Making DNSSEC Future Proof

Moritz Müller

DNSSEC voorbereiden op de toekomst

Moritz Müller

Graduation Committee

Chair / secretary:	Prof.dr. J.N. Kok
Supervisor:	Prof. dr. ir. A. Pras
Supervisor:	Prof. dr. ir. R.M. van Rijswijk-Deij
Co-supervisor:	Dr. C.E.W. Hesselman

Members:

Prof. dr. ir.	G.J. Heijenk	University of Twente, The Netherlands
Prof. dr.	J.L. van den Berg	University of Twente, The Netherlands
Prof. dr.	O. Hohlfeld	Brandenburg University of Technology, Germany
Prof. dr.	G. Smaragdakis	Delft University of Technology, The Netherlands
Dr.	S. Dickinson	Sinodun Internet Technologies Ltd., United Kingdom

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

DSI Ph.D. Thesis Series No. 18-170
Digital Society Institute
P.O. Box 217
7500 AE Enschede, The Netherlands

Stichting Internet Domeinregistratie Nederland
P.O. Box 5022
6802 EA Arnhem, The Netherlands

ISBN: 978-90-365-5181-6
ISSN: 2589-7721
DOI: 10.3990/1.9789036551816

Typeset with LaTeX. Printed by Gildeprint, The Netherlands.
Cover design by Doe Rustig ontwerp.

MAKING DNSSEC FUTURE PROOF

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof.dr.ir. A. Veldkamp,
on account of the decision of the graduation committee,
to be publicly defended
on Friday 24 September 2021 at 12:45

by

Moritz Christian Müller

born on January 2, 1989
in Stuttgart, Germany.

This dissertation has been approved by:

Supervisors:
Prof. dr. ir. A. Pras
Prof. dr. ir. R.M. van Rijswijk-Deij

Co-supervisor:
Dr. C.E.W. Hesselman

Acknowledgments

This thesis would not have been the same without the help and support of many. The next two pages are an attempt to do those important people some justice.

First, I would like to thank my employer SIDN, and especially my manager Cristian, who gave me the opportunity and the confidence to pursue my PhD in part-time. I highly appreciate this rare possibility to combine large parts of my daily work with my PhD research at the university. Also during my PhD, Cristian provided valuable support and feedback as my co-supervisor. At SIDN, I had the opportunity to work together with an extremely talented and fun team. My colleagues from SIDN Labs not only dragged me into the world of academic papers and conferences or helped with my research, but also made every moment working at SIDN worthwhile, with inspiring discussions, lunchtime walks (that even resulted in walking the Nijmeegse Vierdaagse), and foosball.

Then, I thank the University of Twente, my alma mater where I graduated in 2015, but returned already two years later to start my PhD adventures. Here, I especially would like to thank my supervisor Aiko who welcomed me with open arms at the Design and Analysis of Communication Systems (DACS) group. Thank you for giving me the freedom to choose my own research directions and challenging me to think beyond my own scientific discipline. Also, I thank my supervisor and mentor Roland. Roland guided me through the last years, helped me through times of confusion and lack of orientation, pushed me, trusted my decisions, and removed superfluous commas from my texts rigorously. Aiko, Roland, and the current and former members of DACS made my weekly trips to the east of the Netherlands always inspiring and a nice change of the daily routine. From the members of DACS, I especially would like to thank Wouter, who is also one of my paranymphs, with whom I not only collaborated on research but who also made trips to conferences more fun.

Besides my colleagues at SIDN and the University of Twente, I also had the chance to collaborate with smart and inspiring people from other organizations. Those people were co-authors of my papers, gave valuable feedback on my research, or provided data. Organizations include NLnet Labs, Virginia Tech, Internetstiftelsen, RIPE NCC, and many more. A special thanks also to the members of my committee, for taking the time to read this thesis and for providing valuable feedback.

Then, I would like to thank the people that were not directly involved in this thesis, but without them it would not have been the same or would have never existed in the first place. First of all, I would like to thank my dad. Together we

made it through some tough times, and without your trust and support I would not be where I am today. You always had faith in my decisions and helped me to find my own way. Apparently, I did not need to know Latin after all. Also, many thanks to Sascha for being a great friend, even when living more than 9.000 km apart, and to Franziska and the rest of my family in Germany for your support and thoughts, despite me living in *Holland* for more than 6 years. In those 6 years I not only made new friends, with a special shout-out to my second paranymph and climbing viking Anders, but also got invited to the Brus-family who helped me feel at home in the Low Countries.

And this brings me to Aafke. Sharing basically one room for more than one and half years while pursuing a PhD would have been a challenge for some relationships. Ours, however, just grew stronger and I am thankful for every single day I can spend with you.

Abstract

The Internet has become an essential part of our daily lives. This became even more clear during the COVID-19 pandemic when suddenly our interaction with others moved almost entirely online. This also meant, that it became more important than ever that the Internet worked reliably.

The Internet consists of a number of core components. One of these core components is the Domain Name System (DNS). The DNS is responsible for translating names like `www.rivm.nl`, the domain name of the Dutch National Institute for Public Health, to computer readable IP addresses, like `131.224.245.84`. Virtually every time users want to visit `www.rivm.nl`, or any other domain name, their computer looks up an IP address in the DNS.

The DNS protocol was not developed with security in mind. An attacker can manipulate information in the DNS such that a domain name would direct users to a malicious website, instead of the website they wanted to visit in the first place. There, the attacker could, for example, try to infect the users' computers.

The DNS Security Extensions (DNSSEC) address this vulnerability at its core. With DNSSEC, owners of domain names can digitally sign the information attached to their domain names. Everyone receiving this information can validate that it is correct with the help of the signature and a public key. In the example above, we can be sure that `131.224.245.84` is actually the IP address of `www.rivm.nl`, thanks to DNSSEC.

DNSSEC relies on public key cryptography algorithms. Using insecure algorithms could allow attackers to forge signatures and thus manipulate information in the DNS unnoticed. This means that we could not trust any DNS message anymore.

Unfortunately, every algorithm, currently used in DNSSEC, can be broken by a technological development which has gained more traction in the last years: quantum computers. These computers have the potential to calculate some mathematical problems faster than the computers we use today. Two of those mathematical problems lay the foundation of every cryptographic algorithm used in DNSSEC. These cryptographic algorithms are effectively broken as soon as a powerful enough quantum computer exists, thereby rendering DNSSEC useless. Luckily, the cryptographic community is currently working on cryptographic algorithms that can neither be broken by current computers nor by quantum computers, so called *quantum-safe* algorithms.

In this thesis, we take the first steps to prepare DNSSEC for the threat posed by quantum computers. Only when DNSSEC is able to transition to quantum-safe algorithms, we can trust the DNS, and consequentially the Internet, in the future.

More concretely, we study which problems the DNS faces when introducing new algorithms in general and quantum-safe algorithms in particular. Then, we propose and evaluate solutions that makes it easier to replace algorithms in DNSSEC. Even though quantum computers might still be decades away, we show in this thesis that we need to start understanding and preparing the transition to quantum-safe algorithms now.

In our *first contribution* we take a look back studying the replacement of algorithms in DNSSEC in the past. This allows us to identify major problems that slow down the wider deployment of more secure algorithms. Here, one problem stands out. Domain name operators shun away from the seemingly complex process of replacing one algorithm with another. This process is called an algorithm rollover and if it is not carried out correctly, domain names can become unreachable. We can only transition to quantum-safe algorithms if operators feel comfortable with the different stages of algorithm rollovers.

As our *second contribution*, we reduce the complexity of algorithm rollovers, by giving operators more insight into the process and thus addressing one of the main reasons for not rolling an algorithm. Before we can reduce the complexity, we study which aspects of the DNS make algorithm rollovers complex. We find that uncertainties when timing the algorithm rollovers and the lack of insight into the impact on users is holding operators back. In order to address this, we develop a measurement method to monitor all stages of an algorithm rollover from the perspective of their clients and end users. Then, we evaluate our method during the first algorithm rollover of the Swedish Country Code Top Level Domain (ccTLD) `.se`. The operators of `.se` confirmed that the insights, gained through our measurements, gave them more confidence throughout the rollover. The operators of the ccTLDs of Brazil (`.br`) and Denmark (`.dk`) applied our method as well.

Then, we shift our attention to a special component of the DNS. The DNS is a tree-like naming system, starting at the *root*, with Top-Level-Domains (TLDs) (e.g. `.com` and `.nl`) as branches of the root and *second level domains* (e.g. `rivm.nl` or `sidnlabs.nl`) as the branches of TLDs. The *root* acts as a starting point for lookups in the DNS. This special role also means that the security of each domain name in the DNS relies on the security of the root. So it is crucial that the root can transition to quantum-safe algorithms as well.

As our *third contribution*, we show that the root is, in principle, ready to roll to more secure algorithms as well. We demonstrate this by measuring a *key* rollover —a prerequisite for an algorithm rollover. Replacing the key at the root, and thus also the algorithm, is in particular complex and was carried out only once until now. During such a rollover, millions of recursive resolvers could potentially fail to retrieve information from the DNS if they would not be able to update their local copy of the new key. As a consequence, large parts of the Internet would become unreachable. In this thesis, we show that the key was replaced successfully at the root and at most resolvers. At the same time, we found several issues during the key rollover that could complicate future algorithm rollovers as well and discuss possible solutions. We published parts of our measurement results live during the

key-replacement process, which were followed by the DNS community and main stakeholders of the root closely. After the key rollover, we published our findings at a scientific conference for which we received the distinguished paper award.

In our *fourth and last contribution* we look ahead, with the threat of quantum-computers appearing on the horizon. We show that quantum-safe algorithms are suitable for DNSSEC as well, but with significant caveats. These algorithms, which are currently being assessed, have larger keys, larger signatures, or both. In this thesis, we demonstrate that these attributes can be a challenge for DNS and DNSSEC. For example, the DNS cannot accommodate the cryptographic keys of some of the quantum-safe algorithms. We propose measures to address this and other problems. Also, we share our findings with the DNS and operator community, raising awareness of this issue as one of the first.

Through these contributions, we make DNSSEC more future proof. Thereby, DNSSEC is more prepared to protect the information in the DNS, and sub-sequentially the users on the Internet, against the threats to come.

Samenvatting

Het internet is een essentieel onderdeel van ons dagelijks leven. Dit werd nog duidelijker tijdens de coronapandemie, toen we opeens alleen nog online contact konden houden met vrienden, familie en collega's. Dit betekent ook dat het nog belangrijker is geworden dat het internet altijd betrouwbaar en goed functioneert.

Het internet bestaat uit een aantal belangrijke onderdelen. Een van deze onderdelen is het Domain Name System (DNS). Het DNS is verantwoordelijk voor het vertalen van namen zoals `www.rivm.nl`, de domeinnaam van het Nederlandse Rijksinstituut voor Volksgezondheid en Milieu, naar een voor een computer leesbaar IP-adres, zoals `131.224.245.84`. In principe wordt elke keer dat iemand de website van het RIVM bezoekt een IP-adres in het DNS opgevraagd.

Toen het DNS protocol ontwikkeld werd speelde veiligheid nog geen grote rol. Dit betekent dat een aanvaller informatie in het DNS kan manipuleren. Een eindgebruiker kan daardoor bijvoorbeeld omgeleid worden naar een malafide website, waar dan de computer met een virus geïnfecteerd kan worden.

De DNS Security Extensions (DNSSEC) adresseren deze kwetsbaarheid. Met DNSSEC kunnen eigenaren van domeinnamen hun domeinnamen digitaal ondertekenen. Iedereen die vervolgens het IP-adres van de domeinnaam in het DNS opvraagt, kan met hulp van de handtekening en een publieke sleutel controleren of deze correct is. In het voorbeeld van `www.rivm.nl` kunnen we dankzij DNSSEC zeker zijn dat het bijbehorende IP-adres ook daadwerkelijk `131.224.245.84` is.

DNSSEC maakt hiervoor gebruik van asymmetrische encryptiealgoritmen. Een kwetsbaarheid in een algoritme zou als gevolg hebben dat aanvallers handtekeningen zouden kunnen vervalsen. Hierdoor kan informatie in het DNS toch onopgemerkt gemanipuleerd worden.

Helaas is er een ontwikkeling die mogelijk alle algoritmen die we in DNSSEC gebruiken kwetsbaar kan maken: kwantumcomputers. Een toekomstige kwantumcomputer kan sommige wiskundige problemen veel sneller oplossen dan computers die we op dit moment gebruiken. Twee van deze problemen zijn de basis voor de encryptiealgoritmen in DNSSEC. Zodra er een kwantumcomputer bestaat die sterk genoeg is om de onderliggende wiskundige problemen op te lossen, worden deze algoritmen kwetsbaar, en daarmee dus ook DNSSEC. Als oplossing worden op dit moment algoritmen ontwikkeld die ook veilig zijn voor kwantumcomputers. Deze algoritmen worden ook *kwantumveilige* algoritmen genoemd.

In dit proefschrift nemen we de eerste stappen om DNSSEC voor te bereiden op het gevaar van kwantumcomputers. Alleen als DNSSEC van kwantumveilige algoritmen gebruik kan maken kunnen we de informatie in het DNS, en daarmee

het internet, vertrouwen. We onderzoeken welke problemen er zijn die het introduceren van nieuwe algoritmen in het algemeen, en kwantumveilige algoritmen in het bijzonder, kunnen tegenhouden. Op basis hiervan ontwikkelen en evalueren we oplossingen die het makkelijker maken om de huidige algoritmen door kwantumveilige algoritmen te vervangen. Het zou nog tientallen jaren kunnen duren voordat er kwantumcomputers gemaakt worden die sterk genoeg zijn om de tot nu toe gebruikte algoritmen te breken. We laten zien dat het desondanks nu al nodig is de om de transitie naar kwantumveilige algoritmen voor te bereiden.

In onze *eerste bijdrage* van dit proefschrift blikken we daarom eerst terug en onderzoeken we hoe onveilige algoritmen in DNSSEC in het verleden zijn vervangen. Hierbij identificeren we problemen die het toepassen van veiligere algoritmen, op grotere schaal, hebben tegengehouden. Één probleem valt vooral op: beheerders van domeinnamen zijn terughoudend om een algoritme te vervangen vanwege de hiermee geassocieerde risico's. Het vervangen van een algoritme wordt algoritmerollover genoemd en een fout tijdens zo'n rollover kan als gevolg hebben dat een domeinnaam onbereikbaar wordt. Beheerders van domeinnamen moeten voldoende vertrouwen in de procedures hebben voordat ze naar kwantumveilige algoritmen durven over te stappen.

In de *tweede bijdrage* van dit proefschrift geven we domeinnaambeheerders meer inzicht in het proces van een algoritmerollover. Daardoor zullen zij in de toekomst eerder geneigd zijn om over te stappen naar veiligere algoritmen. Hiervoor onderzoeken we eerst waarom algoritmerollovers complex en riskant zijn. We laten zien, dat beheerders vaak onzeker zijn hoe ze algoritmerollovers moeten timen en dat ze niet genoeg zicht hebben op het gevolg van de rollover op hun eindgebruikers. Daarom ontwikkelen we een meetmethode waarmee domeinnaambeheerders elke stap van hun algoritmerollover vanuit het perspectief van de eindgebruikers kunnen controleren. We evalueren onze meetmethode tijdens de eerste algoritmerollover van de Zweedse Country Code Top Level Domain (ccTLD) `.se`. De beheerders van .se hebben bevestigd dat onze meetmethode hun meer vertrouwen heeft gegeven tijdens de rollover. Na `.se` hebben ook de ccTLDs van Brazilië (`.br`) en Denemarken (`.dk`) onze methode toegepast.

Vervolgens verschuiven we onze aandacht naar een bijzonder onderdeel van het DNS. Het DNS is opgebouwd als een boom die begint bij de *root* (wortel). Top-Level-Domains (TLDs) (b.v. `.com` en `.nl`) zijn de takken van de root en *second level domeinnamen* (b.v. `rivm.nl` of `sidnlabs.nl`) zijn takken van de TLDs. De root is het startpunt voor aanvragen in het DNS en speelt ook voor DNSSEC een belangrijke rol. Hierdoor hangt de veiligheid van elke domeinnaam af van de veiligheid van de root en het is daarom van cruciaal belang dat ook de root in toekomst naar kwantumveilige algoritmen kan overstappen.

In de *derde bijdrage* laten we zien dat de root, in principe, ook klaar is voor een algoritmerollover. Hiervoor meten we de eerste sleutelrollover van de root. Een sleutelrollover is een voorwaarde voor een algoritmerollover. Een sleutelrollover van de root is bijzonder complex en is tot nu toe slechts een keer uitgevoerd. Tijdens dit soort rollovers bestaat de kans dat miljoenen recursive resolvers geen in-

formatie meer van het DNS kunnen opvragen als zij niet op tijd hun lokale kopie van de root sleutel vervangen. Hierdoor zouden grote delen van het internet onbereikbaar kunnen worden. In dit proefschrift laten we zien dat de sleutel van de root, grotendeels, succesvol is vervangen. Tevens hebben we meerdere complicaties kunnen identificeren, die bij een toekomstige algoritmerollover tot problemen zouden kunnen leiden. Daarom laten we ook zien hoe deze problemen aangepakt zouden kunnen worden. We hebben onze bevindingen gedeeltelijk tijdens de sleutelrollover gepubliceerd. De DNS-gemeenschap heeft deze metingen live kunnen volgen. Na de rollover hebben we onze resultaten bij een academische conferentie gepubliceerd waarvoor we de "Distinguished paper" prijs hebben ontvangen.

In onze *vierde en laatste bijdrage* van dit proefschrift kijken we vooruit. We laten zien dat we, in principe, kwantumveilige algoritmen in DNSSEC kunnen toepassen, dit echter wel met een aantal kanttekeningen. De kwantumveilige algoritmen die op dit moment worden ontwikkeld hebben vaak grotere sleutels, grotere handtekeningen, of allebei. In dit proefschrift laten we zien dat deze kenmerken een grote uitdaging zijn voor het DNS en DNSSEC. Het is ruimtetechnisch bijvoorbeeld niet mogelijk om de sleutels van sommige algoritmen in een DNS bericht te versturen. We stellen daarom verschillende oplossingen voor die het alsnog mogelijk maken om kwantumveilige algoritmen in DNSSEC toe te passen. We delen onze bevindingen met de DNS- en beheerdersgemeenschap, en vragen, als een van de eersten, aandacht voor deze problemen.

Door deze bijdragen maken we DNSSEC toekomstbestendiger, en zijn we beter voorbereid om het DNS, en daarmee het internet in zijn geheel, tegen toekomstige gevaren te beschermen.

Contents

CHAPTER 1

Introduction

The worldwide COVID-19 pandemic not only caused a sudden increase in the demand for toilet paper [1] but also made many realize the importance of the Internet. For example, many people worked remotely from home for the first time [2], which often was for companies the only possibility to keep their business running. At the same time, the Internet became for many of us the only medium to stay in touch with friends and family [3].

Already before the SARS-CoV-2 outbreak, the Internet was an important tool in our daily lives. The crisis, however made it clear also to people not working in the field of Computer Science, that the Internet has become essential. Our reliance on the Internet during the pandemic even went so far that some feared [4] that restricted Internet access could directly harm people's health and lives.

These examples show that it is crucial that the Internet works reliably more than ever. This not only includes that services on the Internet must be reachable all the time but also that the requested service has not been manipulated. For example, users that would like to visit a website need to be sure that they are directed to the site they were expecting. Especially websites of health authorities became often frequented sources of information during the crisis. For example, the number of requests for the website of the RIVM (the Dutch National Institute for Public Health and the Environment)[1] increased five times during the first COVID-19 wave, as can be seen in Figure 1.1.

After typing the *domain name* of the RIVMs's website `www.rivm.nl` into their browser, users need to be sure that they are actually directed to the correct website. If not, users could, for example, get served with false information about the pandemic, or their computer could get infected with malicious software.

1.1 THE DNS AND ITS SECURITY EXTENSIONS

The security protocol ensuring that users can be certain that they are actually visiting the website they requested is the main subject of this thesis. Before the browser can retrieve the content for `www.rivm.nl` it first needs to *look up* the Internet Protocol (IP) address with which the domain name is associated. At the IP address,

[1] Similar to the Robert Koch Institute (RKI) in Germany or the Centers for Disease Control and Prevention (CDC) in the U.S.

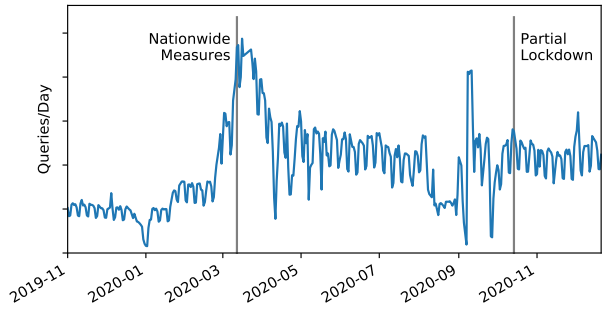


Figure 1.1: DNS queries per day to `rivm.nl`, observed at the operator of `.nl`

`131.224.245.84` in case of `www.rivm.nl`, the browser can fetch the website's content. If this lookup is manipulated in some way, the browser might receive the wrong IP address and could, in turn, fetch a potentially malicious website.

This lookup is made possible by the Domain Name System (DNS). The DNS is the *naming system* of the Internet and translating domain names into IP addresses is its main capacity. Virtually any time a user wants to visit a website or opens an app on her phone, the browser or her phone looks up one or more domain names in the DNS. Moreover, with the rise of the "Internet of Things", more and more physical devices rely on the DNS as well [5]. Only with the help of the DNS, they can connect to their manufacturer to fetch updates, and even in some cases, can be controlled by their user [6].

Unfortunately, the DNS was not designed with security in mind. This became especially clear when Dan Kaminsky [7] found a severe flaw in the design of the DNS. This flaw allows attackers to manipulate certain components in DNS, such that the answers from the DNS could not be trusted anymore. This attack and attacks that followed [8] [9] made it clear that this vulnerability cannot be fixed with small patches here and there but needed a new solution to tackle this problem from the ground up. Also other protocols like Transport Layer Security (TLS), responsible for the padlock in the browser's address bar, cannot protect users from such attacks.

This is when the DNS Security Extensions (DNSSEC) entered the stage. DNSSEC has the goal to enable anyone who requests information from the DNS to verify whether the received information is correct. In case of the example above this means that anyone who looks up the IP address of `www.rivm.nl` can be sure that `131.224.245.84` is correct.

This is achieved with public key cryptography. With DNSSEC, operators can cryptographically sign information they publish at their domain name. Everyone who fetches this information can receive a public key and a signature. With their help, the receiver then can verify whether the received information has not been manipulated and that the information was actually published by the domain name operator. Thereby, DNSSEC adds message *integrity* and *authenticity* to the DNS.

DNSSEC was standardized in its current form already in 2005 [10], but was not

widely deployed. In 2010, adoption of DNSSEC increased, after the most crucial part of the DNS, the *root*, got protected with DNSSEC. This made it easier to deploy DNSSEC at other parts of the DNS as well. Today, more than 50% of all domain names are protected with DNSSEC in some parts of the world and numbers are still rising.[2]

1.2 NEW THREATS TO DNSSEC

Now, ten years later and at the time of writing this thesis, it is time to look ahead. DNSSEC has proven to successfully protect the DNS in the past, but ten years is a long time in computer science. Computers are getting faster and especially one technological development that sounded like science fiction some years ago is getting closer to becoming reality: the quantum computer. Such a computer can calculate some mathematical problems more efficiently than computers that we use today and thereby threaten the security of the public-key cryptographic algorithms used in DNSSEC. This means in the context of the DNS and DNSSEC that an attacker could manipulate a DNS message without the receiver noticing. As a consequence, not a single answer from the DNS could be trusted anymore.

We believe that it is important to prepare for this threat now already. Also in the past, security experts have found vulnerabilities in some of the algorithms used in DNSSEC, which can be exploited without a quantum computer. The replacement of these algorithms is taking years, as we will show in this thesis. At the same time, quantum computers are evolving fast [11], [12]. A quantum computer, powerful enough to break current algorithms, *could* exist in around 15 years, according to the Dutch research organization TNO [13].

1.3 OBJECTIVE, RESEARCH QUESTIONS AND APPROACH

From the observations of the previous section we can see that we need to understand how we can transition to algorithms that can withstand attacks of quantum computers as early as possible. We argue that only if this is achieved, DNSSEC can provide integrity to DNS messages also in the next 20 to 30 years.

1.3.1 Objective

In this thesis we want to contribute to a future-proof DNSSEC and thus define the following objective:

> *to prepare DNSSEC for quantum computing, by identifying problems when introducing new cryptographic algorithms and developing and evaluating solutions to address them.*

To achieve this goal we have identified five research questions. We list them in the section below and describe for each how we address them.

[2] `http://www.secspider.net/`

1.3.2 Research Questions and Approach

The goal of our research can only be achieved if every component of the DNS is able to use and understand cryptographic signing algorithms that cannot be broken by quantum computers. In DNSSEC, the replacement of an old algorithm with a new one is called an *algorithm rollover*. Our overarching question is:

> **Main RQ**—*Is the DNS ecosystem ready for algorithm rollovers?*

To answer this question, we need to answer four sub research questions that look at different aspects of the complexity of algorithm transition.

First, we need to understand if it is actually challenging to introduce new algorithms and thus ask:

> **RQ 1**—*What are the complexities when introducing new algorithms in DNSSEC?*

We address this research question in Chapter 3 and Chapter 4.

Approach to RQ1 We use two complementary studies to approach RQ1. First, we study the *life cycle* of DNSSEC signing algorithms, using passive and active measurements, surveys, and anecdotal evidence at Top Level Domains (TLDs) and second level domain names. For the passive measurements, we relied on data covering 5 TLDs (`.com`, `.net`, `.org`, `.nl` and `.se`) over a period of more than 5 years. For the active measurements, we relied on more than 11,000 Vantage Points (VPs). These measurements give us insight into the barriers of algorithm deployment, during standardization, when being deployed in software, and during their roll-out in operations.

Second, we study the first ever replacement of the root's *main signing key*. The security of the root is crucial for the security of the DNS. An attacker able to manipulate information from the root can spoof any information in the DNS. Thus, the root must be signed with a secure algorithm at any time. A prerequisite for exchanging the signing algorithm in the root zone is the possibility to replace the DNSSEC trust anchor. The trust anchor is a copy of the root public key (also called Key Signing Key (KSK)) which is locally configured by each DNSSEC validating party. We monitor the first, and up to now only replacement of this trust anchor, the so-called *root KSK rollover*. This helps us to understand which barriers we may face when transitioning to another algorithm at the root. Also in this study, we rely on a combination of active and passive measurements, now additionally including data from the root servers themselves and a larger set of VPs (more than 50,000 in total). This allows us to study the rollover from all relevant perspectives.

While answering *RQ1* we find that one of the main barriers is the complex process of replacing one algorithm with another at domain names already secured with DNSSEC. If this *algorithm rollover* is not carried out carefully, a domain name could become unreachable for millions of users. At the same time, however, it is crucial

for the security of a domain name that operators feel comfortable transitioning to another algorithm if necessary. Thus, it is clear that we need to help operators with this predicament, but in order to do so we first need to know:

RQ 2—*What makes algorithm rollovers complex?*

We address this research question in Chapters 4, 5, and 6.

Approach to RQ2 The complexity of algorithm rollovers depends on *where* we carry it out. Due to its unique position in the DNS we find that algorithm rollovers at the root have different challenges than algorithm rollovers at lower levels on the DNS hierarchy. For this reason, we revisit the study in which we measure the first KSK rollover at the root. Then, we shift our attention towards algorithm rollovers at TLDs and second-level domain names. With the help of a literature study and active measurements, mapping the prevalence of certain resolver behavior and testing different failure scenarios in the wild, we show which aspects of the rollover are the most challenging.

One of the challenges has its origin in the behavior of recursive resolvers. In a separate study, we actively and passively measure name server selection behavior of recursive resolvers and discuss its effect on algorithm rollovers. Here, we rely on more than 9,000 active VPs and data from the root servers and from a TLD.

We will show that timing is the most critical aspect of an algorithm rollover. For domain name operators in the process of an algorithm rollover, this means that they need to know when their name servers serve the same keys and signatures. Only then operators can be certain that resolvers get all the necessary information to keep successfully resolving the domain name and can proceed with the algorithm rollover. If operators would proceed too early, their domain name can become unreachable for some resolvers. This leads us to the question:

RQ 3—*What can we do to reduce the complexity of algorithm rollovers?*

The research question *RQ3* is answered in Chapter 4 and Chapter 5.

Approach to RQ3 First, we focus on the root. In Chapter 4, we discuss how we can optimize the distribution of the DNSSEC trust anchor. This can also reduce the chance of validation errors during algorithm rollovers at the root. We base our proposed improvements on the insights we derived from our own measurements of the root KSK rollover. Then, in Chapter 5, we propose a method to monitor algorithm rollovers, especially for TLDs and second-level domain names. Our goal is to give operators more confidence when timing the different stages of the rollover and to be able to detect issues early. To qualitatively evaluate whether our method reduces the complexity we apply our method during the first algorithm rollover of the Swedish Country Code Top Level Domain (ccTLD) .se.

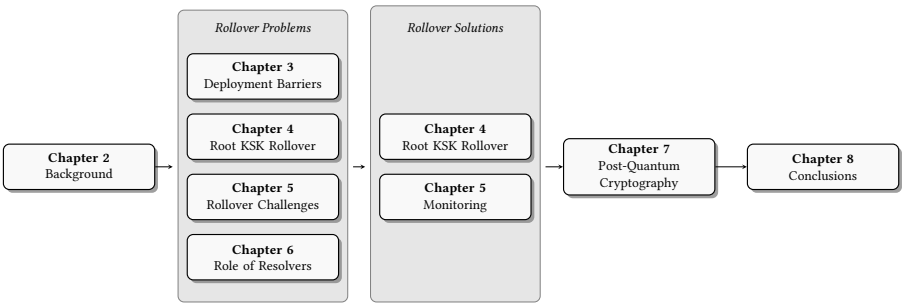


Figure 1.2: Thesis outline

The barriers and solutions discussed up till now can make the transition to new algorithms easier, including new quantum-safe algorithms. These algorithms, however, also have different characteristics than algorithms currently standardized for DNSSEC, stretching the boundaries of current DNS deployments. Thus, before we can transition to these algorithms we need to understand:

RQ 4—*Are there quantum-safe algorithms suitable for DNSSEC?*

We address this research question in Chapter 7.

Approach to RQ4 We examine the properties of quantum-safe algorithms that have a high chance of being standardized in the future. Here, we combine a literature study of the documents describing the algorithms with active performance measurements in a test bed using their optimized implementations. Then, we compare whether their properties differ significantly from those of algorithms currently used in DNSSEC. We rely on existing literature, operational experience, and our own measurements of DNSSEC usage in the wild to identify problems that can hinder the deployment of quantum-safe algorithms in DNSSEC and propose workarounds.

1.4 ORGANIZATION AND MAIN CONTRIBUTIONS

The remainder of this thesis is divided into eight chapters, and Figure 1.2 shows them in a schematic overview. In this section we briefly highlight the main contributions of each chapter.

Contributions Chapter 2: Background

In this chapter we provide background on the DNS and DNSSEC. We mainly focus on the aspects relevant for the research in this thesis. Additionally, to further motivate our research, we provide the reader with a short introduction on quantum computing and the threat quantum computers pose to the security of DNSSEC.

Contributions Chapter 3: Algorithm Deployment Barriers

The goal of this chapter is to identify the main problems we face when deploying new algorithms in DNSSEC. To do so, we analyze the life cycle of a cryptographic signing algorithm used in DNSSEC. This life cycle consists of four stages: *(i)* the standardization of the algorithm in the Internet Engineering Task Force (IETF), *(ii)* the implementation of the algorithm in DNSSEC software and in the domain name registration channel, *(iii)* the deployment of the new algorithm at domain names and the deployment of the updated software, and finally *(iv)* the replacement of old algorithms with new ones. At each stage of this chapter we discuss which problems hindered algorithm deployment in the past, but also how each stage could be accelerated.

The main contributions of this chapter are:

- Providing the first complete analysis of the DNSSEC algorithm life cycle;
- Quantifying the time it took for the algorithm ECDSAP256 to gain wider deployment, taking the whole life cycle into account;
- Identifying at each stage of the life cycle barriers that slow down or hinder algorithm deployment;
- Discussing measures to accelerate each stage of algorithm deployment;
- Showing that an algorithm rollover is the major problem hindering the deployment of new algorithms.

This chapter is based on the following peer-reviewed publication:

- 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 Proceedings of the ACM Internet Measurement Conference (IMC '20). Association for Computing Machinery, New York, NY, USA, 295–308. [14]

Contributions Chapter 4: Key Exchange at the Root

Where Chapter 3 discusses the replacement of an algorithm at TLDs (like `.com` and `.nl`) and at second level domain names (like `example.com`), this chapter analyzes the *root*, studying its first ever KSK rollover. A successful root KSK rollover is a requirement for replacing the algorithm at the top of the DNSSEC hierarchy. While we have shown in the previous chapter that algorithm rollovers at TLDs and second level domain names are a significant problem for introducing a new algorithm, we want to understand in this chapter whether this is also the case for algorithm rollovers at the root. Therefore, we measure the preparation for the first ever root KSK rollover in October 2018, the rollover itself, and the subsequent maintenance from different vantage points in the DNS. The contributions in this chapter are twofold. It

highlights problems that could hinder future algorithm rollovers but also proposes solutions. In particular, the contributions are:

- Validating independently the success of the first ever root KSK rollover;
- Identifying issues when introducing a new root KSK at validating resolvers that can slow down the introduction of a new algorithm;
- Demonstrating that almost all root servers experienced an unexpected and worrisome increase in DNSSEC-related queries after the revocation of the old key;
- Finding software bugs in implementation of DNS resolvers that lead to the worrisome increase of DNSSEC-related queries;
- Proposing central key management by operating systems, thereby improving the distribution of a new key and reducing the barrier for introducing a key with a new algorithm.

Already during the rollover, we shared live updates of our measurement. The DNS community and the operators responsible for carrying out the rollover followed them closely. For the accompanying paper, we received the *Distinguished Paper Award* at the Internet Measurement Conference 2019. The award committee appreciated the timeliness of our paper and acknowledged its good execution. Furthermore, we presented our research at several venues in front of DNS operators and the management team of the root.

This chapter is based on the following peer-reviewed publication:

- 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 '19). Association for Computing Machinery, New York, NY, USA, 1–14. [15]

Contributions Chapter 5: The Complexity of Algorithm Rollovers

This chapter dives deeper into the complexity of algorithm rollovers. Here, we mainly focus on TLDs and second level domain names but some of our observations apply to the root as well. Algorithm rollovers are considered complex and risky because they can render domain names unavailable for resolvers if they are not carried out correctly. The main reason is that validating resolvers, which *cache* information about domain names, can have mismatching information about a domain's keys and signatures, leading to validation failures. Guidelines [16], [17] explain *how* operators should time their rollover, but do not go in depth about *why* timing is important. Also, these guidelines do not take real world deployment into account. In this chapter our contributions are twofold, discussing problems of algorithm rollovers and providing solutions:

- Demonstrating the risk of wrongly timed algorithm rollovers through measurements with more than ten thousand vantage points;
- Identifying non standards conforming resolver behavior, which could cause outages if not taken into account;
- Showing that the number of resolvers that follow a strict interpretation of DNSSEC standards is small. This finding can speed up the algorithm rollover process;
- Developing a monitoring method to give domain operators more insight into the rollover process and thereby more control;
- Publishing software that allows operators to easily monitor a rollover themselves.

The proposed monitoring method was applied during the first ever algorithm rollover of the Swedish ccTLD `.se`. They confirmed that the monitoring gave them more confidence during the rollover, demonstrating that this can reduce the barrier for future algorithm rollovers. The operators of `.br` (Brazil) relied on our monitoring method as well. At each stage of the rollover, they used the gained insights to decide when it was safe to proceed [18]. Also `.dk` (Denmark) monitored their rollover with our method. The chapter is based on the following peer-reviewed publication:

- Moritz Müller, Taejoong Chung, Alan Mislove and Roland van Rijswijk-Deij, "Rolling With Confidence: Managing the Complexity of DNSSEC Operations," in IEEE Transactions on Network and Service Management, vol. 16, no. 3, pp. 1199-1211, Sept. 2019 [19]

Contributions Chapter 6: The Role of Resolvers

Certain behavior of recursive resolvers contributes to the complexity of algorithm rollovers. For example, caching speeds up responses to clients, reduces the query load at authoritative name servers but also slows down algorithm rollovers. Another measure, affecting the timing of algorithm rollovers, is the fact that resolvers distribute their queries across authoritative name servers. This is especially critical if resolvers follow a "strict" interpretation of DNSSEC standards. Only from the moment when all authoritative name servers serve the cryptographic keys and signatures can an operator be sure that these resolvers do not fail to validate signatures. In this chapter, we measure name server selection strategies of resolvers. Our main contributions are:

- Analyzing name server selection strategies of recursive resolvers, for the first time "in the wild";
- Quantifying the share of resolvers that query two or more authoritative name servers, highlighting that it is important that operators take the publication delay into account during rollovers.

Besides the effect on algorithm rollovers, name server selection strategies can also negatively affect lookup performance. In this chapter, we also recommended the deployment of *anycast* at all name servers, with similar coverage of all name servers. Anycast is a technique to replicate a name server in different locations. By following our recommendations, resolvers reach a name servers as close as possible, independent of which they query. This recommendation was followed by the operators of the Dutch ccTLD `.nl`, who upgraded their name server setup accordingly. The findings of this chapter are based on the following peer-reviewed publication:

- Moritz Müller, Giovane C. M. Moura, Ricardo de O. Schmidt, and John Heidemann. 2017. Recursives in the Wild: Engineering Authoritative DNS Servers. In Proceedings of the 2017 Internet Measurement Conference (IMC '17). Association for Computing Machinery, New York, NY, USA, 489–495. [20]

Contributions Chapter 7: Preparing DNSSEC for Quantum-Safe Cryptography

Finally, in this chapter we discuss if we can also deploy algorithms in DNSSEC that can withstand attacks of a quantum computer. A powerful quantum computer could break all algorithms currently standardized for DNSSEC in polynomial time, rendering DNSSEC useless. At the time of writing this thesis, the US National Institute of Standards and Technology (NIST) is assessing algorithms that promise to withstand attacks from current as well as quantum computers. All of these algorithms have in common that they have larger keys, larger signatures, or both, compared to algorithms currently used in DNSSEC. DNS and its underlying transport protocol are sensitive to message size and therefore large signatures or keys could negatively affect transmission. In this chapter, we study if these attributes can hinder the deployment of quantum-safe algorithms in DNSSEC. The main contributions are:

- Assessing the applicability of quantum-safe algorithms in DNSSEC;
- Defining critical requirements that quantum-safe algorithms need to fulfill if they should be used in DNSSEC;
- Selecting candidate quantum-safe algorithms, most suitable for DNSSEC;
- Proposing measures to work around limitations that some of the quantum-safe algorithms face when getting deployed in DNSSEC.

This chapter is based on the following peer-reviewed publication:

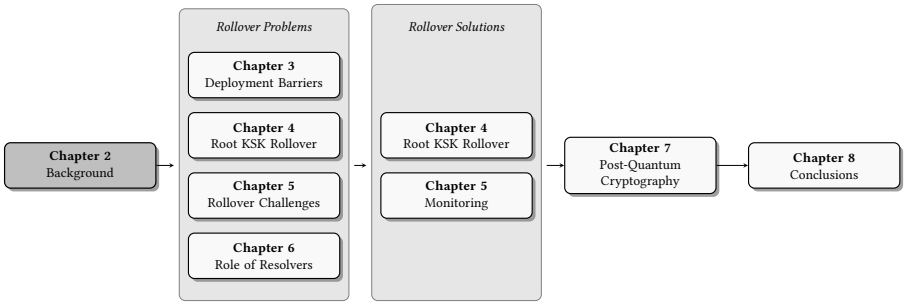
- Moritz Müller, Jins de Jong, Maran van Heesch, Benno Overeinder, and Roland van Rijswijk-Deij. 2020. Retrofitting Post-Quantum Cryptography in Internet Protocols: A Case Study of DNSSEC. SIGCOMM Computer Communication Review 50, 4 (October 2020), 49–57. [21].

Contributions Chapter 8: Conclusions

In the last chapter of this thesis we revisit the research goal and draw overall conclusions. Here, we also discuss potential future research directions.

CHAPTER 2

Background



In this chapter we explain the basic principles of the DNS and DNSSEC. These lay the foundations for the following chapters. First, we describe the components of the DNS relevant to this thesis. Then, we explain the goal of DNSSEC, how it extends the DNS, and how it affects DNS operations. Last, we give an introduction to quantum computing and how quantum computers could threaten the security of the DNS. We describe the background specific to individual studies in the corresponding chapters of this thesis.

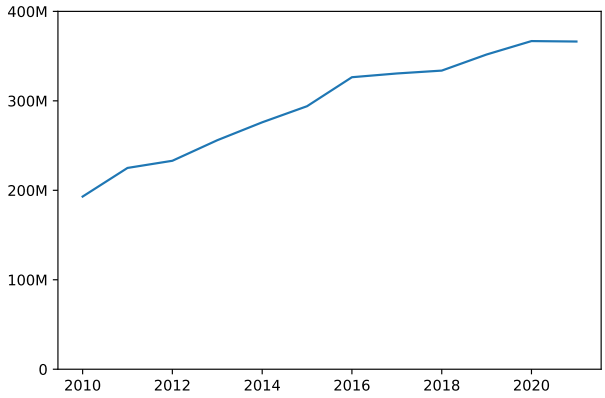


Figure 2.1: Rise of registered domain names from 2010 to 2020 [22]

2.1 THE DOMAIN NAME SYSTEM

First, we start by giving some context in which the DNS was developed initially and the scale in which the DNS is used today.

The Domain Name System was first introduced in 1983 [23]. Back then, it had the goal to simplify the mapping of names and addresses in the Advanced Research Projects Agency Network (ARPANET), the predecessor of the Internet. Before the introduction of the DNS, the mappings between names and addresses was managed and published centrally in the `HOST.TXT` file. Hosts in the ARPANET would then fetch the file on a regular basis. The network grew and the costs for distributing the `HOST.TXT` became too large and the central management of the file too inflexible. The DNS addresses both problems by distributing the management of the name space and its access. Eventually, the ARPANET evolved into the Internet as we know it today and also there the DNS remains the main naming system where it proved to be very scalable. Whereas in 1987, the DNS contained only 20,000 names, it now contains more than 350 million registered domain names (see also Figure 2.1). Almost every connection setup on the Internet is preceded by a lookup of one of the domain names in the DNS, making it an indispensable core component of today's Internet.

2.1.1 Name Space

The name space of the DNS is hierarchical and distributed, as shown in Figure 2.2. This separation also has an influence on how DNSSEC is deployed and managed, as we will show below.

In this figure, `example.com` is at the *2nd level* of DNS name space. The first level is `.com` and at the top of the hierarchy is the root (sometimes represented as a dot '.'). Domains like `.com` or `.org` are called top-level-domains (TLD). If they represent a

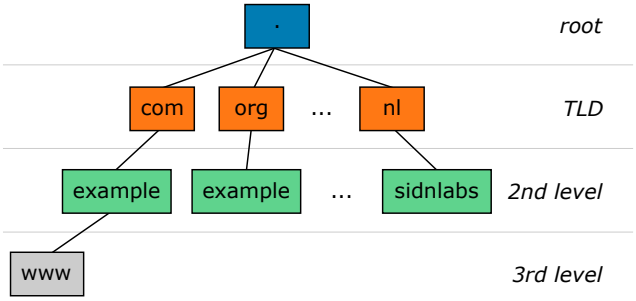


Figure 2.2: The hierarchy of the DNS name space

country, e.g. `.nl` for the Netherlands, they are called country-code TLDs (ccTLDs). Domain names lower on the hierarchy are called by the depth of their level respectively (e.g. 3rd level, in case of `www.example.com` in Figure 2.2). Domain names on different levels on the hierarchy are in a parent-child relationship. A domain name higher in the hierarchy is the *parent* of the domain name one level lower in the hierarchy. This lower domain name is, in turn, the *child*.

Each name on each level can be managed by a different entity. For example, the entity *authoritative* (responsible) for `example.com` is a different entity than the one authoritative for `.com`.

2.1.2 DNS Components

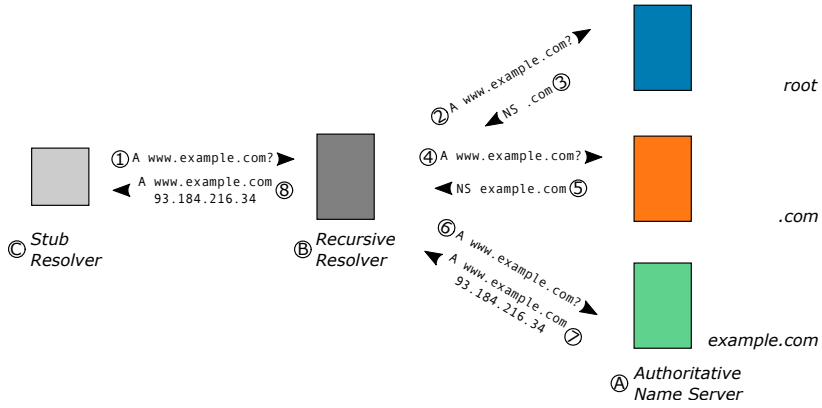


Figure 2.3: DNS components and example lookup of the A Resource Record (RR) for `www.example.com`

The DNS introduces three active components to manage the DNS name space and to retrieve information from it. These components enable the DNS to distribute data and access. Figure 2.3 shows their relationship.

