From the Beginning: Key Transitions in the First 15 Years of DNSSEC

Eric Osterweil, Pouyan Fotouhi Tehrani, Thomas C. Schmidt, *Member, IEEE*, and Matthias Wählisch, *Member, IEEE*

***Abstract*—When the global rollout of the DNS Security Extensions (DNSSEC) began in 2005, a first-of-its-kind trial started: The complexity of a core Internet protocol was magnified in favor of better security for the overall Internet. Thereby, the scale of the loosely-federated delegation in DNS became an unprecedented cryptographic key management challenge. Though fundamental for current and future operational success, our community lacks a clear notion of how to empirically evaluate the process of securely transitioning keys. In this paper, we propose two building blocks to formally characterize and assess key transitions. First, the *anatomy of key transitions*, i.e., measurable and well-defined properties of key changes; and second, a novel *classification model* based on this anatomy for describing key transition practices in abstract terms. This abstraction allows for classifying operational behavior. We apply our proposed transition anatomy and transition classes to describe the global DNSSEC deployment. Specifically, we use measurements from the first 15 years of the DNSSEC rollout to detect and understand which key transitions have been used to what degree and which rates of errors and warnings occurred. In contrast to prior work, we consider all possible transitions and not only 1:1 key rollovers. Our results show measurable gaps between prescribed key management processes and key transitions in the wild. We also find evidence that such noncompliant transitions are needed in operations.**

***Index Terms*—Domain name system, DNSSEC, PKI, key rollover, Internet measurement, information security.**

I. Introduction

KEY TRANSITIONS are critical for cryptographically enhanced infrastructures at the Internet-scale. The Internet is composed of loosely-federated administrative domains, and managing cryptographic keys under those conditions raises operational challenges in particular whenever domains depend on one another. Mismanagement of cryptography can lead to security shortfalls in those systems and infrastructures that depend on them (e.g., [1]–[4]). Regarding DNS, Paul Mockapetris and Kevin J. Dunlap (1988) wrote, "Distributing authority for a database does not distribute a corresponding amount of expertise" [5]. DNSSEC, which began its global deployment in 2005, implicitly raised the question: *Can* distributing responsibility to manage cryptographic material *teach us* the corresponding amount of expertise?

In DNSSEC it is common to periodically change the cryptographic keys in use. In some cases this is done as a hygienic prescription, in others it can be an emergency response to a security event (such as a key compromise or a cryptosystem weakness). The changing of keys is a process called key transitioning (or key rollover). In this process, a system gracefully transitions from using keys that are *departing*, i.e., being removed, to using keys that are *remaining*, i.e., kept unchanged or being newly added, while ensuring continuity of protections during the change. Following structured and validated process models for key transitions is critical for maintaining the security assurances of the overall system. Different infrastructures dictate different processes and prescribe them in different ways [6]–[9]. While guidance for DNSSEC key life cycle management and timing exists in RFCs, a more foundational evaluation framework is missing so that the community can quantify and evaluate *all* operational aspects. Such a framework would not only allow us to compare real deployments to prescribed guidance but to each others as well.

In this work, we propose a novel method to precisely model DNSSEC key transitions and apply this to analyze

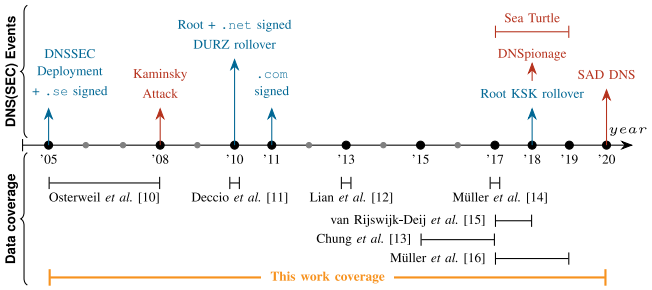


Fig. 1. Notable DNS(SEC) deployment events (blue) and security incidents (red) during the measurement periods of related work (black) and this work (orange).

Manuscript received 10 December 2021; revised 14 April 2022; accepted 14 June 2022. Date of publication 1 August 2022; date of current version 31 January 2023. This work was supported by the Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation and workforce development. For more information about CCI, visit cyberinitiative.org. Additional support in parts was provided by the German Federal Ministry of Education and Research (BMBF) within the projects *Deutsches Internet-Institut* (grant no. *16DII111*) and *PRIMEnet*. The associate editor coordinating the review of this article and approving it for publication was R. Badonnel. *(Corresponding author: Pouyan Fotouhi Tehrani.)*

Eric Osterweil is with the Department of Computer Science, George Mason University, Fairfax, VA 22030 USA (e-mail: eoster@gmu.edu).

Pouyan Fotouhi Tehrani is with Weizenbaum Institute and Fraunhofer FOKUS, 10589 Berlin, Germany (e-mail: pft@ieee.org).

Thomas C. Schmidt is with the Department of Computer Science, Hamburg University of Applied Sciences, 20099 Hamburg, Germany (e-mail: t.schmidt@haw-hamburg.de).

Matthias Wählisch is with the Institute of Computer Science, Freie Universität Berlin, 14195 Berlin, Germany (e-mail: m.waehlisch@fu-berlin.de).

Digital Object Identifier 10.1109/TNSM.2022.3195406

and classify the data from the first 15 years of DNSSEC deployment (2005–2020). We define the composing elements required by our model as the *anatomy of DNSSEC key transitions*. Furthermore, we propose a measurement methodology to quantify key transitions observed in the wild. By using our anatomy and transition model, we are able to model key transition behaviors in the wild from both RFCs [6], [7] and from related work in the literature [17]. Our measurements cover $\approx$ 19 million key transitions. They reveal a significant amount of operational heterogeneity, many of which deviate significantly from standards without necessarily degrading security. Our contributions in this work are threefold:

1) *Anatomy:* We examine the timing features of keys while transitioning and propose an *anatomy of DNSSEC key transitions*, which defines a candidate set of measures that are necessary to measurably characterize key transitions.
2) *Transition classification:* Based on our proposed anatomy, we present a novel methodology, which we use to concisely quantify and analyze key transitions.
3) *Longitudinal study:* To illustrate the generality of this work, we present measurements of operational key transitions that span 15 years of DNSSEC deployment, covering a number of notable events that have not been fully analyzed by related works (see Figure 1).

The remainder of this paper is organized as follows. Section II summarizes background about DNSSEC. In Section III, we propose our key transitions anatomy and explain how we construct our model of transition. We follow this by proposing a security analysis of DNSSEC key life cycle and key transitions in Section IV. We present the measurement corpus used in this work in Section V. Section VI details our methodology to derive a continuous model of the DNSSEC key lifetimes from discrete observations. Section VII introduces our approach to classify life cycle management and transitions of keys. Based on that, we proceed to our extensive use case study in Section VIII. We cover related work in Section IX and discuss our results in Section X. Finally, we conclude in Section XI.

II. Background

The DNS is a hierarchically administered global database of Resource Records (RRs), which are inserted and removed whenever operators choose. The design of the DNS allows the administrator of any sub-tree (called a DNS *zone*) to delegate the management authority of any branch under their zone to another authority. Delegations are implemented when a zone parent adds Name Server entries, i.e., `NS` RRs, that point to the DNS name servers of that sub-zone. This hierarchical delegation allows administrators to operate their zones independently, and requires only a one-time coordination as long as the name servers remain the same.

To compensate for a number of security threats (see [18], [19]), DNS evolved to have the *DNS Security Extensions* (DNSSEC), whose specifications underwent their final round of standardization [20]–[22] in 2005. DNSSEC overloads the hierarchical namespace of DNS with cryptographic key learning and verification. By design, DNSSEC-enabled zones generate and manage their own cryptographic key pairs using any set of DNSSEC standardized cryptosystems. Operators then encode the public portion of their key pair in a new RR type, `DNSKEY`. A DNSSEC signature, an `RRSIG`, is generated by the respective private portion of a `DNSKEY` over each set of same-type RRs, called an *RRSet*, and is always returned with each DNSSEC response. An `RRSIG` specifies its *inception* and *expiration* times to limit its period of validity and to resist replay attacks. As these dates are defined in absolute values, DNSSEC implicitly requires "loose time synchronization" between authoritative nameservers and validating resolvers [19].

DNSSEC specifies that zones should manage two separate classes of `DNSKEY`s: Zone Signing Keys (ZSKs) and Key Signing Keys (KSKs). While the cryptographic material used for these keys is fundamentally identical, their key life cycle management and roles are distinct: whereas ZSKs are used to sign all of a zone's contents (e.g., `A` and `NS` records), KSKs are only used to sign `DNSKEY` RRSets. The root zone uses a well-known, self-signed KSK. All other zones need to have their KSKs authorized by a Delegation Signer (`DS`) record in their parent zone so that keys at each zone can be globally verified by Relying Party (RP) software (also called validating recursive resolvers, or just validators). In this way, validators use the KSK of the DNS Root zone as a Trust Anchor (TA) and point of departure to construct unambiguous verifiable paths to any DNSSEC zone in the hierarchy through recursive tracing of secure delegations. This follows the same way DNS zones are normally resolved recursively via `NS` records. The entire secure delegation chain from the DNS root to delegated zones, the *chain of trust*, is the element that creates operational dependence between the cryptographic management of zones and their parents.

To validate a single RRSet, an RP relies on various pieces of information (i.e., `DNSKEY`s, `RRSIG`s, `DS` records) from possibly different zones and sources (i.e., authoritative servers or caches). Validation of an RRSet in a given zone succeeds if (*i*) a verifiable path to that zone exists and (*ii*) the signature over that RRSet is valid. Breaking the first requirement causes an RP to determine the state of data as insecure, while neglecting the second ends with a bogus state [20]. These two requirements also constrain when and how operators can insert and remove RRs without defeating the protection provided by DNSSEC. Note that a valid signature presupposes that formal cryptographic requirements are met (e.g., digests are correctly calculated), the key generating the signature is included in the `DNSKEY` RRSet, and the set is available and valid for at least the total validity period of the signature. We provide a thorough discussion on temporal constraints with a focus on key transitions in Section IV.

III. Modeling Key Transitions

Key transition refers to the procedure of modifying the set of valid DNS keys over time. The growing literature discusses DNSSEC key transitions in terms of key "rollovers" [14], [16], i.e., a single new key replacing the only existing key. Based

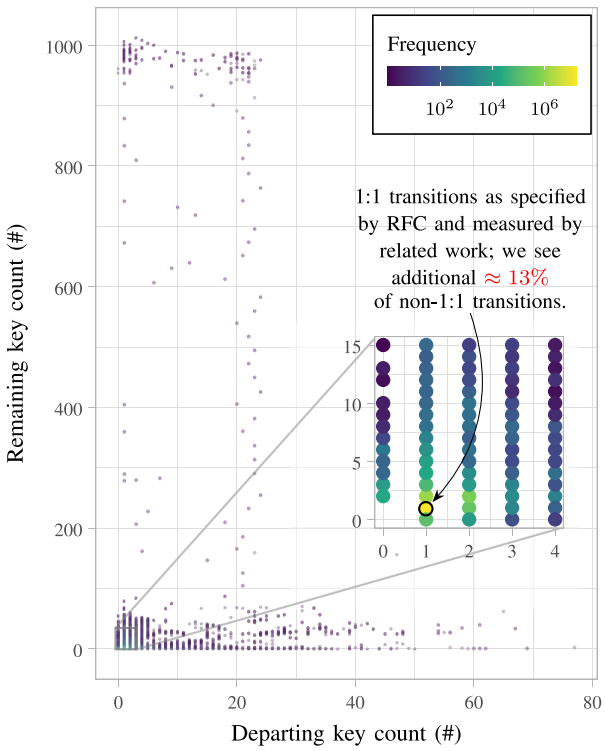


Fig. 2. Frequencies of key transitions with different cardinalities between 2005–2020.

on a consistent monitoring of the first 15 DNSSEC years, we find evidence that the global DNSSEC deployment follows a different reality as is depicted in Figure 2. Besides the expected one-to-one transitions, a notable portion (13%) of transitions involve more than two keys and require a more expressive model. To reflect this reality, we introduce a generic model of key transitions.

In our generic model, a transition is characterized by an effective change of the DNSSEC key set, i.e., a transition succeeds only if the removal of one or more *departing keys* results in an altered set of *remaining keys*. It should be noted that the set of remaining keys includes both newly added keys (if any) and keys which existed throughout the transition. Our model allows us to evaluate both simple key rollovers involving a single departing and a single new key as well as more complex key transitions involving multiple keys, also seen in the wild.

In the following, we introduce a temporal model of DNS keys, which we use to define a *transition anatomy* and provide a method to capture the semantics of key transitions in terms of *transition classes*.

A. Anatomy of a Key Transition

Key transitions are measurable through changes in DNSKEY resource records, and their respective RRSIG records as published by the authoritative servers of the records. In the case of KSKs, changes in DS records require monitoring in the parent zone. While such changes are present, we consider a transition as ongoing.

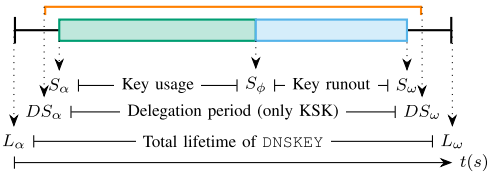


Fig. 3. A temporal model spanning the *total lifetime* of a DNSKEY. RRSIGs generated by the key define active *key usage*, during *key runout* no new signatures are created while existing ones remain valid, and RRSIG(s) covering respective DS records define the key *delegation period*.

