DNS for Fun and Profit

Roy Arends
Telematica Instituut
Brouwerijstraat 1
7523 XC Enschede
roy@dnss.ec

Peter Koch [*]
DENIC e.G.
Wiesenhüttenplatz 26
60329 Frankfurt/Main
koch@denic.de

1 Background

The DNS is a well studied and well known application service protocol. Systems and appliances around the net have been using DNS for years and many security issues have been discussed. Recently, two things have again droven the attention to this *old horse*. First, after more than a decade of work, DNS Security extensions (DNSSEC) have finally reached a level of maturity that deployment is in the reach of a few months. Second, a number of additional, new approaches have chosen to use DNS as a base technology, stressing the infrastructure, posing new demands on stability and security. Among those are ENUM and MARID/SPF to name a few. We try to report on and review some new developments, following a *black hat – white hat* approach. The talk will consist of several episodes, some of which will demonstrate weird ideas, but also a lesson to be learned. So the word *profit* in the title aims at an increase in mental, not fiscal wealthiness.

The following paragraphs will sketch the different episodes, starting with the state of affairs in DNSSEC, DNS protocol conformance, DNS and IPv6 and potential security implications and two methods to use DNS as a universal communication protocol, at least questioning some protocol based filtering policies.

2 DNS finally secure

After many years of work and refinement, the DNSSEC specification has finally been approved by the IESG. For the first time after RFC 2065 and RFC 2535 a specification is awaiting publication that is well understood, tested and considered deployable [DNSSEC1] [DNSSEC2] [DNSSEC3]. Still, some issues need further work, including the *zone walking problem* and *key management* questions.

[*]Part of this work was done when the author was with Universität Bielefeld, Technische Fakultät

2.1 DNSSEC-bis

The successor of [RFC2535] will be a set of documents currently sitting in the RFC Editor's queue, which has been approved by the IESG on 2004-09-27. While there are many subtle changes to the protocol, some of them are visible to zone maintainers and thus worth noting[1]:

- The scope of the `KEY` record has been limited to DNSSEC keys to overcome the subtyping problem.
- The *Delegation Signer* (`DS`) resource record type has been newly defined to carry the necessary information for securing delegations.
- A complete type code rollover changed the numeric code and the mnemonics of `KEY`, `SIG` and `NXT` to `DNSKEY`, `RRSIG` and `NSEC`, respectively.
- Not only did the name of `NXT` change, its successor `NSEC` also employs a different data format.
- Support for EDNS0 is now mandatory for DNSSEC compliant servers and resolvers.

2.1.1 Limiting the Scope of `KEY` Records

The `KEY` record was designed with extensibility in mind and thus carries a one octet *protocol* field to identify its actual scope, be that DNSSEC, email or IPsec. This flexibility is paid for with some disadvantages.

The Subtyping Problem By specifying the *protocol* field the `KEY` RR was effectively subtyped, i.e. a class of different keys can be carried within the `KEY` RR where the application is identified by information inside the `KEY` RR. The advantage is that it needs only one RR type (and the *protocol* field IANA registry) to cover several similar applications. Since traditionally the deployment of RR types was difficult and slow, key distribution within the DNS could benefit from this approach. The disadvantage, however, is that an application cannot ask for a specific subtype. Instead, it has to ask for the `KEY` RRset at a specific domain name and examine the response to find those `KEY` RRs it is interested in (or none at all). Not only is this a waste of bandwidth, it also easily leads to very large response packets and thus increases the likelihood of TCP requeries.

Parents should not blindly sign ... As in real life, some parents are not likely to sign everything the child may present. Since `SIG`natures always have to cover an RRset, a parent zone would have to sign all `KEY` RRs present at the child's zone apex, including those unrelated to DNSSEC. Since the parent might not be able or willing to verify those non-DNSSEC keys, it would probably refuse to sign them not to expose itself to liability claims.