	Type	Description
General	A	IPv4 Address
	AAAA	IPv6 Address
	NS	Domain name of the Authoritative Name Server
	TXT	Can hold text strings, used for different purposes
DNSSEC	DNSKEY	Public Key (ZSK and KSK)
	RRSIG	Digital signature, covering another RR
	DS	Delegation Signer, indicates that a delegated zone is signed

Table 2.1: Relevant Resource Records

- **Authoritative Name Server** Ⓐ Authoritative name servers hold the information in the DNS. Information can include, for example, the mapping of a domain name to an IP address or information about the domain's mail server. A name server can be authoritative for parts of the name space, like `.com`. It can also delegate authority to its children, like `example.com`. Then, the authoritative name server holds information about the child's name server as well. A domain name can have multiple authoritative name servers. This increases resilience and distributes load, but can also complicate zone management, as we will show in Chapter 6.

- **Recursive Resolvers** Ⓑ Recursive resolvers query authoritative resolvers to look up the information requested by their clients. They follow delegations of authoritative name servers *recursively* until the sought information is found or another condition is met that ends the lookup. Recursive resolvers also have a cache to store results temporarily. We explain the details of the lookup below.

- **Stub Resolvers** Ⓒ Stub resolvers run on the computer of the clients, requesting information in the DNS. Stub resolvers are, typically, directly implemented by the operating system, or by an application like a browser. They do not perform a recursive DNS lookup themselves but employ recursive resolvers to do so. Clients can have multiple recursive resolvers at their disposal, e.g. to increase resilience against outages.

2.1.3 Storing, Retrieving, and Transporting Data in the DNS

Now, we explain how information in the DNS is stored, how the information is retrieved, and how it is transported between the different DNS components.

Storing data In the DNS, data is directly attached to a domain name. The data is structured in Resource Records (RR). Data can be, for example, an IP address (of RRs type `A` for an IPv4 and `AAAA` for an IPv6 address), the name of an authoritative

name server (of RR type NS), or strings of text (of RR type TXT). Table 2.1 lists the most relevant RR types for this thesis (we explain RRs related to DNSSEC below). Multiple RRs are combined in a *zone file*, which is then published by the authoritative name server.

Every RR has the same basic structure and consists of 6 fields.

- **NAME** contains the domain name to which the RR is attached, e.g. `www.example.com`.
- **TYPE** defines the record type, e.g. A or NS.
- **CLASS** is a remnant from the early days of the DNS and is almost always set to IN (for the Internet system).
- **TTL** Time-To-Live, defines the maximum time a resolver can cache an RR in seconds before it *should* be discarded – more details in the next section.
- **RDLENGTH** contains the length of the *RDATA* field.
- **RDATA** contains the actually data attached to the domain name, e.g. an IPv4 address in case of an A RR. This field is of variable length, but originally limited to 512 bytes [24].

Retrieving data In order to lookup data attached to a domain name, clients employ resolvers. Typically, this is a *stub resolver*, running directly on the computer or mobile phone. The stub resolver forwards the query to a *recursive resolver* that traverses the name space in order to retrieve the requested data. To illustrate this process, we explain the lookup for the A record of `www.example.com` in Figure 2.3. Here, ① a freshly installed recursive resolver receives a query from the stub resolver of one of its clients. At this point, the resolver does not have any information stored in its cache, but only knows the IP addresses of the root servers. ② As a consequence, the resolver will start the recursive lookup by asking the authoritative name servers of the root for the A record of `www.example.com`. ③ The root, not knowing the answer to this query, will refer the recursive resolvers to its child – the authoritative name servers of `.com`. ④ The recursive resolver selects one of the provided authoritative name servers to send the query. ⑤ The selected authoritative name server will, again, not know the answer but refer the recursive resolver downwards to the authoritative name servers of `example.com`. ⑥ Finally, these name servers are authoritative for the data of `www.example.com` and ⑦ respond to the recursive resolver with the corresponding A record. ⑧ The recursive resolver returns the answer to the client.

Now, the recursive resolver also caches the received information from the authoritative name servers. How long resolvers should cache the data is defined by the Time To Live (TTL) field in the RR. This enables the resolver to respond to future queries for the same data faster and reduces the load at the authoritative name servers.

Root Letter	Operator
A	Verisign, Inc.
B	University of Southern California, Information Sciences Institute
C	Cogent Communications
D	University of Maryland
E	NASA (Ames Research Center)
F	Internet Systems Consortium, Inc.
G	US Department of Defense (NIC)
H	US Army (Research Lab)
I	Netnod
J	Verisign, Inc.
K	Réseaux IP Européens Network Coordination Centre (RIPE NCC)
L	Internet Corporation for Assigned Names and Numbers (ICANN)
M	WIDE Project

Table 2.2: Root servers and their operators [25]

Transporting data By default, DNS messages are transmitted via the User Datagram Protocol (UDP). If the authoritative name server has reason to believe that a DNS message is too large to fit in a single datagram then it signals this to recursive resolvers by setting a special flag (`TC` – Truncation) in the response. This flag instructs the recursive resolver to retry, but using the Transport Control Protocol (TCP) instead.

Originally, the maximum size of a DNS message was limited to 512 bytes [24]. The Extension Mechanisms for DNS (EDNS0) addresses this limitation. With EDNS, resolvers can indicate that they support a larger DNS message size, up to 64 kB. If the response fits within the limit signaled by the resolver, then it does not need to fall back to TCP but will receive the response directly via UDP.

This is, for example, necessary for zones where DNSSEC is deployed. DNSSEC adds additional information to a response, often causing messages to be larger than 512 bytes (see also Section 2.2). EDNS introduces an additional pseudo RR (`OPT`), which can be added by a resolver to a DNS query. The format of the pseudo `OPT` RR is the same as the format of regular RRs but repurposes some of its fields. For example, the `CLASS` field now encodes the supported message size and the `TTL` allows the encoding of new flags.

2.1.4 Ecosystem

One of the main design goals of the DNS was to let operators maintain their own part of the name space [23]. As a consequence, the DNS ecosystem has developed in a way in which it involves multiple stakeholders that are all connected and that need to collaborate. This interaction has an influence on how the DNS is managed and how information in the DNS is accessed. In this section, we introduce the most important roles in the ecosystem of relevance to this thesis.

Root servers The root servers are at the top of the DNS name space. Currently, 13 root servers exist, named with the letters *A* to *M* [25]. The 13 root servers are operated by 12 different organizations, listed in Table 2.2. The root servers are authoritative for the *root zone*. This zone file contains all delegated top level domains and their corresponding authoritative name servers, and is managed by ICANN.[1]

A new installation of a recursive resolver has only the IP addresses of the 13 root servers pre-configured. A recursive resolver sends a query to one of the root servers for every domain name that is not cached and for which it does not know its authoritative name servers.

Top Level Domains and registries Currently, 1,589 TLDs are delegated in the root zone file.[2] We differentiate between two types of TLDs: TLDs for generic purposed (like `.com`, `.net`, `.org`) and ccTLDs established for countries or territories (like `.nl` or `.de`). Each TLD is managed by a *registry operator*. The registry operator (or just *registry*) keeps track of the registrations of 2nd level domain names under the TLD and generates the zone file. In case of `.com` the registry is Verisign and in case of `.nl` the registry is the Stichting Internet Domeinregistratie Nederland (SIDN).

Registrars and DNS providers Everyone can register a 2nd level domain name at a *registrar*. When a new domain name is added, changed or removed, the registrar communicates this change to the registry. Then, the corresponding registry updates the zone file. The registrar also informs the registry of the authoritative name servers of the domain name. The owner of a domain name is referred to as a *registrant*. The relationship between registry, registrar and registrar is sometimes abbreviated to *RRR*.

Recursive resolvers Traditionally, an organization would run a recursive resolver in their own network. Then, clients in the same network can benefit from the shared cache of the recursive resolver. Alternatively, Internet Service Providers (ISPs) provide a recursive resolver for their customers. Even though both setups are still common today, clients now also often rely on public DNS services, provided by third parties on the Internet. These parties, like Google or Cloudflare run large scale resolver networks that promise high performance and high availability [26].[3] Some estimate that around 15% of the Internet's population is relying on such public DNS services [27].

2.2 THE DNS SECURITY EXTENSIONS

By design, the DNS is insecure, since security was not very relevant in the early days of the ARPANET. In practice, this means that neither the content of the messages sent is confidential, nor is the authenticity of the sender or the integrity of

[1] Not directly, but through PTI, an ICANN affiliate `https://pti.icann.org/`.
[2] A full list can be found here `https://www.iana.org/domains/root/db`.
[3] Probably best known by their respective IPv4 addresses `8.8.8.8` and `1.1.1.1`.

the message guaranteed. To many, this became painfully clear when Dan Kaminsky [7] found a severe flaw in the design of the DNS in 2008. This *cache poisoning* attack allows an off-path attacker to inject false DNS responses into the cache of a recursive resolver. As a consequence an attacker could, for example, direct users to a malicious website.

DNSSEC can prevent such an attack but was not widely deployed at the time when the attack was published. After Kaminsky's finding, DNSSEC gained more traction and even though it is still far from being deployed universally, DNSSEC now secures millions of domain names and users.[4]

Additionally, DNSSEC has laid the foundation for other security protocols. For example, DNS-based Authentication of Named Entities (DANE) relies on the security guarantees of DNSSEC to publish digital certificates in the DNS. These certificates are used, among others, to protect the communication between mail servers. Mail servers use the certificates, published in the DNS to set up authenticated and encrypted connections with each other. DNSSEC guarantees that the published certificate is correct.

In this section, we explain in more detail how the DNS is vulnerable to attacks. Then, we describe how DNSSEC can render such attacks futile. To achieve this, DNSSEC extends the DNS with new record types and adds new principles, like the chain of trust. This also affects the way in which zones are managed and requires modifications of recursive resolvers. We explain the main aspects below.

2.2.1 Attacking the DNS

In the original design of the DNS, a recursive resolver cannot verify if a received message actually originated from the queried authoritative name server. Also a resolver cannot verify whether or not a message has been tampered with on its way. Upon receiving a DNS response, the recursive resolver only checks the following parameters:

a) Every DNS query contains a random query ID. Does the response contain the same ID?

b) Does the response arrive at the same port from which the query was sent?

c) Does the response come from the same IP address to which the query was sent?

If every check passes, the resolver caches the responses for the time defined by the TTL.

To demonstrate an attack, we revisit the example lookup shown in Figure 2.3. Imagine now that an attacker wants to spoof the answer to the query for the `A` record of `www.example.com`. As soon as the resolver has issued its query (step ⑥ in

[4]Current deployment at domain names (`http://www.secspider.net/`) and recursive resolvers(`https://stats.labs.apnic.net/dnssec/`).

Figure 2.3), an attacker starts trying to construct a DNS response that also fits the three criteria above. Only with one important difference: Her response contains an `A` record that points to a malicious website.

For a successful attack she has to guess the following parameters of the response: *a)* The query ID is a random number between 0 and 65,535. Since DNS relies on UDP by default, the attacker can send thousands of DNS messages at a low cost in a short time, guessing the correct query ID by brute force. *b)* The port number can take values between 0 and 65,535, but starts in practice at 1,024.[5] The attacker can also guess this number by brute force. Additionally, techniques exist to reduce the search space and to speed up the attack [9]. *c)* Since the IP address of the queried authoritative name server is public, the attacker can pretend to send her response from the same IP address as well. Many networks still allow sending spoofed IP packets[6] which also enable this DNS cache poisoning attack.

For the attack to be successful, an attacker needs to make sure that her fake response arrives at the recursive resolver before the legitimate response (step ⑦ in Figure 2.3). If this is the case and all three parameters are correct, the recursive resolver accepts the response, returns it to the client, and caches the malicious `A` record for the time set by the attacker.

Also here, an attacker can use additional techniques to increase the chance that her fake response arrives before the legitimate response. Using these techniques, Man et al. [9] demonstrated that they could successfully spoof a DNS response in less than 10 minutes.

As a reaction to the Kaminsky attack, stop-gap measures have been proposed to decrease the impact, but these leave the DNS still vulnerable and do not address the vulnerability by its roots.[7] At the same time, new methods have been found to circumvent the proposed measures, increase the success rate of an attack [9], or find other vulnerabilities in resolvers that make cache poisoning possible [8].

2.2.2 Protecting the DNS

DNSSEC addresses the issues that make DNS vulnerable to the described attacks at their core. With DNSSEC, everyone who looks up information in the DNS can validate whether the received information is correct or not. A fake response sent by an attacker would not pass the validation, preventing the attacker from poisoning the cache.

DNSSEC achieves this with the help of public key cryptography. With DNSSEC, domain name operators can cryptographically sign their DNS zone. Then, recursive resolvers, requesting data of the domain name, can validate whether the received response is correct. Invalid messages are discarded.

[5] The first 1,024 port are reserved for specific purposes.

[6] `https://spoofer.caida.org/summary.php`

[7] For example, the port from which the resolver would send a query was usually was not chosen at random before the publication of the Kaminsky attack.

Signing

DNSSEC requires modifications at the side of authoritative name server and at the recursive resolvers. At the authoritative name server, DNSSEC introduces three components: *i*) the private key, which signs the RRs in the zone. *ii*) the signatures, created by the private key, and published with the signed RRs in the zone. *iii*) the public key, used by recursive resolvers to validate the signatures.

The private key is kept secret, but the signature and the public key are published in new RRs in the zone file. The signature is published in an `RRSIG` RR and the public key in a `DNSKEY` RR.

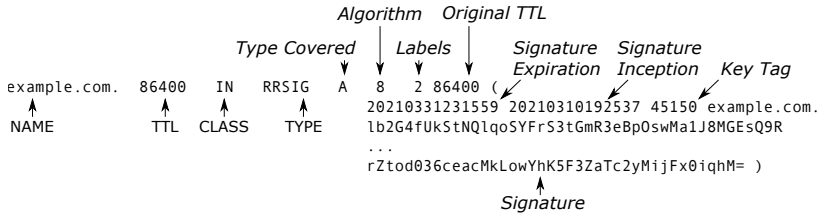


Figure 2.4: Example of a signature covering the `A` record of `example.com`

The signature The signature is returned by the authoritative name server together with the signed RRs. Figure 2.4 shows an example of an `RRSIG` covering the `A` record of `example.com`. The data field in the `RRSIG` RR contains the following information:

- **Type Covered** signals the RR type the signature covers, `A` in case of the example in Figure 2.4.
- **Algorithm** indicates the cryptographic signing algorithm used to create the signature, RSASHA256 in case of the example. RSASHA256 is identified by the number 8 (Table 3.1 in Chapter 3 lists all algorithms standardized for DNSSEC and their identifiers).
- **Labels** the number of labels of the covered domain name, two in the example in Figure 2.4. Necessary for validating wildcard records [28].
- **Original TTL** the TTL of the covered RR. The signature is calculated also over the TTL of the original RR, which might not be known to the recursive resolver. For this reason, the original TTL is also included in the `RRSIG` RR.
- **Signature Expiration and Inception** the period in which the signature should be considered valid.
- **Key Tag** hint to the public key with which the signature was created.
- **Signature** the actual signature.

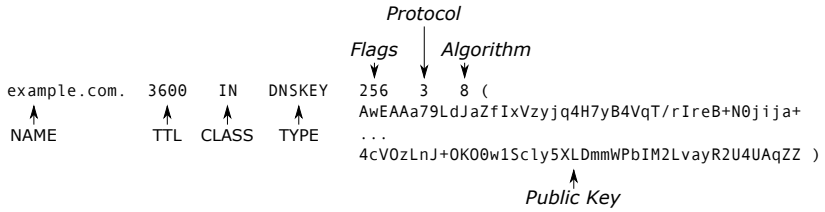


Figure 2.5: Example of a DNSKEY RR of example.com

The public key The public key records are requested by validating resolvers in a separate lookup. Figure 2.5 shows an example of a DNSKEY RR of example.com.

The fields in the DNSKEY RR contain the following information:

- **Flags** indicate the purpose of the key. Bit 7 of the flags field indicates whether the public key can be used to validate signatures and is always set for active keys. Bit 15 indicates whether the key acts as a *secure entry point*. We explain the difference below. In the example in Figure 2.5, the key does not have bit 15 set (ergo *256*).
- **Protocol** currently always set to 3.
- **Algorithm** indicates the cryptographic signing algorithm used to create the key. See again Table 3.1, Chapter 3 for all standardized algorithms.
- **Public Key** holds the public key material.

Often, one public-private key pair is used to sign and validate the records that contain the actual information in the zone (e.g. the A or NS record). This key is called the Zone Signing Key (ZSK) (flag 7 set, ergo 256). Then, another public-private key pair is used to sign every DNSKEY in the zone. This, so called, KSK simplifies some of the operational challenges DNSSEC introduces, which are explained below. The KSK usually also has bit 15 in the flag field set (flag 7 and 15 set, ergo 257).

Validation

A resolver that uses DNSSEC to verify whether a message is correct is called a validating resolver. When such a resolver queries for the A RR of www.example.com, it signals to the authoritative name server that it also wants to receive the accompanying signatures. The resolver does this by setting the *DO* flag in the DNS query. If the record is signed, the authoritative name server responds with the signature along with the requested A record. To validate the signature, the validating resolver then also queries the authoritative name server for the DNSKEY records.

When the DNS message is valid, the recursive resolver returns the response to the client initiating the query. Additionally, it sets the AD (Authenticated Data) flag in the response, indicating that the message has been validated. If the message could

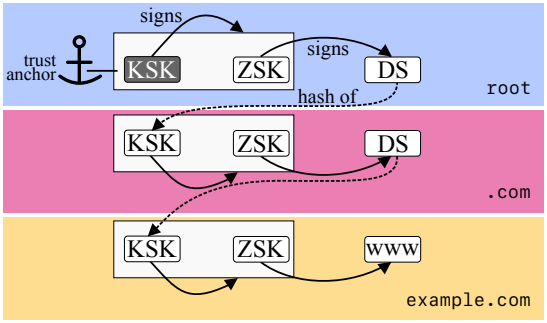


Figure 2.6: DNSSEC Trust Chain

not be validated, the recursive resolver responds with an empty response, only containing the error code `SERVFAIL`. In DNSSEC, we consider an invalid message *bogus*, a message with valid signatures *secure*, and an unsigned message *insecure*.

DNSSEC trust chain

At this point, the validating resolver has the key and signature of `example.com`. With only these records, however, the validating resolver still cannot be sure that the received `A` record is actually correct since an attacker could have spoofed the signature and key as well.

To address this problem, DNSSEC introduces the concept of a *chain of trust*. In such a chain, every level of a domain name, from the root, the TLD `.com`, to `example.com` needs to be signed. Then, the ZSK of the parent authenticates the KSK of its children by linking them together. For this purpose, DNSSEC introduces the Delegation Signer (DS) RR. The `DS` records contains a hash of the KSK of the child. Then, the `DS` is signed by the ZSK of the parent, which again is signed by its KSK. These steps are continued until the root zone. As a consequence, the security of the chain of trust ultimately relies on the KSK of the root zone, which is called the *trust anchor*. Figure 2.6 visualizes this chain with the example of `example.com`.

Now, the recursive resolver only needs to securely receive the *trust anchor*, instead of the KSKs of every signed zone. Typically, recursive resolvers have the trust anchor pre-configured at the time they are installed. Validating resolvers need to update their configuration only when the KSK of the root is replaced. We analyze this process in Chapter 4.

Example of a signed zone

Example 1 brings the introduced new RRs together, showing a schematic overview of a signed zone file. In this example, lines *1*, *3*, and *4* show the RRs that the zone file already contains *before* DNSSEC is deployed. When the zone is signed, the signatures are added as well (lines *2*, *5*, and *8*). Also the public keys are added (lines *6* and *7*).

In this example, two public keys are added. One signs the DNSKEY RRs (the KSK, line *6*). This key also acts as the secure entry point and is marked with the number 257 in the data field (bit 7 and bit 15 set in the flag field). The secure entry point is linked with the parent's zone, as described in Section 2.2.2. The other signs every other record (the ZSK, line *7*) and is marked with number 256 (bit 7 set in the flag field). The introduction of two separate keys can, in some cases, simplify zone management, as we will show later in this section.

	domain name	*TTL*	*type*	*value*
1	example.com.	86400	A	93.184.216.34
2	example.com.	86400	RRSIG	A 8 2 86400 . . .
3	example.com.	86400	NS	a.iana-servers.net.
4	example.com.	86400	NS	b.iana-servers.net.
5	example.com.	86400	RRSIG	NS 8 2 86400 . . .
6	example.com.	3600	DNSKEY	257 3 8 AwEAAZ0a. . .
7	example.com.	3600	DNSKEY	256 3 8 AwEAAa79. . .
8	example.com.	3600	RRSIG	DNSKEY 8 2 3600 . . .

Example 1: Schematic view of a DNSSEC-signed zone snippet for example.com

2.2.3 Defending the DNS Against Cache Poisoning

By allowing operators to sign their zone, and resolvers to validate the signatures, DNSSEC can effectively protect against cache poisoning attacks. To demonstrate this, we revisit the attack described in Section 2.2.1 where an attacker tries to spoof the answer for the A record of www.example.com.

Imagine that the attacker, again, has successfully guessed the query ID and port number, and made sure that her fake response arrived before the legitimate one. Only this time the A record of www.example.com is signed with DNSSEC.

In such a scenario, the validating resolver knows already that example.com is secured with DNSSEC. This was signaled by the DS record, which is now also returned by the .com name servers when delegating the resolver to the name servers of example.com. As a consequence, the validating resolver will discard the unsigned, fake response sent by the attacker. The resolver has successfully parried the cache poisoning attack.

If the attacker tries to inject a record in a resolver's cache she now must also be able to forge the signatures. In turn, that means that the security of the signatures relies heavily on the security of the cryptographic algorithms used to create the keys and signatures. If an attacker would be able use a vulnerability in an algorithm to recreate the private key, she could create legitimate looking signatures. Thus, only if the used algorithms are secure, also the created signatures can be trusted. This is the reason why we need to make sure that we can transition to secure algorithms in DNSSEC efficiently.

2.2.4 DNSSEC Operations

DNSSEC can protect against attacks on the DNS but also adds complexity to operations. Most relevant to this thesis is the fact that operators of a signed zone need to make sure that the chain of trust between their zone and the root zone is intact. If not, resolvers might not be able to successfully validate information in the operator's zone anymore, rendering it effectively offline.

During normal operations the chain between the child and the parent stays untouched. However, if the used algorithm at the child is not considered secure anymore, then the KSK needs to be replaced. As a consequence, also the `DS` record, containing a hash of the old KSK, at the parent needs to be updated. This, so called, algorithm rollover requires careful timing and good communication between the child and the parent. If not, the chain of trust might become incomplete for some validating resolvers, leading to validation errors. Clients of resolvers would then only receive an error message. In case of the example in Figure 2.3, a client would not be able to visit the website of `www.example.com` until the misconfiguration is resolved.

Such a misconfiguration is especially risky when the root replaces its KSK. Since the root's KSK is the trust anchor of the whole trust chain, a misconfiguration affects the whole DNS hierarchy. We explain the details of this operational challenge in more detail in Chapter 4 and Chapter 5.

Other events that require the replacement of the KSK are, for example, in case of a key compromise or when it is required by a key-management policy demanding regular rollovers.

2.2.5 Other DNS Security and Privacy Protocols

Besides DNSSEC, other protocols and extensions exist that have the goal to secure the DNS. We briefly discuss the most relevant protocols in order to give the reader a better picture of the security landscape in which DNSSEC is being deployed. The protocols discussed can be deployed complementary with DNSSEC.

Confidentiality DNS-over-TLS (DoT) [29] and DNS-over-HTTPS (DoH) [30] add confidentiality to DNS messages. In case of DoT, DNS messages are encrypted directly on the transport layer, whereas in case of DoH, DNS messages are wrapped in the Hypertext Transfer Protocol (HTTP) protocol. Both protocols encrypt the messages between a client, sending a DNS query, and a server, sending the response.

Especially DoH gained wider deployment. Among others, Google's Chrome browsers has enabled DoH for some of their users [31]. The browsers of these users now send encrypted DNS queries to the resolver if the resolver supports DoH and is trusted by Google.

Privacy DNS queries can contain sensitive information [32]. In case of DoT and DoH, a recursive resolver or a name server can still see the full content of a query, but this is not always necessary. In the past, queries from recursive resolvers to

the root would contain the full query name (e.g. `www.example.com`), even though a query containing only the TLD would have been sufficient for the root servers to redirect the recursive resolver. Query Name Minimisation [33] addresses this potential privacy issue. Recursive resolvers applying Query Name Minimisation only query for the part of the domain name necessary for the recursive lookup, reducing the leakage of information.

Recursive resolvers, however, would still see the query name and could link the domain name with the IP address of its client. A new proposal, called Oblivious DNS Over HTTPS [34], has the goal to address this issue, by chaining two recursive resolvers. Here, the query is encrypted between the client and a target resolver. An intermediary resolver, between the client and the target, acts as a proxy. In this setup, the proxy knows the client but not the content of the query, and the target knows the content of the query but not the client. If both resolvers are run by independent parties, the privacy of the client and the DNS message is guaranteed.

2.3 QUANTUM COMPUTING

Quantum computers are computers that have the potential to threaten the security of many protocols we use today. In comparison with current computers, quantum computers use quantum mechanics for computations. Thereby, they can calculate some mathematical problems significantly faster than current computers.

Two of these problems lay the foundation for many cryptographic algorithms used in today's Internet protocols, including TLS and DNSSEC. An attacker with a powerful quantum computer at her disposal could, in case of DNSSEC, forge signatures, thereby undermining DNSSEC. Currently, no quantum computer exists that is powerful enough to break current algorithms, but progress is being made. The goal of this section is to give the reader an intuition into why quantum computers might pose a threat to cryptography and therefore also to DNSSEC. We also explain how it could impact DNSSEC, which developments are needed before this threat could become reality, and how much time we still may have to make the DNS quantum-safe.

2.3.1 Attacking Public Key Cryptography

The security of the cryptographic signing algorithms used in DNSSEC is based on mathematical problems that are hard to solve by computers that we have today (current computers). Such problems have in common that they are easy to solve in one direction but hard to solve in the reverse direction and they are often referred to as trapdoor functions. Common problems are calculating the discrete logarithm, on which among others the algorithms of DSA and ECDSA rely, and the factorization of an integer, on which Rivest–Shamir–Adleman (RSA) relies. As an example, we describe how quantum computers can be used to solve the problem of integer factorization efficiently.

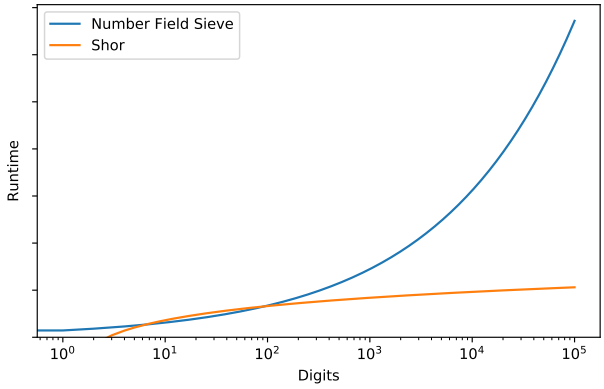


Figure 2.7: Runtime of current and quantum factoring algorithms [35]

Basically, the problem is that given an integer n it is hard to factorize it into its prime factors p and q, such that:

$$p * q = n$$

Going into the opposite direction, multiplying both factors p and q is easy for computers, even with large numbers. The exact details are out of scope, but this principle allows the construction of a private key, used to sign a message, and a public key used to validate the message. In RSA, n is the public key, known to anyone. The private key is derived from p and q, and thus both are kept secret.

A naive approach to reconstruct p and q from n, and consequentially the private key, would be to take all known prime numbers smaller than n and test whether they are an integer factor of n. This approach, however is slow (exponential runtime), especially when applied on numbers with 300 or more digits as used in the RSA signing algorithms. Algorithms that can run on current computers exist that can solve this problem faster (faster than exponential runtime but slower than polynomial runtime), but still not fast enough to efficiently attack public-key cryptography [36].

There, however, does exist a more efficient algorithm that runs only on quantum computers. In 1994, mathematician Peter Shor published a description of an algorithm that can factor integers in polynomial time [37]. As shown in Figure 2.7, Shor's algorithm significantly outperforms the most efficient current algorithm (a Number Field Sieve) when factorizing large numbers. This means that Shor's algorithm could make finding private keys feasible in practice, breaking all public-key signing algorithms currently used in DNSSEC.

There is only one crucial requirement: we first need quantum computers, and they need to be powerful enough to execute Shor's algorithm.

2.3.2 Principles of Quantum Computers and Shor's Algorithm

Quantum computers are computers that rely on quantum mechanics for calculations. Current computers, in contrast, rely on classical mechanics. In practice, this means that current computers calculate with bits which can either take the state of *0* or *1*. In contrast, quantum computers calculate with qubits. Also qubits, when measured, take either the state 0 or 1, but with one important difference: before we measure the state of a qubit it has both values at the same time with some probability. This state is called superposition. This makes it possible to run algorithms on quantum computers that cannot run on current computers and is one of the two attributes of qubits that could make quantum computers a threat to many cryptographic algorithms. We go into more detail about qubits in Explainer 1 below.

The other principle is entanglement. Here, the states of two or more qubits are highly correlated in a way such that the measured state of the second qubit is always the same as the measured state of the first qubit. This attribute allows quantum computers to work highly parallel by design. Both, superposition and entanglement allow quantum computers to vastly outperform current computers in some cases.

Shor's algorithm makes use of both features. The algorithm consists of a "classical" part which runs on current computers, and a "quantum" part, running on quantum computers. In the latter part lays the speed up when solving the factorization problem, shown in Figure 2.7. In the first part of the algorithm, we can reduce the problem of factorization to the problem of order-finding, using a "classical" computer. Then, we can solve the problem of finding the order of a group with the help of a quantum computer, using the quantum Fourier transformation. Finally, we can use the found order to calculate the factor. The last parts runs again on a current computer.

2.3.3 Impact on DNSSEC

This shows that quantum computers applying Shor's algorithm can, in theory, solve the problems underlying many algorithms significantly faster than today's computers. This makes it possible to forge the *secret keys* of these algorithms efficiently. If the secret key was used to encrypt a message, then an attacker could decrypt the cipher text, revealing the secret. In case the secret key is used to sign a message then an attacker can use the forged key to impersonate the signer. DNSSEC relies on the latter application.

In this section, we show how a quantum computer and Shor's algorithm can be used to perform an attack on the DNS and discuss its impact. Furthermore, we describe what is needed to carry out such an attack in practice. Last, we explain why it is important to studying quantum-safe algorithms for DNSSEC already now.

Attacking DNSSEC with a quantum computer We can use Shor's algorithm to attack DNSSEC such that a validating resolver cannot differentiate anymore between signatures created by the domain operator and signatures created by an attacker. Effectively, this means that message authenticity and integrity could not be guaranteed

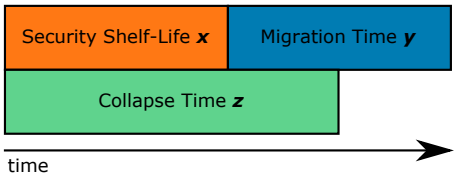


Figure 2.8: Mosca's quantum risk model

anymore.

In such an attack, the attacker would use a quantum computer and Shor's algorithm to reconstruct the private key of the zone operator. Then, the attacker can use the reconstructed private key to sign arbitrary records. The rest of the attack follows the same steps as attacks against zones not signed with DNSSEC. If the attacker has the position of a Man-in-the-middle (MITM) between the authoritative name server of the impersonated domain name and the recursive resolver, then the attacker can insert her forged records and signatures directly into the network stream. If the attacker is off-path, then she would need to carry out a cache poisoning attack, as described by Kaminsky [7].

The impact of a successful attack against a signed message would be even more devastating than the one of a cache poisoning attack against an unsigned message. In the latter case, a recursive resolver needs to treat an unsigned message in the DNS as potentially insecure anyways. In the former case, however, a validating resolver has successfully validated the forged signature, assuming that the message must be correct. This gives the client of a validating resolver with a false sense of security. Additionally, other protocols rely on the security guarantees of a valid DNSSEC signature (see Section 2.2). Since these guarantees would not hold anymore, also the security of these protocols would be undermined unnoticed.

When do we need to act? To answer this question, we apply Mosca's quantum risk model [38]. This simple model can help answering the question, at what point in time quantum computers can become a threat for systems relying on potentially vulnerable cryptographic algorithms. It consists of three variables:

- x **Security Shelf Time** – The time that information needs to be kept secure or the authenticity of a signature needs to be guaranteed.
- y **Migration Time** – The time it takes for the system migrate to quantum-safe algorithms.
- z **Collapse Time** – The time it takes until quantum computers can attack the current system.

The reasoning behind the model is that the security guarantees of a system hold as long as:

Physical and logical qubits

The implementation of qubits in quantum computers is error prone. This is why we need to differentiate between *physical* qubits, implemented in a quantum computer, and *logical* qubits, with which we can actually implement algorithms. Due to their sensitivity to errors, physical qubits cannot be directly used for calculations. Instead, a quantum error correcting code is applied to multiple physical qubits. The result of the error correcting code is one logical qubit with which we then can perform operations. Currently, estimates are that hundreds of physical qubits are needed to encode one logical qubit [42].

$$x + y < z$$

Meaning, we can consider a system secure as long as the time it takes for a system to migrate to a quantum safe algorithms *plus* the time the security guarantees of the system must hold is *smaller* than the time it takes until a powerful enough quantum computer becomes available. Figure 2.8 visualizes the relationship between these variables.

Status of quantum computer development Applying this formula to DNSSEC, we first look at the time it takes until quantum computers could attack the algorithms used in DNSSEC (*collapse time z*). This is hard to estimate. Estimates are that in order to factor an n-bit number a circuit with $2n + 3$ logical qubits would be required [39]. For a 2048 bit RSA number, this translates to 4,099 logical qubits. In practice, the number of *physical* qubits is significantly higher (see Explainer 1 for the difference between physical and logical qubits). Looking at the current state of research, one paper claims that factorizing a 2048 bit RSA integer in 8 hours would require 20 million physical qubits [40]. Others [41] estimate that about 1 million physical qubits and 100 days would be required for the same attack. In comparison, the largest quantum computers have around physical 50 qubits [41], as of writing this thesis.

This shows that current quantum computers are not a threat yet, but there is good reason to believe that quantum computers can become more powerful. A study by the German Federal Office for Information Security (BSI) from 2020 [41] discusses the progress of the development of quantum computers. Here, they find that the developments of the last years demonstrate that the system integrations challenge, necessary to scale quantum computers, can be mastered. Also, the research on quantum computers is taking up pace, now that more private organizations are getting involved. The authors of the BSI study believe that this can also increase the chance of ground breaking findings, which could address some of the major road blocks in scaling quantum computers. The Dutch research organization TNO tried to estimate how long it might take until a quantum computer, powerful enough to break current algorithms, will be developed [13]. In their most optimistic estimation

they expect that such computers exists in around 15 years. It might take twice as long according to their more pessimistic estimation.[8]

At the same time, research of quantum computers still faces setbacks frequently. For example, a technology that promised to address some of the major challenges of scaling quantum computers got set back several years after doubts about the legitimacy of previous research results were raised [43].

Because of these uncertainties, it is important to know how fast we need to transition to quantum-safe algorithms when the first promising quantum computers appear on the horizon. We could still take measures on time if we would understand which factors influence the length of the transition beforehand. For this reason, we first look at the time the security guarantees of DNSSEC need to hold (*Security Shelf Time x*).

Attack time window In DNSSEC, the Security Shelf Time is the time recursive resolvers trust the public key. An attacker that has replicated a private key of a TLD or of a domain lower on the hierarchy, can successfully spoof responses until the attacked recursive resolver drops the compromised public key from its cache. This happens either when the TTL of the public key, or the signature of the public key has expired. For the signed domain names of `.com`, the median `TTL` of `DNSKEY` records is 1 hour. Signatures are valid longer, their median life time is 21 days [21].

In case of the KSK of the root the attack window is the time it takes until the trust anchor is replaced at validating resolvers. This time can vary greatly, since it is dependent on software vendors shipping upgrades or operators replacing the key manually. We describe this process in more detail in Chapter 4. In case resolvers implement protocols to update trust anchors automatically, it would take 30 days before a new key is accepted if the root KSK would be broken today.

Applying both considerations to Mosca's risk model, the *Security Shelf Time x* for DNSSEC ranges between a couple of hours and multiple days. This, however, is relatively short in comparison to the time it takes to move the complete DNS ecosystem to a quantum-safe algorithm.

Migration time The time it takes until the DNS ecosystem has fully migrated to quantum-safe algorithms (*Migration Time y*) is the second variable influencing the point in time at which the security of the DNS is at risk. None of the Internet protocols, currently more widely used, have transitioned to quantum-safe algorithms yet, but experience from the past indicates that such a process may take years. For example, in TLS current broken cryptographic algorithms were replaced with more secure ones, but this process took more than 10 years [44].

Also in DNSSEC some algorithms used in the early days are not considered secure anymore and are getting replaced. In Chapter 3 of this thesis, we analyze how long this transition is taking and show that also for DNSSEC this transition takes 10 years or more. Since quantum-safe algorithms are not standardized yet and, as we

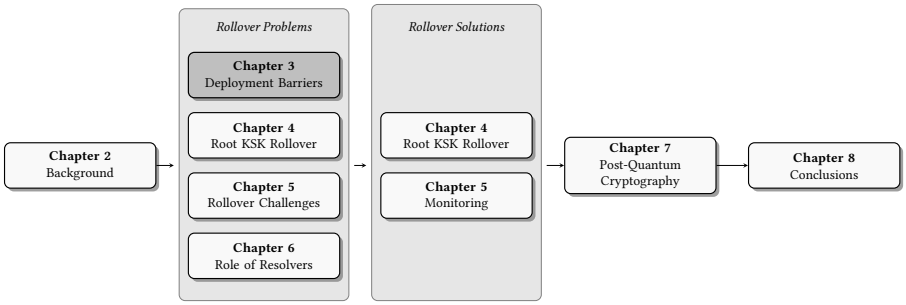
[8] In their optimistic scenario, the authors assume that the number of physical qubits would double every year, but only every second year in their more pessimistic scenario.

show in Chapter 7, their deployment might require some more drastic measures, the *Migration Time* of DNSSEC to quantum-safe algorithms will likely take more than a decade.

Adding up the *Security Shelf Time* x and the *Migration Time* y of DNSSEC it becomes clear, that quantum computers do not pose a threat yet. It does, however, also show that studying the transition to more secure algorithms in DNSSEC is necessary already today. In the next chapters, of this thesis we take the first steps in this direction.

CHAPTER 3

Algorithm Deployment Barriers



In this chapter, we study the main barriers for introducing a new algorithm in DNSSEC. The goal is to answer the question if DNSSEC has achieved algorithm agility. A protocol has achieved algorithm agility if it is possible to replace algorithms easily, according to RFC 7696 [45]. If this would be the case for DNSSEC, it would have already taken the first step to move to more secure algorithms fast in the future. To achieve algorithm agility, each stage of algorithm deployment should be free of major barriers. In this chapter, we look at the deployment of new algorithms in DNSSEC in the past. This allows us to identify barriers that hold back algorithm deployment and which should be addressed before we can easiliy transition to more secure algorithms in the future. The study discussed in this chapter was carried out in 2020 and published as a research paper at an academic conference [14]. The results were also presented at an ICANN DNSSEC workshop [46].

3.1 INTRODUCTION

In the previous chapter we have shown that the security of DNSSEC relies heavily on the security of the underlying algorithms. When DNSSEC was standardized, operators had the choice of just three algorithms to sign their domain names. Over the past 15 years, 9 new algorithms were added and 5 were deprecated [47]. New algorithms were introduced that were more secure or that had more attractive attributes, like smaller keys and signatures.

As shown in the previous chapter, quantum computers could threaten the security of all algorithms standardized for DNSSEC. There, we have also shown that we need to understand how fast we can transition to secure algorithms before the threat of quantum computers becomes more imminent.

In this chapter we study if it was possible to replace algorithms *easily* in the past. We study how long this process took and which problems occurred. These insights help us to understand when quantum computers might become a threat to DNSSEC.

In DNSSEC, introducing new algorithms and replacing existing ones is a four-stage process and we explain it in more detail in Section 3.2: (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.

We analyze the full algorithm life cycle to answer the question: *Has DNSSEC achieved algorithm agility?* If, this would be the case, then DNSSEC would have taken the first step to easily transition to quantum-safe algorithms in the future. 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 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 chapter we first introduce the algorithms currently standardized for DNSSEC and the algorithm life cycle in more detail (Section 3.2). Section 3.3 discusses related work. We describe our data sets in Section 3.4. Then,

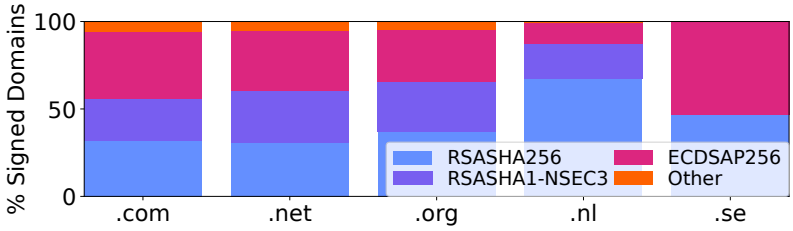


Figure 3.1: Signed domains per algorithm and TLD

our analysis is split into four parts. First, we analyze the process for standardizing new algorithms in Section 3.5. Next, we describe, how well software, registries and registrars support algorithms (Section 3.6). Then, we analyze their deployment at domain names (Section 3.7.1), and at resolvers (Section 3.7.2). Finally, we analyze the deprecation and replacement of algorithms in Section 3.8. To conclude, we discuss to what extent DNSSEC has achieved algorithm agility and which are the most pressing problems when transitioning to a new algorithm (Section 3.9).

3.2 DNSSEC ALGORITHMS AND LIFE CYCLE

	ID	Algorithm	First Draft	Standardized	Days	DNSSEC Signing	DNSSEC Validation
DNSKEY algorithms	1	RSAMD5 [48]	Aug. 2000	May 2001	273	MUST NOT	MUST NOT
	3	DSA [49]†	Sep. 1997	May 2004	546	MUST NOT	MUST NOT
	5	RSASHA1 [48]†	Aug. 2000	May 2001	273	NOT RECOMMENDED	MUST
	6	DSA-NSEC3-SHA1 [50]†	Jan. 2005	Mar. 2008	1520	MUST NOT	MUST NOT
	7	RSASHA1-NSEC3-SHA1 [50]†	Jan. 2005	Mar. 2008	1520	NOT RECOMMENDED	MUST
	8	RSASHA256 [51]	Feb. 2006	Oct. 2009	1338	MUST	MUST
	10	RSASHA512 [51]	Feb. 2006	Oct. 2009	1338	NOT RECOMMENDED	MUST
	12	ECC-GOST [52]†	Apr. 2009	Jul. 2010	456	MUST NOT	MAY
	13	ECDSAP256SHA256 [53]	Jan. 2011	Apr. 2012	456	MUST	MUST
	14	ECDSAP384SHA384 [53]	Jan. 2011	Apr. 2012	456	MAY	RECOMMENDED
	15	ED25519 [54]	Jul. 2015	Apr. 2017	581	RECOMMENDED	RECOMMENDED
	16	ED448 [54]	Jul. 2015	Apr. 2017	581	MAY	RECOMMENDED
DS alg.	1	SHA-1 [55]	May 2001	Dec. 2003	944	MUST NOT	MUST
	2	SHA-256 [56]	Nov. 2005	May 2006	181	MUST	MUST
	3	GOST R 34.11-94 [52]	Apr. 2009	Jun. 2010	456	MUST NOT	MAY
	4	SHA-384 [53]	Jan. 2011	Apr. 2012	456	MAY	RECOMMENDED

Table 3.1: Algorithms standardized for the use in DNSSEC. Recommendations from RFC 8624 [47]. Algorithms marked with † are not recommended or strongly not recommend since 2019-06-11.

DNS operators can currently choose between 12 different algorithms to sign their domain names (top of Table 3.1), 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 chapter, we use names to refer to an algorithm. To ensure interoperability, RFC 8624 [47] 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 3.1 lists these as well. In case a resolver does not support an algorithm, it should treat the RR as *insecure* [57]. Figure 3.1 shows a breakdown of the signing algorithms used in the five TLDs we analyse in this study 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 chapter.

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 [47] 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* [16], [19].

3.3 RELATED WORK

York et al. [58] 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.* [59] 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.* [60] analyze the *quality* of DNSSEC deployments. They find that often insecure algorithms are deployed. A 2016 study by Van Rijswijk-Deij *et al.* [61] analyses early deployment of ECDSA. Finally, in recent work Müller *et al.* [21] perform a case study of the feasibility of using quantum-safe cryptographic algorithms in DNSSEC. This study also forms Chapter 7 of this thesis.

In this study 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. [58], and extend this work by analyzing real world data. Like Van Rijswijk-Deij et al. [61], we rely on data collected by the OpenINTEL DNS measurement platform [62], which now covers more than five years (see Section 3.4). This allows us to study the adoption and deprecation of additional algorithms compared to [61]. We carry out additional active measurements, providing a new perspective on algorithm deployment. Thereby, our study has a similar focus as work by Kotzias *et al.* [44]. They show how TLS deployments and cipher suites have changed over several years. In Section 3.9, we compare their findings in TLS with our findings in DNSSEC.

