Exploring the Overhead of DNSSEC

Bernhard Ager *Holger Dreger* *Anja Feldmann*
TU München *TU München* *TU München*

Abstract

Even though the key ideas behind DNSSEC have been introduced quite some time ago DNSSEC has not yet seen large scale deployment. This is in large part due to the anticipated overhead of DNSSEC. While the overheads have been reduced by the introduction of the delegation signer model [14], it is still not clear if they are bearable.

Therefore we in this paper examine the actual overheads of DNSSEC. We first examine how the packet sizes of an DNS trace increases if DNSSEC would be used. Then we explore the CPU and memory overheads imposed by DNSSEC by replaying a DNS client trace in a testbed initialized with roughly 100,000 zones.

1 Introduction

Since the early days of the Internet people rely on the Domain Name System (DNS) [1, 2]. DNS maps human memorable hostnames to machine usable addresses. Its redundant and distributed design has made DNS an indispensable, yet from the users perspective transparent, component of the Internet. On the other hand the conventional DNS does not provide any mechanism to assure the authenticity of the retrieved information to its users. In the 1990's several real world attacks (e. g., [12], [18]) highlighted how easy it is to misuse DNS for attacking unsuspecting users (and administrators). This led to several proposals for enhancing or replacing DNS, of which an cryptographically enhanced version called DNSSEC is the current front runner. During the last 10 years several variations of DNSSEC have been discussed (e. g., [3]) yet the final decision is still outstanding. This is partly due to the ambitious goals of DNSSEC: full backwards compatibility as well as data integrity and data authenticity. Among the concerns hindering deployment are that it is unclear how much overhead DNSSEC introduces on the name servers, after all a critical part of the Internet architecture.

In this paper we explore the costs imposed by DNSSEC, in terms of network bandwidth as well as resource consumption on name servers, using a trace based analysis and by replaying a DNS query trace in a testlab. We find that deploying DNSSEC can result in substantially larger packets, depending on the cryptographic cypher, and therefore increased bandwidth usage and packet fragmentation. Furthermore our experiments indicate that this may easily balloon memory usage on caching servers by a factor of four.

The remainder of the paper is organized as follows: in Section 2 we give an overview of DNSSEC highlighting possible overheads. Section 3 discusses the environment in which we collected the datasets for our study. Section 4 presents our initial results regarding the per packet overheads while Section 5 examines the overheads imposed on the name servers. Finally, in Section 6 we summarize our experience.

2 Overview of DNSSEC

In this section we give a short overview of the DNS design and briefly discuss how DNSSEC extends it in order to guarantee integrity and authenticity.

DNS: DNS relies on a distributed database with a hierarchical structure. The root of the DNS system is centrally administered and serves its *zone* information via a collection of *root servers*. The root servers delegate responsibility for specific parts (zones) of the hierarchy to other *name servers*, which may in turn delegate the responsibility to other name servers. In the end each site is responsible for its own *sub-domain* and maintains its own database containing its information and operates an authoritative name server. An alternative view of this name space is one of a tree with labels at the nodes separated by dots. Information associated with any particular name is composed of *resource records* (RRs) which contain the type of the resource and data describing it.

The whole database is usually queried by end hosts using a local name server. If this name server receives a query for information that it does not have, it must contact another name server. If the server does not know how to contact the authoritative one for the zone it will contact one of the root servers. Those will usually not have an answer to the query but will *refer* the client to the authoritative server of the subtree the client's queried name lies in. For example the root zone . delegates the *org.* zone to another name server, so the client gets a referral from the root server to one for *org.*. For efficiency reasons DNS relies heavily on caching. All information that a name server delivers to a client is cached on the client for a duration specified in the TTL field of the RR. Caching is usually not done on end hosts, so called stub resolvers, but rather on dedicated machines on the path, so called *caching resolvers* or *forwarders*.

