DNSSEC Key Ceremonies

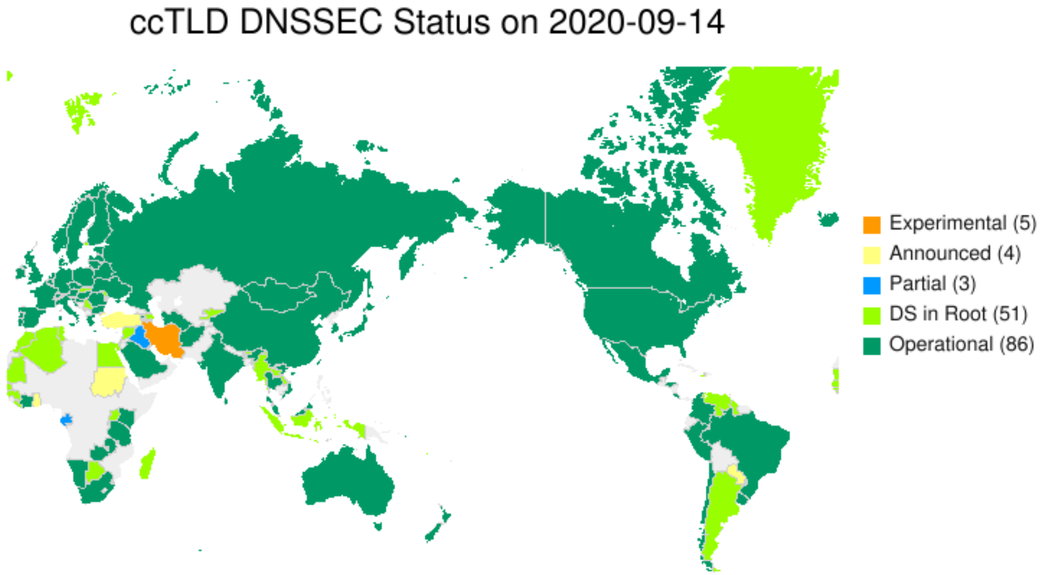
standardising and automating key security

Benno Overeinder
Berry van Halderen
Roland van Rijswijk-Deij

NLNETLABS

Project rationale

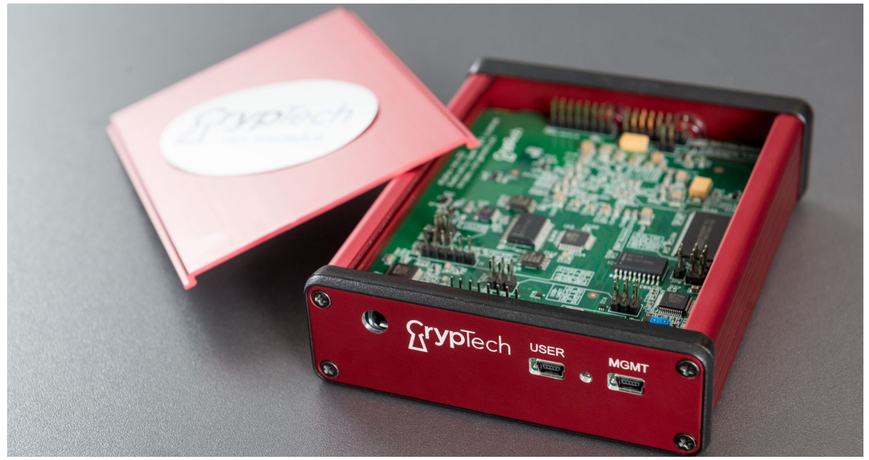
- **DNSSEC** has seen **widespread adoption** over the past decade
- Almost **all top-level domains** are now signed
- High-value domains (such as TLDs) **need strong key protection**
- **Often use HSMs** to protect key material

