DNSOP Working Group
Internet-Draft
Intended status: Standards Track
Expires: 11 August 2025

L. Fernandez
E. Bergstrm
J. Stenstam
The Swedish Internet Foundation
S. Crocker
Edgemoor Research Institute

7 February 2025

Distributed DNSSEC Multi-Signer Bootstrap
draft-leon-distributed-multi-signer-00

Abstract

This document presents an architecture for a distributed DNSSEC multi-signer model that most closely resembles "model 2" in [RFC8901].

It defines two multi-signer specific entities: the "multi-signer agent" (MSA) that is responsible for the multi-signer process and the "combiner", which is responsible for "combining" unsigned zone data from the zone owner with zone data under control of the MSA. It introduces a new DNS RRtype, HSYNC, that is used by the zone owner to designate the chosen DNS Providers (signing and/or serving the zone). Furthermore it describes a mechanism for the MSAs to establish secure communication with each other, either via “pure DNS” communication secured by DNS SIG(0) signatures on each message or via a RESTful API secured by TLS. Finally, the document describes two models for multi-signer process synchronization: “leader/follower mode” and “peer mode” and the mechanism by which a set of MSAs decide which model to use for a given zone.

The scope of the document is only the distributed aspect of DNSSEC multi-signer up to the point where secure communication and synchronization method between MSAs has been established. The “multi-signer algorithms” that deal with the actual synchronization required for multi-signer operation are described in [I-D.draft-ietf-dnsop-dnssec-automation].

TO BE REMOVED: This document is being collaborated on in Github at: https://github.com/johanix/draft-leon-dnsop-distributed-multi-signer (https://github.com/johanix/draft-leon-dnsop-distributed-multi-signer). The most recent working version of the document, open issues, etc, should all be available there. The authors (gratefully) accept pull requests.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The issue of how to eliminate so-called "single points of failure" from systems to make them more robust is a recurring theme in systems design and so also for DNS. In the DNS case redundancy is addressed by having multiple name servers for the same zone. However, when the zone is DNSSEC-signed there is traditionally an additional single point of failure: the so-called "signer" of the zone.

In multi-signer ([RFC8901]) model 2, a process is described by which it is possible to use more than one signer (each with its own set of keys), by having the signers (or their agents) communicate and exchange data that should be signed by the other signer. The most obvious example is that each signer's Key-Signing Key must sign a DNSKEY RRset that contains the Zone-Signing Keys for all signers.

To synchronize data between signers two models are possible: either a "centralized" model where a single "controller" decides what changes are needed, or a "distributed" model where the signers themselves (or an agent of each signer) decide what changes are needed.

The first model has been implemented previously, and while it works from a technical point of view, it is not a good solution from a risk management point of view. The primary problem is that the signers have difficulty accepting that an external third party (the controller) has the ability to modify data (in a customer zone).

This document is an attempt to address the synchronization problem by proposing a distributed model without a central controller.

The communication between signers has two parts: first it is necessary to find out what data each signer has for a zone. Once all data has been collected it is possible to compute what changes are needed to the zone data at each signer. That triggers the second phase where the zone data for the individual signers is changed to get them in sync with each other. All of this is done automatically.

However, from a slightly different perspective, the multi-signer alternative is the more general case of DNSSEC signing, with the (very common) case of a single signer being a special case.

From that point of view, this document proposes an architecture for a completely automated, distributed multi-signer model together with a seamless transition path from the current single-signer model to the multi-signer model. From the zone owners point of view, the transition is done through the addition of a new RRtype, HSYNC, that is used to designate the chosen DNS Providers, their responsibilities and information to enable the DNS Providers to locate each other.

Knowledge of DNS NOTIFY [RFC1996] and DNS Dynamic Updates [RFC2136] and [RFC3007] is assumed. DNS SIG(0) transaction signatures are documented in [RFC2931].

1.1. Requirements Notation

The key words "*MUST*", "*MUST NOT*", "*REQUIRED*", "*SHALL*", "*SHALL NOT*", "*SHOULD*", "*SHOULD NOT*", "*RECOMMENDED*", "*NOT RECOMMENDED*", "*MAY*", and "*OPTIONAL*" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

...

3. Requirements

The requirements for an architecture for distributed multi-signer are defined as follows:

* Assuming all zone transfers are correctly set up, a zone owner MUST be able to signal to the individual multi-signer providers information sufficient for the providers to identify each other and establish secure communication.

* The zone owner MUST be able to signal the intent to onboard an additional multi-signer provider. This MUST automatically initiate the multi-signer “add signer” process, as described in [RFC8901].

* The zone owner MUST be able to signal the intent to offboard an existing multi-signer provider. This MUST automatically initiate the multi-signer “remove signer” process, as described in [RFC8901].

* All signalling from zone owner to multi-signer providers SHOULD be carried out via data in the served zone, to ensure that all providers get the same configuration information at (almost) the same time.

* By engaging a set of multi-signer providers (one or more), the zone owner MUST give up control over the following records:
  - All DNSSEC related records in the zone
  - Any CDS and/or CSYNC RRsets
* It SHOULD be possible but NOT MANDATORY for the zone owner to also delegate the management of the NS RRset to the set of DNS Providers.

4. Multi-Signer Use Cases

4.1. Primary Use Case

The primary use case for the proposed multi-signer architecture is the following scenario: A zone owner needs to remove the single point of failure that the DNSSEC signer constitutes. For this reason it contracts with two or more “multi-signer capable” DNS providers. Each such DNS provider provides the following service:

* Receive the unsigned zone via zone transfer.
* Locate all active DNS Providers via the HSYNC RRset as published by the zone owner. Establish secure communication with all remote DNS Providers (via their agents).
* Update the DNSKEY, CDS and CSYNC RRsets as needed, based on synchronization with the remote signers (or their agents).
* Update the NS RRset if allowed by the zone owner, based on synchronization with the remote DNS Providers (or their agents).
* Sign the zone, using own DNSKEYs, but with a published DNSKEY RRset that includes the DNSKEYs of other signers.
* Possibly distribute the signed zone to a set of downstream authoritative nameservers under own control.
* Possbly distribute the signed zone to non-signing downstream DNS Providers.