DNSSEC Extensions: DNSSEC is an enhanced version of DNS which relies on public key cryptography. Surprisingly it turned out to be way harder than anticipated to integrate a "Public Key Infrastructure" into DNS which allows each client to verify the signatures. The first "established" model [4] used impractical key handling, caused significant administrative overhead, imposed unnecessary DNS traffic, and did not allow incremental deployment [16]. The Delegation Signer (DS) model [8] overcomes these limitations by, e. g., introducing islands of security. Accordingly, our analysis is based on this approach [10, 9].

To verify an answer secured by DNSSEC an authentication chain from the DNS answer to a trusted key has to be built. Hereby it is assumed that the resolver has a well known trusted key for the root zone. Each zone guarantees for the correctness of the answer records using a digital signature. The chain is built from the bottom to the top, according to the DNS name hierarchy. The correctness of the zone's key is guaranteed by a signature from the zone's parent and so on, until the root zone is reached. If all authentication steps are successful, the DNS answer must be correct (unless one of the keys has been compromised).

To enable authentication the necessary signatures are associated with each name and are accessible via four additional RR types: *RRSIG* RRs include digital signatures for "Resource Record sets" (RRset). (RRSIG adds 46 bytes plus key size plus zone name length overhead for RSA and 70 bytes plus zone name for ECC.) An RRset is a set of RRs with the same name, the same class, the same type, and also the same TTL. *DNSKEY* RRs contain the public keys for the zone (adding 18 byte plus key size overhead). Each signed RRset has to provide such a key in order to enable the resolver to verify the signatures. *DS* RRs are configured at delegation points in the parent zone, if and only if the child-zone is secure. In effect these indicate the status of the child-zone. The most important content of the DS RR is a cryptographic hash (digest) of the child-zone's DNSKEY, where the child's name and the DS RR's name are identical. (Adding 36 bytes overhead.) Together with the DS record's RRSIG the authenticity of the child zone's DNSKEY can be verified. *NSEC* RRs provide a mechanism for verifying the nonexistence of records. Without authenticated nonexistence, an attacker may choose to answer a query with a referral, but not include a DS record. The resolver could interpret this as "zone exists, but is unsecured". In this case delegation can easily be forged and the attacker did not even have to sign the zone data. (NSEC adds 23 bytes plus name length plus label length overhead.) To provide space for all this overhead, DNSSEC capable name servers and resolvers must support the EDNS0 extension [6]. This implies that every DNSSEC packet contains an extra OPT pseudo record.

Type	Overhead	Comment
DNSKEY	18 + key size	RSA or ECC
DS	36	SHA-1 digest
RRSIG	46 + key size + |zone|	RSA
	70 + |zone|	ECC
NSEC	23 + |name| + |label|	

Table 1: Size of DNSSEC RRs in octets

Problems with DNSSEC: Signing and verification are mathematically complex operations and consume processing power. The necessary DNSSEC RRs are relatively large. Therefore DNSSEC capable name servers have larger memory and network bandwidth requirements. To highlight this Table 1 summarizes some of the overheads imposed by DNSSEC for RSA as well as ECC [17]. This can result in significantly larger packets. While not per se a problem, larger packets might have to be fragmented or even truncated. In the worst case this could lead to a fallback from UDP to TCP during the query process. To avoid some of these problems, DNSSEC [10] requires that the minimum accepted answer size of every DNSSEC aware resolver is 1220 Bytes (instead of 512). It recommends a minimum accepted answer size of 4000 Bytes.

3 Datasets

Our analysis utilizes two different kinds of datasets collected from the "Münchener Hochschulnetz" (MHN), a network that provides Internet access to two major universities, several research institutes and quite a few dormitories in Munich. Using a monitor port at the single upstream link between the MHN and the Internet we collected packet level traces of all DNS queries and answers. Second, we gained access to query logs for the main DNS forwarders inside the MHN. Hereby, it is important to note, that the MHN requires all resolvers inside the MHN to use one of three forwarders. This policy is enforced by blocking almost all outgoing traffic to port 53. Since the focus of this paper is on the overhead imposed by a representative sample population on the DNS infrastructure, all our analyses are based on client traces. We excluded DNS Transactions that originate outside of our local network (that is queries for the zones served by the local nameservers).