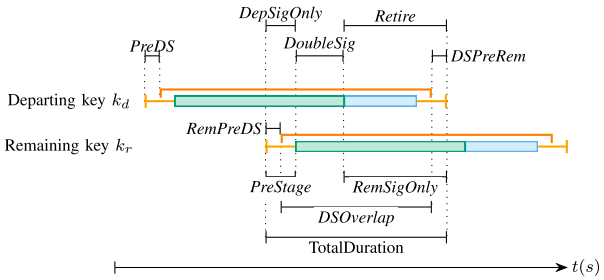


Fig. 4. Anatomy of a 1.1 key transition.

Before we describe the temporal aspects of DNSKEY transitions, we first need to define a life cycle model for DNSKEYs, which adequately describes a key existence from its inception throughout its usage, run-out, and its expiration. These four phases are depicted in Figure 3.

The only temporal information that is explicitly expressed in DNSSEC about resource records is encoded in (*i*) TTL values and (*ii*) the validity period as defined by RRSIGs. TTL values are indicators used by caching resolvers to *locally* determine a time window after which a record should be considered stale and flushed from the cache. The main purpose of TTLs is to establish and maintain eventual consistency in caches [20, Sec. 8.1]. In contrast, the validity of DNSKEYs as denoted through RRSIGs provides information that can be used to reconstruct the life cycle of any key. Accordingly, we create a life cycle model of keys using (*i*) signatures *over* and (*ii*) signatures *generated by* those keys. Using RRSIGs *over* a key (recall that a single DNSKEY can be signed multiple times throughout its lifetime) the total lifetime of that key can be measured. We denote the earliest and latest point in time when the key was singed as L_α and L_ω, respectively. The RRSIGs *generated* by a DNSKEY, in turn, can be used to determine when the key was active and in use. Formally we denote the interval from earliest and latest times that data was verifiable by this key $[S_\alpha, S_\omega]$, the time when the key stopped generating new signatures S_ϕ, and subsequently the duration in which the key was only used to verify existing signatures $[S_\phi, S_\omega]$. Additionally, for KSKs only, the signatures over DS records of the parent zone are used to infer the period in which the parent zone was securely delegating to this key ($[DS_\alpha, DS_\omega]$).

We now use this temporal model to characterize a simple 1:1 key rollover. To this end we define a *transition anatomy* based on the following ten features, which include measures regarding a remaining key k_r and a departing key k_d, as visualized in Figure 4:

TABLE I
TEMPORAL FEATURES OF TRANSITION ANATOMY AND THEIR RESPECTIVE INTERVALS

Feature	Interval
PreDS	$[L_\alpha(k_d), DS_\alpha(k_d)]$
DoubleSig	$[S_\alpha(k_d), S_\phi(k_d)] \cap [S_\alpha(k_r), S_\phi(k_r)]$
PreStage	$[L_\alpha(k_r), S_\alpha(k_r)]$
DepSigOnly	$[S_\alpha(k_d), S_\phi(k_d)] \cap [T_\alpha, T_\omega] - DoubleSig$
Retire	$[S_\phi(k_d), L_\omega(k_d)] \cap [T_\alpha, T_\omega]$
DSOverlap	$[DS_\alpha(k_d), DS_\omega(k_d)] \cap [DS_\alpha(k_r), DS_\omega(k_r)]$
RemSigOnly	$[S_\alpha(k_r), S_\phi(k_r)] \cap [T_\alpha, T_\omega] - DoubleSig$
DsPreRem	$[DS_\omega(k_d), L_\omega(k_d)]$
RemPreDS	$[L_\alpha(k_r), DS_\alpha(k_r)]$
TotalDuration	$[T_\alpha, T_\omega]$
⌞ for ZSK	$[L_\alpha(k_r), L_\omega(k_d)]$
⌞ for KSK	$[min(L_\alpha(k_r), DS_\alpha(k_r)), max(L_\omega(k_d), DS_\omega(k_d)]$

1) *PreDS:* When a corresponding DS record was introduced for k_d in the parent zone.
2) *DoubleSig:* The period during the transition when both departing and remaining keys were *actively* signing.
3) *PreStage:* The time during which the remaining key was valid, but *before* being used to verify zone data.
4) *DepSigOnly:* The duration during the key transition when *only* the departing key was in use.
5) *Retire:* The duration during the key transition when signatures generated by departing key run out (run out described in Figure 3).
6) *DSOverlap:* The duration (if at all) that DS(es) for the departing and remaining keys overlapped.
7) *RemSigOnly:* The duration during the key transition when *only* the remaining key was in use.
8) *DsPreRem:* If the departing key was covered by a DS, the amount of time that the key was valid *after* DS(es) no longer delegated to it.
9) *RemPreDS:* When a corresponding DS record was introduced for the remaining key k_r in the parent zone.
10) *TotalDuration:* Duration of the entire transition.

Table I describes these features, using our life cycle notation of involved keys presented in Figure 3.

Based on our anatomy, we are able to describe arbitrary transitions of *n* departing keys to *m* remaining keys, using $\binom{n+m}{2}$ pairwise 1 : 1 transitions. Our measurements of operational zones over 15 years indicate that this distinction is important. For example, consider a zone of *n* keys (with possibly different initial inception times) and 1 to *n* keys in use to sign data. If that zone transitions to *m* keys (where 1 to *m* are used to sign data), a number of unknowns arise: Which key(s) rolled over to which other keys? Did all the departing keys roll over to all of the remaining keys? If some keys persisted, are they (partially) replaced, as well?

Figure 2 clearly shows that while the majority of observed key transitions change by one key (gaining or losing), many transitions cause alterations of $|m-n| > 1$. These fine granular observations of real deployments illustrate why additional expressiveness is needed, and why many of the previous discussions and characterizations of "rollovers" in the literature apply only to cases in which a single *active* key is rolling over to another single active key.

We extend these discussions by observing that if more than one key is added or departed at the same time, these are multiple concurrent transitions at the same timestamp. The intuition here is that no single departing key is measurably more pivotal than another. Thus, we define each departing key as transitioning to any of the keys that remain. This definition of key transitions allows us to measure operational behavior and answer questions such as: How many active keys are in use? When are transitions aborted or rolled back? When are secure delegations (from DS records) correct? Or, does a zone remain secure (see Section IV) as transition is ongoing. In addition, our transition model measures the relative ages of (the remaining to departing) keys: *newer*, *older*, or the *same* age. These semantics, although unmentioned in the RFCs, are useful in some process models (e.g., [17] discussed below).

Our proposed anatomy is a fine-grained description of the atomic timing components that are necessary and sufficient to fully characterize key transitions. Whether key transitions are being performed manually, part of a process, fully automatized, or result from unsupervised ad hoc changes in a DNSSEC zone, the diversity of their activities is concisely represented by this anatomy.

B. Transition Classes

Our transition anatomy allows for the precise reconstruction and description of any pairwise key transition. Special care needs to be taken when *characterizing* transitions abstractly, as RFC 5011 [6] or RFC 7583 [7] do for example. We discretize the value of each transition feature (see Table I) instead of using their absolute values from empirical measurements. Different combinations of the resulting discretized feature set then represent transition classes. Transition classes allow us to compare and to assess whether prescribed security guarantees are preserved while keys are changing (e.g., by adhering to specifications such as RFCs).

To classify a given transition, we first calculate the interval for each of its features using empirical measurements, e.g., *PreDS* = [1618016262, 1618048662]. We then discretize the features (1) through (9) based on whether their interval widths (see Table I) are < 0, $= 0$, > 0, e.g., *PreDS* > 0. For intervals defined as intersections of other intervals, e.g., *DoubleSig*, the respective interval widths are always non-negative, whereas other measures can assume negative values, e.g., the width of *PreDS* interval can be negative when a KSK is securely delegated *before* being signed and included in the DNSKEY RRset ($L_\alpha(k_d) \geq DS_\alpha(k_d)$). For cases in which an interval is undefined, e.g., measurements of DS records for ZSKs that have no DS record to measure, we use the *N/A* placeholder. In Section VII, we will see that N/A cases do not have an impact on transition classes.

Discretization facilitates an empirically simple comparison of completely independent key transitions. For example, if two keys in a transition are both observed to be signing data at the same time, their observed *DoubleSig* interval width would be a finite value. This would then be discretized as >0. Two other

keys, in another transition (possibly in another zone) would likely have different interval widths, but would be assigned to the same discrete classification value, and would thus enable comparisons between these transitions. As a result, every pairwise key transition can be represented as an ordered set of discretized features.

IV. Security Analysis of Key Transitions

In this section, we discuss the temporal constraints that must be satisfied during key transitions to avoid zone data to be invalidated as insecure or bogus (see Section II). Recall that a single record is considered valid if all RRsets that are necessary for the validation of that record are also valid at the time of verification. To maintain this temporal requirement, each zone operator must ensure that at any point in time (*i*) secure delegations from the parent zone cover at least one KSK that signs the DNSKEY RRset, and (*ii*) all records are signed by at least one valid ZSK that is included in the DNSKEY RRset. Breaking the first condition causes zone data to be invalidated as insecure, and ignoring the second leads to a bogus state. This temporal requirement must also be maintained during key transitions, just as DNSKEY, DS RRsets, and RRSIGs are undergoing changes. The main challenge, however, is not only to maintain a secure state on authoritative nameservers but to ensure that (*i*) RPs can base validations on RRsets that are cached somewhere in the network and still carry valid signatures, and (*ii*) changes at primary authoritative nameservers are not instantaneously observed by relying parties (RPs) [23]. Given a secure data state and a well-functioning RP, we consider a transition as secure if its modifications to keys, signatures, and secure delegations prevent a bogus or insecure state during and after the transition. This definition also accounts for adversaries that can replay valid RRsets that might already be removed from the authoritative nameservers and purged from caches. Our following security analysis considers the relation of a key to its generated signatures, and its relation to other keys and delegation signers.

A. Temporal Relationship of a Key and Its Signatures

Based on our temporal model of DNSSEC keys (see Figure 3), the following condition applies for a key and its generated signatures: $L_\alpha \leq S_\alpha \leq S_\omega \leq L_\omega$. This condition states that no signature validity period should precede or exceed the total (non-zero) lifetime of the key that generated it. By extension, DNSKEY sets that have a valid signature may not be prematurely removed from authoritative nameservers without the risk of introducing inconsistencies through caching or replay attacks. The following example illustrates such a situation: an RP caches the DNSKEY RRset of a zone and its RRSIG(s) at t_0. Before the signature over the key set is expired, the authoritative nameserver replaces the ZSK at t_1, uses the KSK to generate a new signature over the DSNKEY RRset, and finally re-signs all other RRsets in the zone with the new ZSK. At t_2, before either the signature or the TTL of the old DNSKEY expires, the recursive resolver fetches, for example, A records and their signatures generated with the new ZSK from the authoritative nameserver. At this point, the resolver considers the cached key set as valid (TTL spans at t_3) yet fails to validate the retrieved RRset and concludes a bogus state (see Figure 5). Even if the TTL was expired on all caches and the old DNSKEY RRset was purged at t_2, an adversary could still poison RP caches with the old key set and cause a failed validation. This clearly illustrates that TTLs do not affect the foundational security analyses.

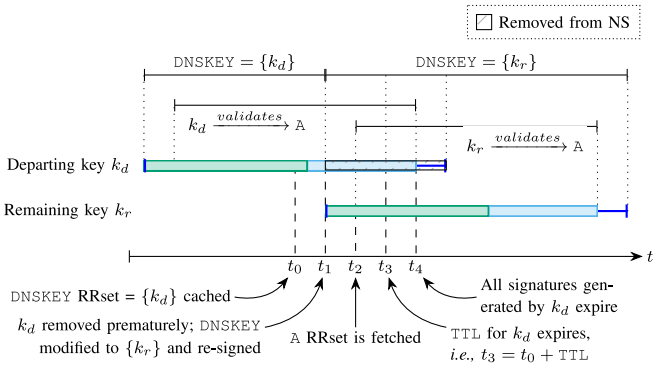


Fig. 5. An example of how modifying and re-signing of an RRset (DNSKEY here) can cause failed validation in caching resolvers, while a valid signature over the RRset still exists.

The scenario of failed validation because of bad timing becomes more severe in case of DS records. When a newly signed DS record is fetched from an authoritative server but the RP uses a departing, cached DNSKEY for validation, a complete child zone becomes insecure. This case, however, is more complicated because the only requirement to maintain the chain of trust is to have at least one active KSK securely delegated even if multiple KSKs are present and are actively signing the DNSKEY RRset.

B. Temporal Relationship Among Keys and Delegation Signers

The second temporal constraint states that involved keys in a transition must have overlapping lifetimes (*TotalDuration* > 0) as changes on primary authoritative nameservers are not instantaneously observable by RPs [23]. Changes to delegation signers must also account for additional delays because as the operator of a zone can only request a modification to DS records to a parent zone, with no control over the timing when exactly changes are applied by that parent zone.

