The Reality of Algorithm Agility: Studying the DNSSEC Algorithm Life-Cycle

Moritz Müller
SIDN Labs and University of Twente

Willem Toorop
NLnet Labs

Taejoong Chung
Virginia Tech

Jelte Jansen
SIDN Labs

Roland van Rijswijk-Deij
University of Twente and NLnet Labs

ABSTRACT

The DNS Security Extensions (DNSSEC) add data origin authentication and data integrity to the Domain Name System (DNS), the naming system of the Internet. With DNSSEC, signatures are added to the information provided in the DNS using public key cryptography. Advances in both cryptography and cryptanalysis make it necessary to deploy new algorithms in DNSSEC, as well as deprecate those with weakened security. If this process is *easy*, then the protocol has achieved what the IETF terms *"algorithm agility"*.

In this paper, we study the lifetime of algorithms for DNSSEC. This includes: (i) standardizing the algorithm, (ii) implementing support in DNS software, (iii) deploying new algorithms at domains and recursive resolvers, and (iv) replacing deprecated algorithms. Using data from more than 6.7 million signed domains and over 10,000 vantage points in the DNS, combined with qualitative studies, we show that DNSSEC has only partially achieved algorithm agility. Standardizing new algorithms and deprecating insecure ones can take years. We highlight the main barriers for getting new algorithms deployed, but also discuss success factors. This study provides key insights to take into account when new algorithms are introduced, for example when the Internet must transition to quantum-safe public key cryptography.

ACM Reference Format:
Moritz Müller, Willem Toorop, Taejoong Chung, Jelte Jansen, and Roland van Rijswijk-Deij. 2020. The Reality of Algorithm Agility: Studying the DNSSEC Algorithm Life-Cycle. In *Internet Measurement Conference (IMC '20), October 27–29, 2020, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/TBA

1 INTRODUCTION

The DNS Security Extensions (DNSSEC) add integrity and authenticity to the DNS [2]. Operators of domain names like `example.com` can attest with DNSSEC that information, such as an IP address, is actually associated with their domain. Clients that validate this information can detect whether data has been tampered with. By May 2020, over 7.4M domain names have deployed DNSSEC [61].

IMC '20, October 27–29, 2020, Pittsburgh, PA, USA

ACM ISBN TBA…$TBA
https://doi.org/TBA

When DNSSEC was standardized, operators had the choice of just three algorithms to sign their domains with. Over the past 15 years, 9 new algorithms were added and 5 were deprecated [67]. New algorithms can replace insecure algorithms, or have more attractive attributes, like smaller keys and signatures. *Algorithm agility* has been achieved, if this replacement can be carried out *easily*, according to RFC 7696 [27].

As in other Internet protocols, algorithm agility in DNSSEC is crucial, because we do not know how fast attacks on cryptographic algorithms evolve, only that, at some point, algorithms will be broken [27]. This becomes even more urgent with the rise of quantum computers [9]. Even though it is still not clear, *when* quantum computers will become generally available, they do have the potential to break *all* current algorithms used in DNSSEC. Then, it becomes crucial to replace vulnerable algorithms by *quantum-safe* algorithms easily and fast.

In DNSSEC, introducing new algorithms and replacing existing ones is a four-stage process and we explain it in more detail in Section 2.2: (1) standardization at the Internet Engineering Task Force (IETF), (2) implementation in software and at entities responsible for registering and publishing domain names, (3) deploying algorithms at domain names and rolling out validating resolvers, and (4) deprecating insecure algorithms.

In this work, we analyze the full algorithm life cycle to answer the question: *Has DNSSEC achieved algorithm agility?* We find both *barriers*, which make algorithm adoption *harder* and *drivers*, that make it *easier* to adopt new algorithms. Using a mix of passive and active measurements, and anecdotal evidence we show that:

(i) *Standardizing* new algorithms is typically takes several years.
(ii) *Support* (1) in software of new algorithms is often held back by the lack of support in cryptographic libraries or their distribution in operating systems; and (2) registries and registrars can have a positive impact on algorithm deployment.
(iii) *Deployment* of new algorithms (1) on authoritative servers (*i.e.,* signing DNS records) is mainly driven by domain names that have not deployed DNSSEC before, rolling from a deprecated to a new algorithm happens rarely; and (2) on resolvers (*i.e.,* validating DNS records) is mainly driven by large providers.
(iv) *Deprecation and Replacement* of insecure algorithms at domains and resolvers is a multi-year effort.

In the remainder of this paper we first describe the background of DNSSEC and the algorithm life cycle in more detail (Section 2). Section 3 discusses related work. We describe our data sets in Section 4. Then, our analysis is split into four parts. First, we analyze the process for standardizing new algorithms in Section 5. Next,

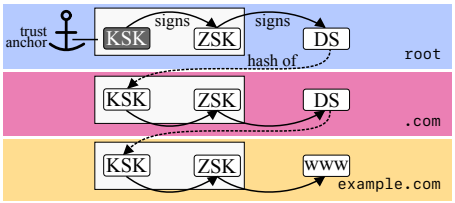


Figure 1: DNSSEC Trust Chain

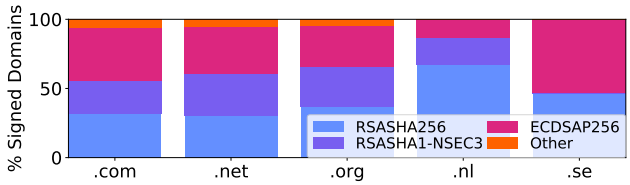


Figure 2: Signed domains per algorithm and TLD

we describe, how well software, registries and registrars support algorithms (Section 6). Then, we analyze their deployment at domain names (Section 7.1), and at resolvers (Section 7.2). Finally, we analyze the deprecation and replacement of algorithms in Section 8. To conclude, we discuss to what extent DNSSEC has achieved algorithm agility and where there is room for improvement (Section 9).

2 BACKGROUND

We first explain the necessary components of the DNS and DNSSEC involved when introducing or revoking an algorithm. Then, we describe the process itself.

2.1 The DNS Components

2.1.1 DNS and DNSSEC. Before users can visit the website of `www.example.com`. their computers must first *resolve* the corresponding IP address. To retrieve the IP, users rely on *recursive resolvers*, often located at their Internet service provider or at cloud providers. The resolvers query the *authoritative name servers* of `example.com`, where the IP is published in a *resource record* (RR). The DNS is a hierarchical naming system, where different *name servers* are `authoritative` for every part of the domain name (`example`, `com`, and the root '`.`').

Without DNSSEC, resolvers cannot verify whether the received information is correct. With DNSSEC, domain name operators can *sign* the RRs of their domain name using public key cryptography. To do so, they have different cryptographic algorithms at their disposal. The signatures and the public key are then published together with the actual information in additional RRSIG (RRset Signature) and DNSKEY RRs.

To avoid resolvers from having to trust every single public key, the DNS root and the top level domains (TLD) like `.com` are also signed. The root zone signs a hash of the public key of `.com` and publishes it together with the signature. The authoritative name servers of `.com` in turn sign the public key of `example.com` and publish the corresponding hash in a delegation-signer record (DS). This creates a *chain of trust* between the public key of the root and the key of `example.com` (Figure 1). For administrative reasons, operators can create a Key Signing Key (KSK) which signs a Zone Signing Key (ZSK). Once a resolver has fetched all necessary records it can validate the signatures. Incorrect signatures are considered *bogus* and unsigned records *insecure*.

2.1.2 Domain Administration. Domain name owners (*registrants*) buy their domain names at *registrars* which in turn are connected to *registries*, which administer TLDs. If registrants run their own name servers they can sign their records themselves. Otherwise, they need to rely on their registrar. In any case, the registrar relays their public key to the registry.

2.2 DNSSEC Algorithms

DNS operators can currently choose between 12 different algorithms to sign their domain names (top of Table 1), and 4 hashing algorithms are standardized to calculate the hash of the key in the DS record (bottom). Algorithms are identified (left column) by a number, which is used by DNS software in DNSSEC-specific resource records. Throughout this paper, we use names to refer to an algorithm. To ensure interoperability, RFC 8624 [67] specifies which algorithms software needs to support. The document differentiates between signing and validation and currently gives guidelines ranging from `MUST NOT`, `NOT RECOMMENDED`, `MAY`, `RECOMMENDED` to `MUST`. Table 1 lists these as well. In case a resolver does not support an algorithm, it should treat the RR as *insecure* [3]. Figure 2 shows a breakdown of the signing algorithms used in the five TLDs we analyse in this paper for May 2020. This reveals that some algorithms are preferentially used and the usages also differ across TLDs. We will dig deeper and attempt to explain these phenomena in the remainder of this paper.

The *life cycle* of DNSSEC algorithms consists of:

(i) *Standardization:* An Internet Draft needs to be standardized in the IETF. It specifies parameters and RR format to be used as well as the algorithm identifier.
(ii) *Support:* Software, responsible for signing RRs must support the new algorithm and resolvers need to be able to validate the signatures. Also, the new signatures and keys need to be *published* at the name servers of the domain itself and a DS record needs to be published at the parent domain (`.com` in case of `example.com`).
(iii) *Deployment:* When the software and the registration channels support the new algorithms, domain names can be signed. Resolvers that have been updated will then also treat the new signatures as *secure*.
(iv) *Deprecation and Replacement:* Algorithms are deprecated because they are not considered secure enough. RFC 8624 [67] defines which algorithms should not be used anymore. DNS operators that still rely on deprecated algorithms for signing should replace these. This *algorithm rollover* needs to be carried out carefully. If not, resolvers could fail to validate the signatures and could consider the RR *bogus* [40, 49].