Data	Dir.	Size	Queries	Answ.	Start	Dur.
TR02	out	5.4G	32.8M	23.0M	11/19/02	5.8d
TR04	out	6.0G	36.6M	31.0M	10/06/04	2.0d
QL05	all	39.8M	1.01M	-	01/21/04	44m

Table 2: Summary of the datasets.

An overview of the data used in this paper is given in Table 2. TR02 and TR04 are packet level traces covering only UDP port 53 while QL05 is a query log from the main DNS forwarder. The trace TR02 contains almost six days of traffic starting on 11/19/2002 at 22:28 GMT and ending on 11/25/2002 at 16:34 GMT. While recording the trace, 335 packets were dropped. A total 0.156 % of all UDP packets were ignored. 89.5K queries as well as 56.5K answers were eliminated, since they are either irrelevant for Delegation Signer DNSSEC or due to decoding problems. Irrelevant queries include, e. g., 77.5K update queries, 7K notifications and 5K other queries. Of the 38K answers with decoding problems 19K are from a single source. A second trace TR04, captured under similar conditions, covers two weekdays, starting on 10/6/2004 at 10:06 GMT and ending on 10/08/2004 at the same time. But with 107,899,229 packets and no dropped packets it is a bit larger than TR02. The 107.1M UDP packets split into 58.2M DNS queries and 48.9M replies. In this trace a total of 0.09 % packets were excluded from further analysis.

The query log was captured directly from the main DNS forwarder, *dns1.lrz-muenchen.de*. Since the diversity of requests grows with the length of a trace and therewith the resource requirements of our experiments we limited the analysis to a roughly 50 minute portion of the overall log from 01/21/2005 10:56 to 11:40 GMT containing 1,008,384 queries.

4 DNSSEC: per packet overhead

In this section we examine, based on a packet trace of DNS requests, how much the per query and response overhead is as well as how much the overall increase in bandwidth is for DNSSEC. The main idea is to transform every DNS packet from TR02 and TR04 into a DNSSEC packet with the same content. This transformation is based on the DS-DNSSEC RFCs [8, 10, 9, 5] and Internet-Draft [15] as well as experiments with BIND [13] (version 9.3.0).

Transformation of DNSSEC queries: If the DNS query already contains an OPT pseudo record the DO bit has to be set and if necessary the acceptable answer size has to be adjusted to a minimum of 1220 bytes. Note, that this does not change the payload size. Otherwise an appropriate OPT pseudo record has to be added which increases the payload by 11 octets.

Transformation of DNSSEC answers: To derive DNSSEC answers from DNS ones several cases have to be considered: If the answer status is **Refused**, **FormErr**, or **ServFail** the answer contains no RRs. Accordingly no modifications are necessary. If the answer is **NotImpl** and the RRs are from the name server zone then the RRs have to be signed. Otherwise there is no change. In case of a **NXDomain** answer, two NSEC RRs have to be constructed and signed: one for denying the existence of the exact name, one for denying the existence of a suitable wildcard (see Section 2). For the **NoError** answers the kind of answer contained in the packet determines the necessary adjustments. If the packet contains a final answer, the answer, authority and additional sections of the DNS answer are signed. Otherwise if the answer section of the NoError answer is empty, but the authority section contains NS RRs, the packet contains a referral. In this case a DS RR is added to the authority section. Note that only the DS RR has to be signed as the NS RRs belong to the child zone. Otherwise the packet contains a negative answer. In this case, a NSEC RR is added and the packet is signed. In a final step an OPT RR is added unless the answer already contains one.

Signing a packet always entails adding RRSIGs for every RRset for the purpose of authentication. For all packets except referrals DNSKEY RRs for the Key and Zone Signing Keys (KSK/ZSK) are added. The reasoning behind this optimization is that the key information is already cached especially for the top level domains [2, 10]. For the RSA experiments we choose a KSK/ZSK key length of 1200/1024 bits [15], for ECC a key length of 144/136 bits [17]. For more details see [10].

The final payload increase results from applying the necessary transformations to a packet, summing up the single overheads shown in Table 1.

