## **DNSSEC Key Ceremonies** standardising and automating key security **Benno Overeinder** Berry van Halderen Roland van Rijswijk-Deij





# Project rationale

- DNSSEC has seen widespread adoption over the past decade
- Almost all top-level domains are now signed
- High-value domains (such as TLDs) **need** strong key protection
- Often use HSMs to protect key material

ccTLD DNSSEC Status on 2020-09-14









- HSMs for high-value domains are often airgapped for additional security
- Signing with keys in air-gapped HSMs requires a process or ceremony
- E.g. done for the DNS Root (organised by IANA)
- Ceremonies are sometimes witnessed by community representatives or stakeholders

## Ceremonies





source: <u>https://www.iana.org/dnssec/ceremonies</u>







# Ceremony requirements & design

- Until now, these **key ceremonies** are often bespoke in terms of process and tooling
- Making standardised guidelines can help the community; we documented this
- Ensure secure ceremonies and help automate the process

● < > 🗉 🔘 🗊 🚍 🔒 github.com	C <u>0</u> <u>1</u> <u>1</u>
Search or jump to / Pull requests Issues Marketplace Explore	Ļ +- 🍾
NLnetLabs / dnssec-ceremony-doc	Unwatch - 16 ☆ Star 0 약 Fork 2
🕻 > Code 🕛 Issues 🏦 Pull requests 🕞 Actions 🛄 Projects 🛄 Wiki 🕛 Sec	curity 🖂 Insights 🔯 Settings
<sup>3</sup> master - dnssec-ceremony-doc / CEREMONY.md	Go to file
<b>rijswijk</b> High level description of ceremony building blocks	Latest commit 59f588b on Nov 3 🕤 History
२२ 1 contributor	
DNSSEC Key Signing Suite Documentation Key Signing Ceremonies: Requirements, Options and Implem	nentation
229 Lines (142 sloc) 28.3 KB DNSSEC Key Signing Suite Documentation Key Signing Ceremonies: Requirements, Options and Implem Copyright (c) 2019-2020 NLnet Labs	nentation
229 Lines (142 stoc) 28.3 KB DNSSEC Key Signing Suite Documentation Key Signing Ceremonies: Requirements, Options and Implem Copyright (c) 2019-2020 NLnet Labs Released under Creative Commons CC 4.0 BY-SA (see LICENSE)	nentation
229 Lines (142 stoc) 28.3 KB DNSSEC Key Signing Suite Documentation Key Signing Ceremonies: Requirements, Options and Implem Copyright (c) 2019-2020 NLnet Labs Released under Creative Commons CC 4.0 BY-SA (see LICENSE) Punding Acknowledgement Work on the DNSSEC Key Signing Suite was supported by a grant from the European Comm Foundation (NGIO PET project).	nentation
229 Lines (142 stoc)       28.3 KB         DNSSEC Key Signing Suite Documentation         Key Signing Ceremonies: Requirements, Options and Implem         Copyright (c) 2019-2020 NLnet Labs         Released under Creative Commons CC 4.0 BY-SA (see LICENSE)         Funding Acknowledgement         Work on the DNSSEC Key Signing Suite was supported by a grant from the European Common Common (NGI0 PET project).         About this document	nentation mission managed by the NLnet
229 Lines (142 stoc)       28.3 KB         DNSSEC Key Signing Suite Documentation         Key Signing Ceremonies: Requirements, Options and Implem         Copyright (c) 2019-2020 NLnet Labs         Released under Creative Commons CC 4.0 BY-SA (see LICENSE)         Punding Acknowledgement         Work on the DNSSEC Key Signing Suite was supported by a grant from the European Comr Foundation (NGI0 PET project).         About this document         This document is intended for operators of high value domains that deploy, or intend to dep security measures to protect their DNSSEC key material. In particular, this document is intended for so-called "offline Key Signing Keys", in which the root signing key in a domain is stored possibly even "air gapped".	mission managed by the NLnet ploy DNSSEC, and that want to take ended for setting up protection measures away from the DNSSEC signing system, or









- Better standardisation enables better automation
- We introduce the concept of **a "recipe"**; coherent set of instructions for key ceremony automation in the secure **environment** with the air-gapped HSM
- Built the tools to execute recipes and prototype integration with **OpenDNSSEC** signer

# Automation with recipes



•			0 =	🔒 github.com	Ċ	• t 7 +
	An examp	le of a reci	pe with the most commo	action types is given below:		
	{	"recipes "recipe" {	<pre>SpecVersion": "v1.0", ': "actions": [</pre>	<pre>': "haveKey", "actionParams": { ': "haveKey", "actionParams": { ': "deleteKey", "actionParams": { ': "generateKey", "actionParams": ': "produceSignedKeyset", "actionF</pre>	<pre> }, "cooked": { }, "cooked": { }, "cooked": { { }, "cooked": { Params": { }, "coo Params": { }, "coo</pre>	<pre>} }, } }, } }, } }, } , } , .ked": { } }, .ked": { } },</pre>
	}	1				

The parameters for each of the action types are specified further down

### Specifying key

The recipe specification is designed to be flexible in how keys are specified, in particular keys that are part of a keyset that needs to produceSignedKeyset action. Two models are supported, a model in which all keys, including ZSKs, are generated in , and a model in which some keys are generated outside of the "bunker", but need to be included in the signed keyse This means that there are also two ways to specify keys in action parameter sets; by reference, or direct. The two ways to specify keys are shown in detail below

### Key by reference:

This way of referring to keys may be used in haveKey, deleteKey, generateKey and produceSignedKeyset actions.

