TLS)

DANE also supports sharing of OPENPGP keys whilst also eradicating or minimizing chances of MitM attacks. Domains can publish their OpenPGP keys in the OPENPGPKEY Resource Record and clients can query using the method mentioned above to get the OpenPGP key, thus, avoiding sniffing attackers.

DANE is the most important use of DNSSEC as it uses the security features and dependability of the DNSSEC responses and also solves the problem of asking for public keys and being deviated by attackers into believing into false keys. As DANE works on DNSSEC, the authenticity of the key is maintained by the provider and any kind of key compromise can also be periodically changed and rotated. [23]

2.4.2 TLSA

TLSA records specify the keys or certificates of a domain's TLS servers that are used to certify the origin of a website using the DANE infrastructure.

They are identified with the following 3 parts:

- The port number of the TLS server.
- The protocol used (udp, tcp, sctp, or user defined).
- TLS Server host name

They have the following data elements with the below specified range:

- Certificate usage: 0-255
- Selector: 0-255
- Matching type: 0-255
- Certificate association data: Hexadecimal.

An example of a TLSA record of mailbox.org is as follows:

```
dig _443._tcp.mailbox.org TLSA +dnssec @8.8.8.8 +short

3 1 1 29681D7841E22492CC6BAC79698B59192D474CC36E413627D5F06060 766ED702

3 1 1 51E89750F7D9D0867C2840DC66D9F635E2844541A38EF7586DB59088 B846FE59

3 1 1 ECE6745A10D73A2A5B0DCCE5C319F0CFB281F48941CA548A9648F58E D1AB2A56
```

3 1 1 996AD31D65E03F038B8EC950F6F26611529DA03E3A283E4400CBA2ED D04B8A88

TLSA 7 4 3600 20200828171640 20200729170232 5719 mailbox.org.
Z6Y9vHXDKnAkp9z+wRVL0oVT2bIYE+ZavXuf/5SZQwiw25baFN7L+wYe
kJyeCEVWqW9YP3LJFxyzhW3D0kXt9/ZV8DusaSdE2SxwAvLnDtAwepFo
9FP5HIz0kqCKUCTPw/A2DUZQKgNsRdSa4itgxwjKw75LZXHwfGlwjNIt fdQ=

TLSA 10 4 3600 20200828171640 20200729170232 48028 mailbox.org.
o3PH4YnlvKuyC+fanvTHzq73ORIWfh6jfPDJRTB4tnocXbwYqTyKW0Ik
T2hrbPbdMubs+gTYltgo6p7u8fGu62MOjDxysCNO6eLJLNJHRnUjI8VP
L0NfQX+o984JYJR/kM6QwLOkULSK+bgea80UE9aoO0Ca8IX2uoQH1IMD aBw=

2.4.3 OPENPGPKEY

An OpenPGP key is a publicly transferable public key for encrypting email messages. It is used by email providers. The OPENPGPKEY RR has the data format as follows:

For an email “support@posteo.de” [36] in order to find the OPENPGPKEY RR, we have to find the sha256 hash dump of the string before the @ sign, i.e. “support”, append the hash value uptill 28 octets or 56 characters with ._openpgpkey.weberdns.de and query using dig. The response provides the openpgpkey within the OPENPGPKEY RR as:

dig faa … f227dc6a1._openpgpkey.posteo.de openpgpkey +multi +dnssec +noadditional +noauthority

; <<>> DiG 9.11.3-1ubuntu1.12-Ubuntu <<>>
faae1b97e57e3e121216948b8dda2a429ea72d6dd81c164f227dc6a1._openpgpkey.weberdns.de
openpgpkey +multi +dnssec +noadditional +noauthority

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55341

;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:

; EDNS: version: 0, flags: do; udp: 65494

; OPT=5: 05 07 08 0a 0d 0e 0f (".......")

; OPT=6: 01 02 04 ("...")

; OPT=7: 01 (".")

;; QUESTION SECTION:

;faae1b97e57e3e121216948b8d….a72d6dd81c164f227dc6a1._openpgpkey.posteo.de. IN OPENPGPKEY

;; ANSWER SECTION:

faae1b97e57e3e121216948b8dda2a429ea72d6dd81c164f227dc6a1._openpgpkey.posteo.de. 3600 IN OPENPGPKEY (mQINBFnWqkYBEAC7zGWHvK87anwlSTpx7L3R80IQ0op3 …..

1dxktJ2ENCwlaor8w)

(The response is shortened to fit the document)

2.5 Protocol modifications necessary for DNSSEC implementation

DNSSEC signed zones includes DNS Public Key (DNSKEY), Resource Record Signature (RRSIG), Next Secure (NSEC), and (optionally) Delegation Signer (DS) records as new records for implementation of DNSSEC. It also adds Checking Disabled (CD) and Authenticated Data (AD) message header bits. Furthermore, in order to support the larger DNS message sizes due to the newly added DNSSEC RRs, DNSSEC also needs EDNS0 support. It requires support for the DNSSEC OK (DO bit) EDNS header bit in order to enable a security aware resolver to indicate that it wishes to receive DNSSEC RRs in response messages through its queries. [9][20]

2.5.1 Authenticating Name and Type Non-Existence

The NSEC record allows a security-aware resolver to authenticate a negative reply for either name or type non-existence with the same mechanisms used to authenticate other DNS replies, explained previously.

2.5.2 Including DNSKEY RR

The zone's administrator generates one or more public/private key pairs and uses the private key(s) to sign RRsets and to create RRSIG RRs in a zone. The private key is stored away from the internet while the public key is published as the DNSKEY RR.

2.5.3 Including RSIG RR

For each authoritative RRset in a signed zone, there MUST be at least one RRSIG record that meets the following requirements:

- The RRSIG owner name, class, type and TTL field should be equal to the RRset owner name, class, type and TTL respectively.

- The Labels field in RRSIG is equal to the number of labels in the RRset, excluding the the null root label and the leftmost label in case it is a wildcard.

- The RRSIG Signer's Name field is equal to the name of the RRset containing zone.

- The RRSIG Algorithm, Signer's Name, and Key Tag fields identify a zone key DNSKEY record at the zone apex.

2.5.4 Including NSEC RR

For each owner name in the zone that has a delegation point NS RRset should also contain an NSEC resource record. [16]
The TTL value for the NSEC RR SHOULD be the same as the minimum TTL value field in the zone SOA RR and the type bitmap of every NSEC resource record in a signed zone also should indicate the presence of both the NSEC record the corresponding RRSIG record.

2.5.5 Including DS RR

As the DS resource record establishes authentication chains between DNS zone, the DS RRset must be present at a delegation point of child signing. All DS RRsets in a zone MUST be signed, should not appear at the zone’s apex should point to a DNSKEY RR of the child its TTL is to be same as the TTL of the delegating NS RRset.

2.5.6 CNAME record changes

If a CNAME RRset is present at a name in a signed zone, appropriate RRSIG and NSEC RRsets are REQUIRED at that name. This is a modification to the original CNAME definition given in RFC1034 [24]. The original definition of the CNAME RR did not allow any other types to coexist with a CNAME record, but a signed zone requires NSEC and RRSIG RRs for every authoritative name. To resolve this conflict, this specification modifies the definition of the CNAME resource record to allow it to coexist with NSEC and RRSIG RRs. An RRSIG RR MUST have the same class as the RRset it covers. The TTL value of an RRSIG RR MUST match the TTL value of the RRset it covers.