To avoid going bogus, a zone operator must guarantee the continuity of signatures over zone data. As DNSSEC allows for multiple signatures over the same RRset at the same time, it suffices that for each RRsets in a zone to have at least one signature that is valid at all times. Transitions that involve multiple keys must make sure that for any RRset signed by a departing key, a remaining key exists which signs that RRset no later than its signatures expire. The most basic solution for this is to activate keys, i.e., create RRSIGs, as soon as keys are introduced in the zone (*PreStage* $= 0$). This way, any active departing key can stop generating new signatures (runout, i.e., *Retire* > 0) and be removed after its generated signatures are all expired ($L_\omega \geq S_\omega$). This approach, however, expands the zone size as multiple valid signatures are present

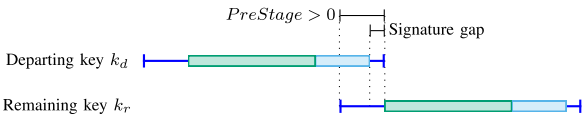


Fig. 6. An example of improperly timed ZSK transition with pre-staged remaining key but belated activation causing bogus validation due to a signature gap.

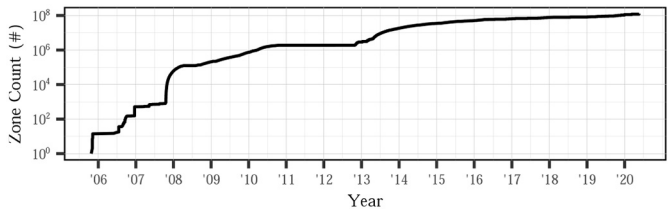


Fig. 7. Total number of monitored DNSSEC-enabled zones.

simultaneously during the transition. To address this, keys can be introduced to a zone and start signing later (*PreStage* > 0) to reduce the time window of overlapping signature. Here, the downside is the more challenging timing in making sure that there is no signature gap for any available RRset. Figure 6 depicts an example where the remaining key is pre-staged but activated too late, thus, leaving a gap during which zone data cannot be validated.

With KSKs, additional care must be given to the timing of `DS` records in order to maintain the chain of trust during the transition and avoid going insecure. In details, this depends on how the KSK is being transitioned: if *PreStage* > 0, the new `DS` record can be added to the parent zone *after* the key is included (*RemPreDs* > 0) but no later than expiration date of previous DS, while the old delegation can be removed before the departing key expires (*DSPreRem* > 0), yet no later than its runout period ends. This approach can be used to minimize the interaction between the zone operator and its parent zone by combining the request to remove the old delegation and adding the new one (*DSOverlap* $= 0$). If *PreStage* $= 0$, any combination of *RemPreDS* and *DSPreRem* that does not cause a gap between delegation signers ($DS_\alpha(k_r) \leq DS_\omega(k_d)$) can be applied. This implies that when all signatures of the departing key expire, one remaining active key must still be securely delegated.

V. Monitoring System and Data Corpus

In this section, we introduce the DNSSEC measurements taken from our monitoring system [24] that we used to evaluate key transitions in the wild. We give an overview of our monitoring system, describe its operational aspects, and discuss the characteristics of the resulting data corpus, which covers the first 15 years of the global DNSSEC rollout.

A. Monitoring System

Our monitoring system collects DNSSEC records (`DNSKEY`s, `DS`es, `RRSIG`s) alongside other types of RRsets, network PMTU measurements, name server versions, and many other relevant measurements by polling *all* of *every* zone name servers (as specified by both the zone and predecessor `NS` records) from distributed vantage points across the Internet [24], [25]. This comprehensive polling is a critical feature for observing key transitions with complex process models such as those specified in RFC 8901 [26].

In order to capture the holistic status of the global DNSSEC deployment, we broadly define zones as being *DNSSEC-enabled* if they have deployed one or more `DNSKEY` records. The set of DNSSEC zones in this corpus was learned from proactive crawling of online sources, `NSEC`-walking [25], [27], user-submissions, selected top-lists (Alexa Top One Million [28], the Majestic Million [29], and the Cisco Umbrella Popularity [30]), and other techniques. The development of the number of monitored zones throughout the years is depicted in Figure 7, and the detailed numbers are presented in Appendix A.

As our automatic discovery methods might catch non-production zones, i.e., secure zones which are deployed for testing purposes, we apply a conservative heuristic to keep only production zones. First, all zones under `arpa` TLD are marked as production, then any zone that points to an active Web or mail server is added to an include-list [10]. The remaining zones are considered as non-production. This way, we keep only zones for which we have positive indications of actually being in production. It should be noted that since there is no systematic mechanism to discover all zones in the global DNSSEC, measurements can be subject to sample bias; nonetheless, we argue that our measurements of over 9.5 million DNSSEC zones at the time of this writing represent a relevant set on which to measure general behaviors of key transitions in the wild.

Our monitoring system polls DNSSEC zones concurrently in four ways:

1) The monitoring system repeatedly queries all zones from each of its remote polling locations. These full measurements poll hundreds of zones concurrently, and when finished, the system starts again. As the number of DNSSEC zones has increased, the full-corpus polling period has grown from taking days to taking weeks, making the periodicity of polling a varying interval.
2) In parallel, it polls the DNS Root zone and all Top-Level Domains (TLDs) once per day, which is half the TTL period of those zones.
3) In addition, it polls popular DNSSEC-enabled domains from our top-list collection every two days.
4) Finally, the interactive Website of the monitoring system allows users to poll any zone on-demand.

Noteworthy, though, is that zones that delegate to others are implicitly queried and measured multiple times beyond individual schedule. For example, the `.com` zone is not only monitored once but its keys and their *usages* are also re-observed for every delegated domain below, in order to assess `NS` and `DS` records. For example, the root zone is measured over 1,300 times every day, once for every Top-Level Domain polled. This greatly increases the frequency of observation, albeit in an aperiodic way.

Our monitoring system has occasionally undergone migration to new hosting infrastructures, database backups, and

other operational maintenance events during the 15 years, leaving irregular gaps in our data corpus. We control for gaps in our methodology, for details see Section VI.

B. Data Corpus

This work makes use of data from October 1, 2005 to August 31, 2020. The resulting data corpus encompasses over 30.8 billion DNSSEC measurements from 9,535,615 DNSSEC-enabled zones with 35,882,395 distinct DNSKEYs. We observed 58,193,197 points in time when keys were either added or removed from zones. Of those changes, 17,965,575 key transitions were detected (see Section III-A) and analyzed, which we will discuss in Section VIII.

Our data corpus spans several events that are notable for the global deployment of DNSSEC. First, multiple Top-Level Domains (TLDs) such as .com, .edu, and 145 country-code TLDs (ccTLDs) deployed DNSSEC for the first time. Second, the announcement of a crucial large-scale DNS cache poisoning attack vector called the Kaminsky Attack [31], [32], whose remediation was publicized to be the deployment of DNSSEC. Third, the DNS Root zone enabled and rolled out DNSSEC for the first time using a Deliberately Unvalidatable Root Zone (DURZ) key. Fourth, in 2010, the DNS Root zone performed its first ever Key Signing Key (KSK) transition, from the DURZ key to the 2010 KSK. Fifth, over 1,200 DNSSEC-enabled new generic Top-Level Domains were added since 2013. Finally, the Root KSK was transitioned for the second time ever in 2018, after being started and paused during 2017. Figure 1 depicts a number of these notable events and incidents and by overlaying the measurement periods of related work and this study, shows this work has a uniquely complete longitudinal dataset to draw conclusions from.

VI. Methodology

The initial step in our methodology is to reconstruct the lifetime of DNSKEYs according to our discrete measurements and in accordance with our proposed temporal model (see Section III-A). This is a deceptively challenging step because when keys are provisioned into zones there are no semantics to express (or for zone administrators to even know) life cycle information. In general, key lifetime management and changes such as re-signings or deployment of new keys may occur at varying times between polling cycles. This will lead to gaps in time between three events: (*i*) when these changes happen on the authoritative name servers, (*ii*) when we poll the zones, and (*iii*) when we observe them in use. This may lead to changes that a measurement system completely misses regardless of the polling frequency. A simple example is the replacement of a DNSKEY multiple times between two polls.

We now give a brief overview of our methodology and then present the key building blocks in more detail.

A. Overview

Our first step in reconstructing the complete and continuous lifetime of keys from our discrete measurements is to infer a so-called *observable* for any unique <DNSKEY, RRSIG> tuple and any respective secure delegation period (if applicable) while preserving points of time when this specific observable was seen during our measurements. If the lifetime of a key is extended by re-signing (i.e., generating a new RRSIG over the same DNSKEY), our methodology creates a new observable tuple. In this sense, an observable can be considered as a piece of evidence based on a given RRSIG denoting that a key has been valid from the inception to expiration dates of that RRSIG. Figure 8 depicts *one* observable for key k_x. The lifetime of this single observable duration $[L_\alpha, L_\omega]$ is calculated from the signature covering k_x; its delegation period $[DS_\alpha, DS_\omega]$ is inferred from the signature in the parent zone covering DS record(s). *fs* and *ls* denote the times when this single observable was first and last seen during the measurements. Note that *fs* is recorded once per observable, and never changed again, while *ls* is updated when the same <DNSKEY, RRSIG> tuple is encountered again in a subsequent measurement.

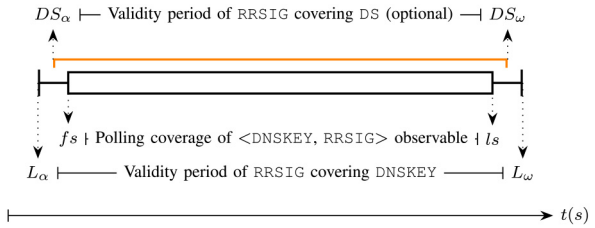


Fig. 8. Visualization of an observable for a given <DNSKEY,RRSIG> tuple.

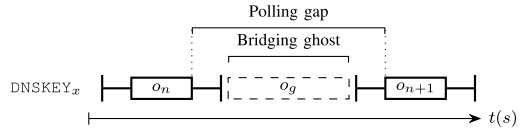
Individual observables expand the information about a key lifetime from a discrete snapshot to a continuous timeline within the validity period of its covering RRSIG. Such extrapolation of observables, however, might still leave some *gaps* in our continuous lifetime model. Furthermore, other life cycle information, such as key usage (see Figure 3), cannot be inferred from RRSIGs over keys but must be measured from signatures *generated* by those keys when they are in use. To address this we introduce a novel three-step methodology that we call *Bridging, Busting,* and *Binding*:

- **Bridging** Extend observables by filling in measurement gaps with place-holder observables, which we call bridging "ghosts".
- **Busting** Use collected evidence, e.g., RRSIGs over non-DNSKEY records, to remove (or "bust") incorrect ghosts.
- **Binding** For any given DNSKEY combine remaining contiguous observables into a continuous holistic life cycle model.

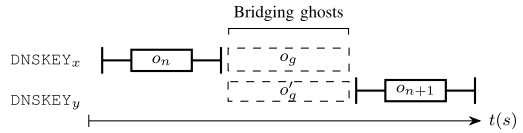
This process extends sets of observables into full key life cycles and builds a basis to calculate related statistics such as signing frequency, measure management errors, and compute other aggregate behaviors. It also accounts for gaps in our data corpus.

B. Bridging

Bridging begins by time-ordering observables for each zone. If the maximum time of a single observable (i.e., $max(L_\omega, ls)$) is less than the minimum time of the next observable (i.e.,



(a) Bridging between observables of the same DNSKEY



(b) Bridging between observables of different DNSKEY

Fig. 9. Filling in observation gaps with bridging ghosts.

$min(L_\alpha, fs)$, then a *ghost record* is inserted, which exactly covers the missing time. Note that a ghost does not necessarily fill the whole gap between two polling cycles, but only the time for which no temporal information can be inferred (see Figure 9(a)). If the ghost observable covers a gap between the observables of the same DNSKEY (but different RRSIGs), then the ghost observable proposes that the key existed continuously during the gap, as shown in Figure 9(a). If, on the other hand, the gap exists between observables of two different DNSKEYs, then the process cannot know precisely when the older DNSKEY$_x$ stopped being present and the newer DNSKEY$_y$ was deployed, if or when they may have overlapped, or if there was a period of no key(s) being present. Therefore, at this stage, a bridge is *temporarily* used, whereby a trailing ghost for DNSKEY$_x$ is inserted until both the next observable and a leading (overlapping) ghost for DNSKEY$_y$ are inserted, starting just after the previous observable, as seen in Figure 9(b). Ghosts represent optimistic assumptions about consistency between observations, but in the next phase we *bust* ghosts if additional evidence proves they are incorrect or need to be adjusted.