	ID	Algorithm	First Draft	Standardized	Days	DNSSEC Signing	DNSSEC Validation
DNSKEY algorithms	1	RSAMD5 [16]	Aug. 2000	May 2001	273	MUST NOT	MUST NOT
	3	DSA [63]†	Sep. 1997	May 2004	546	MUST NOT	MUST NOT
	5	RSASHA1 [16]†	Aug. 2000	May 2001	273	NOT RECOMMENDED	MUST
	6	DSA-NSEC3-SHA1 [43]†	Jan. 2005	Mar. 2008	1520	MUST NOT	MUST NOT
	7	RSASHA1-NSEC3-SHA1 [43]†	Jan. 2005	Mar. 2008	1520	NOT RECOMMENDED	MUST
	8	RSASHA256 [37]	Feb. 2006	Oct. 2009	1338	MUST	MUST
	10	RSASHA512 [37]	Feb. 2006	Oct. 2009	1338	NOT RECOMMENDED	MUST
	12	ECC-GOST [15]†	Apr. 2009	Jul. 2010	456	MUST NOT	MAY
	13	ECDSAP256SHA256 [24]	Jan. 2011	Apr. 2012	456	MUST	MUST
	14	ECDSAP384SHA384 [24]	Jan. 2011	Apr. 2012	456	MAY	RECOMMENDED
	15	ED25519 [58]	Jul. 2015	Apr. 2017	581	RECOMMENDED	RECOMMENDED
	16	ED448 [58]	Jul. 2015	Apr. 2017	581	MAY	RECOMMENDED
DS alg.	1	SHA-1 [20]	May 2001	Dec. 2003	944	MUST NOT	MUST
	2	SHA-256 [23]	Nov. 2005	May 2006	181	MUST	MUST
	3	GOST R 34.11-94 [15]	Apr. 2009	Jun. 2010	456	MUST NOT	MAY
	4	SHA-384 [24]	Jan. 2011	Apr. 2012	456	MAY	RECOMMENDED

Table 1: Algorithms standardized for the use in DNSSEC. Recommendations from RFC 8624 [67]. Algorithms marked with † are not recommended or strongly not recommend since 2019-06-11.

3 RELATED WORK

York et al. [68] were the first to look at algorithm agility in DNSSEC and worked on an informational draft in the IETF where they identify aspects of DNS infrastructure that need to be upgraded to cope with new algorithms. Chung *et al.* [10] look at the influence of registrars on DNSSEC deployment in general. They find that a few registrars are responsible for driving DNSSEC deployment but can also create barriers. Le *et al.* [44] analyze the *quality* of DNSSEC deployments. They find that often insecure algorithms are deployed. A 2016 study by Van Rijswijk-Deij *et al.* [60] analyses early deployment of ECDSA. Finally, in recent work Müller *et al.* [**?**] perform a case study of the feasibility of using quantum-safe cryptographic algorithms in DNSSEC.

In this paper we are the first to look at the *complete life cycle* of DNSSEC algorithms. We look at the aspects of algorithm deployment, as identified by York et al. [68], and extend this work by analyzing real world data. Like Van Rijswijk-Deij et al. [60], we rely on data collected by the OpenINTEL DNS measurement platform [61], which now covers more than five years (see Section 4). This allows us to study the adoption and deprecation of additional algorithms compared to [60]. We carry out additional active measurements, providing a new perspective on algorithm deployment. Thereby, our paper has a similar focus as work by Kotzias *et al.* [41]. They show how TLS deployments and cipher suites have changed over several years. In Section 9, we compare their findings in TLS with our findings in DNSSEC.

4 DATASETS

In this paper, we rely on active and passive measurements as well as qualitative studies, which we discuss below:

Standardization. We study IETF mailing lists [35], combined with first-hand experience [55]. Our goal is to find *anecdotal* evidence that allows us to identify success-factors for, and barriers to algorithm deployment.

Support. We analyze the algorithm support of DNS software of eight signers/authoritative name servers and recursive resolvers (see Table 2). All of them are open source, which allows us to study release notes and change logs. The most popular of them, BIND, has a market share of more than 50% according to some reports [29]. We further assess the algorithm support for 20 registrars by registering a domain ourselves. For the support at registries, we carry out a survey among registries of (European) country code TLDs, responsible for managing 15 different TLDs. The survey can be found here [**?**].

Signing. To study the deployment of algorithms at domain names we rely on the daily, active, DNS measurements of OpenINTEL [61]. Among others, OpenINTEL queries daily for `DNSKEY` and `DS` RRs of all `.com`, `.net` and `.org` domains from March 2015 and `.nl` and `.se` from mid 2016. Together, these cover 6.7M signed domain names and roughly 45% of domains overall [**?**]. We focus on these TLDs because we measure them for up to 5 years and because the ccTLDs `.nl` and `.se` have the highest share of signed domain names. Additionally, we rely on archives of the root zone. From 2010 to 2014, we study our own daily copies, from the end of 2014 we rely on a public archive [17].

Validation. We measure the uptake of DNSSEC validation with RIPE Atlas [54], querying domain names under our control using their pre-configured resolvers. RIPE Atlas is a global measurement network with over 11,000 small devices called probes and over 650 larger devices called anchors. These vantage points are spread across around 9,000 Autonomous Systems. Our domains are signed with different algorithms and depending on the response we receive from the resolver we can determine the algorithms it supports. We receive responses from more than 20,000 unique IP addresses. Our measurements start in April 2017 and run every hour. We describe the details of our measurements in Section 7.2.

Ethical Considerations. Our data sets do not contain personal information. Our active measurements are carried out in cooperation with RIPE and are in line with their ethical guidelines [39]. Furthermore, rather than performing our own measurements, we re-use data collected by the OpenINTEL project [61].

5 ALGORITHM STANDARDIZATION

We begin our study of the algorithm life-cycle by looking at the *standardization* of new algorithms. Most notably, we discuss the reasons why certain algorithms were standardized and others were not, by looking at several proposals for standardization and their outcome. This provides additional insight into the potential barriers new algorithm proposals may encounter in the standardization process. The general process of standardization is explained in more detail in [7].

5.1 Process

The choice of introducing a new DNSSEC algorithm falls to the relevant IETF Working Group, and is determined by consensus. This process is started by a working group member submitting a proposal for standardization, in the form of an Internet Draft with an initial specification of how to use the new algorithm. Working group members then discuss the proposal and determine whether it will be adopted by the working group.

After adoption, the details of the draft are discussed. If working group consensus is reached, a working group last call is issued. If no issues are raised during a last call, a formal request for publication of the draft is made. After that, the document is evaluated by an IETF Area Director, sent out for another last call in the wider IETF community, and reviewed by the IESG. Finally, the proposal is reviewed by the RFC Editor for wording and consistency, and after that, is published as an RFC [32].

5.2 Barriers

This process usually takes around one year and Table 1 shows the duration for *standardized* algorithms. In some cases, however, algorithms never get standardized or standardization takes several years. This is caused by several barriers.

National Cryptographic Algorithms. Some proposed algorithms are developed by nation-states. For example, the GOST standards have their origin in the Soviet Union. A proposal to standardize the GOST algorithm for DNSSEC was adopted despite known theoretical attacks [46, 47], and despite the fact that at that time, the only available specification was in Russian. The consensus of the working group was that, as an important national cryptographic algorithm, the use of GOST should be standardized. An English translation was published during the standardization process, and the requirement in the proposal was changed from 'RECOMMENDED (to implement)' to 'MAY (optional)' [55].

Necessity and Support. Algorithms that should get standardized for DNSSEC need to be an improvement to existing ones and supported in software. In 2019, a year after the deprecation of GOST, a proposal was submitted to update the use of GOST to the revised GOST-2012. The draft update has been adopted by the working group but controversies are under discussion at the time of writing this paper. For example, missing software support and the fact that it does not outperform existing algorithms (e.g. in terms of signing speed or signature size) is criticized [6]. As another example, DSA-SHA2 was not adopted by the working group since there was little interest, as the existing DSA was not used much. The working group members preferred to either use RSA or work on ECDSA. The proposal expired without much further discussion [18].

Timing. Even though an algorithm can be widely used outside of DNSSEC, the working group might still not adopt it due to different priorities. At the time of its initial proposal, SHA256 itself was widely available, so implementation would not be a blocking issue for adoption. However, the question was raised whether it was the right time to add additional algorithms to DNSSEC. The consensus of the working group was to suspend work on the standardization of SHA256, and first focus on other DNSSEC-related work [5]. The initial proposal expired on September 30, 2006.

Unrelated design choices. The standardization of algorithms can also be blocked by issues unrelated to the algorithm itself. A second proposal to standardize SHA256 was submitted in December 2007. Over the year that followed, most of the discussion was not centered on RSA or SHA256 itself, but on a new method to prove denial-of-existence (NSEC3 [43]). Due to historic reasons, whether or not NSEC3 is used in a zone was signaled by using a separate DNSSEC algorithm identifier until that point, such as 7 for RSA-SHA1 with NSEC3 (as opposed to 5 for RSA-SHA1 with NSEC). After much discussion, the consensus of the working group was that support for RSA-SHA256 —and subsequent new algorithms— would also imply support for NSEC3, meaning that NSEC3 could be used in a zone without separate signaling. This historical anecdote also explains the two missing algorithm identifiers, 9 and 11, in Table 1; because at first SHA256/SHA512 with NSEC3 required a separate identifier, algorithm numbers 9 and 11 are marked as *reserved* [30] to this day.

Takeaways for future algorithms. Every algorithm is different, which makes it hard to draw general conclusions. Still, from our observations we see that algorithms that do not outperform existing algorithms, e.g. in terms of signature size or better security, have a harder time getting standardized. Sufficient software support is recommended and national algorithms are met with some skepticism. Meanwhile, DNSSEC itself has matured such that changes in the design of the protocol are likely not a barrier for algorithm standardization anymore.

6 ALGORITHM SUPPORT

After standardization of an algorithm, the software responsible for creating and validating signatures must adopt the algorithm. Also, the entities responsible for registering domain names and propagating key material must be able to process the new keys and signatures. In this section, we show how both affect algorithm adoption.