4.2. Secondary Use Case

A slightly different use case is where a zone owner has a desire to replace one DNSSEC-signing DNS provider with another. In the first step it onboards the new DNS provider by adding a HSYNC RR with HSYNC State="ON" thereby identifying the new DNS provider and signalling its role. This informs both the present DNS providers and the incoming DNS provider about the addition of the new DNS provider and the onboarding process is automatically initiated.

Once the onboarding operation is completed the zone owner may trigger the pending removal of another DNS provider by changing the HSYNC State flag for the outgoing DNS Provider to "OFF". This informs all the present DNS providers about the pending removal and the offboarding process is automatically initiated.

4.3. Tertiary Use Case

A third use case is where a zone owner wants to migrate from a single-signer model to a multi-signer model, but as a first step only wants to transition the existing signer to be designated via a single HSYNC record. Once that is done the zone owner can continue the transition to a full multi-signer model at a later time by adding more HSYNC records.

5. The Distributed Multi-Signer Model

The primary difference between monolithic and distributed multi-signer is that the former has a central "controller" while the latter doesn't. But there is still an absolute need for synchronization between the different participants in the distributed multi-signer setup.

There are three immediate aspects for the design of a distributed multi-signer architecture:

* The first is "transport": how to communicate between the individual instances in a multi-signer system.

* The second is "synchronization": who decides what changes are needed where.

* The third is source of truth for different types of zone data. The zone owner is the source of truth for all unsigned zone data, except DNSSEC data. The signer is the source of truth for all DNSSEC data in the zone. Traditionally, the source of truth for the NS RRset is the zone owner, but with multiple DNS Providers having the option of moving that responsibility to the DNS Providers would be an important improvement.

5.1. Multi-Signer Agent: Integrated Signer vs Separate Agent

In a distributed setup there must be a service located with each multi-signer DNS Provider that manages communication with other DNS Providers. This is referred to as the multi-signer agent, or MSA. As not every DNS Provider needs to be signing the zone, the term is not entirely perfect, but sufficient.

It is possible to implement support for the synchronization and communication needs directly into each “signer” (i.e. typically an authoritative nameserver with the ability to do online DNSSEC signing). In this case the signer implements the MSA functionality.

However, it is also possible to separate the multi-signer functionality into a separate agent. This agent sits next to the signer, and is under the same administrative control (the "DNS Provider"), but is a separate piece of software. When using this design each signer has an MSA attached next to it. Each MSA is configured as a “secondary nameserver” to a signer and receives the (signed) zone from this signer.

The “separate MSA” design has the major advantage of leaving the signer almost entirely out of the multi-signer complexity. The requirements are only that the “signer” treats the MSA as a normal secondary (sends NOTIFY messages and responds to zone transfer requests) and that the MSA has a mechanism that allows it to make changes to zones upstream of the “signer” to satisfy the multi-signer requirements for synchronization of certain RRsets in the apex of the zone.

In this document the design using a separate MSA is used, while pointing out that it is possible to integrate this into a future “signer” that implements both DNSSEC signing and the MSA functionality.

5.2. Source of Truth

A common design for DNSSEC signing (regardless of multi-signer) is to use a separate, bump-on-the-wire signer. This is a signer that receives the unsigned zone via an incoming zone transfer, signs the zone, and publishes the signed zone via an outbound zone transfer. In such a design the source of truth has been split up between the "zone owner" (source of truth for all non-DNSSEC zone data), and the signer (source of truth for all DNSSEC data in the zone).

In a distributed multi-signer architecture the source of truth is further split up into three participants:

* The zone owner is the source of truth for all unsigned zone data, except DNSSEC data and possibly the NS RRset.

* The signer is the source of truth for all data generated via DNSSEC signing: own DNSKEYs, NSEC/NSEC3 RRs, RRSIGs, etc.

* The MSA is the source of truth for the RRsets that must be kept in sync across all the signers for the zone. This includes the DNSKEYs from other signers, CDS and CSYNC RRsets. Possibly also the NS RRset.

The NS RRset is an interesting special case. Traditionally the NS RRset is maintained by the zone owner, but based on data from the DNS providers (as authoritative nameservers is a primary service for the DNS provider). However, in a distributed multi-signer architecture the NS RRset should preferably be maintained by the MSA. For this reason the proposed design makes control of the NS RRset explicit and the responsibility of the zone owner to choose whether to retain control or delegate to the MSAs. Hence:

* The MSA is the source of truth for the NS RRset, subject to the policy of the zone owner, as described in the HSYNC RRset.

Making the control of the NS RRset explicit is useful regardless of whether a zone uses multiple signers or single signer.

To be able to keep the signer as simple as possible, the changes to the NS, DNSKEY, CDS and CSYNC RRsets must be introduced into the unsigned zone before the zone reaches the signer. Likewise, to keep the zone owner as simple as possible (i.e. not involved in the details of the multi-signer automation) these changes must be introduced into the unsigned zone after the zone leaves the zone owner.

5.2.1. The COMBINER

The consequence of these requirements is that the DNSKEY, CDS and CSYNC RRsets (and possibly the NS RRset) are maintained via a separate piece of software inserted between the zone owner and the signer. This is referred to as the multi-signer COMBINER.

The COMBINER has the following features:

* It supports inbound zone transfer of the unsigned zone from the zone owner.

* It receives updates for the NS, DNSKEY, CDS and CSYNC RRsets from the MSA. Typically the mechanism used is DNS UPDATE with a TSIG signature, as this is easy to configure in a local context. However, other mechanisms, including APIs, are possible.

* It stores all data received from the MSA separate from the zone data received from the zone owner.