3.4 DATASETS

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

Standardization We study IETF mailing lists [63], combined with first-hand experience [64]. 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 3.2). 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 [65]. 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 [66].

Signing To study the deployment of algorithms at domain names we rely on the daily, active, DNS measurements of OpenINTEL [62]. 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 [22]. 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 [67].

Validation We measure the uptake of DNSSEC validation with RIPE Atlas [69], querying domain names under our control using their pre-configured resolvers. See Explainer 2 for more details on RIPE Atlas. For our measurements, we rely on over

RIPE Atlas

is a global Internet measurement network. By 2021, the network consists out of more than 11,000 nodes located in 176 different countries.[a] In principle, everyone can use RIPE Atlas to execute measurements on the Internet. These include basic Ping measurements, e.g. to measure the reachability and responsiveness of IP addresses, but also more advanced HTTP measurements to measure web servers. In our studies, we mostly relied on DNS measurements, with which we can, among others, initiate nodes to lookup records of domain names. For these measurements, we can instruct nodes to send DNS queries directly to authoritative name servers, or to employ a recursive resolver, set by the owner of the node.

Volunteers run the nodes and the RIPE NCC (the regional Internet registry for Europe, West Asia, and the former USSR) operates the platform that coordinates the measurements and collects and processes the measurement results. Nodes fall into two categories: *probes* and *anchors*. Probes are lightweight measurement nodes which can be hosted by anyone. For this reason, they might not be available all the time or attributes assigned to the probe, like its geographical location, might be incorrect.[b] Anchors, in contrast, are more powerful instances, usually managed by organization and have higher availability. The majority of probes are dedicated physical machines, but since 2020 the RIPE NCC made it possible to run software probes as well, for example on a virtual machine.

We rely on RIPE Atlas in every study of this thesis. Some measurements were carried out in cooperation with the RIPE NCC, allowing us to run measurements with every available probe and with high frequency. In each chapter, we describe in more detail the individual RIPE Atlas measurements and the number of involved probes. The results of our RIPE Atlas measurements are publicly available for reproducibility. At the end of this thesis, in Appendix A, we list the individual measurements.

Despite the large number of probes and their distribution across thousands of networks (3,699 IPv4 Autonomous Systems (ASs) and 1,620 IPv6 ASs), RIPE Atlas does not reflect the general Internet population. For example, RIPE Atlas probes are still underrepresented in countries on the African and the South American continent. Software probes have the goal to address this issue since they can be deployed more easily in hard-to-reach locations [68].

[a] `https://atlas.ripe.net/results/maps/network-coverage/`

[b] The status and other attributes of a probe are published on an individual website, provided by the RIPE NCC, e.g.: `https://atlas.ripe.net/probes/33302/`.

11,000 vantage points, 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 3.7.2.

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 [70]. Furthermore, rather than performing our own measurements, we re-use data collected by the OpenINTEL project [62].

3.5 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 [71].

3.5.1 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 [72].

3.5.2 Barriers

This process usually takes around one year and Table 3.1 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 [73], [74], 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)' [64].

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 conducting this study. 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 [75]. 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 [76].

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 [77]. 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 [50]). 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 3.1; because at first SHA256/SHA512 with NSEC3 required a separate identifier, algorithm numbers 9 and 11 are marked as *reserved* [78] 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 ones, 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.

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

3.6 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.

3.6.1 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, (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.

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 3.2 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 [79] and one of the developers co-authored the standard [53]. 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 [80].

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 [47].

Support of *deprecated* algorithms varies between DNS software. Unbound added the *option* to treat SHA-1 based algorithms as insecure in 2017 [79]. BIND9 and the name server of PowerDNS removed support for GOST in 2019 and 2020 respectively [81], [82]. 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.

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 3.2). 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 [83], [84]. Both of these libraries, however, typically need to be installed manually by operators (*i.e.,* are not part of default OS installations).

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.

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

The versions of software shipped with a particular Ubuntu release is not usually updated [85] 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 [86]. Therefore, OpenBSD users generally receive software updates faster, including support for new algorithms.

3.6.2 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

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.* [59] 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

[2] The `CDS` and `CDNSKEY` protocols [87], [88] 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.

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 3.3: 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.

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.* [59], 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 3.3 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 [59]. 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 [47]. 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 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.

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 [90], which algorithms they support. We received responses for

[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 [59].

[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 [89].

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 this study (May 2020) [91]. By the time of writing this thesis (March 2021), DNSViz now supports ED448 as well.

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 [92] and `.nl` followed later the same 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 [93].

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.

3.7 ALGORITHM DEPLOYMENT

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

3.7.1 Signing

For signing, we focus on the ECDSA-based algorithm ECDSAP256 since it is widely deployed and our data sets cover its rollout 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

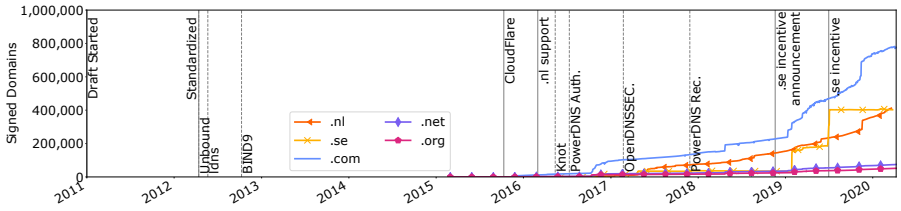


Figure 3.2: Domains signed with ECDSA256 and resolvers able validating this algorithm

level. It is less prone to issues during transport and makes DNSSEC less attractive for Denial of Service Attacks [94].

Slow uptake

Even though signers BIND9 and LDNS already supported ECDSAP256 in 2012, we only see 21 domains signed in Figure 3.2 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 [95]. 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 [96]. 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 [60]. But in November 2018

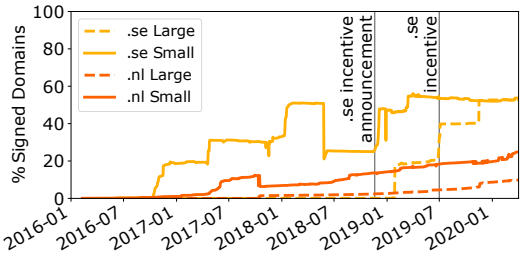


Figure 3.3: Share of .se and .nl domains signed with ECDSAP256, by provider size

.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) [92]. On July 1, 2019, the new incentives came into force.

Large operators

From the preceding analysis, it appears that DNS providers (registrars or third-party operators) drive adoption. We verify this by grouping domain names by DNS provider, using the MNAME in the SOA record (previously applied in [60]). 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 3.3, 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 3.3 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.

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*).

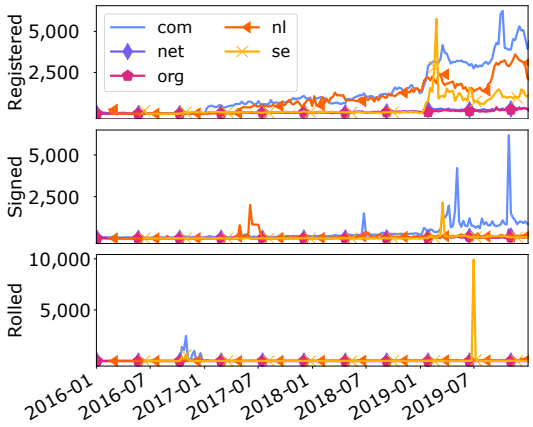


Figure 3.4: Newly signed with ECDSA per week

Figure 3.4 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 3.4 (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 to an *upgrade* to ECDSA. In less than 1% of all algorithm rollovers operators were changed at the same time.

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 3.5). In October 2013, the first new Generic Top Level Domains (gTLDs) were delegated [97]. For new gTLDs, DNSSEC deployment is mandatory [98] 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).

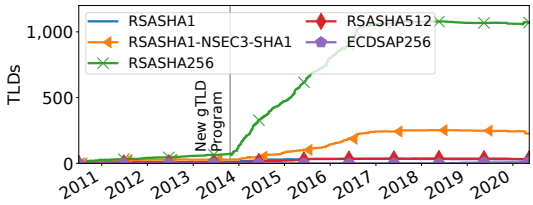


Figure 3.5: Algorithms used to sign TLDs

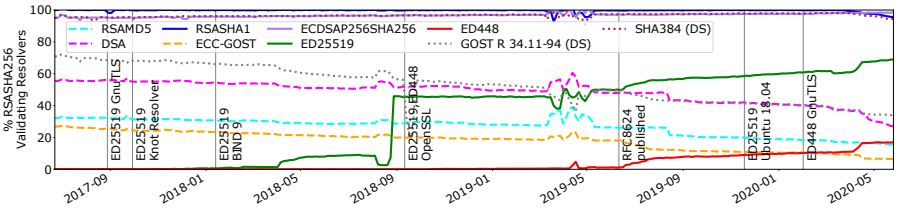


Figure 3.6: Overall progression of DNSKEY algorithm support in DNSSEC validating resolvers, measured via RIPE Atlas

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.

3.7.2 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 [99] and the OpenDNS resolver fail on RSA-MD5 (1).

The DNS zones for all the DNSKEY algorithms are subzones of the domain name `rootcanary.net` [100]. 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

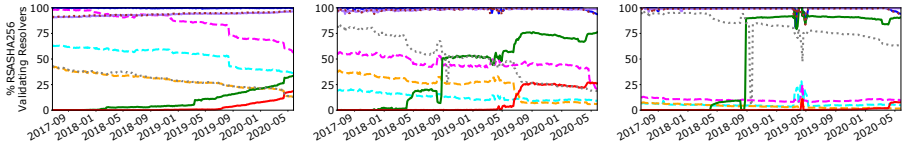


Figure 3.7: Validation at internal, forwarding and external resolvers

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 [57] (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.

DNSKEY algorithm support in validating resolvers

Figure 3.6 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 3.6 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.

The uptake of ED25519 and ED448

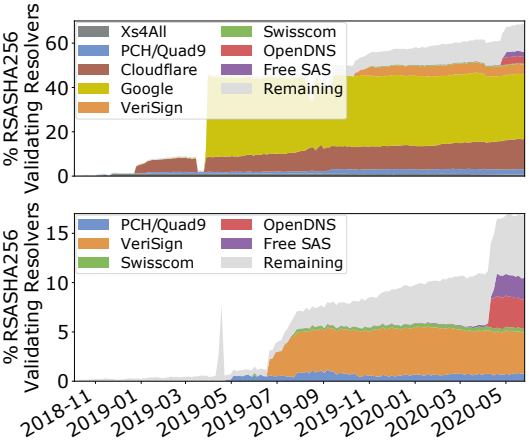


Figure 3.8: Uptake of ED25519 and ED448 validation

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 [101]). 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.

In Figure 3.7 (left) we can see that internal resolvers adopt ED25519 slower than external resolvers (Figure 3.7 right). We therefore zoom in on external resolvers.

Figure 3.8 shows the top Autonomous System Number (ASN) 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 Feb-

ruary 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.

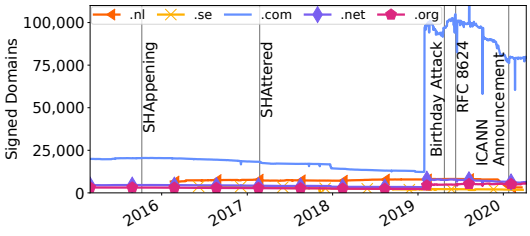
3.8 ALGORITHM DEPRECATION AND REPLACEMENT

DNSSEC algorithms can also be *deprecated*. This happens when the IETF publishes a new standard, advising against the use of an algorithm [47]. 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.

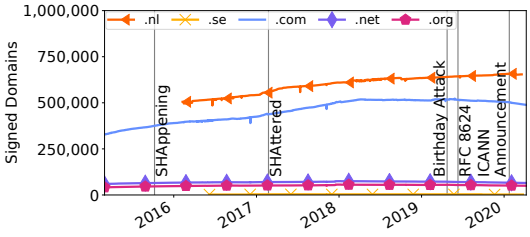
Since the standardization of DNSSEC, three algorithms must not be used to sign zones anymore: DSA, DSA-NSEC3-SHA1 and ECC-GOST. For security reasons, two more are not recommended for signing anymore: RSASHA1 and RSASHA1-NSEC3-SHA1. Additionally, SHA-1 and GOST should not be used when creating `DS` records.

3.8.1 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 [102], DSA-NSEC3-SHA1 and RSASHA1-NSEC3-SHA1 were standardized three years after. Since 2010, NIST disallows use of SHA-1 for governmental agencies [103]. Attacks in 2015 [104] (referred to as SHAppening), 2017 [105] (SHAttered) and 2019 [106]



(a) RSASHA1



(b) RSASHA1-NSEC3-SHA1

Figure 3.9: Deployment of deprecated algorithms

(Birthday Attack) undermined its security further. Also in 2019, RFC 8624[47] advised against using DSA/SHA1 and DSA-NSEC3-SHA1 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 [107].

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).

Second level domain names

Figures 3.9a and 3.9b 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.

TLDs

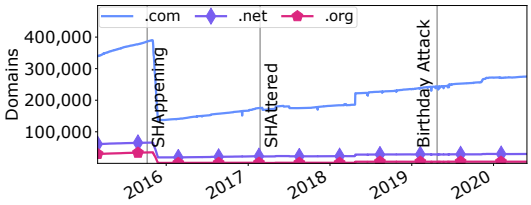
Early deployments at TLDs preferred mostly RSASHA1 but RSASHA1-NSEC3-SHA1 and especially RSA/SHA-256 gained ground fast (see Figure 3.5). RSASHA1-NSEC3-SHA1 reached its highest deployment in February 2018 and shrinks continuously since then. Although hardly visible, in April 2020, 16 TLDs moved away from signing with RSASHA1-NSEC3-SHA1. These, however, all belong to the same provider.

DS algorithms

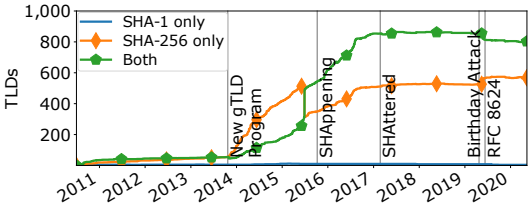
SHA-1 is also used to create DS records that link the KSK of the child with the ZSK of the parent. Operators can choose between four different hash algorithms (see Table 3.1). At .nl, DS records are created by the registry, resulting in SHA-256 DS records only. At the other TLDs all four algorithms are seen.

Figure 3.10a 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 [108]. Figure 3.10b shows, only in the first days of DNSSEC deployment SHA-1 was more often used than SHA-



(a) TLDs: SHA-1 only



(b) Root: All combinations

Figure 3.10: DS digest algorithm

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.

3.8.2 Resolver Validation

Algorithms RSAMD5, DSA and DSA-NSEC3-SHA1 must not be considered secure by resolvers [47]. GOST may still be validated.

Figure 3.6 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 3.7a). 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 [47] 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 3.6 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.

3.9 CONCLUDING REMARKS

At the start of this chapter, we asked: *Has DNSSEC achieved algorithm agility?* Based on our analysis, we conclude, rather unsatisfactorily: *only partially.* The introduction of ECDSA has shown that 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 of other algorithms, show that 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 [15], with no algorithm rollover in sight.

We can also see similarities with TLS. Kotzias et al. [44] 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.

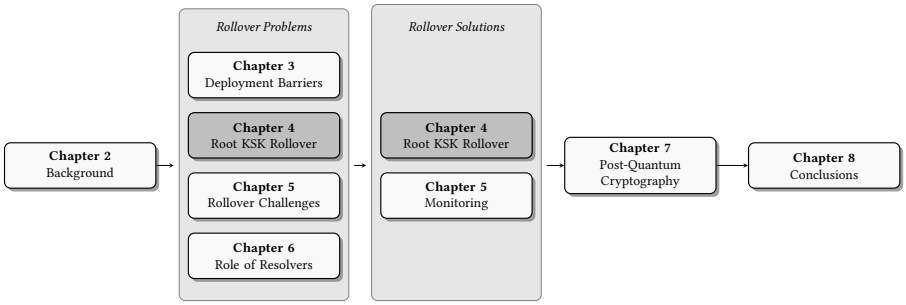
Looking into the future, the threat of quantum computers and Shor's algorithm appear on the horizon [37]. Based on our observations we conclude that quantum-safe algorithms will only be deployed widely, (i) if they are well supported in soft-

ware and the registration channel, (ii) operators actually feel the need to move (e.g. because current algorithms are broken or through financial incentives), and (iii) algorithm rollovers can be carried out easily. 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.

In the next chapter, we study one main component of the DNS that we left out so far: the root. We study the first ever root KSK rollover – a prerequisite for algorithm rollovers at the root.

CHAPTER 4

Key Exchange at the Root



In the previous chapter, we discussed the main problems when deploying new algorithms at TLDs and second-level domain names. There, we found that algorithm rollovers are one of the main challenges that hinder the deployment of new algorithms. In this chapter, we turn our attention to the root. Here, we study the first ever root KSK rollover. A KSK rollover is a prerequisite for rolling the algorithm at the root. A successful root KSK rollover would show that the DNS ecosystem would have reached an important milestone on its way to deploying a new algorithm at the root as well. In this chapter, we identify which barriers still exist and propose measures to improve future KSK and algorithm rollovers. The root rolled its KSK in 2018 and we published our study covering this event at an academic conference in 2019 [15]. There, it also received the Distinguished paper award. Furthermore, some of the measurements of this study were published live during the rollover itself. After the rollover, we presented our results also at other venues (RIPE 79 [109] and NLUUG [110]).

4.1 INTRODUCTION

In the previous chapter, we have identified *algorithm rollovers* as one of the major barriers for moving to new algorithms in DNSSEC. Since the security of DNSSEC relies heavily on the security of the root it is very important that also the root can transition to more secure algorithms in the future.

Since 2010, the root of the DNS is secured with DNSSEC. As of writing this thesis, the root is still signed with the same algorithm with which it was signed more than 10 years ago: RSASHA256. This also means that we cannot observe the effects of an algorithm rollover directly.

However, in 2018 and eight years after the root was secured with DNSSEC the KSK was replaced for the first time, following established policy that requires regular rollovers of the root KSK [111]. The steps necessary to replace an algorithm are similar to the steps necessary to replace the KSK at the root. If the replacement of the KSK would fail, then likely an algorithm rollover would fail as well. In this chapter, we measure the root KSK rollover, assessing whether it was a success or not.

The Root KSK Rollover (hereafter "the rollover"), required years of preparation and was considered risky. Stakeholders expected, in the worst case, millions of Internet users (up to 13%) to become unable to resolve a domain name [112].

The Internet Corporation for Assigned Names and Numbers (ICANN), the organization responsible for coordinating and rolling the key, collected feedback from the community before the rollover. Two risks were most feared: (i) resolvers that would not update their local copy of the key [112] and (ii) resolvers that could not retrieve the key material from the root because it might exceed a packet size that cannot be safely handled by some networks (we explain these two risks in more detail in Section 4.2).

Leading up to the initially scheduled date of the rollover in October 2017, ICANN and its stakeholders carried out measurements to estimate the potential impact of both risks and considered the former acceptable. The actual impact of the former, however, was still hard to estimate. One of the reasons was the introduction of a new protocol that enabled resolvers to signal their configured key to the root server operators (RFC 8145 [113], we explain the protocol in more detail in Section 4.3.1). This protocol signaled that a significant number of resolvers only had the old key configured and this led to the decision to postpone the rollover [114]. Rescheduling the rollover gave researchers the opportunity to understand which resolvers sent this signal and estimations were that only a few users would be negatively affected by the rollover [115]. This gave ICANN the confidence to move forward with the rollover. The actual rollover was carried out on October 11th, 2018. In their March 2019 review of the rollover, ICANN concluded that *"there were no significant outages"* and that the rollover *"was an overwhelming success"* [116].

In this chapter we provide a comprehensive analysis of the rollover, starting from the publication of the new key in July 2017 until the removal of the old key in March 2019. We use data that was actively and passively collected at key points in

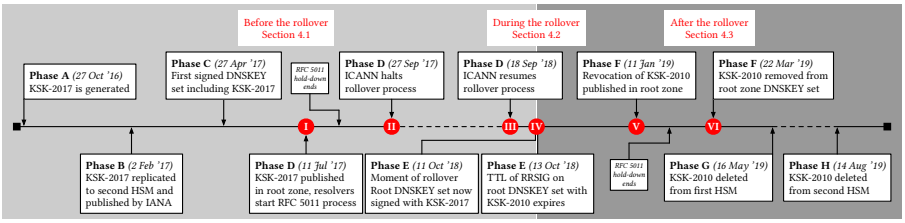


Figure 4.1: Time-line of the Root KSK rollover

the DNS ecosystem over the entire duration of the rollover. We, as members of the DNS community, actively supported the rollover process with timely data analyses. This provides us with a unique perspective that covers multiple vantage points of the rollover.

The main contributions of this chapter are that we:

(i) Provide the first in-depth analysis of the root KSK rollover, a unique event with an impact on the global Internet;

(ii) Cover the event from multiple perspectives, that of root operators, of resolver operators, and end users;

(iii) Validate ICANN's conclusion that the event was a success and show that, while this conclusion generally holds for end users, there are observable challenges at all stages of the rollover;

(iv) Perform an in-depth analysis of the causes of the challenges seen at all stages of the rollover;

(v) Give recommendations for improving telemetry, processes for root key management and future rollovers.

In the remainder of the chapter, we outline the stages of the root rollover and the risks involved (Section 4.2). Next, we introduce our measurement methods and data (Section 4.3). Then, we split the analysis of the rollover into three sections, *before*, *during* and *after* the rollover (Section 4.4). In Section 4.5 we discuss related work and in Section 4.6 we provide recommendations for better telemetry and rollover process improvements based on our analysis. In this section, we also discuss the implications of our findings on a future algorithm rollover. We conclude the chapter in Section 4.7. Here, we also discuss what our results mean for future algorithm rollovers at the root.

4.2 THE ROOT KSK ROLLOVER

It is considered good operational practice that operators of zones signed with DNSSEC be able to periodically change, or "roll," the zone's cryptographic keys. A rollover

might be necessary in case of a security breach, in case operators want to upgrade to a new algorithm, or because they follow a key management policy [117]. The root zone's ZSKs are rolled every calendar quarter [118]. When the root zone was first signed in 2010, it was generally accepted that the KSK would be rolled after a period of 5 years [111]. The parties involved in operating the root zone began discussing and planning a KSK rollover in 2013, but this work was put on hold when the NTIA announced its intention to transition oversight of the Internet Assigned Numbers Authority (IANA) functions to the Internet community [119]. Work on the rollover resumed in 2015, culminating in a 2016 Rollover Design Team report [112]. ICANN and Verisign, in their respective roles as the IANA Functions Operator and Root Zone Maintainer, used the design team report to develop a final set of operational plans [120]. These plans describe the process for replacing the old KSK, further referred to as *KSK-2010*, with a new KSK, now referred to as *KSK-2017*. Figure 4.1 shows a timeline of each of the phases of the rollover as described in the operational plan. We have highlighted six key events in red labeled **I** – **VI**. These six events are the focus of this study. In the rest of this section, we explain the risks as identified in the design team report and specific considerations that stem from the special role of the root's KSK as a *trust anchor*.

Risks during the rollover

The design team report [112] identifies two major risks: validating resolvers that are unable to configure the new KSK as a trust anchor, and the increase in response size of the `DNSKEY` RRset at certain stages of the rollover process.

DNSKEY RRset changes Resolvers need a copy or a hash of the root KSK, and to configure it as a trust anchor. Some modern resolvers, e.g. BIND, ship with the current root KSK configured as a trust anchor. Thus, resolvers shipped with only *KSK-2010* need a mechanism to fetch *KSK-2017* before the rollover. If this does not occur, these resolvers fail validation as soon as they need to validate a signature signed with *KSK-2017*, when the root zone is published with its `DNSKEY` RRset signed by *KSK-2017* (**IV** in Figure 4.1).

Resolvers that receive a `DNSKEY` RRset without a key that matches their trust anchor may start sending extra `DNSKEY` queries to the root. There are two reasons for this: First, some resolver implementations are designed to retry failures, including validation failures, at some or all of the available authoritative name servers. Second, resolvers typically cache such a failure for a short time only (so-called *negative caching*). Once the cached failure expires, the process starts anew. Negative caching times are typically much shorter than the TTL of the root `DNSKEY` RRset (currently 48 hours).

Clients relying on resolvers with an incorrectly configured trust anchor may receive responses with the `SERVFAIL` error code because the resolver failed to perform DNSSEC validation. ICANN's KSK rollover design team expected the number of resolvers that could not update their trust anchor to be low [112]. This degree of

Figure 4.2: DNSKEY response sizes during the rollover

confidence was based on the RFC 5011 mechanism implemented by most resolvers and that we describe in the next section. In Section 4.4.2, we measure the actual impact of the rollover on resolvers and clients.

Response size changes Due to the KSK/ZSK split, the size of most responses remains the same during the KSK rollover. Only the size of a DNSKEY response changes. Figure 4.2 illustrates the sizes of various DNSKEY responses that occur throughout the rollover process, varying from 864 to 1,425 octets. The sizes shown in the figure include the question and standard EDNS0 data. Some root servers have deployed DNS cookies, which adds another 28 octets to the sizes shown. These response sizes can exceed the Maximum Transmission Unit (MTU) of some networks, which can cause fragmentation of UDP packets. Firewalls and other middle-boxes sometimes block fragmented packets [121], [122], which can hinder resolvers when trying to receive the DNSKEY record set and thus make it impossible for them to validate signatures. The measurements carried out by ICANN and the community leading up to the rollover indicated up to 6% of resolvers could be affected by this problem. These serve less than 1% of users and most do not perform DNSSEC validation [112]. Root servers may also receive an increased number of ICMP packets signaling the packet size exceeds the network's MTU. Clients relying on these resolvers could experience an increased response time or receive a DNS SERVFAIL response. We study the impact of increased response sizes during the revocation in Section 4.4.3, when the highest packet size during the rollover process occurred.

Updating trust anchors

DNSSEC allows validators to automatically update their trust anchors through an in-band mechanism in the DNS, known as RFC 5011 [123], which works as follows. At the start of a rollover, the new key (*KSK-2017*, introduced at **I**) is added to the DNSKEY RRset, but the RRset is only signed with the then current trust anchor (*KSK-2010*). This signals to resolvers that support RFC 5011 that they should start the process of accepting the newly introduced key as a trust anchor. Acceptance is not effective immediately; instead, a *hold-down timer* starts, lasting 30 days. Only if the resolver has seen the new key consistently throughout the hold-down period will it accept the new key. This prevents malicious actors who have gained access to a trust anchor from instantly injecting a new trust anchor. Once the new trust anchor comes into effect, the old one may be revoked. In RFC 5011 this is achieved

by publishing a DNSKEY RRset in which the old key is marked with a revocation flag (at **V**). Again, after a 30-day hold-down the trust anchor is then removed by resolvers. Most resolver software (e.g. BIND, Unbound and Knot) supports RFC 5011 and among popular implementations, only PowerDNS lacks support. The widespread support of RFC 5011 gave the Rollover Design Team confidence that most resolvers would pick up the new key on time [112].

This KSK rollover was the first real test of RFC 5011. Since the publication of RFC 5011 in 2007, new technologies have been introduced that were not considered back then. This includes widespread use of virtual machines and containers, configuration management tools such as Puppet and Ansible, and DNS resolvers running on inexpensive, and hard-to-update home and small office routers.

Where RFC 5011 specifies an *in-band* approach, an *out-of-band* approach is discussed in RFC 7958 [124]. In this approach, resolvers and other applications can retrieve keys and/or hashes directly from the website of IANA as an XML document. Applications can use various approaches to validate correctness of this information, e.g., trusting protections provided by TLS or a digital (PGP) signature file, published separately. The Unbound resolver software uses this mechanism in situations when updates via RFC 5011 fail [125].

With both mechanisms, it is not possible for third parties to determine which resolvers have configured *KSK-2017*. To address this, new resolver software supports protocols that try to provide this insight. We use these protocols to measure the deployment of *KSK-2017* in Sections 4.4.1 and 4.4.3 and discuss their use in Section 4.6.

4.3 DATASETS AND METHODOLOGY

We use a broad set of passive and active measurements at different vantage points in the DNS hierarchy to cover the most critical phases of the rollover. We discuss these datasets and how we use them to analyse the rollover below. We also make the processed data sets and the accompanying scripts for each figure available [126].

4.3.1 Passive Measurements

The DNS root system has 13 root server identities, each of which is run by one operator (see for a full list Table 2.2 in Chapter 2) [25]. At various stages of the rollover, we use passive datasets from select root servers or aggregate data for all of the root servers from a public repository. More specifically, we use the following datasets:

Root queries The Domain Name System Operations Analysis and Research Center (DNS-OARC) collects DNS traces from various name servers including the root system. This includes their well-known annual Day-in-the-Life (DITL) datasets [127]. Given the significance of the KSK rollover, DNS-OARC co-ordinated a DITL data collection from root operators spanning an 82-hour window around the dates of

the actual rollover. We utilized this data, available to researchers and DNS-OARC members, to provide a holistic view of root query traffic during the rollover.

Our analysis, however, extends to well before and after the rollover. To support this, we make use of query datasets collected at three root servers, A, B and J. This non-public longitudinal data, spanning 2017–2019, was made available by Verisign (A/J Root) and the University of Southern California's Information Sciences Institute (B Root). These datasets are used throughout the analysis in Section 4.4 whenever we require detailed information about specific resolvers that exhibit anomalous behavior. Note, however, that other root servers might show different query patterns [128].

RSSAC measurements The ICANN Root Server System Advisory Committee (RSSAC) [129] advises ICANN about operational matters relating to the DNS root system. RSSAC defined a set of metrics that all root server operators are expected to publish on a daily basis [130]. The resulting data is published as YAML files, accessible through a public GitHub repository [131], with data going back to 2013. In this paper, we make use of the RSSAC002 data on traffic sizes to the root, as a proxy for `DNSKEY` queries in Section 4.4.3 and to estimate the impact of the increased `DNSKEY` RRset size in Section 4.4.3. The data is available for all root servers, except G Root.

Trust anchor signals RFC 8145 [113] describes a protocol allowing DNSSEC validators to signal the keys in their trust anchor set. RFC 8145 signals are 16-bit "key tags," encoded as hexadecimal values in DNS queries. *KSK-2010* has key tag `19036`, or `4a5c` in hexadecimal. *KSK-2017* has keytag `20326`, or `4f66` in hexadecimal. A validator that implements RFC 8145 periodically sends a query whose first label starts with the string "`_ta-`" followed by a hyphen-separated list of hexadecimal key tag values. It then appends the name of the zone to which the keys belong.[1] Table 4.1 shows root zone trust anchor signal strings and their meanings.

In this paper we use two RFC 8145 data sets: (i) all trust anchor signals received by A, B and J Root from up to 100,000 distinct IP addresses daily, and (ii) trust anchor signals provided to ICANN by most of the root server operators from up to 200,000 distinct IP addresses daily; ICANN provided us with a subset of this data covering February 1st to March 29th, 2018.

4.3.2 Active measurements

Resolver state By using only data collected at the root, we miss the perspective of the client. To add this perspective, we rely on public measurements [100], that make use of the RIPE Atlas measurement network [69] (see for more details on RIPE Atlas Explainer 2). We send our measurements through the probes' recursive resolvers, pre-configured by the probe owner or learned through DHCP. This allows us to observe the transition from *KSK-2010* to *KSK-2017* (event **IV**) and the revocation

[1] In case of the root zone there is nothing to append. An example non-root zone trust anchor signal with appended zone is `_ta-4b61.dlv.isc.org`.

Query String	Which trust anchor(s)?
`_ta-4a5c`	Only *KSK-2010*
`_ta-4a5c-4f66`	Both *KSK-2010* and *KSK-2017*
`_ta-3039`	Has a non-IANA trust anchor
`_ta-4a5c-4f66-8235`	*KSK-2010* & *-2017* and a non-IANA trust anchor

Table 4.1: Root zone RFC 8145 trust anchor signals

DNS response code		State
Valid Signature	Bogus Signature	
`NOERROR`	`NOERROR`	insecure
`NOERROR`	`SERVFAIL`	secure
`SERVFAIL`	`other`	bogus

Table 4.2: Combination of response codes indicating the state of the measured resolver

of *KSK-2010* (event **V**) from the perspective of resolvers and measure whether they continue to validate DNSSEC signatures successfully. The public measurements we leverage consist of two queries sent every hour and check whether resolvers validate correctly. The first query asks for the `A` record of a domain with a valid signature, the second for a domain with a bogus signature. The response codes of both measurements can be combined (see Table 4.2) to establish if a resolver (i) does not validate DNSSEC signatures (state *insecure*), (ii) validates signatures correctly (state *secure*) or (iii) fails to validate (state *bogus*). Secure resolvers changing state to *insecure* or *bogus* at any stage of the rollover may be indicative of that resolver experiencing problems. In addition to the public measurements, we schedule our own measurement which queries each resolver for the `DNSKEY` RRset of the root, to measure uptake of *KSK-2017* during the rollover.

Using 10,004 RIPE Atlas probes (all probes available at the time of our measurement) and their recursive resolvers gives 18,277 vantage points (VPs), located in 3,647 autonomous systems (ASs). To find how many resolvers these VPs cover, we send hourly queries for a domain under our control, using the probe ID and a random string as a sub-label to avoid caching. Our authoritative name server responds with the IP address of the resolver that served the query. Using this method, we observe 35,719 upstream IPs located in 3,141 ASs over the period in which we conducted the measurement.

Root sentinel As discussed, RFC 8145 allows resolvers to signal which trust anchors it uses to upstream authoritative name servers. What was lacking, however, is a way for resolver users and other third parties to actively ask resolvers which trust anchors they use. This led to the introduction of RFC 8509, the so-called "Root

Query String	Is <KEY-TAG> a trust anchor?	
	Yes	No
root-key-sentinel-is-ta-<KEY-TAG>	Valid response	SERVFAIL
root-key-sentinel-not-ta-<KEY-TAG>	SERVFAIL	Valid response

Table 4.3: RFC 8509 Root Sentinel queries

Sentinel" [132]. Given that the specification was only finalized in December 2018, it could not reliably be used to monitor the root KSK rollover (although we do observe early implementations). We do, however, include Root Sentinel measurements to study adoption of this new form of telemetry and to observe the revocation of *KSK-2010* in 2019 from the perspective of resolvers.

The Root Sentinel is an *active* measurement mechanism. A client can send two special queries to resolvers to ask what trust anchors they currently have to validate DNSSEC responses. The first query type allows a client to ask if a DNSKEY with a certain key tag *is* a trust anchor, the second type allows a client to ask the inverse (whether a specific DNSKEY *is not* a trust anchor). The resolver returns a valid response to the first type if the specified key is a trust anchor, and a SERVFAIL error if it is not. For the second query type, the opposite behavior applies. Table 4.3 shows what the queries look like. Note, while RFC 8145 uses hexadecimally encoded key tags, RFC 8509 uses decimal key tags. Thus, to query for the presence of *KSK-2010* and *KSK-2017*, ...-is-ta-19036 and ...-is-ta-20326 are used.

Our goal is to examine (i) how many resolvers support Root Sentinel queries, and for those that do, (ii) if they correctly have the new key (*KSK-2017*) and remove the old key (*KSK-2010*) when it is revoked (event **V**). To do so, we set up a domain under our control. The name server for this domain is configured to return a DNSSEC-signed A record for Root Sentinel queries. We then use RIPE Atlas to issue four Root Sentinel queries (i.e., each of the two Root Sentinel queries for the old and new key) under our test domain. For this measurement, we extended our coverage of the global resolver population by including additional measurements using the Luminati proxy network [133]. This gives us more visibility in residential networks. Luminati is a paid HTTP/S proxy service enabling clients to route traffic via the Hola Unblocker Network. Luminati currently provides over 187 million potential exit nodes. When receiving an HTTP request, exit nodes send a DNS request to *their resolver* and then issue the HTTP/S request. This allows us to measure resolver behavior. For more details on using Luminati for network and DNS measurements, we refer to Chung et al. [134], [135].

4.3.3 Ethical Considerations

The measurement data collected at the root of the DNS consists of aggregate data (RSSAC002), telemetry signals (RFC 8145), DNSKEY queries and aggregates of popular queries for telemetry sources identified as showing non-standard behavior. Only in rare cases do we identify specific resolver operators (not end users) so we can

contact them in order to gain an understanding of unexpected resolver behavior (Section 4.4.3).

Most of our active measurements leverage well-established public measurement platforms, such as RIPE Atlas, where strict guidelines exist. The exception to this are our Luminati measurements. To use the Luminati service, we first note that we paid the operators of Luminati for access, and strictly follow their License Agreement [136]. The owners of exit nodes agreed to route Luminati traffic through their hosts. Furthermore, we took great care to ensure that all traffic only flowed toward domains under the authors' control, which serve empty web pages. Given that we are only interested in information about the RFC 8509 behavior of DNS resolvers, we discard any end user IP addresses from our logs.

4.4 ANALYSIS

The next sections discuss the most relevant events of the rollover (**I** – **VI** in Figure 4.1), starting before the rollover (**I** – **III**) in Section 4.4.1, followed by the rollover itself (**IV**) in Section 4.4.2 and ending after the rollover (**V** – **VI**) in Section 4.4.3.

4.4.1 Before the Roll

Early RFC 8145 data

RFC 8145, published April 2017, was quickly adopted by open source resolver implementers. BIND supports it from mid-2016 with the functionality enabled by default, Unbound since April 2017, enabling it by default in October 2017, and Knot since November 2017, again enabled by default.

We began looking for evidence of RFC 8145 signals in A/J Root data from May 2017. By September 2017 we see trust anchor signals from approximately 1,300 unique source IPs per day. Figure 4.3 shows these early trust anchor signals. The *KSK-2010* line shows what fraction of RFC 8145 sources sends signals for the old trust anchor, and the *KSK-2017* line shows signals for the new trust anchor. Note that these signals are independent; in other words: a single source may send signals for both *KSK-2010* and *KSK-2017*.

As Figure 4.3 shows, initially almost all sources had only *KSK-2010*. There is some slight increase in uptake of *KSK-2017* starting in June, before *KSK-2017* was published in the root zone. This increase can be explained by installations that received the new trust anchor as part of a software update, or from those where an administrator manually added it. ISC, for example, added the new key to BIND's code repository on the same day it was made operational and published by IANA (February 2nd, 2017).

When *KSK-2017* is published in the root zone on July 11th, 2017, validators that implement RFC 5011 begin the process of accepting the new key. After seeing the key published (and correctly signed) for 30 continuous days (the RFC 5011 *Add Hold-Down Time*), a validator adds the new key to its trust anchor set. Thus, from August 10th, we observe a rapid rise in signalers reporting *KSK-2017* over the two days after

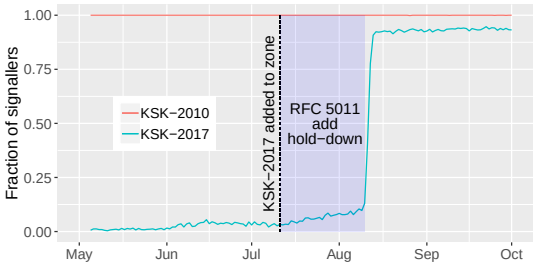


Figure 4.3: Early RFC 8145 trust anchor signals (2017)

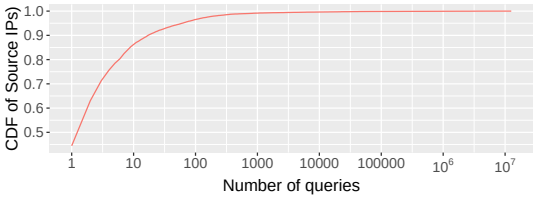


Figure 4.4: CDF of addresses vs. queries in B Root data sending only *KSK-2010* signals

the hold-down period ends. Because the TTL of the `DNSKEY` record set is 48 hours, the shift is not immediate.

After the 30-day hold-down ends, some 8% of signalers still do not report having *KSK-2017*. Operators watching this data hoped this population would continue to shrink. However, it remained at this level through the end of September. This is the primary reason why, on September 27th 2017, ICANN made the difficult decision to postpone the rollover [114]. As late as August 2019, around 1% of signalers still report only having *KSK-2010*.

Unusual *KSK-2010* RFC 8145 signalers

During continued monitoring of the RFC 8145 signals, ICANN began observing two unusual artifacts: (i) a large fraction of resolvers failed to pick up and trust *KSK-2017*, as measured by resolvers sending only RFC 8145 *KSK-2010* signals and seen in Figure 4.5, and (ii) many of the data points came from IP addresses sending only small numbers of queries, as seen in Figure 4.4. Note that the fraction of resolvers not trusting *KSK-2017* actually got worse, not better, between the end of Figure 4.3 and the beginning of Figure 4.5. These artifacts led to the question *"Why do so many new addresses appear that send RFC 8145 signals indicating they only trust* KSK-2010*?"*

To answer this question, we compare the RFC 8145 signal data from ICANN to all DNS queries arriving at B Root over a four week period from March 1st–29th, 2018. We focus this analysis on B Root, because unlike the data from ICANN which only contains RFC 8145 signals, for B Root we have full access to all queries received.

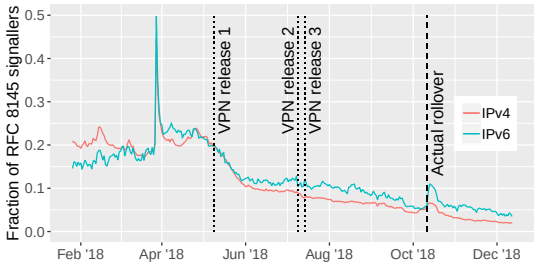


Figure 4.5: Addresses signaling only *KSK-2010*

	Description	Count
A	Unique sources in ICANN data	1,206,840
B	Sources from **A** signaling *KSK-2010*	508,533
C	Sources from **B** sending only one signal	310,839
D	Unique Sources in ICANN data to B Root	309,140
E	Sources from **D** signaling *KSK-2010*	113,467
F	Sources from **E** signaling just once	16,403
G	Sources from **F** sending 1-9 queries	6,702

Table 4.4: Narrowing the observed data

We narrow the data to those addresses that behave unexpectedly: they send *a single signal* for *KSK-2010* to B Root, and send *only 1–9 other queries* to B Root in the period covered. The narrowing down of the full list of IP addresses ICANN observed to just these anomalously behaving addresses is shown in Table 4.4.

To test if there is any commonality in other query names sent by these sources, we extract and correlate the top query names sent by these addresses (shown in Table 4.5). Beyond the RFC 8145 signals ("`_ta-4a5c`") and queries for root-zone data ("`.`" (period)), the next highest two requested names are a Virtual Private Network (VPN) provider's primary and secondary domain (anonymized in Table 4.5). This commonality in top queries strongly indicates the discovery of a likely cause of *KSK-2010* signals from sources sending otherwise low-volume traffic. Searching the VPN provider's software, taken from their Android release, revealed an embed-

Query-Name	Count
`_ta-4a5c`	15,447
`.`	9,182
`VPN-PROVIDER.com`	3,156
`VPN-PROVIDER-ALTERNATE.com`	415
`_sip._udp.OTHER-DOMAIN.com`	86

Table 4.5: Top query names from anomalous sources

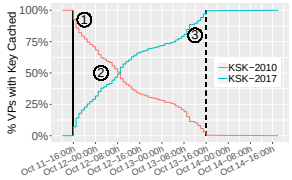


Figure 4.6: Key transition for all VPs

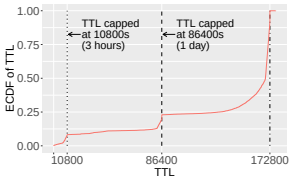


Figure 4.7: Reported DNSKEY TTL

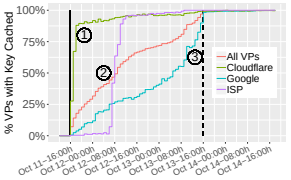


Figure 4.8: *KSK-2017* on large resolvers

ded "`root.key`" file containing just *KSK-2010* and not *KSK-2017*. The embedded libraries found in the software also revealed a library name matching the Unbound project [137], a popular DNSSEC-validating resolver.

We contacted the VPN provider on April 17th, 2018. They confirmed our findings and indicated that multiple products were affected. Subsequently, they released updated versions of their product to address the issue, as marked in Figure 4.5. The desktop software update had the most dramatic impact, significantly decreasing the number of *KSK-2010* signals seen at the root. The first mobile update with the new key set also showed a small dip in *KSK-2010* signals, though the second mobile update exhibited a less visible impact.

Key takeaway before the roll A single application can significantly influence trust anchor signaling, and the fact that it was an end-user application is largely responsible for the high number of signals. Given that DNSSEC validation in end-user applications will become more common in the future, this needs to be considered for future rollovers.

4.4.2 During the Roll

As *KSK-2010* signals returned to the 8% range by mid-2018, ICANN revised its plans for the rollover [138]. After community feedback on these plans, ICANN proceeded with the rollover [139]. On October 11th, 2018, at 16:00h UTC the KSK is rolled (event **IV**). From then on, root servers return a DNSKEY RRset signed with *KSK-2017*. In this section we show how resolvers picked up the new RRset. We then examine what happens to resolvers that do not have *KSK-2017* as a trust anchor, and how operators solve the problems this causes.