6.1 Software

Three different requirements need to be met such that an algorithm is *fully* supported in software: (i) the DNS software, used to create and validate signatures, needs to support the algorithm,

Software	OpenSSL	GnuTLS	Libsodium	Libdecaf	Botan	Task	ECDSAP256 (Days)	ED25519
BIND 9	✓*					*S,V*	2012-10-09 (179)	2018-01-23 (343)
Knot Resolver		✓*				*V*	(at release 2016-05-30)	2017-09-29 (227)
Knot DNS		✓*				*S*	(at release 2016-05-30)	2017-09-29 (227)
LDNS	✓*					*S*	2012-05-21 (38)	2019-07-26 (892)
OpenDNSSEC	✓*				✓*	*S*	2017-02-22 (1,776)	—
PowerDNS Auth. Server	✓*		✓	✓		*S*	2016-07-11 (1,550)	2017-06-23 (129)
PowerDNS Recursor	✓*		✓	✓		*V*	2017-11-27 (2,054)	2017-06-13 (119)
Unbound	✓*					*V*	2012-04-13 (0)	2017-05-30 (105)

Table 2: Popular software used for signing (*S*) and validation (*V*), their supported cryptographic libraries (*Default) and their support of ECDSAP256 and ED25519 (days after standardization).

(ii) cryptographic libraries must support the algorithm, and (iii) it must be possible to install both components (signer software and cryptographic library) on an operating system. We discuss how long it took until each of these three components supported the algorithms ECDSAP256 and ECDSAP384, and ED25519 and ED448. Also, we show if deprecated algorithms are still supported.

6.1.1 DNS Software. DNS software *usually* is either responsible for *signing*, and possibly serving DNS zone files, or for resolving domain names and *validating* the signatures. Authoritative name servers, in principle only need to publish a record, and do not need an understanding of the underlying cryptographic functions. Table 2 lists 8 common open source DNS implementations and the tasks they fulfill.

Unbound and LDNS supported ECDSAP256 within a few weeks and BIND9 followed within a few months, but for other software it took up to 5 years. In the former case, Unbound already provided *running code* during the algorithm standardization process [50] and one of the developers co-authored the standard [24]. This could have sped up its support. The resolver PowerDNS did support ECDSA before 2017 but relied on a cryptographic library which had to be installed manually. When PowerDNS Authoritative Server added support for ECDSA *out of the box* it also started using ECDSAP256 for signing by default [52].

By May 2020, all but one signer implementation support ED25519. All popular resolvers added support within a year after standardization. Even though every implementation supports the stronger version of ECDSAP256 (ECDSAP384), the stronger version of ED25519 (ED448) is not supported by Knot and PowerDNS requires an additional library. Software vendors might skip implementation of ED448 since it is expected that ED25519 will become the recommended default [67].

Support of *deprecated* algorithms varies between DNS software. Unbound added the *option* to treat SHA-1 based algorithms as insecure in 2017 [50]. BIND9 and the name server of PowerDNS removed support for GOST in 2019 and 2020 respectively [36, 53]. BIND9 additionally removed support for DSA and DSA-NSEC3-SHA1 in the same version. From then on, the software treats these algorithms as *insecure*. From our own experience [1] we know that support in libraries and RFC recommendations play an important role when deciding which algorithm should be supported.

[1]Some of our authors work for NLnet Labs, the developers of Unbound and NSD.

6.1.2 Cryptographic Libraries. Algorithm support in DNS software relies on the use of cryptographic libraries. The majority of DNS implementations automatically uses OpenSSL for cryptographic operations if detected (see Table 2). Libraries such as libsodium and libdecaf provide *additional* algorithm support for some software.

Libraries do not need to support new algorithms in order for them to be standardized for DNSSEC or vice versa. Whereas all major libraries supported ECDSA already *before* it was standardized for DNSSEC, ED25519 and ED448 were only supported *after* the RFC was published. Also, OpenSSL stopped supporting GOST two years prior to its deprecation in the IETF, but GnuTLS only started supporting GOST one year before its deprecation.

In case of ED25519 end ED448, libraries that supported these algorithms did exist prior to standardization. The initial releases of libdecaf and libsodium were in 2013 and 2014 respectively [12, 22]. Both of these libraries, however, typically need to be installed manually by operators (*i.e.,* are not part of default OS installations).

6.1.3 Operating Systems. The software and libraries discussed above can be installed manually or can be downloaded as a *package*. Since the latter approach is much more convenient, we assume most operators prefer it to manually compiling and installing DNS software. To understand how long it takes until operating systems provide DNS software supporting the different algorithms *of-the-shelf*, we look at two popular operating systems with different software management approaches: Ubuntu and OpenBSD.

The versions of software shipped with a particular Ubuntu release is not usually updated [8] and for stability reasons, operators prefer releases with long-term support (LTS). In case of ED25519, this meant that even though all popular DNS software in Ubuntu 18.04 LTS already supported the algorithm, the OpenSSL version did not. It required a *stable release update* for OpenSSL late 2019 for full algorithm support – more than 2.5 years after standardization.

In contrast, in OpenBSD new software releases are added when the corresponding package is updated by its maintainer [26]. Therefore, OpenBSD users generally receive software updates faster, including support for new algorithms.

6.2 Registration Ecosystem

Besides DNS software, the parties involved in publishing the DNSSEC records must also support new algorithms. The registrar needs to be able to upload the DNSKEY record or the DS record to the registry and the registry must be able to publish the DS record.

Registrar (Authoritative Nameservers)	Default Algorithm	DS Upload	DS Algorithm Support: RSAMD5	Diffie-Hellman	DSA	RSASHA1	DSA-NSEC3-SHA1	RSASHA1-NSEC3-SHA1	RSASHA256	RSASHA512	ECC-GOST	ECDSAP256	ECDSAP384	ED25519	ED448	PRIVATEDNS	PRIVATEOID
Alibaba (hichina.com)	13	Web	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
GoDaddy (domaincontrol.com)	13	Web	●	●	●	●	●	●	●	●	●	●	●	✗	✗	●	●
NameCheap (r...-servers.com)	13	Web	●	✗	●	●	✗	●	●	●	●	●	●	✗	✗	●	●
Google (googledomains.com)	8	Web	✗	✗	●	●	●	●	●	●	●	●	●	●	●	✗	✗
OVH (ovh.net)	7	Web	✗	✗	✗	●	✗	●	●	●	✗	●	●	✗	✗	✗	✗
1AND1 (1and1)	8	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Network Solution (worldnic.com)	✗	Email	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
NameBright (namebrightdns.com)	✗	Email	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
register.com (register.com)	✗	Email	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
Amazon (aws-dns)	✗	Web	●	●	●	●	●	●	●	●	●	●	●	✗	✗	●	●
DreamHost (dreamhost.com)	✗	Email	●	●	●	●	●	●	●	●	●	●	●	✗	✗	✗	✗
Rightside (name.com)	✗	Web	✗	✗	●	●	●	●	●	●	●	●	●	●	●	✗	✗
eNom (name-services.com)	✗	Email	✗	✗	●	●	●	●	●	●	✗	✗	✗	✗	✗	✗	✗
123-reg (123-reg.co.uk)	✗	Email	✗	✗	●	●	✗	●	●	●	●	●	●	✗	✗	✗	✗
HostGator (hostgator.com)	✗	Email	✗	✗	✗	●	✗	✗	●	●	✗	●	●	✗	✗	✗	✗
Bluehost (bluehost.com)	✗	Email	✗	✗	✗	✗	✗	✗	●	●	●	●	●	✗	✗	✗	✗
WIX (wixdns.net)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
WordPress (wordpress.com)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Yahoo (yahoo.com)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Xinnet (xincache.com)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Table 3: Table showing the results of our study of algorithm support for popular 20 registrars; only 6 registrars support DNSSEC on their nameservers with 3 different algorithms, five of which (except 1AND1) also support DNSSEC when the owner is the DNS operator by allowing the registrant to upload a DS record. 15 registrars support DNSSEC for external nameservers but only 4 of them support all signing algorithms.

6.2.1 Registrar. Registrars play a critical role in deploying DNSSEC; they have an almost exclusive role creating a chain of trust by uploading a DS record for a domain name to the registry.[2] It is thus crucial to understand how their signing algorithms are chosen and what other algorithms are supported. Moreover, earlier work by Chung *et al.* [10] showed that 20 out of 31 DNS operators that manage at least 54.3% of .com, .net, and .org domains are registrars[3], which indicates that default authoritative nameservers provided by registrars contribute significantly to the DNS and DNSSEC ecosystem.

Registrants usually have two options for deploying DNSSEC: they can use the default name servers provided by the registrar (thus, the owner does not have any control over choosing algorithms) or external name servers such as the ones managed by the domain owner or a third-party operator such as Cloudflare. For the latter, the owners can freely choose an algorithm to sign their DNS records, but still need to communicate with their registrar to build a chain of trust by uploading a DS record through a web interface on the registrar website or via an e-mail; the owners convey four fields of the DS record to the registrar: (1) the hashing algorithm of the DS record (*i.e.,* the digest type of the linked KSK), (2) the digest (hash of the linked KSK), (3) the key tag of the linked KSK, (4) algorithm number of the linked KSK. In principle, registrants can upload *any* DS *record regardless of which algorithm was used for their KSK* to their registrar because (1) the key tag and algorithm number of the linked KSK are just hints for resolvers to quickly identify the KSK when multiple KSKs are presented and (2) the uploaded DS records will be stored in the database of the registry, thus the registrar does not need to store them in their name servers.[4]

To understand how popular registrars support DNSSEC algorithms, we first register a .com domain via the top 20 popular registrars (the same list as used by Chung *et al.* [10], where 50% of all domains are registered) and examine their algorithm support both when the registrar is the DNS operator and when the owner is the DNS operator; Table 3 summarizes the results of this experiment.

There, we find that six registrars support DNSSEC on their default name server, four more than reported by Chung *et al.* in 2017 [10]. When focusing on default algorithms that they support on their name servers, we observe that three registrars (Alibaba, GoDaddy, NameCheap) use ECDSAP256 and two registrars use RSASHA256, both of which are "must implement" per the best current practice [67]. However, we also notice that OVH still uses RSASHA1-NSEC3-SHA1, which is known to be vulnerable to a hash collision attack, thus not recommended for signing.