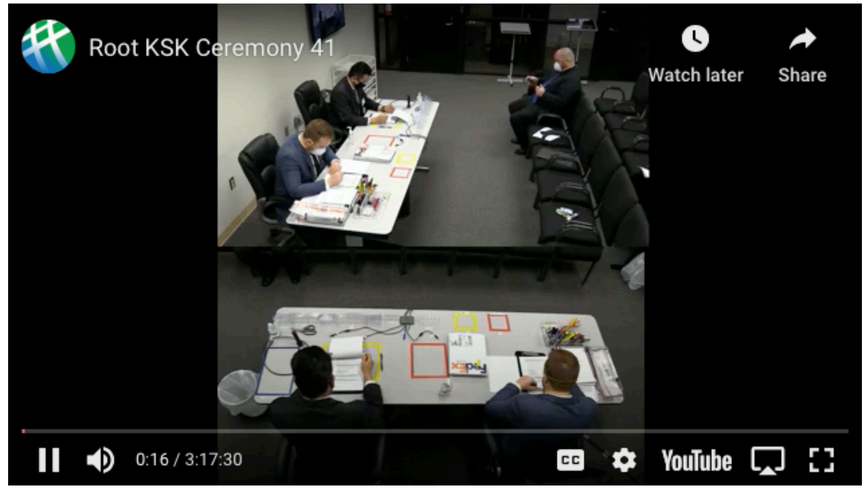


NLNETLABS

Ceremonies

- **HSMs** for high-value domains are often **air-gapped 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