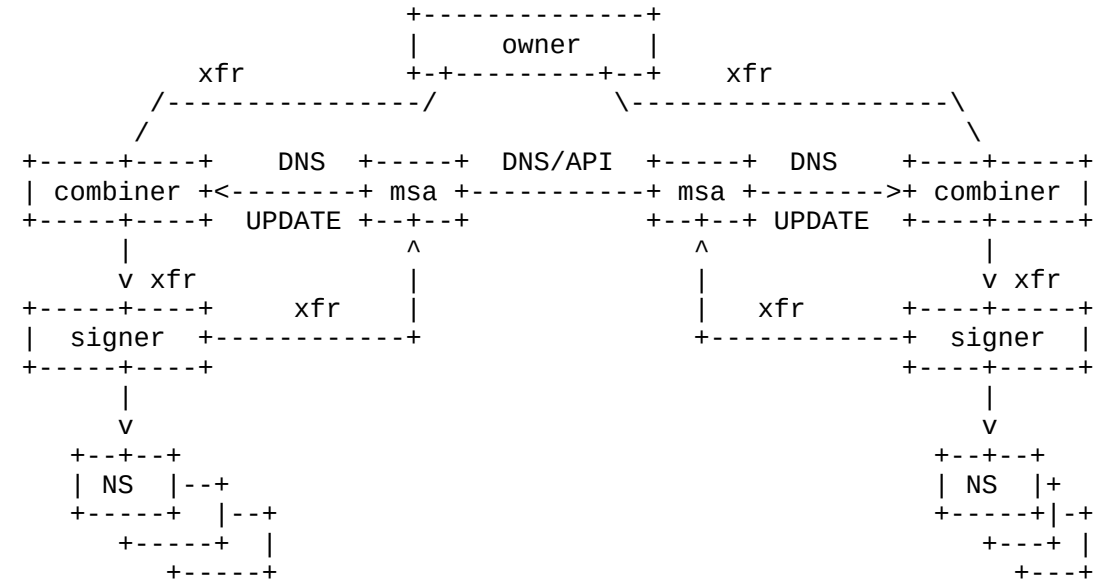
* Whenever it receives a new unsigned zone from the zone owner it COMBINES zone data from the zone owner (the majority of the zone) with specific zone data under control of the MSA: three specific RRsets, all in the apex of the zone: the DNSKEY,CDS and CSYNC RRsets.

* It is policy free. I.e. the COMBINER is not making any judgement about what data to include in the zone from the four defined RRsets. That judgement is the role of the MSA.

* It does not sign the zone.

* It provides outbound zone transfer of the combined zone to the signer.

Example setup with two signers showing the logical flow of zone data between the zone owner, the COMBINER, the signer and the MSA:



5.3. The DNS Provider

A "DNS Provider" is a term that is most commonly used to refer to an entity that provides authoritative DNS service to one or more zone owners. In the context of this document it is used to refer to an entity that provides some subset of the following services:

* Signing a zone received from the zone owner.

* Serving the zone via a set of authoritative nameservers.

* Distributing the signed zone to other downstream DNS Providers.

In addition to the above services a DNS Provider MUST also provide:

* An MSA for synchronization with other DNS Providers

* A COMBINER for the management of changes to the zone via the synchronization among MSAs (if it provides a signer)

I.e. in the setup above there are two DNS Providers, both of which are "complete" in the sense that they provide all three of the above services.

6. Identifying the Designated Signers

It is the responsibility of the zone owner to choose a set of "DNS Providers", either internal or external to the zone owners organization. These DNS Providers MUST be clearly and uniquely designated via publication in the HSYNC RRset, located at the apex of the zone and consisting of one HSYNC record for each signer.

The HSYNC RRset MUST be added, by the zone owner, to the, typically unsigned, zone that the zone owner maintains so that this RRset is visible to the downstream DNS Providers and their multi-signer agents.

7. The HSYNC RRset

The HSYNC RR has the zone name that publishes the HSYNC RRset as the owner name (i.e. the HSYNC RRset must be located at the apex of the zone). The RDATA consists of four fields "State","NSMgmt", "Sign" and "Identity":

zone.example. IN HSYNC State NSMgmt Sign Identity.

State: Unsigned 8-bit. Defined values are 1=ON and 2=OFF. The value 0 is an error. Values 3-127 are presently undefined. Values 128-255 are reserved for private use. The presentation format allows either as integers (1 or 2) or as tokens (“ON” or “OFF”).

NSMgmt: Unsigned 8-bit. Defined values are 1=Zone owner and 2=MSA. The value 0 is an error. Values 3-255 are presently undefined (and not expected to be defined). The presentation format allows either as integers (1 or 2) or as tokens (“OWNER” or “AGENT”).

Sign: Unsigned 8-bit. Defined values are 1=YES and 2=NO. The value 0 is an error. If Sign=YES for a particular HSYNC record, then the signer associated with that Identity is a designated signer for the zone.

Identity: Domain name. Used to uniquely identify the Multi-Signer Agent for the DNS Provider that the MSA represents.

Example:

zone.example. IN HSYNC ON AGENT YES msa.provider.example.

7.1. Semantics of the HSYNC State Field

The HSYNC State field is used to signal to all MSAs what the status of each MSA is from the point-of-view of the zone owner. The two possible values are "ON" and "OFF" where "ON" means that the MSA is a currently designated signer for the zone and "OFF" means that the MSA is previously designated signer for the zone that is in the process of being offboarded.

The reason for the "OFF" state is that the offboarding process involves the remaining signers (hence the signalling) and it is important to know which signer is being offboarded so that the correct data may be removed in the correct order during the multi-signer "remove signer" process (see [RFC8901]).

Once the offboarding process is complete the HSYNC RR for the offboarded MSA may be removed from the zone at the zone owners discretion.

7.2. Semantics of the HSYNC NSMgmt Field

The NSMgmt field is used to signal to the MSAs who is responsible for the contents of the NS RRset for the zone. The two possible values are "OWNER" and "AGENT".

The value "OWNER" signals that the zone owner is responsible for the NS RRset and is responsible for updating the NS RRset (either with or without the unified data from all MSAs). In this case the MSAs MUST NOT instruct the COMBINER to update the NS RRset.

The value "AGENT" means that the MSAs representing DNS Providers that sign the zone are responsible for the contents of the NS RRset. In this case the these MSAs MUST instruct the COMBINER to update the NS RRset with the unified NS RRset data from all MSAs.

7.2.1. Limitation of Scope for NS Management

For the purpose of this document the NSMgmt Field only covers the NS RRset. I.e. it does not include the address records of in-bailiwick authoritative nameservers. The reasons are:

* Limiting the possibility of DNS Providers "polluting" the name space of the zone.

* Keeping the specification simpler, as the concept of "delegated" NS management is new.