Surprisingly, when the owner is the DNS operator, we find that not all registrars allow owners to choose any algorithm to sign their records; only four registrars (Alibaba, Network Solutions, Namebright, and Register.com) support all algorithms. They even support algorithms 253 (PRIVATEDNS) and 254 (PRIVATEOID), which are the numbers for private algorithms and will never be assigned to a specific algorithm.

However, the other 11 registrars partially support specific algorithms, which ultimately restricts the set of algorithms that the domain name owners can use to sign their DNS records: considering that an algorithm number in a DS record has to be matched with

[2]The CDS and CDNSKEY protocols [21, 42] allow DNS operators to upload their DS records directly to the registry; however, even after three years since their standardization, only the .cz and .ch registries have deployed it.

[3]The other 11 DNS operators are third-party DNS operators and domain parking services, but only one DNS operator (Cloudflare) among them supports DNSSEC.

[4]Registrars may want to check the validity of the uploaded DS record before forwarding it to the registry, but a previous study showed that registrars rarely do so [10].

the algorithm number of the corresponding KSK, limiting the pool of algorithm numbers of a DS record can have a side effect; domain owners may not able to freely choose an algorithm for their DNSKEY *even if they manage their own DNS authoritative servers*; for example, domain name owners who purchase domains from eNom cannot sign their DNS records with ECDSA algorithms (*i.e.*,ECDSAP256 or ECDSAP384).[5]

We also observe that relatively new signing algorithms are rarely supported; for example, we find that only six registrars among 15 registrars support external DS records support algorithms based on EdDSA (ED25519 and ED448), which were standardized in April 2017. Overall, these results signify that registrars also have a key role for a broader adoption of newer signing algorithms.

6.2.2 Registry. To understand if registries have an influence on the deployment of algorithms, we asked the members of the association of European country code top-level domain registries, CENTR [11], which algorithms they support. We received responses for 15 different TLDs, operated by 13 different registries. Additionally, we know that .com and .net do not restrict the choice of algorithms.

Combining the results, four TLDs support *all* algorithms. One only denies DS records with algorithm numbers 252-255, reserved for private algorithms or for other purposes. Seven TLDs support all *recommended* algorithms but not all deprecated algorithms. Five TLDs do not support the latest algorithms ED25519 and ED448 and one does not support algorithm ED448. Registries that limit the number of supported algorithms, always deny GOST. Nine TLDs also do not support algorithms DSA and DSA-NSEC3-SHA1 and three TLDs do not support RSASHA1 and RSASHA1-NSEC3-SHA1.

Five registries that do not support ED25519 plan to support it by the end of 2021. One registry stated the lack of debugging tools as a reason why ED448 is not supported. This likely refers to DNSViz, a popular tool to visualize and test DNSSEC deployments which did not support ED448 at the time of writing (May 2020) [13].

Three registries do not support some of the deprecated algorithms because of security concerns. Others actively promote the use of certain algorithms. .se gives financial incentives for signing with algorithms RSASHA256, RSASHA512, ECDSAP256, ECDSAP384, ED25519 and ED448 [66] and .nl is following this year. One other registry is promoting the use of algorithms ECDSAP256 and ECDSAP384 but is not handing out incentives. Of the operators that allow all algorithms, one of them has a very liberal policy for what they accept in their zone and thus also do not want to restrict the selection of algorithms [**?**].

Takeaways for future algorithms. Since all studied DNS software relies on additional libraries for cryptographic functions, algorithm support in those libraries is crucial. OpenSSL and GnuTLS are most used and new algorithms should get implemented in both libraries to become widely supported. Registrars and registries often are slow when adding support for new algorithms. Operators that want to deploy a new algorithm should bring their intention forward to encourage registrars and registries to add support. Both also play a major role when *deploying* new algorithms, which we show in the next section.

[5]Domain name owners may be able to deploy ZSKs with ECDSA algorithms, and choose a different algorithm for the KSK; however, this trick is prohibited by RFC 6840 and often causes resolvers to fail validation [64].

7 ALGORITHM DEPLOYMENT

After software, registrars and registries support an algorithm, it can be deployed at (i) signed domain names and (ii) validating resolvers.

7.1 Signing

For signing, we focus on the ECDSA-based algorithm ECDSAP256 since it is widely deployed and our data sets cover its deployment almost entirely. ECDSA algorithms with curve P-256 and P-384 were already standardized for DNSSEC in 2012. ECDSA has smaller signatures and keys compared to RSA while achieving a similar security level. It is less prone to issues during transport and makes DNSSEC less attractive for Denial of Service Attacks [62].

7.1.1 Slow Uptake. Even though signers BIND9 and LDNS already supported ECDSAP256 in 2012, we only see 21 domains signed in Figure 3 in 2015. By the end of 2015, 11K domain names were signed with ECDSAP256, 98% of them operated by Cloudflare. This can be explained by the fact that in September that year, Cloudflare started a public beta, allowing their customers to enable DNSSEC for their domain names using ECDSAP256 [19]. In 2016, Knot and the authoritative name server of PowerDNS started supporting ECDSA as well. Shortly after, the number of ECDSAP256 domains in .com rises significantly, which can be attributed to domains operated by domainnameshop.com.

Of the domains that were signed with ECDSAP256 by the end of 2015 and still registered by the end of 2016, 82% were still signed with the same algorithm. Still in 2014, reports discouraged operators from signing with ECDSAP256 because of issues with validating resolvers [28]. The fact that the majority of domains did not move away from this algorithm shows these issues did not affect signed domains in 2015 anymore.

In the same year .nl started supporting ECDSA, but initial uptake was lackluster. By the end of 2016 ECDSAP256 accounted for just 0.2% of signed .nl domain names whereas by the end of 2015, already 1.5% of signed domains at .com, .net, .org used ECDSAP256. The most likely explanation for the slow uptake in .nl is that operators that already had DNSSEC enabled (at that time almost half of domains in .nl) saw no good reason to switch to a different signing algorithm.

From 2019 ECDSAP256 domains in .com grew three times and .se even 12 times. First, 100K domains get signed with ECDSAP256 at once in January. The vast majority (99%) of those domain names are operated by Binero, a Swedish registrar. Also the number of ECDSAP256 domains at .com increases by 26K – also caused by Binero. The second jump occurs in June. Also in this case, one single operator is responsible for 99% of new signatures (One.com). These jumps correlate with the adjustment of the DNSSEC incentives by the Swedish registry. Registrars already received discounts for DNSSEC deployment [44]. But in November 2018 .se announced that they would adjust their incentives, requiring that domains are signed with either ECDSAP256 based algorithms, Edward Curve based algorithms or with RSASHA256 and RSASHA512 (with an adequate key length) [66]. On July 1, 2019, the new incentives came into force.

7.1.2 Large Operators. From the preceding analysis, it appears that DNS providers (registrars or third-party operators) drive adoption.

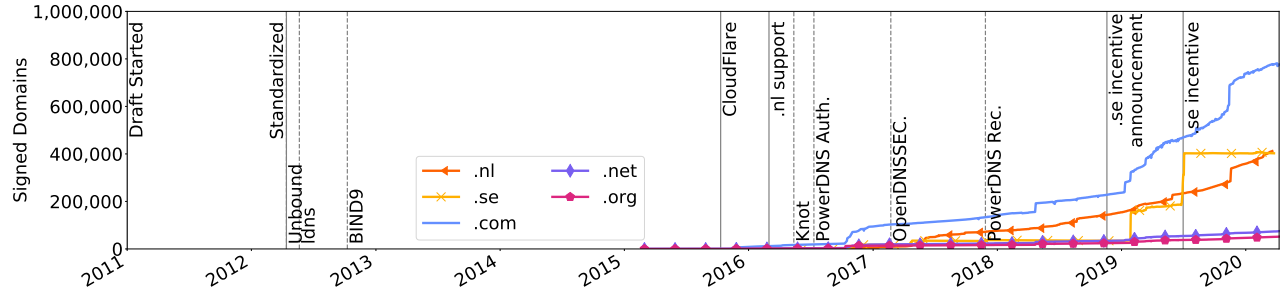


Figure 3: Domains signed with ECDSA256 and resolvers able validating this algorithm

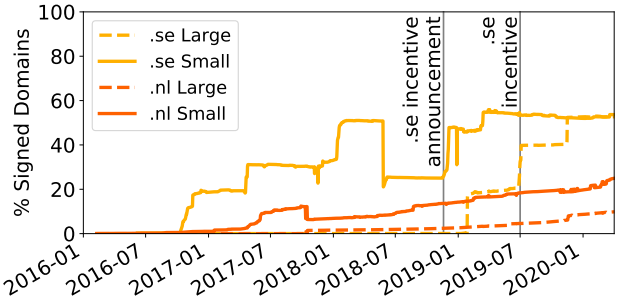


Figure 4: Share of `.se` and `.nl` domains signed with ECDSAP256, by provider size

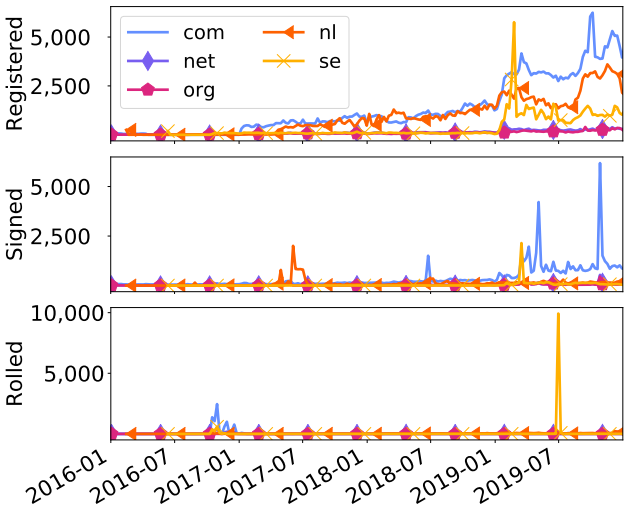


Figure 5: Newly signed with ECDSA per week

We verify this by grouping domain names by DNS provider, using the `MNAME` in the `SOA` record (previously applied in [44]). Then, we sort the providers by size in descending order and label the first providers that together are responsible for 80% of the signed domains as large, the others as small.

