DNSOP
Internet-Draft
Expires: September 2, 2005

O. Kolkman
RIPE NCC
R. Gieben
NLnet Labs
March 2005

DNSSEC Operational Practices
draft-ietf-dnsop-dnssec-operational-practices-04.txt

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 as key generation, key storage, signature generation, key rollover and related policies.

Table of Contents

1. Introduction 4
1.1 The Use of the Term 'key' 4
1.2 Time Definitions 5
2. Keeping the Chain of Trust Intact 5
3. Keys Generation and Storage 6
3.1 Zone and Key Signing Keys 6
3.1.1 Motivations for the KSK and ZSK Separation 6
3.1.2 KSKs for high level zones 7
3.2 Randomness 8
3.3 Key Effectivity Period 8
3.4 Key Algorithm 9
3.5 Key Sizes 9
3.6 Private Key Storage 10
4. Signature generation, Key Rollover and Related Policies . . . 11
4.1 Time in DNSSEC 11
4.1.1 Time Considerations 11
4.2 Key Rollovers 13
4.2.1 Zone-signing Key Rollovers 13
4.2.2 Key-signing Key Rollovers 17
4.2.3 Difference Between ZSK and KSK Rollovers 18
4.2.4 Automated Key Rollovers 19
4.3 Planning for Emergency Key Rollover 19
4.3.1 KSK Compromise 20
4.3.2 ZSK Compromise 20
4.3.3 Compromises of Keys Anchored in Resolvers 20
4.4 Parental Policies 21
4.4.1 Initial Key Exchanges and Parental Policies Considerations 21
4.4.2 Storing Keys or Hashes? 21
4.4.3 Security Lameness 22
4.4.4 DS Signature Validity Period 22
5. Security Considerations 23
6. Acknowledgments 23
7. References 24
7.1 Normative References 24
7.2 Informative References 24
Authors' Addresses 25
A. Terminology 25
B. Zone-signing Key Rollover Howto 26
C. Typographic Conventions 26
D. Document Details and Changes 29
D.1 draft-ietf-dnsop-dnssec-operational-practices-00 29
D.2 draft-ietf-dnsop-dnssec-operational-practices-01 29
D.3 draft-ietf-dnsop-dnssec-operational-practices-02 29
D.4 draft-ietf-dnsop-dnssec-operational-practices-03 29
D.5 draft-ietf-dnsop-dnssec-operational-practices-04 30

Intellectual Property and Copyright Statements 31

1. Introduction

During workshops and early operational deployment tests, operators and system administrators 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 resigning 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 which, 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 RFC2119 [4] language does not apply.

This document obsoletes RFC2541 [7]

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 [11]). 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 stopped 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 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.
  The key effectivity period can span multiple signature validity periods.
- "Maximum/Minimum Zone TTL"
  The maximum or minimum value of the TTLs from the complete set of RRs in a zone.

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 [2] 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 their clients, part of a chain of trust.

As mentioned in the introduction, the procedures herein are intended to ensure maintenance of zones, such as resigning 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 transfered to other secondary authoritative nameservers and clients may be fetching data from caching non-authoritative servers.

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 assure 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 to deliberately break the chain of trust and making secured sub domains 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 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 flag to distinguish between them during operations. The dynamics and considerations are discussed below.

To make zone resigning and key rollover procedures easier to implement, it is possible to use one or more keys as Key Signing Keys (KSK). These keys will only sign the apex DNSKEY RR set in a zone. Other keys can be used to sign all the RRsets in a zone and are referred to as Zone Signing Keys (ZSK). 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 [1] 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:

- o No parent/child interaction is required when ZSKs are updated.
- o 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 when verifying the KSK is only used to verify the zone's keyset.
- o 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.
- o 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 ZSK. If a ZSK is compromised, it can be simply dropped from the key set. The new key set is then resigned 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 trusted 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 Usage Time can be on the order of years we suggest planning for a key effectivity of 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 sub domains (except in the case of resolvers that have locally configured the public key of a sub domain). Therefore, extra care should be taken with high level zones and strong keys 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 name space of all resolvers using that root zone (except in the case of resolvers that have locally configured the public key of a sub domain). 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 Randomness

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 RFC1750 [3]. 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.

