Network Working Group
Internet-Draft
Expires: April 23, 2006

J. Ihren
Autonomica AB
O. Kolkman
RIPE NCC
B. Manning
EP.net
October 23, 2005

An In-Band Rollover Mechanism and an Out-Of-Band Priming Method for DNSSEC Trust Anchors. draft-ietf-dnsext-trustupdate-threshold-01

Status of this Memo

This Internet-Draft will expire on April 23, 2006.

Copyright Notice

Abstract

The DNS Security Extensions (DNSSEC) works by validating so called chains of authority. The start of these chains of authority are usually public keys that are anchored in the DNS clients. These keys are known as the so called trust anchors.

This memo describes a method how these client trust anchors can be replaced using the DNS validation and querying mechanisms (in-band) when the key pairs used for signing by zone owner are rolled.

This memo also describes a method to establish the validity of trust anchors for initial configuration, or priming, using out of band mechanisms.

Table of Contents

1. Terminology

The key words "MUST", "SHALL", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in RFC2119 [1].

The term "zone" refers to the unit of administrative control in the Domain Name System. In this document "name server" denotes a DNS name server that is authoritative (i.e. knows all there is to know) for a DNS zone. A "zone owner" is the entity responsible for signing and publishing a zone on a name server. The terms "authentication chain", "bogus", "trust anchors" and "Island of Security" are defined in [4]. Throughout this document we use the term "resolver" to mean "Validating Stub Resolvers" as defined in [4].

We use the term "security apex" as the zone for which a trust anchor has been configured (by validating clients) and which is therefore, by definition, at the root of an island of security. The configuration of trust anchors is a client side issue. Therefore a zone owner may not always know if their zone has become a security apex.

A "stale anchor" is a trust anchor (a public key) that relates to a key that is not used for signing. Since trust anchors indicate that a zone is supposed to be secure a validator will mark the all data in an island of security as bogus when all trust anchors become stale.

It is assumed that the reader is familiar with public key cryptography concepts [REF: Schneier Applied Cryptography] and is able to distinguish between the private and public parts of a key based on the context in which we use the term "key". If there is a possible ambiguity we will explicitly mention if a private or a public part of a key is used.

The term "administrator" is used loosely throughout the text. In some cases an administrator is meant to be a person, in other cases the administrator may be a process that has been delegated certain responsibilities.

1.1 Key Signing Keys, Zone Signing Keys and Secure Entry Points

Although the DNSSEC protocol does not make a distinction between different keys the operational practice is that a distinction is made between zone signing keys and key signing keys. A key signing key is used to exclusively sign the DNSKEY Resource Record (RR) set at the apex of a zone and the zone signing keys sign all the data in the zone (including the DNSKEY RRset at the apex).

Keys that are intended to be used as the start of the authentication chain for a particular zone, either because they are pointed to by a parental DS RR or because they are configured as a trust anchor, are called Secure Entry Point (SEP) keys. In practice these SEP keys will be key signing keys.

In order for the mechanism described herein to work the keys that are intended to be used as secure entry points MUST have the SEP [2] flag set. In the examples it is assumed that keys with the SEP flag set are used as key signing keys and thus exclusively sign the DNSKEY RRset published at the apex of the zone.

2. Introduction and Background

When DNSSEC signatures are validated the resolver constructs a chain of authority from a pre-configured trust anchor to the DNSKEY Resource Record (RR), which contains the public key that validates the signature stored in an RRSIG RR. DNSSEC is designed so that the administrator of a resolver can validate data in multiple islands of security by configuring multiple trust anchors.

It is expected that resolvers will have more than one trust anchor configured. Although there is no deployment experience it is not unreasonable to expect resolvers to be configured with a number of trust anchors that varies between order 1 and order 1000. Because zone owners are expected to roll their keys, trust anchors will have to be maintained (in the resolver end) in order not to become stale.

Since there is no global key maintenance policy for zone owners and there are no mechanisms in the DNS to signal the key maintenance policy it may be very hard for resolvers administrators to keep their set of trust anchors up to date. For instance, if there is only one trust anchor configured and the key maintenance policy is clearly published, through some out of band trusted channel, then a resolver administrator can probably keep track of key rollovers and update the trust anchor manually. However, with an increasing number of trust anchors all rolled according to individual policies that are all published through different channels this soon becomes an unmanageable problem.