In absolute numbers, the large majority of domains are indeed administered by large providers (90% by the end of April 2020). If we look at the share of domains per provider group in Figure 4, then 52% of signed domains by large providers use ECDSAP256, compared to 45% by small providers. Small providers, however, adopt ECDSAP256 *faster* than large providers. Only after the `.se` registry changed its incentives, large providers caught up. This is also true for `.nl`, where 20% of signed domain names at small providers use ECDSAP256, compared to 10% at larger providers. At `.com`, `.net` and `.org`, ECDSAP256 adoption at small and large providers have reached similar levels, with a difference of 3% and 1%. The drop in Figure 4 in May 2018 is caused by one provider signing 99K domains with RSASHA256, growing the total number of signed domains significantly and thus reducing the share of ECDSAP256 domains.

7.1.3 Algorithm Rollovers. For a protocol to achieve algorithm agility it is necessary that deployments can easily *roll* from one algorithm to another. To understand if this is also the case in DNSSEC we check for each day in our data set (a) which domain names are now signed with ECDSA and not registered a week before (*newly registered domains*), (b) which were registered a week before but not signed (*newly signed domains*) and (c), which domains were signed the day before, but with a different algorithm (*rolled domains*).

Figure 5 show that only a minority of domains deploying ECDSA were signed before. The majority of domains are newly registered or not signed before (69% and 24% of all domains). Occasionally large numbers of domain names roll to ECDSA. For example, `domainnameshop.com` rolled 10K domain names from RSASHA256 to ECDSAP256 in October 2016, and in July 2019 `one.com` did the same for more than 80K domain names. Algorithm rollovers can cause outages if they are not carried out correctly and operators might stop signing their domains before moving to ECDSA. For example, before Binero started signing its `.se` domains with ECDSA in January 2019, they stopped signing them for more than one month. We see their return in Figure 5 (center) as a spike in newly registered domain names.

Newly signed domains, often get signed after changing their DNS operator. We compared the operator on the day a domain was signed with ECDSA with the operator a week before for domains of `.net`. 33% of newly signed domains changed DNS operator just before ECDSA signing was enabled. Operator-change rarely leads

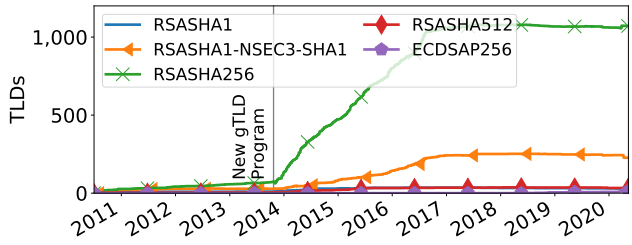


Figure 6: Algorithms used to sign TLDs

to an *upgrade* to ECDSA. In less than 1% of all algorithm rollovers operators were changed at the same time.

7.1.4 Transition at the Root. The root zone was signed in July 2010 and two months later 35 TLDs were signed. From that point on RSASHA256 (8) was the most used signing algorithm and still is today (Figure 6). In October 2013, the first new generic TLDs (gTLD) were delegated [31]. For new gTLDs, DNSSEC deployment is mandatory [33] and the majority choose RSASHA256. In 2018 we see the first TLD signed with ECDSA, three years after we see the first second level domain signed with this algorithm. Today, only 14 TLDs rely on ECDSA making it the least used algorithm at TLDs.

All of the TLDs that are now signed with ECDSA are country code TLDs (ccTLD). Eight ccTLDs rolled from RSASHA256, even though that algorithm is still considered secure. Two TLDs rolled from algorithms that are not recommended anymore. This indicates that security has only played a minor role when moving to ECDSA.

The main barrier to adoption of ECDSA for TLDs is likely the complexity of algorithm rollovers. Since April 2017, more than 90% of TLDs are signed and since the total number of TLDs did not increase since then, the only way of adopting ECDSA is by algorithm rollover. Since 2010, however, only 68 algorithm rollovers were carried out *in total*, demonstrating reluctance at TLD operators.

7.2 Resolver Validation

We test resolver validation support for `DNSKEY` algorithms using RIPE Atlas, by sending two DNS queries for each `DNSKEY` algorithm: one for a validly signed domain name and one with a broken signature. A resolver is considered as validating a certain algorithm when an answer is returned for the validly signed name (return code `0`) *and* an error response for the invalid name. If responses containing an answer are received for both names, the resolver is considered not validating. All other combinations are considered resolver failures for the specific algorithm. At the time of writing only two resolvers have this failure; Google Public DNS resolver [65] and the OpenDNS resolver fail on RSA-MD5 (1).

The DNS zones for all the DNSKEY algorithms are subzones of `rootcanary.net` [59]. The DS record for `.net` uses DS algorithm SHA-256 which resolvers minimally need to support to be able to validate any of our subzones for the DNSKEY algorithms. Therefore all subzones for the DNSKEY algorithms, as well as the `rootcanary.net` domain, also use `DS` algorithm SHA-256 in the secure delegation.

Likewise, the root and `.net` (and `rootcanary.net`) are signed with `DNSKEY` algorithm RSA-SHA256 which thus needs to be minimally supported to validate any of our subzones. We measure `DS` algorithm support by testing for `DNSKEY` algorithm RSA-SHA256 in subzones using the different `DS` algorithms in the secure delegation. According to [3] (section 5.2) unsupported `DS` algorithms should be treated as non-existent, so if an otherwise validating DNSSEC resolver is detected to be *not validating* with a zone which uses a specific `DS` algorithm, than that resolver is considered to not support that `DS` algorithm. This does not work perfectly; The Google Public DNS Resolver validates a zone for which it supports the `DNSKEY` algorithm and for which it detects a `DS` record in the parent with a corresponding keytag, even when it does not support the `DS` algorithm.

7.2.1 DNSKEY algorithm support in validating resolvers. Figure 7 shows the overall progression of `DNSKEY` algorithm support in DNSSEC validating resolvers measured via RIPE Atlas. The amount of support per algorithm is relative to the number of resolvers capable of validating RSA-SHA256; the `DNSKEY` algorithm which needs to be supported minimally for DNSSEC validation. A few algorithms are left out as they have largely the same progression as algorithms which are shown; DSA-NSEC3-SHA1 support is identical to DSA; RSASHA1-NSEC3-SHA1 support is identical to RSASHA1; RSASHA512 support is identical to RSASHA256 at 100% throughout the measurement period; ECDSAP384SHA384 support is identical to ECDSAP256SHA256.

The dent in ED25519 support and the slight bumps in RSAMD5, DSA and ECC-GOST support are due to a bug in Cloudflare's DNSSEC validation which was disabled for all (freshly signed) domains in August 2018. The erratic progression in April 2019 was due to DNSSEC issues with DNSSEC validation in Google Public DNS resolvers. This impacted our measurements more severely than real users experienced, as Google monitored and debugged their DNSSEC validation with the help of our domains and measurement results, where they had mitigations for other domains. The decrease of ED25519 support in March 2020 is relative. At that time there was an increase of new RIPE Atlas probes with DNSSEC validating resolvers, however without ED25519 support.

Figure 7 also has vertical markers indicating when algorithm ED25519 and ED448 gained support in software, libraries and distributions. This allows us to determine which support in software, library or distribution caused which uptake of DNSSEC validation.

7.2.2 The uptake of ED25519 and ED448. At the time of writing, on RIPE Atlas, almost 70% of the DNSSEC validating resolvers support ED25519, and almost 17% supports ED448. To understand who supported these algorithms first, we also run measurements exposing the IP address seen at the authoritative (`o-o.myaddr.l.google.com TXT` and `whoami.akamai.net A` amongst others [14]). Those measurements are also scheduled for all resolvers on all RIPE Atlas probes and anchors, for every hour. This allows us to identify resolvers located in the same network as the probe (*internal*), resolvers that are in the same network but forward their queries (*forwarding*) and resolvers that are located in a different network (*external*). Whereas internal resolvers are mostly resolvers of local networks or ISPs, external resolvers are the ones run by large DNS providers like Google or Cloudflare.

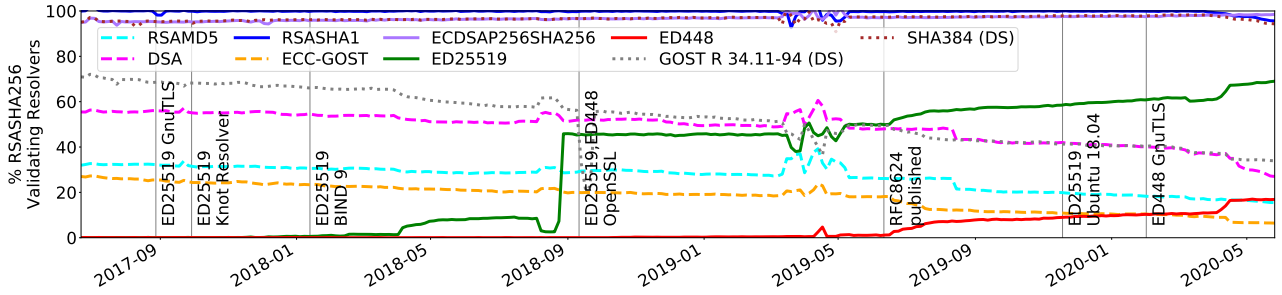


Figure 7: Overall progression of DNSKEY algorithm support in DNSSEC validating resolvers, measured via RIPE Atlas