Results: Table 3 summarizes the increase in UDP payload for TR02. We consider queries and answers separately and sub-classify answers into “noErr” (successful) and “NXDomain”. Furthermore the successful answers are separated into “final answers”, “referrals” and “empty answers”. In addition we had 1.6M packets with error-code “FormErr”, 1.7M “ServFail”, 5K “NotImp”, and 0.3M “Refused”. The DNS size is computed by summing the UDP payloads of the corresponding packets. The DNSSEC size (not shown) is computed by transforming the DNS packets. Instead the last column “Factor for DNSSEC” captures the increase in the payloads relative to the DNS size. While the “raw” trace captures the influence of the popularity of each name we also considered a “normalized” version which eliminates duplicate queries. We consider a query as a duplicate if its key, consisting of the IP address of the servers, the queried name, its type, and the query/answer flag, is a duplicate. The assumption underlying this approach is that all answers for a queried name from one host are the

Type	Count		DNS Size		DNSSEC factor		
					RSA		ECC
	all	norm	all	norm	all	norm	all
Query	32.8M	5.1M	1.5G	0.3G	1.1	1.1	1.1
noErr	20.0M	4.2M	3.7G	0.7G	4.1	5.3	2.3
Final	6.8M	2.5M	1.2G	0.5G	6.2	5.7	3.0
Ref.	10.9M	1.3M	2.3G	0.2G	2.0	2.6	1.6
Empty	2.2M	390K	.2G	44M	11.7	10.6	5.2
NXD.	1.4M	500K	.2G	57M	12.7	12.9	6.2

Table 3: TR02: DNS size vs. DNSSEC size.

same except for ordering.

Table 3 highlights that the overhead is highly dependent on the packet type. Generally the size increase is much more moderate when using the less-well established ECC signatures [17]. As neither query packets nor answer packets of type Refused and FormErr are signed the overhead consists of at most an additional OPT section. This results in an average overhead factor between 1.1 and 1.2. Empty answers and NXDomain packets are on the other end of the spectrum. Initially they did not contain any RR. But when signed one, respectively two NSEC, two DNSKEY RRs, and the RRSIG RRs have to be added. This bloats the DNS packets by a factor up to twelve. Final answers are also subject to major increases as their RRs have to be signed. On the other hand the increase for referrals is moderate, since the DS model does not include the key of the parent zone. This reduces the overhead significantly (e.g., by 684 bytes for 1200/1024 bit RSA keys). On the other hand this can lead to additional queries to retrieve the DNSKEYs resulting in an delay of roughly one additional round-trip.

That the results for the "raw" trace are similar to the ones for the "normalized" trace highlights that the overhead of DNSSEC is intrinsic. However there is a major difference in the overall factors: 3.4 for "raw" vs. 4.6 for "normalized". This is due to a shift in the composition of the traces. Referrals contribute 42.1 % of the volume for "normalized", but only 18.0 % for "raw".

Although the distribution among the packet types is significantly different in TR04 the DNSSEC factors are similar. TR04 contains more erroneous answer packets (error-code NXDomain and ServFail) whereas the especially expensive NXDomain packets (DNSSEC factor of 14.0 in TR04) results in an overall size increase factor of 4.2 for the raw and 5.8 for the normalized trace.

While the overall increase is of interest to determine bandwidth demands and size the memory requirements of servers it is even more interesting to inspect the distributions. Figure 4, left shows a comparison of DNS vs. DNSSEC sizes for referrals from TR02. The shapes of the probability density functions over the packet sizes are very similar just shifted to the right for DNSSEC. After all DNSSEC always adds an additional DS RR to these packets. For final answers the size increase depends on the number of RRsets for which the server is authoritative as well as the length of the zone name. Furthermore RRsets differ in size. Figure 4, right shows a scatter plot of DNS packet size vs. DNSSEC packet size for all final answer packets in TR02. The identifiable lines have a slope of 1 and are due to the content of the original DNS packets: The parallel lines have a vertical distance of 174 bytes; the size of a RRSIG RR. Accordingly all packets within a line have the same number of authoritative RRsets.