For Key Signing Keys a reasonable key effectivity period 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.

Using these recommendations will lead to rollovers occurring frequently enough to become part of 'operational habits'; the procedure does not have to be reinvented every time a key is

replaced.

Key effectivity periods can be made very short, as in the order of 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 still needs 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 NIST. The creation of signatures is roughly done at the same speed as with RSA, but is 10 to 40 times as slow for verification [11].

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 (theoretical) cracks, we recommend the usage of SHA1.

In 2005 some discoveries were made that SHA-1 also has some weaknesses. Currently SHA-1 is strong enough for DNSSEC. It is expected that a new hashing algorithm is rolled out, before any attack becomes practical.

3.5 Key Sizes

When choosing key sizes, zone administrators will need to take into account how long a key will be used and how much data will be signed during the key publication period. It is hard to give precise recommendations but Lenstra and Verheul [10] supplied the following table with lower bound estimates for cryptographic key sizes. Their recommendations are based on a set of explicitly formulated parameter settings, combined with existing data points about cryptographic systems. For details we refer to the original paper.

Year	RSA Key Sizes	Year	RSA Key Sizes
2000	952	2015	1613
2001	990	2016	1664
2002	1028	2017	1717
2003	1068	2018	1771
2004	1108	2019	1825
2005	1149	2020	1881
2006	1191	2021	1937
2007	1235	2022	1995
2008	1279	2023	2054
2009	1323	2024	2113
2026	2236	2025	2174
2010	1369	2027	2299
2011	1416	2028	2362
2012	1464	2029	2427
2013	1513		
2014	1562		

For example, should you wish your key to last three years from 2003, check the RSA key size values for 2006 in this table. In this case it should be at least 1191 bits.

Please keep in mind that nobody can see into the future, and that these key lengths are only provided here as a guide.

When determining a key size one should take into account that a large key will be slower during generation and verification. For RSA, verification, the most common operation, will vary roughly with the square of the key size; signing will vary with the cube of the key size length; and key generation will vary with the fourth power of the modulus length. Besides larger keys will increase the sizes of the RRSIG and DNSKEY records and will therefore increase the chance of DNS UDP packet overflow. Also see Section 3.1.1 for a discussion of how keys serving different roles (ZSK v. KSK) may need different key strengths.

3.6 Private Key Storage

It is recommended that, where possible, zone private keys and the zone file master copy 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,

perhaps by sneaker-net, to the networked zone primary server machine.

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 [5] 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 DNS are relative. The SOA RR's refresh, retry and expiration timers are counters that are used to determine the time elapsed after a slave server synchronized (or tried to synchronize) with a master server. The Time to Live (TTL) value and the SOA RR minimum TTL parameter [6] 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 [2] 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 be at least one maximum TTL smaller than the signature validity period.
  Resigning 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 authentication chain. A low TTL could cause two problems:
  1. During validation, some data may expire before the validation is complete. The validator should be able to keep all data, until is completed. This applies to all RRs needed to complete the chain of trust: DSs, DNSKEYs, RRSIGs, and the final answers i.e. the RR set that is returned for the initial query.
  2. Frequent verification causes load on recursive nameservers. Data at delegation points, DSs, 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 zone will not respond on queries. The time of expiration is set in the SOA record and is relative to the last successful refresh between the master and the slave server.
  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 than 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 or smaller 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 time 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 secondary zones expire; How quickly can I reach an administrator of secondary servers to load a valid zone? All these arguments 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 is when zone material signed with an old key is being validated by a resolver which 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 which 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 set Rollover

This section shows how to perform a ZSK rollover without the need to sign all the data in a zone twice - the so-called "pre-publish

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 "HOWTO" for this kind of rollover can be found in Appendix B.

normal	pre-roll	roll	after
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)

normal: Version 0 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.

pre-roll: 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. This equates to two times the Maximum Zone TTL.

roll: At the rollover 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.