A security-aware resolver MUST include an EDNS OPT pseudo-RR with the DO bit set when sending queries. A security-aware resolver MUST support a message size of at least 1220 octets, SHOULD support a message size of 4000 octets, and MUST use the "sender's UDP payload size" field in the EDNS OPT pseudo-RR to advertise the message size that it is willing to accept. A security-aware resolver's IP layer MUST handle fragmented UDP packets correctly regardless of whether any such fragmented packets were received via IPv4 or IPv6.

2.5.7 The DO Bit

The resolver side of a security-aware recursive name server MUST set the DO bit when sending requests, regardless of the state of the DO bit in the initiating request received by the name server side.

2.5.8 The CD Bit

The CD bit exists in order to allow a security-aware resolver to disable signature validation in a security-aware name server's processing of a particular query. The CD bit must be copied from the query to the CD bit to the resolver side along with the rest of the initiating query, so that the resolver side is “willing to perform whatever authentication its local policy requires”.[20]

2.5.9 The AD Bit

The name server side SHOULD set the AD bit if and only if the resolver side considers all RRsets in the Answer section and any relevant negative response RRs in the

Authority section to be authentic by confirming the chain of trust to be intact. However, for backward compatibility, a recursive name server MAY set the AD bit when a response includes unsigned CNAME RRs.

2.5.10 RRset in the Answer section

The RRSIG RRs have a higher priority for inclusion than any other RRsets that can be needed to be included. But in case of space problem, the name server MUST set the TC bit. It is also same for the Authority section of the response, but the Additional section requires the name server to NOT set the TC bit solely because these RRSIG RRs didn't fit.

2.5.11 Include DNSKEY RRs in response

With the DO bit set and requesting the SOA or NS RRs at the apex of a signed zone, a security-aware authoritative name server for that zone can return the zone apex DNSKEY RRset in the Additional section. Furthermore, in this situation, the DNSKEY RRset and associated RRSIG RRs have lower priority additional section items. The name server should also calculate the space required for the DNSKEY RRset and its corresponding RRSIG RR(s) and availability in the response message or must not include them and also is not supposed to set the TC bit solely because these RRs didn't fit.

2.5.12 Including NSEC RRs in response

With the DO bit set, a security-aware authoritative name server, for a signed zone, is not supposed to include NSEC RRs in the occurrence of No Data, Name Error, Wildcard Answer and Wildcard having No Data. Also, in case of space un-availability of these NSEC and RRSIG RRs, the name server should set the TC bit.

2.5.13 Including DS RRs

In case a DS RRset is present at the delegation point, the name server is supposed to give both the DS RRset and its associated RRSIG RR(s) in the Authority section along with the NS RRset. Otherwise when no DS RRset is present at the delegation point, the name server is supposed to give both the NSEC RR that proves that the DS RRset is not present and the NSEC RR's associated RRSIG RR(s) along with the NS RRset by placing the NS RRset before the NSEC RRset and its associated RRSIG RR(s). We can

see that the inclusion of these DS, NSEC, and RRSIG RRs increases the size of referral messages and may cause some or all glue RRs to be omitted, thus in case of space in case of space un-availability, the name server MUST set the TC bit

2.6 Problems for Building, Operational Measures and Practices

DNSSEC deployment has several difficulties:

1. Backward-compatibility should be present
2. “zone enumeration” should be prevented
3. DNSSEC should be deployed over wide variety of DNS servers and resolvers
4. Disagreement in ownership of TLD root keys.
5. Overcoming the complex perception of DNSSEC and its deployment [1]

2.6.1 Deployment and interoperability

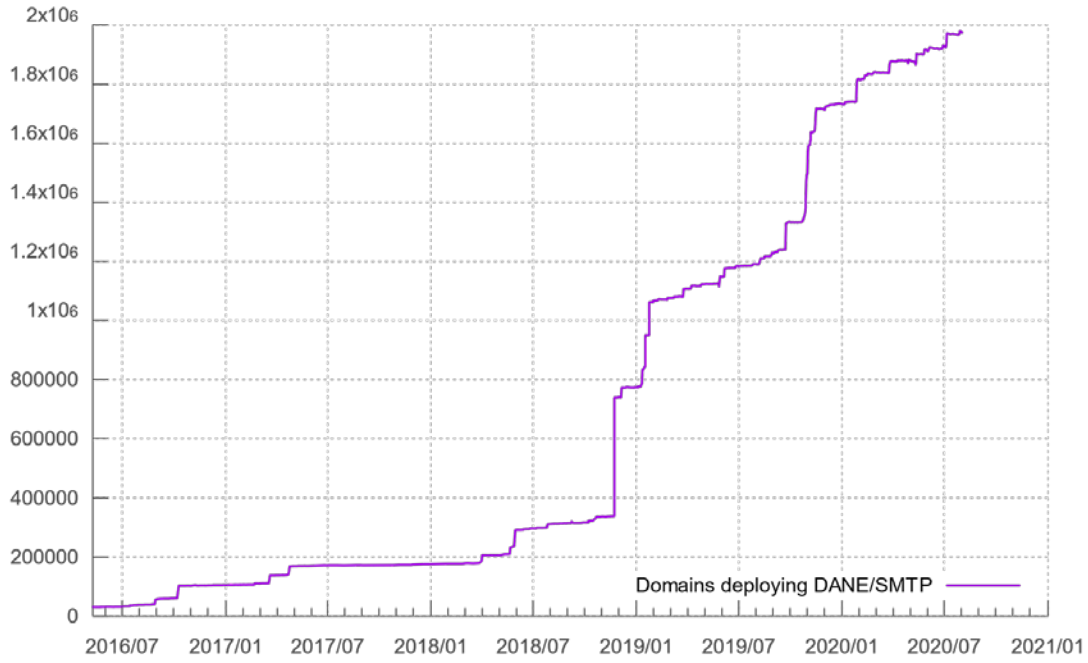
There are a lot of DNS resolvers that still do not support DNSSEC validation, and many domains are not signed with DNSSEC. Figure 2 shows all the current statistics of DNSSEC Deployment at Zones. It shows in July 2020 that the no of DS Resource Records sets present are around 12 million and the DNSKEYs that are currently working as around 11 million. Thus we can infer that some subdomains still don not have DNSSEC implemented. Also we can see in Figure 3 the current DANE implementations. We can infer from the graph that in the initial phase, due to multiple adaptability nuances in the minds of DNS providers, [1] there was a slow rate of adaptability. Currently it shows a near linear rise of adaptability. Thus a promising future adaptation.

Figure 2: DNSSEC Statistics in July 2020

Accessed from https://stats.dnssec-tools.org/ in July 2020, showing the current statistics of DNSSEC deployment.

Figure 3: Domains deploying DANE/SMTP

Accessed from https://stats.dnssec-tools.org/ in July 2020, showing the current statistics of DANE/SMTP deployment.