C. Busting

Ghost observations model place-holders of inferred data that may have existed between the data we observed. For each zone, after the initial optimistic Bridging phase, our process begins to examine keys in relation to each other and incorporating additional evidence to detect if a ghost assumption can be refuted (and thereby *busted*). For this, additional measured data (such as NS, SOA, A, and associated RRSIGs) allow the process to determine if, or how, a ghost should be busted. For example, if a ghost bridges a key, but another key was seen during that time, we can determine the ghost-key was not present for the period when the other key was seen. The ghost is, then, busted by truncating it to the time interval(s) that are not covered by the other key. Alternately, if ghosts for two keys overlap between a transition (see Figure 9(b)), information about which of them was able to verify signatures over other measured data is used to determine when one key was removed and the other was present, and truncate ghost overlaps accordingly. Overall, ghosts may be truncated if a zone's data was observed and the key was absent, and affirmed if a ghost's key was seen to be signing other records. Ghosts

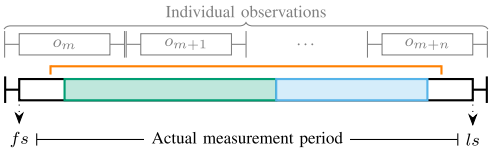


Fig. 10. Observations are bound together to find minimum and maximum timestamps for inception, expiration, first seen, and last seen. Measuring bounds also seek evidence of when keys were used to generate signatures (green region) and how long their signatures were valid for (blue region).

which may have been causes by measurement outages are removed altogether. Gaps caused by outages are distinguished through their relatively long duration. For our purposes, we take the yearly statistics of our measurement system crawl times, i.e., a complete round of polling all zones, to bust gaps caused by outages: for any given year all gaps that are larger than the mean crawl time plus four standard deviations are considered as an *outage gap*, are busted and removed. While the removal of long-ghosts could result in missed key transitions and key management behaviors, not removing them could alternately enshrine inaccurate assumptions.

D. Binding

For every key, all the contiguous observations (including unbusted ghosts) are used to *Bind* observables into one single continuous key, as seen in Figure 10. Bound keys describe the life cycle of a DNS key in terms of our temporal model (Figure 3) with two additional features. The first timestamp that the key was measured (*fs*) and the last time a key was observed (*ls*). Here, all the observed RRSIG records that could be verified by each key, regardless of which DNS record type they covered, are used to quantify whether keys were in active use. The inception timestamp of each RRSIG is used as evidence to indicate when a key pair was signing data ($[S\alpha, S\phi]$). The expiration of that signature specifies the run-out, or the duration that a key's data was verifiable while no new signatures were detected ($[S\phi, S\omega]$).

VII. Classifying Keys and Transitions

Using our model of continuous life cycles for DNSKEYs, we move on to classification of DNSKEY life cycle states and key rollovers and more complex key transition types.

A. Key Life Cycle Classification

We present our novel classification scheme of measurable errors in key life cycle management in Figure 11. Based on this model we observe previously undetected errors in live deployments, which we explain in more detail in Section VIII.

This classification scheme defines six types of key life cycle management errors depending on when a key is introduced in the zone and how inception and expiration dates of the respective RRSIG are defined. For example, if the expiration predates the inception date we classify the key as *inverted*, or if key was observed before its actual inception date, we label it as a *future* key. Avoiding these errors is a necessary (yet insufficient) precondition for *valid* cryptographic protections.

TABLE II
MAPPING OF KEY TRANSITION PROCESSES SPECIFIED IN DNSSEC RFCS AND OTHER LITERATURE TO TRANSITION ANATOMIES. IN EACH ROW, GRAY CELLS SHOW THE MANDATORY FEATURES THAT MUST BE FULFILLED TO MAP A TRANSITION TO A PREDEFINED CLASS, WHEREAS THE OTHER CELLS DESCRIBE SOFT CONSTRAINTS, WHICH ONLY CAUSE WARNINGS IF NOT FOLLOWED EXACTLY. CELLS SHOWING "–" INDICATE WILDCARDS

		Features of the Transition Anatomy									
	Transition Class	*PreDS*	*DoubleSig*	*PreStage*	*DepSigOnly*	*Retire*	*DSOverlap*	*RemSigOnly*	*DSPreRem*	*RemPreDS*	*TotalDuration*
Based on RFCs	*ZSK Pre-Pub*	–	$= 0$	> 0	> 0	> 0	–	> 0	–	–	–
	ZSK Double-Sig	–	> 0	$= 0$	$= 0$	$= 0$	–	$= 0$	–	–	–
	KSK Double-DS	< 0	$= 0$	$= 0$	$= 0$	$= 0$	> 0	> 0	< 0	< 0	–
	KSK Double-KSK	> 0	> 0	$= 0$	$= 0$	> 0	$= 0$	> 0	> 0	> 0	–
	KSK Double-RRset	> 0	> 0	$= 0$	$= 0$	> 0	$= 0$	> 0	$\neq 0$	–	–
	KSK Update-TA	–	–	–	–	–	–	–	–	–	$\geq 30d + AvgSig$
Based on [17]	*KSK Emergency 2*	–	$= 0$	$= 0$	$= 0$	$= 0$	$= 0$	> 0	–	≥ 0	–
	KSK Emergency 3	–	$= 0$	> 0	–	> 0	–	–	> 0	> 0	extended
	ZSK Emergency 2	–	$= 0$	$= 0$	$= 0$	$= 0$	–	> 0	–	–	–
	ZSK Emergency 3	–	–	> 0	–	–	–	–	–	–	extended

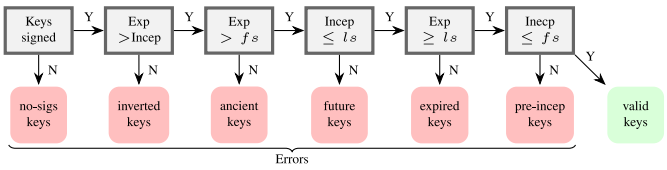


Fig. 11. Classification of key life cycle management states. Each state results from an observable property (rectangle) that is either fulfilled (Y) or not fulfilled (N). Properties include expiration (Exp) and inception (Incep) dates, as well as first (*fs*) and last seen (*ls*) of operational usage.

B. RFC-Based Classification

The IETF has specified many aspects of how DNSSEC zones and data should be configured and maintained across numerous RFCs [6], [7], [23], [33]. Among those are several processes that model ways in which DNSKEYs should be rolled over. In RFC 5011 [6], Trust Anchor (TA) rollover is specified for zones whose predecessor zones do not securely delegate DS records. Additionally, in RFC 7583 [7], processes are described for how zone administrators should transition their ZSKs and KSKs.

First we characterize RFC guidelines in terms of our transition model and then investigate how transitions in the wild conform to these guidelines (in Section VIII). We classify deviations as either warnings, i.e., the behavior does not strictly follow the RFC guidelines, yet, does not disturb validation; or as errors, which render the validation as bogus or insecure. It should be noted that while conventional IETF parlance (e.g., MUST vs. SHOULD in RFC 8174 [34]) often makes this distinction explicit, the measurable quantities in RFC-7583 [7] do not use MUST, MAY, SHOULD, etc. We, therefore, semantically assign these values based on the overall processes. For example, if a transition was specified as needing to have *Retire* > 0, but it was 0, this would only be a warning because the key transition would still allow keys to verify data for a zone. In contrast to this, if a KSK must be present *before* a DS record in order to let resolvers securely verify the KSK during a *Double-DS* transition, then *PreDS* ≤ 0 is a critical error.

RFC 5011 (Update of Trust Anchors): This transition is specified by a Finite State Machine with timers and a rigorous process model [6]. The process-model, however, is specified from a resolver's perspective (i.e., timers that a resolver should set internally), which do not always directly correlate to observable timing of DNSKEYs in authoritative zones. Additional guidance [35] was written for authoritative zone administrators, which lends itself more directly to being measured. These two publications [6], [35] define the specification for RFC 5011 transitions as those whose *TotalDuration* $\geq 30days + min(15d, \frac{1}{2} \times TTL)$, where 30 days are specified in [35] and the key's Time-To-Live (TTL) is specified as an additional component of the period. Additionally, those keys that are being removed must be revoked *and* be used to sign their own revoked DNSKEY set. In order to be conservative and permissive, we model the upper-bound (i.e., *max* instead of *min*) from each key's own average signature period. We conservatively modeled those zones that did adhere to RFC 5011 timing, but *did not* revoke as still being RFC 5011 compliant, but flagged them with a warning.

RFC 7583 (DNSSEC Key Rollover Timings): RFC 7583 specifies the process models for both ZSKs and KSKs, and they are more stringent. For ZSK, these transitions are defined as either being "Pre-Publication" or "Double-Signature"; and for KSK as one of "Double-DS," "Double-KSK," "Double-RRset," or "Double-RRSIG". Note that the final type is identified as unrealistic in the RFC, and not even fully described there; therefore, we also omit it here.

Table II summarizes how RFC-based classifications are translated into our transition anatomy from Section III. An illustrative example is provided in Figure 12, depicting an excerpt of transitions for a representative zone, 4d.cz.. As is shown, the zone performs a correct ZSK pre-publication transition while (at other times) performing double-KSK transitions *with warnings* (see Table II). Other examples of key transitions are visualized in Appendix C.

C. Non-IETF Prescriptions: Emergency Key Transitions

In addition to the above guidance from RFCs, other prior work by Wang and Xiao [17] proposed an approach for

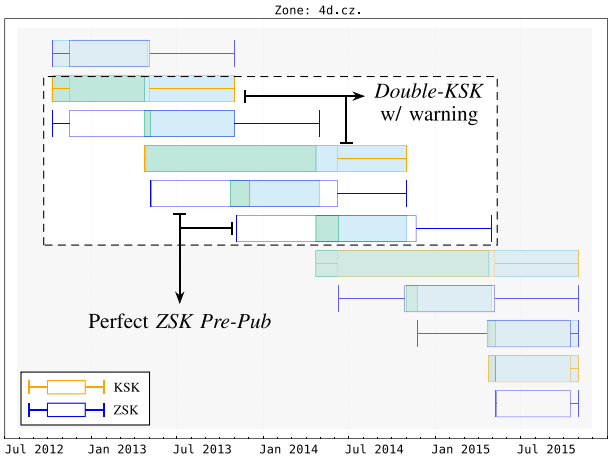


Fig. 12. An example from our measurements showing how `4d.cz.` zone performs a perfect *ZSK Pre-Pub* transition and a *Double-KSK* transition with warnings (*Double-Sig* $\not>$ 0 and *Retire* $\neq$ 0).

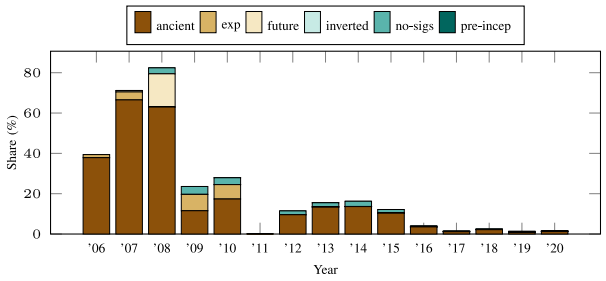


Fig. 13. Classified key management errors.

conducting emergency key transitions. There, the authors present 10 candidate process prescriptions for emergency key transitions, which are distinct from conventional RFC guidance. It is noteworthy that this prior work considers keys transitions to stand-by keys which are envisioned to be perpetually present before the emergencies. This, therefore, does not precisely specify the *TotalDuration* threshold, but suffices to make the "extended" duration feature a necessary detectable discriminator in these emergency transitions classification. Table II shows how the timing constraints detailed in that work can be used to classify and detect these events specified by our anatomy and transition methodology, just as with RFC guidance. Due to space limitations, we only include Emergency Transitions 2 and 3.

Additionally, aspects of that work use features of our anatomy and our model of key life cycles that the RFC processes did not (specifically, the TotalDuration and relative ages). For example, the specification requires remaining keys to be *newer* (see *relative ages* in Section III-A), *ZSK Emergency 2* mandates the feature-set of the behavior-based classification *Multi-Signatures* (described below, in Section VII-D), and *ZSK Emergency 4* requires *Cutover*s.

In summary, using the prescriptions of prior work, we detected 49,894 emergency ZSK transitions (20,919 that were transitioning *back* or *aborting* transitions to older or same age keys), and 1,780,984 emergency KSK transitions (149,406 transitioning *back* or *aborting* to older or same age keys).

D. Behavior-Based Classification

In our behavior-based classification, we classify key transitions as being "Multi-Signatures," "Co-Present," "Cutovers" (with degrees of certainty), or "Unknown". This classification approach uses more holistic considerations of all keys in a transition (not just pair-wise) while ignoring the relationships to `DS` records. Here, key transitions were classified based on (*i*) the type and the total count of overlaps they had with *all* other *active* keys, and (*ii*) whether they were used to verify zone data during their transitions.