	Size ≤	NXD	noErr	Final	Ref.	Empty
RSA	1,228	.005	.790	.701	1	.633
RSA	1,480	.231	.951	.921	1	.996
RSA	2,056	.999	.997	.991	1	.999
RSA	4,008	1.000	.999	.999	1	1.000
ECC	1,228	.998	.999	.998	1	.999
ECC	1,480	.999	.999	.999	1	.999

Table 4: Size thresholds: fraction of smaller answer pkts.

The last plot already shows that some DNSSEC packets may become subject to fragmentation and/or truncation even though the minimum DNS message size has been increased from 512 bytes to 1220 bytes. Further inspection shows that Query, FormErr, ServFail, and Refused packets are not limited by the maximum packet sizes at all. After all they do not contain any answers. Such packets account for more than 60 % of all packets, but less than one third of the DNS size and one tenth of the DNSSEC size. Still the other packets – especially answers with noError or NXDomain error code – can become quite big. Table 4 examines what percentages fulfill certain important thresholds: 1220 octets (1228 UDP size) is the minimum message size supported by every DNSSEC aware resolver. IP networks often use a MTU of 1500 bytes. This implies that only UDP packets of size ≤ 1480 bytes can be sent without fragmentation. The tool `dig` uses a default maximum DNSSEC answer size of 2048 bytes. Finally 4000 octets is recommended as a lower bound for the maximum answer size [7]. (BIND's default is 4096 octets.)

Our evaluation shows for RSA signatures, that virtually all DNSSEC packets, derived from both of our traces, are smaller than 4008 bytes. Moreover most (99.8 %) are smaller than 2056 bytes. Yet about 77 % (TR02) and 72 % (TR04) of the noErr and NXDomain packets will require fragmentation. One might assume that empty and NXDomain answers have a real problem if the answer size were to be limited to 1220 bytes. However the name server can drop one or both DNSKEY RRs as is done for referrals. This again reduces the size by 684 octets for RSA, ensuring that the name servers are *not required* to set the truncation bit (TC) [10]. When using ECC signatures instead of RSA ones almost all pack-

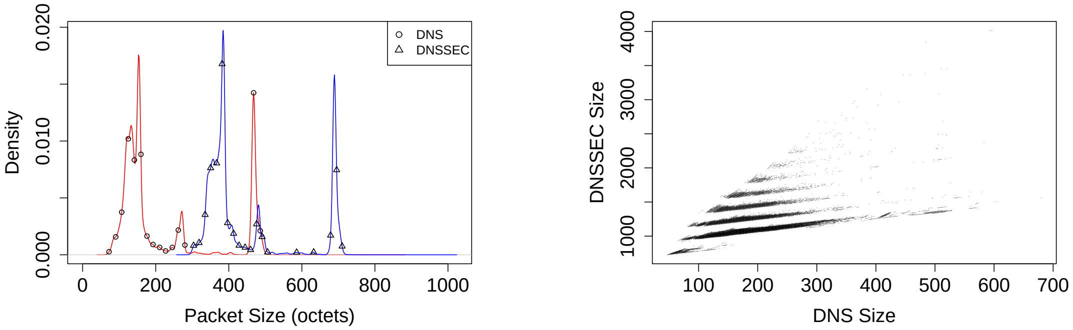

Figure 1: DNSSEC name resolution (TR02): including the relevant RRs for RSA signatures

ets are smaller than 1,228 bytes, regardless of their type, see Table 4.

5 DNSSEC: memory and CPU overhead

In order to evaluate how much CPU as well as memory overhead DNSSEC imposes on name servers and caches we replayed a DNS query trace, QL05 , against DNS as well as DNSSEC name servers in a testlab. The DNSSEC experiments rely on RSA signatures only, since ECC is not yet implemented in BIND.