after: DNSKEY 10 is removed from the zone. The key set, now only containing DNSKEY 1 and DNSKEY 11 is resigned 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 "after" as DNSKEY 12 and again a newer one, numbered 13, in "2nd after":

normal	roll	after
SOA0	SOA2	SOA3
RRSIG10(SOA0)	RRSIG11(SOA2)	RRSIG11(SOA3)
DNSKEY1	DNSKEY1	DNSKEY1
DNSKEY10	DNSKEY10	DNSKEY11
DNSKEY11	DNSKEY11	DNSKEY12
RRSIG1(DNSKEY)	RRSIG1 (DNSKEY)	RRSIG1(DNSKEY)
RRSIG10(DNSKEY)	RRSIG11(DNSKEY)	RRSIG11(DNSKEY)

2nd roll	2nd after
SOA4	SOA5
RRSIG12(SOA4)	RRSIG12(SOA5)
DNSKEY1	DNSKEY1
DNSKEY11	DNSKEY12
DNSKEY12	DNSKEY13
RRSIG1(DNSKEY)	RRSIG1(DNSKEY)
RRSIG12(DNSKEY)	RRSIG12(DNSKEY)

Note that the key introduced after the rollover 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 sig rollover".

During the rollover 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.

normal	roll	after
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)	

normal: Version 0 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.

roll: At the rollover 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 exist 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.

after: 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 resigned 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 rollover phase and the period between rollovers should be at least the "Maximum Zone TTL".

Making sure that the rollover phase lasts until the signature expiration time of the data in version 0 of the zone is recommended. This way all caches are cleared of the old signatures. However, this date 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 set 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 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.

Double signature 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.

normal	roll	after
SOA0	SOA1	SOA2
RRSIG10(SOA0)	RRSIG10(SOA1)	RRSIG10(SOA2)
DNSKEY1	DNSKEY1	DNSKEY2
	DNSKEY2	
DNSKEY10	DNSKEY10	DNSKEY10
RRSIG1 (DNSKEY)	RRSIG1 (DNSKEY)	RRSIG2(DNSKEY)
	RRSIG2 (DNSKEY)	
RRSIG10(DNSKEY)	RRSIG10(DNSKEY)	RRSIG10(DNSKEY)

normal: Version 0 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.

roll: During the rollover 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.

after: 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 premises 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. A zone-key rollover can be handled in two different ways: pre-publish (Section Section 4.2.1.1) and double signature (Section 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 work for a KSK rollover. The record that are to be pre-published are the parental DS RRs.

The pre-publish method has some drawbacks. We first describe the rollover scheme and then indicate these drawbacks.

normal	pre-roll	roll	after
Parent:			
SOA0	SOA1	SOA2	SOA3
RRSIGpar(SOA0)	RRSIGpar(SOA1)	RRSIGpar(SOA2)	RRSIGpar(SOA3)
DS1	DS1	DS1	DS2
	DS2	DS2	
RRSIGpar(DS)	RRSIGpar(DS)	RRSIGpar(DS)	RRSIGpar(DS)
Child:			
SOA0	SOA0	SOA1	SOA1
RRSIG10(SOA0)	RRSIG10(SOA0)	RRSIG10(SOA1)	RRSIG10(SOA1)
DNSKEY1	DNSKEY1	DNSKEY2	DNSKEY2
DNSKEY10	DNSKEY10	DNSKEY10	DNSKEY10
RRSIG1 (DNSKEY)	RRSIG1 (DNSKEY)	RRSIG2(DNSKEY)	RRSIG2 (DNSKEY)
RRSIG10(DNSKEY)	RRSIG10(DNSKEY)	RRSIG10(DNSKEY)	RRSIG10(DNSKEY)

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

The drawbacks of these scheme are that during the pre-roll phase the parent cannot verify the match between the DS RR and DNSKEY2 using the DNS. Besides, we introduce a "security lame" DS record 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 that:

- ZSK rollovers are easy to automate as only the local zone is involved.
- A KSK rollover needs interaction between the 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.
- All time and TTL considerations presented in Section 4.2 apply to an automated 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 authentication chain exists. An authentication chain remains intact for:

- as long as a signature over the compromised key in the authentication 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 an authentication chain to your compromised key exists, your name-space 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 authentication 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 roll-over, the malicious key holder can spoof data so that it appears to be valid. Note that this kind of attack is more likely to occur in a localized part of the network topology i.e. downstream from where the spoof takes place.

