Rolling With Confidence: Managing the Complexity of DNSSEC Operations

Moritz Müller, Taejoong Chung, Alan Mislove, and Roland van Rijswijk-Deij

***Abstract*—The domain name system (DNS) is *the* naming system on the Internet. With the DNS security extensions (DNSSECs) operators can protect the authenticity of their domain using public key cryptography. DNSSEC, however, can be difficult to configure and maintain: operators need to replace keys to upgrade their algorithm, react to security breaches or follow key management policies. These tasks are not trivial. If operators do not time changes to their keys right, caching resolvers may not have access to the correct keys, potentially rendering DNS zones unavailable for minutes or hours. While best current practices give abstract guidelines on how to introduce and withdraw keys, information on how to monitor and control actual rollovers in a live environment is lacking. More specifically, it is challenging for operators to know *when* to introduce or withdraw keys based on the state of the network. Our main contribution is to help operators answer this question and to address this barrier for deploying DNSSEC. We develop a method with which operators can monitor the replacement of DNSSEC keys, called a *rollover*. Thereby, they can make confident decisions during the rollover and make sure their zone stays available at all times. We validate the method with an algorithm rollover of the Swedish TLD *.se* and provide an open source tool with which operators can monitor their rollover themselves.**

***Index Terms*—DNS, DNSSEC, automation, key rollover, key management, monitoring.**

I. Introduction

THE DOMAIN Name System (DNS) is the hierarchical, distributed naming system of the Internet. The original design of the DNS does not include mechanisms to guarantee the authenticity and integrity of information stored in the DNS. The DNS Security Extensions (DNSSEC) address this shortcoming but also introduce a previously unknown level of complexity [1].

DNSSEC is a public key infrastructure that builds on top of DNS. Operators of domains, such as `example.com`, sign information stored in their zone with their private key. A resolver that looks up the information of `example.com` can validate the signature with the public key and returns an error if this validation fails.

Operators continuously need to keep DNSSEC-signed domains up to date. The most common task is the regular refresh of signatures (which have a limited validity). Less frequent, but more complex tasks include changing keys due to a security breach, upgrading to a new key algorithm, or following a key-management policy [2]. These procedures, called *key rollovers*, include adding and withdrawing keys and signatures across multiple stages, and are a *"fact of life when using DNSSEC"* [3]. Even the main key of the root zone of the DNS is in the process of being replaced over 2018 and early 2019 – for the first time since the deployment of DNSSEC [4].

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* [5]. It not only affected the TLD itself but also the over 5 million domains that were registered under *.nl* at that time. This demonstrates how complicated and critical rollovers are, and might be one of the reasons for the low adoption of DNSSEC [6].

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 article 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 deploying DNSSEC, and thus increasing the overall security of the DNS. 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 [7].

Manuscript received June 29, 2018; revised February 28, 2019 and May 2, 2019; accepted May 4, 2019. Date of publication May 10, 2019; date of current version September 9, 2019. This work was supported in part by National Science Foundation grant 1563320. The associate editor coordinating the review of this paper and approving it for publication was F. De Turck. *(Corresponding author: Moritz Müller.)*

M. Müller and R. van Rijswijk-Deij are with the Design and Analysis of Communication Systems, University of Twente, 7522 NH Enschede, The Netherlands (e-mail: moritz.muller@sidn.nl).

T. Chung is with the Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623 USA.

A. Mislove is with Cybersecurity and Privacy Institute, Northeastern University, Boston, MA 02115 USA.

Digital Object Identifier 10.1109/TNSM.2019.2916176

The *.se* ccTLD has a high DNSSEC penetration, with over half of all domains being signed [8], including domains for banks, government and other services. Equally, DNSSEC validation is common in Sweden, with over 70% of users using validating DNS resolvers [9]. 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 registry of the Brazilian ccTLD *.br* used this tool to monitor their algorithm rollover in August 2018 [10], following our method.

The remainder of the article is organized as follows: Section II introduces the DNS, DNSSEC and key rollovers. Next, Section III describes in detail what can go wrong during a rollover and why. Then, we propose our method in Section IV and validate each stage of the method with the algorithm rollover of *.se* in Section V. Related work is discussed in Section VI. Finally, we summarise our conclusions in Section VII.

II. Background

In this section, we cover the basic principles of DNS and DNSSEC. These are needed to understand why DNSSEC rollovers are risky and how our method addresses these risks.