2.6.2 Interoperability Problems

Devices that are between the resolver and the name server can forestall DNSSEC administration, by blocking EDNS queries and responses, blocking long and/or fragmented responses, or expelling DNSSEC records from DNS responses.[1][10] This is due to the between DNSSEC enabled DNS responses and ‘legacy’ DNS responses, support for EDNS, special RR types, and long responses there are such interoperability issues. [10]

Now the question arises as to why are DNSSEC responses so much longer, and what are the other added differences? The reason is that, while sending a DNS request the client adds an EDNS OPT pseudo RR to the query, and sets the DO bit, indicating support of DNSSEC. Also, it includes a multiple signing and verification keys. The problem of such oversized DNSSEC responses are that DNS responses are normally sent over UDP and long DNS responses over UDP get fragmented when they cross the maximal transmission unit (MTU) of the path. Fragmentation is also known to be vulnerable to attacks. In [26] it is shown that an attacker can send mal-constructed fragments, tricking the defragmentation mechanism into reassembling them together with the fragments of legitimate sources Also in [26], it is shown that this does not need eavesdropping or MitM capabilities. Thus, due to the currently-common case of permissive validation, as DNS poisoning only requires a valid DNS response, with the correct port the attacker

can send a spoofed second fragment, combining it with the legitimate first fragment, to successfully perform DNS poisoning; see [26]. To avoid interoperability problems due to long responses, domains use shorter public keys and signatures. [10] Thus, elliptic curve DSA, was standardized for DNSSEC [27]; however, this algorithm is implemented only by a negligible number of domains as given by the study in 2013 (0.002%) [10]. Cloudflare [28] claims to have solved the two challenges of zone enumeration and DDos attacks by using ECDSA. Algorithm 13, algorithm 16, etc. It is also recommended in (towards) that the DNSSEC community should consider developing cipher suite negotiation to allow domains to use multiple keys and algorithms like in TLS connections(tls). Furthermore, some DNS proxies may also not support EDNS and/or not support or drop the DNSSEC record types [21] thus resulting in responses not reaching the client. Due to concerns about interoperability issues, many resolvers that support DNSSEC, allow unvalidated responses, thereby allowing downgrade attack [20]. A downgrade attack is where an attacker sends fake responses that appear similar to responses passing through non-interoperable devices. As an example, let’s consider an attacker that poisons the NS and glue A RRs, and provides a forged delegation NS RR to redirect the clients to a host of attacker’s choosing. The NS and A RRs are defined to be the child zone’s authoritative data and they are kept unsigned in the parent zone. Even if the attacker does not have the secret signing key of the child and cannot forge signatures on the DNS data, he or she can go on with the attack as when the resolver will revert to plain DNS without DNSSEC protection.

2.6.3 Anchors of trust

DNSSEC ensures authenticity and integrity of DNS Responses. The underlying condition is that all the nodes on the path from the root to the parent zone support DNSSEC and are trusted, but, if any one of the parent zones on the path does not have DNSSEC deployed, the clients would not be able to construct a chain of trust. Thus, even if the child zone supports DNSSEC, its clients are still exposed to DNS cache-poisoning attacks. And it being in an Island of Security. Furthermore, poisoning attacks on clients will be possible by a corrupt parent as without DNSSEC, the connection returns to normal DNS and has the vulnerability of corrupt parent attack mentioned below in details. Corrupt Parent Attack DNS has two types of trust relationships: an inter-organizational relationship between the top level domain (TLD) and its children like com and example.com and an intra-organizational relationship like example.com and foo.example.com , within an organization. DNSSEC does not distinguish between

the two types of hierarchies. The parent zone is assumed to delegate the DS RR and the name server (NS) resource record (RR) of the child, along with the associated IP address (A RR). Such a trust model is appropriate for zones within the same organization, like example.com and foo.example.com, but , risk is involved while trusting the parent not to abuse inter-organizational trust relationship. The parent zone can forge the delegation records of the child and then provide this forged (yet correctly signed) DS RR. Thus misleading the clients of the child zone into accepting the forged DNSKEY and trusting the resource records signed with it . This is called as the corrupt parent attack. [1] This attack can be mitigated with additional mechanisms to validate the domain keys. It also resembles the issue of islands of security. Trust anchor repositories (TARs) is recommended to mitigate both these problems. Trust Anchor Repositories (TARs) TARs are recommended to be used as an alternative to establish chain of trust and to verify keys of islands of security. TARs are designed to maintain the trust anchors and to support early adopters and to speed up deployment of DNSSEC. They can also be used to address the inter-organizational trust issues. In order to publish trust anchors the zones should create a DLV resource records (RRs) into the DLV repository. The domain some-domain.tld should configure a DLV record for a hash of its key mapped to somedomain.tld.dlv.isc.org. Resolvers would first try to follow the standard DNS tree, when the process does not result in a chain-of-trust, the resolver should query the DLV from the DLV record.

2.7 Vulnerable DNSSEC configurations

In [1] two issues have been discussed that have supposedly introduced vulnerabilities in the protection offered by DNSSEC. They are as follows:

2.7.1 Pitfalls of Inter-domain Dependencies

The discussion mainly revolves around Inter-domain dependencies. The RRs that are used are NS, MX and CNAME records. Inter-domain dependencies may curb the security effectiveness offered by DNSSEC against cache poisoning. This is especially thought of happening during incremental deployment, i.e. when DNSSEC is only partially implemented in domains.

As discussed, it the (paper) we now point out the vulnerabilities probable due to the different records.

1) Dependency via NS records: As an example, it is explained in the paper as let's suppose, to search for a subdomain e.g., foo.bar, a resolver query to the nameserver ns.bar of the parent domain (bar). Also, the nameserver responds with a reference of the name server(s) of foo.bar, ns1.foo.bar. But it is also possible that the name server of foo.bar is in a separate domain like let's assume fie.baz as ns1.fie.baz. Thus, even if the parent domain bar and foo.bar uses DNSSEC, the referral sent by ns.bar is unsigned, as in DNSSEC name servers provide signature for its own domains. Bernstein[31] Observed that an MitM attacker can spoof these referral responses, but still possibility of poisoning of DNS queries cannot be possible as in DNSSEC, resolvers do not follow the delegation responses that are unsigned by the DS record of the parent, i.e. without the chain of trust, resolvers do not coincide to the request. This is not the case when name servers in a domain not protected by DNSSEC, and thus the vulnerability may still exist.

   In addition, when there is a name server in a different domain, the mapping has some security implications as it can be controlled by a third-party administrator of baz and fie.baz. Also, if the domain has its name server(s) in a different domain like foe.baz then domain foo.bar also depends on and trusts the domain fee. This situation is called as DNS transitive property and poses a threat to DNSSEC security properties.

2) Dependency via CNAME to Unsigned Domain: CNAME is normally used in mapping of services, like, ftp or web servers, to different names. This comes handful for web hosting, content delivery networks (CDNs), etc.