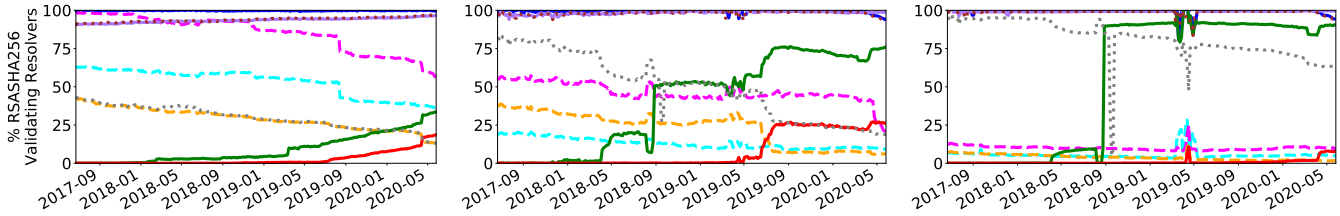


Figure 8: Validation at internal, forwarding and external resolvers

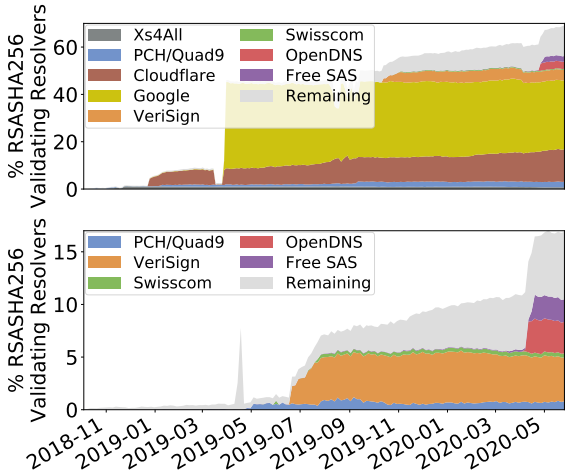


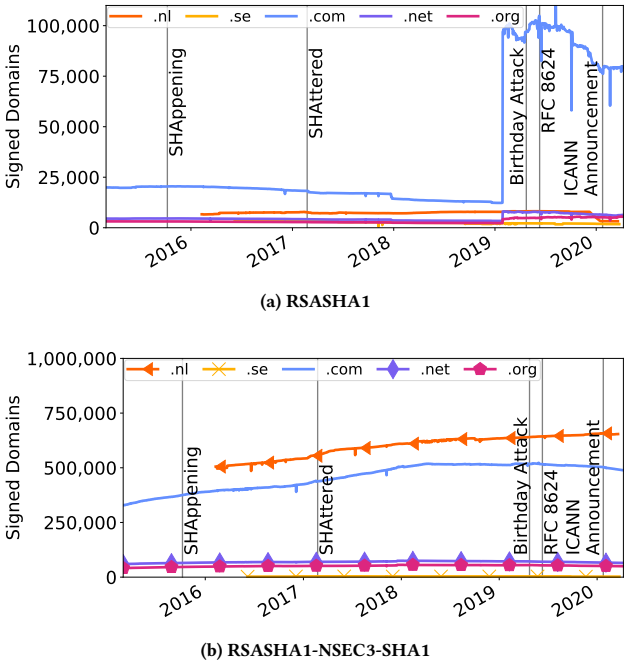
Figure 9: Uptake of ED25519 and ED448 validation

In Figure 8 (left) we can see that internal resolvers adopt ED25519 slower than external resolvers (Figure 8 right). We therefore zoom in on external resolvers.

Figure 9 shows the top Autonomous System Numbers (ASNs) associated with those IP addresses as seen at the authoritative for resolvers that support validating ED25519 and ED448 respectively. Looking at top ASNs, we notice that the resolvers within ASNs that support ED448, do also support ED25519, but not vice versa. Moreover, except for AS42 (PCH/Quad9), all other top ASNs that support ED448 started supporting both Edwards-Curves simultaneously. The simultaneous support of ED25519 and ED448 might indicate software using cryptographic libraries that support both, such as OpenSSL since September 2018 or GnuTLS since February 2020 (used by Unbound and Knot Resolver and optionally with PowerDNS) opposed to crypto libraries that support only one of them, such as libsodium and libcafe (used with PowerDNS).

AS42 (Quad9) is the only one that supported ED25519 first (already noticeable since January 2018) and later gained support for ED448 (in May 2019). We know from private communication that AS42 uses a mix of PowerDNS and Unbound. This suggests a deployment of PowerDNS linked against libsodium and Unbound linked against a recent OpenSSL since May 2019. The number of resolvers from AS42 on RIPE Atlas supporting both algorithms is twice the amount of the ones supporting only ED25519, suggesting that on RIPE Atlas we hit about 50% PowerDNS and 50% Unbound instances.

Takeaways for future algorithms. Operators only move to a new algorithm if there is an *incentive*, e.g. a smaller signatures or a financial reward. Security threats encourage a rollover only if the threat is imminent. This is the most important takeaway if new algorithms should be supported widely. Algorithm rollovers are still perceived as dangerous. If not only new domains should deploy a new algorithm but also already signed domains, then their operators need to have tools and documentation at hand to have the confidence to carry out such an algorithm rollover. The number of resolvers validating a new algorithm rises gradually with every OS upgrade if a new algorithm is implemented in the standard cryptographic libraries and resolvers. However, if a new algorithm should gain wide support fast, then operators of large resolving services can speed up its adoption.



(a) RSASHA1

(b) RSASHA1-NSEC3-SHA1

Figure 10: Deployment of deprecated algorithms

8 ALGORITHM DEPRECATION AND REPLACEMENT

DNSSEC algorithms can also be *deprecated*. This happens when the IETF publishes a new standard, advising against the use of an algorithm [67]. In practice, zones should not sign with an algorithm that is not recommended, but a resolver should still treat signatures as secure. Only if an algorithm must not be used anymore, validating resolvers must treat signed RRs as unsigned.

Since the standardization of DNSSEC, three algorithms must not be used to sign zones anymore: DSA, DSA-NSEC3-SHA1 and ECC-GOST. For security reasons, two more are not recommended for signing anymore: RSASHA1 and RSASHA1-NSEC3-SHA1. Additionally, SHA-1 and GOST should not be used when creating DS records.

8.1 Signing

The cryptographic hash function SHA-1 is part of four algorithms standardized for DNSSEC. DSA/SHA-1 and RSASHA1 are part of the original DNSSEC specifications [4], DSA-NSEC3-SHA1 and RSASHA1-NSEC3-SHA1 were standardized three years after. Since 2010, NIST disallows use of SHA-1 for governmental agencies [51]. Attacks in 2015 [38] (referred to as SHAppening), 2017 [57] (SHAttered) and 2019 [45] (Birthday Attack) undermined its security further. Also in 2019, RFC 8624[67] advised against using 3 and 6 for signing and validation and recommends against signing with RSASHA1 and RSASHA1-NSEC3-SHA1. Resolvers should still validate the latter. RFC 8624 now also strongly advises against using SHA-1 for creating DS records. In January 2020, ICANN also asked operators to no longer use SHA-1 based signing algorithms [25].

We analyze the effect of the attacks on SHA-1 and the official deprecation of related algorithms. Already in 2015 and at the beginning of our measurements, fewer than 400 domain names in our data set were signed with DSA and DSA-NSEC3-SHA1 and this halved by May 2020. We therefore focus on signing algorithms RSASHA1 and RSASHA1-NSEC3-SHA1 (not recommended) and DS algorithm SHA-1 (must not be used).

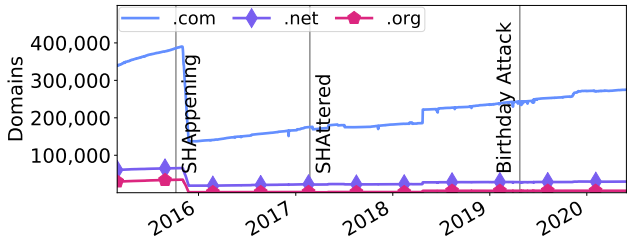
8.1.1 Second level domain names. Figures 10a and 10b show the number of domains using DSA and DSA-NSEC3-SHA1. RSASHA1 is the less common algorithm of the two, and at the beginning of our measurements 27K domain names in .com, .net or .org used it for signing. The impact of the published attacks is visible. After *SHAppening*, the total number of domains signed decreased by 6% within one year and after *SHAttered* by another 20% . End of January 2019, however, RSASHA1 sees a revival. In April the same year, another attack was announced and in June RFC 8624 recommends against using RSASHA1 for signing. Half a year later the share of domains decreased again by 25%. Since end January 2020, however, the total number of domains is rising again. More than 99% of these domains are operated by NameBright, which seem to add the same DNSKEY with this algorithm to each domain by default without actually signing the other records.

In contrast RSASHA1-NSEC3-SHA1 still remains popular. The number of domains signed with RSASHA1-NSEC3-SHA1 is increasing in *absolute* numbers in the first 3 years of our measurements. At its peak in May 2019, 500K .com domain names are signed, a growth of 50% since the beginning of our measurements. In *relative* numbers, however, the share of signed domains decreases by 37%. In June 2019, RFC 8624 was published and until ICANN's announcement not to use SHA-1 anymore, the usage decreases by 18%. Since the announcement, the share dropped by another 8%. By May 2020, 25% of signed domains rely on RSASHA1-NSEC3-SHA1. For 90% of these domains, only three providers are responsible: ovh.net, transip.net and anycast.me.

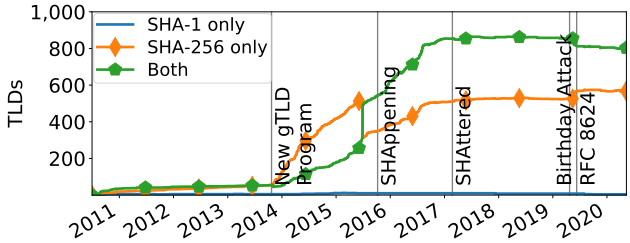
Of the domains that were signed with RSASHA1-NSEC3-SHA1 at its peak and that were still registered one year later, 93% were still signed with the same algorithm. 6% were unsigned and only 1% were signed with a different one. For domains signed with RSASHA1, 36% are unsigned and only 10% of the signed domains have rolled to another algorithm a year after its high. This indicates that the decline of both algorithms can be linked more to canceled domains or domains that turned off DNSSEC than to algorithm rollovers.

