Network Working Group
Request for Comments: 4641
Obsoletes: 2541
Category: Informational

O. Kolkman
R. Gieben
NLnet Labs

September 2006

DNSSEC Operational Practices

Status of This Memo

Copyright Notice

Abstract

This document describes a set of practices for operating the DNS with security extensions (DNSSEC). The target audience is zone administrators deploying DNSSEC.

The document discusses operational aspects of using keys and signatures in the DNS. It discusses issues of key generation, key storage, signature generation, key rollover, and related policies.

This document obsoletes RFC 2541, as it covers more operational ground and gives more up-to-date requirements with respect to key sizes and the new DNSSEC specification.

Table of Contents

1. Introduction

This document describes how to run a DNS Security (DNSSEC)-enabled environment. It is intended for operators who have knowledge of the DNS (see RFC 1034 [1] and RFC 1035 [2]) and want to deploy DNSSEC. See RFC 4033 [4] for an introduction to DNSSEC, RFC 4034 [5] for the newly introduced Resource Records (RRs), and RFC 4035 [6] for the protocol changes.

During workshops and early operational deployment tests, operators and system administrators have gained experience about operating the DNS with security extensions (DNSSEC). This document translates these experiences into a set of practices for zone administrators. At the time of writing, there exists very little experience with DNSSEC in production environments; this document should therefore explicitly not be seen as representing 'Best Current Practices'.

The procedures herein are focused on the maintenance of signed zones (i.e., signing and publishing zones on authoritative servers). It is intended that maintenance of zones such as re-signing or key rollovers be transparent to any verifying clients on the Internet.

The structure of this document is as follows. In Section 2, we discuss the importance of keeping the "chain of trust" intact. Aspects of key generation and storage of private keys are discussed in Section 3; the focus in this section is mainly on the private part of the key(s). Section 4 describes considerations concerning the public part of the keys. Since these public keys appear in the DNS one has to take into account all kinds of timing issues, which are discussed in Section 4.1. Section 4.2 and Section 4.3 deal with the rollover, or supercession, of keys. Finally, Section 4.4 discusses considerations on how parents deal with their children's public keys in order to maintain chains of trust.

The typographic conventions used in this document are explained in Appendix C.

Since this is a document with operational suggestions and there are no protocol specifications, the RFC 2119 [7] language does not apply.

This document obsoletes RFC 2541 [12] to reflect the evolution of the underlying DNSSEC protocol since then. Changes in the choice of cryptographic algorithms, DNS record types and type names, and the parent-child key and signature exchange demanded a major rewrite and additional information and explanation.

1.1. The Use of the Term 'key'

It is assumed that the reader is familiar with the concept of asymmetric keys on which DNSSEC is based (public key cryptography [17]). Therefore, this document will use the term 'key' rather loosely. Where it is written that 'a key is used to sign data' it is assumed that the reader understands that it is the private part of the key pair that is used for signing. It is also assumed that the reader understands that the public part of the key pair is published in the DNSKEY Resource Record and that it is the public part that is used in key exchanges.

1.2. Time Definitions

In this document, we will be using a number of time-related terms. The following definitions apply:

- "Signature validity period" The period that a signature is valid. It starts at the time specified in the signature inception field of the RRSIG RR and ends at the time specified in the expiration field of the RRSIG RR.

- "Signature publication period" Time after which a signature (made with a specific key) is replaced with a new signature (made with the same key). This replacement takes place by publishing the relevant RRSIG in the master zone file. After one stops publishing an RRSIG in a zone, it may take a while before the RRSIG has expired from caches and has actually been removed from the DNS.

- "Key effectivity period" The period during which a key pair is expected to be effective. This period is defined as the time between the first inception time stamp and the last expiration date of any signature made with this key, regardless of any discontinuity in the use of the key. The key effectivity period can span multiple signature validity periods.

- "Maximum/Minimum Zone Time to Live (TTL)" The maximum or minimum value of the TTLs from the complete set of RRs in a zone. Note that the minimum TTL is not the same as the MINIMUM field in the SOA RR. See [11] for more information.

2. Keeping the Chain of Trust Intact

Maintaining a valid chain of trust is important because broken chains of trust will result in data being marked as Bogus (as defined in [4] Section 5), which may cause entire (sub)domains to become invisible to verifying clients. The administrators of secured zones have to realize that their zone is, to verifying clients, part of a chain of trust.

As mentioned in the introduction, the procedures herein are intended to ensure that maintenance of zones, such as re-signing or key rollovers, will be transparent to the verifying clients on the Internet.

Administrators of secured zones will have to keep in mind that data published on an authoritative primary server will not be immediately seen by verifying clients; it may take some time for the data to be transferred to other secondary authoritative nameservers and clients may be fetching data from caching non-authoritative servers. In this light, note that the time for a zone transfer from master to slave is negligible when using NOTIFY [9] and incremental transfer (IXFR) [8]. It increases when full zone transfers (AXFR) are used in combination with NOTIFY. It increases even more if you rely on full zone transfers based on only the SOA timing parameters for refresh.

For the verifying clients, it is important that data from secured zones can be used to build chains of trust regardless of whether the data came directly from an authoritative server, a caching nameserver, or some middle box. Only by carefully using the available timing parameters can a zone administrator ensure that the data necessary for verification can be obtained.

The responsibility for maintaining the chain of trust is shared by administrators of secured zones in the chain of trust. This is most obvious in the case of a 'key compromise' when a trade-off between maintaining a valid chain of trust and replacing the compromised keys as soon as possible must be made. Then zone administrators will have to make a trade-off, between keeping the chain of trust intact -- thereby allowing for attacks with the compromised key -- or deliberately breaking the chain of trust and making secured subdomains invisible to security-aware resolvers. Also see Section 4.3.

3. Keys Generation and Storage

This section describes a number of considerations with respect to the security of keys. It deals with the generation, effectivity period, size, and storage of private keys.

3.1. Zone and Key Signing Keys

The DNSSEC validation protocol does not distinguish between different types of DNSKEYs. All DNSKEYs can be used during the validation. In practice, operators use Key Signing and Zone Signing Keys and use the so-called Secure Entry Point (SEP) [3] flag to distinguish between them during operations. The dynamics and considerations are discussed below.

To make zone re-signing and key rollover procedures easier to implement, it is possible to use one or more keys as Key Signing Keys (KSKs). These keys will only sign the apex DNSKEY RRSet in a zone. Other keys can be used to sign all the RRSets in a zone and are referred to as Zone Signing Keys (ZSKs). In this document, we assume that KSKs are the subset of keys that are used for key exchanges with the parent and potentially for configuration as trusted anchors -- the SEP keys. In this document, we assume a one-to-one mapping between KSK and SEP keys and we assume the SEP flag to be set on all KSKs.