It is possible to make an argument for delegating management of address records for in-bailiwick authoritative nameservers, but this document does not.

7.3. Semantics of the HSYNC Sign Field

The Sign field is used to signal to all MSAs whether the zone owner requests that the DNS Provider that the MSA represents should sign the zone or not. The two possible values are "YES" and "NO" where "YES" means that the MSA represents a currently designated signer for the zone and "NO" means that the MSA does not.

When Sign=NO the MSA MUST still participate in the communication between MSAs for the zone, but MUST NOT instruct the COMBINER to update the NS RRset.

8. Communication Between MSAs

For the communication between MSAs there are two choices that need to be made among the designated MSAs for a zone. The first is what "transport" to use for the communication. The second is what "synchronization" model to use when executing future multi-signer processes.

The two defined transport alternatives are:

* DNS-based communication (mandatory to support)

* REST API-based communication

Each has pros and cons and at this point in time it is not clear that one always is better than the other. To simplify the choice of transport DNS-based communication is mandatory to support and the REST API-based communication may only be used if all MSAs support it. Supported transports are signaled in the Multi-Signer EDNS(0) Option (see section NNN below).

The two defined synchronization alternatives are:

* Leader/Follower synchronization (mandatory to support)

* Peer-to-Peer synchronization

Just as for transport, supported synchronization models are signaled in the Multi-Signer EDNS(0) Option (see section NNN below).

Regardless of the synchronization model and communication method used, the MSAs SHOULD exchange all needed information about the zone and the DNS Provider they represent to enable the multi-signer processes to execute correctly. This includes notifications about changes to DNSKEYs, changes to the NS RRset, etc. Depending on synchronization model it may also include instructions for changes to the zone.

8.1. MSA Communication via DNS

This transport alternative is based on the observation that all the communication needs between MSAs can be expressed via DNS messages. Notifications are sent as DNS NOTIFY messages. Requests for changes to a zone are sent as DNS UPDATE messages, etc. The sole remaining communication requirement is for how to communicate information about the current state between MSAs in an ongoing multi-signer process. For this reason a dedicated EDNS(0) opcode specifically for multi-signer synchronization is proposed.

This model is based on [I-D.draft-berra-dnsop-keystate] that solves a similar problem for delegation synchronization between child and parent, which has already been implemented and shown to work.

8.2. MSA Communication via REST API

REST APIs are well-known and a natural fit for many distributed systems. The challenge is mostly in the initial setup of secure communication. The certificates need to be validated, preferably without a requirement on trusting a third party CA. The API endpoints for each MSA need to be located. Once secure communication has been established, using a REST API for MSA communication is straight-forward.

8.3. Locating Remote Multi-Signer Agents

When an MSA receives a zone via zone transfer from the signer it will analyze the zone to see whether it contains an HSYNC RRset. If there is no HSYNC RRset the zone MUST be ignored by the MSA from the point-of-view of multi-signer synchronization.

If, however, the zone does contain an HSYNC RRset then the MSA must analyze this RRset to identify the other MSAs for the zone via their target names in each HSYNC record. If any of the other MSAs listed in the HSYNC RRset is previously unknown to this MSA then secure communication with this other MSA MUST be established.

Secure communication can be achieved via various transports and it is up to the MSAs in the zone's HSYNC records to determine amongst themselves. This document proposes two transports: "DNS" and "API". "DNS" is designated as as a baseline that MSAs MUST support to be compliant.

The following two subsections describe the mechanism by which an MSA SHOULD locate a remote MSA and establish secure DNS-based and API-based communications, respectively.

8.3.1. Locating a Remote DNS-Method Multi-Signer Agent

Locating a remote MSA using the DNS mechanism consists of the following steps:

* Lookup and DNSSEC-validate a URI record for the DNS protocol for the HSYNC identity. This provides the domain name and port to which DNS messages should be sent.

* Lookup and DNSSEC-validate the SVCB record of the URI record target to get the IP addresses to use for communication with the remote MSA. If the returned SVCB record includes a "port=NNN" hint then this MUST be ignored. I.e. the port to use is defined by the URI record.

* Lookup and DNSSEC-validate the KEY record of the URI record target name. This enables verification of the SIG(0) public key of the remote MSA once communication starts.

Example: given the following HSYNC record for a remote MSA:

zone.example. IN HSYNC ON AGENT YES msa.provider.com.

The local MSA will look up the URI record for msa.provider.com:

_dns._tcp.msa.provider.com. IN URI 10 10 "dns://ns.msa.provider.com:5399/" _dns._tcp.msa.provider.com. IN RRSIG URI …

which triggers a lookup for ns.msa.provider.com. SVCB to get the IPv4 and IPv6 addresses as ipv4hints and ipv6hints in the response to the SVCB query:

ns.msa.provider.com. IN SVCB 1 ipv4hint=5.6.7.8 ipv6hint=2001::53
ns.msa.provider.com. IN RRSIG SVCB …

and also a look up for the KEY record for ns.msa.provider.com, which may look like this:

ns.msa.provider.com. IN KEY … ns.msa.provider.com. IN RRSIG KEY …

Once all the DNS lookups and DNSSEC-validation of the returned data has been done, the local MSA is able to initiate communication with the remote MSA and verify the identity of the responding party via the validated KEY record for the remote MSAs SIG(0) public key.

8.3.2. Locating a Remote API-Method Multi-Signer Agent

Locating a remote MSA using the API mechanism consists of the following steps:

* Lookup and DNSSEC-validate the URI record for for the HTTPS protocol for the HSYNC identity. This provides the base URL that will be used to construct the individual API endpoints for the REST API. It also provides the port to use.

* Lookup and DNSSEC-validate the SVCB record for the URI record target. This provides the IP-addresses to use for communication with the MSA. If the returned SVCB record includes a "port=NNN" hint then this MUST be ignored. I.e. the port to use is defined by the URI record.

* Lookup and DNSSEC-validate the TLSA record for the port and protocol specified in the URI record. This will enable verification of the certificate of the remote MSA once communication starts.