2.1 Dangers of Stale Trust Anchors

Whenever a SEP key at a security apex is rolled there exists a danger that "stale anchors" are created. A stale anchor is a trust anchor (i.e. a public key configured in a validating resolver) that relates to a private key that is no longer used for signing.

The problem with a stale anchors is that they will (from the validating resolvers point of view) prove data to be false even though it is actually correct. This is because the data is either signed by a new key or is no longer signed and the resolver expects data to be signed by the old (now stale) key.

This situation is arguably worse than not having a trusted key configured for the secure entry point, since with a stale key no lookup is typically possible (presuming that the default configuration of a validating recursive nameserver is to not give out data that is signed but failed to verify.

The danger of making configured trust anchors become stale anchors

may be a reason for zone owners not to roll their keys. If a resolver is configured with many trust anchors that need manual maintenance it may be easy to not notice a key rollover at a security apex, resulting in a stale anchor.

In Section 3 this memo sets out a lightweight, in-DNS, mechanism to track key rollovers and modify the configured trust anchors accordingly. The mechanism is stateless and does not need protocol extensions. The proposed design is that this mechanism is implemented as a "trust updating machine" that is run entirely separate from the validating resolver except that the trust updater will have influence over the trust anchors used by the latter.

In Section 4 we describe a method [Editors note: for now only the frame work and a set of requirements] to install trust anchors. This method can be used at first configuration or when the trust anchors became stale (typically due to a failure to track several rollover events).

The choice for which domains trust anchors are to be configured is a local policy issue. So is the choice which trust anchors has prevalence if there are multiple chains of trust to a given piece of DNS data (e.g. when a parent zone and its child both have trust anchors configured). Both issues are out of the scope of this document.

3. Threshold-based Trust Anchor Rollover

3.1 Why Rollover?

Any cryptographic system must be able to periodically replace the secret keys used. Reasons for this include everything from accidental compromise to cryptographic exposure through prolonged use. In this case we argue that the compromise case is the most crucial and difficult one, as the exposure can be managed through controlled, periodic rollovers while the compromise case causes either no rollover (because it isn't detected in time) or immediate emergency rollover.

Also it is crucial to note that for the compromise case a compromised key must not enable the attacker to do his own rollover into new keys under attacker control. Especially in the case of a compromise that isn't immediately detected this is important.

3.2 The Rollover

When a key pair is replaced all signatures (in DNSSEC these are the RRSIG records) created with the old key will be replaced by new signatures created by the new key. Access to the new public key is needed to verify these signatures.

Since zone signing keys are in "the middle" of a chain of authority they can be verified using the signature made by a key signing key. Rollover of zone signing keys is therefore transparent to validators and requires no action in the validator end.

But if a key signing key is rolled a resolver can determine its authenticity by either following the authorization chain from the parents DS record, an out-of-DNS authentication mechanism or by relying on other trust anchors known for the zone in which the key is rolled.

The threshold trust anchor rollover mechanism (or trust update), described below, is based on using existing trust anchors to verify a subset of the available signatures. This is then used as the basis for a decision to accept the new keys as valid trust anchors.

Our example pseudo zone below contains a number of key signing keys numbered 1 through Y and two zone signing keys A and B. During a key rollover key 2 is replaced by key Y+1. The zone content changes from:

```
example.com.  DNSKEY key1
example.com.  DNSKEY key2
example.com.  DNSKEY key3
```

```
        ...
        example.com.  DNSKEY keyY

        example.com.  DNSKEY keyA
        example.com.  DNSKEY keyB

        example.com.  RRSIG DNSKEY ... (key1)
        example.com.  RRSIG DNSKEY ... (key2)
        example.com.  RRSIG DNSKEY ... (key3)
        ...
        example.com.  RRSIG DNSKEY ... (keyY)
        example.com.  RRSIG DNSKEY ... (keyA)
        example.com.  RRSIG DNSKEY ... (keyB)
```

to:

```
example.com.  DNSKEY key1
example.com.  DNSKEY key3
...
example.com.  DNSKEY keyY
example.com.  DNSKEY keyY+1


example.com.  RRSIG DNSKEY ... (key1)
example.com.  RRSIG DNSKEY ... (key3)
...
example.com.  RRSIG DNSKEY ... (keyY)
example.com.  RRSIG DNSKEY ... (keyY+1)
example.com.  RRSIG DNSKEY ... (keyA)
example.com.  RRSIG DNSKEY ... (keyB)
```

When the rollover becomes visible to the verifying stub resolver it will be able to verify the RRSIGs associated with key1, key3 ... keyY. There will be no RRSIG by key2 and the RRSIG by keyY+1 will not be used for validation, since that key is previously unknown and therefore not trusted.