A. DNS

The DNS uses resource records (RR) to map domain names, such as `example.com`, to values. For example, an `A` record maps a domain name to an IPv4 address, an `NS` record maps a domain name to the authoritative name server for a domain and a `TXT` record contains some string, e.g., instructions for receiving authenticated mail [11]. These records are stored in a zone file. A zone file contains one Start of Authority (`SOA`) RR defining the parameters of the domain and contains a serial number to identify the version of the zone. Operators typically publish their zone at two or more authoritative name servers. Clients that, for example, want to know the IPv4 address of `example.com` use *recursive resolvers* to look up the corresponding `A` record. Recursive resolvers query one of the authoritative name servers of `example.com` to retrieve the record and return it to the client.

The DNS is designed to encourage caching at resolvers. After a resolver receives the response to a query it stores the record for some time in its cache [12]. If the resolver receives another request for the cached record it returns the record directly, without querying the authoritative name servers again, thus improving performance. DNS records contain a time-to-live (TTL) value specifying how long a record may be cached before it should be discarded. After the record is discarded, the resolver has to query the name servers again to retrieve the record. Caching is one of the major reasons why rollovers have to be carried out carefully and we explain *why* in more detail in Section III.

[1] https://www.simpleweb.org/wiki/index.php/Traces

[2] https://github.com/SIDN/rollover-mon.

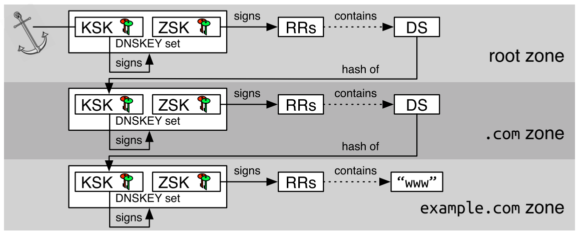


Fig. 1. DNSSEC chain of trust, starting from the root.

B. DNSSEC

With DNSSEC, operators of domains can sign the content of their zone using public key cryptography. Resolvers can validate the signatures and verify whether the content, fetched from the name servers, is correct.

When deploying DNSSEC, operators usually introduce two keys (DNSKEYs) in their zone (see Fig. 1): the Zone Signing Key (ZSK) that is used to sign RRs (such as `A` or `NS` records) and a Key Signing Key (KSK) that only signs the ZSK [3]. The signatures over RRs (`RRSIG`) are published together with the accompanying RRs in the zone and share the same TTL.

1) Establishing the Trust Chain: Validating resolvers, by default, do not trust the keys of `example.com`, but *only* trust the keys of the root zone (configured as a "trust anchor" [13]). The operator of the parent domain, `.com` in Fig. 1, signs a hash of the KSK of `example.com` and publishes it in a `DS` (delegation signer) record in its own zone. The `DS` references the KSK of `example.com` and indicates that the domain is signed. Because the `DS` of the KSK of `.com` is published and signed by the root zone as well, a "chain of trust" between `example.com` and the root is established.

In order to validate records, resolvers need access to every record along the chain of trust of `example.com`. Once a resolver has fetched the records from the servers it tries to validate the signature. If the signatures are valid then the resolver considers the zone "secure". It caches the records and the validation state until the TTL of the record has expired.

2) Validation Failures: The trust chain can break for many reasons. For example, if the `DS` record at the parent does not match the KSK of the child, if a signature has expired, or if a resolver does not have the correct key to validate a signature. The latter case can occur when a resolver has the wrong `DNSKEY` in its cache or if the key is not available at the authoritative name servers.

C. Key Rollovers

Operators that deploy DNSSEC need to roll their 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 [2]. In general, rollovers fall into three categories and vary in complexity: ZSK rollovers,