   The paper [1] has used an example from data found on paypal.com in 2015 by the authors from some response packets from a Wireshark run. It is seen that paypal.com has DNSSEC enabled and the zone PayPal provides properly signed keys, but still, the client querying paypal.com has to travel a lengthy way through the CNAME chain Consider again paypal.com. Also it is seen that only the first CNAME delegation is signed, but the other four are not, therefore, even though the domain www.paypal.com is signed, the IP address 23.44.242.234 (which is the actual one) is unsigned. This again enables MitM attackers to mess with the mappings and change them to a different IP address, and it will also get accepted by resolvers. Thus, DNS cache poisoning attacks are possible.

3) Dependency via MX to Unsigned Domain: An unsigned dependency on a domain via an MX record, can also allow cache poisoning attacks on the mail server. Different other range of attacks, including credentials theft which can be done by exploiting password recovery via email as proposed by Kaminsky is also deemed probable [1].

2.7.2 Pitfalls of Non-Deniability of Existence

Bernstein, [31], pointed out that the NSEC is has possibility of zone enumeration that can be done zone walking the NSEC chain. In order to fix the zone walking NSEC3, as an alternative was proposed [17], which is a chain of hashed names that was said could probably prevent enumerability. Subsequently Bernstien [31] also proved that NSEC3 also can leak information and allow zone walking

2.7.3 Attacks on oversized DNSSEC responses

As we have discussed above that DNSSEC queries require Extensions to DNS i.e., EDNS to send cryptographic keys and/or signatures and extra RRs. In order to accompany this, generally a packet greater than that of normal DNS UDP packets need to be sent (greater than 512 bytes). These responses are used by attackers to launch an amplification DoS attacks. It is done generally by sending to one or more DNS servers a large number of requests, with spoofed source IP address. Name servers on receiving such requests serve by sending a much longer response to the spoofed source IP address mentioned by the attacker. The amplification factor is calculated by the ratio of the number of response bytes sent by the attacking DNS server, to the number of request bytes sent from the hosts controlled by the attacker. Mostly the victim is a network or host that is on the receiving end.

Other defenses include:

1. A maximal quota can be imposed on queries from the same source IP address. But this does not work against attacks on the amplifying servers itself, also it fails when there are too many spoofed IP addresses from the same victim network, as mentioned in the Coremelt attack [1].
2. Challenge-response mechanisms can also be used. They redirect the request to an another domain name, where random challenge (‘nonce’) is used to authenticate that the response is not from a spoofed IP.

3. Use of TCP is recommended instead of UDP as done in top-level domains.

Notably, DNSSEC is also not needed to be implemented for this attack.

In Large top-level domains like com and net, servers refrain from the usage of UDP, instead they use TCP to send long responses. TCP is more secure as attackers cannot complete the three-way handshake using spoofed source IP address; however, the use of TCP poses different challenges mentioned in. This is done so that the servers do not exceed the MTU of networks in the path from the name server to the resolver, and hence have fragmented packets which an attacker can exploit by tricking the defragmentation mechanism into reassembling them together causing a fragmentation attack discussed above. [27] The oversized vulnerability exists for the use of RSA algorithms in DNSSEC. This is due to longer keys. The use of ECDSA shortens the keys and thus removes the problem. It is still a myth that DNSSEC still suffers from this problem, but since the use of ECDSA, the problem is averted.

2.8 Key Generation and Rollover mechanisms

Cryptographic keys are very sensitive as in case of their leak, a valuable target is jeopardized. Thus, to ensure continuity, keys are changed from time to time. Currently there are different algorithms used for Key Generation mostly RSASHA256, RSA-SHA1, and ECDSA. The use of ECDSA is a trend during the writing of this document as, it has a lower key size that help in averting challenges and vulnerabilities faced in case of longer key sizes.

The hard problem is to keep the UDP packets less than the MTU to ensure packet fragmentation is not done and thus avert rouge attackers, more about this attack is in section 2.7.3. The keys are recommended [9] to be generated offline and isolated from the network via an air gap or, at a minimum, high-level secure hardware.

2.8.1 ZSK rollover

2.8.1.1 Pre-Publish Key Rollover

In this case rollover of keys of the zone can be done without the need to sign all the data in a zone twice – the “pre-publish key rollover” with the added advantage in case of key compromise. For, if the old key is compromised somehow, the new key would already had been distributed.

Steps:

Initially the old DNSKEY TTL is realized, in order to start the process of key rollover.

Creating of new DNSKEY before expiry of the old DNSKEY occurs.

New RRSIGs keys are then generated with the new DNSKEY while the old DNSKEY still remains in circulation.

Old DNSKEY is removed and the KSK of the zone is used to sign the new DNSKEY, thus finishing the process of key rollover.

2.8.1.2 Double Signature Zone Signing Key Rollover

The steps involved in this type of rollover are:

1. Initially the old DNSKEY TTL is realized, in order to start the process of key rollover.
2. Creating of new DNSKEY before expiry of previous occurs and the RRs are signed with the new DNSKEY and also the old one. Thus a double signing occurs. This produces two RRSIGs one already existing and a new one. The new RRSIGSs are made from the new key, while the old DNSKEY still remains in circulation
3. The old DNSKEY is removed, and the KSK signs the new DNSKEY, thus finishing the steps of rollover.

The comparison of the above two methods are:

Pre-publish key rollover: This rollover does not involve signing of the zone data twice, instead, before the final rollover, the new key is already published and available for crypt analysis attacks. This process also requires four steps one step more than the other. Pre- publish scheme also requires more work from the parent when used in case of KSK rollovers explained later.

Double signature ZSK rollover: The drawback of this scheme is that during the rollover there are double the times of signatures present. This is a huge problem for huge zones. The only advantage is that it has only three steps.

2.8.2 KSK rollover

For a Key Signing Key rollover the following steps can be followed for a Deployment for a Double Signature Key Signing Key Rollover:

1. Initially the old DS TTL of the parent is realized by the child. The child verifies the TTL and starts the process.
2. New DS Key is generated for the child by the parent, but is not pointed to the DNSKEY of the child. New DNSKEY is generated by the child, but the child has to wait until the new DS of the parent points to the newly generated DNSKEY. It waits for the signing of the new key by the parent's newly generated DS key. The child waits then for the expiry of the old DS TTL. New RRSIGs of the child zone is also generated while creating the new DNSKEY as in the above rollover process of ZSK rollover.
3. DS change: The parent replaces old DS with new DS
4. DNSKEY removal: Old DNSKEY is removed.

One of the main drawbacks of this scenario is that the scheme has phase of new DS when the parent cannot verify the match between the new DS RR and new DNSKEY RR using DNS as the new DNSKEY is not published. Furthermore, the child in this case bears all the responsibility for maintaining a valid chain of trust. It is also based on the assumption that the parent has only one DS RR. Rfc (6781)

2.8.3 DS signature validity period:

The maximum signature validity period of the DS record is dependent upon the fact that how long a child zone is able to be vulnerable after a key compromise. IF it is shortened then, there are operational risks on the parent. It is a compromise between the operational problems and damage minimizing for the child. This may result in the validity period to be around a week and a month.

2.9 DNSSEC states

A validating resolver is able to determine the following four states of DNSSEC implementations:

Secure: In this case the resolver has a trust anchor and also a consistent chain of trust and no breaks in the chain of DS records, also is able to verify it.