4.3.1 KSK Compromise

When the KSK has been compromised the parent must be notified as soon as possible using secure means. The key set of the zone should be resigned as soon as possible. Care must be taken to not break the authentication chain. The local zone can only be resigned with the new KSK after the parent's zone has created and reloaded its zone with the DS created from the new KSK. Before this update takes place it would be best to drop the security status of a zone all together: the parent removes the DS of the child at the next zone update. After that the child can be made secure again.

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

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 with a KSK compromise. The zone must still be resigned 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 from the old compromised key may lead to verification problems. The pre-publication scheme as discussed above minimizes such problems.

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 the DNS, for example, looking them up on an SSL secured announcement website.

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 (or its registry). 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. I.e. 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 off-band check on the validity of the DNSKEY, has the benefit that it reduces the chances of user error. A parental DNSKEY download tool can make use of the SEP bit [1] 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 off-band verification is still needed when the key-material is fetched via the DNS. The parent can never be sure whether the DNSKEY RRs have been spoofed or not.

4.4.2 Storing Keys or Hashes?

When designing a registry system one should consider which of the DNSKEYs and/or the corresponding DSs 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 system at least support storing DS records.

It may also be useful to store DNSKEYs, since having them may help

during troubleshooting and, so 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 Whois database, 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 the design of the customer interface and the method by which data is transfered between registrant and registry; Will the child zone owner be able to upload DS RRs with unknown hash algorithms or does the interface only allows DNSKEYs? In the registry-registrar model one can use the DNSSEC EPP protocol extensions [9] 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. During key exchange a parent should make sure that the child's key is actually configured in the DNS before publishing a DS RR in its zone. Failure to do so could cause the child's zone being marked as Bogus.

Child zones should be very careful 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 the 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. Other considerations, such as how often the zone is (re)signed can also be taken into account.

We consider a signature validity period of around one week to be a

good compromise between the operational constraints of the parent and minimizing damage for the child.

In addition to the signature validity period, which sets a lower bound on the amount 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. By lowering the TTL, the authoritative servers will see more queries, on the other hand a low TTL increases the speed with which new DS RRs propagate through the DNS. As argued in Section 4.1.1, the TTL should be a fraction of the signature validity period.

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 draft were the result of collective efforts during workshops, discussions and try outs.

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 and Niall O'Reilly.

Some material in this document has been shamelessly copied from RFC2541 [7] by Donald Eastlake.

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).

7. References

7.1 Normative References

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

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

7.2 Informative References

[3] Eastlake, D., Crocker, S., and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

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

[5] Eastlake, D., "Secure Domain Name System Dynamic Update", RFC 2137, April 1997.

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

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

[8] Gudmundsson, O., "Delegation Signer (DS) Resource Record (RR)", RFC 3658, December 2003.

[9] Hollenbeck, S., "Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)", draft-hollenbeck-epp-secdns-07 (work in progress), March 2005.

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

[11] Schneier, B., "Applied Cryptography: Protocols, Algorithms, and Source Code in C", 1996.

Authors' Addresses

Olaf M. Kolkman
RIPE NCC
Singel 256
Amsterdam 1016 AB
The Netherlands

Phone: +31 20 535 4444
Email: olaf@ripe.net
URI: http://www.ripe.net/

Miek Gieben
NLnet Labs
Kruislaan 419
Amsterdam 1098 VA
The Netherlands

Email: miek@nlnetlabs.nl
URI: http://www.nlnetlabs.nl

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 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 [2]. An RRset in DNSSEC is marked "Bogus" when a signature of a 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.

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 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 by another at the end of a key effectivity period.

Secure Entry Point key or SEP Key: A KSK that has a parental DS record pointing to it. Note: this is not enforced in the protocol. A SEP Key with no parental DS is security lame.
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 or ZSK: A Zone Signing Key (ZSK) is 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 Howto

Using the pre-published signature scheme and the most conservative method to assure oneself that data does not live in caches here follows the "HOWTO".
Step 0: The preparation: Create two keys and publish both in your key set. Mark one of the keys as "active" and the other as "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 as "published" to sign your data i.e. mark it as "active". Stop using the key that was marked as "active", mark it as "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 KEYx, where x is a number, 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 192.168.1.1" 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: SOA's are represented as SOAx, where x is the serial number.