8.1.2 TLDs. Early deployments at TLDs preferred mostly RSASHA1 but RSASHA1-NSEC3-SHA1 and especially RSA/SHA-256 gained ground fast (see Figure 6). RSASHA1-NSEC3-SHA1 reached its highest deployment in February 2018 and shrinks continuously since then. Although hardly visible, in April 2020, 16 TLDs moved away from signing with RSASHA1-NSEC3-SHA1. These, however, all belong to the same provider.

8.1.3 DS Algorithms. SHA-1 is also used to create DS records that link the KSK of the child with the ZSK of the parent. Operators can choose between four different hash algorithms (see Table 1). At .nl, DS records are created by the registry, resulting in SHA-256 DS records only. At the other TLDs all four algorithms are seen.



(a) TLDs: SHA-1 only



(b) Root: All combinations

Figure 11: DS digest algorithm

Figure 11a shows the number of domains that publish a DS with SHA-1 only. Right after the attack on SHA-1 end of 2015, the number of domains in .com using a SHA-1 DS halves. This was caused by domains operated by OVH transitioning from SHA-1 to SHA-256. By May 2020, domains in .com, .net and .org mostly moved away from SHA-1 – more than 75% publish a SHA-256 DS only. At .se, 60% publish both. 80% of .com domains that also publish DS records with SHA-1 belong to only 5 operators.

At the root only SHA-1 and SHA-256 are supported [34]. Figure 11b shows, only in the first days of DNSSEC deployment SHA-1 was more often used than SHA-256. After new gTLDs are added, TLDs only publishing a DS with SHA-256 become the majority. In June 2015, this changes after TLD operator AriDNS carries out a KSK rollover for its new gTLDs and additionally adds a DS record with SHA-1. From then on, the majority of TLDs have DS records with both algorithms. In 2019, and around the time another attack against SHA-1 was published, one operator removed additional SHA-1 DS records. By the end of May 2020, 59% of TLDs still have a DS record with SHA-1 published in the root.

8.2 Resolver Validation

Algorithms RSAMD5, DSA and DSA-NSEC3-SHA1 must not be considered secure by resolvers [67]. GOST may still be validated.

Figure 7 shows support for algorithms RSAMD5 and DSA is declining even before they were officially obsoleted in June 2019. Support declines more rapidly after publication of the RFC. Internal resolvers deprecated algorithms slower than forwarding and external ones, but they are catching up (Figure 8a). At the beginning of our measurement 60% still supported RSAMD5, three years later this is only 36%. Only very few domains are affected: already at the beginning of our active measurements in 2015, only 300 domains were signed with these algorithms. By May 2020, only 180 are left.

Support for ECC-GOST, although marked as 'MAY' in RFC 8624 for DNSSEC Validation, is also declining and is currently the least supported DNSKEY algorithm on RIPE Atlas. Also here, the publication of RFC 8624 [67] accelerates deprecation further. Only 23 domain names in our data set are signed with GOST, and will now be considered *insecure* by the vast majority of resolvers.

RSASHA1-NSEC3-SHA1 is deprecated for signing but not for validation and resolvers must still be able to validate it. Figure 7 shows that 96% of resolvers in our data set still do so.

Implications for future algorithms. If in the future insecure algorithms should be deprecated fast then DNS software and registration channels need to remove support as well. Removing support at signers makes it harder for operators to sign their domains with deprecated algorithms. Also, if resolvers stop validating these algorithms, operators do not have an incentive to use them for signing. At the same time, operators, again, should be confident rolling their algorithm. Only then, the insecure algorithm is actually replaced with a more secure one.

9 DISCUSSION AND OUTLOOK

At the start of this paper, we asked: *has DNSSEC achieved algorithm agility?* Based on our analysis, we conclude, rather unsatisfactorily: *only partially*. The introduction of ECDSA has shown new algorithms *can* get standardized, gain wide support in software, registrars and registries, resolvers and get deployed at domain names. But only over the better part of a decade. The example of ECDSA, but also other algorithms, shows the DNSSEC protocol itself is rarely the obstacle but rather the slow adoption at registrars, lacking of-the-shelf software support, and hesitant DNS operators. Financial incentives may help, though.

However, when replacing a deprecated algorithm with a new one the DNSSEC protocol appears to be a barrier. *Algorithm rollovers* are still rarely carried out, likely because of their complexity, but also because of lacking incentives. Most used algorithms are still secure *enough*. On the upside, TLD operators slowly seem to exercise algorithm rollovers more often. On the down side, the root zone only carried out its first KSK rollover in 2018 [48], with no algorithm rollover in sight.

We can also see similarities with TLS. Kotzias et al. [41] showed that also in TLS, transition to more secure versions takes time. TLS 1.0 was used for more than 10 years, even though alternatives existed. Also insecure ciphers are still very common, having the goal to preserve backwards compatibility. This has not been an issue in DNSSEC so far, since only few deployed algorithms have been deprecated. This may, however, become a challenge in the future. In TLS, large operators (browser vendors in this case) play an even bigger role and have drastically reduced the time for new algorithms to be deployed. A more centralized DNS could have a positive impact on algorithm transition as well, but with the downside of more concentration of power in the hands of a few big cloud-based operators. Finally, as in TLS, the publication of new attacks can influence algorithm transition, but their impact varies and cannot be predicted.

Looking into the future, the threat of quantum computers and Shor's algorithm appear on the horizon [56]. Based on our observations we conclude that quantum-safe algorithms will only be deployed widely, if (i) they are well supported in software and the registration channel *and* (ii) operators actually feel the need to move (e.g. because current algorithms are broken or through financial incentives). In any case, our results show that we need to starting thinking about the transition to algorithms that can withstand the threat of quantum computers now, as the introduction of new algorithms in DNSSEC takes years.

ACKNOWLEDGMENTS

The authors would like to thank the following organizations (in alphabetical order): the CENTR members who participated in our ccTLD survey and RIPE. Furthermore, we would like to thank Chris Faber, our shepherd Mark Allman, Rohan Durrant, Tony Finch, Ulrich Wisser and the anonymous IMC reviewers.

This work was partly funded by CONCORDIA, the Cybersecurity Competence Network supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 830927, and NSF grants CNS-1901090, CNS-1850465.