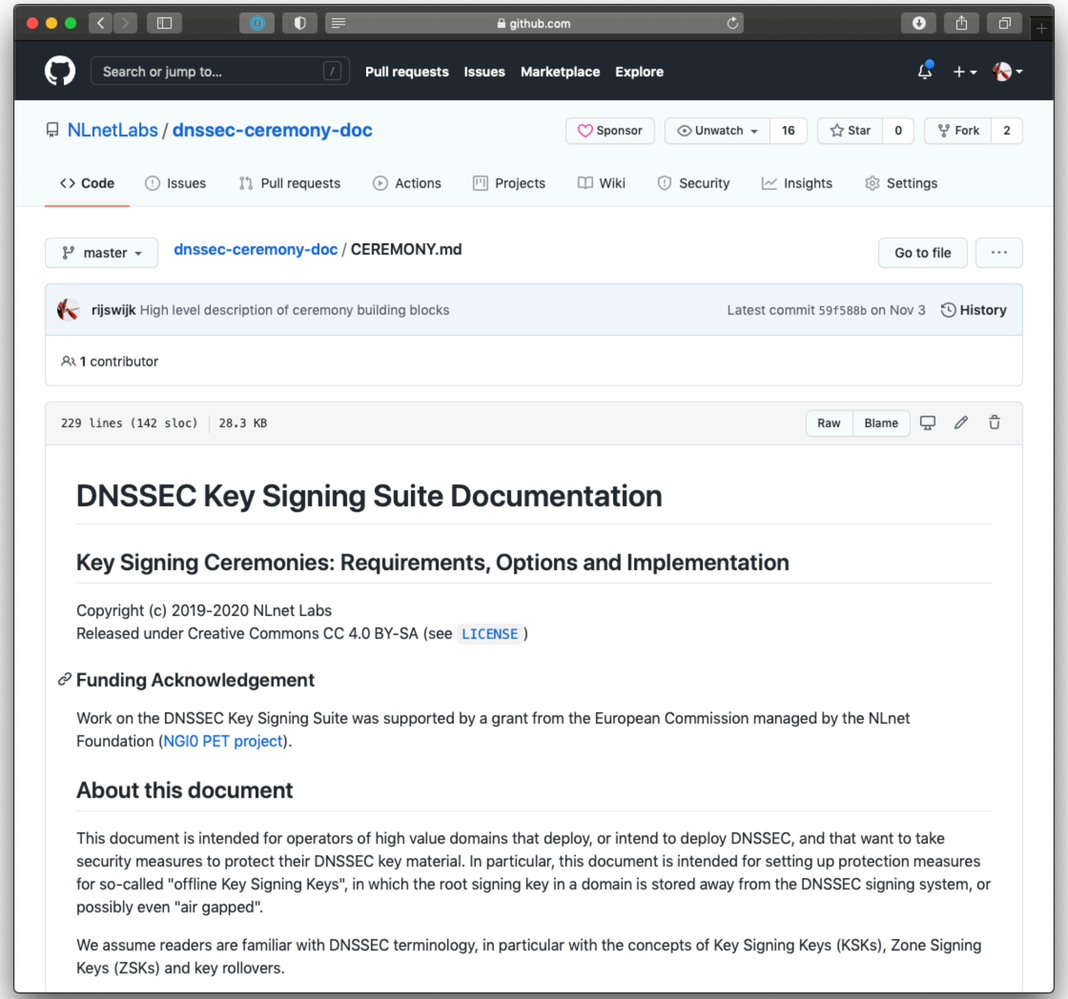


source: https://www.iana.org/dnssec/ceremonies

NLNETLABS

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



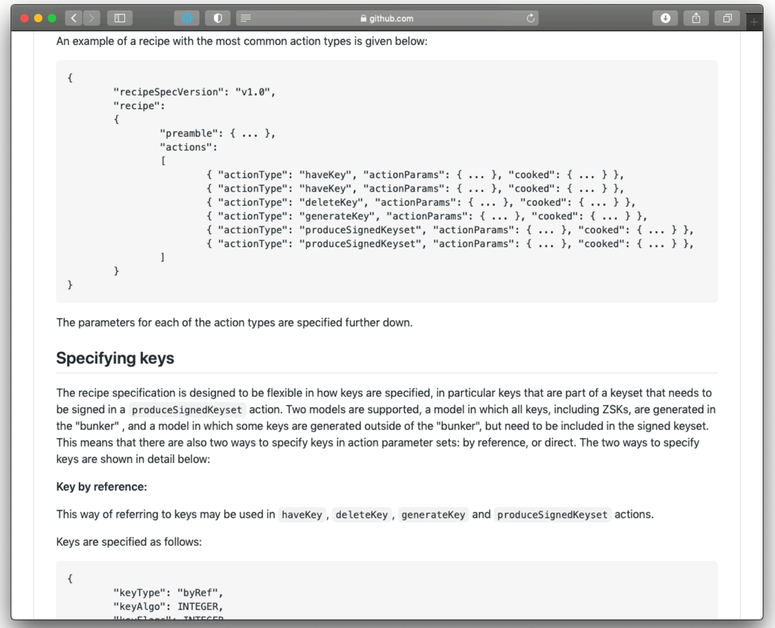
NLNETLABS

Automation with recipes

- **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



NLNETLABS

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
repositories:
  Bunker: &primary
    module: /usr/lib/softhsm/libsofthsm2.so
    label:  Bunker
    pin:    1234
```

Sample configuration file

```
$ oks cook
Recipe Testing generated at 2020-11-27 09:5
Recipe step 1: Process key used in migratio
Recipe step 2: generateKey
Recipe step 3: Generation key used for next
Recipe step 4: Export key hex 521c5afa8ce4c
Recipe step 5: produceSignedKeyset
Recipe step 6: produceSignedKeyset
Recipe step 7: deleteKey
Recipe completed.
```

NLNETLABS

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: [ {
        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"
        } }
    ]
  }
}
```

```
{ actionType: produceSignedKeyset
  actionParams:
  { ownerName: nl
    inception: 2020-11-27 09:59:07
    expiration: 2020-12-27 09:58:07
    ttl: 60
    keyset: [ {
        ...
  }
  cooked:
  {
      '''
      nl.   60      IN      DNSKEY  257 3 8 AwEAAb5si0v8pvOpY
      nl.   60      IN      DNSKEY  256 3 8 AwEAAbmwnNpRAIUFo
      nl.   60      IN      RRSIG   DNSKEY 8 1 60 20201227095
      '''
  }
```