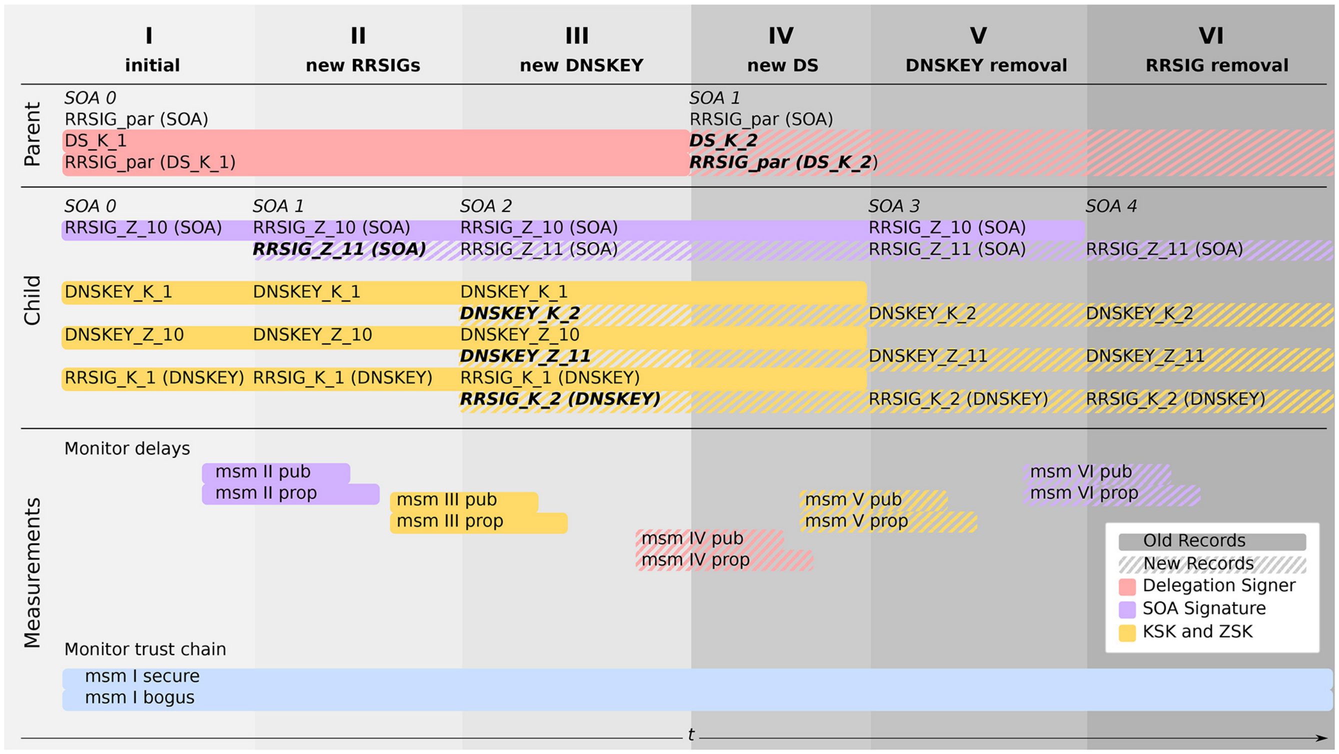


Fig. 2. Stages of an algorithm rollover (as in RFC 6781 [3]), the expected records in each stage in the zone of the parent and of the child, and accompanying measurements to monitor the rollover. Records in bold mark their first appearance.

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 [3]. 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) [3]. 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. Fig. 2 shows the stages of an algorithm rollover over time and we describe the stages in more detail in Section IV.

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 III and discuss which factors have an influence on the timing of the stages.

A special kind of rollover has occured in October 2018, where the KSK of the root zone was replaced [14]. Because this key acts as a trust anchor (the anchor symbol in Fig. 1), it is also configured directly at validating resolvers. Thus, even though the rollover at the root shares stages of rollovers described in this paper, rolling the root KSK is even more challenging and requires additional measurements which are out of scope of this paper.

III. Rollover Failure Modes

Key rollovers are an essential part of operating a signed zone, but also make operating a zone more complicated and risky. If a rollover is not carried out correctly, resolvers can fail validation, thus rendering the zone unavailable. 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.

A. *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.

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; Fig. 2 shows a time-line 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 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 Fig. 2 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*).

1) 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 receives the new record.

2) Propagation Delay: Resolvers do not query for the new record before their local copy 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 a zone 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 [12] 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%) [15].

Fig. 3 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.

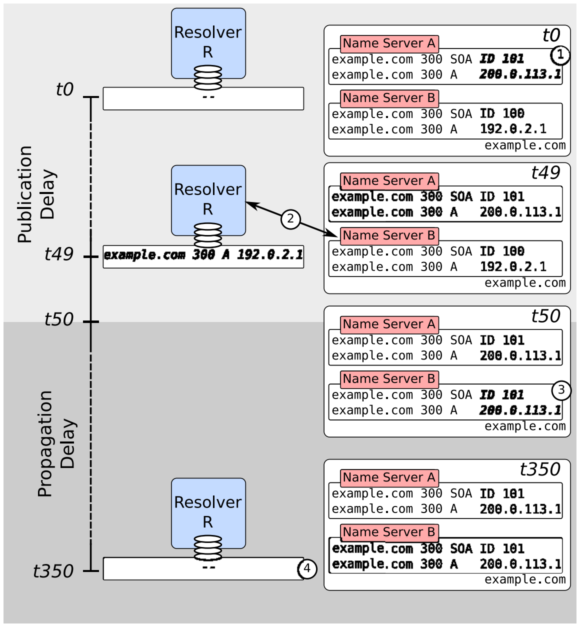

Resolver R
t0
--
Name Server A
example.com 300 SOA ID 101
example.com 300 A 200.0.113.1
1
Name Server B
example.com 300 SOA ID 100
example.com 300 A 192.0.2.1
example.com
Publication Delay
t49
example.com 300 A 192.0.2.1
2
Name Server A
example.com 300 SOA ID 101
example.com 300 A 200.0.113.1
Name Server B
example.com 300 SOA ID 100
example.com 300 A 192.0.2.1
example.com
t50
Name Server A
example.com 300 SOA ID 101
example.com 300 A 200.0.113.1
Name Server B
example.com 300 SOA ID 101
example.com 300 A 200.0.113.1
3
example.com
Propagation Delay
t350
--
4
Name Server A
example.com 300 SOA ID 101
example.com 300 A 200.0.113.1
Name Server B
example.com 300 SOA ID 101
example.com 300 A 200.0.113.1
example.com