The key transition

To measure the transition from the old to the new RRset, we use RIPE Atlas probes (see Section 4.3.2) to send DNSKEY queries and then analyzed the results. Figure 4.6 shows when resolvers drop the old RRset from their cache and query the root for

the new one. [2] Right after the new key is published, resolvers begin showing cached signatures from *KSK-2017*. Within the first hour 7% of the resolvers have the new RRset. Sixteen hours later over 50% of resolvers have the new RRset. At 48 hours after the roll, the old RRset should have been removed from the caches of all resolvers; 99.5% of our vantage points return *KSK-2017* signatures at that point. After 11 more days, the last "lagging" vantage points pick up the new RRset (not shown in Figure 4.6).

Because the root DNSKEY RRset has a TTL of 48 hours, we expected half of vantage points to have the new RRset after 24 hours. As Figure 4.6 shows, however, this point is already reached after just 16 hours. In Figure 4.7 we plot the TTLs for the root DNSKEY RRset as reported by each vantage point when it receives the new RRset for the first time. More than 20% of vantage points report a TTL that is lower than 1 day, and around 10% even report a TTL lower than three hours. This indicates that some resolvers cut the TTL to a value lower than 48 hours, also explaining why the new RRset was picked up earlier than expected.[3] What this also means is that had a failure occurred during the rollover, we would likely have seen this sooner than intuitively expected, which is important to consider for future rollovers.

Another thing that stands out in Figure 4.6, are sudden "jumps" in the adoption of *KSK-2017* (marked ①–③). We correlate these jumps with adoption at resolvers often used by RIPE Atlas probes in Figure 4.8. The jumps respectively correspond to adoption of the new RRset by Cloudflare (①), a German ISP (②) and Google (③). Operators of the Cloudflare resolvers publicly commented that someone used their web interface to purge the DNSKEY RRset of the root from the cache right after the rollover [141]. This explains why the resolvers fetched the new RRset soon after the roll. This spurred us to check if other operators *purposely flushed their caches* before or after the rollover to either keep the old status for as long as possible, or force the new situation as soon as possible. To find evidence, we looked for vantage points that report a TTL close to 48 hours just before or after the rollover. We find three resolvers that fetched the keyset just before the roll (effectively locking in the old situation for almost 48 hours). A large European ISP privately confirmed they did this to avoid problems right after the rollover, allowing them to monitor the news from other operators after the roll [142].

Impact on validating resolvers

Now that we know how resolvers picked up the new RRset, we check if they experience any problems once they have the new RRset. For resolvers that do experience problems, we expect them to either fail validating signatures (become *bogus*) or turn off validation altogether (become *insecure*). We use RIPE Atlas measurements (see Section 4.3.2) to identify resolvers that were continuously *secure* 88 hours before the roll but turned *bogus* or *insecure* at any point within 56 hours after the roll.

[2] We published updates of this figure on social media and on the website of NLnet Labs to give the community insight into the progress of the roll.

[3] E.g., Unbound caches RRsets for a maximum of 24 hours by default [140].

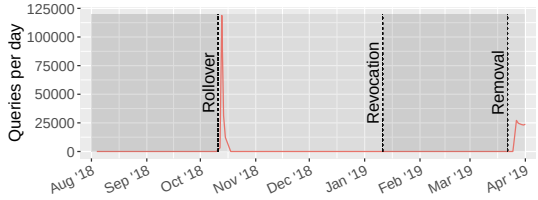


Figure 4.9: DNSKEY queries from ISP "EIR" to A/J Root

We summarize resolver behavior observed through RIPE Atlas in Table 4.6. Row **A** shows the total number of resolvers observed during the rollover. Of these, 1,717 (**B**+**C**) always validate signatures correctly before the roll but 970 (2.7%) turn *bogus* and 747 (2.1%) *insecure* some time after. We check how often problematic resolvers query for the DNSKEY of the root, using DNS-OARC DITL data collected during the rollover (see Section 4.3.1). If a resolver changes state *and* sends more DNSKEY-queries, we conclude that this change is caused by problems with the rollover. We see DNSKEY queries from 519 sources at the root (**D**). Of these, 509 (**E**) send more DNSKEY queries after than before the roll. For 359 resolvers, the increase in DNSKEY queries exceeds 1.5 times (**F**). The majority, 342 resolvers (**G**), return to their normal DNSKEY query pattern within an hour. We assume operators intervened and fixed these resolvers. For 138 resolvers (**H**) we keep observing unusually high numbers of DNSKEY queries for over an hour. They only return to their normal behavior after a median of more than 39 hours. Only three resolvers (**I**) continue sending unusually high numbers of queries throughout the entire measurement period. The fact that more than 60% of the resolvers get fixed within one hour is a strong sign that resolvers in our data set are used actively and that operators noticed issues during the rollover relatively quickly. We discuss resolvers that send excessive numbers of DNSKEY queries in more detail in Section 4.4.3.

	Upstream Resolvers	Count
A	Unique sources in RIPE Atlas data	35,719
B	↳ from **A** always secure before and bogus after	970
C	↳ from **A** always secure before and insecure after	747
D	↳ from **B** and **C** sending DNSKEY queries	519
E	↳ from **D** reach maximum DNSKEY queries after	509
F	↳ from **E** w. 1.5× DNSKEY queries after	359
G	↳ from **F** fixed within 1h	218
H	↳ from **F** fixed after 1h	138
I	↳ from **F** that did not get fixed	3

Table 4.6: Data of RIPE Atlas measurements

The user's perspective

From the analysis above, we cannot gauge the actual impact on end users. During our measurements, 175 RIPE Atlas probes (1% of all vantage points) relied exclusively on one of the *bogus* resolvers (set **B** in Table 4.6), thus were not able to receive any valid response at some point after the rollover. More than 70% of these probes, however, suffered problems only an hour or less. 166 probes could rely on at least one other resolver to serve their queries and were not affected by the failing resolver.

Other work [143] shows users move to public DNS providers in case of issues with the resolver of their ISP. Therefore, we analyzed if vantage points change to the public resolvers of Google, Cloudflare or OpenDNS. We found only two vantage points. One of these used the resolver of the Irish ISP EIR. This ISP experienced a well-publicized DNS outage [144] during the rollover, and the DNS community speculated this outage was caused by EIR's resolvers failing validation. Using the RIPE Atlas measurements, we identify the IP addresses of EIR's resolvers. Then, we count how many `DNSKEY` queries these resolvers send to A/J Root per day (see Figure 4.9). Starting from October 12th, queries increase, reaching a peak one day after the roll and returning to normal after 3 days. Keeping in mind that RIPE Atlas probes actively switched resolvers at the same time, this is a strong sign that the outage of EIR was indeed caused by validation errors. Note, Figure 4.9 shows the number of `DNSKEY` queries from EIR rising again after removal of *KSK-2010*. We discuss this increase Section 4.4.3.

Key takeaways during the roll We observed few resolvers with serious problems. Where such problems occurred, they were solved promptly by operators. Less than 0.01% of the resolvers we monitored during the rollover experienced problems that lasted beyond our observation window.

4.4.3 After the Roll

We now discuss what happened after the rollover, from the point when all resolvers should have a `DNSKEY` RRset signed by *KSK-2017*, to the removal of *KSK-2010* from the root zone.

Revocation of *KSK-2010*

As discussed in Section 4.3.2, the Root Sentinel standard (RFC 8509) was published too late to be useful for the actual rollover. We can, however, study revocation of *KSK-2010* with resolvers that adopted this protocol. Using all RIPE Atlas probes, we send out Root Sentinel queries from August 2018 to August 2019. Figure 4.10 shows the Root Sentinel signals observed over this period. As the figure shows, overall, the number of resolvers supporting Root Sentinel queries steadily increases to 2,419 resolvers in 720 ASs by the middle of August 2019. This is encouraging given the early stage of deployment of the protocol. After the revocation of the old key (event **V**), the number of resolvers with *KSK-2010* drops to almost zero while the number of

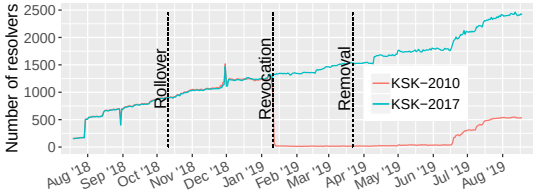


Figure 4.10: Root Sentinel observations with RIPE Atlas

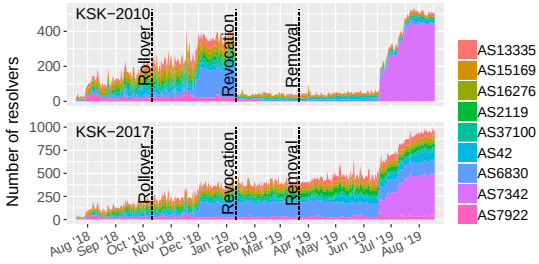


Figure 4.11: Top 9 ASs supporting Root Sentinel queries observed through RIPE Atlas

resolvers with *KSK-2017* keeps increasing. Interestingly, some 20 resolvers continue to signal having *KSK-2010* in their trust anchor store. This implies either a manually configured trust anchor, or a failure in their RFC 5011 processing. Then, from the middle of June 2019, *KSK-2010* starts making a surprising comeback. We explain why further down in Section 4.4.3.

As RIPE Atlas provides a limited view, we also used Luminati to measure a total of 52,378 resolvers serving 589,928 exit nodes — from 210 countries and 7,867 ASs — over a period of 14 days from March 28th 2019. From these, we select resolvers on which we were able to test all four combinations of Root Sentinel queries (Table 4.3). This leaves 21,563 resolvers, to which 385,520 exit nodes sent queries at least once. We further split these into resolvers that support Root Sentinel queries and ones that do not.[4] We finally determine which trust anchor(s) resolvers that support the Root Sentinel signal as present in their trust store. The vast majority — 21,056 (97.63%) resolvers from 5,311 ASs — do not support RFC 8509. These resolvers cover 330,891 (85.8%) exit nodes. Only 468 (2.2%) resolvers from 164 ASs support Root Sentinel queries and have only *KSK-2017*; these resolvers cover 33,266 (8.6%) exit nodes indicating that a few large ASs support RFC 8509 queries, including Telenor (Norway), Bezeq (Israel) and Meo (South Africa). We also note that 39 resolvers (0.19%) still signal they have *KSK-2010* configured.

Finally, we compare our observations through RIPE Atlas and Luminati. Fig-

[4]Note: a resolver that supports RFC 8509 correctly will return a valid response to *only one* of the two queries with the same key tag.

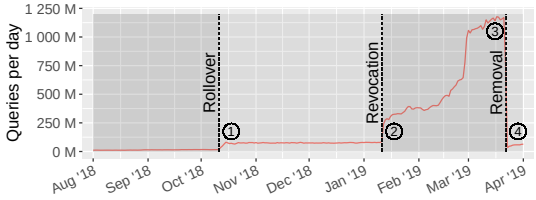


Figure 4.12: DNSKEY queries to A/J Root after the rollover

ure 4.11 shows the top 9 ASs with resolvers supporting RFC 8509 in our RIPE Atlas measurements. Comparing this to Luminati, we find that 43 resolvers from AS2119 (Telenor), 10 from AS16276 (OVH), 10 from AS6830 (Liberty Global), and 2 from AS7922 (Comcast), are observed in the same state through both RIPE Atlas and Luminati. Figure 4.11 also shows a surprising increase of *KSK-2010* from June 2019, we explain why in Figure 4.4.3.

Increase in DNSKEY queries

As mentioned at the end of Section 4.4.2, we observed an increase in DNSKEY queries from certain resolvers at various stages of the roll. We analyse this phenomenon in more detail here, especially because of the sharp increase in queries after the revocation of *KSK-2010* to the extent that at some point a worrying amount — up to 10% — of traffic to the root consisted of DNSKEY queries.

We start by analyzing the total amount of DNSKEY queries to the root. DNSSEC validators must regularly verify their locally configured trust anchor(s) against the zone's published DNSKEY records. In other words: validators periodically issue DNSKEY queries for the root zone. Due to the retry behavior of implementations, a validator with an out-of-date trust anchor is likely to send more than the normal amount of DNSKEY queries. This behavior was already observed in 2009 — before the root zone was signed — during a KSK rollover for an `in-addr.arpa` zone operated by RIPE. The group investigating that incident called it "rollover and die" [145].

Just after the root KSK rollover on October 11th, 2018, root name servers observed an increase in DNSKEY queries. Figure 4.12 shows the query rate for A/J Root. The increase was gradual, ramping up over the course of two days as the DNSKEY RRset timed out from resolver caches. Pre-rollover the rate was around 15 million queries per day. Post-rollover it increased five-fold, to 75 million (①). An even more dramatic increase occurred when *KSK-2010* was revoked (Event **V** in Figure 4.1). Immediately after the revocation, A/J Root see a sudden spike in DNSKEY queries (②), jumping from 75 million to over 200 million queries per day within 24 hours. The DNSKEY query rate continued to climb over the following weeks and months, exceeding one billion per day in March 2019 (③). At this point, DNSKEY queries comprised 7% of the total traffic received at A/J Root. The final phase of the rollover sees *KSK-2010* removed from the root zone on March 22nd, 2019. To everyone's surprise, the DNSKEY query rate dropped dramatically immediately after *KSK-2010* was removed.

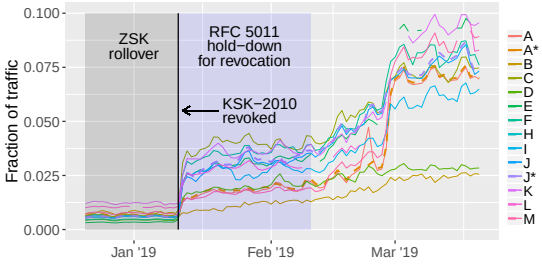


Figure 4.13: DNSKEY query increases for all root servers

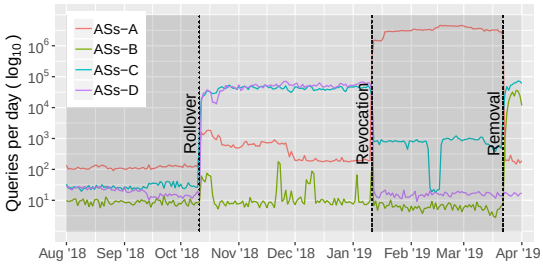


Figure 4.14: AS DNSKEY query patterns to A/J Root

As Figure 4.12 shows (④), the rate dropped and slowly crept back up to post-rollover levels as seen in October, November, and December 2018.

Figure 4.12 only shows data for A/J Root. To confirm similar increases at other root servers, we use the RSSAC002 data (see Section 4.3.1). The RSSAC002 data does not have a dataset specifically identifying DNSKEY queries, however we can infer the presence of such queries by examining the response size dataset. Figure 4.13 shows the percent of responses between 1232–1472 bytes as solid lines. The dashed lines — marked A* and J* — are actual A/J Root traffic and show a strong correlation. Not all root servers saw the same increase in queries, but we currently lack sufficient information to explain this.

Deeper inspection of the A/J Root traffic shows vastly differing DNSKEY query patterns on a per AS basis. Figure 4.14 shows the average of multiple ASs whose DNSKEY queries exhibit distinct patterns at different times throughout the rollover. Some ASs expressed a systemic trend of increased DNSKEY queries post-rollover and even higher rates post-revocation (ASs-A). Other ASs only exhibited an increase in DNSKEY queries after the removal of *KSK-2010* (ASs-B). Likewise, some ASs show increased rates post-rollover until revocation (ASs-D) and again after removal (ASs-C). To better profile these resolvers, we issued `version.bind` queries to IP addresses expressing the various behaviors. While the response rate was low (4.3% of ±18K resolvers), the majority returned older versions of BIND (45% BIND 9.9.x, 34% BIND 9.8.x, and 13% BIND 9.10.x).

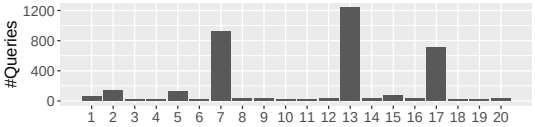


Figure 4.15: DNSKEY queries for root during experiments

Explaining the increase in DNSKEY queries To find the cause of the increased query rates, we studied traffic coming from individual, high-volume sources. Outreach efforts at a global DNS scale are challenging, but we were able to contact multiple operators willing to help diagnose the DNSKEY query increase. One operator (a large French cloud hoster), stated their servers were running BIND 9.8.2 on CentOS 6.7 and the logs contained large numbers of validation errors. Another set of sources identified as sending excessive DNSKEY queries to the root, came from 8 addresses in a single subnet at a large midwestern university. Their staff quickly identified a DNS lab exercise that had been left running inside virtual machines (VMs). After shutting down the VMs, we confirmed that the excess DNSKEY traffic had stopped. From the university's class instructions, we hypothesized that the DNSKEY query spikes were the result of ISC's BIND software running in a specific state: (i) the DNSSEC `managed keys` did not contain *KSK-2017*, but did contain *KSK-2010*; (ii) the `dnssec-enable` flag was set to `false`; and (iii) the `dnssec-validation` flag was unset, leaving it in its default state of `yes`.

To verify this hypothesis, we performed experiments to test for bugs related to BIND's behavior in the absence of a valid trust anchor. We set up a *BIND 9.11.5-P4* resolver (the oldest supported release at the time), configuring it as per the university's class instructions. We also ensured that BIND's managed keys file contained only *KSK-2010*. Then, we ran 20 experiments in which we started a fresh copy of BIND configured as specified above. In each run, we sent ten sets of queries to BIND for test domains in seven TLDs at 30-second intervals, recording DNSKEY queries sent by the resolver, along with timestamps. Figure 4.15 shows the results. Each experiment start time was normalized to zero and overlayed in Figure 4.16, showing highly variable query patterns in each run (note experiments 7, 13 and 17).

Both plots show wide variations in behavior of the resolver under test. At times it behaves as expected, sending only a few DNSKEY queries after initializing. At other times, the resolver seems stuck in a state where every incoming request causes the resolver to send out a flurry of DNSKEY queries.

From the analysis of events **V** and **VI**, and the corresponding DNSKEY loads seen at the root (Figure 4.12 and Figure 4.13) we conclude there are likely two different bugs causing the increase in queries. One bug is likely the cause of the increase in DNSKEY queries shortly after the rollover (event **IV**) and after *KSK-2010* is removed (event **VI**). Another bug is likely the cause of the extreme query loads seen in 4.13, when *KSK-2010* was present but with the revoke bit set. We have reached out to the developers of BIND to confirm our hypotheses, but have not received any feedback as of September 13th, 2019. What remains unclear is why operators have not noticed

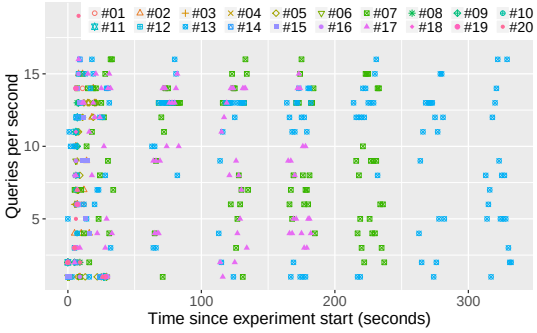


Figure 4.16: Time-normalized graph of experiments

this broken resolver behavior, as we expect these resolvers to return SERVFAIL errors to every query. We speculate only one resolver in a group is failing, with an alternate succeeding on behalf of their clients. This behavior is a well-known fact from other work [146].

To facilitate reproducibility, we published experiment configurations and scripts in a public GitHub repository [147].

Increased response size

Another potential risk during the rollover, identified in the 2016 Rollover Design Team report [112], was the increase in size of the DNSKEY RRset (see Section 4.2). When *KSK-2010* was revoked, this size reached its maximum value of 1,425 bytes. We analyzed if this increase hindered resolvers fetching the record set and, as a result, caused validation errors. While there are other moments during the rollover at which the response size is significantly higher than usual, we focus on the revocation event since that is when the maximum size was reached.

The first sign we expected to see if resolvers experience problems is an increase in fallback to TCP. We studied the RSSAC002 data concerning traffic types, and found no evidence of such an increase during revocation. Note, however, this data does not contain information on individual query types such as DNSKEY. If resolvers are also unable to fall back to TCP, then they may become unable to fetch the DNSKEY RRset altogether. We use the measurements from RIPE Atlas to detect whether any vantage points were unable to retrieve the DNSKEY RRset from the root after the increase in size. Resolvers are marked as unable to retrieve the DNSKEY RRset if they cannot fetch the RRset within 5 seconds.

Out of 17,925 vantage points, 1,975 (11%) are able to fetch the DNSKEY RRset before revocation, but fail to fetch it at least once 48 hours after the revocation. Only 67 of these (0.4%) never manage to fetch the key set after the revocation. Even though the IPv6 minimum MTU is 1,280 bytes, vantage points that contact resolvers via IPv6 did not fail more often than those using IPv4. We also found no resolvers that turned *bogus* after the revocation. This leads us to conclude that the increased

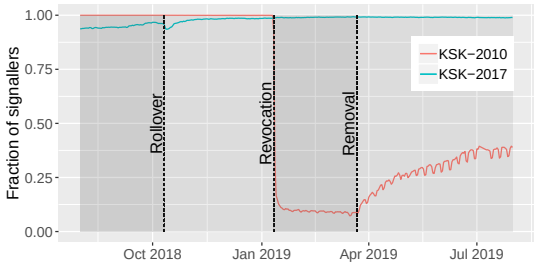


Figure 4.17: RFC 8145 signals August 2018 to August 2019

response size during revocation only caused problems for a few resolvers and did not impact validators. This was also expected by the KSK rollover design team [112].

The return of *KSK-2010*

We end this section with a surprising comeback. As mentioned in Section 4.4.3, the number of resolvers that signal support for *KSK-2010* is on the rise again since its removal from the root zone DNSKEY RRset. This increase is also visible in the RFC 8145 signals sent to root servers. Figure 4.17 shows that by the end of July 2019 almost 39% of signalers again report having *KSK-2010* in their trust anchor set. This, of course, raises the question why a retired trust anchor is making this comeback. While it is impossible to attribute the observed rise to a single source, we have convincing evidence of the most likely cause: DNS resolver software shipping with built-in or pre-configured trust anchors.

First, we note that the current long-term supported version of Ubuntu (18.04 LTS) ships with Unbound version 1.6.7, which supports RFC 8145. In addition, Ubuntu also includes a pre-configured trust anchor package that includes both *KSK-2010* and *KSK-2017*, and enables DNSSEC validation by default. We verified that, upon startup, Unbound loads both trust anchors, marks *KSK-2010* as "missing", but as the trust anchor is still configured, Unbound signals its presence in its RFC 8145 telemetry. Any installation of Ubuntu 18.04 LTS with Unbound that was running for at least 30 days[5] when *KSK-2010* was published as revoked will have cleaned up the old trust anchor. However, any installation (or re-installation) after February 20, 2019 could not complete RFC 5011 revocation and retained *KSK-2010* as a trust anchor. We also verified the behavior of another popular open source DNS resolver implementation on the same OS. Ubuntu 18.04 LTS ships with BIND version 9.11.3, which includes both *KSK-2010* and *KSK-2017* as built-in trust anchors. By default, the Ubuntu package for BIND is configured to perform DNSSEC validation using the built-in trust anchors. Upon startup, however, if BIND does not find a configured trust anchor in the DNSKEY RRset returned by the root servers, it will not signal this trust anchor in its RFC 8145 telemetry. This does not mean, however that the trust

[5]The RFC 5011 *Remove Hold-down Time*.

anchor is removed. We verified that BIND retains *KSK-2010* in its trust anchor file on disk, so if the key were ever to return in the root DNSKEY RRset we expect BIND to accept it as a valid trust anchor again.

Second, as mentioned previously, Figure 4.11 shows an increase in *KSK-2010* beginning in the middle of June 2019 from a single network, AS7342. As it happens, this is the origin AS for Verisign's public DNS service.[6] The rise in *KSK-2010* signalers corresponds to an upgrade of the software used on the public DNS resolver. The newly deployed version supports the Root Sentinel (RFC 8509) and is packaged with a configuration that includes both *KSK-2010* and *KSK-2017* as trust anchors.

The two examples above explain most of the return of *KSK-2010* in Figure 4.11 and at least some of the return in Figure 4.17. They are illustrative of software still shipping with *KSK-2010* as trust anchor. This does not mean that these are the only examples, though, there are likely other packages with similar behavior. One question we have not discussed yet is whether the comeback of *KSK-2010* can be considered problematic. We discuss this in more detail in Section 4.6.

Key takeways after the roll The biggest problem during the whole process, arguably, occurred after the roll with the significant increase in DNSKEY queries. This problem was not foreseen in the design report [112], underlining the importance of independent studies of such major events on the Internet and confirming the need for meaningful telemetry. Additionally, it is clear trust anchor management is complex and that shipping trust anchors with software has long-lasting effects. We come back to this in Section 4.6.

4.5 RELATED WORK

As we discussed in the introduction, the root DNSSEC KSK rollover is a first-of-its-kind event. Thus, our discussion of related work will focus on earlier studies that have looked at the operation of the DNS root server system and the impact of DNSSEC on the performance of DNS resolvers. Huston [148] independently confirms our finding that the Irish ISP EIR suffered outages but does not provide a more thorough analysis.

The earliest work to study DNS traffic to root servers by Danzig et al. [149] dates back to 1992, five years after DNS was adopted as the Internet's naming system [150]. This study illustrates that software bugs that cause excessive traffic are a problem of all ages, as they find multiple bugs in algorithms meant to improve DNS resilience. In 2001, Brownlee et al. [151] study almost two weeks of traffic to F Root. Again, they find a surprising amount of problematic traffic to the root, with 14% of queries consisting of malformed address (A) queries. In 2003, Wessels et al. [152] studied 24 hours of F Root traffic and concluded an astonishing 98% of queries were malformed or unnecessary. Since 2006, DNS-OARC collects so-called Day-in-the-Life (DITL) datasets [127], which typically includes traffic to most root servers. In 2008, Castro

[6] https://www.verisign.com/en_US/security-services/public-dns/index.xhtml

et al. [128] analyzed three years of DITL data to characterize root server traffic and also found that 98% of queries were unnecessary.

Apart from studying traffic at the root, past work also looked at operational changes to the root system. A particularly impactful event is the change of the IP address of a root server. Since resolvers have to be configured *a priori* with the IP addresses of root servers to bootstrap DNS resolution, such events have a major impact. Many root servers have undergone such changes, and Lentz et al. [153] study one such change for D Root in an academic paper in 2013. This study concludes that such address changes take a long time to propagate to the global resolver population, with the old address still seeing significant amounts of traffic months after the change. The authors suggest that such IP address changes may actually be beneficial, as they serve as some form of a "garbage collection" for old implementations. A similar notion could be said to apply to rollovers of the root KSK. In 2015, Wessels et al. [154] show how the aftereffects of an address change linger, finding that the old IP address for J Root still receives on average 400 queries per second from some 130,000 sources *thirteen years* after the address change.

The effects of the root KSK rollover on resolvers studied in this paper are part of the impact of DNSSEC on resolvers. Earlier work studies other aspects of the impact of DNSSEC, including the performance impact of DNSSEC validation [155]–[158] and the risks, in terms of availability and security, of packet fragmentation of large DNSSEC responses [8], [121]. Even though [121] conclude that up to 10% of resolvers could have problems handling larger DNSSEC responses, we did not observe failures when the DNSKEY response size increased. Other popular DNSSEC signed zones have served records larger than 1,425 bytes and validating resolvers probably took measures to handle large responses already. Finally, the way DNSSEC is organized as a Public Key Infrastructure is highly relevant for the root KSK rollover studied in this paper. Yang et al. provide a detailed overview of why the DNSSEC PKI is organized the way it is today [159].

4.6 DISCUSSION AND RECOMMENDATIONS

Improving telemetry A key challenge faced during the KSK rollover was sparse and distorted telemetry from resolvers. Ideally, those responsible for the rollover would want to know both the exact state of resolvers (in terms of DNSSEC validation) and how important these resolvers are (in terms of the number of clients relying on them). This provides actionable intelligence that allows prioritization of "important" resolvers (serving millions of users).

Clearly, during the root KSK rollover discussed in this paper such comprehensive telemetry was not available. While RFC 8145 saw significant deployment before the rollover, it was difficult to interpret its signals. This was mostly due to four reasons: first, RFC 8145 only allows for passive observations by — in this case root — DNS operators. Thus, in case of problems, it is impossible to query resolvers for further state information. Second, there is no telemetry on the query volume a resolver processes, making it hard to judge how relevant or risky a resolver with problems is.

	RFC 8145	RFC 8509
Signaling	Automatic	Requires query
Which TAs are revealed	All configured	Only those queried
Supports non-root TAs	Yes	No
Collection method	Passive	Active
Vulnerable to manipulation	Yes	Only to on-path attackers

Table 4.7: Supported features of existing telemetry

Third, RFC 8145 may propagate through upstream systems (NATs, DNS forwarders, caches and other middle-boxes), leading to distorted signals and hiding systems with actual problems. Fourth, although we have not seen any evidence of tampering, an attacker could artificially inflate the number of resolvers that have not acquired the new key by spoofing RFC 8145 telemetry signals. Such an attack could adversely influence the decision-making process around whether or not to proceed with a planned rollover. Despite the limitations of RFC 8145, however, *without it* ICANN and the DNS community would have been completely blind and some problems were actually solved due to RFC 8145 telemetry.

The Root Sentinel (RFC 8509) addresses the first limitation of RFC 8145. It uses active measurements from the client perspective to establish the DNSSEC trust anchors configured on a resolver. While standardized too late to be of use during the current rollover, our analysis shows RFC 8509 is seeing rapid deployment and provides useful signals as of September 13$^{\text{th}}$, 2019. Nevertheless, RFC 8509 also suffers from the second and third limitations discussed for RFC 8145 albeit with different signal distortion (e.g. assuming a Root Sentinel query is sent to resolvers at a large ISP while it is actually handled by a local forwarder). Table 4.7 summarizes the supported features of the existing telemetry protocols.

Based on our analysis of the current rollover, we recommend exploring incremental improvements to both RFC 8145 and RFC 8509. The quality of such signaling would be greatly improved if it were possible to identify true signal sources, identify cases where signals are forwarded, and estimate the number of users being serviced. We recognize that there are serious concerns around such detailed signaling. Weighing the tradeoffs requires further thought and debate in the community.

Another issue compounding the difficulties of interpreting resolver validation problems is the ambiguity of the `SERVFAIL` error code validators send upon failure. Effectively only by combining results from different measurements (Table 4.2) can we be reasonably confident that a resolver has issues with DNSSEC validation. We therefore strongly support a draft under review in the IETF that proposes to send extended error codes for DNSSEC failures [160].

Introducing a standby key There is an ongoing debate in the DNS community about introducing a KSK standby key in the root zone by default [161]. Effectively, because the rollover was postponed by a year, this has already been tested for a

single standby key, without leading to issues with, e.g., response sizes. We therefore think it safe to introduce such a standby key as multiple community members have suggested. An immediate benefit of this is that resolvers are much more likely to pick up the new key if it is pre-published for a longer period. Given the rollover policy of the root [111], such a standby key could even be published years in advance.

Trust anchor distribution The 2018 KSK rollover was the first time a large population of DNSSEC validators needed to update their trust anchor. At the start of the process, the design team expected RFC 5011 to be the main means through which validators keep their trust anchors up to date [112]. Our observations suggest that where RFC 5011 was used, it generally worked as intended. In the few instances where problems did occur, this was either due to validators lacking permission to persist state to disk, or loss of state due to, e.g. container or virtual machine teardown and reinitialization. The latter issue has the potential to become a bigger problem moving forward, as the proliferation of container technologies was not envisioned when RFC 5011 was authored 11 years ago. Lastly, we are also beginning to see DNSSEC validation in end user applications (e.g. the VPN client from Section 4.4.1), often with hard-coded trust anchors (a search on GitHub yields thousands of examples of this). This raises the question if *in-band* updates through RFC 5011 remain the main means for trust anchor management going forward.

As noted earlier, some resolver implementations distribute trust anchors in their software packages (thus these get refreshed with software updates). While this works to some extent, it does not scale to encompass applications performing validation. Additionally, we observed that there may be significant delays when retiring trust anchors, as evidenced by the surprising comeback of *KSK-2010*.

Based on these results, we advocate that the preferred method to distribute trust anchors should be with operating systems *out-of-band*. Some distributions (e.g. Debian Linux) have already started doing so. Applications can then rely on the OS and we strongly urge against hard-coding of trust anchors. In addition to this, OS distributors should tightly manage these trust anchors when they are replaced. In Section 4.4.3, we ended with the question if the retention of the retired *KSK-2010* was problematic. On the face of it, the answer to this question is "No", since the key was retired according to a schedule, and all copies of the key have now been destroyed. Consider, however, two scenarios, one in which a key is revoked because it has been compromised, and one in which the algorithm for the key has been compromised. It is evident that a speedy retraction of such a key as a trust anchor is imperative, and it is also evident that the current practice we observed does not suffice. Given the inertia of solving this issue Internet-wide, we would recommend an additional security practice: if a key needs to be revoked, then the root `DNSKEY` RRset should include the revocation signal until there is a reasonable certainty that systems have been updated to remove the trust anchor. This practice guarantees that software that correctly implements RFC 5011 will not use the compromised key as a trust anchor.

Relevance for algorithm rollovers We have shown that, despite some problems, the root KSK rollover was a success. We can now ask ourselves: What are the implications of this finding for a future root algorithm rollover?

We argue that the root is also ready for an algorithm rollover. The process of rolling the algorithm is the same as for rolling the KSK, aside from two aspects.

First, it is not only important that validating resolvers replace their trust anchor but also that they support the new algorithm. Measurements, like the active measurements conducted in Section 3.7.2 of the previous chapter can help to estimate algorithm support. For example, if we would roll to ECDSAP256 today, then around 4% of resolvers in our data set would not be able to validate the signatures anymore, causing them to turn *insecure*. These resolvers would still be able to resolve domain names, but will not benefit from the protection provided by DNSSEC anymore. In the end, the question remains how many resolvers or users can be left behind when rolling to a new algorithm. This question is not trivial, and similar to questions raised before the root KSK rollover. Then, it was unclear how many resolvers must signal support for *KSK-2017* before continuing with the rollover. In the end it was up to ICANN and the community to find the right threshold. This will also be necessary when rolling to a new algorithm.

Second, algorithm rollovers might require two additional stages, which are not necessary during a KSK rollover. Before publishing a new key, the root zone operators first need to sign the zone with the new key, but publish only the signatures. This is necessary because some validating resolvers expect signatures for every signing algorithm for which a key is published. If the root zone operators would not publish these new signatures in advance, some validating resolvers would consider the root zone bogus. We describe the details of algorithm rollovers, and of this issue in particular in Section 5.3.2 of the next chapter. This additional stage slows down the rollover process slightly[7], but also increases the response size of responses from the root and the root zone overall[8]. Whereas the increased response size is likely not a problem with current algorithms like ECDSAP256, it might become a problem with quantum-safe algorithms, which we discuss in more detail in Chapter 7. These algorithms can have significantly larger signatures, which can cause problems during transmission as described in Section 4.2.

Besides these two aspects, an algorithm rollover would face the same problems as the next root KSK rollover. Thus, an algorithm rollover would benefit from the proposed measures in this section (improved telemetry, introduction of a standby key, central trust anchor distribution) as well.

4.7 CONCLUDING REMARKS

In this chapter we provide a comprehensive analysis of the very first DNSSEC Root KSK Rollover. We show the rollover did not pass without problems: hundreds of actively used resolvers failed to validate signatures at some point during the rollover.

[7] Depending on the TTL, and at least 48 hours in case of the root.

[8] Depending on the chosen algorithm, e.g. 64 bytes for every added ECDSAP256 signature.

Nevertheless, this is only a minute share of the total resolver population and most problems were fixed quickly. Additionally, thousands of resolvers exhibit anomalous behavior during the rollover process, though it remains unclear if this caused problems for end users. The significant traffic increase to root servers, seen after the revocation of *KSK-2010* requires attention from the DNS community with future rollovers in mind. We demonstrated that at least some of these queries can likely be attributed to bugs in resolver software.

We also demonstrate that telemetry, used to measure deployment of new keys, was significantly distorted by a single application (a VPN client). We analyzed a complementary protocol, which while potentially a valuable addition, still has drawbacks. Based on our experiences, we provide recommendations for incremental improvements to both protocols. In addition to this, we observe that trust anchor distribution — which the rollover design team expected to happen mostly *in-band* — requires attention for future rollovers, and provide recommendations for alternatives.

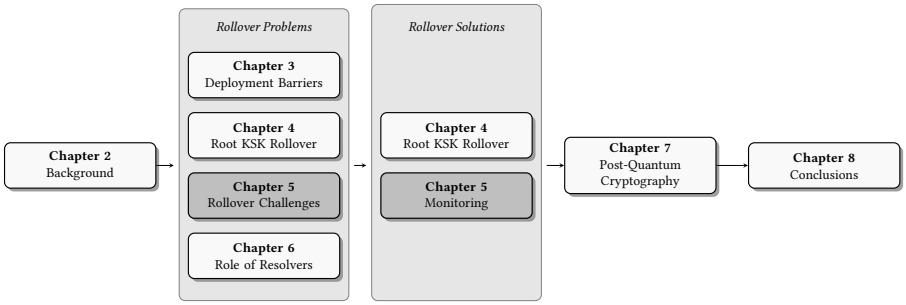
While, of course, our work focused heavily on anomalies, our analysis supports ICANN's conclusion that the rollover was indeed an overall success. As with earlier changes to the root system, some systems will fail and this study shows that the Root KSK rollover was no different. These failures, however, were limited to a very small set of resolvers and got fixed fast, limiting the impact. This gives us confidence that this first ever rollover certainly should not be the last.

Because of this success, we also argue that the root is, in principle, ready for an algorithm rollover. Nevertheless, some aspects, like additional timing considerations can complicate algorithm rollovers and additional questions, e.g regarding the readiness of resolvers, need to be answered.

In the next chapter, we dive deeper in the aspects that complicate algorithm rollovers, including the root, but also propose and assess measures to address them.

CHAPTER 5

The Complexity of Algorithm Rollovers



In Chapter 3, we have shown that the complexity of rollovers is one of the major problems when transitioning to a new algorithm. We have found that many operators shun away from algorithm rollovers. In this chapter, we study in more detail why algorithm rollovers are complex. Then, we propose measures to reduce the complexity. Concretely, we develop a measurement method that allows operators to spot issues during a rollover early and help them to time their rollover correctly. Our goal is to reduce the complexity of algorithm rollovers, addressing one of the major barriers when maintaining a secure DNS. We applied our monitoring method during the first algorithm rollover of the Swedish ccTLD `.se`*. The operators of* `.se` *confirmed that the insights gained through our measurements gave them confidence in their procedures during their rollover. After* `.se`*,* `.br` *(Brazil) and* `.dk` *(Denmark) applied our monitoring method as well. We published the study on which this chapter is based in a scientific journal [19] and a tool, allowing operators to monitor their own rollover, as open source [162]. Additionally, we presented our work to a broader operator community at a RIPE meeting [163].*

5.1 INTRODUCTION

We have shown in Chapter 3 that algorithm rollovers are a major barrier for transitioning to more secure algorithms at TLDs and second level domain names. We have observed that only a small number of domain names, that have already deployed DNSSEC, roll to a new algorithm. Also, in the rare cases that operators roll an algorithm it is often not carried out according to best practices.

If a rollover goes wrong, it can gravely impact the reachability of a domain and its children. Resolvers may fail validation and render the domain unreachable for hours. This has even happened to large zones, such as the Dutch country code top-level domain (ccTLD) *.nl* [164]. This incident not only affected the TLD itself but also the over 5 million domains that were registered under *.nl* at that time.

Timing issues during the rollover are one of the major reasons for failures. DNS resolvers cache records to reduce response times. This makes it hard for operators to know which information is held by resolvers and when it is safe to add or withdraw keys. Best common practices give guidelines at which stage to introduce and withdraw keys and signatures but do not give strong recommendations and leave it entirely up to the operators to make the right decisions at the right wall clock time. Thus, operators that perform a rollover want to know: *(i) when is it safe to add new keys and signatures and withdraw old ones?* and *(ii) is my zone secure at all times during a rollover?*

Our contributions in this chapter are threefold: **(i)** We propose a new measurement technique with which operators can answer both questions above so they can roll their keys with confidence. They know when it is safe to add and withdraw the keys and can monitor every stage of the rollover from the perspective of their clients. Thereby, we mitigate one of the biggest dangers during rollovers, in turn reducing one of the barriers to transition to a new algorithm, and thus increasing the overall security of the DNS on the long term. Further, **(ii)** we carry out comprehensive measurements of the most complex type of rollover: a live algorithm rollover on a production zone, specifically the *.se* ccTLD. These measurements were performed upon request of, and in collaboration with the *.se* ccTLD operator IIS [165]. The *.se* ccTLD has a high DNSSEC penetration, with over half of all domains being signed [166], including domains for banks, government and other services. Equally, DNSSEC validation is common in Sweden, with over 70% of users using validating DNS resolvers [167]. Consequently, a failure during the rollover would have disastrous consequences for Swedish society. To the best of our knowledge, this is the *first* time that an algorithm rollover was monitored from start to finish. These measurements provide insight into behavior of resolvers and authoritative name servers and allow operators to plan their rollovers accordingly. The measurement results are publicly available.[1] Last, **(iii)** we develop and publish an open source tool with which operators can easily monitor their rollover themselves.[2] The registries of the Brazilian ccTLD *.br* and of the Danish ccTLD *.dk* used this tool to monitor their

[1] `https://www.simpleweb.org/wiki/index.php/Traces`
[2] `https://github.com/SIDN/rollover-mon`

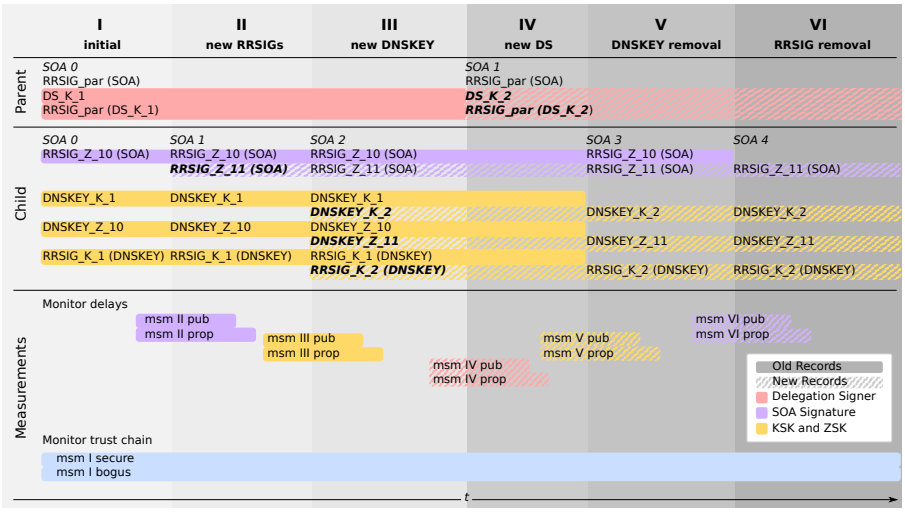


Figure 5.1: Stages of an algorithm rollover (as in RFC 6781 [16]), the expected records in each stage in the zone of the parent and of the child, and accompanying measurements to monitor the rollover. Records in bold mark their first appearance in during the rollover.

algorithm rollover [18], [168], following our method.

The remainder of the chapter is organized as follows: Section 5.2 discusses the stages of key rollovers. Next, Section 5.3 describes in detail what can go wrong during a rollover and why. Then, we propose our method in Section 5.4 and validate each stage of the method with the algorithm rollover of *.se* in Section 5.5. Related work is discussed in Section 5.6. Finally, we summarize our conclusions in Section 5.7.

5.2 KEY ROLLOVERS

Operators that deploy DNSSEC need to roll their keys. A rollover might be necessary in case of a security breach, because they follow a key management policy, or in case operators want to upgrade to a new algorithm [117].

In general, rollovers fall into three categories and vary in complexity: ZSK rollovers, KSK rollovers and rollovers in which the signature algorithms are changed – so-called algorithm rollovers. Depending on the category, operators need to follow different procedures described in detail in RFC 6781 [16]. In all cases, the goal is to keep the chain of trust between the zone, for which the keys are rolled, and the root zone intact.

If operators want to roll their ZSK, only the key itself and the related signatures are replaced, but the KSK stays unchanged. Operators can either have the new

ZSK already published in their zone before the start of the rollover (pre-publish key rollover) or sign their zone with the old and the new key at the same time during the rollover (so called double-signature key rollover) [16]. In contrast, during a KSK rollover operators need to change the KSK, the signatures and ask the parent to update the DS (single-type rollover). Here, the ZSK stays unchanged. In case of an algorithm rollover the cryptographic algorithm of both the KSK and ZSK is changed. Therefore, both keys need to be replaced. Because the KSK is replaced, the DS at the parent needs to be updated as well.

Any key rollover is carried out in multiple stages in which new signatures and keys are added or withdrawn. Between each stage, the operator needs to leave enough time such that resolvers can receive the new records. If not done correctly, caching resolvers might not be able to validate signatures. Figure 5.1 shows the stages of an algorithm rollover over time and we describe the stages in more detail in Section 5.4.

ZSK rollovers are the simplest form of a rollover. Operators do not have to involve a third-party and thus have full control over when to add and withdraw the necessary records. KSK and algorithm rollovers usually require that the DS at the parent is updated and therefore make it more difficult for operators to define the right timing. We explain this and other issue in more detail in Section 5.3 and discuss which factors have an influence on the timing of the stages.