REFERENCES

[1] Anonymous. [n.d.]. Anonymized for double-blind peer review.
[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. 2005. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard). , 21 pages. https://doi.org/10.17487/RFC4033 Updated by RFCs 6014, 6840.
[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. 2005. Protocol Modifications for the DNS Security Extensions. RFC 4035 (Proposed Standard). , 53 pages. https://doi.org/10.17487/RFC4035 Updated by RFCs 4470, 6014, 6840, 8198.
[4] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. 2005. Resource Records for the DNS Security Extensions. RFC 4034 (Proposed Standard). , 29 pages. https://doi.org/10.17487/RFC4034 Updated by RFCs 4470, 6014, 6840, 6944.
[5] Various Authors. [n.d.]. IETF DNSEXT Mailing List Archive, June 2006. https://mailarchive.ietf.org/arch/browse/namedroppers/?gbt=1&index=7M2k1FW261sYIoDIqinA8WOrOZs.
[6] Various Authors. [n.d.]. IETF DNSOP Mailing List Archive, June 2020. https://mailarchive.ietf.org/arch/msg/dnsop/EYv_-LfkyiQmdguYpbJ3LLX-4Ls/.
[7] S. Bradner. 1996. The Internet Standards Process – Revision 3. RFC 2026 (Best Current Practice). , 36 pages. https://doi.org/10.17487/RFC2026 Updated by RFCs 3667, 3668, 3932, 3978, 3979, 5378, 5657, 5742, 6410, 7100, 7127, 7475, 8179.
[8] Brian Murray. 2020. Ubuntu Wiki: Stable Release Updates. https://wiki.ubuntu.com/StableReleaseUpdates.
[9] Lily Chen, Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. 2016. *Report on Post-Quantum Cryptography*. US Department of Commerce, National Institute of Standards and Technology.
[10] Taejoong Chung, Roland van Rijswijk-Deij, David Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. 2017. Understanding the Role of Registrars in DNSSEC Deployment. In *Proceedings of the Internet Measurement Conference (IMC), 2017* (London, United Kingdom) *(IMC '17)*. Association for Computing Machinery, New York, NY, USA, 369–383. https://doi.org/10.1145/3131365.3131373
[11] Council of European National Top-Level Domain Registries (CENTR). 2020. CENTR Homepage. https://centr.org.
[12] Frank Dennis. 2020. libsodium Releases. https://github.com/jedisct1/libsodium/releases?after=0.4.
[13] DNS OARC. 2020. DNSViz | A DNS visualization tool. https://dnsviz.net/.
[14] DNSThought website. 2018. Atlas measurements used with DNSThought. https://dnsthought.nlnetlabs.nl/raw/.
[15] V. Dolmatov (Ed.), A. Chuprina, and I. Ustinov. 2010. Use of GOST Signature Algorithms in DNSKEY and RRSIG Resource Records for DNSSEC. RFC 5933 (Proposed Standard). , 9 pages. https://doi.org/10.17487/RFC5933 Updated by RFC 6944.
[16] D. Eastlake 3rd. 2001. RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS). RFC 3110 (Proposed Standard). , 7 pages. https://doi.org/10.17487/RFC3110 Updated by RFC 6944.
[17] Tony Finch. 2020. Archive of the DNS root zone. https://github.com/fanf2/saveroot.
[18] Internet Engineering Task Force. [n.d.]. History of draft-hoffman-dnssec-dsa-sha2. https://datatracker.ietf.org/doc/draft-hoffman-dnssec-dsa-sha2/history/.
[19] Dani Grant. 2015. Announcing Universal DNSSEC: Secure DNS for Every Domain. https://blog.cloudflare.com/introducing-universal-dnssec/.
[20] O. Gudmundsson. 2003. Delegation Signer (DS) Resource Record (RR). RFC 3658 (Proposed Standard). , 19 pages. https://doi.org/10.17487/RFC3658 Obsoleted by RFCs 4033, 4034, 4035, updated by RFC 3755.
[21] O. Gudmundsson and P. Wouters. 2017. Managing DS Records from the Parent via CDS/CDNSKEY. RFC 8078 (Proposed Standard). , 10 pages. https://doi.org/10.17487/RFC8078
[22] Mike Hamubrg. 2020. libdecaf Changelog. https://github.com/krionbsd/libdecaf/blob/master/ChangeLog.
[23] W. Hardaker. 2006. Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records (RRs). RFC 4509 (Proposed Standard). , 7 pages. https://doi.org/10.17487/RFC4509
[24] P. Hoffman and W.C.A. Wijngaards. 2012. Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC. RFC 6605 (Proposed Standard). , 8 pages. https://doi.org/10.17487/RFC6605
[25] Hoffman, Paul. 2020. It's Time to Move Away From Using SHA-1 in the DNS. https://www.icann.org/news/blog/it-s-time-to-move-away-from-using-sha-1-in-the-dns.
[26] Holland, Nick and Mestdagh, Steven and Knight, Joel and Jackson, Eric and Vandeputtem Wim and Cappuccio, Chris. 2020. OpenBSD FAQ - Package Management . https://www.openbsd.org/faq/faq15.html.
[27] R. Housley. 2015. Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms. RFC 7696 (Best Current Practice). , 19 pages. https://doi.org/10.17487/RFC7696
[28] Geoff Houston. 2014. ECDSA and DNSSEC. https://blog.apnic.net/2014/10/23/ecdsa-and-dnssec/.
[29] Geoff Houston. 2015. Happy Eyeballs for the DNS. OARC 2015 Fall Workshop, https://indico.dns-oarc.net/event/24/contributions/377/attachments/341/600/2015-10-04-dns-dual-stack.pdf.
[30] Internet Assigned Numbers Authority (IANA). [n.d.]. Domain Name System Security (DNSSEC) Algorithm Numbers. https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml.
[31] ICANN. 2020. Delegated Strings. https://newgtlds.icann.org/en/program-status/delegated-strings.
[32] Internet Engineering Task Force (IETF). [n.d.]. IESG states. https://datatracker.ietf.org/help/state/draft/iesg.
[33] Internet Corporation for Assigned Names and Numbers (ICANN). 2012. Transition to Delegation. In *gTLD Applicant Guidebook*.
[34] Internet Corporation for Assigned Names and Numbers (ICANN). 2013. User Documentation on Delegating and Redelegating a Generic Top-Level Domain (gTLD). https://www.icann.org/en/system/files/files/gtld-drd-ui-10sep13-en.pdf.
[35] Internet Engineering Task Force (IETF). 2020. Mail Archive. https://mailarchive.ietf.org/arch/.
[36] Internet Systems Consortium (ISC). 2019. Release Notes for BIND Version 9.14.0. https://downloads.isc.org/isc/bind9/9.14.0/RELEASE-NOTES-bind-9.14.0.txt.
[37] J. Jansen. 2009. Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC. RFC 5702 (Proposed Standard). , 10 pages. https://doi.org/10.17487/RFC5702 Updated by RFC 6944.
[38] Pierre Karpman, Thomas Peyrin, and Marc Stevens. 2015. Practical Free-Start Collision Attacks on 76-step SHA-1. In *Annual Cryptology Conference*. Springer, 623–642.
[39] Robert Kisteleki. 2016. Ethics of RIPE Atlas Measurements. https://labs.ripe.net/Members/kistel/ethics-of-ripe-atlas-measurements.
[40] O. Kolkman, W. Mekking, and R. Gieben. 2012. DNSSEC Operational Practices, Version 2. RFC 6781 (Informational). , 71 pages. https://doi.org/10.17487/RFC6781
[41] Platon Kotzias, Abbas Razaghpanah, Johanna Amann, Kenneth G Paterson, Narseo Vallina-Rodriguez, and Juan Caballero. 2018. Coming of Age: A Longitudinal Study of TLS Deployment. In *Proceedings of the Internet Measurement Conference (IMC), 2018*. 415–428.
[42] W. Kumari, O. Gudmundsson, and G. Barwood. 2014. Automating DNSSEC Delegation Trust Maintenance. RFC 7344 (Proposed Standard). , 18 pages. https://doi.org/10.17487/RFC7344 Updated by RFC 8078.
[43] B. Laurie, G. Sisson, R. Arends, and D. Blacka. 2008. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. RFC 5155 (Proposed Standard). , 52 pages. https://doi.org/10.17487/RFC5155 Updated by RFCs 6840, 6944.
[44] Tho Le, Roland van Rijswijk-Deij, Luca Allodi, and Nicola Zannone. 2018. Economic Incentives on DNSSEC Deployment: Time to Move from Quantity to Quality. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–9.
[45] Gaëtan Leurent and Thomas Peyrin. 2019. From Collisions to Chosen-Prefix Collisions Application to Full SHA-1. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 527–555.
[46] Florian Mendel, Norbert Pramstaller, and Christian Rechberger. 2008. A (Second) Preimage Attack on the GOST Hash Function. In *Fast Software Encryption*, Kaisa Nyberg (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 224–234.

[47] Florian Mendel, Norbert Pramstaller, Christian Rechberger, Marcin Kontak, and Janusz Szmidt. 2008. Cryptanalysis of the GOST Hash Function. In *Advances in Cryptology – CRYPTO 2008*, David Wagner (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 162–178.

[48] Moritz Müller, Matthew Thomas, Duane Wessels, Wes Hardaker, Taejoong Chung, Willem Toorop, and Roland van Rijswijk-Deij. 2019. Roll, Roll, Roll your Root: A Comprehensive Analysis of the First Ever DNSSEC Root KSK Rollover. In *Proceedings of the Internet Measurement Conference (IMC), 2019*. 1–14.

[49] M. Müller, T. Chung, A. Mislove, and R. van Rijswijk-Deij. 2019. Rolling With Confidence: Managing the Complexity of DNSSEC Operations. *IEEE Transactions on Network and Service Management* 16, 3 (2019), 1199–1211.

[50] NLnet Labs. 2020. Unbound Resolver Change Log. https://nlnetlabs.nl/svn/unbound/tags/release-1.8.0/doc/Changelog.

[51] National Institute of Standards and Technology (NIST). [n.d.]. Secure Hashing. https://web.archive.org/web/20110625054822/http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html.

[52] PowerDNS. 2020. Changelogs for 4.0.x — PowerDNS Authoritative Server documentation. https://doc.powerdns.com/authoritative/changelog/4.0.html.

[53] PowerDNS. 2020. Changelogs for 4.2.x — PowerDNS Authoritative Server documentation. https://doc.powerdns.com/authoritative/changelog/4.2.html.

[54] RIPE NCC Staff. 2015. RIPE Atlas: A Global Internet Measurement Network. *Internet Protocol Journal (IPJ)* 18, 3 (Sep 2015).

[55] S. Rose. 2013. Applicability Statement: DNS Security (DNSSEC) DNSKEY Algorithm Implementation Status. RFC 6944 (Proposed Standard). , 7 pages. https://doi.org/10.17487/RFC6944

[56] Peter W Shor. 1994. Polynomial Time Algorithms for Discrete Logarithms and Factoring on a Quantum Computer. In *International Algorithmic Number Theory Symposium*. Springer, 289–289.

[57] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. 2017. The first collision for full SHA-1. In *Annual International Cryptology Conference*. Springer, 570–596.

[58] O. Sury and R. Edmonds. 2017. Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC. RFC 8080 (Proposed Standard). , 7 pages. https://doi.org/10.17487/RFC8080

[59] Roland van Rijswijk-Deij, Taejoong Chung, David Choffnes, Alan Mislove, and Willem Toorop. 2017. The Root Canary: Monitoring and Measuring the DNSSEC Root Key Rollover. In *Proceedings of the 2017 SIGCOMM Posters and Demos, Part of ACM SIGCOMM 2017*. ACM Press, Los Angeles, CA, USA.

[60] Roland van Rijswijk-Deij, Mattijs Jonker, and Anna Sperotto. 2016. On the adoption of the elliptic curve digital signature algorithm (ECDSA) in DNSSEC. In *2016 12th International Conference on Network and Service Management (CNSM)*. IEEE, 258–262.

[61] Roland van Rijswijk-Deij, Mattijs Jonker, Anna Sperotto, and Aiko Pras. 2016. A High-Performance, Scalable Infrastructure for Large-Scale Active DNS Measurements. *IEEE Journal on Selected Areas in Communications* 34, 6 (2016), 1877–1888.

[62] Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. 2015. Making the case for elliptic curves in DNSSEC. *ACM SIGCOMM computer communication review* 45, 5 (2015), 13–19.

[63] S. Weiler. 2004. Legacy Resolver Compatibility for Delegation Signer (DS). RFC 3755 (Proposed Standard). , 9 pages. https://doi.org/10.17487/RFC3755 Obsoleted by RFCs 4033, 4034, 4035, updated by RFCs 3757, 3845.

[64] S. Weiler (Ed.) and D. Blacka (Ed.). 2013. Clarifications and Implementation Notes for DNS Security (DNSSEC). RFC 6840 (Proposed Standard). , 21 pages. https://doi.org/10.17487/RFC6840

[65] Willem Toorop. 2017. All RSAMD5 signed zones are BOGUS. https://issuetracker.google.com/issues/62692406.

[66] Wisser, Ulrich. 2020. Internetstiftelsen: .se incentives. private correspondence.

[67] Paul Wouters and Ondřej Surý. 2019. Algorithm Implementation Requirements and Usage Guidance for DNSSEC. RFC 8624. https://doi.org/10.17487/RFC8624

[68] Dan York, Ondřej Surý, Paul Wouters, and Ólafur Guðmundsson. 2018. *Observations on Deploying New DNSSEC Cryptographic Algorithms*. Internet-Draft draft-york-dnsop-deploying-dnssec-crypto-algs-06. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-york-dnsop-deploying-dnssec-crypto-algs-06 Work in Progress.