Keys are specified as follows:

"keyType": "byRef" "keyAlgo": INTEGER,







# Cooking the recipe

- Complexity in creating recipe, not executing it! No complex actions undertaken during ceremony.
- 3 tools for the ceremony
  - generate recipe
  - process in secure environment
  - **export** results into operational environment

version: 1	
repositorie	es:
Bunker: 8	<pre>cprimary</pre>
module:	/usr/lib/softhsm/libsofthsm2.
label:	Bunker
pin:	1234

Sample configuration file

\$ OKS (	COOK	
Recipe	Testing	generated at 2020-11-27 09
Recipe	step 1:	Process key used in migrat
Recipe	step 2:	generateKey
Recipe	step 3:	Generation key used for ne
Recipe	step 4:	Export key hex 521c5afa8ce
Recipe	step 5:	produceSignedKeyset
Recipe	step 6:	produceSignedKeyset
Recipe	step 7:	deleteKey
Recipe	complete	ed.









```
actionType: produceSignedKeyset
actionParams:
 ownerName: nl
  inception: 2020-11-27 09:59:07
  expiration: 2020-12-27 09:58:07
  ttl: 60
  keyset: [ {
      key: {
        keyType: byRef
        keyID: hex 4556957b8ea06427974a50973d5d0d31
        keyFlags: KSK
        keyAlgo: "8"
      key: {
        keyType: byRef
        keyID: hex 521c5afa8ce4cc2fde07bd9d40f77b3e
        keyFlags: ZSK
        keyAlgo: "8"
  signedBy: [ {
      key: {
        keyType: byRef
        keyID: hex 4556957b8ea06427974a50973d5d0d31
        keyFlags: KSK
        keyAlgo: "8"
      } }
```

# Before and after cooking

```
actionType: produceSignedKeyset
actionParams:
 ownerName: nl
 inception: 2020-11-27 09:59:07
  expiration: 2020-12-27 09:58:07
 ttl: 60
  keyset: [ {
cooked:
                         DNSKEY 257 3 8 AwEAAb5si0v8pv0pY
   nl.
         60
                 ΙN
                 ΙN
                         DNSKEY 256 3 8 AwEAAbmwnNpRAIUFo
         60
   nl.
                                 DNSKEY 8 1 60 20201227095
   nl.
         60
                 ΙN
                         RRSIG
    . . .
```



# Producing the recipe

- **Recipe can be produced** entirely **beforehand**, w/o need for observers
- The tool supports actually preproducing recipes
- The kasp section specifies the key and signing policy
- Prototype has a full set of features to support ceremonies, with **limitations on** how these ceremonies are structured

\$ oks -c oks.conf produce example.com 2021-12-31 "Keysets for year 2021"

version: 1 repositories: Operational: module: /usr/lib/softhsm/libsofthsm2.so label: OKS 1234 pin: kasp: P3D refresh: P1M validity: inceptionoffset: PT3600S ttl: 60 ksk: algo: 8 size: 2048 lifetime: 1Y zsk: algo: 8 size: 1024 lifetime: 1M transport: key: label: recipekey size: 2048













# **Consume and ODS integration**

- Output a set of **ZSK keys** that need to become active over time
- **Consume** processing **split** in **two** 
  - all **new keys** will be **imported** into the **HSM**
  - with specific time, it will produce the signed **keyset** appropriate for that time
- Integration w/ OpenDNSSEC
  - **ODS** has separate **signer** and **enforcer** components
  - produced output is ready-to-use signer configuration (being ODS or potential other signing solutions like BIND or Knot DNS)

\$ oks consume

### \$ oks consume 202102010000











## Future work

## Get community feedback!

- If an interest exists, take recipe API to IETF for standardisation
- Adoption by other OSS DNSSEC implementers

### About this document

This document specifies the file format for DNSSEC Key Signing Ceremony Recipes. A recipe is a single document that specifies all of the actions that must be performed in the secure environment in which offline key material is stored and in which the key signing ceremony takes place. The recipe file format is used by the tools included in the DNSSEC Key Signing Suite.

a github.com

## Terminology

We foresee this document potentially evolving into an Internet draft; for this reason, this document uses the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY" and "MAY NOT" in accordance with the guidelines set out in RFC 2119 [1].

### Introduction

The goal of the DNSSEC Key Signing Suite is to support owners of high value domains (such as top-level domains) with key ceremonies that rely on offline







# Further reading

## • We wrote a blog about the project: https://blog.nlnetlabs.nl/supporting-dnssec-key-signing-ceremonies/











# Thank you! Questions?

- https://nlnetlabs.nl/
- **9** @nlnetlabs
- labs@nlnetlabs.nl