3.1.1. Motivations for the KSK and ZSK Separation

Differentiating between the KSK and ZSK functions has several advantages:

- No parent/child interaction is required when ZSKs are updated.
- The KSK can be made stronger (i.e., using more bits in the key material). This has little operational impact since it is only used to sign a small fraction of the zone data. Also, the KSK is only used to verify the zone's key set, not for other RRSets in the zone.
- As the KSK is only used to sign a key set, which is most probably updated less frequently than other data in the zone, it can be stored separately from and in a safer location than the ZSK.
- A KSK can have a longer key effectivity period.

For almost any method of key management and zone signing, the KSK is used less frequently than the ZSK. Once a key set is signed with the KSK, all the keys in the key set can be used as ZSKs. If a ZSK is

compromised, it can be simply dropped from the key set. The new key set is then re-signed with the KSK.

Given the assumption that for KSKs the SEP flag is set, the KSK can be distinguished from a ZSK by examining the flag field in the DNSKEY RR. If the flag field is an odd number it is a KSK. If it is an even number it is a ZSK.

The Zone Signing Key can be used to sign all the data in a zone on a regular basis. When a Zone Signing Key is to be rolled, no interaction with the parent is needed. This allows for signature validity periods on the order of days.

The Key Signing Key is only to be used to sign the DNSKEY RRs in a zone. If a Key Signing Key is to be rolled over, there will be interactions with parties other than the zone administrator. These can include the registry of the parent zone or administrators of verifying resolvers that have the particular key configured as secure entry points. Hence, the key effectivity period of these keys can and should be made much longer. Although, given a long enough key, the key effectivity period can be on the order of years, we suggest planning for a key effectivity on the order of a few months so that a key rollover remains an operational routine.

3.1.2. KSKs for High-Level Zones

Higher-level zones are generally more sensitive than lower-level zones. Anyone controlling or breaking the security of a zone thereby obtains authority over all of its subdomains (except in the case of resolvers that have locally configured the public key of a subdomain, in which case this, and only this, subdomain wouldn't be affected by the compromise of the parent zone). Therefore, extra care should be taken with high-level zones, and strong keys should be used.

The root zone is the most critical of all zones. Someone controlling or compromising the security of the root zone would control the entire DNS namespace of all resolvers using that root zone (except in the case of resolvers that have locally configured the public key of a subdomain). Therefore, the utmost care must be taken in the securing of the root zone. The strongest and most carefully handled keys should be used. The root zone private key should always be kept off-line.

Many resolvers will start at a root server for their access to and authentication of DNS data. Securely updating the trust anchors in an enormous population of resolvers around the world will be extremely difficult.

3.2. Key Generation

Careful generation of all keys is a sometimes overlooked but absolutely essential element in any cryptographically secure system. The strongest algorithms used with the longest keys are still of no use if an adversary can guess enough to lower the size of the likely key space so that it can be exhaustively searched. Technical suggestions for the generation of random keys will be found in RFC 4086 [14]. One should carefully assess if the random number generator used during key generation adheres to these suggestions.

Keys with a long effectivity period are particularly sensitive as they will represent a more valuable target and be subject to attack for a longer time than short-period keys. It is strongly recommended that long-term key generation occur off-line in a manner isolated from the network via an air gap or, at a minimum, high-level secure hardware.

3.3. Key Effectivity Period

For various reasons, keys in DNSSEC need to be changed once in a while. The longer a key is in use, the greater the probability that it will have been compromised through carelessness, accident, espionage, or cryptanalysis. Furthermore, when key rollovers are too rare an event, they will not become part of the operational habit and there is risk that nobody on-site will remember the procedure for rollover when the need is there.

From a purely operational perspective, a reasonable key effectivity period for Key Signing Keys is 13 months, with the intent to replace them after 12 months. An intended key effectivity period of a month is reasonable for Zone Signing Keys.

For key sizes that match these effectivity periods, see Section 3.5.

As argued in Section 3.1.2, securely updating trust anchors will be extremely difficult. On the other hand, the "operational habit" argument does also apply to trust anchor reconfiguration. If a short key effectivity period is used and the trust anchor configuration has to be revisited on a regular basis, the odds that the configuration tends to be forgotten is smaller. The trade-off is against a system that is so dynamic that administrators of the validating clients will not be able to follow the modifications.

Key effectivity periods can be made very short, as in a few minutes. But when replacing keys one has to take the considerations from Section 4.1 and Section 4.2 into account.

3.4. Key Algorithm

There are currently three different types of algorithms that can be used in DNSSEC: RSA, DSA, and elliptic curve cryptography. The latter is fairly new and has yet to be standardized for usage in DNSSEC.

RSA has been developed in an open and transparent manner. As the patent on RSA expired in 2000, its use is now also free.

DSA has been developed by the National Institute of Standards and Technology (NIST). The creation of signatures takes roughly the same time as with RSA, but is 10 to 40 times as slow for verification [17].

We suggest the use of RSA/SHA-1 as the preferred algorithm for the key. The current known attacks on RSA can be defeated by making your key longer. As the MD5 hashing algorithm is showing cracks, we recommend the usage of SHA-1.

At the time of publication, it is known that the SHA-1 hash has cryptanalysis issues. There is work in progress on addressing these issues. We recommend the use of public key algorithms based on hashes stronger than SHA-1 (e.g., SHA-256), as soon as these algorithms are available in protocol specifications (see [19] and [20]) and implementations.

3.5. Key Sizes

When choosing key sizes, zone administrators will need to take into account how long a key will be used, how much data will be signed during the key publication period (see Section 8.10 of [17]), and, optionally, how large the key size of the parent is. As the chain of trust really is "a chain", there is not much sense in making one of the keys in the chain several times larger then the others. As always, it's the weakest link that defines the strength of the entire chain. Also see Section 3.1.1 for a discussion of how keys serving different roles (ZSK vs. KSK) may need different key sizes.

Generating a key of the correct size is a difficult problem; RFC 3766 [13] tries to deal with that problem. The first part of the selection procedure in Section 1 of the RFC states:

1. Determine the attack resistance necessary to satisfy the security requirements of the application. Do this by estimating the minimum number of computer operations that the attacker will be forced to do in order to compromise the

> security of the system and then take the logarithm base two of that number. Call that logarithm value "n".
>
> A 1996 report recommended 90 bits as a good all-around choice for system security. The 90 bit number should be increased by about 2/3 bit/year, or about 96 bits in 2005.

[13] goes on to explain how this number "n" can be used to calculate the key sizes in public key cryptography. This culminated in the table given below (slightly modified for our purpose):

System requirement for attack resistance (bits)	Symmetric key size (bits)	RSA or DSA modulus size (bits)
70	70	947
80	80	1228
90	90	1553
100	100	1926
150	150	4575
200	200	8719
250	250	14596

The key sizes given are rather large. This is because these keys are resilient against a trillionaire attacker. Assuming this rich attacker will not attack your key and that the key is rolled over once a year, we come to the following recommendations about KSK sizes: 1024 bits for low-value domains, 1300 bits for medium-value domains, and 2048 bits for high-value domains.