[1]For an overview of DNSSEC see [Koch01] and [SK2001]

To avoid both the subtyping and policy or liability problems, the KEY RR was restricted per [RFC3445] to only contain DNSSEC relevant key material. Subsequently new RR types have been proposed by their respective supporting communities, e.g. for distributing SSH and IPsec keys in the DNS.

2.1.2 The DS Record Type

With DNSSEC as of [RFC2535] the parent's SIGnature over the child's KEY RRset was stored within the child zone. Getting a new signature from the parent meant to communicate the KEYs, have the parent generate the SIG RR, to receive that and finally to incorporate it into the child zone. Frequent key rollover for the child was thus cumbersome. On the other hand, insecure zones (those not making use of DNSSEC) need to be securely announced as insecure. [RFC2535] required a signed NULL KEY in the parent zone for this, increasing zone size and being inconsistent with where the authoritative KEY data for the zone apex would reside. Also, a key rollover at the parent with the need to re-sign all KEY RRsets for secured delegations would have become an operational nightmare if not impossible since all maintainers of delegated (secured) zones would have to be contacted and instructed to replace the SIG records at their zones' apex with the newly generated ones. A situation like this would have suggested to store the child's SIGnatures at the parent, which in turn might have had negative impact on child side key rollover.

To escape this no-win situation, a level of indirection was introduced by the specification of the DS record [RFC3658]. With DS two roles are identified for keys: they can appear as *zone signing keys* (ZSK), where the private key is used to generate the signatures over authoritative zone data (as before) and as *key signing keys*, which only sign the ZSK and which are the only ones signed by the parent. It may happen that KSK and ZSK are identical (in which case there is no indirection), but by splitting the roles key rollover is simplified for both the child and the parent.

The basic idea is that the DS contains a SHA-1 hash of the child zone's KSK. The DS is signed by and authoritatively served from the parent zone, securing the child zone. A parent initiated key rollover can be performed without interaction with the child.

The child zone has its ZSK signed by its KSK. The public parts of both the KSK and the ZSK are served from the apex of the child zone. The KSK can be exposed less frequently than the ZSK and thus be stored securely. The ZSK is needed to sign the other records in the zone, is used more often and thus may be changed rather frequently, with a new ZSK needing a signature by the KSK. A ZSK roll over does not need an interaction with the parent zone.

In addition, the DS record can be used to implement trust anchors which bypass the delegation hierarchy (which is the basis for the trust chain) to stipulate DNSSEC deployment with islands of trust.

2.1.3 Type Code Rollover

Shortly after DS was specified and implemented in pilots, an incompatibility was found in how DS-aware servers hand out NXT records for unsecured delegations, which make up the vast majority to date. The server would add to a referral response (containing an NS RRset) a signed NXT record to prove that no DS exists and the delegation was indeed unsecured. However, resolver implementations existed at that time, which would probably interpret this incorrectly to mean that the delegation did not exist at all.

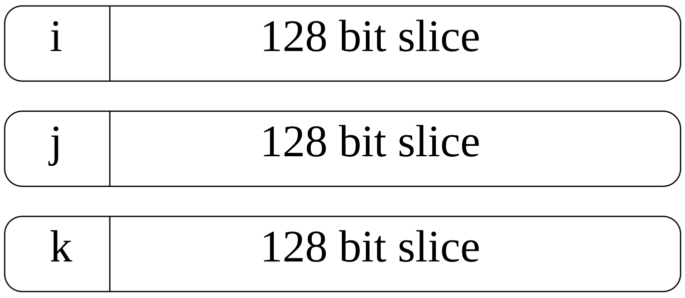
Since this bug (*Jakob's Bug*) was related to an underspecification in [RFC2535], a protocol change was needed, including changing the numeric code for NXT to avoid the aforementioned misinterpretation. For a variety of reasons it turned out that the other codes (for KEY and SIG) had to be changed as well. Finally, to avoid confusion at the zone file level, also the mnemonics were changed, so SIG became RRSIG, KEY became DNSKEY, which also reflected the now limited scope and NXT was named NSEC [RFC3755]. Apart from subsequent changes to NSEC (see below) the internal structure of the three RR types remained unaltered.