Insecure: The case is when the validating resolver has a trust anchor, a chain of trust and at some delegation point in the chain of trust, a signed proof of a non-existence of a DS record, i.e., break in the chain of delegation. Thus, making the subsequent chain insecure.

Bogus: IF the validation of chain of trust fails at some point even though with the presence of a trust anchor and secure delegation it a Bogus state. It can happen due to missing, expired or presence of unsupported algorithms in signatures, etc.

Intermediate: When no trust anchor is present, such a state is posed, this is the default operation mode.

2.10 DNSSEC Algorithm numbers

DNSSEC [32] has specific number defined for algorithms used for implementation of DNSSEC. Currently as mentioned above ECDSA is mostly used, we can see from the table that the number of the said algorithm is 13 Also from Figure 4 and 5 below we can see that the number of users of algorithm 13 is more than the others. Other notable algorithms are 7 and 8, but due to size constrains and also security constraints they got replaced by algorithm 13. The other algorithms though specified in some RFCs is not used and were dismissed after a certain time. DNSSEC currently does not have any cipher suite negotiations, thus a proper algorithm should be there that is widely accepted.

Figure 4: Algorithm usage in DNSSEC

Accessed from https://stats.dnssec-tools.org/ in July 2020, showing the current statistics of Algorithm usage in KSK of DNSSEC deployments over the world.

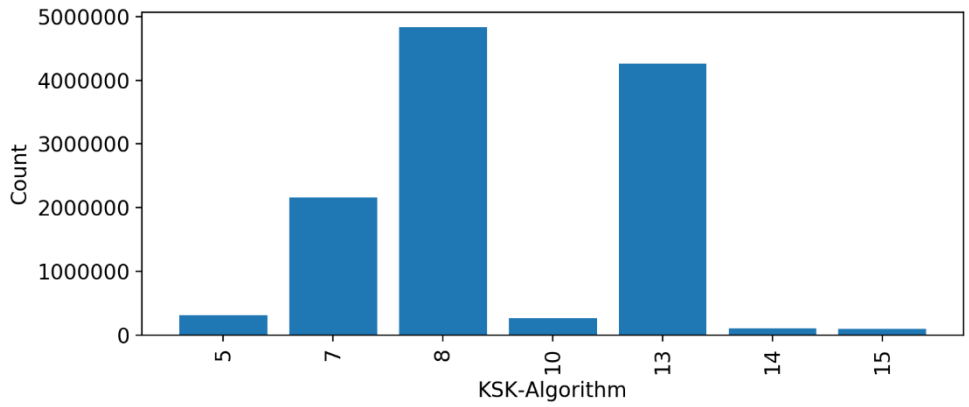


Figure 5: Algorithm usage in ZSK in DNSSEC

Accessed from https://stats.dnssec-tools.org/ in July 2020, showing the current statistics of Algorithm usage in ZSK of DNSSEC deployments over the world.

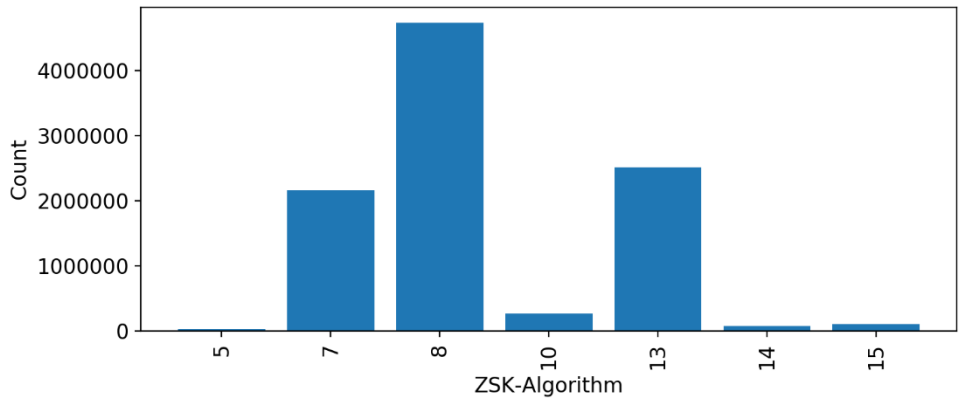


Table 5: DNSSEC Algorithm numbers

Accessed from https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml in July 2020, showing the number and names of Algorithms used for DNSSEC implementations.

Number	**Description**	**Mnemonic**	**ZoneSigning**	**Reference**
0	Delete DS	DELETE	N	[RFC4034]
1	RSA/MD5 (deprecated, see 5)	RSAMD5	N	[RFC3110]
2	Diffie-Hellman	DH	N	[RFC2539]
3	DSA/SHA1	DSA	Y	[RFC3755]
4	Reserved			[RFC6725]
5	RSA/SHA-1	RSASHA1	Y	[RFC3110]
6	DSA-NSEC3-SHA1	DSA-NSEC3-SHA1	Y	[RFC5155]
7	RSASHA1-NSEC3-SHA1	RSASHA1-NSEC3-SHA1	Y	[RFC5155]
8	RSA/SHA-256	RSASHA256	Y	[RFC5702]

9	Reserved			[RFC6725]
10	RSA/SHA-512	RSASHA512	Y	[RFC5702]
11	Reserved			[RFC6725]
12	GOST R 34.10-2001	ECC-GOST	Y	[RFC5933]
13	ECDSA Curve P-256 with SHA-256	ECDSAP256SHA256	Y	[RFC6605]
14	ECDSA Curve P-384 with SHA-384	ECDSAP384SHA384	Y	[RFC6605]
15	Ed25519	ED25519	Y	[RFC8080]
16	Ed448	ED448	Y	[RFC8080]
17-122	Unassigned			
123-251	Reserved			[RFC4034]
252	Reserved for Indirect Keys	INDIRECT	N	[RFC4034]
253	private algorithm	PRIVATEDNS	Y	[RFC4034]
254	private algorithm OID	PRIVATEOID	Y	[RFC4034]
255	Reserved			[RFC4034]

Chapter 3: deSEC

deSEC is a free DNS hosting service deployed in May 2020 in Berlin. It is an open-source software supported by Secure Systems Engineering GmbH (SSE). Designed with security in mind, and supported by a diligent development crew from SSE, deSEC has a worldwide anycast network of high-performance frontend DNS servers located in Europe, USA, Chile, Brazil, South Africa, Hong Kong, Singapore, and Sydney. Thus, having a global footprint and also seamless services. Most of the above servers are supported by SSE. As mentioned in the website desec.io, the eye-catching specifications provided by deSEC can be listed as the following:

1. DNSSEC: deSEC implements DNSSEC for all domains hosted through it
2. IPv6: deSEC is completely aware of AAAA-records and supports IPv6 addresses. Their name servers are compatible and reachable via IPv6.
3. DANE / TLSA: deSEC supports DANE implementation through addition of TLSA records, thus hardening domains against fraudulently issued SSL certificates and untrusted CAs. OPENPGPKEY key exchanges are also supported for emails through deSEC.
4. REST API: deSEC supports modern implementations with REST API which can be used in scripts as well as CI/CD pipelines.
5. Low Minimum TTL: deSEC has a fast update framework and publishes DNSSEC and DNS information in a smooth and fast manner (~ seconds).
6. Open Source: deSEC is a free and open source software.
7. Non-Profit: deSEC is a non-profit organization based in Berlin. All operations, statuses and guidelines are abiding to the German law and can be found in https://desec.io/about/