Whether a domain is of low, medium, or high value depends solely on the views of the zone owner. One could, for instance, view leaf nodes in the DNS as of low value, and top-level domains (TLDs) or the root zone of high value. The suggested key sizes should be safe for the next 5 years.

As ZSKs can be rolled over more easily (and thus more often), the key sizes can be made smaller. But as said in the introduction of this paragraph, making the ZSKs' key sizes too small (in relation to the KSKs' sizes) doesn't make much sense. Try to limit the difference in size to about 100 bits.

Note that nobody can see into the future and that these key sizes are only provided here as a guide. Further information can be found in [16] and Section 7.5 of [17]. It should be noted though that [16] is already considered overly optimistic about what key sizes are considered safe.

One final note concerning key sizes. Larger keys will increase the sizes of the RRSIG and DNSKEY records and will therefore increase the chance of DNS UDP packet overflow. Also, the time it takes to validate and create RRSIGs increases with larger keys, so don't needlessly double your key sizes.

3.6. Private Key Storage

It is recommended that, where possible, zone private keys and the zone file master copy that is to be signed be kept and used in off-line, non-network-connected, physically secure machines only. Periodically, an application can be run to add authentication to a zone by adding RRSIG and NSEC RRs. Then the augmented file can be transferred.

When relying on dynamic update to manage a signed zone [10], be aware that at least one private key of the zone will have to reside on the master server. This key is only as secure as the amount of exposure the server receives to unknown clients and the security of the host. Although not mandatory, one could administer the DNS in the following way. The master that processes the dynamic updates is unavailable from generic hosts on the Internet, it is not listed in the NS RR set, although its name appears in the SOA RRs MNAME field. The nameservers in the NS RRSet are able to receive zone updates through NOTIFY, IXFR, AXFR, or an out-of-band distribution mechanism. This approach is known as the "hidden master" setup.

The ideal situation is to have a one-way information flow to the network to avoid the possibility of tampering from the network. Keeping the zone master file on-line on the network and simply cycling it through an off-line signer does not do this. The on-line version could still be tampered with if the host it resides on is compromised. For maximum security, the master copy of the zone file should be off-net and should not be updated based on an unsecured network mediated communication.

In general, keeping a zone file off-line will not be practical and the machines on which zone files are maintained will be connected to a network. Operators are advised to take security measures to shield unauthorized access to the master copy.

For dynamically updated secured zones [10], both the master copy and the private key that is used to update signatures on updated RRs will need to be on-line.

4. Signature Generation, Key Rollover, and Related Policies

4.1. Time in DNSSEC

Without DNSSEC, all times in the DNS are relative. The SOA fields REFRESH, RETRY, and EXPIRATION are timers used to determine the time elapsed after a slave server synchronized with a master server. The Time to Live (TTL) value and the SOA RR minimum TTL parameter [11] are used to determine how long a forwarder should cache data after it has been fetched from an authoritative server. By using a signature validity period, DNSSEC introduces the notion of an absolute time in the DNS. Signatures in DNSSEC have an expiration date after which the signature is marked as invalid and the signed data is to be considered Bogus.

4.1.1. Time Considerations

Because of the expiration of signatures, one should consider the following:

- We suggest the Maximum Zone TTL of your zone data to be a fraction of your signature validity period.

  If the TTL would be of similar order as the signature validity period, then all RRSets fetched during the validity period would be cached until the signature expiration time. Section 7.1 of [4] suggests that "the resolver may use the time remaining before expiration of the signature validity period of a signed RRSet as an upper bound for the TTL". As a result, query load on authoritative servers would peak at signature expiration time, as this is also the time at which records simultaneously expire from caches.

  To avoid query load peaks, we suggest the TTL on all the RRs in your zone to be at least a few times smaller than your signature validity period.

- We suggest the signature publication period to end at least one Maximum Zone TTL duration before the end of the signature validity period.

Re-signing a zone shortly before the end of the signature validity period may cause simultaneous expiration of data from caches. This in turn may lead to peaks in the load on authoritative servers.

- We suggest the Minimum Zone TTL to be long enough to both fetch and verify all the RRs in the trust chain. In workshop environments, it has been demonstrated [18] that a low TTL (under 5 to 10 minutes) caused disruptions because of the following two problems:

  1. During validation, some data may expire before the validation is complete. The validator should be able to keep all data until it is completed. This applies to all RRs needed to complete the chain of trust: DSes, DNSKEYs, RRSIGs, and the final answers, i.e., the RRSet that is returned for the initial query.

  2. Frequent verification causes load on recursive nameservers. Data at delegation points, DSes, DNSKEYs, and RRSIGs benefit from caching. The TTL on those should be relatively long.

- Slave servers will need to be able to fetch newly signed zones well before the RRSIGs in the zone served by the slave server pass their signature expiration time.

  When a slave server is out of sync with its master and data in a zone is signed by expired signatures, it may be better for the slave server not to give out any answer.

  Normally, a slave server that is not able to contact a master server for an extended period will expire a zone. When that happens, the server will respond differently to queries for that zone. Some servers issue SERVFAIL, whereas others turn off the 'AA' bit in the answers. The time of expiration is set in the SOA record and is relative to the last successful refresh between the master and the slave servers. There exists no coupling between the signature expiration of RRSIGs in the zone and the expire parameter in the SOA.

  If the server serves a DNSSEC zone, then it may well happen that the signatures expire well before the SOA expiration timer counts down to zero. It is not possible to completely prevent this from happening by tweaking the SOA parameters. However, the effects can be minimized where the SOA expiration time is equal to or shorter than the signature validity period. The consequence of an authoritative server not being able to update

a zone, whilst that zone includes expired signatures, is that non-secure resolvers will continue to be able to resolve data served by the particular slave servers while security-aware resolvers will experience problems because of answers being marked as Bogus.

We suggest the SOA expiration timer being approximately one third or one fourth of the signature validity period. It will allow problems with transfers from the master server to be noticed before the actual signature times out. We also suggest that operators of nameservers that supply secondary services develop 'watch dogs' to spot upcoming signature expirations in zones they slave, and take appropriate action.

When determining the value for the expiration parameter one has to take the following into account: What are the chances that all my secondaries expire the zone? How quickly can I reach an administrator of secondary servers to load a valid zone? These questions are not DNSSEC specific but may influence the choice of your signature validity intervals.

4.2. Key Rollovers

A DNSSEC key cannot be used forever (see Section 3.3). So key rollovers -- or supercessions, as they are sometimes called -- are a fact of life when using DNSSEC. Zone administrators who are in the process of rolling their keys have to take into account that data published in previous versions of their zone still lives in caches. When deploying DNSSEC, this becomes an important consideration; ignoring data that may be in caches may lead to loss of service for clients.

The most pressing example of this occurs when zone material signed with an old key is being validated by a resolver that does not have the old zone key cached. If the old key is no longer present in the current zone, this validation fails, marking the data "Bogus". Alternatively, an attempt could be made to validate data that is signed with a new key against an old key that lives in a local cache, also resulting in data being marked "Bogus".