Example: given the following HSYNC record for a remote MSA:

zone.example. IN HSYNC ON AGENT YES msa.provider.com.

the local MSA will look up the URI record for msa.provider.com:

_https._tcp.msa.provider.com. IN URI 10 10
“https://api.msa.provider.com:443/api/v2/”
_https._tcp.msa.provider.com. IN RRSIG URI …

which triggers a lookup for api.msa.provider.com IPv4 and IPv6 addresses as hints in an SVCB RR:

api.msa.provider.com. IN SVCB 1 ipv4hint=1.2.3.4
ipv6hint=2001::bad:cafe:443 api.msa.provider.com. IN RRSIG SVCB …

Now we know the IP-address and the port as well as the base URL to use. Finally the TLSA record for _443._tcp.api.msa.provider.com is looked up, with a response that may look like this:

```
_443._tcp.api.msa.provider.com.  IN TLSA 3 1 1 ….
_443._tcp.api.msa.provider.com.  IN RRSIG TLSA …
```

Once all the DNS lookups and DNSSEC-validation of the returned data has been done, the local MSA is able to initiate communication with the remote MSA and verify the identity of the responding party via the TLSA record for the remote MSAs certificate.

8.3.2.1. Fallback to DNS-based Communication

If the API-based communication fails, either because needed DNS records are missing, the TLSA record fails to validate the remote MSAs certificate or the remote MSA simply doesn’t respond, the local MSA MUST fall back to DNS-based communication.

8.4. The Initial HELLO Phase

When two MSAs need to communicate with each other for the first time (because they are both deisgnated signers for the same zone), they need to establish secure communication. This is done in a "HELLO" phase where the MSAs exchange information about their capabilities.

If all the information needed for API-based transport for the remote party was available, the MSA SHOULD attempt an API-based HELLO. If, however, this fails for some reason, it should fall back to DNS-based HELLO.

8.4.1. DNS-based HELLO Phase

When using DNS-based communication the HELLO phase is done by sending a NOTIFY(SOA) for the zone that triggered the need for communication. The NOTIFY message MUST contain a Multi-Signer EDNS(0) Option (see section NNN below).

In the Multi-Signer EDNS(0) Option the OPERATION field MUST have the value "HELLO" (1). Furthermore, the MSA signals its transport and synchronization capabilities in the TRANSPORT and SYNCHRONIZATION fields. This message is signed with the SIG(0) key for the local MSA for which the public key is published as a KEY record for the MSA.

In the response to the NOTIFY, the remote MSA does the same and the two MSAs can now verify each other’s identity and are also aware of the other MSAs transport and synchronization capabilities.

8.4.2. API-based HELLO Phase

When using API-based communication the HELLO phase is done by sending a REST API POST request to the remote MSA at the "/hello" endpoint. The request MUST contain a JSON encoded object with the following fields:

* "transport": The transport capabilities of the local MSA.

* "synchronization": The synchronization capabilities of the local MSA.

The response MUST contain a JSON object with the following fields:

* "transport": The transport capabilities of the remote MSA.

* "synchronization": The synchronization capabilities of the remote MSA.

8.4.3. Interpretation of the HELLO Responses

Once an MSA has received HELLO responses from all other MSAs that are designated signers for the zone, it knows the capabilities of the MSAs as a group. It can then use this information to determine which transport to use:

* If all MSAs support API-based communication, the MSAs will use API-based communication for this zone.

* If one or more MSAs only support DNS-based communication, the MSAs will use DNS-based communication for this zone.

Likewise, each MSA now knows the synchronization capabilities of the other MSAs and can determine which synchronization model to use:

* If all MSAs support the Peer-to-Peer synchronization model, the MSAs will use the Peer-to-Peer synchronization model for this zone.

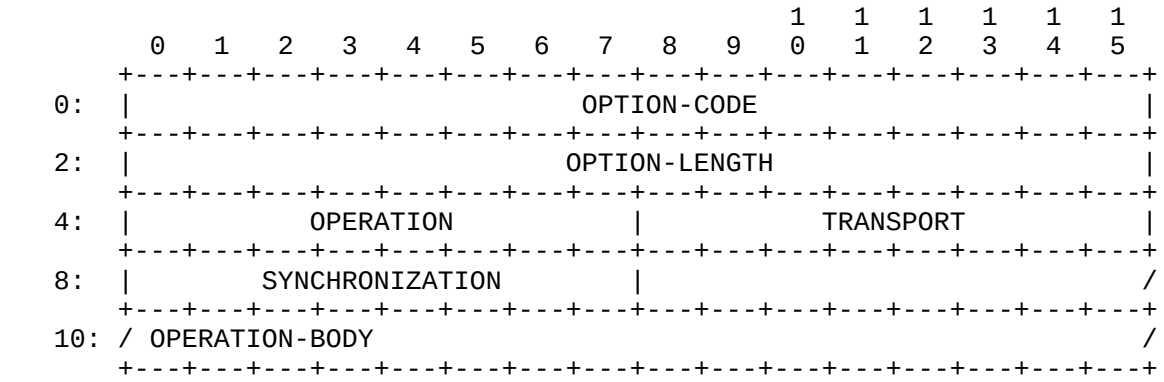
* If one or more MSAs only support the Leader/Follower synchronization model, the MSAs will use the Leader/Follower synchronization model for this zone.

8.5. Multi-Signer EDNS(0) Option Format

This document uses an Extended Mechanism for DNS (EDNS0) [RFC6891] option to include Multi-Signer synchronization information in DNS messages.

This option is structured the same way as the KeyState option described in [I-D.draft-berra-dnsop-keystate], which has been implemented and shown to work for a similar use case. The requirements for multi-signer synchronization are sufficiently different that it is not possible to re-use the KeyState OPT also for this purpose and therefore a new EDNS(0) option is defined here.

The Multi-Signer EDNS(0) option is structured as follows:



Field definition details:

OPTION-CODE: 2 octets / 16 bits (defined in [RFC6891]) contains the value TBD for Multi-Signer.

OPTION-LENGTH: 2 octets / 16 bits (defined in [RFC6891]) contains the length of the payload (everything after OPTION-LENGTH) in octets and should be 4 plus the length of the OPERATION-BODY field (which may be zero octets long).

OPERATION: 8 bits. Signals the type of operation the message performs. This document defines the two operations HELLO and HEARTBEAT. For a complete distributed multi-signer specification a number of additional operations will need to be allocated to be able to describe the states in the different multi-signer processes. This allocation must be done either in a revision to this document or in a subsequent document.