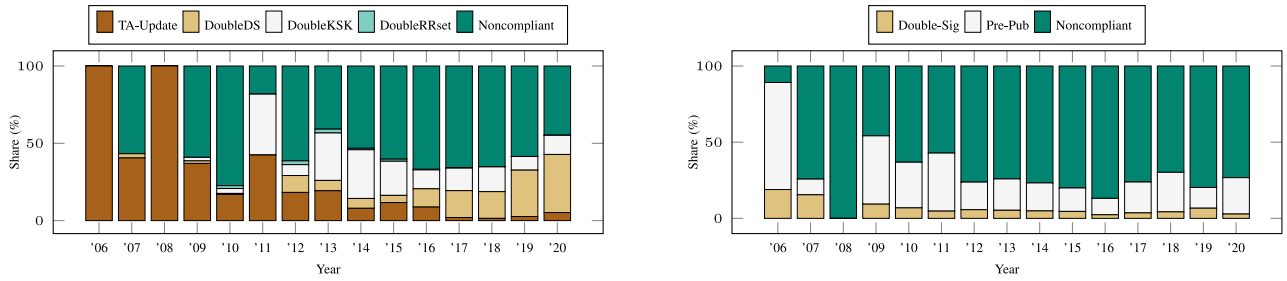
When more than one key was seen to be actively in-use (verifying signatures) at any given time, we classified the transition as a "*Multi-Signature*" transition. This indicates that redundant verification existed for data in a zone during the transition. If, on the other hand, keys were observed to have overlapped, but we did not observe *any* of them in use, we classify the transition keys as being "*Co-Present*." This classification does not represent evidence that keys went unused, but indicates that our observations did not detect usage. Conversely, transitions are classified as "Cutovers" when a single key was observed to transition (or cut-over) to another single key, and they were seen to have been used to verify signatures. This type of transition depends heavily on measurement frequency (whereby the longer the gaps in observations, the more information is estimated from ghost records). Because of this, we further sub-classify Cutovers as "Cutover," "Likely-Cutover," or "Candidate-Cutover." The differences between these are based solely on how much usage (i.e., active signing) was directly observed. If active signing (the inception of `RRSIG`s) was observed for the departing key right up until the remaining key began to be used to verify `RRSIG`s (and not after), then we classify this as a "*Cutover*." If, on the other hand, we did not observe new signatures, but the signature run-out (see Figure 10) of the departing key overlapped with the signatures of the remaining key, we classify the transition as a "*Likely-Cutover*". Finally, if the departing key was used at any point, and later the remaining key began being used with no other signatures seen in the time-gap, we classify this as a "*Candidate-Cutover*." The goal of these distinctions is to make our sub-classification results useful, but to also preserve their transparency.

VIII. Longitudinal Analysis of 15 Years of DNSSEC Key Transitions

In this section, we present the results from answering the following three questions: (*i*) Do keys involved in transitions follow proper life cycle management policies? (*ii*) How does the anatomy of key transitions in practice compare with RFC guidelines specifying rollovers? (*iii*) Which characteristics are popular in key transitions beyond key rollovers in terms of our behavior-based classification?

A. Key Life Cycle Management

The atomic management of the keys can end in errors, independent of transitions of other keys. Using the key life cycle



(a) Key Signing Keys (KSK)

(b) Zone Signing Keys (ZSK)

Fig. 14. RFC-based classification of key transitions for in-use KSK and ZSK.

management classification described in Section VI, we broadly examine the rates at which individual DNSKEYs follow proper key management life cycles. Our results are summarized in Figure 13. Key life cycle management errors were clearly highest during the early years of the DNSSEC global rollout. At that time, the tools that supported DNSSEC were in their nascent stages of maturity, which is very likely the core reason for larger rates of key life cycle management errors. This figure also illustrates that the error rates for keys were highest in 2008, just as the discovery rate of new DNSSEC zones surged and more than doubled the size of the global deployment in just a few months. This all correlated in time with publicity to remediate the newly announced Kaminsky vulnerability [32] by deploying DNSSEC, which could have also correlated with inexperienced operators making operational errors in a rush to secure their deployments.

B. Conformance to RFC Guidelines

KSK and ZSK Transitions in the Wild: Figure 14 summarizes RFC-based classifications of both measured KSKs and ZSKs per year. Any transition that could not be classified as RFC-compliant is marked as *noncompliant*. This figure confirms a common expectation but also reveals new insights. Unsurprisingly, in the early DNSSEC deployment KSKs were transitioned according to RFC 5011 (denoted here as "TA-Update"). This was necessarily the case because at that time there were very few registrars and also few parent zones that were able to offer secure delegations. Most Top-Level Domains did *not* deploy DNSSEC and were, therefore, unable to securely delegate.

In 2008, operational advice was given to deploy DNSSEC to counter the Kaminsky vulnerability [32]. Our data (see Table A.1 and B.3) indicate that the number of DNSSEC-enabled zones more than doubled during three months. Based on our classification, it is clearly visible that subsequently more variety in key transition techniques appeared (see Figure 13(a)). In 2009, the Double-DS transition technique was the most popular. This technique, however, requires additional coordination between an authoritative zone and the operator of its parent zone. Despite an increasing deployment over time, our results illustrate that the popularity of managing transitions based on Double-DS and Double-KSK fluctuates. The majority always conformed to either RFC 5011 or was

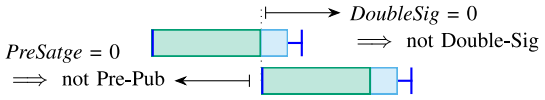


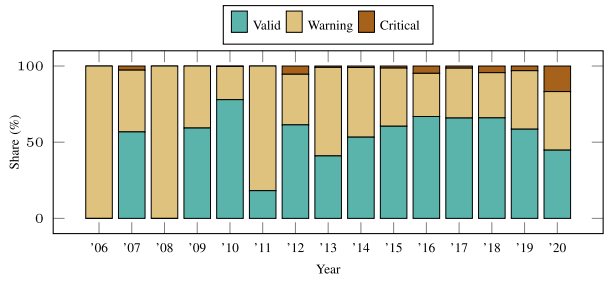
Fig. 15. An example of RFC noncompliant transition performed among others by Root and .com zones.

"noncompliant." This observation indicates that the RFC-specified key transition processes may not properly represent operational behavior.

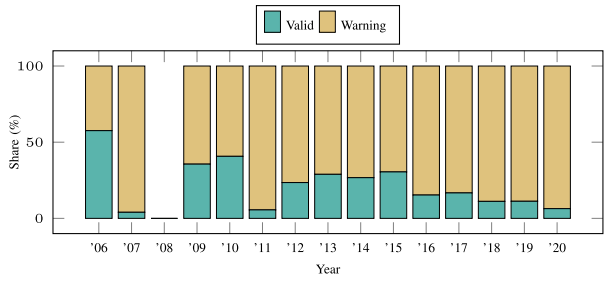
Our results also exhibit an interesting discrepancy compared to related work. In the longitudinal study of Chung *et al.* [13], no Double-DS transitions were reported. However, under .com alone, we observed 17,126 unique zones performing Double-DS transitions during the time period of their daily scans (from 2015-03-01 to 2016-12-31), and 7,256 transitions during the same period as their hourly scans (from 2016-09-29 to 2016-12-31). A concrete example of Double-DS transitions is willemvanveldhuizen.com (see Figure C.3(e)). Further understanding the discrepancies of these findings will be part of our future work.

In comparison to KSK transitions, an even larger portion of ZSK transitions have constantly been nonconforming to RFC specifications (see Figure 14(b)). While the "Double-Signature" alternative never accounted for more than 10% of transitions, it is most noteworthy that between 2007 and 2020 (except for 2009) the majority of the observed ZSK transitions did *not* conform to either prescribed mode of key transition. Ignoring RFC guidelines, however, does not necessarily mean that the zone verification would fail during the transition. Figure 15 depicts an example of such transition performed by prominent zones such as the Root and .com zones. This transition is neither Pre-Pub ($PreStage \not> 0$), nor Double-Sig (DoubleSig $\not> 0$), yet at any time during the transition the RRs can be verified by either or both keys. Here, we also see discrepancies with prior work [13], which might be traced back to different approaches used in classifying transitions.

Implications on Robustness: The warning and error rates for RFC-compliant transitions are notably different for KSKs and ZSKs, as depicted in Figure 16. KSK transitions show more valid cases than ZSK transitions overall. This corresponds also with a higher share of transitions following RFC guidelines (see Figure 14(a)). Later, between 2011 and 2020, critical error rates became even more prominent. This could be the

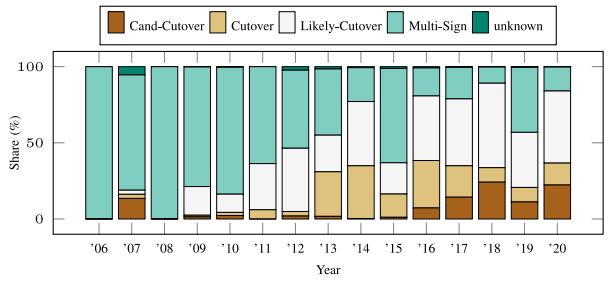


(a) Key Signing Keys (KSK)

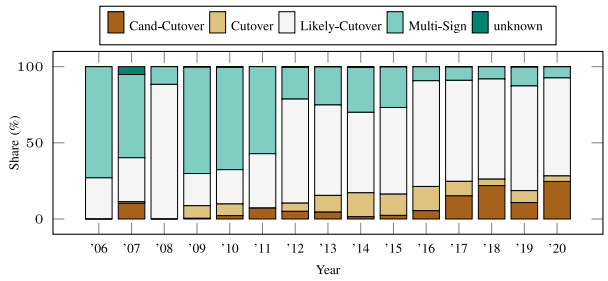


(b) Zone Signing Keys (ZSK, excluding type noncompliant)

Fig. 16. Share of valid, warning, and error rates of in-use KSK and ZSK.



(a) Key Signing Keys (KSK)



(b) Zone Signing Keys (ZSK)

Fig. 17. Behavior-based classification of key transitions for in-use KSK and ZSK.

result of inherent management complexity in terms of timing and data for (longer) chain-of-trust-based transitions between authoritative zones (DNSKEYs) and parent zones (DS records).

For ZSKs, Figure 16(b) shows that the rate of warnings exceeded successes in every year, except in 2006, when the DNSSEC deployment was just beginning, and zones were deploying DNSSEC for the first time. In 2007, warning rates greatly exceeded success. In 2008, we observed only noncompliant transitions (see Table B.3), and from that year on the trend was increasing for success rates. In 2011, an outage in our monitoring system resulted in fewer observed transitions, and the success rates of that smaller set may have been skewed. From 2012, after a brief improvement, we also observe decreasing success rates. This, however, again correlated with a large increase in the discovery rate of new DNSSEC-enabled zones. During this time, the DNS Root was deploying DNSSEC and transitioning its KSK for the first time. This correlation between a large increase in newly deployed DNSSEC zones and increase in error rates suggest that these rates may also be due to operators who were learning how to manage the security postures of DNSSEC in their DNS zones.

C. Holistic Characteristics of Transitions

Using the RFCs to classify key rollovers gives us a helpful start in understanding how to analyze key transitions. However, to overcome the amount of noncompliant and high error rates, we use the alternate behavior-based classification scheme introduced in Section VI.

When applying our behavior-based classification to ZSKs that were seen to be in use, Figure 17(b) shows that almost every key transition observed for 15 years could be classified. Only in 2007, 4.32% transitions were classified as unknown; in all other years, >99% of transitions were classified successfully. The trend over the first 15 years was a shift from Multi-Signature, to cleanly cutting over from a departing key to a remaining key.

Figure 17(a) illustrates that all of these trends were similar in the KSK transitions. A greater portion of the KSK transitions were Multi-Signature than in ZSKs, but the trend towards moving to Cutover (and the small incidence of unknown transitions) mimics the ZSK transitions. This could be due to the same operational approach being used for both ZSKs and KSKs by zone operators, but the DNSSEC tools available have traditionally offered more comprehensive automation for ZSK transitions, since they are managed within a single authoritative zone. By contrast, KSK transitions involve coordination with the parent zone.

IX. Related Work

The topic of DNSSEC key rollovers has recently appeared in the literature. In 2017, the DNS Root zone began publicity around its, then, intentions to transition its KSK for the second time (since DURZ, as discussed in Section V). At that time, van Rijswijk-Deij *et al.* [15] began the Root Canary project to track this specific transition. Later, Müller *et al.* [14] created a related monitoring tool to aid operators in successfully planning and conducting DNSKEY rollovers. More recently and maybe most closely related to our work, Müller *et al.* [16] conducted an in-depth study of key life cycle management in the DNS Root zone. Both of these results synergize with the observations of our work: key life cycle management of

DNSSEC zones is critical to their operational health. Most importantly, and in contrast to prior work, we do not only consider key rollovers but provide a framework to analyze key transitions in general.

As DNSSEC deployment has grown, numerous studies have incrementally tracked its deployment. Among the first measurement studies, Osterweil *et al.* [10] observed that longitudinal measurement of DNSSEC deployment were critical to understand its health, and proposed a set of three metrics to summarize the status of the global deployment. Subsequent large-scale measurements [12] reported to confirm earlier findings, but also began to raise concerns about validating resolvers and the limited number of users being protected by DNSSEC.

More recent work [13] has conducted large-scale longitudinal analysis of the global DNSSEC deployment. Therein, the authors examined two years of DNSSEC deployment data and addressed the identification of key management errors as an area for concern. The authors also discussed the subject of recycled keys as key "sharing" and flagged the behavior as an error. Recycled keys are those that were shared in separate zones or used, removed, and then re-used. DNSSEC zones are often presumed to create distinct keys for themselves without sharing usage with other zones, and that once a key expires and completes its operational lifetime, it will not be used again. In our paper, we treated each appearance of a recycled key as a new key, because when being re-used its operational life cycle is different. From our corpus we identify 35,882,395 distinct `DNSKEY`s, 42,908,290 distinct `DNSKEY`/zone pairs, and 54,337,697 total operational lifetimes of keys. Our measurements confirm the earlier result, and further illustrate that a non-trivial number of `DNSKEY` records were shared between zones, and others may have been returned to service after completing an initial operational lifetime.

While recent related work signals a renewed interest in key transitions, previous literature exists that suggests augmenting the DNSSEC protocol to add explicit semantics that indicate ongoing key transitions. Guette *et al.* [36] suggested an extension to the `DNSKEY` format itself to indicate when key transitions are underway. In subsequent work [37], this approach was evolved by proposing the new Resource Record `KRI`. Interestingly, the semantics that those works suggest as being necessary are already observable in the current DNSSEC, when using the methodology we introduced in this paper. Explicitly exposing key transitions in the DNSSEC may have security implications, though. Nawrocki *et al.* [38] show that the presence of multiple keys in the DNS gets systematically exploited in DDoS attacks.