Note that this example is simplified. Because of operational considerations described in [5] having a period during which the two key signing keys are both available is necessary.

3.3 Threshold-based Trust Update

The threshold-based trust update algorithm applies as follows. If for a particular secure entry point

- if the DNSKEY RRset in the zone has been replaced by a more recent one (as determined by comparing the RRSIG inception dates)

and

- if at least M configured trust anchors directly verify the related RRSIGs over the new DNSKEY RRset

and

- the number of configured trust anchors that verify the related RRSIGs over the new DNSKEY RRset exceed a locally defined minimum number that should be greater than one

then all the trust anchors for the particular secure entry point are replaced by the set of keys from the zones DNSKEY RRset that have the SEP flag set.

The choices for the rollover acceptance policy parameter M is left to the administrator of the resolver. To be certain that a rollover is

accepted up by resolvers using this mechanism zone owners should roll as few SEP keys at a time as possible (preferably just one). That way they comply to the most strict rollover acceptance policy of M=N-1.

The value of M has an upper bound, limited by the number of of SEP keys a zone owner publishes (i.e. N). But there is also a lower bound, since it will not be safe to base the trust in too few signatures. The corner case is M=1 when any validating RRSIG will be sufficient for a complete replacement of the trust anchors for that secure entry point. This is not a recommended configuration, since that will allow an attacker to initiate rollover of the trust anchors himself given access to just one compromised key. Hence M should in be strictly larger than 1 as shown by the third requirement above.

If the rollover acceptance policy is M=1 then the result for the rollover in our example above should be that the local database of trust anchors is updated by removing key "key2" from and adding key "keyY+1" to the key store.

3.4 Possible Trust Update States

We define five states for trust anchor configuration at the client side.

PRIMING: There are no trust anchors configured. There may be priming keys available for initial priming of trust anchors.

IN-SYNC: The set of trust anchors configured exactly matches the set of SEP keys used by the zone owner to sign the zone.

OUT-OF-SYNC: The set of trust anchors is not exactly the same as the set of SEP keys used by the zone owner to sign the zone but there are enough SEP key in use by the zone owner that is also in the trust anchor configuration.

UNSYNCABLE: There is not enough overlap between the configured trust anchors and the set of SEP keys used to sign the zone for the new set to be accepted by the validator (i.e. the number of signatures that verify is not sufficient).

STALE: There is no overlap between the configured trust anchors and the set of SEP keys used to sign the zone. Here validation of data is no longer possible and hence we are in a situation where the trust anchors are stale.

Of these five states only two (IN-SYNC and OUT-OF-SYNC) are part of the automatic trust update mechanism. The PRIMING state is where a validator is located before acquiring an up-to-date set of trust anchors. The transition from PRIMING to IN-SYNC is manual (see Section 4 below).

Example: assume a secure entry point with four SEP keys and a

validator with the policy that it will accept any update to the set of trust anchors as long as no more than two signatures fail to validate (i.e. M >= N-2) and at least two signature does validate (i.e. M >= 2). In this case the rollover of a single key will move the validator from IN-SYNC to OUT-OF-SYNC. When the trust update

state machine updates the trust anchors it returns to state IN-SYNC.

If if for some reason it fails to update the trust anchors then the next rollover (of a different key) will move the validator from OUT-OF-SYNC to OUT-OF-SYNC (again), since there are still two keys that are configured as trust anchors and that is sufficient to accpt an automatic update of the trust anchors.

The UNSYNCABLE state is where a validator is located if it for some reason fails to incorporate enough updates to the trust anchors to be able to accept new updates according to its local policy. In this example (i.e. with the policy specified above) this will either be because M < N-2 or M < 2, which does not suffice to authenticate a successful update of trust anchors.