TRANSPORT: 8 bits. Encodes the transport capabilities of the MSA. With 8 bits it is possible to define up to 8 different transports of which this document defines two: DNS and API.

SYNCHRONIZATION: 8 bits. Encodes the synchronization capabilities of the MSA. With 8 bits it is possible to define up to 8 different synchronization models of which this document defines two: Leader/ Follower and Peer-to-Peer.

OPERATION-BODY: Variable-length. Used to carry operation-specific parameters.

8.5.1. Encoding Transport Capabilities in the Multi-Signer EDNS(0) Option

An MSA signals the union of its transport capabilities by setting the corresponding bits to 1.

0: DNS transport supported (baseline, MUST be supported by all MSAs)

1: API transport supported

2: unused

3: unused

4: unused

5: unused

6: unused

7: unused

8.5.2. Encoding Synchronization Capabilities in the Multi-Signer EDNS(0) Option

An MSA signals its synchronization capabilities by setting the corresponding bits to 1.

0: Leader/Follower synchronization supported (baseline, MUST be supported by all MSAs)

1: Peer-to-Peer synchronization supported

2: unused

3: unused

4: unused

5: unused

6: unused

7: unused

9. Sequence Diagram Example of Establishing Secure Comms - "The Hello Phase"

The procedure of locating another MSA and establishing a secure communication, referred to as "The Hello Phase" is examplified in the sequence diagram below.

The procedure is as follows:

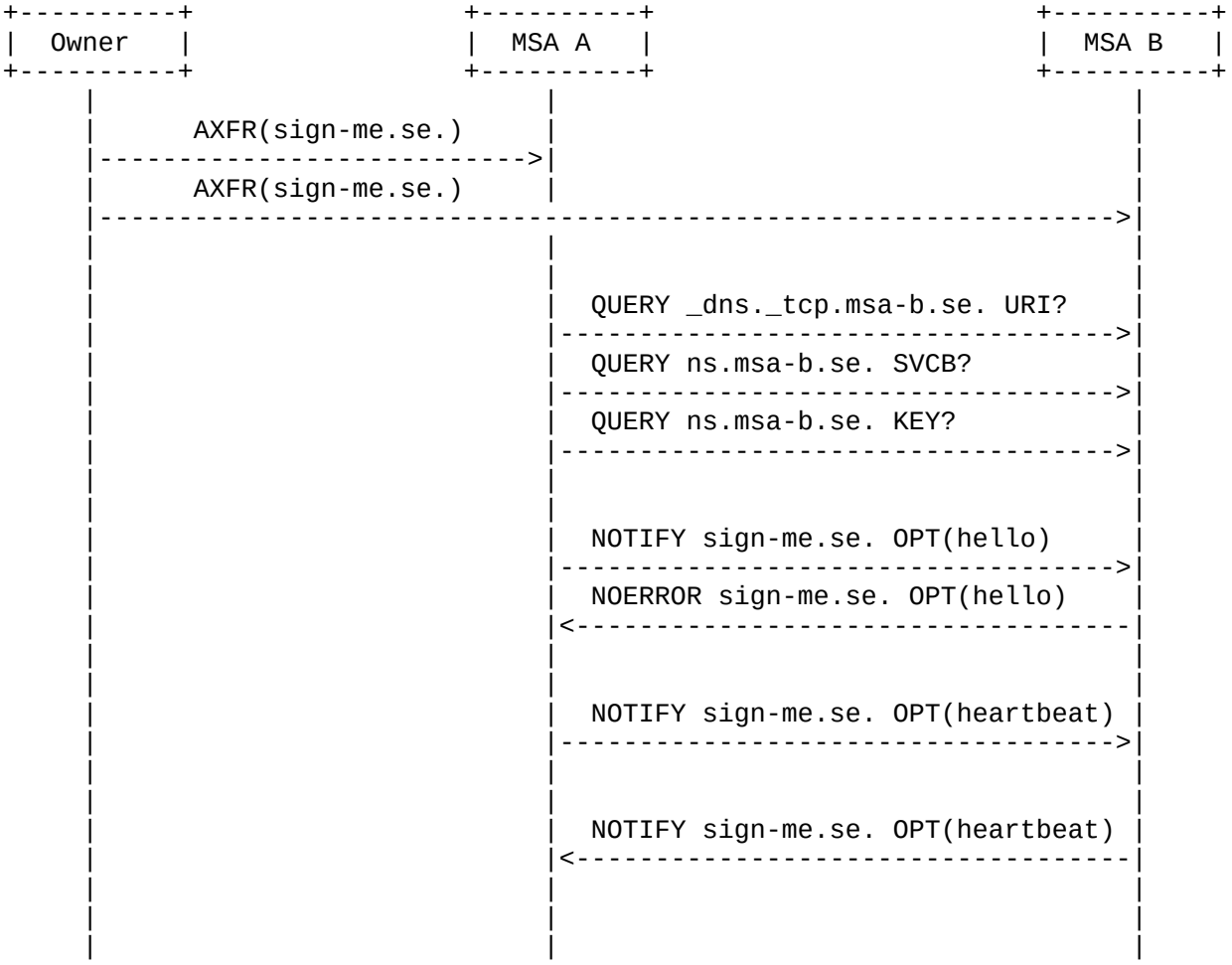
1. The multisigner agents receive a zone via zone transfer. By analyzing the HSYNC RRset each MSA become aware of the identities of the other MSAs for the zone. I.e. each MSA knows which other MSAs it needs to communicate with. Communication with each of these, previously unknown, remote MSAs is referred to as "NEEDED".

2. Each MSA starts aquiring the information needed to establish secure communications with any previously unknown MSAs. Here we only illustrate the baseline case where DNS-based communications is to be used in the following phase. Once all needed information has been collected the communication with this remote MSA is considered to be "KNOWN".

3. Once an MSA has received the required information (URI, SVCB and KEY records in the baseline case) it sends a NOTIFY message with a dedicated Multi-Signer OPT code with OPERATION="HELLO". The sender uses this OPT field to signal its transport and synchronization capabilities. Similarly, the responder signals its capabilities using the same field.