2.1.4 New NSEC data format

The NSEC RR type exists to support authenticated denial of existence of names and/or RRsets. Nonexistence of names is proven by signing the gap between adjacent names within a DNS zone, where this gap is expressed by an NSEC RR which points from one name to its immediate successor (in canonical ordering).

When all the DNSSEC record types were subject to a type code rollover, the format of the NXT record was discussed and found to be insufficient. At the time of its definition [RFC2535], only few RR types had been defined and the introduction of new types was not really encouraged. For that reason, the NXT RR was only able to cover the lowest 127 type codes. A way forward for extension of this space had been layed out but never been detailed afterwards. So, in the light of private types [RFC2929] and larger packet sizes (see below), the NSEC RDATA format was designed to cover all of the code space while at the same time being efficient for the current average case, where any owner holds RRsets of only few different RR types [RFC3845].

Essentially the bitmap of NXT has been replaced by a list of up to 256 bitmap slices of 32 octets (256 bits) each. A bitmap slice is present in the RDATA only if at least one type covered by that slice exists for the particular owner:



In this example i, j and k represent the one octet slice numbers covering types $256*n \ldots 256*n+255$ for $n = i, j, k$. With most of today's DNS owner names the NSEC RR will consist of only a single slice number 0 for types 0 through 255.

2.1.5 Dealing with Increased Packet Size

Not only the `NSEC` RR, which can grow up to more than 8K in size (see above), but even more `DNSKEY` and `RRSIG` records will increase the average DNS response packet size once DNSSEC is in broader use. The current limit is 512 octets per DNS message, which is hardly enough to carry multiple signatures. To avoid truncation, which would cause queries to be reissued over TCP, support and use of increased message sizes is now mandatory for DNSSEC aware servers and resolvers [RFC3226]. The underlying mechnism, EDNS0 [RFC2671], has been available for quite some time, but experience shows that some non-DNS devices with protocol knowledge (i.e. packet filters) sometimes get in the way of it and just drop packets they believe to be beyond protocol limits. Hopefully these effects will vanish over time, but the fragmentation and reassembly potentially caused by message sizes above 512 octets are subject to further study.

2.2 The Zone Walking Problem

As a side effect of spanning the gap between adjacent names, the chain of `NSEC` RRs allows a walk through the zone starting at the zone apex, touching all names in the zone, including delegation points, if any, and finally returning to the zone apex. While this is not a problem per se, it conflicts with many zone maintainers' policies. These zone maintainers currently do not allow bulk access to their zones by restricting `AXFR` and `IXFR` queries to trusted systems, hopefully including the necessary slave servers. These systems are usually identified by IP address using ACLs or by knowledge of a shared secret using TSIG [RFC2845].

Rather late in the *Last Call* process this problem received much attention, where it had been existing for years. While countered by the fact that confidentiality had not been a design criterion for DNSSEC, several TLD registries were severely concerned due to data protection and privacy issues while others feared the sheer query volume should domain name grabbers regularly walk a zone.

After some heated debate it was evident that although DNS data is considered public, there are different views what *public* means in this context and that there were real life issues that would turn out to become serious deployment obstacles. First, `NSEC` made the situation worse than it was. Second, while confidentiality was not a design goal, neither was bulk access to all names in a zone.

The IETF `dnsext` working group decided to accept the concerns and to work on an improved `NSEC` successor, which shall not leak information about other names. Nonetheless, DNSSEC-bis was approved with `NSEC` as it stands, with online signing as a short to medium term option. These issues are currently actively being worked on.