4.2.1. Zone Signing Key Rollovers

For "Zone Signing Key rollovers", there are two ways to make sure that during the rollover data still cached can be verified with the new key sets or newly generated signatures can be verified with the keys still in caches. One schema, described in Section 4.2.1.2, uses

double signatures; the other uses key pre-publication (Section 4.2.1.1). The pros, cons, and recommendations are described in Section 4.2.1.3.

4.2.1.1. Pre-Publish Key Rollover

This section shows how to perform a ZSK rollover without the need to sign all the data in a zone twice -- the "pre-publish key rollover". This method has advantages in the case of a key compromise. If the old key is compromised, the new key has already been distributed in the DNS. The zone administrator is then able to quickly switch to the new key and remove the compromised key from the zone. Another major advantage is that the zone size does not double, as is the case with the double signature ZSK rollover. A small "how-to" for this kind of rollover can be found in Appendix B.

Pre-publish key rollover involves four stages as follows:

initial	new DNSKEY	new RRSIGs	DNSKEY removal
SOA0	SOA1	SOA2	SOA3
RRSIG10(SOA0)	RRSIG10(SOA1)	RRSIG11(SOA2)	RRSIG11(SOA3)
DNSKEY1	DNSKEY1	DNSKEY1	DNSKEY1
DNSKEY10	DNSKEY10	DNSKEY10	DNSKEY11
DNSKEY11	DNSKEY11		
RRSIG1 (DNSKEY)	RRSIG1 (DNSKEY)	RRSIG1(DNSKEY)	RRSIG1 (DNSKEY)
RRSIG10(DNSKEY)	RRSIG10(DNSKEY)	RRSIG11(DNSKEY)	RRSIG11(DNSKEY)

Pre-Publish Key Rollover

initial: Initial version of the zone: DNSKEY 1 is the Key Signing Key. DNSKEY 10 is used to sign all the data of the zone, the Zone Signing Key.

new DNSKEY: DNSKEY 11 is introduced into the key set. Note that no signatures are generated with this key yet, but this does not secure against brute force attacks on the public key. The minimum duration of this pre-roll phase is the time it takes for the data to propagate to the authoritative servers plus TTL value of the key set.

new RRSIGs: At the "new RRSIGs" stage (SOA serial 2), DNSKEY 11 is used to sign the data in the zone exclusively (i.e., all the signatures from DNSKEY 10 are removed from the zone). DNSKEY 10 remains published in the key set. This way data that was loaded

into caches from version 1 of the zone can still be verified with key sets fetched from version 2 of the zone. The minimum time that the key set including DNSKEY 10 is to be published is the time that it takes for zone data from the previous version of the zone to expire from old caches, i.e., the time it takes for this zone to propagate to all authoritative servers plus the Maximum Zone TTL value of any of the data in the previous version of the zone.

DNSKEY removal: DNSKEY 10 is removed from the zone. The key set, now only containing DNSKEY 1 and DNSKEY 11, is re-signed with the DNSKEY 1.

The above scheme can be simplified by always publishing the "future" key immediately after the rollover. The scheme would look as follows (we show two rollovers); the future key is introduced in "new DNSKEY" as DNSKEY 12 and again a newer one, numbered 13, in "new DNSKEY (II)":

initial	new RRSIGs	new DNSKEY
SOA0	SOA1	SOA2
RRSIG10(SOA0)	RRSIG11(SOA1)	RRSIG11(SOA2)
DNSKEY1	DNSKEY1	DNSKEY1
DNSKEY10	DNSKEY10	DNSKEY11
DNSKEY11	DNSKEY11	DNSKEY12
RRSIG1(DNSKEY)	RRSIG1 (DNSKEY)	RRSIG1(DNSKEY)
RRSIG10(DNSKEY)	RRSIG11(DNSKEY)	RRSIG11(DNSKEY)

new RRSIGs (II)	new DNSKEY (II)
SOA3	SOA4
RRSIG12(SOA3)	RRSIG12(SOA4)
DNSKEY1	DNSKEY1
DNSKEY11	DNSKEY12
DNSKEY12	DNSKEY13
RRSIG1(DNSKEY)	RRSIG1(DNSKEY)
RRSIG12(DNSKEY)	RRSIG12(DNSKEY)

Pre-Publish Key Rollover, Showing Two Rollovers

Note that the key introduced in the "new DNSKEY" phase is not used for production yet; the private key can thus be stored in a physically secure manner and does not need to be 'fetched' every time a zone needs to be signed.

4.2.1.2. Double Signature Zone Signing Key Rollover

This section shows how to perform a ZSK key rollover using the double zone data signature scheme, aptly named "double signature rollover".

During the "new DNSKEY" stage the new version of the zone file will need to propagate to all authoritative servers and the data that exists in (distant) caches will need to expire, requiring at least the Maximum Zone TTL.

Double signature ZSK rollover involves three stages as follows:

initial	new DNSKEY	DNSKEY removal
SOA0	SOA1	SOA2
RRSIG10(SOA0)	RRSIG10(SOA1)	RRSIG11(SOA2)
RRSIG11(SOA1)		
DNSKEY1	DNSKEY1	DNSKEY1
DNSKEY10	DNSKEY10	DNSKEY11
DNSKEY11		
RRSIG1(DNSKEY)	RRSIG1(DNSKEY)	RRSIG1(DNSKEY)
RRSIG10(DNSKEY)	RRSIG10(DNSKEY)	RRSIG11(DNSKEY)
RRSIG11(DNSKEY)		

Double Signature Zone Signing Key Rollover

initial: Initial Version of the zone: DNSKEY 1 is the Key Signing Key. DNSKEY 10 is used to sign all the data of the zone, the Zone Signing Key.

new DNSKEY: At the "New DNSKEY" stage (SOA serial 1) DNSKEY 11 is introduced into the key set and all the data in the zone is signed with DNSKEY 10 and DNSKEY 11. The rollover period will need to continue until all data from version 0 of the zone has expired from remote caches. This will take at least the Maximum Zone TTL of version 0 of the zone.

DNSKEY removal: DNSKEY 10 is removed from the zone. All the signatures from DNSKEY 10 are removed from the zone. The key set, now only containing DNSKEY 11, is re-signed with DNSKEY 1.

At every instance, RRSIGs from the previous version of the zone can be verified with the DNSKEY RRSet from the current version and the other way around. The data from the current version can be verified with the data from the previous version of the zone. The duration of the "new DNSKEY" phase and the period between rollovers should be at least the Maximum Zone TTL.

Making sure that the "new DNSKEY" phase lasts until the signature expiration time of the data in initial version of the zone is recommended. This way all caches are cleared of the old signatures. However, this duration could be considerably longer than the Maximum Zone TTL, making the rollover a lengthy procedure.