Using this notation the following zone:

```
example.net.     600     IN SOA  ns.example.net. bert.example.net. (
                                 10         ; serial
                                 450        ; refresh (7 minutes 30 seconds)
                                 600        ; retry (10 minutes)
                                 345600     ; expire (4 days)
                                 300        ; minimum (5 minutes)
                                 )
                 600     RRSIG   SOA 5 2 600 20130522213204 (
                                 20130422213204 14 example.net.
                                 cmL62SI6iAX46xGNQAdQ... )
                 600     NS      a.iana-servers.net.
                 600     NS      b.iana-servers.net.
                 600     RRSIG   NS 5 2 600 20130507213204 (
                                 20130407213204 14 example.net.
                                 SO5epiJei19AjXoUpFnQ ... )
                 3600    DNSKEY  256 3 5 (
                                 EtRB9MP5/AvOuVO0I8XDxy0...
                                 ) ; key id = 14
                 3600    DNSKEY  256 3 5 (
                                 gsPW/Yy19GzYIY+Gnr8HABU...
                                 ) ; key id = 15
                 3600    RRSIG   DNSKEY 5 2 3600 20130522213204 (
                                 20130422213204 14 example.net.
                                 J4zCe8QX4tXVGjV4e1r9... )
                 3600    RRSIG   DNSKEY 5 2 3600 20130522213204 (
                                 20130422213204 15 example.net.
                                 keVDCOpsSeDReyV6O... )
                 600     RRSIG   NSEC 5 2 600 20130507213204 (
                                 20130407213204 14 example.net.
                                 obj3HEp1GjnmhRjX... )
a.example.net.   600     IN TXT  "A label"
                 600     RRSIG   TXT 5 3 600 20130507213204 (
                                 20130407213204 14 example.net.
                                 IkDMlRdYLmXH7QJnuF3v... )
                 600     NSEC    b.example.com. TXT RRSIG NSEC
                 600     RRSIG   NSEC 5 3 600 20130507213204 (
                                 20130407213204 14 example.net.
                                 bZMjoZ3bHjnEz0nIsPMM... )

                 ...
```

is reduced to the following representation:

```
SOA10
RRSIG14(SOA10)

DNSKEY14
DNSKEY15

RRSIG14(KEY)
RRSIG15(KEY)
```

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

Appendix D. Document Details and Changes

This section is to be removed by the RFC editor if and when the document is published.

$Id: draft-ietf-dnsop-dnssec-operational-practices.xml,v 1.31.2.14 2005/03/21 15:51:41 dnssec Exp $

D.1 draft-ietf-dnsop-dnssec-operational-practices-00

Submission as working group document. This document is a modified and updated version of draft-kolkman-dnssec-operational-practices-00.

D.2 draft-ietf-dnsop-dnssec-operational-practices-01

changed the definition of "Bogus" to reflect the one in the protocol draft.

Bad to Bogus

Style and spelling corrections

KSK - SEP mapping made explicit.

Updates from Sam Weiler added

D.3 draft-ietf-dnsop-dnssec-operational-practices-02

Style and errors corrected.

Added Automatic rollover requirements from I-D.ietf-dnsop-key-rollover-requirements.

D.4 draft-ietf-dnsop-dnssec-operational-practices-03

Added the definition of Key effectivity period and used that term

instead of Key validity period.

Modified the order of the sections, based on a suggestion by Rip Loomis.

Included parts from RFC2541 [7]. Most of its ground was already covered. This document obsoletes RFC2541 [7]. Section 3.1.2 deserves some review as it in contrast to RFC2541 does _not_ give recomendations about root-zone keys.

added a paragraph to Section 4.4.4

D.5 draft-ietf-dnsop-dnssec-operational-practices-04

Somewhat more details added about the pre-publish KSK rollover. Also moved that subsection down a bit.

Editorial and content nits that came in during wg last call were fixed.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.