4. When an MSA either gets a NOERROR response to its NOTIFY OPT(hello) message or responds with a NOERROR, it transitions out of "The Hello Phase" with the exchanging party and they transition to the next phaste where they start sending NOTIFY OPT(heartbeat) signals instead. The communication with the remote MSA is now considered to be in the "OPERATIONAL" state.

In the case where one MSA is aware of the need to communicate with another MSA, but the other is not (eg. the zone transfer was dealyed for one of them), the slower one SHOULD respond with a RCODE=REFUSED to any NOTIFY OPT(hello) it receives. Once it is ready, it will send its own NOTIFY OPT(hello) which should be responded to with a RCODE=NOERROR.



10. Synchronization of Changes Between MSAs

There are two defined models for synchronization. The first (Leader/ Follower) has the advantage of more clearly mapping to the original multi-signer model 2, with a single controller. The second model has the advantage of less total communication between MSAs (including no elections) but the potential disadvantage of more fine grained communication during the execution of a multi-signer process.

At this stage it is not clear that one model is superior to the other.

10.1. Leader/Follower Mode

In a leader/follower deployment, a designated multi-signer agent assumes the role of a leader, directing other agents, or followers, through the multi-signer process state transitions. In this mode it is necessary to conduct “elections” where one of the MSAs is chosen as the Leader before initiating a new multi-signer process. Once the Leader has been chosen, this model is mostly equivalent to the original multi-signer “model 2”, with a single controller. The other MSAs (the followers) essentially become proxies between the controller (the Leader) and the DNS Provider each MSA represents.

10.2. Peer Mode

In peer mode, the MSAs still need to locate each other, but instead of relying on trust in each other, each multi-signer agent operates independently as a peer. I.e. each MSA executes each step in the multi-signer process on its own. The communication is essentially reduced to a notification mechanism (“I am now in state N”), although authenticated to avoid having the contents of this communication become an attack vector for an adversary.

10.3. Multi-Signer State Transitions

For the multi-signer process semantics to be fulfilled, a new state transition in a multi-signer process is only possible when all signing DNS Providers (or their MSAs) have reached the same state.

I.e. regardless of whether each MSA traverse the finite state machine separately, or only the Leader does, and the Followers report back when they have suceeded in executing the associated Actions (as described in [I-D.draft-ietf-dnsop-dnssec-automation], they must not be further apart than one transition.

11. Responsibilities of an MSA

Each MSA has certain responsibilites, depending on supported transports and synchronization methods.

11.1. Enabling Remote MSAs to Locate This MSA

For a group of MSAs to be able to communicate securely and synchronize data for a zone, each MSA must ensure that the DNS records needed for secure communication with other MSAs are published:

* URI, SVCB and KEY records required for DNS-based communication secured by SIG(0).

* URI, SVCB and TLSA records required for API-based communication secured by TLS (if supported).

* All of the above MUST be published in a DNSSEC-signed zone under the domain name that is the identity of the MSA.

11.2. Enabling Remote MSAs to Lookup Zone Data Added By This DNS Provider

When using DNS transport between MSAs, four types of information is needed to be conveyed from one party to another:

1. Notifications (sent as DNS NOTIFY).

2. Retrieval of existing data (looked up via DNS QUERY).

3. Changes to existing data (sent as DNS UPDATE).

4. Multi-signer "state" information (sent via the Multi-Signer EDNS(0) OPT).

The second case, i.e. looking up data for a zone that is particular to a specific DNS Provider is typically about the DNSKEY RRs added by that signer or the NS RRs representing the authoritative nameservers for that DNS Provider. This is looked up under domain names constructed from the name of the served zone and the identity of the DNS Provider.

For each zone that is managed, the MSA must ensure that the data needed for synchronization with other MSAs is published:

* The DNSKEY RRset for the zone consisting of the DNSKEYS that the local signer for this DNS Provider uses to sign the zone.

* The CDS RRset for the zone, representing the KSK that the local signer uses to sign the zone (when needed).

* The NS RRs for the zone, consisting of the NS records of the authoritative nameservers that this DNS Provider is responsible for.

* All of the above MUST be published in a DNSSEC-signed zone under the domain name that is the concatenation of the zone name and the identity of the MSA. Example for the zone "zone.example" and the MSA "msa.provider":

zone.example.msa.provider. IN DNSKEY ... zone.example.msa.provider. IN RRSIG DNSKEY ... zone.example.msa.provider. IN NS ... zone.example.msa.provider. IN RRSIG NS ...

12. Migration from Single-Signer to Multi-Signer

The migration from a single-signer to a multi-signer architecture is done by adding the HSYNC RRset to the zone. However, this may be done in several steps.

12.1. Adding a single HSYNC record to an already signed zone

Adding a single HSYNC record to a zone that is already signed by the DNS provider "provider.com" with NSMGMT=OWNER is a no-op that does not change anything:

zone.example. IN HSYNC ON AGENT YES msa.provider.com.

The zone was already signed by the DNS provider "provider.com" and the provider added any needed DNSSEC records, including DNSKEYs. The zone NS RRset was managed by the zone owner. All of this is unchanged by the addition of the HSYNC RRset.

12.2. Changing the HSYNC NSMGMT Field from AGENT To OWNER

In a multi-signer architecture each MSA publishes the data it contributes to the zone under the domain name {zone}.{identity}. I.e. the zone DNSKEYs that the MSA msa.provider. uses are published as

zone.example.msa.provider. DNSKEY ... zone.example.msa.provider. DNSKEY ...

Likewise, the NS records for the zone are published as

zone.example.ns.msa.provider. NS ... zone.example.ns.msa.provider. NS ...

To migrate from "owner maintained" NS RRset to "MSA maintained", the zone owner must verify that the NS RRset as published by the MSA is correct and in sync with the NS RRset as published by the zone owner itself. After this verification the zone owner changes the HSYNC NSMGMT field in the existing HSYNC record from NSMGMT=OWNER to NSMGMT=AGENT.

12.3. Migrating from a Multi-Signer Architecture Back to Single-Signer.

If, for some reason, a zone owner wants to migrate back to a single-signer architecture, the process is essentially the reverse of the migration from single-signer to multi-signer:

1. The zone owner offboards all MSAs but one (the one that will be the single-signer)

2. The zone owner must verify that the NS RRset it publishes (in the unsigned zone) is correct and in sync with the NS RRset as published by the remaining MSA.