Fig. 3. 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.

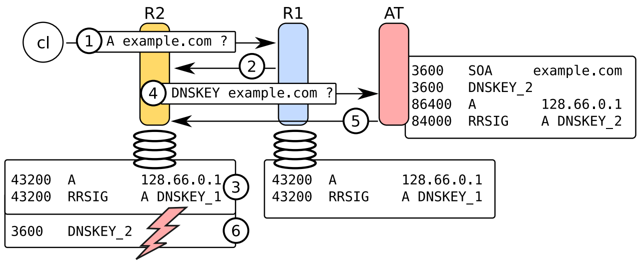


Fig. 4. Ignoring publication delay causes validation error at forwarding and validating recursive resolver.

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.

3) 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. Fig. 4 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

TABLE I
ROLLOVER TYPES AND NECESSARY MEASUREMENTS. ✓ MARKS WHICH STAGES NEED TO BE MONITORED

	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	✓	✓	✓	✓	✓	✓

cached DNSKEY_1, the old ZSK of example.com, it queries R1 again. ④ 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 V-B1.

B. 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"* [16]. 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 [17]. RFC 6781 recommends adding the new signatures before adding the keys which results in two additional stages (II and VI in Fig. 2) when carrying out the algorithm rollover [3]. 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.

IV. 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

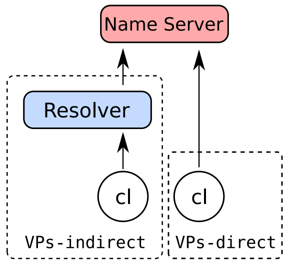


Fig. 5. Vantage points used in the method. VPs-direct receives responses directly from authoritative name servers, VPs-indirect may receive responses from a resolver's cache.

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 Section V-B1). 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. Table I describes which measurements each type of rollover requires.

In the next sections, we describe which vantage points are necessary to monitor the rollover. 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 Section V.

A. Selecting Vantage Points

For measuring the rollover during each stage we require two types of vantage points: (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) (see Fig. 5). 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 III-B).

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, 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 only need one VP of VPs-direct for each authoritative name server of the child and the parent. The exception is the situation in which multiple servers are located behind one address. This is the case when a using a load balancer or a name server is replicated to multiple sites using anycast [18]. Then, 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 V-A, we discuss two of these platforms, RIPE Atlas and Luminati, that when combined provide over 45,000 VPs [19].

B. The Rollover Stages

After selecting vantage points we schedule the measurements. They run 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 Fig. 2). We briefly describe each of them and refer the reader to RFC 6781 [3] for the details:

I: initial	Start of the rollover. Every key in the zone has the same algorithm.
II: new `RRSIGs`	Added new signatures made with the new key and algorithm, but not the new key. This is necessary to prevent errors with strict resolvers (see Section III-B).
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 Fig. 2 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.

C. 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.

1) Monitor the Publication Delay: In each stage we monitor the publication delay by measuring the introduction of the new RRs and the withdrawal of the old ones. We query the servers directly with VPs-direct (msm_II_pub – msm_IV_pub in Fig. 2) and 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 with the new record set. Fig. 2 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.

2) 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 Fig. 2). 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 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.

D. 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 III.

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 III shows the two records, one with a valid, one with a bogus 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 Fig. 2).

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 II).

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

TABLE II
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

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

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.

V. Rollover Validation and Application

In this section we validate our measurement method. We replicate failure modes described in Section III 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 IV provides an overview of the measurements analyzed in this section.

A. 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 RIPE Atlas and Luminati. For replicating the failure modes we only rely on VPs of RIPE Atlas. Both platforms have been used for multiple DNS related measurements in the past, for instance in [20], [21]. In this section we discuss the costs and benefits of these platforms.

1) RIPE Atlas: A RIPE Atlas probe is a device that actively measures Internet connectivity. Volunteers around the world install RIPE Atlas probes in their networks. A probe can send and receive DNS packets and is able to act as a VP for direct and indirect measurements (see Fig. 5). It either sends queries through its pre-configured resolvers or sends queries directly to authoritative name servers. If multiple resolvers are configured, then probes send queries to all of them.