The DNSSEC algorithm rollovers, an aspect that is outside the scope of this work, has been studied in the past [13], [14]. In a recent publication, Müller *et al.* [39] study the lifetime of cryptographic algorithms for DNSSEC to conclude that *algorithm agility* has already been reached for DNSSEC.

Finally, the role of machine learning in assessing security aspects and detecting various attacks, which has been discussed in the related work, can also be useful for our future work specifically with respect to noncompliant transitions. Jin *et al.* [40], for example, make use of machine learning to detect cache poisoning in DNS, specifically those caused by compromised name servers. In the context of Web PKI, Dong *et al.* [41] define a set of features to describe X.509 and apply deep neural networks to detect rogue certificates. And finally, in general context of relational data, Lou *et al.* [42] proposes a method to cluster related data.

X. Discussion

The global rollout of DNSSEC is flourishing, and is giving operators experience at scale in managing cryptography in a core Internet protocol. We believe that now is the right time to investigate what has been (and can be) learned from these experiences.

Capturing the right security model: Caching plays an important role in DNS(SEC) because it enables scalability and availability. This service is controlled by `TTL` values of records. From a security perspective [43], when changing `DNSKEY`s, care must be taken to provision consistent `DNSKEY` and `RRSIG` material so that what can appear in caches remains verifiable at all times (see Section IV).

The DNSSEC *availability* protections are important but distinct from its data *integrity* assurances. Availability is entirely governed by `TTL`, whereas the integrity protection is entirely governed by `DNSKEY`s, `RRSIG`s, and `DS`es and their timings and cryptographic life-times. TTL-based availability protections (designed for caching) are not involved in integrity assurances. Conversely, integrity protections actually mollify cache poisoning attacks, which was a central design goal of DNSSEC [31]. In such attacks, availability is not a factor because adversaries influence the presence of data in caches. Though DNSSEC operations involve the confluence of these two aspects of DNS(SEC). In this work, we focused on the DNSSEC integrity protections and stress the importance of evaluating their exclusive role. Any changes to RRs can succeed within the temporal constraints imposed by DNSSEC (see Section IV), and in turn, how caches regard `TTL` values has no impact on security guarantees provided by DNSSEC. We proposed that the necessary conditions to preserve integrity protections during a DNSSEC key transition are:

1) Zone KSKs must be covered by valid and verifiable `DS` records to establish a chain of trust.
2) `RRSIG`s covering current data (`DSNKEY` and other RRSets) must be verifiable by at least one authorized `DNSKEY` at all times.

These conditions together are sufficient to provide data integrity assurance in the DNSSEC during key transitions. We used our proposed anatomy to describe how to evaluate these in operational deployments. We also discussed (Section IV) the security implications of key transitions both in terms of the relationship of a key to its own signatures, to other keys, and to delegation signers in case of KSK. Our proposed temporal constraints do not only allow for a formal security analysis of transitions but can also be used to design and validate software used by authoritative nameservers to manage key transitions.

Sampling frequency versus TTL-level polling: There is debate in the literature on the general topic of whether to monitor `DNSKEY` transitions at the granularity of DNS `TTL`s

TABLE A.1
TOTAL NUMBER OF MONITORED DNSSEC-ENABLED ZONES PER SOURCE

	Year															
	'06	'07	'08	'09	'10	'11	'12	'13	'14	'15	'16	'17	'18	'19	'20	
AXFR Scrape	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	10071	5087	39106	23001
CZDS Scrape	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	1486740	586367
DLV Crawl	NA	NA	NA	29882	316879	437260	NA	99910	418429	409863	354055	85947	39029	11005	61829	10211
NSEC Walk	NA	NA	8818	17432	39416	125015	NA	45052	187869	189780	165194	44299	19470	5690	32058	5252
Other	2	113	10847	20440	37176	443876	NA	522203	7699153	8078773	6984714	1748770	796517	252877	1685094	297176
P-DNS	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	9669908	9950352
Registry Scrape	NA	1	853	1374	1402	969	NA	176	566	552	1150145	9710245	10571470	3794568	16348389	2477047
User Submission	12	390	39589	77625	137853	139604	NA	354217	7108250	7709995	7344963	1811286	1138116	335147	1826450	286705

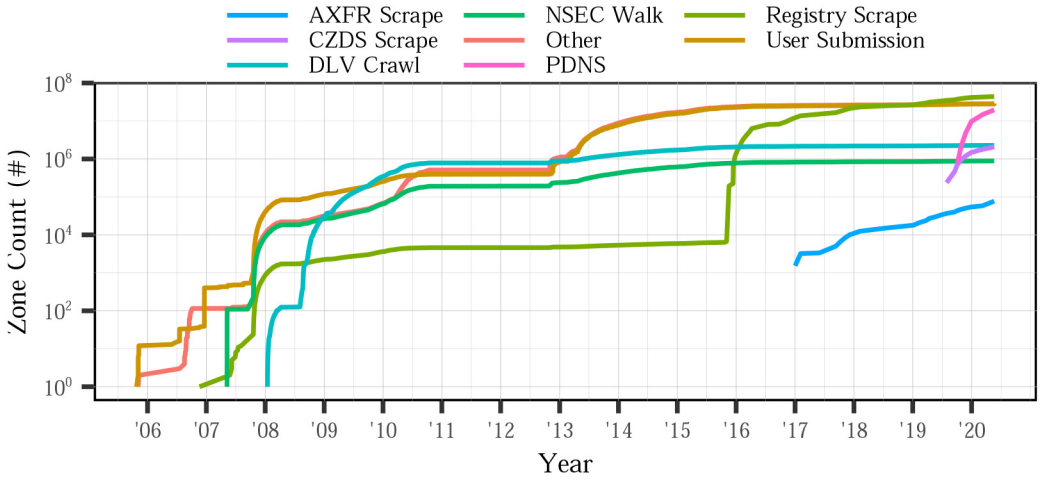


Fig. A.1. Total number of monitored DNSSEC-enabled zones per source and year.

TABLE B.2
TOTAL NUMBER OF KEY MANAGEMENT ERRORS

	Year														
	'06	'07	'08	'09	'10	'11	'12	'13	'14	'15	'16	'17	'18	'19	'20
ancient	175	4765	36176	1998	NA	41217	25409	167665	186119	158774	203820	69694	196446	59613	287679
exp	7	282	40	1399	NA	16929	12	1898	548	82	222	5355	6930	4937	20361
future	NA	52	9426	NA	NA	53	NA	495	232	103	124	52	628	143	325
inverted	NA	NA	NA	NA	NA	NA	NA	NA	28	22	10	4	NA	3	6
no-sigs	NA	NA	1703	679	NA	8018	5039	24889	35518	27482	27931	14958	25906	33372	55879
pre-incep	NA	NA	NA	NA	NA	122	NA	254	NA	NA	NA	NA	8	25	105
valid	280	2064	10098	13211	NA	170944	235059	1061506	1145816	1346260	5537326	5875182	8889718	7368282	23240768

or signature lifetimes. Specifically, work by Chung *et al.* [13] considers TTL to be the primary discriminator of change frequency. We argue that when conducting a key transition, changing zone contents (DNSKEYs or otherwise) before their definitive RRSIG-based lifetimes have elapsed (e.g., at the granularity of TTLs) exposes zones to vulnerabilities (e.g., replay attacks), as discussed in Section IV. This holds because the security guarantees of DNSSEC are orthogonal to TTLs and caching because the TTL values cannot ensure the absence of replayed (possibly compromised) keys into validating resolver caches. More specifically, an adversary can replay compromised keys whose RRSIGs are still valid, regardless of their TTL. For example, if a zone private key was compromised and the operator of the zone was to replace it with a new key (i.e., remove the old key immediately or when TTLs have expired) a zone would still be vulnerable to replay attacks. While this does not stop operators from performing key transitions like this, one of the objectives of our analyses was to demonstrate the relationships (and gaps) between protection and practices.

While our analyses illustrate weaknesses, our measurement corpus actually includes observations taken at half the TTL values of the DNS Root zone and all TLDs whose TTLs are all two days. We posit that one-day polling of these zones is sufficient based on the Nyquist-Shannon sampling theorem [44], [45]. Measuring continuous phenomena with discrete sampling can approximate those phenomena and accurately characterize them, if polling occurs at a frequency that is at least twice the rate of change. This substantiates our three conclusions. (*i*) Key transitions need to be measured at frequencies

TABLE B.3
RFC-BASED CLASSIFICATION: NUMBER OF DIFFERENT KEY TRANSITIONS FOR KSK (TOP) AND ZSK (BOTTOM)

	Year														
	'06	'07	'08	'09	'10	'11	'12	'13	'14	'15	'16	'17	'18	'19	'20
KSK Transitions															
RFC 5011	11	15	4	179	2202	14	759	5395	5101	7084	18061	5445	3905	16100	33845
DoubleDS	0	1	0	8	78	0	455	1862	4087	2847	24411	47035	46284	183326	243542
DoubleKSK	0	0	0	11	410	13	293	8565	20180	13424	25206	39552	42935	53756	80228
DoubleRRset	0	0	0	0	240	0	102	691	613	825	793	718	388	162	396
Noncompliant	0	21	0	286	10036	6	2554	11397	34103	36636	137853	178225	174423	357362	290231
ZSK Transitions															
DoubSig	7	15	0	250	1913	2	1274	17330	16555	20042	39605	79402	95631	232452	116298
PrePub	26	10	0	1202	8326	16	4068	67146	62260	67601	181305	451082	583795	474944	980721
Noncompliant	4	72	43	1229	17555	24	17110	241960	260294	352031	1462711	1693579	1568447	2781118	3012187

TABLE B.4
NUMBER OF VALIDS, WARNINGS, AND ERRORS FOR KSK (TOP) AND ZSK (BOTTOM; EXCLUDING TYPE NONCOMPLIANT)

	Year														
	'06	'07	'08	'09	'10	'11	'12	'13	'14	'15	'16	'17	'18	'19	'20
KSK Transitions															
Critical	0	1	0	0	30	0	225	245	620	820	9892	3781	11700	18749	108796
Valid	0	21	0	287	10103	6	2555	11442	34160	36777	137956	178466	176729	357478	290280
Warning	11	15	4	197	2833	27	1383	16223	29304	23219	58476	88728	79506	234479	249166
ZSK Transitions															
Valid	19	1	0	517	4170	1	1255	24428	21048	26721	33833	88625	75826	79421	69998
Warning	14	24	0	935	6069	17	4087	60048	57767	60922	187077	441859	603600	627975	1027021

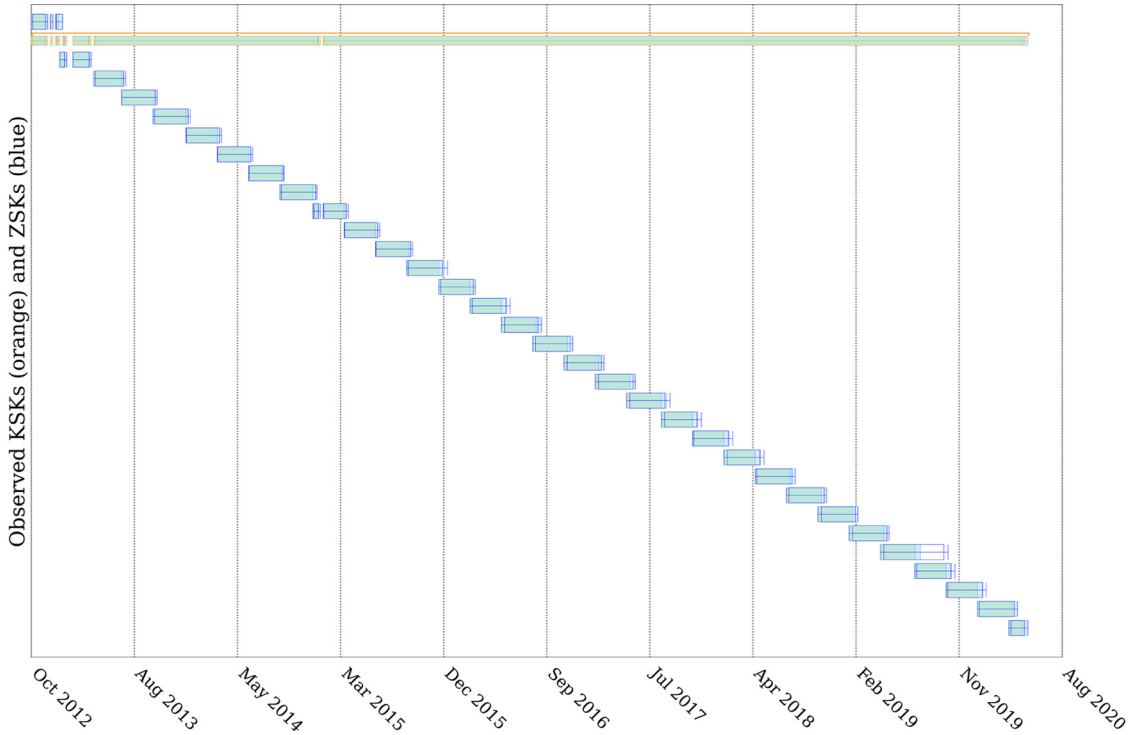
TABLE B.5
BEHAVIOR-BASED CLASSIFICATION: NUMBER OF DIFFERENT KEY TRANSITIONS FOR KSK (TOP) AND ZSK (BOTTOM)