2.3 Key Management Issues

While the on the wire protocol is stable and believed to be mature, operational procedures are still under development. This includes guidelines for key generation, key length considerations and recommendations for key lifetimes and signature validity intervals.

Since the root domain key is the most crucial in DNSSEC, both for deployment and long term stability and security, careful design of generation and handling procedures is most important.

The IETF `dnsop` working group is currently focussing on these issues with an operational guidelines document being in its final stages before publication.

3 The Devil’s in the Dirt

Requirements for behaviour of DNS protocol implementations are thoroughly documented, i.e. for standard operations using request and response messages. However, the DNS protocol offers a multitude of message bits, response types, opcodes, classes, query types and label types in a fashion that makes some mutually exclusive while others are solely used for response messages.

The DNS protocol is more than 20 years old and since its inception, there have been over 50 independent DNS implementations, where some of them have over 20 versions. Not every implementation offers the full set of features the DNS protocol set (currently) has. Some implementations offer features outside the protocol set, and there are implementations that do not conform to standards. Meanwhile, new features are added to – and bugs removed from – implementations.

In order to measure interoperability between different implementations, and to check compliance of implementations with the various DNS specifications, a DNS compliance project was setup. Since the results were surprisingly unique for each implementation, an effort was made to identify different implementations by their behavior. This resulted in a fingerprint tool [FPDNS].

3.1 Fingerprinting DNS implementations

A DNS request message consists, among other data, of a DNS header, which includes a 16 bit space for opcodes (4 bits), status bits (8 bits) and response codes (4 bits). We send requests (Q) enumerating the 16 bit space to every well-known implementation (S) in a lab environment. The received responses (R) are then recorded. This results in a matrix of Q by S, where Q(x)S(y) presents R(x). We use this matrix to build a tree structure. The tree is then used as a base for further differentiation.

To build the tree, we look for x where the set of R(x) is mostly unique. The corresponding Q(x) will be the root of the tree. To build branches, we group identical R in this set of R(x) and find for every group the most unique R(x). Branching continues until the search is exhausted. The branches with either a leaf node, which signifies a set of implementations that behaves similar.

To differ within a set of similar behaving implementations we construct a set of specially crafted request messages. These borderline messages will mostly trigger a response that uniquely identifies the implementation. Input for the construction of messages is derived from the actual source code of an implementation, feature descriptions in documentation, vulnerability reports and simple bug fixes.

In a lab environment, we were able to identify about 15 different implementations with the described technique. After unleashing the technique on a bulk of DNS implementations on the Internet, we were able to identify many more.

In real world configurations, some obstacles that make the tool less effective are active load balancing (where queries are sent to different servers), specialised forwarders and smart firewalls which strictly analyse the composition of query packets.

4 Renaissance of Security by Obscurity?

DNS has been a medium for reconnaissance since the very beginning. For the same time it has been mostly consensus that blocking certain DNS query methods or not entering systems' names into the DNS does not increase their security. Even more, the use of DNS reverse mapping for security or validation purposes has been questioned and providing no reverse mapping, while current sad practice, is again no security improvement since IPv4 address space given to any entity is relatively small and enumerable (including probes and scans) in short time. IPv6 will change this precondition and an organisation's IPv6 address space will very likely be only sparsely populated. Finding potential victims will be harder and while some systems still will be exposed due to services they offer or because they appear in foreign logs due to users' activities, the IPv6 reverse mapping may leak information. It has been advocated that IPv6 reverse mapping poses a risk, decreasing the *obscurity* of systems *hidden* in unexpected portions of ones address space.

4.1 Prelude

One of the goals of IPv6 has been enlarging the address space available globally as well as per organisation. The current IPv6 addressing architecture [RFC3513] suggests that organisations receive a /48 IPv6 address range and build subnets if a fixed address width of /64. That leaves room for up to 65536 subnets with 2^{64} (roughly 10^{19}) addresses each. Even at a scanning rate of 100 addresses per second a full scan of a single subnet would take an Earth's lifetime. An adversary can, however, take advantage of the address configuration strategies [Chown], as documented in [RFC2462]. Sometimes human habits may make guessing even easier, when manual address assigment starts at low numbers. Still, scanning through large portions of the populated address space takes significant time.