The Regional Internet Registry RIPE regulates the usage of its measurement platform with the help of credits [19]. 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

TABLE III
EXAMPLE OF TEST RECORDS TO VALIDATE THE TRUST CHAIN. MSM_I_SECURE AND MSM_I_BOGUS QUERY THE RESPECTIVE RECORDS

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 IV
MEASUREMENTS TO EVALUATE FAILURE SCENARIOS (SECTION V-B) AND STAGE IV OF THE *.se* ROLLOVER (SECTION V-C)

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

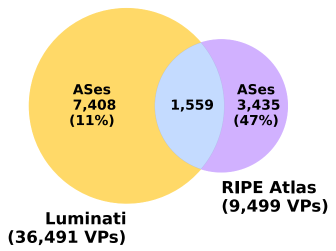


Fig. 6. Unique ASes of VPs during the `.se` Rollover. Share of validating resolvers in brackets.

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.

2) Luminati: Luminati [22] 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 the domain is valid.[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 forwards the response back to the super proxy, which forwards it back to the client. Fig. 7 shows this process schematically. For more details on using Luminati for network measurements, we refer the reader to the study by Chung *et al.* [20].

3) 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 [20]. Hence, RIPE Atlas probes act as both the VPs-direct and VPs-indirect, but Luminati only as the latter.

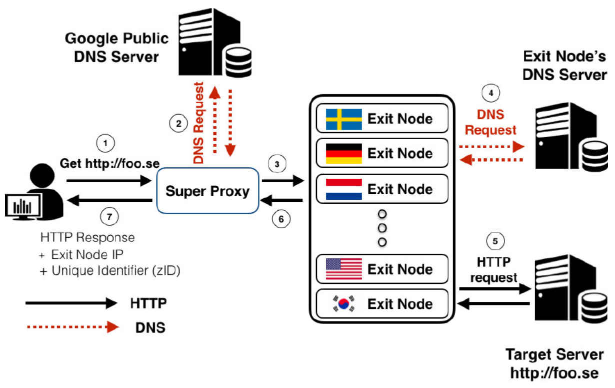


Fig. 7. Timeline of a request in Luminati: the measurement client sends a HTTP request to the super proxy ①; the super proxy makes a DNS request to the Google Public DNS server ②; once the DNS request succeeds, it forwards the HTTP request to one of the exit nodes ③; the exit node makes a DNS request 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.

Probes of RIPE Atlas are very often located behind validating resolvers (see Fig. 6), which is useful to monitor the chain of trust, but also are often not located in residential networks [23]. The opposite is the case for clients of Luminati; the large majority is located in residential networks but only around 12% use validating resolvers [24]. As shown in Fig. 6, 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.

B. Evaluate Failure Modes

In this section, we demonstrate that the issues described in Section III are not only theoretical. We show that operators should indeed monitor rollovers thoroughly, and algorithm

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

rollovers especially, using our method. We use our own second level test domain name (`ourtestdomain.nl.`) and the VPs of RIPE Atlas to replicate different failure modes.

1) Timing Issues: Especially interlinked caches can lead to validation failures during rollovers that are not carried out correctly. We replicate the situation in Section III-A3, where 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

2) 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.

C. 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 `.se` ccTLD was the first ccTLD to deploy DNSSEC [25] 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 [8]. Furthermore, more than 70% of Swedish Internet users rely on validating resolvers [9] to resolve these domains. If the

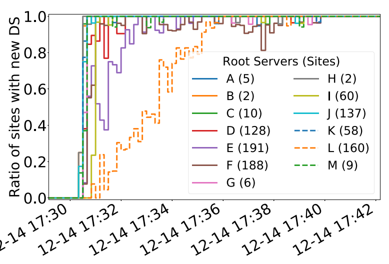


Fig. 8. Share of probes that observe the new `DS` at the Root (#sites in brackets).

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 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 them 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 III-B), this is the first stage where a failure would affect every record in the zone and every validating resolver.

1) 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 Fig. 2).

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 Fig. 8). Note that some root server letters need more time to distribute the new zone 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.

2) Monitor the Propagation Delay: In contrast to the publication delay, the propagation delay, measured with msm_IV_prop in Fig. 2, is significantly longer. Most of the resolvers of VPs-indirect (RIPE Atlas only) pick up the keys within 1 day (see Fig. 9). This is expected since the TTL of the

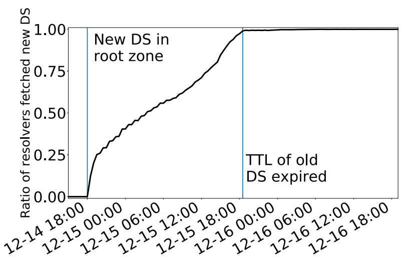


Fig. 9. Share of resolvers that see the new DS, 24 hours after its introduction.