Continuing with the previous example where two of the four SEP keys have already rolled, but the validator has failed to update the set of trust anchors. When the third key rolls over there will only be one trust anchor left that can do successful validation. This is not sufficient to enable automatic update of the trust anchors, hence the new state is UNSYNCABLE. Note, however, that the remaining up-to-date trust anchor is still enough to do successful validation so the validator is still "working" from a DNSSEC point of view.

The STALE state, finally, is where a validator ends up when it has zero remaining current trust anchors. This is a dangerous state, since the stale trust anchors will cause all validation to fail. The escape is to remove the stale trust anchors and thereby revert to the PRIMING state.

3.5 Implementation notes

The DNSSEC protocol specification ordains that a DNSKEY to which a DS record points should be self-signed. Since the keys that serve as trust anchors and the keys that are pointed to by DS records serve the same purpose, they are both secure entry points, we RECOMMEND that zone owners who want to facilitate the automated rollover scheme documented herein self-sign DNSKEYs with the SEP bit set and that implementation check that DNSKEYs with the SEP bit set are self-signed.

In order to maintain a uniform way of determining that a keyset in the zone has been replaced by a more recent set the automatic trust update machine SHOULD only accept new DNSKEY RRsets if the accompanying RRSIGs show a more recent inception date than the present set of trust anchors. This is also needed as a safe guard against possible replay attacks where old updates are replayed "backwards" (i.e. one change at a time, but going in the wrong

direction, thereby luring the validator into the UNSYNCABLE and finally STALE states).

In order to be resilient against failures the implementation should collect the DNSKEY RRsets from (other) authoritative servers if verification of the self signatures fails.

The threshold-based trust update mechanism SHOULD only be applied to algorithms, as represented in the algorithm field in the DNSKEY/RRSIG [3], that the resolver is aware of. In other words the SEP keys of unknown algorithms should not be used when counting the number of available signatures (the N constant) and the SEP keys of unknown algorithm should not be entered as trust anchors.

When in state UNSYNCABLE or STALE manual intervention will be needed to return to the IN-SYNC state. These states should be flagged. The most appropriate action is human audit possibly followed by re-priming (Section 4) the keyset (i.e. manual transfer to the PRIMING state through removal of the configured trust anchors).

An implementation should regularly probe the the authoritative nameservers for new keys. Since there is no mechanism to publish rollover frequencies this document RECOMMENDS zone owners not to roll their key signing keys more often than once per month and resolver administrators to probe for key rollsovers (and apply the threshold criterion for acceptance of trust update) not less often than once per month. If the rollover frequency is higher than the probing frequency then trust anchors may become stale. The exact relation between the frequencies depends on the number of SEP keys rolled by the zone owner and the value M configured by the resolver administrator.

In all the cases below a transaction where the threshold criterion is not satisfied should be considered bad (i.e. possibly spoofed or otherwise corrupted data). The most appropriate action is human audit.

There is one case where a "bad" state may be escaped from in an automated fashion. This is when entering the STALE state where all DNSSEC validation starts to fail. If this happens it is concievable that it is better to completely discard the stale trust anchors (thereby reverting to the PRIMING state where validation is not

possible). A local policy that automates removal of stale trust anchors is therefore suggested.

3.6 Possible transactions

3.6.1 Single DNSKEY replaced

This is probably the most typical transaction on the zone owners part. The result should be that if the threshold criterion is satisfied then the key store is updated by removal of the old trust anchor and addition of the new key as a new trust anchor. Note that if the DNSKEY RRset contains exactly M keys replacement of keys is not possible, i.e. for automatic rollover to work M must be stricly less than N.

3.6.2 Addition of a new DNSKEY (no removal)

If the threshold criterion is satisfied then the new key is added as a configured trust anchor. Not more than N-M keys can be added at once, since otherwise the algorithm will fail.

3.6.3 Removal of old DNSKEY (no addition)

If the threshold criterion is satisfied then the old key is removed from being a configured trust anchor. Note that it is not possible to reduce the size of the DNSKEY RRset to a size smaller than the minimum required value for M.

3.6.4 Multiple DNSKEYs replaced

Arguably it is not a good idea for the zone administrator to replace several keys at the same time, but from the resolver point of view this is exactly what will happen if the validating resolver for some reason failed to notice a previous rollover event.

Not more than N-M keys can be replaced at one time or the threshold criterion will not be satisfied. Or, expressed another way: as long as the number of changed keys is less than or equal to N-M the validator is in state OUT-OF-SYNC. When the number of changed keys becomes greater than N-M the state changes to UNSYNCABLE and manual action is needed.