4.2 A Highly Structured Namespace

The DNS is not only used to map domain names to IP addresses (both v4 and v6), but also to provide a mapping in the opposite direction, i.e. from an address to a name or set of names. For IPv4 this is known as *reverse mapping* in the `IN-ADDR.ARPA` domain. For IPv6 the domain is finally IP6.ARPA [RFC3596]. While the reverse mapping for IPv4 relies upon the octet boundary (with all disadvantages in a CIDR world), IPv6 uses labels of half octet width. An address of 2001:DB8::42 [RFC3849] would thus be represented by the domain 2.4.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA. The namespace is highly structured and 34 labels deep.

Now a brute force search in the DNS would be at least as infeasible as a full range port scan, but it can be significantly shortened by using the additional information given by the DNS. First, the DNS does not only know *leaf nodes*, but also inner nodes. For example, www.example.net usually comes as leaf node (i.e. with no subdomains), where example.net would be an inner node (with at least www.example.net as a subdomain). Leaf nodes must carry data (be owners of at least one RRset), whereas inner nodes may or may not have RRsets associated with them (example.net would usually own at least an SOA and an NS RRset).

The inner nodes without RRsets are also called *empty non-terminals*. They can be identified by querying with QTYPE ANY. If the response shows NOERROR and an empty answer section and is authoritative, then the name exists, carries no RRset but has at least one subdomain *with* data.

If 2001:DB8::42 were the only address in 2001:DB8::/64, a query for 0.0.0.0.0.0.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA would yield a NOERROR/NODATA response, while a query for 1.0.0.0.0.0.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA would lead to an NXDOMAIN. That would drastically reduce the range of names to be queried for, because the latter means there must not exist any node below 1.0.0.0.0.0.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA. The same reasoning is applicable to other structured name spaces, e.g. the e164.arpa domain used for ENUM.

4.3 Proof of Concept

A proof of concept implementation has been done in Perl using Net::DNS, neither optimized for speed nor designed to minimize tracable queries. With this program it was possible to enumerate a university’s /48 in a few minutes. Run times vary depending on how many subnets are available, how they aggregate and how addresses are assigned within a subnet. In any case, a tree walk can be done in reasonable time compared to a full scan.

However, there is a bug present in early versions of BIND 9, which incorrectly returns NXDOMAIN for empty non terminals. This bug has been fixed in BIND 9.3 and was not present in BIND 8. Since BIND 9 is the predominant implementation used to serve IP6.ARPA domains and many have not yet upgraded to the latest releases (9.3.x), the bug prevents full enumeration in many cases.

An enumeration would also be possible with an NSEC zone walk, of course.

4.4 Potential Defenses

Now that we found that even with IPv6 you cannot hide, one might ask how to reinstantiate that nice property. First of all, it is important to understand the difference between security and obscurity. A security policy must not rely upon the hiding of hosts, as long as they are accessible. There are plenty of opportunities where addresses can and will leak, especially when these hosts are endpoint for legitimate traffic, no matter where that traffic originates. There are also efforts underway to sniff DNS traffic on the wire to build huge DNS caches based on observed traffic, which would also leak information about existing IPv6 addresses.

The *ostrich approach* is thus not a security measure but it can only serve as an additional means to make the scanner's life harder, be that a skilled individual, a script kiddy or a virus looking for victims.

4.4.1 Avoid IPv6 Reverse Mapping

Not providing IPv6 reverse mapping at all is the most obvious solution to the enumeration problem, although it may throw the baby out with the bathwater.