	Year														
	'06	'07	'08	'09	'10	'11	'12	'13	'14	'15	'16	'17	'18	'19	'20
KSK Transitions															
Cand-Cutover	0	5	0	7	296	0	81	486	195	683	14957	38777	64779	68512	144642
Cutover	0	1	0	5	267	2	124	8162	22182	9288	64264	55846	25438	57983	93264
Likely-Cutover	0	1	0	91	1551	10	1729	6712	27041	12475	87465	119004	148585	220946	306866
Multi-Sign	11	28	4	380	10804	21	2133	12123	14230	37662	37929	56173	28458	260701	101843
unknown	0	2	0	1	48	0	96	427	436	708	1709	1175	675	2564	1627
ZSK Transitions															
Cand-Cutover	0	10	0	12	612	3	1127	14709	5063	10416	91476	337814	492256	375772	1012625
Cutover	0	1	0	222	2131	0	1209	35893	53307	61372	268118	211801	95878	274009	149507
Likely-Cutover	10	28	38	566	6256	15	15341	193788	179113	250088	1168321	1474685	1478350	2396792	2643776
Multi-Sign	27	53	5	1872	18677	24	4679	81445	100092	116860	154093	194519	179959	431296	299926
unknown	0	5	0	9	118	0	96	601	1534	938	1613	5244	1430	10645	3372

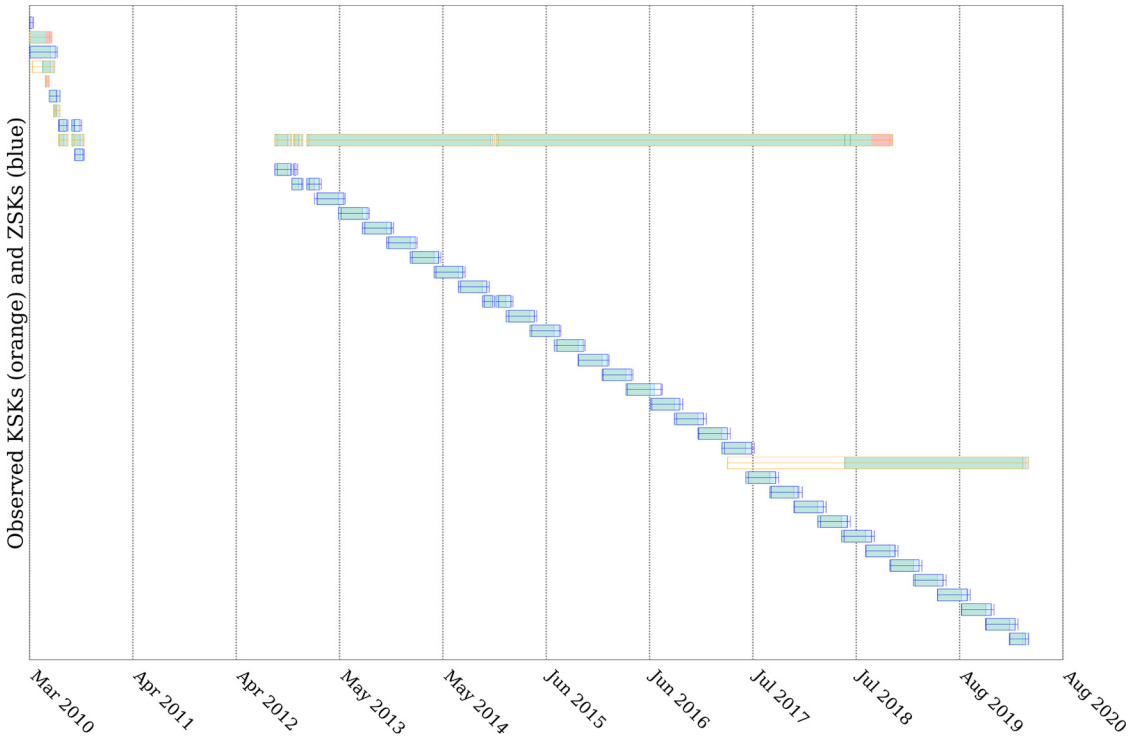
relative to RRSIG lifetimes. (*ii*) Properly operated infrastructures (e.g., the Root and TLDs) perform their operations in accordance with these analyses. (*iii*) This work is able to compare these different timing hypotheses (i.e., RRSIG vs. TTL) using longitudinal measurements from the wild.

Finally, it should be noted that discrete measurements, regardless of the measurement frequency, are going to inevitably miss alterations that happen between subsequent measurements. With regard to our measurements, this means that violations to temporal constraints (Section IV) might be overseen as we construct a continuous model using the binding, bridging, and busting method (Section VI).

Key transitions are complex: The anatomy of key transitions in the wild shows a large diversity in the different ways that zone administrators are deploying them. This diversity has often not conformed to those prescriptions set down in RFCs, but is that a problem of the zone owners (and their software) or with the standards? Based on the ≈18 million transitions that we observed in this work, a more foundational question arises: Are operator practices exposing correctable security problems or are their implementations displaying insights that should be ingested—similar to the Continuous Improvement paradigm [46], which has been applied recently in other areas [47]? This question serves as an additional motivation for the behavior-based classification approach we described in Section VI. We believe that defining an anatomy and measuring related features can serve as a rigorous methodology, which may give rise to a sound feedback loop between standardization and operational practice.



(a) The key management of the `.com` zone shows exemplary regularity and structure in its key transitions.

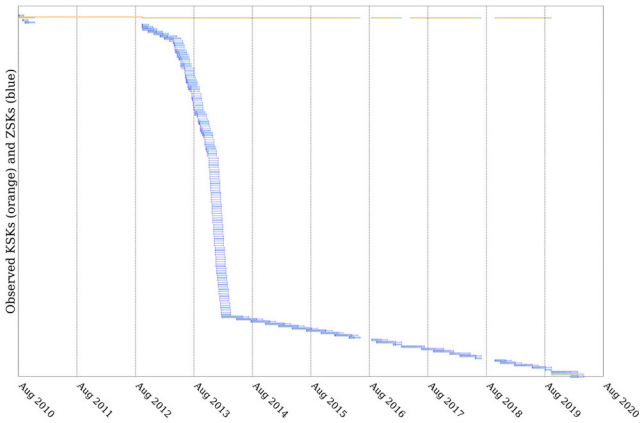


(b) Shown here is the Root zone's KSK transition away from the DURZ key (followed by revocation), and regular periodicity in its ZSK transitions.

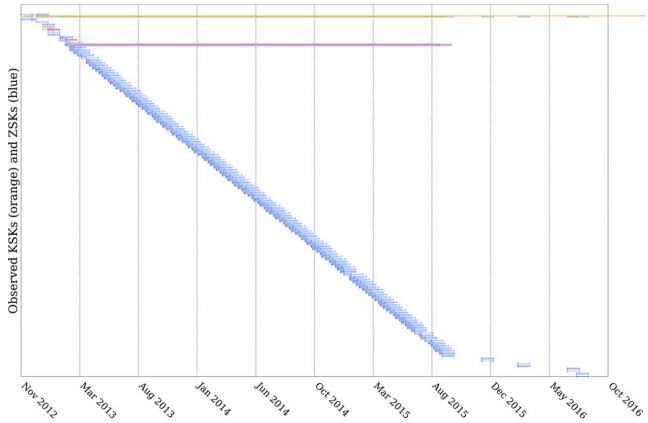
Fig. C.2. The key transition behaviors of two prominent DNSSEC zones.

XI. Conclusion and Outlook

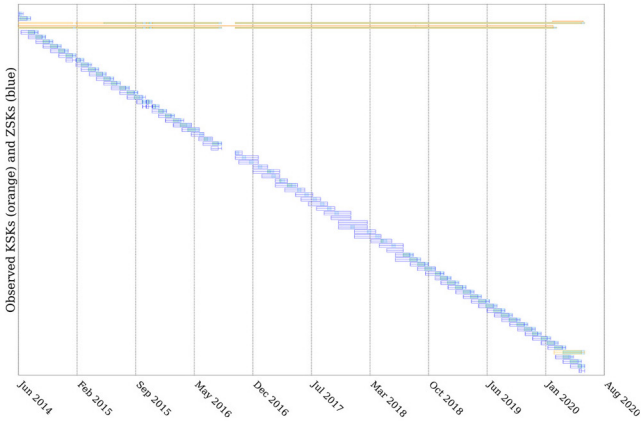
In this work, we defined the *anatomy of key transitions* as ten separate, measurable timing features based on a temporal model for DNS keys. In addition, we identified related measurable aspects of key life cycle management (e.g., relative age and key management errors), and defined a holistic behavior-based classification method for transitions. We then introduced a novel methodology for converting discrete observations into continuous `DNSKEY` lifetimes, named *Bridging, Busting, and Binding*. Using this substrate, we created anatomies of key



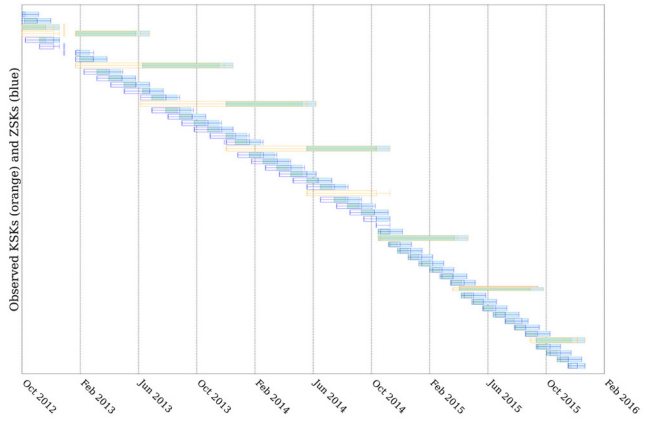
(a) `softwarehotels.se` shows distinctive key transition behavior, with up to 55 valid keys (though not all in use at the same time).



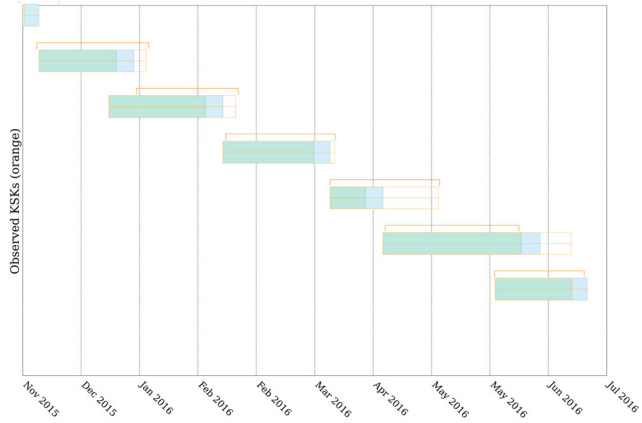
(b) The zone `cyrrax.com` shows high-frequency transitions of its ZSKs, with regular periodicity and extended *Retire* times.



(c) The key management of ARIN (a Regional Internet Registry, RIR), uses extended *PreStage* periods for its keys (ZSKs and KSKs).



(d) The key management of `advantage.gov` visibly changes from having protracted *PreStage* periods to a different model at the end of 2014.



(e) The key management of `willemvanveldhuizen.com` exhibits regular DoubleDS transitions without having any ZSKs in the zone.

Fig. C.3. Key transition patterns from other operational zones in the wild.

transitions from the wild and empirically evaluated this complex phenomenon for the first 15 years of DNSSEC operation.

Our work allows us to measure how well operators have followed guidance and, as an example of the general utility of our methodology, where related work [17] has applied. More generally, we were able to classify the kinds of key transitions operators have been using. We found that the majority of KSK key transitions and the vast majority of ZSK key transitions do not conform to RFC guidance. Additionally, using our behavior-based classification, we also observed that in some years significant rates of key transitions were rolling backward. We conclude that the anatomy and methodology of

this work serve as a useful platform for investigating DNSSEC key transitions in the wild, and results are well suited to inform evolving key transition practices.

As operational guidance, we advise operators should (*i*) aim for transition methods that reduce dependencies on parent zones and (*ii*) use the analyses of this paper to help verify that RRsets covered by keys remain verifiable when seen from external vantage points. Public recursive resolvers may serve as such vantage points to verify the state of cached entries.

Going forward, we intend to examine the specific implications of popular key transitions, to understand where deployed innovations can advise security standards. To better evaluate and understand noncompliant transitions, it is also possible to make use of statistical approaches common in artificial intelligence and machine learning to cluster similar transitions [42] or even to engineer and select new features beside the ten introduced in this work. Further, we intend to investigate the applicability of using our anatomy for other large-scale object security systems, such as the Resource Public Key Infrastructure (RPKI) [8] and the Web PKI [48]. Finally, we intend to make our feature set and the longitudinal dataset from [24] public, to encourage additional investigations from the community.

Appendix A
Raw Data of Monitored Zones

The number of DNSSEC-enabled zones that we have monitord has constantly been increasing in the past 15 years. Figure A.1 depicts the growth of secure zones for various sources in our data corpus. Table A.1 tabulates the absolute values per year and source.