Note that in this example we assumed that the zone was not modified during the rollover. New data can be introduced in the zone as long as it is signed with both keys.

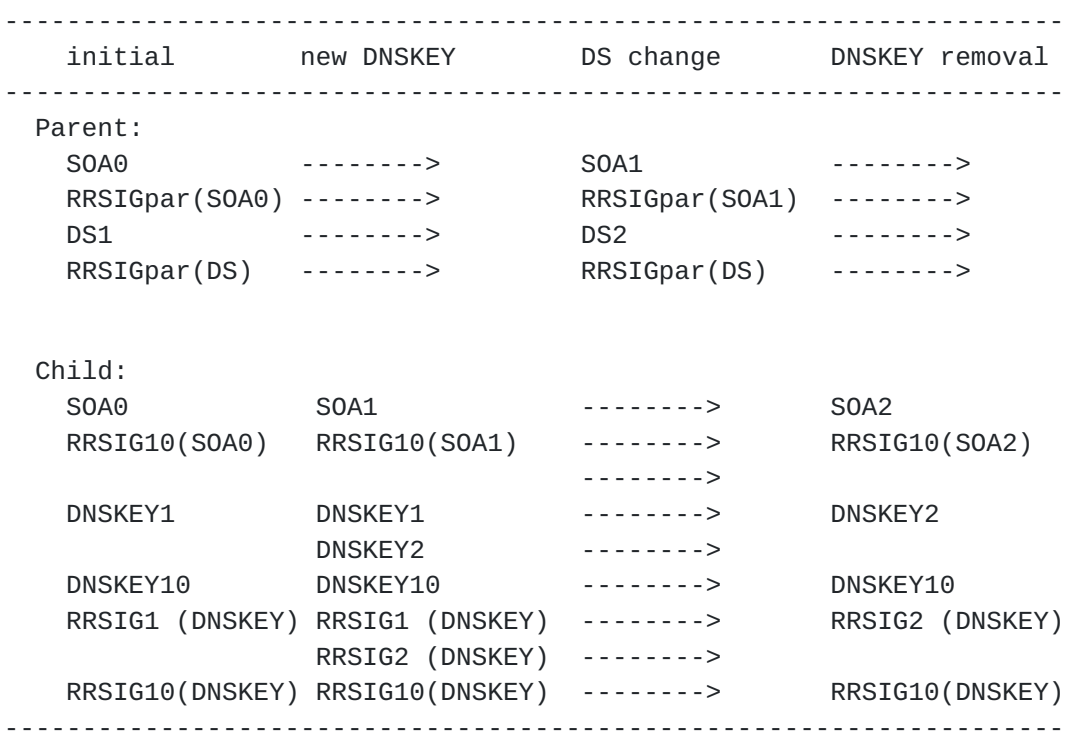
4.2.1.3. Pros and Cons of the Schemes

Pre-publish key rollover: This rollover does not involve signing the zone data twice. Instead, before the actual rollover, the new key is published in the key set and thus is available for cryptanalysis attacks. A small disadvantage is that this process requires four steps. Also the pre-publish scheme involves more parental work when used for KSK rollovers as explained in Section 4.2.3.

Double signature ZSK rollover: The drawback of this signing scheme is that during the rollover the number of signatures in your zone doubles; this may be prohibitive if you have very big zones. An advantage is that it only requires three steps.

4.2.2. Key Signing Key Rollovers

For the rollover of a Key Signing Key, the same considerations as for the rollover of a Zone Signing Key apply. However, we can use a double signature scheme to guarantee that old data (only the apex key set) in caches can be verified with a new key set and vice versa. Since only the key set is signed with a KSK, zone size considerations do not apply.



Stages of Deployment for a Double Signature Key Signing Key Rollover

initial: Initial version of the zone. The parental DS points to DNSKEY1. Before the rollover starts, the child will have to verify what the TTL is of the DS RR that points to DNSKEY1 -- it is needed during the rollover and we refer to the value as TTL_DS.

new DNSKEY: During the "new DNSKEY" phase, the zone administrator generates a second KSK, DNSKEY2. The key is provided to the parent, and the child will have to wait until a new DS RR has been generated that points to DNSKEY2. After that DS RR has been published on all servers authoritative for the parent's zone, the zone administrator has to wait at least TTL_DS to make sure that the old DS RR has expired from caches.

DS change: The parent replaces DS1 with DS2.

DNSKEY removal: DNSKEY1 has been removed.

The scenario above puts the responsibility for maintaining a valid chain of trust with the child. It also is based on the premise that the parent only has one DS RR (per algorithm) per zone. An alternative mechanism has been considered. Using an established trust relation, the interaction can be performed in-band, and the removal of the keys by the child can possibly be signaled by the parent. In this mechanism, there are periods where there are two DS

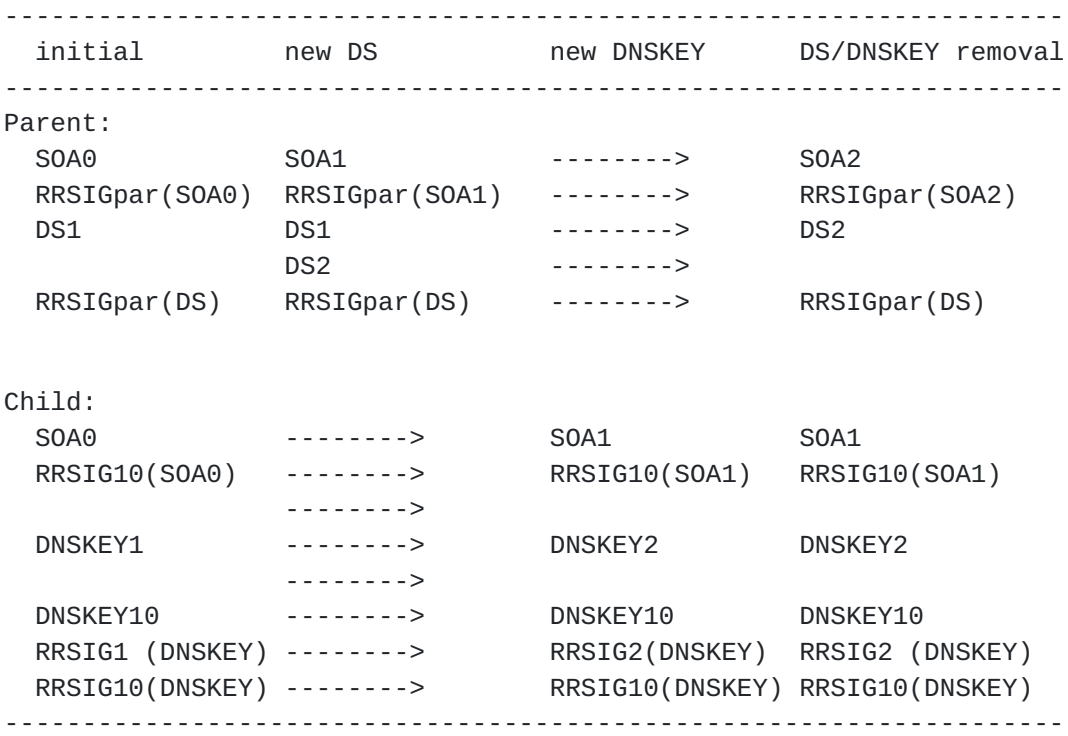
RRs at the parent. Since at the moment of writing the protocol for this interaction has not been developed, further discussion is out of scope for this document.