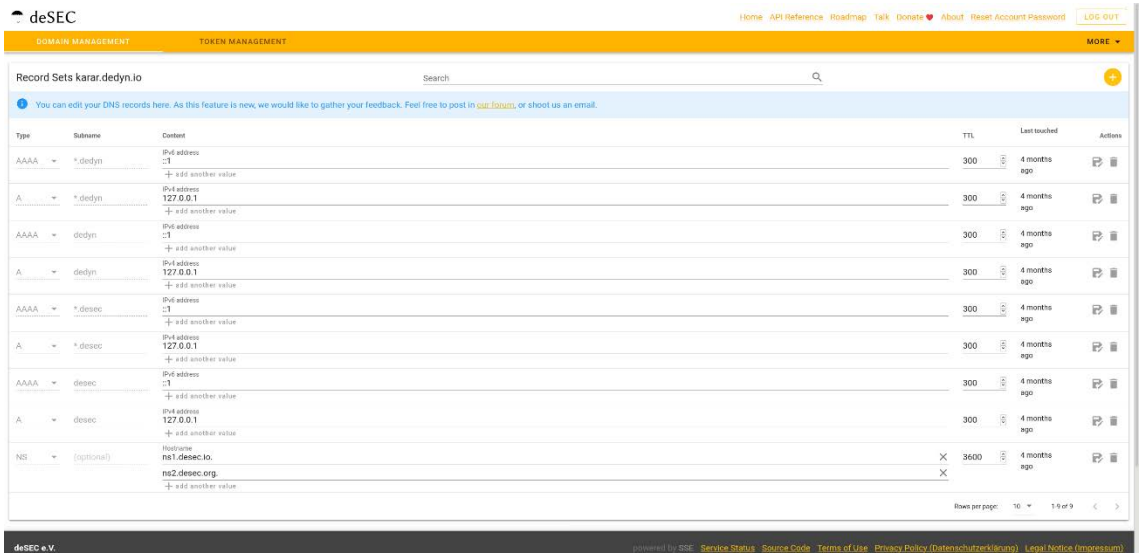
The goal of deSEC and its creators, Nils Wisol and Dr Peter Thomassen is to achieve large-scale adoption of IT security techniques such as the automatic use of cryptography in order to guarantee connection authenticity, confidentiality and

privacy for the general public. They also support research in the field of secure communications.

deSEC uses PowerDNS as the backend DNS software. PowerDNS, [33] founded in the late 1990s, with their Authoritative Server, Recursor and dnsdist products are 100% open source. deSEC uses the Authoritative Server and dnsdist as a high-performance load balancer in front of its secondary servers.

deSEC has an interactive web interface as shown in figure 6 that facilitates the addition, and changes of DNS records. This helps in a modern way of handling of DNS hosting.

Figure 6: DNS resource record status and adding page in deSEC website



3.1 DNSSEC implementation comparison between different domains:

In order to do the following comparative study, the tool form VerisignLabs[34] is used. Also, for visualizing purposes, another web-based interface from DNSViz [35] is used.

Table 6: Comparison between different domains according to thier DNSSEC implementation

Website	No of DS RRs in parent zone	DS algorithm used	DNSKEYs	DNS providers
desec.io	2	Both RSASHA256	2	deSEC
dedyn.io	2	Both RSASHA256	2	deSEC

publicsuffix.zone	1	ECDSAP256SHA256	1	deSEC
securesystems.dev	1	ECDSAP256SHA256	1	deSEC
denic.de	1	RSASHA256	3	DENIC
icann.org	10	All have RSASHA1-NSEC3-SHA1	5	ICANN
posteo.de	1	RSASHA256	2	Posteo
internetsociety.org	2	Both RSASHA1	3	Internet Society
dnssec-failed.org	2	Both RSASHA1	2	comcast

3.2 Inferences:

As we can see that the desec.io uses DNSSEC algorithm 8 which corresponds to RSASHA256 (table 5) but according to the RFC 8624 even though it is stronger algorithm than that deployed by icann.org (algorithm7 RSASHA1-NSEC3-SHA1) is slowly being replaced by ECDSAP256SHA256(algorithm 13) for shorter key and signature size for shorter DNS packages. Domains created at deSEC since 2017 use ECDSA. This is done to avoid the attacks mentioned due to fragmentation in chapter 2.

Furthermore, we can see that the domains hosted by deSEC are very updated in the use of algorithms in the comparison list with the algorithm for DS records, as they use Algorithm 13 (i.e. ECDSA) and thus add to a more secure backbone for connections to the hosted websites.

In some domains hosted by deSEC we can see the presence of a single DS record in the parent zone, this is the use of CSK instead of both KSK and ZSK.

We can also compare with the responses recorded between dnssec-failed.org, figure 7 (an example of broken implementation of DNSSEC) and securesystems.dev, figure 8 (a deSEC hosted domain) that there is proper DNSSEC implementation with the chain of trust being maintained.

Figure 7: DNSSEC implementation of dnssec-failed.org

Taken from https://dnsviz.net/d/dnssec-failed.org/dnssec/ using the tool DNSViz for live DNSSEC graphs on the current date. Accessed on 2020-07-27.