Methodology: Our testbed setup, see Figure 2, includes three name servers, (*ns1*, *ns2*, *ns3*, Athlons XP1800+, Debian). Each of these is authoritative for some subset of the zones, using BIND [13] (version 9.3.0). The basic zone data for the experiment is derived from the query log QL05 , by iteratively resolving the contained queries. The RRs from the answer and the authority sections are used to initialize the zone database. Flawed RRs, e. g., NS RRs with IP addresses in the RDATA part are dropped. (The additional section also turned out to be problematic [11].) In a next step the zones are distributed across the set of available name servers in our testbed. We decided to assign each level of the DNS hierarchy to a different name server process, listening to an IP address of its own, to ensure that all iterative queries have to traverse the full hierarchy. Since zones on the same level are more or less independent they can be placed on the same name server. Each zone is hosted only on one ("primary") name server process, there are no "secondaries". This implies that the records had to be adjusted appropriately. The time to live fields are set according to the needs of the individual experiments. We distributed the more than thirty name server processes onto our three machines in a way that all server instances were enabled to keep their zone data in main memory: We choose to put the largest level, level two, on a machine of its own, *ns2*. The next largest zone, level five, together with the root zone and the top-level zones are put on another machine, *ns1*. The other levels, 3, 4, 6 to 33, ran on *ns3*.

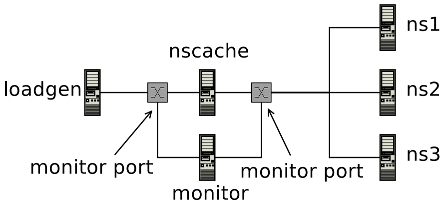

Figure 2: Testbed setup

We generated the queries on *loadgen* using the tool `dig` in batch mode. Memory and CPU usage for the authoritative name servers as well as for the caching name server *nscache* are determined leveraging the Linux per process accounting information from the `proc` filesystem at periodic intervals, i.e., after every 1000 queries.

Memory usage increase: Part of sizing a DNSSEC server respectively a caching name server is to determine the memory footprint of the application. We find that the penalty for authoritative servers is only a factor of two; way smaller than what one may expect given the results of Section 4. For example, the level two server for 55K zones occupied 290 MB for DNSSEC as compared to 156 MB for DNS. To determine the overhead for a caching name server we examine the memory usage of the BIND process on *nscache*. Its increase is rather dramatic, e.g., from 93 MB for DNS to 432 MB for DNSSEC for a workload that consisted of 218K unique queries spread over the complete zone-information using TTL values of 24 hours. This choice of TTL values effectively disables cache expiration. A necessity, as the current version of BIND seems to have a memory allocation bug in DNSSEC mode. If one replays the same queries after all RRs have expired (after 25 hours), this bug causes the memory usage to almost double. It seems that BIND allocates new memory to store the same information again without freeing the older version. As some RRs, especially the negative answer records and their signatures typically have short TTLs it is not surprising that replaying the original query log QL05 with actual TTLs increases the memory usage even more from 87 MB (DNS) to 513 MB (DNSSEC).

CPU usage increase: In our experiments we identify

Level	#DNS	CPU-time DNS	#DNSSEC	CPU-time DNSSEC
0	5.0K	1.0 s	6.4K	2.03 s
1	61.8K	11.56 s	65.4K	13.85 s
2	176.1K	23.80 s	249.1K	39.71 s

Table 5: CPU-time on auth. servers for 1M queries

cached	CPU-time DNS	CPU-time DNSSEC	delay DNS	delay DNSSEC
no	292.8 s	665.4 s	1.8 ms	3.9 ms
yes	37.1 s	45.9 s	0.2 ms	0.2 ms

Table 6: CPU-time on *nscache* for 218K queries

two sources of increased CPU usage due to DNSSEC. For authoritative name servers the overhead can only stem from the larger data volume that has to be sent, as they do not have to perform any cryptographic operations. Table 5 shows the CPU usage (averaged over five experiments) for those queries that the servers for levels 0, 1, and 2 had to answer when sending one million requests to *nscache*. Comparing the CPU times for DNS to those of DNSSEC we observe that DNSSEC introduces an overhead of a factor of 1.1 to 2. However, if we consider the average CPU usage per query the factor is roughly 1.3 to 1.6 for all level 0-2 authoritative servers.