4.2.3. Difference Between ZSK and KSK Rollovers

Note that KSK rollovers and ZSK rollovers are different in the sense that a KSK rollover requires interaction with the parent (and possibly replacing of trust anchors) and the ensuing delay while waiting for it.

A zone key rollover can be handled in two different ways: pre-publish (Section 4.2.1.1) and double signature (Section 4.2.1.2).

As the KSK is used to validate the key set and because the KSK is not changed during a ZSK rollover, a cache is able to validate the new key set of the zone. The pre-publish method would also work for a KSK rollover. The records that are to be pre-published are the parental DS RRs. The pre-publish method has some drawbacks for KSKs. We first describe the rollover scheme and then indicate these drawbacks.



Stages of Deployment for a Pre-Publish Key Signing Key Rollover

When the child zone wants to roll, it notifies the parent during the "new DS" phase and submits the new key (or the corresponding DS) to the parent. The parent publishes DS1 and DS2, pointing to DNSKEY1 and DNSKEY2, respectively. During the rollover ("new DNSKEY" phase), which can take place as soon as the new DS set propagated through the DNS, the child replaces DNSKEY1 with DNSKEY2. Immediately after that ("DS/DNSKEY removal" phase), it can notify the parent that the old DS record can be deleted.

The drawbacks of this scheme are that during the "new DS" phase the parent cannot verify the match between the DS2 RR and DNSKEY2 using the DNS -- as DNSKEY2 is not yet published. Besides, we introduce a "security lame" key (see Section 4.4.3). Finally, the child-parent interaction consists of two steps. The "double signature" method only needs one interaction.

4.2.4. Automated Key Rollovers

As keys must be renewed periodically, there is some motivation to automate the rollover process. Consider the following:

- ZSK rollovers are easy to automate as only the child zone is involved.

- A KSK rollover needs interaction between parent and child. Data exchange is needed to provide the new keys to the parent; consequently, this data must be authenticated and integrity must be guaranteed in order to avoid attacks on the rollover.

4.3. Planning for Emergency Key Rollover

This section deals with preparation for a possible key compromise. Our advice is to have a documented procedure ready for when a key compromise is suspected or confirmed.

When the private material of one of your keys is compromised it can be used for as long as a valid trust chain exists. A trust chain remains intact for

- as long as a signature over the compromised key in the trust chain is valid,

- as long as a parental DS RR (and signature) points to the compromised key,

- as long as the key is anchored in a resolver and is used as a starting point for validation (this is generally the hardest to update).

While a trust chain to your compromised key exists, your namespace is vulnerable to abuse by anyone who has obtained illegitimate possession of the key. Zone operators have to make a trade-off if the abuse of the compromised key is worse than having data in caches that cannot be validated. If the zone operator chooses to break the trust chain to the compromised key, data in caches signed with this key cannot be validated. However, if the zone administrator chooses to take the path of a regular rollover, the malicious key holder can spoof data so that it appears to be valid.

4.3.1. KSK Compromise

A zone containing a DNSKEY RRSet with a compromised KSK is vulnerable as long as the compromised KSK is configured as trust anchor or a parental DS points to it.

A compromised KSK can be used to sign the key set of an attacker's zone. That zone could be used to poison the DNS.

Therefore, when the KSK has been compromised, the trust anchor or the parental DS should be replaced as soon as possible. It is local policy whether to break the trust chain during the emergency rollover. The trust chain would be broken when the compromised KSK is removed from the child's zone while the parent still has a DS pointing to the compromised KSK (the assumption is that there is only one DS at the parent. If there are multiple DSes this does not apply -- however the chain of trust of this particular key is broken).

Note that an attacker's zone still uses the compromised KSK and the presence of a parental DS would cause the data in this zone to appear as valid. Removing the compromised key would cause the attacker's zone to appear as valid and the child's zone as Bogus. Therefore, we advise not to remove the KSK before the parent has a DS to a new KSK in place.

4.3.1.1. Keeping the Chain of Trust Intact

If we follow this advice, the timing of the replacement of the KSK is somewhat critical. The goal is to remove the compromised KSK as soon as the new DS RR is available at the parent. And also make sure that the signature made with a new KSK over the key set with the compromised KSK in it expires just after the new DS appears at the parent, thus removing the old cruft in one swoop.

The procedure is as follows:

1. Introduce a new KSK into the key set, keep the compromised KSK in the key set.

2. Sign the key set, with a short validity period. The validity period should expire shortly after the DS is expected to appear in the parent and the old DSes have expired from caches.

3. Upload the DS for this new key to the parent.

4. Follow the procedure of the regular KSK rollover: Wait for the DS to appear in the authoritative servers and then wait as long as the TTL of the old DS RRs. If necessary re-sign the DNSKEY RRSet and modify/extend the expiration time.

5. Remove the compromised DNSKEY RR from the zone and re-sign the key set using your "normal" validity interval.

An additional danger of a key compromise is that the compromised key could be used to facilitate a legitimate DNSKEY/DS rollover and/or nameserver changes at the parent. When that happens, the domain may be in dispute. An authenticated out-of-band and secure notify mechanism to contact a parent is needed in this case.

Note that this is only a problem when the DNSKEY and or DS records are used for authentication at the parent.

4.3.1.2. Breaking the Chain of Trust

There are two methods to break the chain of trust. The first method causes the child zone to appear 'Bogus' to validating resolvers. The other causes the child zone to appear 'insecure'. These are described below.

In the method that causes the child zone to appear 'Bogus' to validating resolvers, the child zone replaces the current KSK with a new one and re-signs the key set. Next it sends the DS of the new key to the parent. Only after the parent has placed the new DS in the zone is the child's chain of trust repaired.

An alternative method of breaking the chain of trust is by removing the DS RRs from the parent zone altogether. As a result, the child zone would become insecure.

4.3.2. ZSK Compromise

Primarily because there is no parental interaction required when a ZSK is compromised, the situation is less severe than with a KSK compromise. The zone must still be re-signed with a new ZSK as soon as possible. As this is a local operation and requires no communication between the parent and child, this can be achieved fairly quickly. However, one has to take into account that just as

with a normal rollover the immediate disappearance of the old compromised key may lead to verification problems. Also note that as long as the RRSIG over the compromised ZSK is not expired the zone may be still at risk.

4.3.3. Compromises of Keys Anchored in Resolvers

A key can also be pre-configured in resolvers. For instance, if DNSSEC is successfully deployed the root key may be pre-configured in most security aware resolvers.