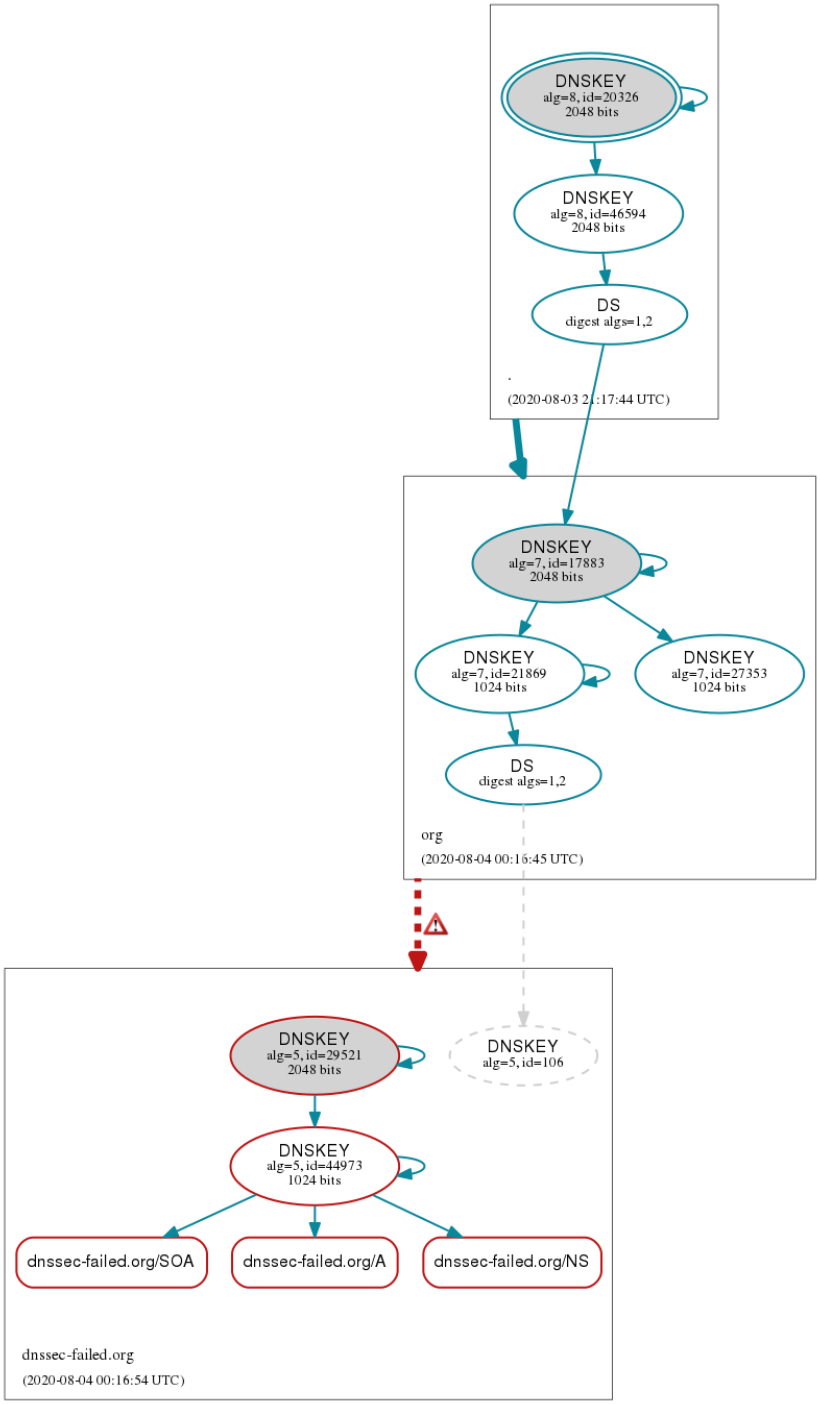
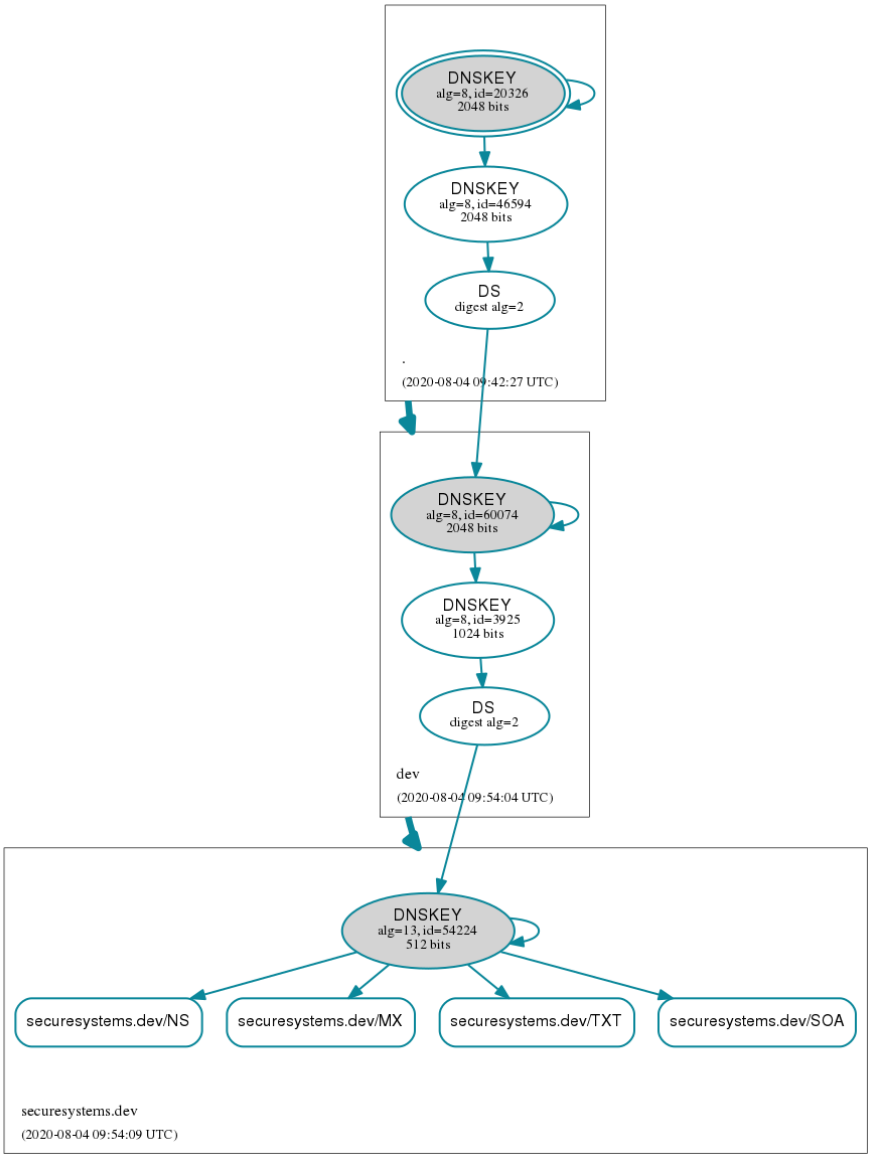


Figure 8: DNSSEC implementation of securesystems.dev

Taken from https://dnsviz.net/d/securesystems.dev/dnssec/ using the tool DNSViz for live DNSSEC graphs on the current date. Accessed on 2020-07-27.



Conclusion

Firstly, we have seen a number of challenges that legacy DNS faces in terms of security which can be minimized by the use of DNSSEC. But since it is a complex system involving multiple key rollover mechanisms and requires extra knowledge on cryptoanalysis, DNSSEC saw a very slow growth in the initial stages. Still a long way to cover, but looking far more promising with the stats in Figure 3 we can agree to the point that DNSSEC is being accepted slowly but surely. Even though we have multiple vulnerabilities still present in DNSSEC systems, one can argue that DNSSEC provides added security, privacy, authenticity and transparency of data transmission through the Internet.

Secondly, we have discussed the major procedures with short overviews of the operational best measures, like key rollovers, also have known several DNSSEC metric and implementation tools like Verisign labs and DNSViz etc., that give us a brief understanding of the current worldwide implementation and also deeper insights and future prospects of DNSSEC. We also have seen some practical implementations like posteo.de that uses DNSSEC for being a secure email provider.

Last but not the least, we did qualitative research on the tool deSEC that provides us with answers to our questions regarding industry standards, best practices compliance and various other security questions. Our findings and research procedures have made us made the conclusion that deSEC is in a promising position in the market of DANE and DNSSEC with its modern API and web views and also added security.

References

[1] Herzberg, Amir & Shulman, Haya. (2013). DNSSEC: Security and availability challenges. 365-366. 10.1109/CNS.2013.6682730. 2013 IEEE Conference on Communications and Network Security (CNS), October 2013, Copyright © 2013, IEEE.