3.7 Removal of trust anchors for a trust point

If the parent of a secure entry point gets signed and it's trusted keys get configured in the key store of the validating resolver then the configured trust anchors for the child should be removed entirely unless explicitly configured (in the utility configuration) to be an exception.

The reason for such a configuration would be that the resolver has a local policy that requires maintenance of trusted keys further down the tree hierarchy than strictly needed from the point of view.

The default action when the parent zone changes from unsigned to signed should be to remove the configured trust anchors for the child. This form of "garbage collect" will ensure that the automatic rollover machinery scales as DNSSEC deployment progresses.

3.8 No need for resolver-side overlap of old and new keys

It is worth pointing out that there is no need for the resolver to keep state about old keys versus new keys, beyond the requirement of tracking signature inception time for the covering RRSIGs as described in Section 3.4.

From the resolver point of view there are only trusted and not trusted keys. The reason is that the zone owner needs to do proper maintenance of RRSIGs regardless of the resolver rollover mechanism and hence must ensure that no key rolled out out the DNSKEY set until there cannot be any RRSIGs created by this key still legally cached.

Hence the rollover mechanism is entirely stateless with regard to the keys involved: as soon as the resolver (or in this case the rollover tracking utility) detects a change in the DNSKEY RRset (i.e. it is now in the state OUT-OF-SYNC) with a sufficient number of matching RRSIGs the configured trust anchors are immediately updated (and thereby the machine return to state IN-SYNC). I.e. the rollover machine changes states (mostly oscillating between IN-SYNC and OUT-OF-SYNC), but the status of the DNSSEC keys is stateless.

4. Bootstrapping automatic rollovers

It is expected that with the ability to automatically roll trust anchors at trust points will follow a diminished unwillingness to roll these keys, since the risks associated with stale keys are minimized.

The problem of "priming" the trust anchors, or bringing them into sync (which could happen if a resolver is off line for a long period in which a set of SEP keys in a zone 'evolve' away from its trust anchor configuration) remains.

For (re)priming we can rely on out of band technology and we propose the following framework.

4.1 Priming Keys

If all the trust anchors roll somewhat frequently (on the order of months or at most about a year) then it will not be possible to design a device, or a software distribution that includes trust anchors, that after being manufactured is put on a shelf for several key rollover periods before being brought into use (since no trust anchors that were known at the time of manufacture remain active).

To alleviate this we propose the concept of "priming keys". Priming keys are ordinary DNSSEC Key Signing Keys with the characteristic that

- The private part of a priming key signs the DNSKEY RRset at the security apex, i.e. at least one RRSIG DNSKEY is created by a priming key rather than by an "ordinary" trust anchor
- the public parts of priming keys are not included in the DNSKEY RRset. Instead the public parts of priming keys are only available out-of-band.
- The public parts of the priming keys have a validity period. Within this period they can be used to obtain trust anchors.
- The priming key pairs are long lived (relative to the key rollover period.)

4.1.1 Bootstrapping trust anchors using a priming key

To install the trust anchors for a particular security apex an administrator of a validating resolver will need to:

- query for the DNSKEY RRset of the zone at the security apex;
- verify the self signatures of all DNSKEYs in the RRset;
- verify the signature of the RRSIG made with a priming key -- verification using one of the public priming keys that is valid at that moment is sufficient;

- create the trust anchors by extracting the DNSKEY RRs with the SEP flag set.

The SEP keys with algorithms unknown to the validating resolver SHOULD be ignored during the creation of the trust anchors.

4.1.2 Distribution of priming keys

The public parts of the priming keys SHOULD be distributed exclusively through out-of-DNS mechanisms. The requirements for a distribution mechanism are:

- it can carry the "validity" period for the priming keys;
- it can carry the self-signature of the priming keys;
- and it allows for verification using trust relations outside the DNS.

A distribution mechanism would benefit from:

- the availability of revocation lists;
- the ability of carrying zone owners policy information such as recommended values for "M" and "N" and a rollover frequency;
- and the technology on which is based is readily available.

5. The Threshold Rollover Mechanism vs Priming

There is overlap between the threshold-based trust updater and the Priming method. One could exclusively use the Priming method for maintaining the trust anchors. However the priming method probably relies on "non-DNS' technology and may therefore not be available for all devices that have a resolver.