If trust-anchor keys are compromised, the resolvers using these keys should be notified of this fact. Zone administrators may consider setting up a mailing list to communicate the fact that a SEP key is about to be rolled over. This communication will of course need to be authenticated, e.g., by using digital signatures.

End-users faced with the task of updating an anchored key should always validate the new key. New keys should be authenticated out-of-band, for example, through the use of an announcement website that is secured using secure sockets (TLS) [21].

4.4. Parental Policies

4.4.1. Initial Key Exchanges and Parental Policies Considerations

The initial key exchange is always subject to the policies set by the parent. When designing a key exchange policy one should take into account that the authentication and authorization mechanisms used during a key exchange should be as strong as the authentication and authorization mechanisms used for the exchange of delegation information between parent and child. That is, there is no implicit need in DNSSEC to make the authentication process stronger than it was in DNS.

Using the DNS itself as the source for the actual DNSKEY material, with an out-of-band check on the validity of the DNSKEY, has the benefit that it reduces the chances of user error. A DNSKEY query tool can make use of the SEP bit [3] to select the proper key from a DNSSEC key set, thereby reducing the chance that the wrong DNSKEY is sent. It can validate the self-signature over a key; thereby verifying the ownership of the private key material. Fetching the DNSKEY from the DNS ensures that the chain of trust remains intact once the parent publishes the DS RR indicating the child is secure.

Note: the out-of-band verification is still needed when the key material is fetched via the DNS. The parent can never be sure whether or not the DNSKEY RRs have been spoofed.

4.4.2. Storing Keys or Hashes?

When designing a registry system one should consider which of the DNSKEYs and/or the corresponding DSes to store. Since a child zone might wish to have a DS published using a message digest algorithm not yet understood by the registry, the registry can't count on being able to generate the DS record from a raw DNSKEY. Thus, we recommend that registry systems at least support storing DS records.

It may also be useful to store DNSKEYs, since having them may help during troubleshooting and, as long as the child's chosen message digest is supported, the overhead of generating DS records from them is minimal. Having an out-of-band mechanism, such as a registry directory (e.g., Whois), to find out which keys are used to generate DS Resource Records for specific owners and/or zones may also help with troubleshooting.

The storage considerations also relate to the design of the customer interface and the method by which data is transferred between registrant and registry; Will the child zone administrator be able to upload DS RRs with unknown hash algorithms or does the interface only allow DNSKEYs? In the registry-registrar model, one can use the DNSSEC extensions to the Extensible Provisioning Protocol (EPP) [15], which allows transfer of DS RRs and optionally DNSKEY RRs.

4.4.3. Security Lameness

Security lameness is defined as what happens when a parent has a DS RR pointing to a non-existing DNSKEY RR. When this happens, the child's zone may be marked "Bogus" by verifying DNS clients.

As part of a comprehensive delegation check, the parent could, at key exchange time, verify that the child's key is actually configured in the DNS. However, if a parent does not understand the hashing algorithm used by child, the parental checks are limited to only comparing the key id.

Child zones should be very careful in removing DNSKEY material, specifically SEP keys, for which a DS RR exists.

Once a zone is "security lame", a fix (e.g., removing a DS RR) will take time to propagate through the DNS.

4.4.4. DS Signature Validity Period

Since the DS can be replayed as long as it has a valid signature, a short signature validity period over the DS minimizes the time a child is vulnerable in the case of a compromise of the child's KSK(s). A signature validity period that is too short introduces the possibility that a zone is marked "Bogus" in case of a configuration error in the signer. There may not be enough time to fix the problems before signatures expire. Something as mundane as operator unavailability during weekends shows the need for DS signature validity periods longer than 2 days. We recommend an absolute minimum for a DS signature validity period of a few days.

The maximum signature validity period of the DS record depends on how long child zones are willing to be vulnerable after a key compromise. On the other hand, shortening the DS signature validity interval increases the operational risk for the parent. Therefore, the parent may have policy to use a signature validity interval that is considerably longer than the child would hope for.

A compromise between the operational constraints of the parent and minimizing damage for the child may result in a DS signature validity period somewhere between a week and months.

In addition to the signature validity period, which sets a lower bound on the number of times the zone owner will need to sign the zone data and which sets an upper bound to the time a child is vulnerable after key compromise, there is the TTL value on the DS RRs. Shortening the TTL means that the authoritative servers will see more queries. But on the other hand, a short TTL lowers the persistence of DS RRSets in caches thereby increasing the speed with which updated DS RRSets propagate through the DNS.

5. Security Considerations

DNSSEC adds data integrity to the DNS. This document tries to assess the operational considerations to maintain a stable and secure DNSSEC service. Not taking into account the 'data propagation' properties in the DNS will cause validation failures and may make secured zones unavailable to security-aware resolvers.

6. Acknowledgments

Most of the ideas in this document were the result of collective efforts during workshops, discussions, and tryouts.

At the risk of forgetting individuals who were the original contributors of the ideas, we would like to acknowledge people who

were actively involved in the compilation of this document. In random order: Rip Loomis, Olafur Gudmundsson, Wesley Griffin, Michael Richardson, Scott Rose, Rick van Rein, Tim McGinnis, Gilles Guette Olivier Courtay, Sam Weiler, Jelte Jansen, Niall O'Reilly, Holger Zuleger, Ed Lewis, Hilarie Orman, Marcos Sanz, and Peter Koch.

Some material in this document has been copied from RFC 2541 [12].

Mike StJohns designed the key exchange between parent and child mentioned in the last paragraph of Section 4.2.2

Section 4.2.4 was supplied by G. Guette and O. Courtay.

Emma Bretherick, Adrian Bedford, and Lindy Foster corrected many of the spelling and style issues.

Kolkman and Gieben take the blame for introducing all miscakes (sic).

While working on this document, Kolkman was employed by the RIPE NCC and Gieben was employed by NLnet Labs.

7. References

7.1. Normative References

[1] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[2] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[3] Kolkman, O., Schlyter, J., and E. Lewis, "Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag", RFC 3757, May 2004.

[4] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[5] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[6] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

7.2. Informative References

[7] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[8] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, August 1996.

[9] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, August 1996.

[10] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, November 2000.

[11] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, March 1998.

[12] Eastlake, D., "DNS Security Operational Considerations", RFC 2541, March 1999.

[13] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, April 2004.

[14] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[15] Hollenbeck, S., "Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)", RFC 4310, December 2005.

[16] Lenstra, A. and E. Verheul, "Selecting Cryptographic Key Sizes", The Journal of Cryptology 14 (255-293), 2001.

[17] Schneier, B., "Applied Cryptography: Protocols, Algorithms, and Source Code in C", ISBN (hardcover) 0-471-12845-7, ISBN (paperback) 0-471-59756-2, Published by John Wiley & Sons Inc., 1996.