A special kind of rollover has occurred in October 2018, where the KSK of the root zone was replaced and which we studied in the previous chapter. Even though the rollover at the root shares stages of rollovers described in this paper, rolling the root KSK is even more challenging and monitoring required additional measurements.

5.3 ROLLOVER FAILURE MODES

We first explain why the right timing is crucial for a successful rollover and provide a concrete example. Then, we discuss why some resolvers require additional stages during algorithm rollovers.

5.3.1 Timing of Rollovers

The factor that has the biggest impact on the success of a rollover is correct timing. At any point in time during the rollover, resolvers need to have access to the keys and signatures that are necessary to validate the records in their cache. If not, the resolver cannot validate these signatures anymore.

For example, if an operator withdraws the old key too soon from its authoritative name servers, resolvers that still have old signatures but no key in their cache will fail to validate them. This makes them consider the operator's zone "bogus".

Therefore, rollovers are carried out in multiple stages; Figure 5.1 shows a timeline for each stage of an algorithm rollover. The goal of each stage is to make sure that resolvers have enough time to pick up the new signatures and keys before the

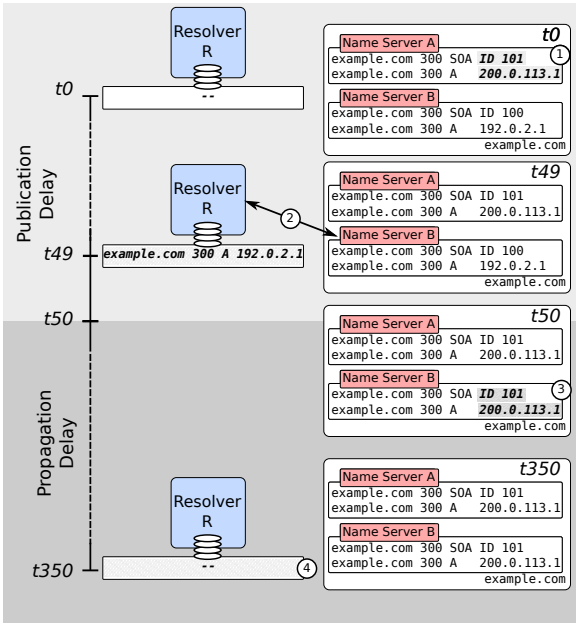


Figure 5.2: Publication and propagation delay. Each resource record consists of the domain (`example.com`), the TTL (300 seconds = 5 minutes), the record type (`SOA` or `A`) and the content. Changes highlighted in bold.

old ones are removed. Thereby, resolvers always have access to the records that they need to validate the signatures in their cache. For example, assume that in Stage IV in Figure 5.1 the operator of the parent zone replaces the `DS` even though resolvers have not yet picked up the new key from Stage III. Then, these resolvers could not establish a chain of trust between the parent and the child and would fail to validate any signature of the latter.

The correct timing is influenced by two factors: (i) the time it takes before a new version of a zone becomes available at every authoritative name server (*publication delay*) and (ii) the time it takes that resolvers pick up the changed records (*propagation delay*).

Publication delay

When operators publish a new record in their zone, it takes time until it is distributed to every authoritative name server. Usually, the zone is updated at one central point and then distributed to the name servers. This creates a period in which the name servers are not in sync and do not serve the same content. Depending on how operators distribute the changes, this *publication delay* might vary from seconds (incremental zone transfers), to minutes (full zone transfers) or even hours (zone transfer upon expired refresh timer). Only after the publication delay has expired and the

name servers are in sync again, operators can be certain that every incoming query from resolvers will receive the new record.

Propagation delay

Resolvers do not query for the new record before their local copy of the record has expired in the cache. Until then, resolvers still serve the old record to their clients. This delay is typically referred to as the *propagation delay*.

Records in the zone of an operator can have different TTLs. The record with the highest configured TTL defines the propagation delay during the rollover. Only after this TTL has expired, operators can assume that none of their records are cached anymore. Resolvers that strictly follow RFC 1035 [169] should not have a propagation delay longer than the original TTL of the record. In practice, operators use TTLs that vary from a few minutes, to hours or even days, depending on the use case and resource record. For example, the most common TTL for `A` records of 2nd level `.se` domains is 1 hour (41% of the domains), followed by 5 minutes (13%) and 1 day (12%) [170].

Figure 5.2 visualizes a general example of publication and propagation delay. ① At *t0*, the operator changes the `A` record and updates the zone at name server *A*. ② 49 seconds later, at *t49*, the resolver queries name server *B*, which is in not in sync yet, for the `A` record and stores it in its cache. ③ Right after, at *t50*, name server *B* also receives the new version of the record. The servers are in sync and the publication delay of the new record has passed. ④ Another 299 seconds later, the TTL of the `A` record in the cache of the resolver has also expired and thus, the propagation delay for the resolver has also passed. It will fetch the new `A` record of `example.com` after it has received a new query for this record.

This example shows, that the maximum time it takes for recursive resolvers to drop an old `A` record after its initial introduction at name server *A* at *t0* is roughly 350 seconds. Summarizing: operators *must* assume that changes to a zone only become visible to every recursive resolver *after* their publication delay *and* their propagation delay have expired.

Impact when disregarding timing

These delays play a significant role when rolling keys; operators need to ensure that any combination of cached records will still validate at all times. This is especially the case when signatures and keys are obtained independent from another.

For example, some resolvers do not query authoritative name servers directly, but instead rely on an upstream resolver to handle their queries. Figure 5.3 describes such a situation. Here, an upstream resolver (R1) has cached the `A` record of `example.com` signed with the old key `DNSKEY_1`. At the same moment, the operator of `example.com` rolls its keys.

① Then, a forwarding, validating resolver (R2) queries R1 for the `A` record. ② R2 receives the `A` record together with its signature and wants to validate it. ③ Because R2 has not cached `DNSKEY_1`, the old ZSK of `example.com`, it queries R1 again. ④

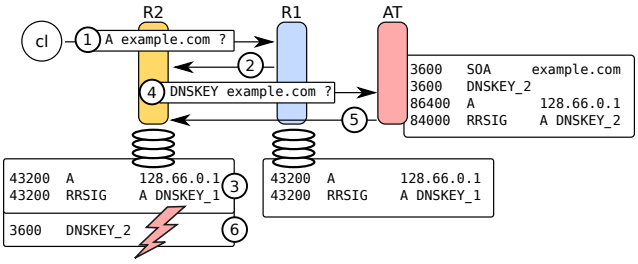


Figure 5.3: Ignoring publication delay causes validation error at forwarding and validating recursive resolver

R1 does not have DNSKEY_1 cached either and it therefore has to query the name server (AT). The operator of AT is in the middle of a key rollover and has already deleted the old ZSK from its zone. ⑤ Thus, R1 only receives the new key, DNSKEY_2, from AT and forwards it to the validating resolver R2. ⑥ R2 cannot validate the old signature with the new key and returns an error to the querying client.

This is only one scenario in which validation failures can occur during a rollover, because an operator does not wait before the publication delay and propagation delay have expired. Only after both have expired, the operator can be confident that no old signatures are still cached and can safely remove the old key. While this may seem like a far-fetched corner case, we have performed measurements that show that this situation can actually occur in practice in Section 5.5.2.

5.3.2 Downgrade Attack

Operators that carry out an algorithm rollover not only have to take propagation and publication delays into account but also face the challenge of resolvers implementing RFCs differently. This is the case with some older versions of resolvers that follow a strict interpretation of RFC 4035, which states that *"there MUST be an* RRSIG *for each RRset using at least one* DNSKEY *of each algorithm in the [...]* DNSKEY *RRset"* [57]. If not, these resolvers suspect an algorithm downgrade attack and consider the record bogus. In a downgrade attack, an attacker attempts to force a validator to accept signatures made with a weaker algorithm, e.g. for which the attacker is capable of forging signatures.

As a consequence, these resolvers expect that every record has a signature for every algorithm used for the DNSKEYs in the zone [171]. RFC 6781 recommends adding the new signatures before adding the keys which results in two additional stages (II and VI in Figure 5.1) when carrying out the algorithm rollover [16]. Thereby, resolvers have the signatures of both keys in their cache already when the new key is added to the zone. If an algorithm rollover skips these additional stages, it is referred to as a *liberal* algorithm rollover.

	Monitor with Measurement	Trust Chain msm_I	new RRSIG msm_II	new DNSKEY msm_III	new DS msm_IV	DNSKEY removal msm_V	RRSIG removal msm_VI
ZSK	Pre-Publish	●	●	●	✗	●	✗
	Double-Signature	●	✗	●	✗	●	✗
KSK	Single-Type	●	✗	●	●	●	✗
	Algorithm	●	●	●	●	●	●

Table 5.1: Rollover types and necessary monitoring measurements. ●marks stages that need to be monitored with the according measurement.

5.4 MONITORING METHOD

The previous section shows that it is crucial for operators to respect the timing of rollovers. If not, resolvers can fail to validate records. The timing is influenced by the propagation delay and publication delay. We have shown that it is not straightforward to respect these delays and that caching resolvers can threaten the availability of a zone.

In this section we propose a novel measurement method, with which operators can prevent these issues. Operators who follow our measurement method can determine with confidence when it is safe to withdraw old keys and signatures, and can monitor the trust chain from the point of view of their clients.

The method consists of three measurement types that accompany each stage of the rollover. At each stage, operators want to know *(i) when is it safe to add new keys and signatures and withdraw old ones?* and *(ii) is my zone secure from the perspective of resolvers?*

Our first two measurement types monitor the propagation delay and publication delay (see 5.5.2). Thereby, operators know when a stage of a rollover has successfully finished and when they can proceed to the next one. The third measurement type acts as a "canary in the coal mine"; it monitors the trust chain and signals *if* the rolled zone becomes bogus at any stage of the rollover.

We develop our method to monitor the most complicated rollover type: the conservative algorithm rollover. Still, the measurement can be applied to other types of rollovers as well, which will be discussed in the following sections. 5.1 describes which measurements each type of rollover requires.

In the next sections, we first describe which vantage points are necessary to monitor the rollover thoroughly. Then, we describe each stage of the method in more detail. Finally, we describe how operators can use our method to define when it is safe to move to the next stage of the rollover. We validate the method with the rollover at the Swedish ccTLD `.se` in 5.5.

5.4.1 Selecting Vantage Points

For measuring the rollover during each stage we require two types of VPs: (i) VPs to measure the deployment of the necessary records at the authoritative name servers *directly* (VPs-direct) and (ii) VPs to monitor the propagation of records in resolver's caches and to verify these resolvers validate signatures successfully (VPs-indirect)

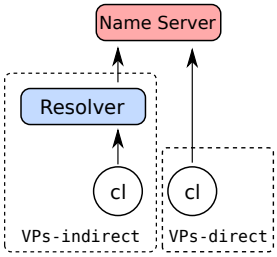


Figure 5.4: Vantage points used in the method. VPs-direct receives responses directly from authoritative name servers, VPs-indirect may receive responses from a resolver's cache.

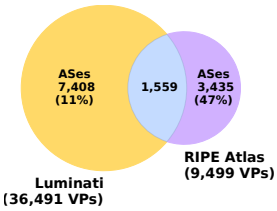


Figure 5.5: Unique ASes of VPs during the .se Rollover. Share of validating resolvers in brackets.

(see Figure 5.4). With VPs-direct, operators make sure that the new RRset is available at every name server. This is a precondition for resolvers to pick up the new RRset and only when all resolvers have picked up the new RRset it is safe to move to the next stage.

To cover different resolver setups and implementations, and to get a realistic view of the clients, we prefer a wide range of VPs that are located in as many different networks and employ as many different recursive resolvers as possible. Thereby, we have a higher chance to discover failures earlier and also cover corner cases, such as strict resolvers (see Section 5.3.2).

Direct VPs must be able to send queries directly to authoritative name servers. The responses must not originate from a cache but must be answered directly from these authoritative name servers. Only then are we able to monitor the current state of the zone at the different servers in real time. In contrast, indirect VPs reflect the "state" of the recursive resolvers. Thus, they must be able to send queries to recursive resolvers that answer their query from cache or query the name servers for them.

VPs that validate the correct publication and propagation of the keys and signatures rely on both VPs-indirect and VPs-direct. VPs-direct query the servers directly and monitor the publication delay. VPs-indirect measure which records a resolver has cached and thereby monitor the propagation delay.

VPs that monitor the trust chain reflect the view of end users and we therefore rely on VPs-indirect. They should cover a broad range of resolvers and networks,

DNS response code		State
msm_I_secure	msm_I_bogus	
NOERROR	NOERROR	insecure
NOERROR	SERVFAIL	secure
SERVFAIL	other	bogus

Table 5.2: The combination of the response codes of msm_I_secure and msm_I_bogus indicates if the trust chain of the rolled domain is intact

such that we can also cover corner cases. Also, it is preferable to select indirect VPs that are behind validating recursive resolvers because they are most likely to be affected by failures during a rollover. The more recursive resolvers we cover, the more our measurements reflect the experience of most clients on the Internet.

In contrast, we typically only need one VP of VPs-direct for each authoritative name server of the child and the parent. The exception to this is the situation in which multiple servers are located behind one address. This is for example the case when a load balancer is used or one name server is replicated to multiple sites using anycast [172]. In this case, a VP can receive different responses, depending on which server its query reaches.

For our measurements, operators can deploy their own VPs, but can also use existing public measurement platforms. In Section 5.5.1, we discuss two of these platforms, RIPE Atlas and Luminati, that when combined provide over 45,000 VPs [69].

5.4.2 The Rollover Stages

After selecting vantage points we must schedule measurements. The measurements are scheduled in parallel to the stages of the rollover and depending on the stage, we monitor either the introduction or withdrawal of a signature (`RRSIG`), key (`DNSKEY`) or `DS` record.

The conservative algorithm rollover consists of 6 stages (see Figure 5.1). We briefly describe each of them and refer the reader to RFC 6781 [16] for the details:

- **I: initial** Start of the rollover. Every key in the zone has the same algorithm.
- **II: new `RRSIG`s** New signatures made with the new key and algorithm are added, but not the new key itself. This is necessary to prevent errors with older, strict, resolvers (see Section 5.3.2).
- **III: new `DNSKEY`** After the new signatures are published at the name servers and have propagated to the resolvers, the new key can be added.
- **IV: new `DS`** After the new key has propagated to resolvers, the old `DS` can be replaced by the new one at the parent.

- **V: remove DNSKEY** The new DS has propagated to resolvers, they should now be able to establish a trust chain with the new key. The old key can be removed.
- **VI: remove RRSIGs** After the old DNSKEY has been dropped from the caches and only the new key is cached, strict resolvers are satisfied as well. The old RRSIGs can be removed, which concludes the rollover.

In the bottom part of Figure 5.1 we show the necessary measurements that accompany each stage. At each stage a different record is added or withdrawn, and because we measure the publication delay and propagation delay independent of each other, we have to schedule new measurements for each stage. The measurements to monitor the trust chain are independent from the changed records and can therefore be scheduled once and can run throughout the rollover.

5.4.3 Define the Right Timing

We now discuss the two measurements that help operators to decide whether a stage has finished successfully and when it is safe to move on with the next stage of the rollover.

The actual algorithm rollover starts in Stage II, when the new signatures are added to the zone and is thus the first stage we monitor. The rollover ends after Stage VI, when the old signatures are removed and also concludes the last measurements.

Monitor the publication delay

In each of the stages, we monitor the publication delay by measuring the introduction of the new RRs and the withdrawal of the old ones. Therefore, we query the servers directly with VPs-direct (msm_II_pub – msm_IV_pub in Figure 5.1).

We start the measurement a few minutes before the zone is updated at the first name server. This creates a baseline and allows us to detect, when the zone has changed at each name server. The publication delay has passed as soon as every VP receives the expected record set from the name servers. From then on, queries towards any name server are responded to with the new record set. Figure 5.1 shows the expected record sets of the child and the parent (SOA_0 – SOA_1 of the parent and SOA_0 – SOA_4 of the child).

Because this should only take a couple of minutes we query the name servers from each VP as frequently as possible. We stop the measurement after every VP-direct receives the expected records from the name servers.

Monitor the propagation delay

We employ VPs-indirect in order to monitor the time it takes until the new state of the zone propagates to resolvers (msm_II_prop – msm_IV_prop in Figure 5.1).

A few minutes before the zone of the child or parent changes, we configure VPs-indirect to query for the record that is supposed to be added or withdrawn next. This

Domain Name	TTL	class	type	value	
`secure.example.com`	`600`	`IN`	`TXT`	*some string*	
`secure.example.com`	`600`	`IN`	`RRSIG`	`RRSIG TXT 8 3 600 (tFzgUjaq[...]ABEbA=)`	←Valid Signature
`bogus.example.com`	`600`	`IN`	`TXT`	*some string*	
`bogus.example.com`	`600`	`IN`	`RRSIG`	`RRSIG TXT 8 3 600 (123456)`	← Bogus Signature

Table 5.3: Examples of two test records, used to validate the trust chain. msm_I_secure and msm_I_bogus query the respective records.

creates a baseline. We continue querying for the record from each VP periodically. The periodicity depends on the TTL of the changed record. The shorter the TTL the faster resolvers drop the old record from their cache and the more frequently the operators should monitor this transition. As a minimum, each VP should query for the new record (i) before the new record set is introduced, (ii) before the TTL has expired and (iii) after the TTL expires. This ensures the whole transition can be monitored.

As soon as the zone is updated at the first name server we expect to see more and more resolvers dropping the old records from their cache, querying for the new record and returning the new record to our VPs. The propagation of the new zone state is successful when every VP receives the new state of the zone from their recursive resolvers. This should take at least the TTL of the added or removed record. Then, we can stop the measurements with VPs-indirect and the operator can safely move to the next stage of the rollover.

5.4.4 Monitor the Trust Chain

Operators want to make sure that their zone stays secure during each stage of the rollover. Therefore, we monitor the chain of trust. This measurement acts as a "canary in the coal mine" and relies on VPs-indirect.

The goal is to measure if resolvers can still resolve and validate signed records of the rolled domain or its delegated domains. Resolvers that were able to resolve and validate the records before but suddenly stop validating or even stop resolving during the rollover are a strong signal that *something* went wrong.

We start this measurement in Stage I of the rollover to establish the baseline state of our VPs. Resolvers can either be successfully validating the signatures of the rolled zone (secure), not validating but successfully resolving (insecure), or not resolving at all (bogus). A deviation from this baseline at any point in time during the rollover signals a failure as described in Section 5.3.

We establish the baseline with the help of two additional RRs (e.g. two arbitrary `TXT` records that contain a random string). We can include the RRs either directly in the monitored zone or in the zone of one of its children. The first RR has a valid `RRSIG` and every resolver that operates correctly should be able to resolve the record. Table 5.3 shows the two records, one with a valid, one with a bogus

Measurement	VP	Start and End Date	Responses
Timing Issues	Atlas	2018-01-18	19,501
Downgrade Attack	Atlas	2018-01-12	19,648
Publication Delay	Atlas	2017-12-14 —22	22,059,735
Propagation Delay	Atlas	2017-12-14 —22	2,978,366
Trust Chain	Atlas	2017-12-14 —22	16,861,137
Trust Chain	Luminati	2017-11-29 —12-20	1,696,262

Table 5.4: Measurements to evaluate failure scenarios (Section 5.5.2) and stage IV of the *.se* rollover (Section 5.5.3)

signature.[3] The second RR has an `RRSIG` which is bogus, and therefore validating resolvers should not validate the signature successfully. Not-validating resolvers should resolve the bogus record without any issues. We query both RRs from each VP-indirect (msm_I_secure and msm_I_bogus in Figure 5.1).

By combining the outcome of the measurements of the secure and bogus records we can determine whether a resolver is (i) a secure resolver and validates the records correctly, (ii) an insecure resolver, or (iii) a resolver that fails to validate the correct signature. Secure resolvers resolve the secure record correctly (response code NOERROR) and return with the response code SERVFAIL when querying for the bogus record. Insecure resolvers return for both records the response code NOERROR. Failing resolvers return at least an error for the secure record but might fail resolving the bogus record as well (see Table 5.2).

We start the measurements in Stage I and stop them when the rollover concludes. Each VP should query for the test records once per TTL to detect failures as fast as possible. Resolvers that change their state, or an increase in bogus resolvers are a strong signal for rollover issues. Operators can debug these issues with msm_II_pub – msm_IV_pub and msm_II_prop – msm_IV_prop. Thereby they will know whether their servers serve the expected records or if resolvers miss necessary records for validation.

5.5 ROLLOVER VALIDATION AND APPLICATION

In this section we validate our measurement method. We replicate failure modes described in Section 5.3 and measure how likely these failures are to occur at resolvers in the wild. Then, we apply our method to the algorithm rollover of the Swedish ccTLD `.se`. Table 5.4 provides an overview of the measurements analyzed in this section.

5.5.1 Selecting Vantage Points (VPs)

To monitor the DNS, and the algorithm rollover in particular, we need the right measurement platform. For our measurements we rely on the vantage points of

[3]The operator should create the signatures with an algorithm that is widely supported by validating resolvers.

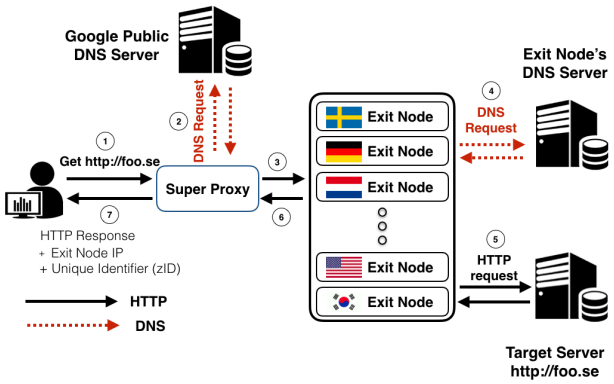


Figure 5.6: Timeline of a request in Luminati: the measurement client sends a HTTP request to the super proxy ①; the super proxy makes a DNS request to the Google Public DNS server ②; once the DNS request succeeds, it forwards the HTTP request to one of the exit nodes ③; the exit node makes a DNS request to its DNS resolver ④, then requests the HTTP content ⑤. The HTTP response is then returned to the super proxy ⑥, then to the client ⑦.

RIPE Atlas and Luminati. For replicating the failure modes we only rely on VPs of RIPE Atlas. In this section we discuss the costs and benefits of these platforms.

RIPE Atlas

As described in Explainer 2, a RIPE Atlas probe is a hardware device that actively measures Internet connectivity. An Atlas probe is able to act as a VP for direct and indirect measurements (see Figure 5.4). It can either send queries through its pre-configured resolvers or can send queries directly to authoritative name servers. If multiple resolvers are configured, then the probes send queries to all of them.

The Regional Internet Registry RIPE regulates the usage of its measurement platform with the help of credits [69]. Users can earn credits, e.g. by hosting their own VP or by sponsoring RIPE. Further, by default RIPE limits the number of measurement results a user can create and the number of simultaneous VPs that can be used at any time. Upon individual request, RIPE may relax these limits but even with them in place, RIPE Atlas is still a useful platform to monitor a rollover.

Instead of using every available VP, operators can only use probes that reflect their client base. For example, the RIPE Atlas API allows users to select probes located only in a certain country or in a certain network. To limit the use of credits, operators can, for example, start monitoring the trust chain (msm_I_secure, msm_I_bogus) just before the next stage of a rollover and stop it when a stage has finished successful.

Luminati

Luminati [133] is a paid HTTP/S proxy service that enables clients to route traffic via the Hola Unblocker Network. Hola Unblocker allows users to route their traffic via a large number of proxies. It is available on multiple platforms such as Windows, Mac, and browser extensions and has been installed by more than 149 million users around the world. Luminati uses machines that installed the Hola Unblocker to allow its customers to route their traffic via the machines.

To route HTTP/S traffic via the Luminati network, a client first sends the request to one of the Luminati servers (called the super proxy). Then, the super proxy looks up the destination domain using Google Public DNS and forwards the HTTP/S request to one of their Hola clients (called the exit node) if it is a valid domain name.[4] An exit node makes a DNS request to its name server, and then makes the HTTP/S request. Once the response comes back from the destination, the exit node will forward the response back to the super proxy, which also forwards it back to the client. Figure 5.6 shows this process schematically. For more details on using Luminati for network measurements, we refer the reader to the study by Chung et al. [134].

Application

We use the VPs from RIPE Atlas to evaluate the failure modes and both platforms to monitor the rollover of `.se`; we obtained more than 9,500 VPs from RIPE Atlas and 36,000 VPs from Luminati. RIPE Atlas allows us to send DNS queries directly to the name servers or via the pre-configured recursive resolver, thus acting as VP-indirect when relying on their resolvers and acting as VP-direct when querying the name servers directly. Luminati, in contrast, only allows us to send HTTP requests via the exit nodes, which makes these exit nodes send DNS queries via recursive resolvers [134]. Hence, RIPE Atlas probes act as both the VPs-direct and VPs-indirect, but Luminati only as the latter.

Probes of RIPE Atlas are very often located behind validating resolvers (see Figure 5.5), which is useful to monitor the chain of trust, but also are often not located in residential networks [173]. The opposite is the case for clients of Luminati; the large majority is located in residential networks but only around 12% use validating resolvers [135]. As shown in Figure 5.5, these two platforms cover a very different set of networks, such that by combining these two different approaches they allow us to have a more comprehensive view on resolvers around the world.

5.5.2 Evaluate Failure Modes

In this section, we demonstrate that the issues described in Section 5.3 are not only theoretical. We show that operators should indeed monitor rollovers thoroughly, and algorithm rollovers especially, using our method. We use our own second level

[4] Google's DNS servers will return a SERVFAIL response to the super proxy if DNSSEC validation fails.

test domain name (`ourtestdomain.nl.`) and the VPs of RIPE Atlas to replicate different failure modes.

Timing issues

As explained previously in Section 5.3.1, especially interlinked caches can lead to validation failures during rollovers that are not carried out correctly. We replicate the situation in Section 5.3.1, in which a forwarding resolver fails validation, with RIPE Atlas probes and a domain under our control.

Our zone consists of one signed `TXT` test record with a TTL of 24 hours and the accompanying key material, with a TTL of 1 hour. Then, we query for the test record from each RIPE Atlas probe, using its pre-configured resolvers. Thereby, validating resolvers query for the test record, its signature and the keys and store them in their cache. Then, we carry out a ZSK rollover and remove the old key from the zone. An hour later, the `DNSKEY` record should have expired from the cache, but the test record should still be cached. Then we query for the test record again.

Forwarding resolvers that do not have the `TXT` test record cached (e.g. because they do not have a cache implemented) now need to query their upstream resolvers again for the `TXT` record and the key. The upstream resolvers should not have the old `DNSKEY` record in their cache anymore but only the `TXT` record and old signature. Therefore, they need to query our name servers for the key again which now respond with the new key. Forwarding resolvers cannot validate the old signature with the new key and therefore fail validation.

Out of 10,155 VPs, at least 38 use a validating forwarding resolver and return an error. This is just one of many scenarios where not respecting the publication and propagation delay leads to failures and shows that respecting these delays is crucial when carrying out a rollover

Downgrade attack

A failure mode that applies to algorithm rollovers, and thus, also to the rollover of `.se` are resolvers that expect signatures with each of the algorithms in the `DNSKEY` RRset of a zone. We use every available RIPE Atlas probe to measure in the wild how many resolvers follow this strict interpretation.

Out of 10,952 probe-resolver pairs, 6 fail for zones that do not provide signatures for every available algorithm. Thus, operators that do not follow the conservative interpretation of RFC 4035 can expect a small number of resolvers to fail validating their zone during the rollover.

5.5.3 The `.se` Use Case

After we have shown that it is indeed necessary to respect the timing during rollovers we now apply our method on the algorithm rollover of `.se`.

In December 2017, the Swedish ccTLD *.se* carried out their first ever algorithm rollover, moving from the RSA/SHA-1 to the RSA/SHA-256 signing algorithm. The

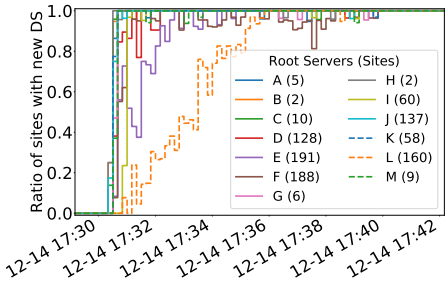


Figure 5.7: Share of probes that observe the new DS at the Root (#sites in brackets)

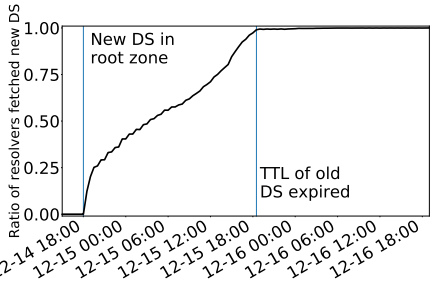


Figure 5.8: Share of resolvers that see the new DS, 24 hours after its introduction

.se ccTLD was the first ccTLD to deploy DNSSEC [174] in 2005, well before the root zone got signed in 2010. At the time of the rollover, .se had more than 1.4 million registered domain names and more than half of the them were signed with DNSSEC [166]. Furthermore, more than 70% of Swedish Internet users rely on validating resolvers [167] to resolve these domains. If the algorithm rollover of .se would fail, the impact on Swedish society would be devastating: the majority of clients that rely on .se domains would likely not be able to reach .se domain names for minutes or even hours.

Therefore, it is crucial for the operator of .se that the rollover succeeds. We apply and validate our method based on this event and demonstrate how it supported the operators of .se during their rollover. During the rollover, we provided the operators of .se insights into their rollover in real-time by processing the measurements and visualizing the results on a dashboard.

In the remainder of this section we rely on the replacement of the DS in Stage IV as a use case. It is one of the most crucial stages in the rollover for two reasons. First, it involves interaction with the parent, which is only partially under the control of the operator. Second, whereas the previous stages could only have a direct impact on resolvers that follow the conservative approach (see Section 5.3.2), this is the first stage where a failure would affect every record in the zone and every validating resolver.

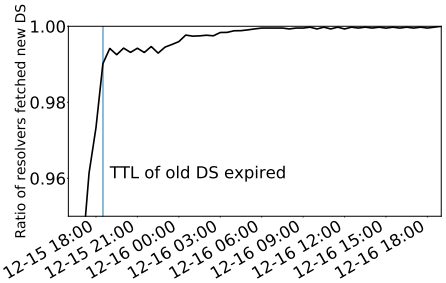


Figure 5.9: Share of resolvers that only observe the new DS after the TTL of the old DS has expired

Monitor the publication delay

Using VPs-indirect of RIPE Atlas, we measure when every server of the root serves the new DS (msm_IV_pub in Figure 5.1).

The DS of .se is replaced at around 18:30 UTC. The first probe observes the new DS at 18:30:25 at J-root and 32 seconds later every root server has the new DS deployed on at least one of their sites. After 5 more minutes, over 99% of the probes receive the new DS (see Figure 5.7). Note that some root server letters need more time to distribute the new zone to their slaves than others: only after 10 minutes every probe receives the new DS. From this point on, the operator can expect that every resolver will receive the new DS from the root.

The root servers are heavily replicated using anycast. Root servers with many sites, however, do not necessarily distribute the new DS across their sites slower than root servers with fewer sites. D-root with more than 120 sites have their sites in sync almost as fast as C-root with only 10 sites. Note that, because of external factors such as network congestion, the publication delay can vary every time a new version of the zone is distributed. A full study of the reasons for propagation delays at root operators is outside the scope of this paper and we suggest to study this phenomenon in future work.

Monitor the propagation delay

In contrast to the publication delay, the propagation delay, measured with msm_IV_prop in Figure 5.1, is significantly longer. Most of the resolvers of VPs-indirect (RIPE Atlas only) pick up the keys within 1 day (see Figure 5.8). This is expected since the TTL of the DS is 24 hours. A small share of resolvers (less than 1%) still have the old DS in their cache 48 hours after its withdrawal and only after 50 more hours the last VP has dropped the old DS. This is likely caused by resolvers that ignore the TTL or do not forward queries to one of the official root servers (see Figure 5.9) [175].

Operators should validate whether these lagging resolvers send a significant share of queries to their authoritative name servers. If so, they might want to try to contact the operators of the resolver to fix this issue before moving on to the next

stage. If not, they can safely move on to the next stage and neglect these lagging resolvers.

Based on these measurements, the operators of .se know that they have to wait at least the publication delay of 10 minutes and the propagation delay of 48 hours before moving on to Stage V. Then, they can withdraw the old DNSKEY from their zone safely.

Monitor the trust chain

For the entire duration of the rollover, we monitor the trust chain of .se from the perspective of a second-level .se domain (msm_I_secure and msm_I_bogus in Figure 5.1).

A caching resolver can only detect a failure in the trust chain if the record that caused the failure has expired from cache. In .se, the TTL of the DS is 1 hour. As a consequence, we can detect failures for .se or the root with one second-level domain only once per hour.

To address this shortcoming, we create five second-level test domains with one validly signed and one bogus signed record each. From each VP-indirect we query the records of every domain once per hour. We schedule the measurements such that each VP spreads its queries to the domains equally. Thus, within an hour a VP sends 5 queries to a bogus record and 5 queries to a secure one. By combining the response codes of the queries to the same domains, we can detect a failing resolver of a VP at least every 12 minutes.

We again monitor Stage IV, in which the DS is replaced at the root. Because the TTL of the DS at the root is 24 hours we would not see the impact of this failure immediately. Figure 5.10 shows the VPs- indirect (RIPE Atlas and Luminati) that are secure, insecure or bogus before and after the DS is replaced. We do not observe an increase in bogus resolvers and the number of secure resolvers also stays stable. This shows that .se remains secure during the rollover and, very likely, also end users do not experience issues. In fact, our measurements show that .se is secure during *every* stage of the rollover. This is the desired result for its operator and thus, the rollover is carried out successfully.

Lessons learned and other use cases

In this use case, we used all available RIPE Atlas and Luminati VPs. In order to reduce cost and the impact of the measurements on network resources, operators can select VPs that reflect their actual client base. For example, by selecting VPs that are located in the network of their local ISPs and VPs that make use of large public DNS providers such as Google, operators can likely cover most resolvers that their clients rely on [143]. In order to cover corner cases, operators should still employ as many VPs as possible. Only then, also forwarding resolvers or resolvers behind load balancers are covered.

After applying our method to .se we also supported the operators of the Brazilian ccTLD .br in applying our method to their algorithm rollover [18]. In October 2018,

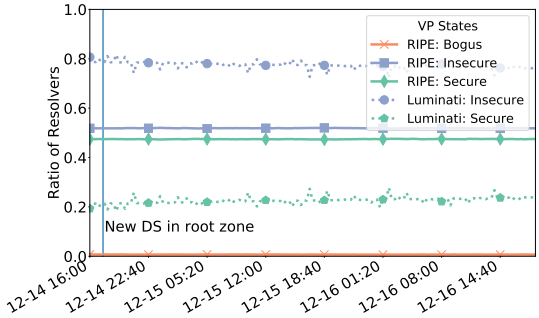


Figure 5.10: State of VPs that successfully validated signed `.se` domains before and after the new DS was introduced

they rolled the keys from RSA-SHA1 to the new elliptic curve algorithm ECDSA-P256-SHA256 and decided to follow the liberal approach for the rollover. Again, the operators relied on every available VP of RIPE Atlas and Luminati. As with `.se`, their rollover was carried out successfully and the measurements, set up according to our method described in this paper, did not show any significant failures [176]. This also demonstrates that the number of resolvers that follow the strict approach for the rollover (as described in Section 5.5.2) was not significant enough to jeopardize a liberal rollover. In 2020, also the operators of the Danish ccTLD `.dk` switched to ECDSA-P256 [168]. Also they relied on our measurement method, using it mostly to monitor the trust chain.

5.6 RELATED WORK

Operators have multiple tools at hand that allow them to debug errors in DNSSEC and automate rollovers and can rely on rough guidelines on how to roll their keys. In comparison to the method described in this paper, all of these tools, however, lack *concrete* recommendations about the correct timing during the roll.

5.6.1 Challenges of DNSSEC Rollovers

The particular risks of DNSSEC KSK rollovers have been described in academic literature multiple times in the past. In 2007, Ariyapperuma et al. note that DNSSEC rollovers are a risk which has not been addressed at the operational level and Yang et al. particularly describe how the effects of caching can break the chain of trust [159], [177].

Chung et al. measure DNSSEC rollovers on second level domains in `.com`, `.net` and `.org` over a period of 21 months [135]. During that time, only 30% of the signed domains carried out a KSK rollover, which suggests that operators consider KSK rollovers too risky. Of the domains that rolled their keys, 7% of them did not respect the propagation delay of the keys, which may have caused validation errors.

Besides caching, an increased response size for DNSKEY queries can also cause a risk during a rollover. This increase may cause packets to be fragmented and possibly blocked on their way to the resolvers. Van Rijswijk et al. describe this risk and analyze how elliptic curve cryptography can address this [94].

5.6.2 Rollover Guidelines

Because of the added complexity of DNSSEC rollovers, three informational guidelines have been published in the IETF intended to help operators roll their keys correctly. RFC 4641 [178] is now considered obsolete and is updated by RFC 6781 [16]. RFC 6781 describes the different rollover types in detail and explains each step an operator has to carry out. The document, however, does not give concrete guidelines, when to proceed from one step of the rollover to another. RFC 7583 [17] makes more concrete recommendations about the timing of a rollover, but because the actual time it takes for records to propagate across the DNS can differ from expected behavior, just following these recommendations is likely not sufficient to achieve a flawless rollover.

5.6.3 Debugging DNSSEC

Open source tools such as DNSViz and Zonemaster can debug the configuration of a zone, including DNSSEC records and the chain of trust [91], [179]. Operators can use these tools to check the publication of records at their name servers. They are not suitable for monitoring the propagation of records. Also, these tools were not developed for continuous monitoring and are thus not suited to monitoring a longer running process such as a rollover. With our methodology on the other hand we can measure the propagation and publication continuously throughout the whole process.

The measurement platform of APNIC continuously measures the number of clients that rely on validating resolvers. However, the platform is not public and does not focus on the validity of individual domain names [167].

5.6.4 Automating Rollovers

In the early days of DNSSEC deployment, operators had to create, introduce and remove keys and signatures manually. This requires many manual steps and, like most manual processes, is prone to errors. Today, tools exist to mostly automate rollovers and are implemented either directly in the name server software, are exclusively developed to manage DNSSEC of a zone, or support decision making during the roll.

For example, since version 9.7, BIND can automate the process of creating new keys and adding them to the zone [180]. In case of a KSK or algorithm rollover, however, operators need to withdraw old keys manually from the zone. Also, the interaction with the parent needs to be done manually. With our method, operators know when they can safely remove the `DS` records from parent. The name

server software Knot DNS can also carry out rollovers automatically, including algorithm rollovers [181]. If configured, Knot DNS will check automatically at the parent whether the new key is updated and waits an additional TTL before removing the old key. As shown in Section 5.5.3, waiting one TTL might be not long enough for every resolver to drop the old keys and signature from cache. Operators who follow our method will have more confidence when it is safe to remove the old key and can manually instruct Knot DNS to do so.

OpenDNSSEC is a tool that automatically keeps track of DNSSEC keys and handles DNSSEC signing [182]. It can also automate rollovers to some extent. If operators pre-configure their publication and propagation delays, OpenDNSSEC can carry out ZSK rollovers fully automated. KSK and algorithm rollovers, however, still require manual work. OpenDNSSEC uses fixed timers and cannot detect when the DS record at the parent is published and has propagated. Thus, operators still have to monitor the publication and propagation themselves and communicate the state of the DS to OpenDNSSEC. Our method allows operators to do so which lets them safely continue with the rollover. Other commercial tools exist but they invariably require some manual interaction of the operators as well [183].

An attempt to automate the interaction with the parent is described in RFC 7344. This standard introduces CDS (Child DS) and CDNSKEY (Child DNSKEY) records with which operators can signal to their parent zone that they want to add, roll, or delete their DS [87]. DNS provider Cloudflare supports CDS and CDNSKEY and also the TLDs .ch, .li and .cz update the DS if they detect a CDS or CDNSKEY record at one of their child domains but overall, the adoption of this standard is low [59], [184]–[186].

Still, none of the tools can say with confidence when the keys and signatures have propagated to the resolvers and thus, active monitoring is still necessary. Our method describes how operators should actively monitor their rollovers and gives them the confidence when the required records are public and have propagated.

5.7 CONCLUDING REMARKS

In this chapter, we have demonstrated the complexity of DNSSEC rollovers and algorithm rollovers in particular and how to address them, using our novel measurement method. We have shown that issues with timing and legacy resolver software during the rollover are not only theoretical and can have a severe impact on the availability of a zone.

Because failure is not an option for operators, we contributed a measurement method to prevent failures from happening. With the help of our method, operators know when it is safe to proceed in each stage of a rollover. In addition to this, our method allows them to confirm that their clients can validate their zone at any point in time during the rollover.

We demonstrated this by applying our method to the algorithm rollover of the Swedish ccTLD .se and we showed that the rollover of .se was carried out without issues for clients. The operators, however, reported small issues, especially during the creation of their new keys. Because our approach provided them with insight

into their rollover they had the confidence to continue the rollover regardless of these problems [165].

Our method provides the final link to fully automate DNSSEC rollovers. As shown in Section 5.6, tools and protocols exist to automatically create and publish new keys and signal to the parent that they should be updated. With our method, operators now also know exactly when it is safe to withdraw old keys and signatures. We publish our tool as open source software, so any operator can set up the measurements necessary to implement this method themselves, using the vantage points of RIPE Atlas.[5] Our command-line tool, implemented in Python, first schedules every measurement described in this article for a zone defined by the operator using the RIPE Atlas API. It then processes the measurement results at a configurable interval. Finally, it gives as output the current state of the publication and propagation of the changed records. Operators can use the output to identify lagging name server instances or resolvers and to decide when to move to the next stage of the rollover. In the future, the output of this tool can also be used as an input for software such as OpenDNSSEC, Knot DNS or other DNSSEC signer software. For example, these signer implementations could automatically withdraw old keys, if the new key has propagated to at least 99% of the vantage points. Thereby, we close the last gap of fully automating rollovers, reduce their risks and address one of the barriers when deploying DNSSEC. This paves the way for more DNSSEC-signed zones in the future, which would increase the security of the DNS overall. The tool was also applied by the operators of `.br` and `.dk`.

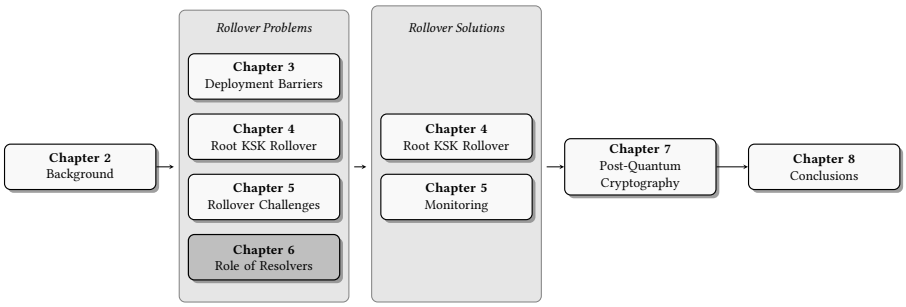
The provided tool, but also the fact that the algorithm rollovers of all three ccTLDs were a success, should give operators of other signed zones also the confidence to roll to a more secure algorithm. Only then, DNSSEC can provide security for the DNS also on the long term.

Before we look forward, with the threat of quantum computers on the horizon, we first dive deeper into one aspect that affects the timing of rollovers also: the way resolver select authoritative name servers.

[5] Source code and documentation available here: `https://github.com/SIDN/rollover-mon`

CHAPTER 6

The Role of Resolvers



In the previous chapter, we have shown that timing is crucial when it comes to carrying out algorithm rollovers. This is especially the case for some "strict" resolvers that require an additional stage during the rollover. In this chapter, we show how resolver behavior, not related with DNSSEC operations at first sight, can complicate algorithm rollovers. The root cause lays in the way resolvers choose between authoritative name servers when sending their queries. In our study, we measure this behavior "in the wild". The results have been first published as a paper in a peer-reviewed conference [20].

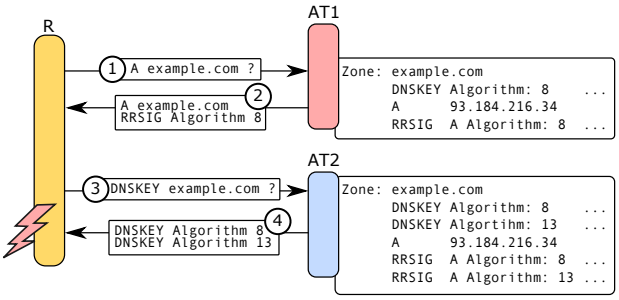


Figure 6.1: Scenario of an algorithm rollover in which out of sync authoritative name servers can lead to validation failures at "strict" validating resolvers

6.1 INTRODUCTION

In the previous chapter, we have discussed the complexities of algorithm rollovers. There, we have also explained that algorithm rollovers require two additional stages in comparison to "regular" KSK rollovers. This is necessary because some resolvers expect a signature with each algorithm occurring in the DNSKEY set [57]. If this is not the case, these resolvers may consider the zone bogus, as missing signatures for certain algorithms is seen as a downgrade attack.