Appendix B
Raw Data of Key Transition Classifications

Throughout the paper we normalize our data to better distinguish and highlight trends. Tables B.2–B.5 tabulate the raw numbers used to generate Figures 13–17.

Appendix C
Transitions in the Wild

To illustrate the complete key life cycles of zones, Figures C.2 and C.3 include both ZSKs (in blue) and KSKs (in yellow). Red sections indicate periods in which keys have the revoke bit set [6]. Notable in these (and other) zones is the difference of life cycles of ZSKs and KSKs.

References

[1] A. Langley. "Enhancing Digital Certificate Security." Google Security. 2013. [Online]. Available: https://security.googleblog.com/2013/01/enhancing-digital-certificate-security.html

[2] O. Kolkman. "DNSSEC, DANE, and Diginotar, APNIC 35 Conference." Conference Presentation. 2013. [Online]. Available: https://conference.apnic.net/__data/assets/pdf_file/0005/58901/dnssec-diginotar-dane_1361864377.pdf

[3] M. Prince. "How the Consumer Product Safety Commission is (Inadvertently) Behind the Internet's Largest DDoS Attacks." 2016. [Online]. Available: https://blog.cloudflare.com/how-the-consumer-product-safety-commission-is-inadvertently-behind-the-internets-largest-ddos-attacks/

[4] N. Trenaman. "Lessons Learned on Improving RPKI." 2020. [Online]. Available: https://labs.ripe.net/Members/nathalie_nathalie/lessons-learned-on-improving-rpki

[5] P. Mockapetris and K. J. Dunlap, "Development of the domain name system," in *Proc. ACM SIGCOMM*, New York, NY, USA, 1988, pp. 123–133.

[6] M. StJohns, "Automated updates of DNS security (DNSSEC) trust anchors," IETF, RFC 5011, Sep. 2007. [Online]. Available: http://tools.ietf.org/rfc/rfc5011.txt

[7] S. Morris, J. Ihren, J. Dickinson, and W. Mekking, "DNSSEC key rollover timing considerations," IETF, RFC 7583, Oct. 2015. [Online]. Available: http://tools.ietf.org/rfc/rfc7583.txt

[8] M. Lepinski and S. Kent, "An infrastructure to support secure Internet routing," IETF, RFC 6480, Feb. 2012. [Online]. Available: http://tools.ietf.org/rfc/rfc6480.txt

[9] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management—Part 1: General (revised)," NIST, Gaithersburg, MD, USA, NIST document Special Publication 800-57, Mar. 2007.

[10] E. Osterweil, M. Ryan, D. Massey, and L. Zhang, "Status of the DNSSEC deployment," in *Proc. ACM IMC*, New York, NY, USA, 2008, pp. 231–242.

[11] C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra, "Quantifying DNS namespace influence," *Comput. Netw.*, vol. 56, no. 2, pp. 780–794, Feb. 2012.

[12] W. Lian, E. Rescorla, H. Shacham, and S. Savage, "Measuring the practical impact of DNSSEC deployment," in *Proc. 22nd USENIX Security Symp.*, Washington, DC, USA, 2013, pp. 573–588.

[13] T. Chung *et al.*, "A longitudinal, end-to-end view of the DNSSEC ecosystem," in *Proc. 26th USENIX Security Symp.*, Washington, DC, USA, 2017, pp. 1307–1322.

[14] M. Müller, T. Chung, A. Mislove, and R. van Rijswijk-Deij, "Rolling with confidence: Managing the complexity of DNSSEC operations," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 3, pp. 1199–1211, Sep. 2019.

[15] 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 *Proc. ACM SIGCOMM Posters Demos*, New York, NY, USA, 2017, pp. 63–64.

[16] M. Müller *et al.*, "Roll, roll, roll your root: A comprehensive analysis of the first ever DNSSEC root KSK rollover," in *Proc. ACM ICN*, New York, NY, USA, 2019, pp. 1–14.

[17] Z. Wang and L. Xiao, "Emergency key rollover in DNSSEC," in *Proc. IEEE 13th Int. Conf. Trust Security Privacy Comput. Commun.*, 2014, pp. 598–604.

[18] S. M. Bellovin, "Using the domain name system for system break-ins," in *Proc. 5th USENIX Security Symp.*, 1995, p. 18.

[19] D. Atkins and R. Austein, "Threat analysis of the domain name system (DNS)," IETF, RFC 3833, Aug. 2004. [Online]. Available: http://tools.ietf.org/rfc/rfc3833.txt

[20] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS security introduction and requirements," IETF, RFC 4033, Mar. 2005. [Online]. Available: http://tools.ietf.org/rfc/rfc4033.txt

[21] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Resource records for the DNS security extensions," IETF, RFC 4034, Mar. 2005. [Online]. Available: http://tools.ietf.org/rfc/rfc4034.txt

[22] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Protocol modifications for the DNS security extensions," IETF, RFC 4035, Mar. 2005. [Online]. Available: http://tools.ietf.org/rfc/rfc4035.txt

[23] O. Kolkman, W. Mekking, and R. Gieben, "DNSSEC operational practices, version 2," IETF, RFC 6781, Dec. 2012. [Online]. Available: http://tools.ietf.org/rfc/rfc6781.txt

[24] E. Osterweil. "Secspider." [Online]. Available: https://secspider.net/ (Accessed: Jan. 2021).

[25] E. Osterweil, D. Massey, and L. Zhang, "Deploying and monitoring DNS security (DNSSEC)," in *Proc. Annu. Comput. Security Appl. Conf.*, 2009, pp. 429–438.

[26] S. Huque, P. Aras, J. Dickinson, J. Vcelak, and D. Blacka, "Multi-signer DNSSEC models," IETF, RFC 8901, Sep. 2020. [Online]. Available: http://tools.ietf.org/rfc/rfc8901.txt

[27] O. Kolkman. "DNSSEC HOWTO, A Tutorial in Disguise." Apr. 2010. [Online]. Available: https://www.dns-school.org/Documentation/dnssec_howto.pdf

[28] "Alexa Top Sites." Amazon. [Online]. Available: https://www.alexa.com/ (Accessed: Jan. 2021).

[29] "The Majestic Million." Majestic. Jan. 2021. [Online]. Available: https://majestic.com/reports/majestic-million

[30] "Cisco Umbrella 1 Million." Cisco. Jul. 2020. [Online]. Available: https://umbrella.cisco.com/blog/cisco-umbrella-1-million
[31] D. Kaminsky, "Black ops 2008: It's the end of the cache as we know it," presented at the Black Hat USA Conf. Presentation, 2008.
[32] *Vulnerability Note VU#800113: Multiple DNS Implementations Vulnerable to Cache Poisoning*. U.S. CERT Vulnerability Notes Database, Pittsburgh, PA, USA, Jul. 2008.
[33] O. Kolkman and R. Gieben, "DNSSEC operational practices," IETF, RFC 4641, Sep. 2006. [Online]. Available: http://tools.ietf.org/rfc/rfc4641.txt
[34] B. Leiba, "Ambiguity of uppercase vs lowercase in RFC 2119 key words," IETF, RFC 8174, May 2017. [Online]. Available: http://tools.ietf.org/rfc/rfc8174.txt
[35] W. Hardaker and W. Kumari, "Security considerations for RFC5011 publishers," IETF, Internet-Draft, Jul. 2018.
[36] G. Guette, B. Cousin, and D. Fort, "Algorithm for DNSSEC trusted key rollover," in *Proc. Int. Conf. Inf. Netw.*, 2005, pp. 679–688.
[37] G. Guette, "Automating trusted key rollover in DNSSEC," *J. Comput. Security*, vol. 17, no. 6, pp. 839–854, 2009.
[38] M. Nawrocki, M. Jonker, T. C. Schmidt, and M. Wählisch, "The far side of DNS amplification: Tracing the DDoS attack ecosystem from the Internet core," in *Proc. ACM IMC*, New York, NY, USA, 2021, pp. 419–434.
[39] M. Müller, W. Toorop, T. Chung, J. Jansen, and R. van Rijswijk-Deij, "The reality of algorithm agility: Studying the DNSSEC algorithm lifecycle," in *Proc. ACM Internet Meas. Conf.*, New York, NY, USA, 2020, pp. 295–308.
[40] Y. Jin, M. Tomoishi, and S. Matsuura, "A detection method against DNS cache poisoning attacks using machine learning techniques: Work in progress," in *Proc. 18th IEEE NCA*, 2019, pp. 1–3.
[41] Z. Dong, K. Kane, and L. J. Camp, "Detection of rogue certificates from trusted certificate authorities using deep neural networks," *ACM Trans. Privacy Security*, vol. 19, no. 2, pp. 1–31, Sep. 2016.
[42] Z. Luo *et al.*, "AutoSmart: An efficient and automatic machine learning framework for temporal relational data," in *Proc. 27th ACM SIGKDD*, New York, NY, USA, 2021, pp. 3976–3984.
[43] J. Andress, *The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice* (Syngress Basics Series). San Diego, CA, USA: Elsevier Sci., 2014.
[44] H. Nyquist, "Certain topics in telegraph transmission theory," *Trans. Amer. Inst. Electr. Eng.*, vol. 47, no. 2, pp. 617–644, 1928.
[45] C. E. Shannon, "Communication in the presence of noise," *Proc. IRE*, vol. 37, no. 1, pp. 10–21, Jan. 1949.
[46] W. E. Deming, *Out of the Crisis*. Cambridge, MA, USA: MIT Press, 1982.
[47] N. Bhuiyan and A. Baghel, "An overview of continuous improvement: From the past to the present," *Manag. Decis.*, vol. 43, no. 5, pp. 761–771, 2005.
[48] "Web PKI OPS (wpkops)." IETF. [Online]. Available: https://datatracker.ietf.org/wg/wpkops/charter/ (Accessed: Dec. 2020).

Eric Osterweil received the Ph.D. degree from the University of California at Los Angeles, Los Angeles in 2010. After graduating, he joined the Verisign Labs research team, where he became a Principal Scientist working in the Office of the Chief Security Officer. He is an Assistant Professor of Computer Science with George Mason University, where he directs the Measurable Security Lab. His research foci are the cybersecurity of Internet critical infrastructures, inter-organizational object-security, and Internet measurements. He is the Former Co-Chair/Vice-Chair of ICANN's Second Security, Stability, and Resiliency Review Team, an Active Member of the IETF, the maintainer of multiple open source software projects (e.g., the DANE reference library libCanute and the high-speed DNS polling library libVantages), and the Inceptor (and ongoing curator) of the oldest dataset of DNSSEC measurement dataset (SecSpider).

Pouyan Fotouhi Tehrani is currently pursuing the Ph.D. degree of computer science with the Weizenbaum Institute for the Networked Society under the supervision of M. Wählisch and T. C. Schmidt. His research evolves around communication networks for emergency and crisis with a focus on object security in fragmented and challenged networks. With publications on namespace management in Information Centric Networks (ICN), X.509-based identification on the Web, and security of DNSSEC, his research addresses both technical and organizational aspects of existing security approaches on the Internet and their proper adaptation for the next-generation Internet, e.g., named-data networks.

Thomas C. Schmidt (Member, IEEE) studied mathematics, physics, and German literature from Freie Universitaet Berlin and University of Maryland. He received the Ph.D. degree from FU Berlin in 1993. Since then, he has continuously conducted numerous national and international research projects. He was the Principal Investigator in a number of EU, nationally funded and industrial projects as well as a Visiting Professor with the University of Reading, U.K. He is a Professor of Computer Networks and Internet Technologies with the Hamburg University of Applied Sciences, where he heads the Internet Technologies research group. Prior to moving to Hamburg, he was the Director of a Scientific Computer Centre, Berlin. His continued interests lie in the development, measurement, and analysis of large-scale distributed systems like the Internet. He served as a Co-Editor and Technical Expert in many occasions and is actively involved in the work of IETF and IRTF. Together with his group he pioneered work on an information-centric Industrial IoT and the emerging data-centric Web of Things. He is a Co-Founder of several large open source projects and Coordinator of the community developing the RIOT operating system - the friendly OS for the Internet of Things.

Matthias Wählisch (Member, IEEE) received the Ph.D. degree in computer science with highest honors from Freie Universität Berlin. He is an Assistant Professor of Computer Science with Freie Universität Berlin, heading the Internet Technologies Research group. He published more than 150 peer-reviewed papers (e.g., at ACM HotNets, ACM CoNEXT, ACM IMC, USENIX Security, and The Web Conference). His research and teaching focus on efficient, reliable, and secure Internet communication. This includes the design and evaluation of networking protocols and architectures as well as Internet measurements and analysis. His efforts are driven by the goal to improve Internet communication based on sound research. He is the PI of several national and international projects, supported by overall 5.1M EUR grant money. Since 2005, he has been actively involved in Internet standardization (IETF/IRTF). His research results have been distinguished multiple times. He received the Young Talents Award of Leibniz-Kolleg Potsdam for outstanding achievements in advancing the Internet, as well as the Excellent Young Scientists Award (10,000 EUR) for his contributions to the Internet of Things and their prospective entrepreneurial practice. He received the Best of ACM SIGCOMM CCR Award in 2018 and 2019. He co-founded and still coordinates some successful open source projects in the context of Internet of Things (RIOT) and secure Internet routing (e.g., RTRlib).