[18] Rose, S., "NIST DNSSEC workshop notes", June 2001.

[19] Jansen, J., "Use of RSA/SHA-256 DNSKEY and RRSIG Resource Records in DNSSEC", Work in Progress, January 2006.

[20] Hardaker, W., "Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records (RRs)", RFC 4509, May 2006.

[21] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 4366, April 2006.

Appendix A. Terminology

In this document, there is some jargon used that is defined in other documents. In most cases, we have not copied the text from the documents defining the terms but have given a more elaborate explanation of the meaning. Note that these explanations should not be seen as authoritative.

Anchored key: A DNSKEY configured in resolvers around the globe. This key is hard to update, hence the term anchored.

Bogus: Also see Section 5 of [4]. An RRSet in DNSSEC is marked "Bogus" when a signature of an RRSet does not validate against a DNSKEY.

Key Signing Key or KSK: A Key Signing Key (KSK) is a key that is used exclusively for signing the apex key set. The fact that a key is a KSK is only relevant to the signing tool.

Key size: The term 'key size' can be substituted by 'modulus size' throughout the document. It is mathematically more correct to use modulus size, but as this is a document directed at operators we feel more at ease with the term key size.

Private and public keys: DNSSEC secures the DNS through the use of public key cryptography. Public key cryptography is based on the existence of two (mathematically related) keys, a public key and a private key. The public keys are published in the DNS by use of the DNSKEY Resource Record (DNSKEY RR). Private keys should remain private.

Key rollover: A key rollover (also called key supercession in some environments) is the act of replacing one key pair with another at the end of a key effectivity period.

Secure Entry Point (SEP) key: A KSK that has a parental DS record pointing to it or is configured as a trust anchor. Although not required by the protocol, we recommend that the SEP flag [3] is set on these keys.

Self-signature: This only applies to signatures over DNSKEYs; a signature made with DNSKEY x, over DNSKEY x is called a self-signature. Note: without further information, self-signatures convey no trust. They are useful to check the authenticity of the DNSKEY, i.e., they can be used as a hash.

Singing the zone file: The term used for the event where an administrator joyfully signs its zone file while producing melodic sound patterns.

Signer: The system that has access to the private key material and signs the Resource Record sets in a zone. A signer may be configured to sign only parts of the zone, e.g., only those RRSets for which existing signatures are about to expire.

Zone Signing Key (ZSK): A key that is used for signing all data in a zone. The fact that a key is a ZSK is only relevant to the signing tool.

Zone administrator: The 'role' that is responsible for signing a zone and publishing it on the primary authoritative server.

Appendix B. Zone Signing Key Rollover How-To

Using the pre-published signature scheme and the most conservative method to assure oneself that data does not live in caches, here follows the "how-to".

Step 0: The preparation: Create two keys and publish both in your key set. Mark one of the keys "active" and the other "published". Use the "active" key for signing your zone data. Store the private part of the "published" key, preferably off-line. The protocol does not provide for attributes to mark a key as active or published. This is something you have to do on your own, through the use of a notebook or key management tool.

Step 1: Determine expiration: At the beginning of the rollover make a note of the highest expiration time of signatures in your zone file created with the current key marked as active. Wait until the expiration time marked in Step 1 has passed.

Step 2: Then start using the key that was marked "published" to sign your data (i.e., mark it "active"). Stop using the key that was marked "active"; mark it "rolled".

Step 3: It is safe to engage in a new rollover (Step 1) after at least one signature validity period.

Appendix C. Typographic Conventions

The following typographic conventions are used in this document:

Key notation: A key is denoted by DNSKEYx, where x is a number or an identifier, x could be thought of as the key id.

RRSet notations: RRs are only denoted by the type. All other information -- owner, class, rdata, and TTL--is left out. Thus: "example.com 3600 IN A 192.0.2.1" is reduced to "A". RRSets are a list of RRs. A example of this would be "A1, A2", specifying the RRSet containing two "A" records. This could again be abbreviated to just "A".

Signature notation: Signatures are denoted as RRSIGx(RRSet), which means that RRSet is signed with DNSKEYx.

Zone representation: Using the above notation we have simplified the representation of a signed zone by leaving out all unnecessary details such as the names and by representing all data by "SOAx"

SOA representation: SOAs are represented as SOAx, where x is the serial number.

Using this notation the following signed zone:

```
example.net.      86400  IN SOA  ns.example.net. bert.example.net. (
                         2006022100   ; serial
                         86400        ; refresh (  24 hours)
                         7200         ; retry   (   2 hours)
                         3600000      ; expire  (1000 hours)
                         28800 )      ; minimum (   8 hours)
                  86400  RRSIG   SOA 5 2 86400 20130522213204 (
                               20130422213204 14 example.net.
                               cmL62SI6iAX46xGNQAdQ... )
                  86400  NS      a.iana-servers.net.
                  86400  NS      b.iana-servers.net.
                  86400  RRSIG   NS 5 2 86400 20130507213204 (
                               20130407213204 14 example.net.
                               SO5epiJei19AjXoUpFnQ ... )
                  86400  DNSKEY  256 3 5 (
                               EtRB9MP5/AvOuVO0I8XDxy0... ) ; id = 14
                  86400  DNSKEY  257 3 5 (
                               gsPW/Yy19GzYIY+Gnr8HABU... ) ; id = 15
                  86400  RRSIG   DNSKEY 5 2 86400 20130522213204 (
                               20130422213204 14 example.net.
                               J4zCe8QX4tXVGjV4e1r9... )
```

```
                    86400  RRSIG   DNSKEY 5 2 86400 20130522213204 (
                                   20130422213204 15 example.net.
                                   keVDCOpsSeDReyV6O... )
                    86400  RRSIG   NSEC 5 2 86400 20130507213204 (
                                   20130407213204 14 example.net.
                                   obj3HEp1GjnmhRjX... )
a.example.net.      86400  IN TXT  "A label"
                    86400  RRSIG   TXT 5 3 86400 20130507213204 (
                                   20130407213204 14 example.net.
                                   IkDMlRdYLmXH7QJnuF3v... )
                    86400  NSEC    b.example.com. TXT RRSIG NSEC
                    86400  RRSIG   NSEC 5 3 86400 20130507213204 (
                                   20130407213204 14 example.net.
                                   bZMjoZ3bHjnEz0nIsPMM... )
                    ...
```

is reduced to the following representation:

```
    SOA2006022100
    RRSIG14(SOA2006022100)
    DNSKEY14
    DNSKEY15

    RRSIG14(KEY)
    RRSIG15(KEY)
```

The rest of the zone data has the same signature as the SOA record, i.e., an RRSIG created with DNSKEY 14.

Authors' Addresses

Olaf M. Kolkman
NLnet Labs
Kruislaan 419
Amsterdam 1098 VA
The Netherlands

EMail: olaf@nlnetlabs.nl
URI: http://www.nlnetlabs.nl

R. (Miek) Gieben

EMail: miek@miek.nl

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).