If an operator decides to skip the two additional stages, these *strict* resolvers can encounter a mismatch between published algorithms and signatures in some situations. For example, strict resolvers might be *forwarding* their queries to two different recursive resolvers (similar to the situation in Figure 5.3 of Chapter 5). A strict resolver might fail validation if it fetches the key set from one recursive resolver, already serving the new version of the zone, but the signature from the other recursive resolver. If the latter resolver still has the old version of the zone cached then the strict, forwarding resolver, might end up with two keys with two different algorithms, but with only one signature, leading to a validation failure.

In another example, a strict resolver sends its queries to different name servers which are not in sync yet, as described in Figure 6.1. In the previous chapter, we have shown that it takes time before different authoritative name servers, and their anycast instances, serve the same content of the zone. Imagine now, that the strict resolver receives only the signature of the old algorithm from the lagging name server (① and ② in Figure 6.1). Then, ③ it queries the second authoritative name server for the keys. If this name server already serves the new version of the zone, it returns two keys with two different algorithms, again leading to a validation failure ④.

In this chapter we want to understand, how often recursive resolver spread their queries across multiple authoritative name servers, potentially causing validation failures as described above. There are many different implementations of recursive resolvers with a multitude of software releases. It is not defined how they select between authoritative servers, and we cannot determine which implementations

run where, nor how many of each exist. Early work [187] shows that the behavior across different recursive resolvers is diverse, with some making intentional choices and others alternating across all NSes for a service. While this result has been reconfirmed, to our knowledge, there is no public study on how this interacts with different design choices of name server deployments, nor how it should influence its design.

The first contribution of this chapter is to *re-evaluate how recursive resolvers select authoritative name servers* (Section 6.4), but in the wild, with the goal of learning from the *aggregate* behavior in order to better engineer authoritative deployments. We answer this question with a controlled study of an experimental, worldwide, name server deployment using Amazon Web Services (AWS) coupled with global data from the Root DNS servers and the `.nl` TLD (Section 6.5).

Our key results are that most recursive resolvers check all authoritative name servers over time (Section 6.4.1), about half of recursive resolvers show a preference based on latency (Section 6.4.2), and that these preferences are most significant when authoritative name servers have large differences in latency (Section 6.4.3).

These findings also demonstrate that one certain behavior of resolvers, unrelated to DNSSEC at first sight, can complicate DNSSEC operations. Thereby, we show once more how easily DNSSEC can lead to errors if operators are not careful.

Our second contribution, is not directly related to DNSSEC, but to DNS performance. Based on our findings in this study we suggest *how DNS operators can optimize a DNS service* to reduce latency for diverse clients (Section 6.7). In order to achieve optimal performance we conclude that all NSes need to be equally strong and therefore recommend to use anycast at all of them. This new recommendation augments existing practices about operation of individual anycast services [172], [188], with advice about DNS services that employ multiple NSes.

6.2 MODERN DNS SETUP

Figure 6.2 shows, as an example, how the components of the DNS are deployed in the wild today. Here, zones are distributed across multiple name servers (AT). These name servers in turn, are sometimes distributed even further, using anycast. With anycast, multiple name server that share the same IP address are placed in different locations. Then, the routing protocol of the Internet makes sure that queries, directed towards the IP address arrives at the anycast *site*, closest to the recursive resolver.[1] For more details on Internet routing and BGP we refer the reader to Moura et al. [189].

By the time of this study in 2017, most of TLDs within the root zone use 4 NSes, but some use up to 13, and each of these NSes can be replicated and globally distributed using IP anycast and load balancers [189]. Second level domains like `example.com` under TLDs like `.com`, `.net` and `.org` have a median of 2 NS records (mean of 2.3, 2.4, and 2.4) and the domain names of `.nl` have a median of 3 NS records (mean of 2.6 as of 2017-08-01).

[1] Closest in the view of the routing system, not necessarily closest geographically

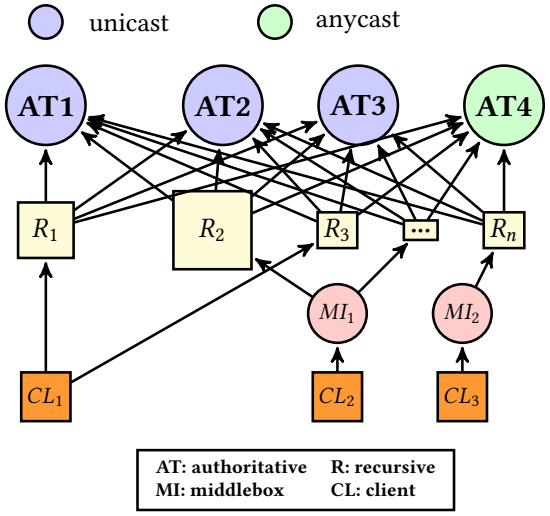


Figure 6.2: TLD Setup, Recursive Resolvers, Middleboxes and Clients

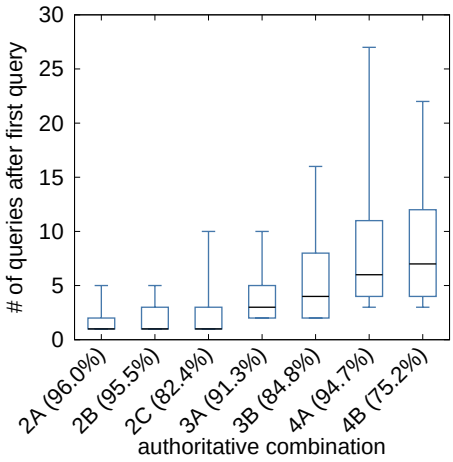


Figure 6.3: Queries to probe all authoritative name servers, after the first query (Boxes show quartiles and whiskers 10/90%ile)

Recursive resolvers (*R* in Figure 6.2), receiving a query from their clients (*CL* in Figure 6.2) and that cannot find the answer in their local cache then have the choice of multiple authoritative name server to send their query to.

Besides the local cache with information on DNS records, many resolvers also keep an *infrastructure cache* with information on the latency (Round-Trip Time (RTT)) of each queried authoritative server, grouped by IP address. The infrastructure cache is used to make informed choices among multiple authoritative name servers for a

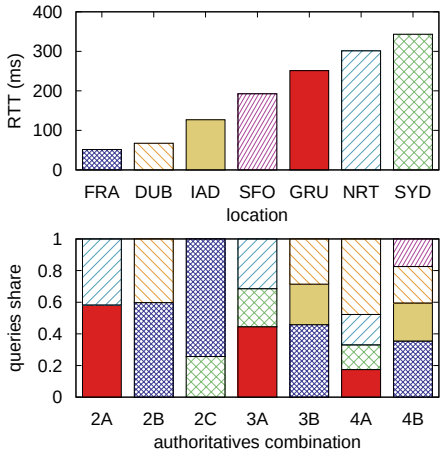


Figure 6.4: Median RTT (top) and query distribution (bottom) for combinations of authoritative name servers

ID	locations (airport code)	VPs
2A	GRU (São Paulo, BR), NRT (Tokyo, JP)	8,702
2B	DUB (Dublin, IE), FRA (Frankfurt, DE)	8,685
2C	FRA, SYD (Sydney, AU)	8,658
3A	GRU, NRT, SYD	8,684
3B	DUB, FRA, IAD (Washington, US)	8,693
4A	GRU, NRT, SYD, DUB	8,702
4B	DUB, FRA, IAD, SFO (San Francisco, US)	8,689

Table 6.1: Combinations of authoritative name servers we deploy and the number of VPs they see

given zone. For example, Unbound [190] implements a smoothed RTT (SRTT), and BIND [191] an SRTT with a decaying factor. Some implementations of recursive resolvers, particularly those for embedded devices like home routers, may omit the infrastructure cache.

6.3 MEASUREMENTS AND DATASETS

Next we describe how we measure the way recursive resolvers choose authoritative servers, using both active measurements and passive observations of production DNS at the root and `.nl`. Our work focuses on measurements from the field, so that we capture the actual range of current behavior, and to evaluate *all* currently used resolvers. Our work therefore complements prior studies that examine specific implementations in testbeds [187]. Their work are definite about why *a* recursive makes a choice, but not on *how many* such resolvers are in use.

6.3.1 Measurement Design

To observe recursive-to-authoritative mapping on the Internet, we deploy authoritative servers for a test domain (`ourtestdomain.nl`) in 7 different datacenters, all reachable by a distinct IPv4 unicast address. Sites are hosted by Amazon, using NSD 4.1.7 running on Ubuntu Linux on AWS EC2 virtual machines.

We then resolve names serviced by this test domain from about 9,700 vantage points (VPs) distributed over 3,300 Autonomous Systems (ASes) (of which 1,040 ASes host 2 or more probes), all the RIPE Atlas probes that are active when we take each measurement [69] (see Explainer 2 for more details on RIPE Atlas). Each VP is a DNS client (a CL in Figure 6.2) that queries for a DNS TXT resource record using an IPv4 address.

Each VP uses whatever their local configured recursive is. Those recursive resolvers are determined by the individual or ISP hosting each VP. Overall, we observe over 11,000 unique IP addresses of upstream recursive resolvers at our authoritative name servers, located in over 2,500 ASes.

To determine which authoritative NS the VP reaches, we configure each NS with a *different* response for the same DNS TXT resource. While most studies of anycast catchment use DNS CHAOS-class queries, where a query on the `hostname.bind` or `id.server` identifies a specific authoritative [192], CHAOS queries would be answered directly by the configured recursive server. We use Internet-class queries that pass through a recursive to the authoritative. The resulting dataset from the processing described is publicly available at our website [193] and at RIPE Atlas [194].

Cold caches. DNS responses are extensively cached [195]. We insure that caches do not interfere with our measurements in several ways: our authoritative name servers are used only for our test domain, we set the time-to-live (TTL) [150] of the TXT record to 5 seconds, use unique labels for each query, and run separate measurements with a break of at least 4 hours, giving recursive resolvers ample time to drop the IP addresses of the authoritative name servers from their infrastructure caches.

Name servers location. We deploy 7 combinations of authoritative servers located around the globe (Table 6.1).

We identify each by the number of sites (2 to 4) and a variation (A, B, or C). The combinations vary geographic proximity, with the authoritative name servers close to each other (2B, 3B, 4B) or farther apart (2A, 2C, 3A, 4A). For each combination we determine the recursive-to-authoritative mapping with RIPE Atlas, querying the TXT record of the domain name every 2 minutes for 1 hour. We choose 2 to 4 name servers because it reflects the most common name server deployments and is enough to provide geographic diversity. While we consider "only" one hour of data, it seems unlikely that authoritative selection is strongly affected by diurnal factors.

Measurement challenges and considerations. We consider several challenges that might interfere with our measurements.

Atlas probes might be configured to use multiple recursive resolvers and, therefore, in our analysis we consider unique combinations of probe ID and recursive IP

as a single VP (or client, in Figure 6.2);

Middleboxes (load balancers, DNS forwarders) between VPs and recursive resolvers (MI in Figure 6.2) or recursive resolvers which use anycast may interfere, causing queries to go to different recursive resolvers or to warm up a cache. Full studies of DNS resolution are quite involved [26] and outside the scope of this paper. We confirm that middleboxes have only minor effects on our data by comparing client and authoritative data. Specifically, we compare Figure 6.5 to the same plot using data collected at the authoritative name servers for all recursive resolvers that send at least five queries during one measurement.

The two graphs are basically equivalent, suggesting that middleboxes do not significantly distort what we see at the clients.

Because of the use of these middleboxes we refrain from trying to identify the implementations of the recursive resolvers directly. Our VPs (RIPE Atlas probes) are unevenly distributed around the globe, with far more in Europe than elsewhere [173], [196], [197].

To take this uneven distribution into account when we study geographic effects, we group probes by continent and analyze them individually in most research questions.

We focus on UDP DNS for IPv4, not TCP or IPv6. The majority of our VPs have IPv4 connectivity only [173] (69%) and so fully study of IPv6 does not make sense. However, we verify that our results apply to IPv6 by repeating a subset of our measurements there. We use the VPs capable of IPv6 to query authoritative name servers reachable only via IPv6 addresses and we confirm that, overall, recursive resolvers follow the same strategy when querying via IPv6.

We focus on DNS over UDP because it is by far the dominant transport protocol today (more than 97% of connections for `.nl` [198] and most Root DNS servers [130]).

Finally, our results are based on one service, the country-code (ccTLD) for the Netherlands (`.nl`). Our results are about recursive and authoritative resolvers and are not specific to this domain. We believe our results generalize to other domains (both ccTLDs and general TLDs), but additional study is needed.

6.3.2 Root DNS and TLD Data

We use passive measurements from the DITL (Day In The Life of the Internet) [199], collected on 2017-04-12 at 10 Root DNS letters (B, G and L are missing). We look at the one-hour sample from 12:00 to 13:00 (UTC), since that duration is sufficient to evaluate our claims. By default, most implementations of recursive resolvers do not treat Root DNS servers different from other authoritative name servers.

We also use traffic collected at 4 authoritative servers of the `.nl` ccTLD [200]. For consistency, we use `.nl` traces from the same time slot as of DITL data. We use these data sets to validate our observations from Section 6.3.1. Note that we cannot enforce a *cold cache* condition in these passive measurements such that a recursive could already prefer an authoritative, and RTT data is not available.

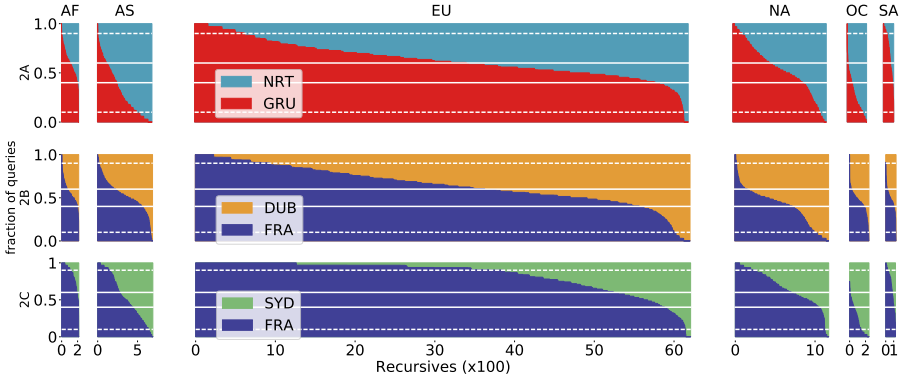


Figure 6.5: Recursive queries distribution for name server combinations 2A (top), 2B (center) and 2C (bottom). Solid and dotted horizontal lines mark VPs with weak and strong preference towards an authoritative.

6.4 ANALYSIS OF RECURSIVE RESOLVER BEHAVIOR

6.4.1 Do recursive resolvers query all name servers?

Our first question is to understand how many recursive resolvers query *all* available authoritative servers. Figure 6.3 shows how many queries, after the very first one, it takes for a recursive to probe all available authoritative name servers (2 to 4 depending on the configuration from Table 6.1).

The percentage of recursive resolvers that query all available authoritative name servers is given in the x-axis labels of Figure 6.3. Most recursive resolvers query all authoritative name servers (75 to 96%), and with two authoritative name servers (2A, 2B, 2C) half the recursive resolvers probe the second authoritative already on their second query; but with four authoritative name servers (4A, 4B) it takes a median of up to 7 queries for the recursive resolvers to query them all. Operators can conclude that all their authoritative name servers are visible to most recursive resolvers.

6.4.2 How are queries distributed per name server over time?

Since most recursive resolvers query all available authoritative servers relatively quickly, we next look at how queries are spread over multiple authoritative name servers, and if this is affected by RTT. Here, our analysis starts once each recursive reaches a hot-cache condition by querying all authoritative name servers at least once.

Figure 6.4 compares the fraction of queries (bottom) received by each authoritative with the median RTT (top) from the recursive resolvers to that authoritative. We see that authoritative name servers with lower RTTs are often favored; *e.g.,* FRA has the lowest latency (51 ms) and always sees most queries overall.

When running multiple authoritative servers, the operator should expect an uneven distribution of queries among them. Servers to which clients see shorter RTT will likely receive most queries.

Our findings in this section, and in Section 6.4.1, confirm those of previous work by Yu *et al.* [187], in which authors show that 3 out of 6 recursive implementations are strongly based on RTT. However, unlike the previous work, our conclusions are drawn from real-world observations instead of experimental setup and predictions based on algorithms.

6.4.3 How do recursive resolvers distribute queries?

We now look at how individual recursive resolvers in the wild distribute their queries across multiple options of authoritative name servers.

Figure 6.5 shows the individual preferences of recursive resolvers (VP/recursive pair, grouped by continent) when having the choice between two authoritative name servers. The x-axis of Figure 6.5 displays all recursive resolvers, and the y-axis gives the fraction of queries every recursive sends to each authoritative. Table 6.2 summarizes these results.

In order to quantify *how many* recursive resolvers are actually RTT based, we consider only VPs that experience a difference in median RTT of at least 50 ms between the authoritative name servers.[2] Based on our observations we define two thresholds for recursive preference: a *weak* preference if the recursive sends at least 60% of its queries to one authoritative (solid lines in Figure 6.5), and a *strong* preference if at least 90% of queries go to one authoritative (dotted lines in Figure 6.5).

We see that 61% of recursive resolvers in 2A (top), 59% in 2B (center) and 69% in 2C have at least a weak preference; and 10%, 12% and 37% have a strong preference in 2A, 2B, and 2C respectively. We show in Figure 6.6 that recursive resolvers with a weak preference develop a stronger preference the longer they query the authoritative name servers. Here we see that after sending queries for 30 minutes, recursive resolvers with a weak preference develop an even stronger preference.

The distribution of queries per authoritative is inversely proportional to the median RTT to each recursive. The bottom plot of Figure 6.5 clearly shows this point, where there is a strong bias for VPs in Europe (EU): VPs largely prefer FRA (Frankfurt) over SYD (Sydney); and the opposite for VPs in Oceania (OC): SYD over FRA.

By contrast, when given a choice between two roughly equidistant authoritative name servers, there is a more even split. We see a roughly even split both when the recursive resolvers are near, with Europe going to Frankfurt and Dublin (configuration 2B, EU to FRA and DUB), or far, where they go to Brazil and Japan (configuration 2A, EU to GRU and NRT). Some VPs still have a preference; we assume these represent VPs in Ireland or Germany. Thus, DNS operators can expect that the majority of recursive resolvers will send most queries to the fastest responding authoritative.

[2] We think that it is reasonable for a recursive to prefer an authoritative over another when it responds at least 50 ms faster.

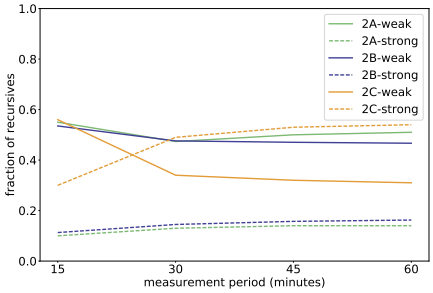


Figure 6.6: Fraction of recursive resolvers that have a weak (solid line) or a strong (dotted line) depending on the length of the measurement

config:	**2A**				**2B**				**2C**			
cont-	**NRT**		**GRU**		**FRA**		**DUB**		**FRA**		**SYD**	
ient	%	RTT	%	RTT	%	RTT	%	RTT	%	RTT	%	RTT
AF	39	467	61	393	57	200	43	204	85	200	15	513
AS	70	130	30	353	53	241	47	261	54	200	46	193
EU	37	310	63	248	65	39	35	53	83	39	17	355
NA	46	190	54	173	41	162	59	152	66	149	33	237
OC	74	201	26	363	46	346	54	335	22	370	78	48
SA	27	364	73	102	49	259	51	259	70	258	30	399

(AF: Africa, AS:Asia, EU: Europe, NA: North America,
OC: Oceania, SA: South America)

Table 6.2: Query distribution and median RTT (ms) for VPs grouped by continent and three different combinations of authoritatives (Table 6.1).

However, a significant share of recursive resolvers (in case of 2B up to 41%) also send up to 40% of their queries to the slower responding authoritative.

To expand on this result, Figure 6.7 compares the median RTT between VPs that go to a given site and the fraction of queries they send to that site, again grouped by continent. Differences between the two points for each continent indicate a spread in preference (differences in queries on the y axis) or RTT (differences in the x axis). We show the results for 2B because in this setup, both authoritative name servers are located rather close to each other such that the VPs should see a similar RTT for both of them. We see that recursive resolvers in Europe that prefer Frankfurt do so because of lower latency (EU VPs that prefer FRA have 13.9 ms lower latency than DUB). In contrast, recursive resolvers in Asia distribute queries nearly equally, in spite of a similar difference in latency (AS VPs see 20.3 ms difference). We conclude that *preferences based on RTT decrease when authoritative name servers are far away* (when they have large median RTT, roughly more than 150 ms).

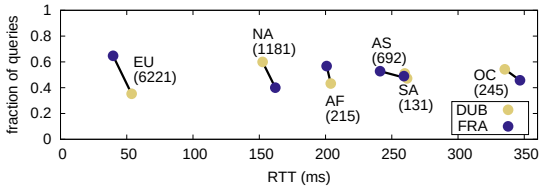


Figure 6.7: RTT sensitivity of 2B (number of VPs in brackets)

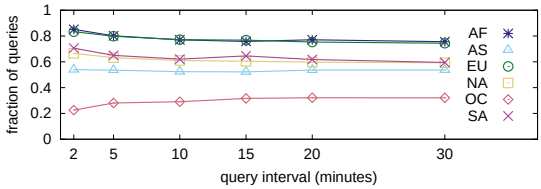


Figure 6.8: Fraction of queries to FRA (remainder go to SYD, configuration 2C), as query interval varies from 2 to 30 minutes

As a consequence, DNS operators who operate two authoritative name servers close to each other can expect a roughly equal distribution from recursive resolvers further away and a preference from recursive resolvers closer by.

6.4.4 How does query frequency influence selection?

Many recursive resolvers track the latency to authoritative name servers (Section 6.2), but how long they keep this information varies. By default, BIND [191] caches latency for 10 minutes, and Unbound caches it for about 15 minutes [190]. In this section, we measure the influence of frequency of queries in the selection of authoritative name servers by the recursive resolvers. To do that, we repeat the measurement for configuration 2C. However, instead of a 2-minute interval between queries, we probe every 5, 10, 15, and 30 minutes. We choose 2C because, in this setup, we observe the strongest preference for one of the two recursive resolvers.

We show these results in Figure 6.8. We see that *preferences for authoritative name servers are stronger when probing is very frequent, but persist with less frequent queries*, particularly at 2 minute intervals. Beyond 10 minutes, the preferences are fairly stable, but surprisingly continue. This result suggests that recursive preference often persist beyond the nominal 10 or 15 minute timeout in BIND and Unbound and therefore, also recursive resolvers that query only occasionally the name servers of an operator can still benefit from a once learned preference.

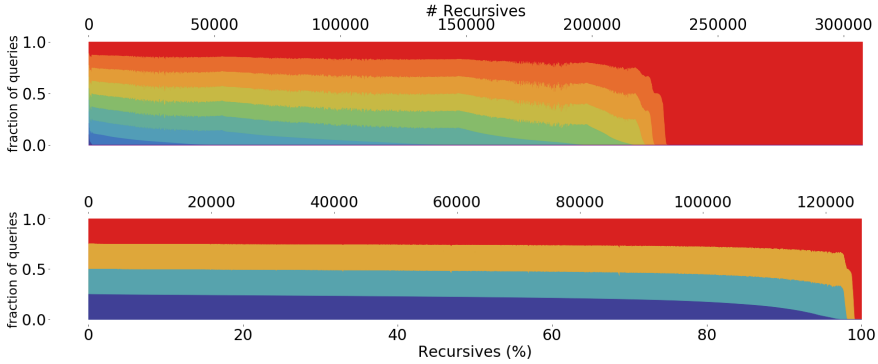


Figure 6.9: Distribution of queries of recursive resolvers with at least 250 queries across 10 out of 13 Root letters (top) and across 4 out of 8 name servers of `.nl` (bottom)

6.5 NAME SERVERS IN PRODUCTION

After analyzing behavior of the recursive resolver for each RIPE Atlas VP in our measurement (Section 6.4), we now focus on validating the results by looking at DNS traffic of production deployments of the Root DNS zone and the `.nl` ccTLD.

Root: We use DITL-2017 [199] traffic from 10 out of 13 Root letters (B, G and L were missing at the point of our analysis) to analyze queries to the root servers (root *letters*). Figure 6.9 (top) shows the distribution of queries of recursive resolvers that sent at least 250 queries to the root servers in one hour. For each VP, the top color band represents the letter it queries most, with the next band its second preferred letter, etc.

While we find that almost all recursive resolvers tend to explore all authoritative name servers (Section 6.4.1), many recursive resolvers (about 20%) send queries to only one letter. The remainder tend to query many letters (60% query at least 6), but only 2% query all 10 authoritative name servers. One reason this analysis of Root traffic differs from our experiment is that here we cannot "clear" the client caches, and most recursive resolvers have prior queries to root letters.

The `.nl` ccTLD: the picture slightly changes for queries to a ccTLD. In the bottom plot of Figure 6.9 we plot the distribution of `.nl` authoritative name servers. The majority of recursive resolvers query all the authoritative name servers which confirms our observations from our test deployment. Here, the number of recursive resolvers that query only authoritative name servers is also smaller than at the Root servers.

We conclude that recursive behavior at the Root and at a TLD is comparable with our testbed, except that a much larger fraction of resolvers have a strong preference for a particular Root letter. The majority of the recursive resolvers send queries to every available authoritative.

6.6 RELATED WORK

To the best of our knowledge, this is the first extensive study that investigates how authoritative server load is affected by the choices recursive resolvers resolvers make.

The study by Yu *et al.* [187] considers the closely related question of how different recursive resolvers choose authoritative name servers. Their approach is to evaluate different implementations of recursive resolvers in a controlled environment, and they find that half of the implementations choose the authority with lowest latency, while the others choose randomly (although perhaps biased by latency). Our study complements theirs by looking at what happens in practice, in effect weighing their findings by the diverse set of software and latencies seen across the 9,000 vantage points, and by all users of the Root DNS servers and `.nl` ccTLD.

Kührer *et al.* [201] evaluates millions of general open recursive resolvers resolvers. They consider open recursive response authenticity and integrity, distribution of device types, and their potential role in DNS attacks. Although similar to our work, they focus on external identification and attacks, not "regular" recursive use. (Using open recursive resolvers in our study for additional measurements is possible future work.)

Also close to our work, Ager *et al.* [202] examine recursive resolution at 50 ISPs and Google Public DNS and OpenDNS. Our study considers many more recursive resolvers (more than 9,000 locations in RIPE Atlas), and we focus on the role those recursive resolvers have in designing an authoritative server system.

Schomp *et al.* [26] consider the client-side of recursive resolvers. Unlike our work, they do not discuss implications for DNS operators. In another work, Korczyński *et al.* [203] have identified second-level domains in the wild whose authoritative DNS servers vulnerable to zone poisoning through dynamic DNS updates [204]. While their work analyzes authoritative servers, it focus on the management of zone files, while we focus on how recursive resolvers choose authoritative name servers.

Finally, other studies such as Castro *et al.* [128] have examined DNS traffic at the Root DNS servers. They often use DITL data (as we do), but typical focus on client performance and balance of traffic across the Root DNS servers, rather than the design of a specific server infrastructure.

After we published our study in 2017, Akamai, an operator of a large Content Delivery Network (CDN), confirmed our results independently [205].

6.7 CONCLUDING REMARKS

In this chapter we have demonstrated that the majority of recursive resolvers distribute their queries across all available name servers. How queries are distributed often depends on the time it takes resolvers to reach the individual name servers.

General remarks: For every domain name operator, this means that when optimizing user latency, *worst-case latency will be limited by the least anycast authorit-*

ative. The implication is that if some authoritative name servers in a server system are anycast, *all* should be. We have shown that most recursive resolvers will always send some queries to all authoritative name servers of a service. Even if one or some authoritative name servers employ large anycast networks for low latency, recursive resolvers will still send some queries to the remaining unicast sites, which implies higher latency. These unicast sites might respond with a short RTT to some clients nearby, but not to clients that are further away and that could be served by other (anycast) sites faster.

While it may seem obvious that all authoritative name servers should have equal capacity, the importance of this relationship is not always clear when making deployment decisions. A DNS operator may seek to improve latency by adding an additional authoritative provided by a large, third-party DNS provider to their current operations, yet not get full value if the two authoritative name servers have different capacity.

At the time of this study, SIDN operated `.nl`, and for SIDN this principle suggested adjusting their architecture. `.nl` relied on 5 unicast authoritative name servers in the Netherlands, and three authoritative name servers that were anycast with sites around the world. Although the anycast authoritative name servers was able to offer lower latency to users from North America, 23% of incoming queries to the unicast name servers in the Netherlands were from the U.S. [198], experiencing worse latency than necessary. Based on our findings, the operators of `.nl` decided to move to a setup in which every name server is distributed with anycast.

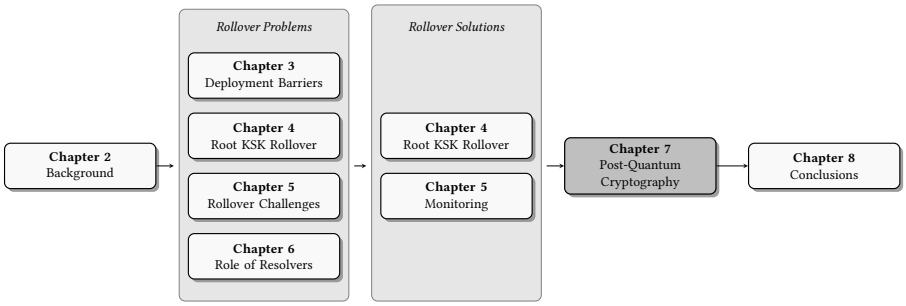
Remarks on DNSSEC: For domain name operators of signed zones, our findings show how DNSSEC can be sensitive to factors, seemingly unrelated to DNSSEC itself. If operators would decide to carry out an algorithm rollover following the liberal approach[3], then the publication delay between different authoritative name servers can be one of the reasons why "strict" resolvers might fail validating the zone. This risk is not negligible, considering that the majority of resolvers spread their queries across multiple name servers.

When following the liberal approach of an algorithm rollover we recommend operators to keep the publication delay as small as possibles, or, if time allows, to follow the conservative approach. In both cases, our monitoring method proposed in Chapter 5 can help operators to assess the impact of their rollover at resolvers and time their rollover accordingly.

[3]Thus, publishing the new keys and signatures at the same time.

CHAPTER 7

Preparing DNSSEC for Quantum-Safe Cryptography



In the final study of this thesis we look ahead, with the threat of quantum computers on the horizon. In the previous chapters, we have shown that algorithm rollovers are complex and one of the major problems that hinder the deployment of algorithms currently standardized for DNSSEC. Also, we have proposed measures to simplify algorithm rollovers. Now, the question remains if DNSSEC is ready to also roll out quantum-safe algorithms. In this chapter, we assess whether quantum-safe algorithms, that are currently evaluated by the cryptography community, can be applied in DNSSEC as well. These algorithms have, in general, different attributes than the algorithms we use in DNSSEC at the moment, like larger keys or signatures. We want to know which problems, besides the rollover itself, we have to face if we transition to these quantum-safe algorithms and discuss how we could address them. The study, on which this chapter is based, has been published in a peer reviewed journal [21]. Also, we presented our findings at several conferences and workshops, e.g. to the operator community at DNS-OARC [206] and the broader DNS community at an ICANN meeting [207].

7.1 INTRODUCTION

As we describe in Section 2.3 of Chapter 2, quantum computing has the potential to solve some computational problems that are currently considered infeasible for existing computers. More concrete, with Shor's algorithm [37], future quantum computers can break *current* cryptographic algorithms such as RSA or Elliptic Curve Cryptography (ECC) in polynomial time, rendering them unusable. Today, many applications rely on these algorithms to provide message confidentiality and integrity, and authentication of the parties communicating and DNSSEC is one of them.

Although a sufficiently powerful quantum computer that can break current public-key cryptography is not available yet, the field of quantum computing is evolving rapidly [11] and quantum algorithms that can be used to break cryptography are also being improved [40]. *Quantum-safe* algorithms are expected to neither be broken efficiently by today's computers nor by quantum computers. Even though experts expect it to take at least another 15 to 30 years before the first quantum computers could break traditional algorithms [13], it is necessary to start transitioning already. We have shown in Chapter 3, that many years are needed to complete such a transition. Since it is difficult to estimate the speed at which quantum computers will be developed, it is prudent to start as early as possible.

The National Institute of Standards and Technology (NIST) has initiated a process to test and standardize quantum-safe algorithms. Currently, four key encapsulation and three signing algorithms are evaluated in the third round of the process and are considered for standardization [208]. NIST expects to select candidates for standardization by early 2022. These quantum-safe algorithms differ in required computational resources for key generation, signing and validation, sizes of keys and signatures, as well as achieved security levels (we list these algorithms and their attributes in Table 7.3). From this it becomes clear, there will not be a single solution that fits all applications, establishing a need to examine which *quantum-safe* algorithms meet the requirements of existing security protocols.

In this chapter, we discuss parameters to assess the readiness of DNSSEC. DNSSEC has not been assessed in the context of post-quantum cryptography before and for which concerns have been raised about the transition to PQC [209].

DNSSEC has strict constraints on *(i)* message size and *(ii)* signature validation and generation throughput, both of which are challenges for many of the proposed quantum-safe algorithms. We analyze which algorithms *could*, at least partially, meet these requirements and propose potential changes to the DNSSEC protocol that can help find a middle ground between constraints of the quantum safe algorithms and of the protocol. Thereby, we take the first steps to prepare DNSSEC for post-quantum cryptography.

These steps could also be applied to protocols with similar constraints and that rely on the same underlying transport protocol as DNSSEC (e.g. certain encapsulations of the Extensible Authentication Protocol (EAP) [210]).

7.2 RELATED WORK AND APPROACH

This is the first study that analyses the applicability of quantum-safe algorithms for protocols with strict constraints on signature length, focusing on message authentication, and the first that studies this for DNSSEC. Related work from Crockett et al. [211] applies quantum-safe algorithms to TLS and SSH and Heesch et al. [212] apply them to OpenVPN and HTTPS. None of these protocols, however, have the same constraints as DNSSEC. Van Rijswijk-Deij et al. [158] evaluate the performance of Elliptic Curve Cryptography in DNSSEC, but using PQC imposes additional size-requirements.

For our research, we derive the requirements of DNSSEC from standards [10], [57], [102], community best practices [213], our own active measurements covering a daily snapshot of the DNS for a representative set of over 220M domain names [62], and operational experience from running the Dutch ccTLD `.nl`.

For this study, we consider quantum-safe signing algorithms that are part of the third round of the NIST standardization process [208]. We consider both finalist and candidate algorithms – seven in total. Table 7.3 shows the key and signature sizes for each algorithm with estimated security level I [214, §4.A.5]. We first select candidate algorithms based on the signature size, meeting the requirements of DNSSEC explained in Section 7.4. Then, we measure the performance of the selected algorithms, using their optimized implementation as provided on the NIST website [215]. For each, we measure how many signatures we can create and verify in 10 seconds for a random message, repeat this 1,000 times, and report the mean performance. We choose a random 86-byte string as message to sign.[1] All measured algorithms rely on current hash functions (SHA256, SHAKE-256 and SHA3) to transform the signed records into a string with standard length. Therefore, the record size only affects the performance of the established hash-function and not the performance of the new signing algorithm itself. We perform the measurement on a single core of a machine equipped with an Intel Xeon Silver 4110 CPU (2.10GHz), 64GB RAM, running Ubuntu 18.04.3 LTS. In the interest of reproducibility, we make our measurement code public [216].

Implementations will likely be further optimized. For this reason, the performance metrics are only a rough estimate. The selection of suitable algorithms for DNSSEC will therefore mostly be based on key and signature size, which are inherent properties of the algorithms unlikely to change in the future.

We compare the record sizes and performance metrics with current algorithms that are commonly used or recommended for DNSSEC [47]: *RSA-2048* belonging to the most popular algorithm family in DNSSEC [62], *ECDSA-P256*, an elliptic curve algorithm, widely deployed because of its small signatures, and *EdDSA-Ed22519* an algorithm based on Edwards Curves which the IETF expects to become the future recommended default for DNSSEC [47]. We benchmark these reference algorithms using the integrated OpenSSL speed test on the same hardware as the PQC algorithms.

[1] The median number of bytes covered by an `RRSIG` [102] of all signed `AAAA` records of domains in `.com` [62].

7.3 POST-QUANTUM CRYPTOGRAPHY

Post-quantum cryptography (PQC) is the group of algorithms that run on a classical computer and can withstand both a conventional and a quantum computer's attack. The 26 years since the invention of Shor's algorithm [37] dictate the time scale of main developments in the field. This is much less time than has been spent, e.g., on cryptanalysis of conventional public key algorithms such as RSA and ECC. For this reason, current standardization efforts (such as the NIST competition [208]) focus on standardizing quantum-safe algorithms based on multiple different mathematical problems. In this section we summarize some of the different approaches to PQC. Regarding the security of these algorithms, we note that many factors play a role in the cryptanalysis of new algorithms and discussing these in detail is out of scope for this document. Nevertheless, depending on the approach PQC schemes take, general observations can be made based on the current state of research.

There are currently five classes of PQC algorithms. Three of these (lattice-based, multivariate and hash-based) are considered for signature schemes in the NIST competition as finalist or alternate candidates [208] and we assess all three (see also Table 7.3). The specific algorithms chosen are the security level I variants, which corresponds in strength to a 128-bit classical key search [214]. This is equivalent in strength to commonly used 256-bit ECC keys and stronger than the current standard RSA-2048, which measures 112 bits in classical security [217] and is widely used for DNSSEC.

The *multivariate* approach (1980s) is based on systems of algebraic quadratic equations over finite fields. Typically, signature schemes are given by underdefined systems, meaning that there are several valid signatures for a public key. This is no problem as long as it is sufficiently difficult to find another valid signature. Although improvements to several attacks have been found recently [218]–[220], multivariate schemes have a good security track record. Generally, they have small signatures with fast verification.

***Lattice-based* cryptography** [221] (1996) builds on the hardness of finding short vectors in a high-dimensional lattice. It used to be impractical, but provably secure, or practical but with a security reduction. Newer schemes combine these and some have submitted provably secure signature schemes to the NIST standardization, such as qTesla-p-I. Like many cryptographic algorithms, they are vulnerable to side-channel attacks [222]–[224]. However, for DNSSEC this is not a major concern, since these attacks require physical access to signers. In general, lattice-based systems form good and allround algorithms with relatively small signatures and keys, combined with fast operations.

***Hash-based* signature schemes** build on the property that it is hard to find a pre-image (input message) for a certain digest or to find two elements with the same digest. A large advantage of these schemes is the solid security basis that only depends on the security of the chosen cryptographic hash function. For this reason, hash-based schemes are considered extremely conservative alternate candidates for standardization [208]. Typically, hash-based signature schemes have

Response Type	RRs in response	RRs added by DNSSEC (covered RR)	Alexa 1M median TTL (mean)
AAAA	≥ 1 AAAA	1 RRSIG (AAAA)	5 min (0.6 h)
DNSKEY	≥ 1 DNSKEY	1 RRSIG (DNSKEY)	60 min (8.3 h)
Non-existent domain (with NSEC)	SOA	1 RRSIG (SOA) 2 NSEC 2 RRSIG (NSEC)	60 min (2.0 h)
NSEC3 Closest-encloser proof (§5.5 of [225])	SOA	1 RRSIG (SOA) ≥ 3 NSEC3 ≥ 3 RRSIG (NSEC3)	10 min (2.8 h)

Table 7.1: Records added by DNSSEC and the median time they are cached of the 1M most popular domains [226]

very small keys, large signatures and require significant computational overhead for signing and verification.

7.4 DNSSEC REQUIREMENTS

In this section we define requirements that modern DNSSEC set-ups demand from cryptographic algorithms. DNSSEC adds additional payload to DNS messages and requires additional computational operations. The choice of cryptographic algorithms has an influence on both.

7.4.1 Additional Payload and Validation Frequency

As described in Section 2.2.2 of Chapter 2, DNSSEC adds a signature to the DNS response of the requested record. In some cases even multiple signatures are transmitted: If the requested information does not exist, then DNSSEC-signed zones provide the resolver with an *authenticated denial of existence* (NSEC(3)). The details of this proof are out of scope, but the response can contain three or more signatures [50], [102] (two bottom rows of Table 7.1).

When an operator uses multiple keys to sign a zone, a signature is attached for each key. This is for example the case during a key rollover, replacing one key with another. Also, operators can sign their zone with different algorithms, resulting in multiple signatures for each used algorithm and two or more DNSKEY records. Also, since most zones split between a ZSK and a KSK, queries for the public key contain both keys along with the signature (resulting in large messages).

The result of the validation is *cached* as defined by the time-to-live (TTL) field in the signed RR. Only after the TTL has expired, will the results have to be validated again.

7.4.2 Cryptographic Requirements

The DNSSEC protocol allows adding new cryptographic algorithms relatively easily, and new algorithms have been proposed and integrated numerous times, as shown in Chapter 3. All algorithms, however, must adhere to boundaries and requirements

set by the design and deployment of DNS, DNSSEC and the underlying transport protocols.

Signature and key size DNS packets are limited, in theory, to 64 kilobytes. Previous research and operational experience, however, have shown that sending large DNS packets is often problematic.

First, the maximum transmission unit (MTU) of the underlying networks can be a limiting factor. Packets larger than the MTU cause fragmentation or trigger a retransmission via TCP. In the best case, this causes additional round trip time (RTT) for transmitting the fragments or for establishing the TCP connection. In the worst case, fragments can never be transmitted and the TCP connection cannot be established because of interfering middle boxes or lack of support. As a consequence, end users, for example, are not able to visit their requested website. Van den Broek et al. [121] have shown that up to 10% of all resolvers might be unable to handle fragments.

Second, fragmented DNS responses can be misused to spoof the cache of recursive resolvers [8]. Both two problems, potential packet loss and the susceptibility to spoofing, encouraged DNS software developers and operators to recommend a maximum supported message size of 1,232 bytes [227].

Third, DNS is often misused in amplification attacks, where thousands of small queries from an attacker trigger large responses directed to a victim. The extra records DNSSEC adds to a response make this attack more effective [228]. With the introduction of elliptic curve based algorithms in DNSSEC, the signatures can be up to 64 bytes small, which partially mitigates this problem [94].

These three reasons lead us to conclude that small signatures are also preferred for quantum-safe algorithms and that *signatures* should not exceed 1,232 bytes. Signatures are transmitted in every DNSSEC message, for example every time an `A` or `AAAA` record is returned (around 55% of all queries [229]) or in response to a query for a non-existing record (around 15%). In the latter case, a response will even contain multiple signatures. Also, they are cached the shortest (see Table 7.1). Therefore, it is crucial that signatures are transmitted reliably, without the risk of packets being dropped or retransmitted. Signatures smaller than 1,232 bytes decrease these risks significantly. Preferably, even, signatures are far below this threshold leaving room for payload and multiple signatures. Public keys, on the other hand, need to be transmitted less frequently, so having larger keys may be acceptable. We explore this in Section 7.6.

Validation Resolvers need to serve their clients as fast as possible. A medium size resolver today processes a few thousand queries per seconds resulting in a few hundred validations [158]. This is far below their maximum capacity. The underlying cryptographic libraries can validate thousands of signatures per second of current algorithms used in DNSSEC (see bottom of Table 7.3). The total number of DNSSEC-signed domain names is still rising and large resolvers likely need to validate ever more signatures. Therefore, we expect that at least 1,000 quantum-safe signatures

Prio	Requirement	Good	Accepted Conditionally
#1	Signature Size	≤ 1,232 bytes	—
#2	Validation Speed	≥ 1,000 sig/s	—
#3	Key Size	≤ 64 kilobytes	> 64 kilobytes
#4	Signing Speed	≥ 100 sig/s	—

Table 7.2: Requirements for quantum-safe algorithms

should be validated per second in our evaluation. This is a conservative boundary and we can expect that future implementations and specialized hardware will also speed up post-quantum algorithms.

Signing Zone operators sign records on five different occasions: (i) when the zone is signed for the first time, (ii) when the key is changed (*rolled*), (iii) when records change, (iv) when a signature expires or, (v) *on-the-fly*. The latter, obviously, is the most time critical. In this approach, signatures are created when a record is queried. This is for example necessary when records are created dynamically depending on the querying resolver and requires signing in milliseconds. This setup is usually only used at CDNs (e.g. Cloudflare [230]); typical operators only re-sign records when they change or when a key rollover takes place. The frequency depends on the zone. Zones of top-level-domains like `.com` and `.nl` change frequently. E.g. every time a new domain is registered new records need to be signed. For *.nl*, zone files are published every 30 minutes, typically requiring around 11,000 new signatures to be created.

To support signing of larger zones, frequent zone file publication, and additional overhead, suitable quantum-safe algorithms must at least be capable of creating 100 signatures per second. Slower algorithms might be acceptable for zones that are less prone to change. For on-the-fly signing, obviously, higher signing speeds are required.

Requirements summary The size of signatures is the most important criterion when selecting an algorithm, followed by the time it takes to validate signatures. Only if signatures can be transferred reliably between name server and resolver and the resolvers can validate the signatures timely, the basic protocol of DNSSEC can stay unchanged. The requirements are summarized in Table 7.2. The third column shows the requirements that we expect algorithms to fulfill and which are marked in blue. Under some circumstances or with some modification of the DNS protocol higher boundaries might be acceptable. These are listed in the last column, marked in orange and are discussed in Section 7.6.