4.4.2 Insert Dummy Addresses

One could add dummy names to the respective `IP6.ARPA` zones, but to really slow down or discourage the enumeration those phantoms would be needed in a volume which by itself would generate probably more operational complexity than one is willing to accept.

4.4.3 Camouflage by Wildcards

Wildcards are an evil toy anyway, so please do not take this as a recommendation. Nevertheless, they offer the opportunity to wipe out the difference between an empty non-terminal and a non-existing name. However, they might be detected and, even worse, lead to operational problems, when they are used in normal operations, e.g. when someone forgot to enter a new hosts address/name pair into the reverse mapping zone. Wildcard interaction with DNSSEC generates its own bag of problems.

4.4.4 Delegate and Refuse

Since in normal operations there is currently little to no need to query anything between a /64 and a /128, one might delegate the 16 domains immediately below the /64 boundary and then again below the /124. Ideally the lower end zones would be delegated to the same set of nameservers as the /64 domain. So we end up with 16 zones up in the tree and an unknown but potentially high number of zones just above the leaves. Now the nameservers could be instructed to refuse queries into the 16 zones while the would continue to answer queries ending up in the leaf zones, i.e. matching a full IPv6 address.

This approach only works if all nameservers consistently implement ACLs and only at a price of higher operational complexity. Also, any use of the DNS that needs to find the enclosing zone (e.g. dynamic updates) may get into trouble. So, also this approach is not generally recommended as it needs further investigation and experimentation.

Finally, maybe none of the defenses is worth the effort and one should accept that the idea of hiding in the width of IPv6 address space was nothing but an illusion.

5 About Open Relays and Open Resolvers

In the early days of the Internet it was common to offer services to everyone, either intentionally or as a side effect. The rise of spam taught us that this approach was not well suited for the real world, so over time most open relays were closed. Spam did not cease but just one abuse mechanism disappeared. In the DNS there is a service similar to open relays: resolvers offering recursive service to anyone. While large ISPs or organisations need one or more recursive name servers (resolvers) to serve their own clients, there is no need to serve the world. Even worse, due to the existence of cache poisoning, this kind of munificence may expose the system to certain attacks. While this has been known for quite some time, we will show how third parties can communicate through open resolvers, using nothing else but DNS protocol conformant packets.

Cache poisoning is done in several ways, each part of a different class of attacks. There is the intrusive attack, where a picaro can sit on the wire and alters host addresses in response packets, which will subsequently be cached for a certain time by a resolver. Another class is the trial and error attack, where a resolver is hosed with a bulk of identical responses, while the identifiers are different. To increase the success of such an attack, the expiry time of the current cached data for a chosen candidate can easily be measured. An open resolver will promptly respond to probes for the expiry time of the current cached data.

Another type of attack is a simple Denial of Service attack against a resolver. The BIND 8 nameserver does not implement a memory-usage maximum. It is trivial to burst a set of qualified queries to a resolver, where the responses will contain records with high TTL values, in order to increase the memory footprint of a cache. Once the size will near physical limitations of some form, the server will stall or worse, stop processing queries.

Using open resolvers as a DoS amplifier is another attack vector. It is trivial to send a single query to a resolver. It is trivial to send a bulk of identical queries to a bulk of open resolvers. Acquiring a set of IP addresses which host open resolvers is trivial as well. Since DNS is UDP based, a scan of a single /16 ('class B') network takes only minutes. Using a large enough bulk of resolvers to query root or TLD servers for a single name can cause the respective domain to experience a significant performance degradation.

A new set of abusiveness is currently being developed that involves using a bulk of open resolvers and their cache as a way to store and transfer data, disguised as DNS data. The data can take any form, from streaming media to bit-torrent seeds [Kami04].

Note that these open resolvers do not have to be recruited to be victimised. They are there for the taking, they simply need to be found. All this can be done within the boundaries of the DNS-protocol. The packets on the wire are solely DNS messages.