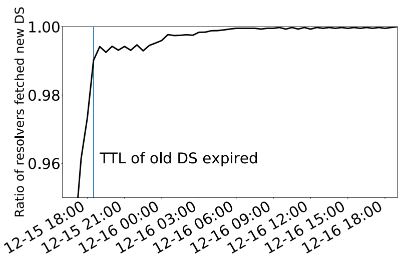


Fig. 10. Share of resolvers that only observe the new DS after the TTL of the old DS has expired.

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 Fig. 10) [26]. 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.

3) 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 Fig. 2).

As described in Section II-B, 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.

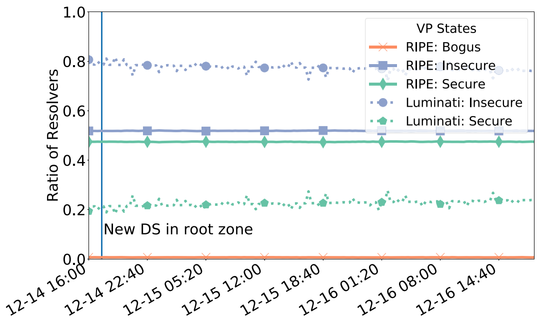


Fig. 11. State of VPs that successfully validated signed .se domains before and after the new DS was introduced.

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. Fig. 11 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.

4) 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 [27]. 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 [10]. In October 2018, 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 [28]. This also demonstrates that the number of resolvers that follow the strict approach for the rollover (as described in Section V-B2) is not significant enough to jeopardize a liberal rollover.

VI. 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.

A. 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 and Mitchell 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 [29], [30].

Chung *et al.* measure DNSSEC rollovers on second level domains in `.com`, `.net` and `.org` over a period of 21 months [24]. 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-Deij *et al.* analyze how elliptic curve cryptography can address this risk [31].

B. 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 [32] is now considered obsolete and is updated by RFC 6781 [3]. 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 [33] 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.

C. 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 [34], [35]. 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 [9].

D. 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 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 [36]. 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 [37]. 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 V-C, 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 automatically keeps track of DNSSEC keys and handles DNSSEC signing [38]. It can also automate rollovers to some extent. If operators pre-configure their publication and propagation delays, OpenDNSSEC can carry out ZSK rollovers automatically. 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 [39].

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` [40]. 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 [6], [41]–[43].

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.

VII. CONCLUSION

In this article, we have demonstrated the complexity of DNSSEC rollovers 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 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 [7].

Our method provides the final link to fully automate DNSSEC rollovers. As shown in Section VI, 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.

Acknowledgment

The authors would like to thank the operator of the Swedish ccTLD `.se` IIS, especially Ulrich Wisser and Roger Murray. Also, they would like to thank Emile Aben and Philip Homburg from RIPE for their support with the Atlas measurement network. Next, they thank Marco Davids and Jelte Jansen from SIDN and Anna Sperotto and Aiko Pras from the University of Twente for their valuable feedback. Finally, they thank the anonymous reviewers for their helpful suggestions and constructive feedback.

References

[1] W. Lian, E. Rescorla, H. Shacham, and S. Savage, "Measuring the practical impact of DNSSEC deployment," in *Proc. USENIX Security Symp.*, Aug. 2013, pp. 573–588.

[2] R. Chandramouli and S. Rose, *Secure Domain Name System (DNS) Deployment Guide*, document SP 800-81, NIST, Gaithersburg, MD, USA, Sep. 2006.

[3] O. Kolkman, W. Mekking, and R. Gieben, "DNSSEC operational practices, version 2," Internet Eng. Task Force, Fremont, CA, USA, RFC 6781, pp. 1–71, Dec. 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6781.txt

[4] R. van Rijswijk-Deij, T. Chung, D. R. Choffnes, A. Mislove, and W. Toorop, "The root canary: Monitoring and measuring the DNSSEC root key rollover," in *Proc. SIGCOMM Posters Demos*, Aug. 2017, pp. 63–64.

[5] M. Davids. (Oct. 2012). *Unbound Mailing List: DNSSEC Validation Failure of .nl TLD*. [Online]. Available: https://unbound.net/pipermail/unbound-users/2012-October/002676.html

[6] T. Chung *et al.*, "Understanding the role of registrars in DNSSEC deployment," in *Proc. ACM Conf. Internet Meas. Conf.*, Nov. 2017, pp. 369–383.