Since our client issues all queries to the caching name server, *nscache*, with the DO bit unset, *nscache* is responsible for verifying any information before sending it to the client. That implies that the BIND process has to verify all signatures for all answers that it receives from the authoritative name servers unless it has the information already cached. In a first experiment we confront *nscache* with 218K unique queries to ensure that it for every query (at a minimum) has to verify the signature of the queried name itself. We used TTLs of 24 hours in order to enable BIND to cache referrals for the duration of the experiment. The goal of a second experiment is to enable *nscache* to serve all information from its cache. Recall that if the name server takes the information from its cache it does not have to perform any cryptographic verification. Therefore we repeated the same set of queries before the TTL expired. Table 5 shows the resulting CPU usages (averaged over five experiments). The CPU time for DNSSEC on *nscache* for the first run increases by a factor of 2.3 if compared to DNS. A fairly small factor given the task at hand. Roughly the same factor applies to the mean delay between queries and answers as experienced by the stub resolvers. On the other hand the numbers indicate that the absolute delay introduced by the additional verifications is hardly perceivable by the end user as long as the name server does not operate at its limit. (The delays are computed from packet level traces collected by the *monitor*, see Figure 2.)

6 Summary

In this paper we evaluate the costs a wide spread DNSSEC deployment would impose in terms of network bandwidth as well as resource consumption on name servers. We base our analysis on real world data from a large client population. For RSA signatures we find that DNS packets grow on average by a factor of 3.4 and 12.7 in the worst case. The less well-established ECC signatures outperform RSA: ECC signatures only impose an average overhead factor of 2.0 and 6.2 in the worst case.

In our testlab we found that DNSSEC operating with RSA signatures leads to significantly higher memory requirements: more so for caching name servers but also for authoritative name servers. CPU usage does not seem to be a show stopper for DNSSEC: although the CPU usage increases, modern hardware should be able to handle the extra processing without impacting the end user performance significantly. Overall we are not able to identify a principle performance hurdle for the deployment of DNSSEC, except that the necessary software changes for DNSSEC, e.g., BIND, are not yet fully matured as highlighted by a memory problem.

References

[1] RFC 1034: Domain names - concepts and facilities, 1987.

[2] RFC 1035: Domain names - implementation and specification, 1987.

[3] RFC 2065: Domain Name System Security Extensions, 1997.

[4] RFC 2535: Domain Name System Security Extensions, 1999.

[5] RFC 2537: RSA/MD5 KEYs and SIGs in the Domain Name System (DNS), 1999.

[6] RFC 2671: Extension Mechanisms for DNS (EDNS0), 1999.

[7] RFC 3226: DNSSEC and IPv6 A6 aware server/resolver message size requirements, 2001.

[8] RFC 3658: Delegation Signer (DS) Resource Record (RR), 2003.

[9] RFC 4034: Resource Records for the DNS Security Extensions, 2005.

[10] RFC 4035: Protocol Modifications for the DNS Security Extensions, 2005.

[11] Ager, B. Performance Evaluation of DNSSEC. Diplomarbeit, Technische Universität München, 2005.

[12] Bellovin, S. M. Using the Domain Name System for System Break-Ins. In *Fifth Usenix UNIX Security Symposium* (1995).

[13] ISC BIND. `http://www.isc.org/sw/bind/`.

[14] Gieben, R. Chain of Trust - The parent-child and keyholder-keysigner relations and their communication in DNSSEC, 2001.

[15] Kolkman, O., and Gieben, R. Internet-Draft: DNSSEC Operational Practices, 2004.

[16] Massey, D., Lewis, E., Gudmundsson, O., Mundy, R., and Mankin, A. Public Key Validation for the DNS Security Extensions. In *DARPA Information Survivability Conference and Exposition (DISCEX II)* (2001).

[17] Schroeppel, R. C., and Eastlake 3rd, D. Internet-Draft: Elliptic Curve KEYs in the DNS, 2004.

[18] Vixie, P. DNS and BIND Security Issues. In *Fifth Usenix UNIX Security Symposium* (1995).