6 Summary

We have presented the current state of securing the DNS protocol against protocol inherent threats with DNSSEC. We have also shown that you can neither hide the software identity of your nameservers – a fingerprint can uniquely identify a huge number of implementations

– nor hide yourself in the IPv6 address range. Finally, the darker parts of the DNS universe were examined, where the DNS is used to either attack systems or to carry traffic and thus potentially can circumvent security policies.

All spots should demonstrate that despite its age the DNS is a vivid protocol still subject to research and further extensions and significantly relevant to security.

Bibliography

[BIND] ISC BIND, `http://www.isc.org/products/BIND/`

[Chown] T. Chown, *IPv6 Implications for TCP/UDP Port Scanning*, Internet-Draft (work in progress), July 2004

[DNSSEC1] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, *DNS Security Introduction and Requirements*, Internet-Draft (work in progress), October 2004.

[DNSSEC2] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, *Protocol Modifications for the DNS Security Extensions*, Internet-Draft (work in progress), October 2004.

[DNSSEC3] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, *Resource Records for DNS Security Extensions*, Internet-Draft (work in progress), October 2004.

[FPDNS] R. Arends, J. Schlyter, *fpdns – Fingerprinting DNS Servers*, `http://www.rfc.se/fpdns/`

[Kami04] D. Kaminsky *Black Ops of DNS Toorcon 2004*, `http://www.doxpara.com/dns_tc/Black_Ops_DNS_TC_files/v3_document.htm`, 2004

[Koch01] P. Koch *Secure DNS - Stand und Perspektiven*, DFN-Bericht 92, DFN-Verein, 2001

[RFC2462] S. Thomson, T. Narten, *IPv6 Stateless Address Autoconfiguration*, RFC 2462, December 1998

[RFC2535] D. Eastlake, *Domain Name System Security Extensions*, RFC 2535, March 1999

[RFC2671] P. Vixie, *Extension Mechanisms for DNS (EDNS0)*, RFC 2671, August 1999

[RFC2845] P. Vixie, O. Gudmundsson, D. Eastlake, B. Wellington, *Secret Key Transaction Authentication for DNS (TSIG)*, RFC 2845, May 2000

[RFC2929] D. Eastlake, E. Brunner-Williams, B. Manning, *Domain Name System (DNS) IANA Considerations*, RFC 2929, BCP 42, September 2000

[RFC3226] O. Gudmundsson, *DNSSEC and IPv6 A6 aware server/resolver message size requirements*, RFC 3226, December 2001

[RFC3445] D. Massey, S. Rose, *Limiting the Scope of the KEY Resource Record (RR)*, RFC 3445, December 2002

[RFC3513] R. Hinden, S. Deering, *Internet Protocol Version 6 (IPv6) Addressing Architecture*, RFC 3513, April 2003

[RFC3596] S. Thomson, C. Huitema, V. Ksinant, M. Souissi, *DNS Extensions to Support IP Version 6*, RFC 3596, October 2003

[RFC3655] B. Wellington, O. Gudmundsson, *Redefinition of DNS Authenticated Data (AD) bit*, RFC 3655, November 2003

[RFC3658] O. Gudmundsson, *Delegation Signer (DS) Resource Record (RR)*, RFC 3658, December 2003

[RFC3755] S. Weiler, *Legacy Resolver Compatibility for Delegation Signer (DS)*, RFC 3755, May 2004

[RFC3757] O. Kolkman, J. Schlyter, E. Lewis, *Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag*, RFC 3757, May 2004.

[RFC3845] J. Schlyter, *DNS Security (DNSSEC) NextSECure (NSEC) RDATA Format*, RFC 3845, August 2004

[RFC3849] G. Huston, A. Lord, P. Smith, *IPv6 Address Prefix Reserved for Documentation*, RFC 3849, July 2004

[SK2001] M. Sanz, S. Kelm, *Einsatz von DNSSEC in der Domain .de*, DFN-Bericht 92, DFN-Verein, 2001