[7] J. Andersson. (Apr. 2018). *Lessons Learned From the .SE Algorithm Rollover*. [Online]. Available: https://www.iis.se/se-tech/lessons-learned-from-the-se-algorithm-rollover/

[8] T. Le, R. van Rijswijk-Deij, L. Allodi, and N. Zannone, "Economic incentives on DNSSEC deployment: Time to move from quantity to quality," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, Apr. 2018, pp. 1–9.

[9] G. Huston. (May 2014). *Measuring the DNS From the Users Perspective*. [Online]. Available: https://ripe68.ripe.net/presentations/164-2014-05-14-huston-dns-measurements.pdf

[10] F. A. C. Neves, ".br DNSSEC algorithm rollover," in *Proc. ICANN 61 DNSSEC Workshop*, San Juan, Puerto Rico, Mar. 2018. [Online]. Available: https://static.ptbl.co/static/attachments/169303/1520891993.pdf?1520891993

[11] M. Kucherawy and E. Zwicky, Eds., "Domain-based message authentication, reporting, and conformance (DMARC)," Internet Eng. Task Force, Fremont, CA, USA, RFC 7489, pp. 1–73, Mar. 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7489.txt

[12] P. Mockapetris, "Domain names—Implementation and specification," Internet Eng. Task Force, Fremont, CA, USA, RFC 1035, pp. 1–55, Nov. 1987. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1035.txt

[13] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS security introduction and requirements," Internet Eng. Task Force, Fremont, CA, USA, RFC 4033, pp. 1–21, Mar. 2005. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4033.txt

[14] *First Root KSK Rollover Successfully Completed*, ICANN, Los Angeles, CA, USA, Oct. 2018. [Online]. Available: https://www.icann.org/news/announcement-2018-10-15-en

[15] *IIS Zone Data*, Internetstiftelsen i Sverige, Stockholm, Sweden, May 2018. [Online]. Available: https://zonedata.iis.se/

[16] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Protocol modifications for the DNS security extensions," Internet Eng. Task Force, Fremont, CA, USA, RFC 4035, pp. 1–53, Mar. 2005. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4035.txt

[17] S. Bortzmeyer. (Feb. 2015). *Unbound Mailing List: BIND Validates but Not Unbound: Who Is Right?* [Online]. Available: https://unbound.net/pipermail/unbound-users/2015-February/003778.html

[18] J. Abley and K. Lindqvist, "Operation of anycast services," Internet Eng. Task Force, Fremont, CA, USA, RFC 4786, pp. 1–24, Dec. 2006. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4786.txt

[19] RIPE NCC Staff, "RIPE atlas: A global Internet measurement network," *Internet Protocol J.*, vol. 18, no. 3, Sep. 2015, pp. 2–26.

[20] T. Chung, D. Choffnes, and A. Mislove, "Tunneling for transparency: A large-scale analysis of end-to-end violations in the Internet," in *Proc. ACM Conf. Internet Meas. Conf.*, pp. 199–213, Oct. 2016.

[21] M. Müller, G. C. Moura, R. D. O. Schmidt, and J. Heidemann, "Recursives in the wild: Engineering authoritative DNS servers," in *Proc. ACM Conf. Internet Meas.*, Nov. 2017, p. 7.

[22] Luminati IO. (May 2018). *Residential IP and Proxy Service for Businesses*. [Online]. Available: https://luminati.io/

[23] 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 *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manag. (IM)*, May 2017, pp. 37–44.

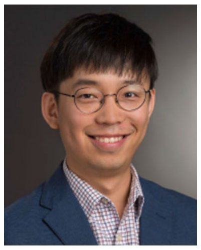
[24] T. Chung *et al.*, "A longitudinal, end-to-end view of the DNSSEC ecosystem," in *Proc. 26th USENIX Security Symp. (USENIX Security)*, Aug. 2017, pp. 1307–1322. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/chung

[25] NLnet Labs. (Sep. 2013). *A Short History of DNSSEC*. [Online]. Available: https://www.nlnetlabs.nl/projects/dnssec/history.html