Algorithm	NIST Verdict	Approach	Private key	Public key	Signature	Sign/s	Verify/s
Crystals-Dilithium-II [231]	Finalist	Lattice	2.8kB	1.2kB	2.0kB		
Falcon-512 [232]	Finalist	Lattice	57kB	0.9kB	0.7kB	3,307	20,228
Rainbow-I_a [233]	Finalist	Multivariate	101kB	158kB	66B	8,332	11,065
RedGeMSS128 [234]	Candidate	Multivariate	16B	375kB	35B	545	10,365
Sphincs$^+$-Haraka-128s [235]	Candidate	Hash	64B	32B	8kB		
Picnic-L1-FS [236]	Candidate	Hash	16B	32B	34kB		
Picnic2-L1-FS [236]	Candidate	Hash	16B	32B	14kB		
EdDSA-Ed22519 [237]		Elliptic curve	64B	32B	64B	25,935	7,954
ECDSA-P256 [237]		Elliptic curve	96B	64B	64B	40,509	13,078
RSA-2048 [237]		Prime	2kB	0.3kB	0.3kB	1,485	49,367

Table 7.3: Signature algorithms in round three of the NIST competition [208] (security level I). DNSSEC candidate algorithms are shaded gray. Attributes meeting DNSSEC's requirements fully or partially are marked blue or orange, others in pink.

7.5 EVALUATING ALGORITHMS

The previous section shows that signature size is the most crucial requirement. We mark the attributes of algorithms that fully or partially fulfil each requirement in blue or orange respectively and use this encoding also in Table 7.3. Attributes that do not fulfil the requirements are marked in pink. We pre-select *aspirant* algorithms that create signatures ≤ 1,232 bytes. This leaves us with three algorithms: Falcon-512, RedGeMSS128, and Rainbow-I_a (marked light gray in Table 7.3). For those, we additionally evaluate signing and validation performance.

The remaining algorithms create signatures larger than 1,232 bytes. Their reliable transmission cannot be guaranteed and they make DNSSEC more attractive as an amplifier in a Distributed Denial of Service (DDoS) attack. For this reason, we do not consider them for DNSSEC any further.

Falcon-512 Falcon [232] is a signature scheme based on NTRU-lattices [238]. It stands out as a computationally efficient algorithm, with an optimized implementation already available. Falcon-512 has the smallest pair of public key and signature, which is particularly relevant for the DNSSEC case. It is the only algorithm where both signatures and public keys fall within the size limit, although both keys and signatures are considerably larger than current non-PQC DNSSEC algorithms. This may still cause problems during transmission, since it is neither possible to ship more than one key at a time, nor to ship more than one signature, or even only one signature and a payload that exceeds 523 bytes. In our test-bed, the performance of Falcon-512 is closest to the current algorithms and meets the requirements of DNSSEC. Further performance improvements are possible using a hardware FPU, AVX2 and FMA opcodes [232].
Its implementation and level-I security strength are delicate; conversely more testing is required to gain trust in its security. NIST currently expects either Falcon or Crystals-Dilithium to be standardized as the primary post-quantum signature scheme at the conclusion of the third round [208].

Rainbow-I_a Rainbow-I_a [239] is a multivariate scheme. It is based on the Unbalanced Oil and Vinegar (UOV) scheme [240]. The signature size of Rainbow-I_a matches the sizes of current recommended algorithms based on elliptic curves and is therefore a good fit for DNSSEC. The public keys, however, are significantly larger and do not fit in DNS packets. As with the signature size, the performance of Rainbow-I_a is comparable to current algorithms and meets the requirements. Its performance can be improved further with AVX2 instructions. A version with a reduced public key size is Cyclic Rainbow, but this comes at the cost of an increase in computational requirements. We note that the adoption of Rainbow-I_a could be hindered by royalties [241].

RedGeMSS128 GeMSS [242] is a multivariate signature scheme of the Hidden Field Equation type. RedGeMSS128 produces the smallest signatures in the GeMSS

family, at security level I even smaller than EdDSA-Ed22519. The public key, however, exceeds the maximum record size of the DNS. First measurements indicate GeMSS signs considerably slower than current algorithms. The usage of SSE2, SSE3 and the AVX2 CPU instructions could improve performance [242]. If new insights show that Rainbow is unacceptable, GeMSS forms an alternate candidate for standardization [208].

7.6 DISCUSSION

The previous section shows that no algorithm fits all requirements perfectly. Falcon, Rainbow and GeMSS come closest, but each has shortcomings: Falcon-512 *technically* meets all requirements but its larger signatures may cause problems, e.g. during rollovers, and make DNSSEC an even more attractive tool for DDoS attacks. In comparison, signatures of Rainbow-I_a and RedGeMSS128 are on par with current recommended algorithms, but their public keys go beyond the supported payload size. All algorithms perform signing and validation fast enough for today's use cases.

We therefore expect changes to the DNSSEC protocol are required before PQC algorithms can be deployed. We now sketch what changes may be needed, setting an agenda for future research.

7.6.1 Increased TCP Support

The greatest bottleneck to deploying Falcon-512 is the large size of keys and signatures. Operators can reduce the size of the key set by relying on CSKs (Combined-Signing-Key – combining the ZSK and KSK), but signed messages might still exceed the threshold of 1,232 bytes in case of larger payloads or if multiple algorithms are used. Nevertheless, keys and signatures could still be safely transmitted using TCP. Today, not every name server supports TCP: we still observe 11% of name servers lacking TCP support [62]. Two developments, however, could help decrease this.

DNS Flag Day [213] is a recurring initiative by software vendors and operators. In 2020, it promotes, among others, the support of TCP. The previous flag day, promoting the support of EDNS, had a positive impact [243], and we expect the same for the upcoming. Also, *encrypted* DNS could increase TCP support. DNS-over-TLS (DoT) [29] and DNS-over-HTTPS (DoH) [30] both rely on TCP as transport and see some traction already [244]. TCP mitigates the threat of DDoS amplification attacks but requires more resources at recursive resolvers and name servers and its impact still needs to be thoroughly measured.

7.6.2 Out-of-band Key Distribution

Increased TCP support is still not sufficient for transmitting the *public key* of Rainbow and GeMSS, since both exceed the maximum DNS payload size of 64 kbytes [245].

This problem can be solved in two ways, both modifying the existing `DNSKEY` RR. One approach is to divide the public key into chunks small enough that they can be

transmitted in one RR. Each chunk is published at a new label of the signed domain and chained with each other. The initial DNSKEY RR would then refer to the first chunk of the actual public key. The advantage of this approach is that it can likely be implemented in a manner that is backward compatible to existing implementations. The disadvantage, however, is that resolvers would need to send multiple queries to fetch a key, increasing the risk of transmission failures.

Alternatively, we propose transmitting the key *out-of-band*. Instead of directly providing the resolver with the key through a DNS record, name servers could serve a URI, instructing the resolver to fetch the key from a web server using HTTP. Because of the *chain of trust*, resolvers can still use the public key of the root to verify the key published on the web server. Resolvers not supporting this mechanism would already today either consider the zone not signed (*insecure*) or fall back to a supported algorithm if the zone is signed with multiple algorithms. Because of the higher TTL (see Table 7.1), an out-of-band transmission of DNSKEY RRs would only occur occasionally. This approach comes with two caveats: first, resolvers would need to support HTTP to fetch the key and second, zone-operators would need to maintain a web server. Whereas the former might be addressed by the rise of DoH (see previous section), the latter might be an additional barrier for operators rolling out DNSSEC and could create additional potential points of failure.

7.6.3 Performance

The two aforementioned measures address the challenges of large keys and signatures. If, however, it turns out that the candidate algorithms are not secure and faster hardware not affordable, then it might be necessary to use algorithms *too slow* for current DNSSEC deployments. One workaround might lay in the fact that resolvers only need to validate signatures if the signed record is not cached. If validating signatures is an expensive operation, decreasing the number of validations may be a solution. AAAA records of the 1M most popular domain names have a median TTL of 5 minutes (see Table 7.1). Increasing the TTL of these RRs to 1 hour would already reduce the workload for resolvers 12 times. Note, however that we expect that optimized implementations and specialized hardware could improve performance, rendering higher TTLs unnecessary.

7.6.4 Other Considerations

Algorithms for high-security zones In this paper, we only considered PQC algorithms at security level I (128-bit security). In the future, however, some zones may have stronger requirements. Consider, for example, the DNS root zone, which has very long-lived keys – the root KSK was only changed for the first time 8 years after its introduction [15]. This long lifetime may increase the risk of a successful attack against the key and may thus require choosing schemes with higher security levels. Fortunately, the remaining PQC algorithms in the third round of the NIST competition leave room for this. For example, RedGeMSS256 offers security level V

with very modest signatures (76B). The public key, however, is significantly larger at 3135kB making changes to the way keys are distributed in DNSSEC inevitable.

Alternatives to DNSSEC The measures described above show that quantum-safe algorithms *can* be applied to DNSSEC, but not without additional effort. Therefore, one could propose to abandon DNSSEC altogether and to find other solutions to guarantee authenticity in the DNS. DNS-over-TLS and DNS-over-HTTPS, for example, rely on TLS and HTTPS and earlier studies have shown that both can support quantum-safe algorithms [211], [212]. On the other hand, neither provides a full replacement for DNSSEC and the trust model is different, making it impossible to realize some of the newer applications of DNSSEC such as DANE [246]. Other alternatives, like DNSCrypt, claim to provide quantum-safe implementations [247], but deployment of DNSCrypt has not gained significant traction, and this seems unlikely to change in the future as there is not IETF specification for the protocol.

7.6.5 Transitioning to PQC

Especially in the early days of their development, trust in new PQC algorithms may still be low. Instead of only signing records with a quantum-safe algorithm, a combination of a conventional and a PQC algorithm can be used. This *hybrid* model [248] is especially valuable for long-lived signatures and keys, since there is a greater risk that an attack against one of the algorithms is successful over their lifetime. Our data [62], shows the average lifetime of a signature is around 34 days (median 21 days), making the risk small. Keys, however, can be much longer lived and replacing a crucial key, such as the one for the DNS root, is non-trivial and takes time [15].

The problem with a hybrid model is that it requires signing with two algorithms concurrently. While this is possible within the specifications of the DNSSEC protocol, it doubles the number of keys and signatures. Realistically, this means that such a model can only be deployed within the constraints discussed in Section 7.4 if both the conventional and the PQC algorithm have small signatures. The best combination would in this case be an elliptic curve algorithm (e.g. ECDSA P-256) with either Rainbow-I_a or RedGeMSS128.

7.7 CONCLUDING REMARKS

We have identified three algorithms, currently under evaluation in the NIST competition, that show great potential to be applied in DNSSEC: *Falcon 512*, *Rainbow-I_a*, and *RedGeMSS128*. Falcon, in principle, could even be deployed in DNSSEC without protocol modifications, but still has the shortcoming of significantly larger keys and signatures than current algorithms. To address these and other challenges, we have proposed extensions and modifications that could make DNSSEC ready for quantum-safe algorithms. These algorithms may also be a fit for protocols with similar strict requirements on key and signature sizes.

Nevertheless, we need to keep in mind that standardizing quantum-safe algorithms for DNSSEC and getting them deployed takes time. If NIST standardizes one or more algorithms, they still need to be standardized in the IETF for the use in DNSSEC. As shown in Chapter 3, even for a rather uncontroversial algorithm like EdDSA-Ed22519, this effort took almost a year [54]. Fourteen months after its standardization we see the first resolvers supporting this algorithm, from roughly 10,000 vantage points [249]. Today, and more than two years later, still 30% of observed validating resolvers lack support and more than 99% of 7M signed domains do not use this algorithm [62]. Furthermore, the DNS root is still signed with an algorithm that was standardized more than 10 years ago, and even though its key has been successfully replaced for the first time in 2018 (see Chapter 4) it is still not clear when the algorithm gets updated.

From these experiences we conclude that making DNSSEC fit for quantum-safe algorithms needs to start as soon as possible; the results in this paper can help make a start to this process.

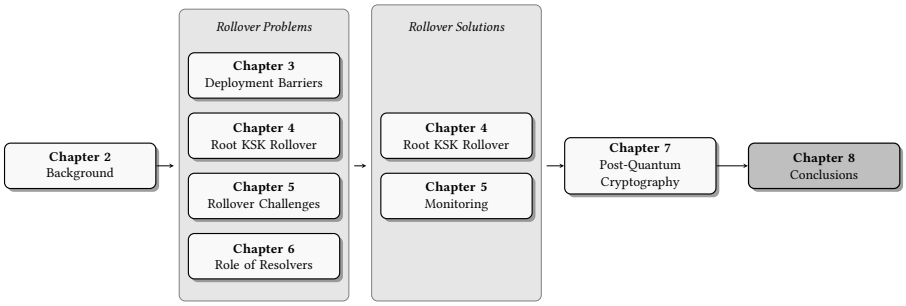
We will continue observing the standardization process and adapt our recommendations if necessary. Already during the course of writing this paper new developments influenced our algorithm selection: the LUOV [250] scheme, which also creates small signatures suitable for DNSSEC, was considered not secure enough anymore after an attack was published [251] and dropped out after the second round. NIST, however, finds its approach still promising and we will assess if future implementations might be a fit for DNSSEC as well.

Also after the publication of our study the developments in the field of post-quantum cryptography did not stand still. By the end of 2020, an improved attack on Rainbow was published [252] reducing the costs of key recovery such that Rainbow would not meet the security requirements of NIST anymore. The authors of Rainbow reject this claim and consider Rainbow still secure enough [253]. After this attack, NIST also raised concerns regarding the lack of diversity among the finalists of the competition [254]. They are now considering to accept other quantum-safe algorithms to the NIST competition as well.

In the next chapter, we conclude this thesis. There, we also discuss potential future research in the field of applying quantum-safe algorithms to DNSSEC.

CHAPTER 8

Conclusions



In the final chapter of this thesis, we revisit the research objective and the research questions we aimed to answer. We discuss, whether we reached our goal and highlight the main contributions of this thesis. Based on our findings, we finally discuss future research directions and propose concrete steps forward.

8.1 MAIN GOAL

The objective of this thesis was:

> *to prepare DNSSEC for quantum computing, by identifying problems when introducing new cryptographic algorithms and developing and evaluating solutions to address them.*

We achieved this goal by applying two consecutive approaches.

First, we *identified barriers* in the DNS and DNSSEC ecosystem that can hinder the wider deployment of new algorithms. In Chapter 3, we measured the standardization and deployment of new algorithms in DNSSEC in the past. Here, we found several problems that need to be addressed before new algorithms become widely deployed and insecure algorithms are replaced. One of the major problems we identified is the lack of confidence of operators to carry out an algorithm rollover, which increases their reluctance for rolling. In Chapter 4 we turned our attention to the root, the top of the DNS hierarchy. Here we found that when replacing the DNSSEC trust anchor, and consequently also introducing a new algorithm in the future, we can face problems when validating resolvers need to retrieve the new key. Especially validating end-user applications and containerized applications were responsible for some of the observed problems. Since we expect that both will become more common in the future, they form a particular barrier for future algorithm rollovers at the root. Before we could develop solutions to address these problems, we first gained a better understanding of the underlying reasons. In Chapter 5 we showed in more detail, which aspects of the DNS make an algorithm rollover complex. We demonstrated that *timing* is most important and in Chapter 6 we showed the behavior of recursive resolvers further increases the complexity. Finally, in Chapter 7, we looked ahead, with the threat of quantum computers on the horizon. Here, we found that the fact that large messages are often not transmitted reliably in the DNS poses the biggest problem when introducing quantum-safe algorithms.

Based on these observations, we *developed and evaluated solutions*. In Chapter 4, we proposed alternatives for distributing root trust anchors more reliably. We believe that by applying this alternative key distribution we can reduce the risk of failing resolvers during a future algorithm rollover at the root. In Chapter 5, we developed and tested a measurement method that gives domain operators more confidence when rolling to a new algorithm thereby making the introduction of new algorithms easier. Last, in Chapter 7, we assessed quantum-safe algorithms. We found candidate algorithms and proposed how we can apply these algorithms in DNSSEC. These measures combined start preparing the way for a secure DNSSEC that can withstand also future attacks on cryptographic algorithms to provide authenticity and integrity in the DNS.

8.2 REVISITING THE RESEARCH QUESTIONS

In this section, we describe in more detail how we have addressed the research questions defined in Chapter 1.

We aimed to answer four sub-research questions, starting with:

> **RQ 1**—*What are the complexities when introducing new algorithms in DNSSEC?*

To answer this research question we carried out two studies. First, in Chapter 3 we analyzed the complete algorithm life cycle of an algorithm in DNSSEC, starting with its standardization in the IETF, continuing with the adoption of a new algorithm in software and in the domain name registration channel, the deployment at domain names and at resolvers, and their deprecation. Here, we found that standardization can seriously slow down adoption, which in some cases can take four years. New algorithms need the support of the community to speed up standardization. Also, important players in the domain registration channel are often slow supporting new algorithms, hindering their deployment at domain names. Only if the market demands the support of new algorithms, also registrars will react.

We found, however, that even if these barriers are overcome, one major barrier remains: the algorithm rollover at a domain name. As an example, at the time of our study only 17% of all domains signed with the modern ECDSAP256 algorithm actually *rolled* their algorithm. The other 83% either were signed for the first time or were newly registered domain names. Also, we occasionally saw that algorithm rollovers were not carried out following standard procedure. Both findings indicate that operators shun algorithm rollovers because of their complexity.

Second, in Chapter 4, we shifted our focus towards the root, which, through its position at the top of the DNS hierarchy, poses an additional challenge when introducing new algorithms. Because the root has not rolled its algorithm yet, we studied the rollover of the KSK, the trust anchor of the DNS, instead. This KSK rollover is a prerequisite for an algorithm rollover, where millions of validating resolvers need to replace their local copy of the root KSK. Here, we found that the replacement of the key at recursive resolvers poses the biggest problem. For example, we found that 8% of resolvers, which signaled their configured trust anchor to the root, did not fetch the new KSK on time and that at least 1% of our observed vantage points failed to validate DNSSEC signatures after the rollover.

From these findings we conclude that the complexity of *algorithm rollovers* poses one of the largest barriers we need to overcome. We wanted to address this barrier but first we needed to understand:

> **RQ 2**—*What makes algorithm rollovers complex?*

We studied this research question in Chapters 4, 5, and 6. In the first chapter we analyzed the root and in the second and third we shifted our attention to TLDs and second-level domain names. Algorithm rollovers at the root are especially complex due to the role of its KSK as the main trust anchor for DNSSEC. As discussed in

Chapter 4, a significant number of resolvers did not pick up the new key. In the same study, we found, that at least 6,000 of them were VPN software which had the old KSK hard-coded in their source code. This, and other findings highlight that the replacement of a trust anchor at recursive resolvers is an error-prone process and needs to be improved.

Then, in Chapters 5 and 6 we studied rollovers especially at TLDs and second-level domain names. Here, resolvers do not need to configure the new key locally, but their behavior must still be taken into consideration when carrying out an algorithm rollover. In Chapter 5, we demonstrated in theoretical scenarios that algorithm rollovers, which are not carefully timed, can lead to resolvers being unable to correctly validate signatures. This exercise was followed by active measurements of the `.se` algorithm rollover. Here, we showed that incorrect timing of a rollover can lead to outages, but that correct timing of an algorithm rollover is complex. For example, 1% of observed resolvers cache records longer than the recommended TTL – something which is not visible to operators of a domain name. We therefore asked:

RQ 3—*What can we do to reduce the complexity of algorithm rollovers?*

The complexity of rollovers at the root and at TLDs has, partially, different root causes. In Chapter 4, we propose measures to distribute the root trust anchor via the operating system. This avoids the error prone key distribution via the RFC 5011 protocol and makes the operating system the only place where the new key needs to be updated (instead of at every application that does DNSSEC validation).

In Chapter 5, we then proposed a measurement method, which gives DNS operators more insight into the timing of a rollover. Also, operators can spot problems during the rollover from the perspective of their clients. We tested the method during the first algorithm rollover of the ccTLD `.se`. This rollover was carried out successfully and the operators of `.se` confirmed that the insights provided by our measurements gave them more confidence when rolling the algorithm. We published software that helps operators to monitor their rollover themselves and the operators of the ccTLDs `.br` (Brazil) and `.dk` (Denmark) used it to successfully monitor their algorithm rollover.

Both solutions reduced the complexity of rollovers for *current* signing algorithms. *Quantum-safe* algorithms, which can also withstand attacks of a quantum computer, however, have different properties than algorithms currently standardized for DNSSEC. We thus asked:

RQ 4—*Are there quantum-safe algorithms suitable for DNSSEC?*

In Chapter 7 we first defined the requirements that every algorithm needs to fulfill that should be standardized for DNSSEC. Based on earlier studies and our own measurements, we concluded that especially the size of signatures is crucial. If signatures become larger than 1,232 bytes the chance of transmission failures increases significantly. Then, we assessed quantum-safe algorithms, that were part of the third round of the NIST competition and found that especially Rainbow-I_a

and RedGeMSS128 are suitable algorithms for DNSSEC. Both, however, also have public keys that are too large to be transmitted in a DNS message. To address this challenge, we discussed the possibility of transmitting public keys out of band.

Now, we have shown that quantum-safe algorithms exist that, with some limitations, are suitable for DNSSEC, and it is time to revisit our main research question:

Main RQ—*Is the DNS ecosystem ready for algorithm rollovers?*

In this thesis, we have shown that, broadly speaking, *the DNS ecosystem is ready for an algorithm rollover.* The first root KSK rollover was a success and lays the foundation for a successful algorithm rollover as well. The remaining open questions can be addressed, as we will discuss in Section 8.3.1 below. Here, we recommend that keys are managed centrally by the Operating System (OS) which can reduce the remaining issues during key distribution.

Also, the challenges of algorithm rollovers at other parts of the DNS ecosystem are manageable. By providing a monitoring tool, we have reduced the complexity of algorithm rollovers. In combination with automated rollovers, which many DNS software now support, algorithm rollovers can be carried out with less effort than a decade ago.

Finally, the DNS is also ready for a future where quantum computers threaten the security of the algorithms currently used in DNSSEC. We have shown that algorithms exist that are *quantum-safe* and that can be applied in DNSSEC as well. These new algorithms will not be a drop-in replacement for current algorithms but the required extensions to the DNS seem feasible. Developing and testing these extensions is future work, which leads us to the final section of this thesis.

8.3 OUTLOOK AND FUTURE RESEARCH

In the final section of this thesis we propose possible future research directions. Future research is necessary to achieve the goal of a fully quantum-safe DNS ecosystem. We believe, based on the findings in this thesis, that especially an algorithm rollover at the root and the implementation of quantum-safe algorithms require more attention.

8.3.1 Algorithm Rollover at the Root

So far, the root has not rolled its algorithm yet. In Chapter 4, we have shown that the root is in principle ready to roll to a different algorithm, since also the root KSK rollover was successful. Despite this, especially two more aspects need our attention before the DNS community will consider the root to be ready for an algorithm rollover.

Understanding Algorithm Support at Resolvers Moving to a different algorithm must not increase the possibility of a successful cache poisoning attack against resolvers and their clients. For this reason, we need to understand which algorithms

are supported by resolvers before deciding when to roll and to which algorithm. Measurement platforms like RIPE Atlas and ad-based studies only provide a limited view or are not transparent. Extended resolver telemetry, similar to RFC 8145 [113] and Root Sentinel [132], where resolvers report algorithm support could provide more insights. But similar to these approaches, we face the issue that we would not know which resolver is relevant enough to potentially postpone an algorithm rollover, and metrics are susceptible to manipulation. An additional metric where resolvers indicate the population of their client base could help identifying relevant resolvers.

With these additional metrics, the community and root zone operators could have enough information to decide when to roll an algorithm, even though we still might not have the whole picture.

Distributing the DNSSEC Trust Anchor In Chapter 4, we have shown that resolvers do not always upgrade to a new trust anchor reliably. This aspect might also hinder the deployment of a new algorithm at the root. We proposed to distribute trust anchors through the operating system, instead of relying on software upgrading their trust anchor individually. Before rolling to a new algorithm, we need to understand how reliably trust anchors are distributed today, almost three years after the first Root KSK rollover. We can study this by analyzing software, operating systems and their deployment. Also, there are no concrete plans for an algorithm rollover yet [255]. This might give us another opportunity to study trust anchor deployment during another root KSK rollover, similar to Chapter 4.

8.3.2 Deploying Quantum-Safe Algorithms in DNSSEC

In Chapter 7, we took the first steps to apply quantum-safe algorithms in DNSSEC. We, however, need to take several more steps before quantum-safe algorithms can be deployed in DNSSEC on wider scale. To achieve this, especially the following two areas need more research.

Understanding the practical implications of alternate key distribution For some quantum-safe algorithms it will be necessary to transfer the public key of a domain name differently than we are used to today. In Chapter 7 we proposed to either split public keys, too large to fit in one resource record, in multiple DNS messages or to distribute keys out-of-band. Before standardizing one of these approaches we first need to understand how they would perform in realistic scenarios. This includes their impact on DNS performance and added overhead for resolvers and name servers. Also, we need to understand which barriers we might face when rolling out the approaches on a larger scale. Barriers could include, for example, networks not permitting out-of-band DNS traffic or the lack of support of unknown or modified DNS resource records.

Rethinking DNSSEC Signing Every finalist in round three of the NIST competition would not change the way DNSSEC signs RRs today. If by the end of the competition none of these algorithms are suitable for DNSSEC, then we might need to consider algorithms that require more extensive changes to the DNS and DNSSEC.

One of those are hash-based algorithms. Hash-based signing algorithms have large signatures and a key can only sign a limited number of RRs before it becomes insecure. At first sight, these properties seem to make hash-based signing algorithms unsuitable for DNSSEC, but with some modifications this might still be feasible. This would change the way we sign zones today and also resolvers would need to follow a different approach to validate records. The feasibility of such a signing schema needs to be analyzed further, which requires a better understanding of the dynamic of large zones (e.g. at TLDs), resolver behavior (e.g. their query distribution across names in a zone), and the operational challenges (e.g. how to keep track of state).

Way Forward Even though the NIST competition and the development of new quantum-safe algorithms is still ongoing, we can already take concrete steps towards making DNSSEC quantum-safe. Here, we briefly sketch out a possible way forward.

First, we can *model* the impact of quantum-safe algorithms on real world traffic. Van Rijswijk et al. [158] assessed the impact of Elliptic-Curve-Cryptography (ECC) on validation performance and we can apply a similar approach with different quantum-safe algorithms as well. As described in Chapter 7, we expect that we cannot transport the signatures of some algorithms reliably and already know that the key size of other algorithms will be even impossible to transport in a DNS message. In this thesis we have proposed solutions, like transmitting keys out-of-band. In order to understand the operational impact and discover not foreseen bottlenecks we propose to implement the proposed solution as prototypes and deploy them in test-beds. At the time of writing this thesis a Master student working on this topic under the supervision of the author of this thesis.

In case the studies above have a positive outcome, we can take the next step forward, assessing quantum-safe algorithms *in the wild*. Here, we can take earlier experiments with TLS as an inspiration. The cloud provider Cloudflare and Google tested the impact of different quantum-safe Key Encapsulation Mechanisms (KEM) on the HTTPS traffic between web-servers of Cloudflare and the Chrome browser [256]. To measure the impact on latency they implemented support for different quantum-safe algorithms in the TLS handshake. We propose to set up a similar experiment in the DNS as well. Here, we will need to cooperate with large resolver operators, with presences worldwide. Then, we would need to sign a domain name with quantum-safe algorithms that seem suitable for DNSSEC from earlier experiments. If necessary, we implement the alternative key distribution mechanisms as well on the side of the authoritative name server and the recursive resolver. With this experiment we can measure the impact of quantum-safe algorithms on transport, name servers and resolvers.

In the meantime, we expect that NIST has standardized one or more quantum-safe signing algorithms. In case our experiments have shown that one of these al-

gorithms is suitable, but that deploying them in DNSSEC would require protocol modifications, we would now propose bringing these protocol modifications forward as an experimental draft in the IETF. This would also give more DNS software developers the opportunity to provide their feedback and experiment with the implementation. At the same time, other algorithms, not part of the NIST competition might mature. We will keep a close eye on them to identify possible interesting candidates for DNSSEC.

We believe that these described steps will help us to prepare DNSSEC for quantum-safe algorithms on time for the arrival of production scale quantum computers.

APPENDIX A

Open Data Management

In this thesis, we relied on different existing data sets and carried out measurements ourselves. If possible and relevant, we made the data sets and measurement results available to the public. Our goal is to allow others to (i) reproduce our results of our studies, thereby increasing transparency, and also (ii) enable others to build on our results.

Additionally, we make the source code with which we carried out our measurements available for two studies. This, again, has the goal to increase transparency and reproducibility, but also to enable a broader community to carry out the measurements themselves.

Chapter	Description	Type	References
Chapter 3	Algorithm deployment at domain names	OpenIntel ♥	—
	Algorithm deployment at resolvers	RIPE Atlas ♣	[101]
Chapter 4	Root query data	DITL ♦ & Operators ♥	[126]
	Root response size	RSSAC ♣	
	RFC 8145 trust anchor signaling	Operators & ICANN ♥	
	Resolver state monitoring	RIPE Atlas ♣	
	RFC 8509 sentinel measurements	RIPE Atlas ding168 & Luminati ♥	
Chapter 6	Active server selection	RIPE Atlas ♣	[194]
	Passive server selection	DITL ♦ & Operators ♥	—

Table A.1: Data sets, created and studied in this thesis. Data sets open (♣), available for members only (♦), or private (♥).

Data Sets Table A.1 lists the data datasets we used or produced in this thesis. We carried out measurements with the help of RIPE Atlas (see Explainer 2 on page 39 for more details on RIPE Atlas) and Luminati [133]. Additionally, we mainly relied on two data sets, provided to us: First, OpenINTEL [62], which captures daily snapshots of large parts of the DNS. Second, the Day In The Life of the Internet (DITL) collection effort, consisting out of traffic captures at the DNS root servers. In some chapters, we also studied additional data sets, specific to one research, listed in Table A.1.

We cannot publish all data sets, due to privacy concerns (e.g. because they con-

tain personal information) or legal restrictions. We, however, made aggregated data sets publicly available in some cases. We refer the reader to the individual chapters for more details on each data set.

Source Code For two of our studies, we also made source code available, listed in Table A.2. In case of Chapter 5, we publish the source code that allows operators to monitor their algorithm rollovers. The Git repository contains instructions to install and run the tool as well. For Chapter 7, we publish the software with which we have measured the performance of quantum-safe algorithms.

Chapter	Description	References
Chapter 5	Monitoring tool for measuring algorithm rollovers	`https://github.com/SIDN/rollover-mon`
Chapter 7	Quantum-safe algorithm performance measurements	[216]

Table A.2: Public source code

APPENDIX B

Ethical Considerations

The measurements we carried out and the data sets we used in this thesis can, potentially, raise ethical concerns. For example, active measurements impose additional load on the measured services thereby threatening their availability and passive measurements can reveal sensitive personal information, like a queried domain name. Following the guidelines from [257] we explain in this section: (a) how we designed our measurements such that they did not cause harm to any person's well being, and (b) how data provided by third parties is collected. Furthermore, (c) we describe how we handled the disclosure of issues at third parties, revealed by our measurements. We discuss ethical issues, specific to individual studies in the respective chapters.

ACTIVE MEASUREMENTS

To carry out active measurements, and as shown in Appendix A, we mostly relied on the RIPE Atlas measurement platform. RIPE Atlas probes are run by volunteers, hosted in their own networks. As discussed in [70], we therefore have to pay attention (i) what we measure, and (ii) how often we measure.

Sending queries to domain names that provide content considered illegal in some countries, might get the host of the probe into trouble with the local authority. In our research, we do not send queries to sensitive domain names and thus, they do not put host of probes at risk. This also applies to measurements carried out with the Luminati proxy network [133].

Also, we need to consider the measurement frequency. A high frequency might require too much bandwidth of the probe's Internet connection, especially in more remote areas. Moreover, the measured services (e.g. an authoritative name server) might receive significant number of requests when our measurement relies on thousands of VPs. In some of our studies, we require a high measurement frequency (e.g. every minute in case of the measurement of the publication delay, see Chapter 5), but only for a few minutes. There, we also measure services (authoritative name servers of a TLD and the root) that are well connected, have sufficient capacity, and where we have the operator's consent (in case of the measured TLD). We recommend operators that would like to apply our monitor method as well to only select VPs that reflect their user base. This reduces load on services further.

In other studies, we run measurements infrequent (e.g. every hour), to well

equipped services (e.g. the authoritative name servers of .nl and the root) or to services under our control. For long running measurements, or measurements that involve all available RIPE Atlas probes we asked the RIPE NCC for consent or even cooperated with them directly.

THIRD PARTY DATA SETS

In our studies, we relied on a number of data sets, provided to us by third parties. Especially the data sets that contain passive packet traces from authoritative name servers can contain sensitive information, like IP addresses or domain names. Despite the fact that authoritative name servers *usually* do not see traffic originating from end users [32], we still need to handle them with care. Only in very specific cases, we analyzed queried domain names and IP addresses. These were used to identify misconfiguration and to reach out to operators in order to inform them about our observations and to get more details as to why we see certain behavior. We do not publish individual domain names nor IP addresses in any of our data sets (see Appendix A).

The data sets, used in this thesis were collected with potential ethical issues in mind. OpenINTEL follows strict ethical guidelines [62] and also data from the Dutch ccTLD .nl is collected under a restrictive privacy policy [200]. Traces from the root servers [127] can only be accessed by OARC members (the organization managing the data collection).

Glossary

ARPANET Advanced Research Projects Agency Network.

AS Autonomous System.

ASN Autonomous System Number.

BSI Federal Office for Information Security.

ccTLD Country Code Top Level Domain.

CDN Content Delivery Network.

DANE DNS-based Authentication of Named Entities.

DDoS Distributed Denial of Service.

DNS Domain Name System.

DNS-OARC Domain Name System Operations Analysis and Research Center.

DNSSEC DNS Security Extensions.

DoH DNS-over-HTTPS.

DoT DNS-over-TLS.

DS Delegation Signer.

EDNS0 Extension Mechanisms for DNS.

gTLD Generic Top Level Domain.

HTTP Hypertext Transfer Protocol.

IANA Internet Assigned Numbers Authority.

ICANN Internet Corporation for Assigned Names and Numbers.

IETF Internet Engineering Task Force.

IP Internet Protocol.

ISP Internet Service Provider.

KSK Key Signing Key.

MITM Man-in-the-middle.

NIST National Institute of Standards and Technology.

OS Operating System.

RIPE NCC Réseaux IP Européens Network Coordination Centre.

RR Resource Record.

RSA Rivest–Shamir–Adleman.

RTT Round-Trip Time.

SIDN Stichting Internet Domeinregistratie Nederland.

TCP Transport Control Protocol.

TLD Top Level Domain.

TLS Transport Layer Security.

TTL Time To Live.

UDP User Datagram Protocol.

VP Vantage Point.

ZSK Zone Signing Key.

Bibliography

[1] Statistisches Bundesamt, *Coronavirus crisis: experimental data reflect current buying behaviour*, https://www.destatis.de/EN/Press/2020/04/PE20_130_61.html, 2020.

[2] S. Milasi, I. González-Vázquez and E. Fernández-Macías, *Telework in the EU before and after the COVID-19: where we were, where we head to*, Science for Policy Brief, 2020.

[3] S. Perez, *Houseparty reports 50M sign-ups in past month amid COVID-19 lockdowns*, https://techcrunch.com/2020/04/15/houseparty-reports-50m-sign-ups-in-past-month-amid-covid-19-lockdowns, 2020.

[4] Human Rights Watch, *End Internet Shutdowns to Manage COVID-19*, https://www.hrw.org/news/2020/03/31/end-internet-shutdowns-manage-covid-19, 2020.

[5] R. Rasmussen, *SAC105: DNS and the Internet of Things: Opportunities, Risks, and Challenges*, https://www.icann.org/en/system/files/files/sac-105-en.pdf, 2019.

[6] E. Lastdrager, C. Hesselman, J. Jansen and M. Davids, "Protecting home networks from insecure iot devices", in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–6.

[7] D. Kaminsky, "Black Ops 2008: It's The End Of The Cache As We Know It", *Black Hat USA*, vol. 2, 2008.

[8] A. Herzberg and H. Shulman, "Fragmentation Considered Poisonous, or: One-domain-to-rule-them-all.org", in *2013 IEEE Conference on Communications and Network Security, CNS 2013*, 2013, pp. 224–232. arXiv: arXiv:1205.4011v1.

[9] K. Man, Z. Qian, Z. Wang, X. Zheng, Y. Huang and H. Duan, "DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels", in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1337–1350.

[10] R. Arends, R. Austein, M. Larson, D. Massey and S. Rose, *DNS Security Introduction and Requirements*, RFC 4033 (Proposed Standard), RFC, Updated by RFCs 6014, 6840, Fremont, CA, USA: RFC Editor, Mar. 2005. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4033.txt.

[11] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, "Quantum supremacy using a programmable superconducting processor", *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[12] H.-S. Zhong, H. Wang, Y.-H. Deng, M.-C. Chen, L.-C. Peng, Y.-H. Luo, J. Qin, D. Wu, X. Ding, Y. Hu *et al.*, "Quantum computational advantage using photons", *Science*, vol. 370, no. 6523, pp. 1460–1463, 2020.

[13] F. Muller and M. Heesch, *Migration to Quantum-Safe Cryptography: Position Paper*, Whitepaper: `https://repository.tno.nl/islandora/object/uuid:49ee6aa3-3eca-4c84-8908-02abd49674c1`, 2020.

[14] M. Müller, W. Toorop, T. Chung, J. Jansen and R. van Rijswijk-Deij, "The Reality of Algorithm Agility: Studying the DNSSEC Algorithm Life-Cycle", in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20, Virtual Event, USA: Association for Computing Machinery, 2020, pp. 295–308. [Online]. Available: `https://doi.org/10.1145/3419394.3423638`.

[15] M. Müller, M. Thomas, D. Wessels, W. Hardaker, T. Chung, W. Toorop and R. v. Rijswijk-Deij, "Roll, Roll, Roll your Root: A Comprehensive Analysis of the First Ever DNSSEC Root KSK Rollover", in *Proceedings of the Internet Measurement Conference*, 2019, pp. 1–14.

[16] O. Kolkman, W. Mekking and R. Gieben, *DNSSEC Operational Practices, Version 2*, RFC 6781 (Informational), RFC, Fremont, CA, USA: RFC Editor, Dec. 2012. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc6781.txt`.

[17] S. Morris, J. Ihren, J. Dickinson and W. Mekking, *DNSSEC Key Rollover Timing Considerations*, RFC 7583 (Informational), RFC, Fremont, CA, USA: RFC Editor, Oct. 2015. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc7583.txt`.

[18] Neves, Frederico A. C., *.br DNSSEC Algorithm Rollover*, ICANN 61 DNSSEC Workshop, San Juan `https://static.ptbl.co/static/attachments/169303/1520891993.pdf?1520891993`, 2018.

[19] M. Müller, T. Chung, A. Mislove and R. van Rijswijk-Deij, "Rolling With Confidence: Managing the Complexity of DNSSEC Operations", *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 1199–1211, 2019.

[20] M. Müller, G. C. M. Moura, R. de O. Schmidt and J. Heidemann, "Recursives in the Wild: Engineering Authoritative DNS Servers", in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC '17, London, United Kingdom: Association for Computing Machinery, 2017, pp. 489–495. [Online]. Available: `https://doi.org/10.1145/3131365.3131366`.

[21] M. Müller, J. de Jong, M. van Heesch, B. Overeinder and R. van Rijswijk-Deij, "Retrofitting Post-Quantum Cryptography in Internet Protocols: A Case Study of DNSSEC", *ACM SIGCOMM Computer Communication Review*, vol. 50, no. 4, 2020.

[22] Yahoo Finance, *Internet Grows to 370.1 Million Domain Name Registrations at the End of the Second Quarter of 2020*, `https://finance.yahoo.com/news/internet-grows-370-1-million-201500616.html`, 2020.

[23] P. Mockapetris and K. J. Dunlap, "Development of the Domain Name System", in *Symposium proceedings on Communications architectures and protocols*, 1988, pp. 123–133.

[24] P. Mockapetris, *Domain names: Implementation specification*, RFC 883, RFC, Obsoleted by RFCs 1034, 1035, updated by RFC 973, Fremont, CA, USA: RFC Editor, Nov. 1983. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc883.txt`.

[25] Internet Assigned Numbers Authority (IANA), *Root Servers*, `https://www.iana.org/domains/root/servers`.

[26] K. Schomp, T. Callahan, M. Rabinovich and M. Allman, "On Measuring the Client-side DNS Infrastructure", in *Proceedings of the 2013 Conference on Internet Measurement Conference*, 2013, pp. 77–90.

[27] APNIC Statistics Website, *Use of DNS Resolvers for World (XA)*, `https://stats.labs.apnic.net/rvrs`, 2020.

[28] A. Conta, P. Doolan and A. Malis, *Use of Label Switching on Frame Relay Networks Specification*, RFC 3034 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Jan. 2001. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc3034.txt`.

[29] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels and P. Hoffman, *Specification for DNS over Transport Layer Security (TLS)*, RFC 7858 (Proposed Standard), RFC, Updated by RFC 8310, Fremont, CA, USA: RFC Editor, May 2016. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc7858.txt`.

[30] P. Hoffman and P. McManus, *DNS Queries over HTTPS (DoH)*, RFC 8484 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Oct. 2018. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc8484.txt`.

[31] Chromium Blog, *A safer and more private browsing experience with Secure DNS*, 2020. [Online]. Available: `https://blog.chromium.org/2020/05/a-safer-and-more-private-browsing-DoH.html`.

[32] S. Bortzmeyer, *DNS Privacy Considerations*, RFC 7626 (Informational), RFC, Fremont, CA, USA: RFC Editor, Aug. 2015. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc7626.txt`.

[33] ——, *DNS Query Name Minimisation to Improve Privacy*, RFC 7816 (Experimental), RFC, Fremont, CA, USA: RFC Editor, Mar. 2016. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc7816.txt`.

[34] E. Kinnear, P. McManus, T. Pauly, T. Verma and C. A. Wood, "Oblivious DNS Over HTTPS", Internet Engineering Task Force, Internet-Draft draft-pauly-dprive-oblivious-doh-06, Mar. 2021, Work in Progress, 20 pp. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-pauly-dprive-oblivious-doh-06.

[35] IBM, *Docs and Resources - IBM Quantum Experience: Shor's Algorithm*, https://quantum-computing.ibm.com/docs/iqx/guide/shors-algorithm, 2021.

[36] C. Pomerance, "A Tale of Two Sieves", in *Notices American Mathematics Society*, Citeseer, 1996.

[37] P. W. Shor, "Polynomial Time Algorithms for Discrete Logarithms and Factoring on a Quantum Computer", in *International Algorithmic Number Theory Symposium*, Springer, 1994, pp. 289–289.

[38] M. Mosca, "Setting the Scene for the ETSI Quantum-safe Cryptography Workshop", in *e-proceedings of 1st Quantum-Safe-Crypto Workshop, Sophia Antipolis*, 2020.

[39] S. Beauregard, "Circuit for shor's algorithm using 2n+ 3 qubits", *arXiv preprint quant-ph/0205095*, 2002.

[40] C. Gidney and M. Ekerå, *How to Factor 2048 bit RSA Integers in 8 Hours Using 20 Million Noisy Qubits*, 2019. arXiv: 1905.09749.

[41] F. K. Wilhelm, R. Steinwandt, B. Langenberg, A. Liebermann Per J.and Messinger, P. K. Schuhmacher and A. Misra-Spieldenner, *Status of quantum computer development*, 2020.

[42] S. Shankland, *Quantum computer makers like their odds for big progress*, https://www.cnet.com/news/quantum-computer-makers-like-their-odds-for-big-progress-soon/, 2020.

[43] T. Simonite, *Microsoft's Big Win in Quantum Computing Was an Error After All*, https://www.wired.com/story/microsoft-win-quantum-computing-error/, 2021.

[44] P. Kotzias, A. Razaghpanah, J. Amann, K. G. Paterson, N. Vallina-Rodriguez and J. Caballero, "Coming of Age: A Longitudinal Study of TLS Deployment", in *Proceedings of the Internet Measurement Conference (IMC), 2018*, 2018, pp. 415–428.

[45] R. Housley, *Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms*, RFC 7696 (Best Current Practice), RFC, Fremont, CA, USA: RFC Editor, Nov. 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7696.txt.

[46] M. Müller, W. Toorop, T. Chung, J. Jansen and R. van Rijswijk-Deij, "The Reality of Algorithm Agility: Studying the DNSSEC Algorithm Life-Cycle", 2020.

[47] P. Wouters and O. Surý, *Algorithm Implementation Requirements and Usage Guidance for DNSSEC*, RFC 8624, Jun. 2019. [Online]. Available: https://rfc-editor.org/rfc/rfc8624.txt.

[48] D. Eastlake 3rd, *RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS)*, RFC 3110 (Proposed Standard), RFC, Updated by RFC 6944, Fremont, CA, USA: RFC Editor, May 2001. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3110.txt.

[49] S. Weiler, *Legacy Resolver Compatibility for Delegation Signer (DS)*, RFC 3755 (Proposed Standard), RFC, Obsoleted by RFCs 4033, 4034, 4035, updated by RFCs 3757, 3845, Fremont, CA, USA: RFC Editor, May 2004. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3755.txt.