6. Security Considerations

6.1 Threshold-based Trust Update Security Considerations

A clear issue for resolvers will be how to ensure that they track all rollover events for the zones they have configure trust anchors for. Because of temporary outages validating resolvers may have missed a rollover of a KSK. The parameters that determine the robustness against failures are: the length of the period between rollovers during which the KSK set is stable and validating resolvers can actually notice the change; the number of available KSKs (i.e. N) and the number of signatures that may fail to validate (i.e. N-M).

With a large N (i.e. many KSKs) and a small value of M this operation becomes more robust since losing one key, for whatever reason, will not be crucial. Unfortunately the choice for the number of KSKs is a local policy issue for the zone owner while the choice for the parameter M is a local policy issue for the resolver administrator.

Higher values of M increase the resilience against attacks somewhat; more signatures need to verify for a rollover to be approved. On the other hand the number of rollover events that may pass unnoticed before the resolver reaches the UNSYNCABLE state goes down.

The threshold-based trust update intentionally does not provide a revocation mechanism. In the case that a sufficient number of private keys of a zone owner are simultaneously compromised the the attacker may use these private keys to roll the trust anchors of (a subset of) the resolvers. This is obviously a bad situation but it is not different from most other public keys systems.

However, it is important to point out that since any reasonable trust anchor rollover policy (in validating resolvers) will require more than one RRSIG to validate this proposal does provide security concious zone administrators with the option of not storing the individual private keys in the same location and thereby decreasing the likelihood of simultaneous compromise.

6.2 Priming Key Security Considerations

Since priming keys are not included in the DNSKEY RR set they are less sensitive to packet size constraints and can be chosen relatively large. The private parts are only needed to sign the DNSKEY RR set during the validity period of the particular priming key pair. Note that the private part of the priming key is used each time when a DNSKEY RRset has to be resigned. In practice there is therefore little difference between the usage pattern of the private

part of key signing keys and priming keys.

7. IANA Considerations

NONE.

8. References

8.1 Normative References

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

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

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

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

8.2 Informative References

[5] Kolkman, O., "DNSSEC Operational Practices", draft-ietf-dnsop-dnssec-operational-practices-01 (work in progress), May 2004.

[6] Housley, R., Ford, W., Polk, T. and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", RFC 2459, January 1999.

Authors' Addresses

Johan Ihren
Autonomica AB
Bellmansgatan 30
Stockholm SE-118 47
Sweden

EMail: johani@autonomica.se

Olaf M. Kolkman
NLnet Labs
Kruislaan 419
1098 VA Amsterdam
NL

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

Bill Manning
EP.net
Marina del Rey, CA 90295
USA

Appendix A. Acknowledgments

The present design for in-band automatic rollovers of DNSSEC trust anchors is the result of many conversations and it is no longer possible to remember exactly who contributed what.

In addition we've also had appreciated help from (in no particular order) Paul Vixie, Sam Weiler, Suzanne Woolf, Steve Crocker, Matt Larson and Mark Kosters.

Appendix B. Document History

This appendix will be removed if and when the document is submitted to the RFC editor.

The version you are reading is tagged as $Revision: 1.2 $.

Text between square brackets, other than references, are editorial comments and will be removed.

B.1 prior to version 00

This draft was initially published as a personal submission under the name draft-kolkman-dnsext-dnssec-in-band-rollover-00.txt.

Kolkman documented the ideas provided by Ihren and Manning. In the process of documenting (and prototyping) Kolkman changed some of the details of the M-N algorithms working. Ihren did not have a chance to review the draft before Kolkman posted;

Kolkman takes responsibilities for omissions, fuzzy definitions and mistakes.

B.2 version 00

- The name of the draft was changed as a result of the draft being adopted as a working group document.
- A small section on the concept of stale trust anchors was added.
- The different possible states are more clearly defined, including examples of transitions between states.
- The terminology is changed throughout the document. The old term "M-N" is replaced by "threshold" (more or less). Also the interpretation of the constants M and N is significantly simplified to bring the usage more in line with "standard" threshold terminlogy.

B.3 version 01

- This is a notice that the draft temporarily expired.

B.4 version 02

- Resurrected the draft
- Added new section on why rollovers are needed with particular

empathatis on the compromise case.

- Updated references and author affiliations.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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