[2] P. Mockapetris, “DOMAIN NAMES - CONCEPTS AND FACILITIES”, RFC1034, Nov 1987 (INTERNET STANDARD) [Online] [Available: http://www.ietf.org/rfc/rfc1035.txt] Accessed: July 2020

[3] P. Mockapetris, “Domain names - implementation and specification,” RFC 1035 (INTERNET STANDARD), Internet Engineering Task Force, Nov. 1987, updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2673, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604. [Online]. Available: http://www.ietf.org/rfc/rfc1035.txt

[4] R. Elz “Clarifications to the DNS Specification” RFC 2181, Internet Engineering Task Force, July 1997, [Online] [Available: http://www.ietf.org/rfc/rfc2181.txt] Accessed: July 2020

[5] Lindsay, David (2007). International Domain Name Law: ICANN and the UDRP. Bloomsbury Publishing. p. 8. ISBN 978-1-84113-584-7.

[6] P. Vixie “Extension Mechanisms for DNS (EDNS0)”, RFC 2671 , Internet Engineering Task Force, Aug 1999, [Online] [Available: http://www.ietf.org/rfc/rfc2761.txt] Accessed: July 2020

[7] J. Damas “Extension Mechanisms for DNS (EDNS(0))” Internet Engineering Task Force , April 2013, RFC 6891, [Online] [Available: http://www.ietf.org/rfc/rfc6891.txt] Accessed: July 2020

[8] A. Gulbrandsen “A DNS RR for specifying the location of services (DNS SRV)” RFC 2782, Internet Engineering Task Force, Feb 2000, [Online] [Available: http://www.ietf.org/rfc/rfc2782.txt] Accessed: July 2020

[9] R. Arends “DNS Security Introduction and Requirements” RFC 4033 Internet Engineering Task Force, Mar 2005, [Online] [Available: http://www.ietf.org/rfc/rfc4033.txt] Accessed: July 2020

[10] A. Herzberg and H. Shulman, “Towards Adoption of DNSSEC: Availability and Security Challenges,” Cryptology ePrint Archive, Report 2013/254, 2013, https://eprint.iacr.org/2013/254 . accessed in July 2020.

[11] Mark Southam, DNSSEC: What it is and why it matters, Network Security, Volume 2014, Issue 5, 2014, Pages 12-15, ISSN 1353 4858 ,https://doi.org/10.1016/S1353-4858(14)70050-9. (http://www.sciencedirect.com/science/article/pii/S1353485814700509)

[12] Z. Wang, "A Revisit of DNS Kaminsky Cache Poisoning Attacks," 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, 2015, pp. 1-6, doi: 10.1109/GLOCOM.2015.7417017.

[13] "DNS Cache Poisoning: A Review on its Technique and Countermeasures," 2018 National Information Technology Conference (NITC), Colombo, 2018, pp. 1-6, doi: 10.1109/NITC.2018.8550085.

[14] DNS Cache Poisoning Vulnerability Explanation and Remedies Viareggio, Italy October 2008 Kim Davies Internet Assigned Numbers Authority https://www.iana.org/about/presentations/davies-viareggio-entropyvuln-081002.pdf [accessed in July 2020]

[15] Trostle, Jonathan & Besien, Bill & Pujari, Ashish. (2010). Protecting against DNS cache poisoning attacks. 25 - 30. 10.1109/NPSEC.2010.5634454.

[16] R. Arends “Resource Records for the DNS Security Extensions”, RFC 4034, Internet Engineering Task Force, Mar 2005, [Online] [Available: http://www.ietf.org/rfc/rfc4034.txt] Accessed: July 2020

[17] D. Conrad “Indicating Resolver Support of DNSSEC” RFC3225, Internet Engineering Task Force, Dec 2001, [Online] [Available: http://www.ietf.org/rfc/rfc3225.txt] Accessed: July 2020

[18] O. Kolkman “Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag” RFC 3757, Internet Engineering Task Force, Mar 2005, [Online] [Available: http://www.ietf.org/rfc/rfc3757.txt] Accessed: July 2020

[19] Flexible and Robust Key Rollover in DNSSEC (open access) Y. Schaeffer, B.J. Overeinder and M. Mekking *In Proceedings of the Workshop on Securing and Trusting Internet Names (SATIN 2012)* March 2012 (https://www.nlnetlabs.nl/downloads/publications/satin2012-Schaeffer.pdf) [accessed in July 2020]

[20] "Protocol Modifications for the DNS Security Extensions," RFC 4035 (Proposed Standard), Internet Engineering Task Force, Mar. 2005, updated by RFCs 4470, 6014, 6840. [Online]. Available:http://www.ietf.org/rfc/rfc4035.txt

[21] B. Laurie, G. Sisson, R. Arends, and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence," RFC 5155 (Proposed Standard), Internet Engineering Task Force, Mar. 2008, updated by RFCs 6840, 6944. [Online]. Available: http://www.ietf.org/rfc/rfc5155.txt

[22] DNS-based Authentication of Named Entities (dane) https://datatracker.ietf.org/wg/dane/charter/ Accessed : July 2020

[23] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," RFC 2246 (Proposed Standard), Internet Engineering Task Force, Jan. 1999, obsoleted by RFC 4346, updated by RFCs 3546, 5746, 6176. [Online]. Available: http://www.ietf.org/rfc/rfc2246.txt

[24] P. Mockapetris"DOMAIN NAMES - CONCEPTS AND FACILITIES"RFC 1034 Internet Engineering Task Force, Nov 1987 [Online]. Available: http://www.ietf.org/rfc/rfc1034.txt

[25] DNSSEC Deployement Statistics https://stats.dnssec-tools.org/ Accessed July 2020

[26] A. Hubert and R. van Mook, "Measures for Making DNS More Resilient against Forged Answers," RFC 5452 (Proposed Standard), Internet Engineering Task Force, Jan. 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5452.txt

[27] P. Hoffman and W. Wijngaards, "Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC," RFC 6605 (Proposed Standard), Internet Engineering Task Force, Apr. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6605.txt

[28] ECDSA: The missing piece of DNSSEC (https://www.cloudflare.com/dns/dnssec/ecdsa-and-dnssec/) Accessed: 20th July 2020

[29] D. Atkins “Threat Analysis of the Domain Name System (DNS)” RFC 3833, Internet Engineering Task Force, Aug 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3833.txt

[30] O. Kolkman “DNSSEC Operational Practices”RFC 4641 Internet Engineering Task Force, Sept 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4641.txt

[31] D. J. Bernstein, “DNS Forgery,” Internet publication at http://cr.yp.to/djbdns/forgery.html, November 2002. Accessed July 2020

[32] O. Kolkman “DNSSEC Operational Practices, Version 2” RFC 6781 Internet Engineering Task Force, Dec 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6781.txt

[33] POWERDNS : What We Do https://www.powerdns.com/whatwedo.html, Accessed 20th July 2020

[34] VERISIGN LABS https://dnssec-debugger.verisignlabs.com/ Accessed 20th July 2020

[35] DNSViz https://dnsviz.net/ Accessed 20th July 2020

[36] Posteo.de , An email provider that uses DNNSEC. posteo.de Accessed 20th July 2020