[50] B. Laurie, G. Sisson, R. Arends and D. Blacka, *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*, RFC 5155 (Proposed Standard), RFC, Updated by RFCs 6840, 6944, Fremont, CA, USA: RFC Editor, Mar. 2008. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5155.txt.

[51] J. Jansen, *Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC*, RFC 5702 (Proposed Standard), RFC, Updated by RFC 6944, Fremont, CA, USA: RFC Editor, Oct. 2009. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5702.txt.

[52] V. Dolmatov (Ed.), A. Chuprina and I. Ustinov, *Use of GOST Signature Algorithms in DNSKEY and RRSIG Resource Records for DNSSEC*, RFC 5933 (Proposed Standard), RFC, Updated by RFC 6944, Fremont, CA, USA: RFC Editor, Jul. 2010. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5933.txt.

[53] P. Hoffman and W. Wijngaards, *Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC*, RFC 6605 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Apr. 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6605.txt.

[54] O. Sury and R. Edmonds, *Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC*, RFC 8080 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Feb. 2017. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8080.txt.

[55] O. Gudmundsson, *Delegation Signer (DS) Resource Record (RR)*, RFC 3658 (Proposed Standard), RFC, Obsoleted by RFCs 4033, 4034, 4035, updated by RFC 3755, Fremont, CA, USA: RFC Editor, Dec. 2003. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3658.txt.

[56] W. Hardaker, *Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records (RRs)*, RFC 4509 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, May 2006. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4509.txt.

[57] R. Arends, R. Austein, M. Larson, D. Massey and S. Rose, *Protocol Modifications for the DNS Security Extensions*, RFC 4035 (Proposed Standard), RFC, Updated by RFCs 4470, 6014, 6840, 8198, Fremont, CA, USA: RFC Editor, Mar. 2005. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4035.txt.

[58] D. York, O. Surý, P. Wouters and Ó. Guðmundsson, "Observations on Deploying New DNSSEC Cryptographic Algorithms", Internet Engineering Task Force, Internet-Draft draft-york-dnsop-deploying-dnssec-crypto-algs-06, Oct. 2018, Work in Progress, 11 pp. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-york-dnsop-deploying-dnssec-crypto-algs-06.

[59] T. Chung, R. van Rijswijk-Deij, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove and C. Wilson, "Understanding the Role of Registrars in DNSSEC Deployment", in *Proceedings of the Internet Measurement Conference (IMC), 2017*, ser. IMC '17, London, United Kingdom: Association for Computing Machinery, 2017, pp. 369–383. [Online]. Available: https://doi.org/10.1145/3131365.3131373.

[60] T. Le, R. van Rijswijk-Deij, L. Allodi and N. Zannone, "Economic Incentives on DNSSEC Deployment: Time to Move from Quantity to Quality", in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2018, pp. 1–9.

[61] R. van Rijswijk-Deij, M. Jonker and A. Sperotto, "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, 2016, pp. 258–262.

[62] R. van Rijswijk-Deij, M. Jonker, A. Sperotto and A. Pras, "A High-Performance, Scalable Infrastructure for Large-Scale Active DNS Measurements", *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 6, pp. 1877–1888, 2016.

[63] Internet Engineering Task Force (IETF), *Mail Archive*, https://mailarchive.ietf.org/arch/, 2020.

[64] S. Rose, *Applicability Statement: DNS Security (DNSSEC) DNSKEY Algorithm Implementation Status*, RFC 6944 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Apr. 2013. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6944.txt.

[65] G. Houston, *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, 2015.

[66] M. Müller, W. Toorop, T. Chung and R. van Rijswijk-Deij, *CENTR Survey: Supported DNSSEC Signing Algorithms*, https://docs.google.com/forms/d/e/1FAIpQLSfpLT_aPWDnBLt2kmkcihgXJPZquy3tQSBvV77-2Ct9izsBwg/viewform?usp=sf_link, 2020.

[67] T. Finch, *Archive of the DNS root zone*, https://github.com/fanf2/saveroot, 2020.

[68] A. Davies and P. Homburg, *RIPE Atlas Software Probes*, PQC-Forum: https://labs.ripe.net/Members/alun_davies/ripe-atlas-software-probes, 2020.

[69] RIPE NCC Staff, "RIPE Atlas: A Global Internet Measurement Network", *Internet Protocol Journal (IPJ)*, vol. 18, no. 3, 2015.

[70] R. Kisteleki, *Ethics of RIPE Atlas Measurements*, https://labs.ripe.net/Members/kistel/ethics-of-ripe-atlas-measurements, 2016.

[71] S. Bradner, *The Internet Standards Process – Revision 3*, RFC 2026 (Best Current Practice), RFC, Updated by RFCs 3667, 3668, 3932, 3978, 3979, 5378, 5657, 5742, 6410, 7100, 7127, 7475, 8179, Fremont, CA, USA: RFC Editor, Oct. 1996. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2026.txt.

[72] I. E. T. F. (IETF), *IESG states*, https://datatracker.ietf.org/help/state/draft/iesg.

[73] F. Mendel, N. Pramstaller, C. Rechberger, M. Kontak and J. Szmidt, "Cryptanalysis of the gost hash function", in *Advances in Cryptology – CRYPTO 2008*, D. Wagner, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 162–178.

[74] F. Mendel, N. Pramstaller and C. Rechberger, "A (Second) Preimage Attack on the GOST Hash Function", in *Fast Software Encryption*, K. Nyberg, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 224–234.

[75] Various Authors, *IETF DNSOP Mailing List Archive, June 2020*, https://mailarchive.ietf.org/arch/msg/dnsop/EYv_-LfkyiQmdguYpbJ3LLX-4Ls/.

[76] I. E. T. Force, *History of draft-hoffman-dnssec-dsa-sha2*, https://datatracker.ietf.org/doc/draft-hoffman-dnssec-dsa-sha2/history/.

[77] Various Authors, *IETF DNSEXT Mailing List Archive, June 2006*, https://mailarchive.ietf.org/arch/browse/namedroppers/?gbt=1&index=7M2k1FW261sYIoDIqinA8WOrOZs.

[78] I. A. N. A. (IANA), *Domain Name System Security (DNSSEC) Algorithm Numbers*, https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml.

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

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

[81] Internet Systems Consortium (ISC), *Release Notes for BIND Version 9.14.0*, https://downloads.isc.org/isc/bind9/9.14.0/RELEASE-NOTES-bind-9.14.0.txt, 2019.

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

[83] M. Hamubrg, *libdecaf Changelog*, https://github.com/krionbsd/libdecaf/blob/master/ChangeLog, 2020.

[84] F. Dennis, *libsodium Releases*, https://github.com/jedisct1/libsodium/releases?after=0.4, 2020.

[85] Brian Murray, *Ubuntu Wiki: Stable Release Updates*, https://wiki.ubuntu.com/StableReleaseUpdates, 2020.

[86] Holland, Nick and Mestdagh, Steven and Knight, Joel and Jackson, Eric and Vandeputtem Wim and Cappuccio, Chris, *OpenBSD FAQ - Package Management*, https://www.openbsd.org/faq/faq15.html, 2020.

[87] W. Kumari, O. Gudmundsson and G. Barwood, *Automating DNSSEC Delegation Trust Maintenance*, RFC 7344 (Proposed Standard), RFC, Updated by RFC 8078, Fremont, CA, USA: RFC Editor, Sep. 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7344.txt.

[88] O. Gudmundsson and P. Wouters, *Managing DS Records from the Parent via CDS/CDNSKEY*, RFC 8078 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Mar. 2017. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8078.txt.

[89] S. Weiler (Ed.) and D. Blacka (Ed.), *Clarifications and Implementation Notes for DNS Security (DNSSEC)*, RFC 6840 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Feb. 2013. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6840.txt.

[90] Council of European National Top-Level Domain Registries (CENTR), *CENTR Homepage*, https://centr.org, 2020.

[91] DNS OARC, *DNSViz | A DNS visualization tool*, https://dnsviz.net/, 2020.

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

[93] Not Disclosed, *On the zone policy of a European ccTLD.* Private correspondence, 2020.

[94] R. van Rijswijk-Deij, A. Sperotto and A. Pras, "Making the Case for Elliptic Curves in DNSSEC", *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 13–19, 2015.

[95] D. Grant, *Announcing Universal DNSSEC: Secure DNS for Every Domain*, https://blog.cloudflare.com/introducing-universal-dnssec/, 2015.

[96] G. Houston, *ECDSA and DNSSEC*, https://blog.apnic.net/2014/10/23/ecdsa-and-dnssec/, Oct. 2014.

[97] ICANN, *Delegated Strings*, https://newgtlds.icann.org/en/program-status/delegated-strings, 2020.

[98] Internet Corporation for Assigned Names and Numbers (ICANN), "Transition to Delegation", in *gTLD Applicant Guidebook*, 2012.

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

[100] R. van Rijswijk-Deij, T. Chung, D. Choffnes, A. Mislove and W. Toorop, "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*, Los Angeles, CA, USA: ACM Press, 2017.

[101] DNSThought website, *Atlas measurements used with DNSThought*, https://dnsthought.nlnetlabs.nl/raw/, 2018.

[102] R. Arends, R. Austein, M. Larson, D. Massey and S. Rose, *Resource Records for the DNS Security Extensions*, RFC 4034 (Proposed Standard), RFC, Updated by RFCs 4470, 6014, 6840, 6944, Fremont, CA, USA: RFC Editor, Mar. 2005. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4034.txt.

[103] N. I. of Standards and T. (NIST), *Secure Hashing*, https://web.archive.org/web/20110625054822/http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html.

[104] P. Karpman, T. Peyrin and M. Stevens, "Practical Free-Start Collision Attacks on 76-step SHA-1", in *Annual Cryptology Conference*, Springer, 2015, pp. 623–642.

[105] M. Stevens, E. Bursztein, P. Karpman, A. Albertini and Y. Markov, "The first collision for full sha-1", in *Annual International Cryptology Conference*, Springer, 2017, pp. 570–596.

[106] G. Leurent and T. Peyrin, "From Collisions to Chosen-Prefix Collisions Application to Full SHA-1", in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2019, pp. 527–555.

[107] Hoffman, Paul, *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, 2020.

[108] Internet Corporation for Assigned Names and Numbers (ICANN), *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, 2013.

[109] M. Müller, M. Thomas, D. Wessels, W. Hardaker, T. Chung, W. Toorop and R. v. Rijswijk-Deij, "Roll, Roll, Roll your Root: A Comprehensive Analysis of the First Ever DNSSEC Root KSK Rollover", 2019.

[110] ——, "Roll, Roll, Roll your Root: A Comprehensive Analysis of the First Ever DNSSEC Root KSK Rollover", 2019.

[111] IANA, *DNSSEC Practice Statement for the Root Zone KSK Operator*, https://www.iana.org/dnssec/dps/ksk-operator/ksk-dps.txt, 2016.

[112] KSK Rollover Design Team, *Root Zone KSK Rollover Plan*, https://www.iana.org/reports/2016/root-ksk-rollover-design-20160307.pdf, Apr. 2016.

[113] D. Wessels, W. Kumari and P. Hoffman, *Signaling Trust Anchor Knowledge in DNS Security Extensions (DNSSEC)*, RFC 8145 (Proposed Standard), RFC, Updated by RFC 8553, Fremont, CA, USA: RFC Editor, Apr. 2017. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8145.txt.

[114] ICANN, *KSK Rollover Postponed*, https://www.icann.org/news/announcement-2017-09-27-en, 2017.

[115] ICANN Board, *Board Approval of KSK Roll*, https://www.icann.org/resources/press-material/release-2018-09-18-en, 2018.

[116] ICANN, *Review of the 2018 DNSSEC KSK Rollover*, https://www.icann.org/en/system/files/files/review-2018-dnssec-ksk-rollover-04mar19-en.pdf, Mar. 2019.

[117] R. Chandramouli and S. Rose, "Secure Domain Name System (DNS) Deployment Guide", *NIST Special Publication*, vol. 800, Sep. 2006.

[118] V. D. PMA, *DNSSEC Practice Statement for the Root Zone ZSK Operator*, https://www.iana.org/dnssec/dps/zsk-operator/dps-zsk-operator-v2.0.pdf, 2017.

[119] NTIA, *NTIA Announces Intent to Transition Key Internet Domain Name Functions*, https://www.ntia.doc.gov/press-release/2014/ntia-announces-intent-transition-key-internet-domain-name-functions, 2014.

[120] ICANN, *Operational Plans for the Root KSK Rollover*, https://www.icann.org/resources/pages/ksk-rollover-operational-plans, 2016.

[121] G. Van Den Broek, R. van Rijswijk-Deij, A. Sperotto and A. Pras, "DNSSEC Meets Real World: Dealing with Unreachability Caused by Fragmentation", *IEEE Communications Magazine*, vol. 52, no. 4, pp. 154–160, Jun. 2014.

[122] C. Kreibich, N. Weaver, B. Nechaev and V. Paxson, "Netalyzr: Illuminating the Edge Network", in *Proceedings of the Internet Measurement Conference (IMC), 2010*, ACM, 2010, pp. 246–259.

[123] M. StJohns, *Automated Updates of DNS Security (DNSSEC) Trust Anchors*, RFC 5011 (Internet Standard), RFC, Fremont, CA, USA: RFC Editor, Sep. 2007. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5011.txt.

[124] J. Abley, J. Schlyter, G. Bailey and P. Hoffman, *DNSSEC Trust Anchor Publication for the Root Zone*, RFC 7958 (Informational), RFC, Fremont, CA, USA: RFC Editor, Aug. 2016. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7958.txt.

[125] NLnet Labs, *Man-Page: Unbound Anchor*, https://www.nlnetlabs.nl/documentation/unbound/unbound-anchor/.

[126] M. Müller, M. Thomas, D. Wessels, W. Hardaker, T. Chung, W. Toorop and R. van Rijswijk-Deij, *Roll Roll Roll Your Root: Accompanying Data Sets*, https://github.com/SIDN/RollRollRollYourRoot.

[127] DNS Operations and Analysis Center (DNS-OARC), *Day-in-the-Life Datasets*, https://www.dns-oarc.net/oarc/data/ditl.

[128] S. Castro, D. Wessels, M. Fomenkov and K. Claffy, "A Day at the Root of the Internet", *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 5, pp. 41–46, 2008.

[129] ICANN, *Root Server System Advisory Committee*, https://www.icann.org/groups/rssac.

[130] R. Caucus, *RSSAC002 version 3 – RSSAC Advisory on Measurements of the Root Server System*, ICANN, 2016.

[131] RSSAC, *RSSAC002 Datasets*, https://github.com/rssac-caucus/RSSAC002-data.

[132] G. Huston, J. Damas and W. Kumari, *A Root Key Trust Anchor Sentinel for DNSSEC*, RFC 8509 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Dec. 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8509.txt.

[133] Luminati IO, *Residential IP and Proxy Service for Businesses*, https://luminati.io/, May 2018.

[134] T. Chung, D. Choffnes and A. Mislove, "Tunneling for Transparency: A Large-Scale Analysis of End-to-End Violations in the Internet", in *Proceedings of the Internet Measurement Conference (IMC), 2016*, 2016.

[135] T. Chung, R. van Rijswijk-Deij, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove and C. Wilson, "A Longitudinal, End-to-End View of the DNSSEC Ecosystem", in *Proceedings of USENIX Security 2017*, 2017.

[136] Luminati, *Luminati End User License Agreement*, https://luminati.io/license.

[137] NLnet Labs, *Unbound DNS Resolver*, https://www.unbound.net/. [Online]. Available: https://www.unbound.net/.

[138] ICANN, *2018 KSK Rollover Operational Implementation Plan*, https://www.icann.org/en/system/files/files/2018-ksk-roll-operational-implementation-plan.pdf, Apr. 2018.

[139] ICANN, Office of the CTO, *Staff Report of Public Comment Proceeding*, https://www.icann.org/en/system/files/files/report-comments-ksk-rollover-restart-23apr18-en.pdf, Apr. 2018.

[140] NLnet Labs, *Man-Page: unbound.conf*, https://nlnetlabs.nl/documentation/unbound/unbound.conf/.

[141] Ó. Guðmundsson, *DNSKEY cache purge*, Comment at the mic during DNS-OARC 29 meeting in Amsterdam, https://www.youtube.com/watch?v=yT51FwPG0jE&t=6782, 2018.

[142] Not Disclosed, *European ISP flushing DNSKEY from cache before the rollover*, Private correspondence, 2018.

[143] W. B. De Vries, R. Van Rijswijk-Deij, P.-T. de Boer and A. Pras, "Passive Observations of a Large DNS Service: 2.5 Years in the Life of Google", in *2018 Network Traffic Measurement and Analysis Conference (TMA)*, IEEE, 2018, pp. 1–8.

[144] S. Murphy, *'Significant percentage' of Eir customers affected by broadband outage*, https://www.rte.ie/news/2018/1013/1002966-eir-outage/, Oct. 2018.

[145] G. Houston, *Roll Over and Die?*, http://www.potaroo.net/ispcol/2010-02/rollover.html, Feb. 2010.

[146] ——, *Measuring the Root Zone KSK Trust*, https://blog.apnic.net/2018/04/11/measuring-the-root-zone-ksk-trust/, Apr. 2018.

[147] W. Hardaker, *Configurations and Scripts to Test BIND Behavior in the Absence of a Valid Trust Anchor*, https://github.com/hardaker/isc-bind-dnskey-bug-test.

[148] G. Huston, *APNIC Blog: Analyzing the KSK Roll*, https://labs.apnic.net/?p=1181, Oct. 2018.

[149] P. B. Danzig, K. Obraczka and A. Kumar, "An Analysis of Wide-Area Name Server Traffic", in *Proceedings of ACM SIGCOMM 1992*, Baltimore, MD, USA: ACM Press, 1992, pp. 281–292.

[150] P. Mockapetris, *Domain names - concepts and facilities*, RFC 1034 (Internet Standard), RFC, Fremont, CA, USA: RFC Editor, Nov. 1987. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1034.txt.

[151] N. Brownlee, K. Claffy and E. Nemeth, "DNS Measurements at a Root Server", in *Proceedings of IEEE GLOBECOM 2001*, vol. 3, San Antonio, TX, USA: IEEE Computer Society, 2001, pp. 1672–1676.

[152] D. Wessels and M. Fomenkov, "Wow, That's a lot of packets", in *Proceedings of the Passive and Active Network Measurement Workshop (PAM 2003)*, San Diego, CA: PAM, 2003.

[153] M. Lentz, D. Levin, J. Castonguay, N. Spring and B. Bhattacharjee, "D-mystifying the D-root Address Change", in *Proceedings of ACM SIGCOMM 2013*, Barcelona, Spain: ACM Press, 2013, pp. 57–62.

[154] D. Wessels, J. Castonguay and P. Barber, "Thirteen Years of "Old J-Root"", in *DNS-OARC 24*, Montréal, Canada, 2015.

[155] B. Ager, H. Dreger and A. Feldmann, "Predicting the DNSSEC Overhead Using DNS Traces", in *Proceedings of the 40th annual IEEE Conference on Information Sciences and Systems, CISS 2006*, Princeton, NJ, USA: IEEE Comput. Soc, 2007, pp. 1484–1489.

[156] W. C. A. Wijngaards and B. J. Overeinder, "Securing DNS: Extending DNS Servers with a DNSSEC Validator", *IEEE Security and Privacy*, vol. 7, no. 5, pp. 36–43, 2009.

[157] D. Migault, C. Girard and M. Laurent, "A Performance View on DNSSEC Migration", in *Proceedings of the 6th International Conference on Network and Service Management (CNSM 2010)*, Niagara Falls, Canada: IFIP, 2010, pp. 469–474.

[158] R. van Rijswijk-Deij, K. Hageman, A. Sperotto and A. Pras, "The Performance Impact of Elliptic Curve Cryptography on DNSSEC Validation", *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 738–750, Apr. 2017.

[159] H. Yang, E. Osterweil, D. Massey, S. Lu and L. Zhang, "Deploying cryptography in Internet-scale systems: A case study on DNSSEC", *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 5, pp. 656–669, Apr. 2011.

[160] W. Kumari, E. Hunt, R. Arends, W. Hardaker and D. C. Lawrence, "Extended DNS Errors", Internet Engineering Task Force, Internet-Draft draft-ietf-dnsop-extended-error-05, Mar. 2019, Work in Progress, 15 pp. [Online]. Available: `https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-extended-error-05`.

[161] Various Authors, *KSK Rollover Mailing List Archive, March 2019*, `https://mm.icann.org/pipermail/kskA-rollover/2019-March/thread.html`.

[162] M. Müller, T. Chung, A. Mislove and R. van Rijswijk-Deij, *Rollover Mon*, `https://github.com/SIDN/rollover-mon`.

[163] ——, "Rolling With Confidence: Managing the Complexity of DNSSEC Operations", in *RIPE 76 Meeting*, 2018.

[164] M. Davids, *Unbound Mailing List: DNSSEC validation failure of .nl TLD*, `https://unbound.net/pipermail/unbound-users/2012-October/002676.html`, Oct. 2012.

[165] Andersson, Jonas, *Lessons learned from the .SE algorithm rollover*, `https://www.iis.se/se-tech/lessons-learned-from-the-se-algorithm-rollover/`, Apr. 2018.

[166] T. Le, R. van Rijswijk-Deij, L. Allodi and N. Zannone, "Economic Incentives on DNSSEC Deployment: Time to Move from Quantity to Quality", in *IEEE/IFIP Network Operations and Management Symposium (NOMS 2018)*, IEEE, Apr. 2018.

[167] Geoff Huston, *Measuring the DNS from the Users perspective*, https://ripe68.ripe.net/presentations/164-2014-05-14-huston-dns-measurements.pdf, May 2014.

[168] DK Hostmaster, *Rollover Mon supports .dk algorithm rollover*, Private correspondence, 2020.

[169] P. Mockapetris, *Domain names - implementation and specification*, RFC 1035 (Internet Standard), RFC, Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2673, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604, 7766, 8482, 8490, Fremont, CA, USA: RFC Editor, Nov. 1987. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1035.txt.

[170] Internetstiftelsen i Sverige, *IIS Zone Data*, https://zonedata.iis.se/, May 2018.

[171] S. Bortzmeyer, *Unbound Mailing List: BIND validates but not Unbound: who is right?*, https://unbound.net/pipermail/unbound-users/2015-February/003778.html, Feb. 2015.

[172] J. Abley and K. Lindqvist, *Operation of Anycast Services*, RFC 4786 (Best Current Practice), RFC, Fremont, CA, USA: RFC Editor, Dec. 2006. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4786.txt.

[173] V. Bajpai, S. J. Eravuchira, J. Schönwälder, R. Kisteleki and E. Aben, "Vantage point selection for IPv6 measurements: Benefits and limitations of RIPE Atlas tags", in *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium*, IEEE, May 2017.

[174] NLnet Labs, *A short history of DNSSEC*, https://www.nlnetlabs.nl/projects/dnssec/history.html, Sep. 2013.

[175] W. Kumari and P. Hoffman, *Decreasing Access Time to Root Servers by Running One on Loopback*, RFC 7706 (Informational), RFC, Fremont, CA, USA: RFC Editor, Nov. 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7706.txt.

[176] Kuroiwa, Cesar, *Algorithm Rollover on .br*, GTER 48/GTS 32 ftp://ftp.registro.br/pub/gts/gts32/01-br-algorithm-rollover.pdf, Dec. 2018.

[177] S. Ariyapperuma and C. J. Mitchell, "Security Vulnerabilities in DNS and DNSSEC", in *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, IEEE, Apr. 2007, pp. 335–342.

[178] O. Kolkman and R. Gieben, *DNSSEC Operational Practices*, RFC 4641 (Informational), RFC, Obsoleted by RFC 6781, Fremont, CA, USA: RFC Editor, Sep. 2006. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4641.txt.

[179] IIS and Afnic, *Zonemaster*, https://github.com/dotse/zonemaster, Jan. 2018.

[180] Internet Systems Consortium, Inc., *BIND DNSSEC Guide*, https://ftp.isc.org/isc/dnssec-guide/dnssec-guide.pdf, Jan. 2017.

[181] Knot-DNS, *Knot DNS Documentation: Operation - DNSSEC Key Rollovers*, https://www.knot-dns.cz/docs/2.7/html/operation.html#dnssec-key-rollovers, Feb. 2019.

[182] OpenDNSSEC Project, *OpenDNSSEC*, https://github.com/opendnssec/opendnssec, Feb. 2018.

[183] Secure64, *Secure64 DNS Signer*, secure64.com, Jan. 2016.

[184] Isasi, Sergi and Shrestha, Vicky, *Expanding DNSSEC Adoption*, https://blog.cloudflare.com/automatically-provision-and-maintain-dnssec/, Sep. 2018.

[185] Knot-DNS, *Automated DNSSEC Provisioning: Guidelines for CDS processing at SWITCH*, https://www.nic.ch/export/shared/.content/files/SWITCH_CDS_Manual_en.pdf, Feb. 2019.

[186] J. Talir, "Automating DNSSEC via CDNSKEY processing", in *RIPE 75*, Oct. 2017.

[187] Y. Yu, D. Wessels, M. Larson and L. Zhang, "Authority Server Selection in DNS Caching Resolvers", *SIGCOMM Computer Communication Review*, vol. 42, no. 2, 80–86, Mar. 2012.

[188] D. McPherson, D. Oran, D. Thaler and E. Osterweil, *Architectural Considerations of IP Anycast*, RFC 7094 (Informational), RFC, Fremont, CA, USA: RFC Editor, Jan. 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7094.txt.

[189] G. C. M. Moura, R. de O. Schmidt, J. Heidemann, W. B. de Vries, M. Müller, L. Wei and C. Hesselman, "Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event", in *Proceedings of the 2016 ACM Conference on Internet Measurement Conference*, Oct. 2016, pp. 255–270.

[190] W. Wijngaards, *Unbound Timeout Information*, https://unbound.net/documentation/info_timeout.html, Nov. 2010.

[191] C. Almond, *Address database dump (ADB) - understanding the fields and what they represent*, https://kb.isc.org/article/AA-01463/0/Address-database-dump-ADB-understanding-the-fields-and-what-they-represent.html, 2017.

[192] S. Woolf and D. Conrad, "Requirements for a Mechanism Identifying a Name Server Instance", johnh: pafiles, RFC 4892, Jun. 2007. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2414.txt.

[193] M. Müller, G. C. M. Moura, R. de O. Schmidt and J. Heidemann, *Recursives in the wild datasets*, https://www.simpleweb.org/wiki/index.php/Traces#Recursives_in_the_Wild:_Engineering_Authoritative_DNS_Servers and https://ant.isi.edu/datasets/all.html#DNS_Recursive_Study-20170323, May 2017.

[194] RIPE NCC, *RIPE Atlas Measurement IDs*, https://atlas.ripe.net/measurements/ID, ID is the experiment ID: 2A: 7951948, 2B: 7953390, 2C: 7967380, 3A: 7961003, 3B: 7954122, 4A: 7966930, 4B: 7960323, 2C-5min: 8321846, 2C-10min: 8323303, 2C-15min: 8324963, 2C-20min: 8329423, 2C-15min: 8335072, Mar. 2017. [Online]. Available: https://atlas.ripe.net/measurements/.

[195] T. Callahan, M. Allman and M. Rabinovich, "On modern DNS behavior and properties", *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 7–15, Jul. 2013.

[196] V. Bajpai, S. J. Eravuchira and J. Schönwälder, "Lessons Learned from using the RIPE Atlas Platform for Measurement Research", *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 3, pp. 35–42, Jul. 2015. [Online]. Available: http://www.sigcomm.org/sites/default/files/ccr/papers/2015/July/0000000-0000005.pdf.

[197] R. d. O. Schmidt, J. Heidemann and J. H. Kuipers, "Anycast Latency: How Many Sites Are Enough?", johnh: pafile, Mar. 2017, pp. 188–200. [Online]. Available: http://www.isi.edu/%7ejohnh/PAPERS/Schmidt17a.html.

[198] SIDN Labs, *.nl stats and data*, http://stats.sidnlabs.nl/#network, Mar. 2017. [Online]. Available: %5Curl%7Bhttp://stats.sidnlabs.nl/%5C#network%7D.

[199] DNS OARC, *DITL Traces and Analysis*, https://www.dns-oarc.net/oarc/data/ditl/2017, Feb. 2017.

[200] M. Wullink, G. C. Moura, M. Müller and C. Hesselman, "ENTRADA: A high-performance network traffic data streaming warehouse", in *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, IEEE, Apr. 2016, pp. 913–918.

[201] M. Kührer, T. Hupperich, J. Bushart, C. Rossow and T. Holz, "Going wild: Large-scale classification of open DNS resolvers", in *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, ACM, Oct. 2015, pp. 355–368.

[202] B. Ager, W. Mühlbauer, G. Smaragdakis and S. Uhlig, "Comparing DNS resolvers in the wild", in *Proceedings of the 10th ACM SIGCOMM conference on Internet Measurement*, ACM, Sep. 2010, pp. 15–21.

[203] M. Korczyński, M. Król and M. van Eeten, "Zone Poisoning: The How and Where of Non-Secure DNS Dynamic Updates", in *Proceedings of the 2016 ACM on Internet Measurement Conference*, ACM, 2016, pp. 271–278.

[204] P. Vixie (Ed.), S. Thomson, Y. Rekhter and J. Bound, *Dynamic Updates in the Domain Name System (DNS UPDATE)*, RFC 2136 (Proposed Standard), RFC, Updated by RFCs 3007, 4035, 4033, 4034, Fremont, CA, USA: RFC Editor, Apr. 1997. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2136.txt.

[205] K. Schomp, "DNS Recursive Resolver Delegation Selection in the Wild", in *DNS-OARC 30*, Shangri-La, Bangkok, 2019.

[206] M. Müller, J. de Jong, M. van Heesch, B. Overeinder and R. van Rijswijk-Deij, "Retrofitting Post-Quantum Cryptography in Internet Protocols: A Case Study of DNSSEC", in *DNS-OARC 34*, Online, 2021.

[207] ——, "Retrofitting Post-Quantum Cryptography in Internet Protocols: A Case Study of DNSSEC", in *ICANN 70 DNSSEC Workshop*, Online, 2021.

[208] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta *et al.*, *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process*. US Department of Commerce, National Institute of Standards and Technology, 2020.

[209] R. Rasmussen, *SSAC Comment to NIST on Quantum Cryptography Algorithms*, `https://www.icann.org/en/system/files/files/sac-107-en.pdf`, 2019.

[210] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson and H. Levkowetz (Ed.), *Extensible Authentication Protocol (EAP)*, RFC 3748 (Proposed Standard), RFC, Updated by RFCs 5247, 7057, Fremont, CA, USA: RFC Editor, Jun. 2004. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc3748.txt`.

[211] E. Crockett, C. Paquin and D. Stebila, "Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH", in *NIST 2nd Post-Quantum Cryptography Standardization Conference 2019*, 2019.

[212] M. van Heesch, N. van Adrichem, T. Attema and T. Veugen, "Towards Quantum-Safe VPNs and Internet", 2019, `https://eprint.iacr.org/2019/1277`.

[213] DNS-Violations, *DNS flag day 2020*, `https://dnsflagday.net/2020/`, Jan. 2020.

[214] N. I. o. S. US Department of Commerce and T. (NIST), *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*, `https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf`, 2016.

[215] ——, *Post-Quantum Cryptography: Round 2 Submissions*, `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`, Jan. 2020.

[216] M. Müller, J. de Jong, M. van Heesch, B. Overeinder and R. van Rijswijk Deij, *Measurement Code: Custom Performance Test*, `https://github.com/SIDN/pqc-dnssec-measurement`, 2020.

[217] N. I. o. S. US Department of Commerce and T. (NIST), "Recommendation for Key Management: Part 1 – General", Special Publication 800-57 P.1 Rev.5, May 2020.

[218] D. Kales and G. Zaverucha, *Forgery Attacks on MQDSSv2.0*, https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/official-comments/MQDSS-round2-official-comment.pdf, Aug. 2019.

[219] M. Bardet, M. Bros, D. Cabarcas, P. Gaborit, R. Perlner, D. Smith-Tone, J.-P. Tillich and J. Verbel, *Improvements of algebraic attacks for solving the rank decoding and minrank problems*, 2020. arXiv: 2002.08322 [cs.CR].

[220] R. Perlner and D. Smith-Tone, *Rainbow band separation is better than we thought*, Cryptology ePrint Archive, Report 2020/702, https://eprint.iacr.org/2020/702, 2020.

[221] M. Ajtai, "Generating Hard Instances of Lattice Problems (Extended Abstract)", in *In Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, ACM, 1996, pp. 99–108.

[222] L. Groot Bruinderink, A. Hülsing, T. Lange and Y. Yarom, "Flush, Gauss, and Reload – A Cache Attack on the BLISS Lattice-Based Signature Scheme", Springer Berlin Heidelberg, 2016, pp. 323–345.

[223] P. Ravi, M. P. Jhanwar, J. Howe, A. Chattopadhyay and S. Bhasin, *Side-channel Assisted Existential Forgery Attack on Dilithium - A NIST PQC candidate*, Cryptology ePrint Archive, Report 2018/821, https://eprint.iacr.org/2018/821, 2018.

[224] S. McCarthy, J. Howe, N. Smyth, S. Brannigan and M. O'Neill, *Bearz attack falcon: Implementation attacks with countermeasures on the falcon signature scheme*, Cryptology ePrint Archive, Report 2019/478, https://eprint.iacr.org/2019/478, 2019.

[225] R. Gieben and W. Mekking, *Authenticated Denial of Existence in the DNS*, RFC 7129 (Informational), RFC, Fremont, CA, USA: RFC Editor, Feb. 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7129.txt.

[226] Alexa, *Top 1M sites from 2020-07-14*, http://s3.dualstack.us-east-1.amazonaws.com/alexa-static/top-1m.csv.zip, Feb. 2020.

[227] G. C. Moura, M. Müller, M. Davids, M. Wullink and C. Hesselman, "Fragmentation, Truncation, and Timeouts: Are Large DNS Messages Falling to Bits?", in *Passive and Active Measurement (PAM 2021)*, 2021.

[228] R. van Rijswijk-Deij, A. Sperotto and A. Pras, "DNSSEC and its potential for DDoS attacks", in *Proceedings of the Internet Measurement Conference (IMC), 2014*, Vancouver, BC, Canada: ACM Press, 2014.

[229] P. Foremski, O. Gasser and G. C. Moura, "DNS Observatory: The Big Picture of the DNS", in *Proceedings of the Internet Measurement Conference*, 2019, pp. 87–100.

[230] Cloudflare, *ECDSA: The missing piece of DNSSEC*, https://www.cloudflare.com/dns/dnssec/ecdsa-and-dnssec/, Dec. 2019.

[231] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler and D. Stehlé, *CRYSTALS-Dilithium, Algorithm Specifications and Supporting Documentation*, https://pq-crystals.org/dilithium/data/dilithium-specification-round2.pdf, Mar. 2019.

[232] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte and Z. Zhang, *Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU*, https://falcon-sign.info/falcon.pdf.

[233] R. team, *Modified Parameters of Rainbow in Response to a Refined Analysis of the Rainbow Band Separation Attack by the NIST Team and the Recent New MinRank attacks*, https://sites.google.com/site/jintaiding/nist-papers, 2020.

[234] A. Casanova, J. Faugère, G. Macario-Rat, J. Patarin, L. Perret and J. Ryckeghem, *GeMSS: A Great Multivariate Short Signature*, https://www-polsys.lip6.fr/Links/NIST/changes{_}round2{_}V2.pdf, 2020.

[235] D. Bernstein, C. Dobraunig, M. Eichlseder, S. Fluhrer, S. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl, T. Lange, M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld and P. Schwabe, *Sphincs+*, https://sphincs.org/data/sphincs+-specification.pdf, Nov. 2017.

[236] M. Chase, D. Derler, S. Goldfeder, J. Katz, V. Kolesnikov, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, X. Wang and G. Zaverucha, *The Picnic Signature Scheme, Design Document v2.1*, https://github.com/microsoft/Picnic/blob/master/spec/design-v2.1.pdf, Jul. 2019.

[237] D. J. Bernstein and T. Lange, *eBACS: ECRYPT Benchmarking of Cryptographic Systems*, https://bench.cr.yp.to/results-sign.html, Skylake (506e3)-sand, 2020. (visited on 27/07/2020).

[238] J. Hoffstein, J. Pipher and J. H. Silverman, "Ntru: A ring-based public key cryptosystem", in *Lecture Notes in Computer Science*, Springer-Verlag, 1998, pp. 267–288.

[239] J. Ding, M. Chen, A. Petzoldt, D. Schmidt and B. Yang, *Rainbow*, https://http://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/submissions/Rainbow-Round2.zip.

[240] J. Patarin, "The Oil and Vinegar Signature Scheme", *Dagstuhl Workshop on Cryptography*, Sep. 1997.

[241] J. Ding, M.-S. Chen, A. Petzoldt and B.-Y. Schmidt Dieter Yang, *Intellectual property statements Rainbow*, https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/ip-statements/Rainbow-Statements.pdf, 2017.

[242] A. Casanova, J. Faugere, G. Macario-Rat, J. Patarin, L. Perret and J. Ryckeghem, *GeMSS: A Great Multivariate Short Signature*, www-polsys.lip6.fr/Links/NIST/GeMSS{_}specification{_}round2.pdf, 2019.

[243] W. Toorop, M. Müller and T. Chung, *Measuring the impact of DNS Flag Day*, https://blog.apnic.net/2019/08/19/measuring-the-impact-of-dns-flag-day/, Aug. 2019.

[244] C. Deccio and J. Davis, "Dns privacy in practice and preparation", in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, 2019, pp. 138–143.

[245] J. Dickinson, S. Dickinson, R. Bellis, A. Mankin and D. Wessels, *DNS Transport over TCP - Implementation Requirements*, RFC 7766 (Proposed Standard), RFC, Updated by RFC 8490, Fremont, CA, USA: RFC Editor, Mar. 2016. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7766.txt.

[246] P. Hoffman and J. Schlyter, *The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA*, RFC 6698 (Proposed Standard), RFC, Updated by RFCs 7218, 7671, Fremont, CA, USA: RFC Editor, Aug. 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6698.txt.

[247] DNSCrypt, *DNSCrypt: Frequently Asked Questions*, https://dnscrypt.info/faq, Feb. 2020.

[248] N. Bindel, U. Herath, M. McKague and D. Stebila, "Transitioning to a Quantum-Resistant Public Key Infrastructure", in *Post-Quantum Cryptography*, T. Lange and T. Takagi, Eds., Cham: Springer International Publishing, 2017, pp. 384–405.

[249] NLnet Labs, *DNSThought Live Measurements*, https://dnsthought.nlnetlabs.nl/, 2020.

[250] W. Beullens, B. Preneel, A. Szepieniec and F. Vercauteren, *LUOV; Signature Scheme proposal for NIST PQC Project*, 2019.

[251] J. Ding, J. Deaton, K. Schmidt, Vishakha and Z. Zhang, "Cryptanalysis of The Lifted Unbalanced Oil Vinegar Signature Scheme", *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1490, 2019.

[252] W. Beullens, "Improved cryptanalysis of UOV and Rainbow", *iacr.org preprint 2020/1343*, 2020.

[253] The Rainbow Team, *Response to Recent Paper by Ward Beullens*, http://precision.moscito.org/by-publ/recent/response-ward.pdf, 2020.

[254] D. Moody, *Diversity of signature schemes*, PQC-Forum: https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/2LEoSpskELs/m/VB1jng0aCAAJ, 2021.

[255] ICANN, Office of the CTO, *Staff Report of Public Comment Proceeding: Proposal for Future Root Zone KSK Rollovers*, https://www.icann.org/en/system/files/files/report-comments-proposal-future-rz-ksk-rollovers-07aug20-en.pdf, Aug. 2020.

[256] K. Kwiatkowski, *Towards Post-Quantum Cryptography in TLS*, blog.cloudflare.com/towards-post-quantum-cryptography-in-tls/, Jun. 2019.

[257] C. Partridge and M. Allman, "Ethical Considerations in Network Measurement Papers", *Communications of the ACM*, vol. 59, no. 10, pp. 58–64, 2016.

About the Author

Moritz Christian Müller was born in Stuttgart, Germany on 2 January 1989. After completing his vocational education to become an IT Specialist in System Integration in 2008 he obtained his Bachelor's degree in Computer Science and Media at the Hochschule der Medien (University of Applied Science) in 2013. In 2014, and while attending the EIT Digital double degree Master program in Security and Privacy, he studied for one year at the University of Trento, Italy[1] and finally arrived in the Netherlands, receiving his Master degree at the University of Twente[2]. Since 2015, Moritz is working in the research team of SIDN, *SIDN Labs*. He first joined SIDN Labs as an intern and became a full time research engineer after his graduation. In 2017, he started pursuing his PhD at the *Design and Analysis of Communication Systems (DACS)* group at the University of Twente next to his job at SIDN Labs.

Already before joining DACS, Moritz worked on research in the field of DNS security and stability at SIDN Labs. During his PhD, he focused mainly on the DNS Security Extensions (DNSSEC) and its complexities, but also collaborated on other DNS related research, e.g. domain name abuse and DNS privacy. He was supervised by Prof. Aiko Pras, and co-supervised by Prof. Roland van Rijswijk-Deij and Cristian Hesselman.

[1] Highest mountain of the province: Marmolada, 3,343 m above sea level.
[2] Highest mountain of the province: Tankenberg, 85 m above sea level.

PUBLICATIONS BY THE AUTHOR

This is a comprehensive list of publications by the author, sorted in reverse chronological order.

- G.C.M. Moura, M. Müller, M. Davids, M. Wullink, C. Hesselman. *Fragmentation, Truncation, and Timeouts: Are Large DNS Messages Falling to Bits?* In: Passive and Active Network Measurement Conference, PAM 2021, 29-31 March 2021, Berlin, Germany.

- M. Müller, W. Toorop, T. Chung, J. Jansen, R. van Rijswijk-Deij. *The Reality of Algorithm Agility: Studying the DNSSEC Algorithm Life-Cycle.* In: Internet Measurement Conference, IMC 2020, 27-29 October 2020, Pittsburgh, USA.

- M. Müller, J. de Jong, M. van Heesch, B. Overeinder, R. van Rijswijk-Deij. *Retrofitting Post-Quantum Cryptography in Internet Protocols: A Case Study of DNSSEC.* In: ACM SIGCOMM Computer Communication Review (CCR) 50 (4), 2020.

- M. Müller, M. Thomas, D. Wessels, W. Hardaker, T. Chung, W. Toorop, R. van Rijswijk-Deij. *Roll, Roll, Roll your Root: A Comprehensive Analysis of the First Ever DNSSEC Root KSK Rollover.* In: Internet Measurement Conference, IMC 2019, 21-27 October 2019, Amsterdam, the Netherlands — **Distinguished Paper Award**.

- M. Müller, T. Chung, A. Mislove, R. van Rijswijk-Deij. *Rolling with Confidence: Managing the Complexity of DNSSEC Operations.* In: IEEE Transactions on Network and Service Management (TNSM) 16 (3), 2019.

- W.B. de Vries, Q. Scheitle, M. Müller, W. Toorop, R Dolmans, R. van Rijswijk-Deij. *A First Look at QNAME Minimization in the Domain Name System.* In: Passive and Active Network Measurement Conference, PAM 2019, 27-29 March 2019, Puerto Varas, Chile — **Best Open Dataset Award**.

- G.C.M. Moura, J. Heidemann, M. Müller, R. de O. Schmidt, M. Davids. *When the dike breaks: Dissecting DNS defenses during DDoS.* In: Internet Measurement Conference, IMC 2018, 31 October - 2 November 2018, Boston, USA.

- M. Müller, R. van Rijswijk-Deij. *Towards an Independent and Resilient DNS.* In: 12th International Conference on Autonomous Infrastructure, Management and Security, AIMS 2018 – PhD Session, 4-8 June 2018, Munich, Germany.

- G.C.M. Moura, M. Müller, M. Davids, M. Wullink, C. Hesselman. *Domain Names Abuse and TLDs: From Monetization Towards Mitigation.* In: IFIP/IEEE Symposium on Integrated Network and Service Management, IM 2017, 8-12 May 2017, Lisbon, Portugal.

- M. Müller, G.C.M. Moura, R. de O. Schmidt, J. Heidemann. *Recursives in the Wild: Engineering Authoritative DNS Servers.* In: Internet Measurement Conference, IMC 2017, 1-3 November 2017, London, Great Britain.
- G.C.M. Moura, R. de O. Schmidt, J. Heidemann, W.B. de Vries, M. Müller, L. Wei, C. Hesselman. *Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event.* In: Internet Measurement Conference, IMC 2016, 14-16 November 2016, Santa Monica, USA.
- M. Wullink, M. Müller, M. Davids, G.C.M. Moura, C. Hesselman. *ENTRADA: Enabling DNS Big Data Applications.* In: APWG Symposium on Electronic Crime Research 2016, 1-3 June 2016, Toronto, Canada.
- M. Wullink, G.C.M. Moura, M. Müller, C. Hesselman. *ENTRADA: A High-Performance Network Traffic Data Streaming Warehouse.* In: IEEE/IFIP Network Operations and Management Symposium, NOMS 2016, 25-29 April 2016, Istanbul, Turkey.
- G.C.M. Moura, M. Müller, M. Wullink, C. Hesselman. *nDEWS: A New Domains Early Warning System for TLDs.* In: IEEE/IFIP Network Operations and Management Symposium, NOMS 2016, 25-29 April 2016, Istanbul, Turkey.