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

[26] W. Kumari and P. Hoffman, "Decreasing access time to root servers by running one on loopback," Internet Eng. Task Force, Fremont, CA, USA, RFC 7706, pp. 1–12, Nov. 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7706.txt
[27] 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 *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, Jun. 2018, pp. 1–8.
[28] C. Kuroiwa. (Dec. 2018). *Algorithm Rollover on .br, GTER 48/GTS 32*. [Online]. Available: ftp://ftp.registro.br/pub/gts/gts32/01-br-algorithm-rollover.pdf
[29] S. Ariyapperuma and C. J. Mitchell, "Security vulnerabilities in DNS and DNSSEC," in *Proc. IEEE 2nd Int. Conf. Availability Rel. Security (ARES)*, Vienna, Austria, Apr. 2007, pp. 335–342.
[30] H. Yang, E. Osterweil, D. Massey, S. Lu, and L. Zhang, "Deploying cryptography in Internet-scale systems: A case study on DNSSEC," *IEEE Trans. Depend. Secure Comput.*, vol. 8, no. 5, pp. 656–669, Sep./Oct. 2011.
[31] R. van Rijswijk-Deij, A. Sperotto, and A. Pras, "Making the case for elliptic curves in DNSSEC," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 13–19, Oct. 2015.
[32] O. Kolkman and R. Gieben, "DNSSEC operational practices," Internet Eng. Task Force, Fremont, CA, USA, RFC 4641, pp. 1–35, Sep. 2006. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4641.txt
[33] S. Morris, J. Ihren, J. Dickinson, and W. Mekking, "DNSSEC key rollover timing considerations," Internet Eng. Task Force, Fremont, CA, USA, RFC 7583, pp. 1–31, Oct. 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7583.txt
[34] *DNSViz*, Sandia Nat. Libraries Verisign Inc., Albuquerque, NM, USA, Jan. 2018. [Online]. Available: https://github.com/dnsviz/dnsviz
[35] IIS and Afnic. (Jan. 2018). *Zonemaster*. [Online]. Available: https://github.com/dotse/zonemaster
[36] *BIND DNSSEC Guide*, Internet Syst. Consortium, Inc., Redwood City, CA, USA, Jan. 2017. [Online]. Available: https://ftp.isc.org/isc/dnssec-guide/dnssec-guide.pdf
[37] Knot-DNS. (Feb. 2019). *Knot DNS Documentation: Operation—DNSSEC Key Rollovers*. [Online]. Available: https://www.knot-dns.cz/docs/2.7/html/operation.html#dnssec-key-rollovers
[38] OpenDNSSEC Project. (Feb. 2018). *OpenDNSSEC*. [Online]. Available: https://github.com/opendnssec/opendnssec
[39] Secure64. (Jan. 2016). *Secure64 DNS Signer*. [Online]. Available: https://secure64.com
[40] W. Kumari, O. Gudmundsson, and G. Barwood, "Automating DNSSEC delegation trust maintenance," Internet Eng. Task Force, Fremont, CA, USA, RFC 7344, pp. 1–18, Sep. 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7344.txt
[41] S. Isasi and V. Shrestha. (Sep. 2018). *Expanding DNSSEC Adoption*. [Online]. Available: https://blog.cloudflare.com/automatically-provision-and-maintain-dnssec/
[42] Knot-DNS. (Feb. 2019). *Automated DNSSEC Provisioning: Guidelines for CDS Processing at SWITCH*. [Online]. Available: https://www.nic.ch/export/shared/.content/files/SWITCH_CDS_Manual_en.pdf
[43] J. Talir, *Automating DNSSEC via CDNSKEY Processing*, RIPE, Amsterdam, The Netherlands, Oct. 2017.

Moritz Müller received the M.S. degree from the University of Twente and the University of Trento in 2015. He is currently pursuing the Ph.D. degree with the Design and Analysis of Communication Systems Group, University of Twente. He is a Research Engineer with SIDN Labs, the research department of the *.nl* registry. He focuses on research to improve the security and stability of the DNS and the Internet in general. He puts his research into practice by supporting operators improving their infrastructure and deploying new standards.



Taejoong (Tijay) Chung received the Ph.D. degree in computer science and engineering from Seoul National University in 2015. He is an Assistant Professor with the Computer Science Department, Rochester Institute of Technology. His research focuses on Internet security, privacy implications, and Internet measurement. He was a recipient of the IRTF Applied Networking Research Prize in 2019, the USENIX Security Distinguished Paper Award in 2017, and the Best Paper Award at IEEE Computer Society Seoul Chapter in 2010.

Alan Mislove received the Ph.D. degree in computer science from Rice University in 2009. In 2009, he joined Northeastern University, where he is a Professor. His research concerns distributed systems and networks, with a focus on using social networks to enhance the security, privacy, and efficiency of newly emerging systems. His research comprises over 50 peer-reviewed papers and has received over 12 000 citations. He was a recipient of the IETF Applied Networking Research Prize in 2018 and 2019.

Roland van Rijswijk-Deij received the Ph.D. degree in computer science from the University of Twente in 2017, where he is an Assistant Professor of network security with the Design and Analysis of Communication Systems Group. He is also a Principal Scientist with NLnet Labs, a not-for-profit foundation that develops open source software for core Internet protocols, including DNS and RPKI. His research interests include network security and network measurements, with a particular interest in DNS and DNSSEC.