3. The zone owner changes the HSYNC NSMGMT field in the HSYNC record from NSMGMT=MSA to NSMGMT=OWNER.

The zone is now essentially back to a single-signer architecture. The remaining HSYNC record may be removed from the zone.

TO BE REMOVED BEFORE PUBLICATION: # Rationale

12.4. Choice of the HSYNC Mnemonic

Initially the mnemonic MSIGNER was used for the HSYNC RRset. However, as work progressed it became clear that we want also non-signing DNS Providers to be able to participate. So the RRset is a signalling mehanism from zone owner to DNS Providers, some of which may or may not be instructed to sign the zone. Therefore we suggest the mnemonic HSYNC to indicate that this is a mechanism for "horizontal synchronization" inside a zone.

But the mnemonic chosen is a very minor point and should a better suggestion come up it would be great.

12.5. Separation of MSA and COMBINER

It is possible to integrate all three multi-signer components (SIGNER, MSA and COMBINER) into a single piece of software (or two pieces, depending on the preferred way of slicing the functionality). However, such a composite module would be a fairly complex piece of software. This document aims to describe the functional separation of the different components rather than make a judgement on software design alternatives. Hence possible implementation choices are left to the implementer.

13. Security Considerations

Multi-signer is a complex system with a number of components and a significant amount of automation. The authors believe that the only way to make a multi-signer architecture useful in practice is via automation. However, automation is a double-edged sword. It can both make the system more robust and more vulnerable.

While all communication between MSAs is authenticated (either via SIG(0) signatures ore TLS), the signalling from the zone owner to the MSAs is via the HSYNC RRset in an unsigned zone. This is a potential attack vector. However, securing zone transfers from zone owner to DNS providers is a well-known issue with lots of existing solutions (TSIG, zone transfer via a secure channel, zone transfer-over-TLS, etc). Employing some of these solutions is strongly recommended.

From a vulnerability point-of-view this architecture introduces several new components into the zone signing and publication process. In particular the COMBINER and the MSAs are new components that need to be secure. The COMBINER has the advantage of not having to announce its location to the outside world, as it only needs to communicate with internal components (the zone owner, the signer and the MSA).

The MSAs are more vulnerable. They need to be discoverable by other MSAs and hence they are also discoverable by an adversary. On the other hand, the MSAs are not needed for a new zone to signed and published, they are only needed when there are changes that require the MSAs to synchronize, which is an infrequent event. Furthermore, should an MSA be unable to fulfill its role during the execution of a multi-signer process, the multi-signer process will simply stop where it is. Regardless of where the stop (or rather pause) occurs, the zone will be fully functional and once the MSA is able to resume its role, the multi-signer process will continue from where it left off.

14. IANA Considerations.

Note to the RFC Editor: In this section, please replace occurrences of "(This document)" with a proper reference.

14.1. HSYNC RR Type

IANA is requested to update the "Resource Record (RR) TYPEs" registry under the "Domain Name System (DNS) Parameters" registry group as follows:

Type HSYNC

Value TBD

Meaning Zone owner designation of DNS providers enabling mutual discovery

Reference (This document)

14.2. New Multi-Signer EDNS Option

This document defines a new EDNS(0) option, entitled "Multi-Signer", assigned a value of TBD "DNS EDNS0 Option Codes (OPT)" registry

TO BE REMOVED UPON PUBLICATION: https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-11 (foo)

Value	Name	Status	Reference
TBD	Multi-Signer	Standard	(This document)

14.3. A New Registry for EDNS Option Multi-Signer Operation Codes

The Multi-Signer option also defines an 8-bit operation field, for which IANA is requested to create and mainain a new registry entitled "Multi-Signer Operations", used by the Multi-Signer option. Initial values for the "Multi-Signer Operations" registry are given below; future assignments in in the 3-127 range are to be made through Specification Required review [BCP26].

OPERATION	Mnemonic	Reference
0	forbidden	(This document)
1	HELLO	(This document)
2	HEARTBEAT	(This document)
3-127	Unassigned	(This document)
128-255	Private Use	(This document)

15. References

15.1. Normative References

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <https://www.rfc-editor.org/info/rfc1996>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <https://www.rfc-editor.org/info/rfc2136>.

[RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <https://www.rfc-editor.org/info/rfc2931>.

[RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <https://www.rfc-editor.org/info/rfc3007>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <https://www.rfc-editor.org/info/rfc6891>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8901] Huque, S., Aras, P., Dickinson, J., Vcelak, J., and D. Blacka, "Multi-Signer DNSSEC Models", RFC 8901, DOI 10.17487/RFC8901, September 2020, <https://www.rfc-editor.org/info/rfc8901>.

15.2. Informative References

[BCP26] Best Current Practice 26, <https://www.rfc-editor.org/info/bcp26>. At the time of writing, this BCP comprises the following:

Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[I-D.draft-berra-dnsop-keystate]
Bergstrm, E., Fernandez, L., and J. Stenstam, "Signalling Key State Via DNS EDNS(0) OPT", Work in Progress, Internet-Draft, draft-berra-dnsop-keystate-01, 7 February 2025, <https://datatracker.ietf.org/doc/html/draft-berra-dnsop-keystate-01>.

[I-D.draft-ietf-dnsop-dnssec-automation]
Wisser, U., Huque, S., and J. Stenstam, "DNSSEC automation", Work in Progress, Internet-Draft, draft-ietf-dnsop-dnssec-automation-03, 19 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-dnssec-automation-03>.

Appendix A. Change History (to be removed before publication)

* draft-leon-distributed-multi-signer-00

  Initial public draft.

Authors' Addresses

Leon Fernandez
The Swedish Internet Foundation
Sweden
Email: leon.fernandez@internetstiftelsen.se

Erik Bergstrm
The Swedish Internet Foundation
Sweden
Email: erik.bergstrom@internetstiftelsen.se

Johan Stenstam
The Swedish Internet Foundation
Sweden
Email: johan.stenstam@internetstiftelsen.se

Steve Crocker
Edgemoor Research Institute
United States
Email: steve@shinkuro.com