NLNETLABS

Producing the recipe

- **Recipe can be produced** entirely **beforehand**, w/o need for observers
- The **tool supports** actually **pre-producing 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
    pin:    1234
kasp:
  refresh:          P3D
  validity:         P1M
  inceptionoffset: PT3600S
  ttl: 60
  ksk:
    algo: 8
    size: 2048
    lifetime: 1Y
  zsk:
    algo: 8
    size: 1024
    lifetime: 1M
transport:
  key:
    label: recipekey
    size: 2048
```

NLNETLABS

Consume and ODS integration

- **Output** a set of **ZSK keys** that need to become active over time

```
$ oks consume
```

- **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

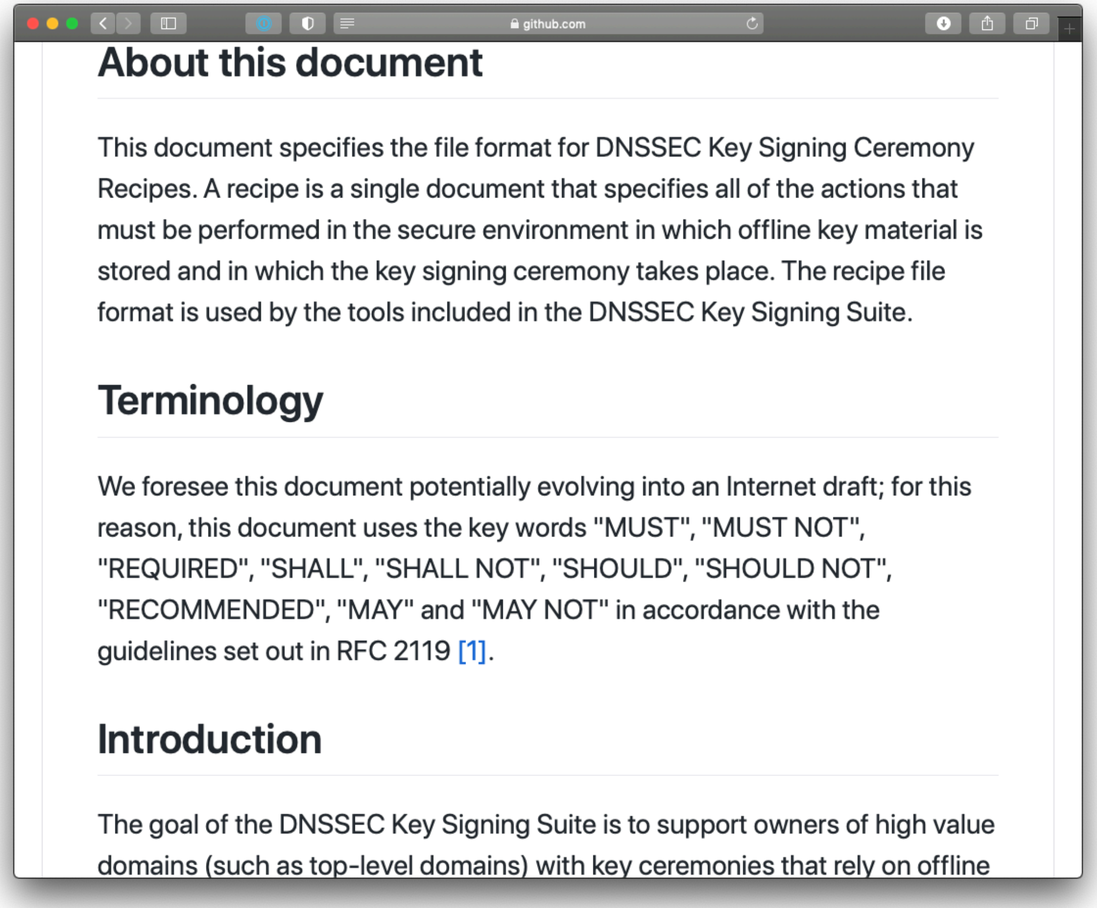
```
$ oks consume 202102010000
```

- **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)

NLNETLABS

Future work

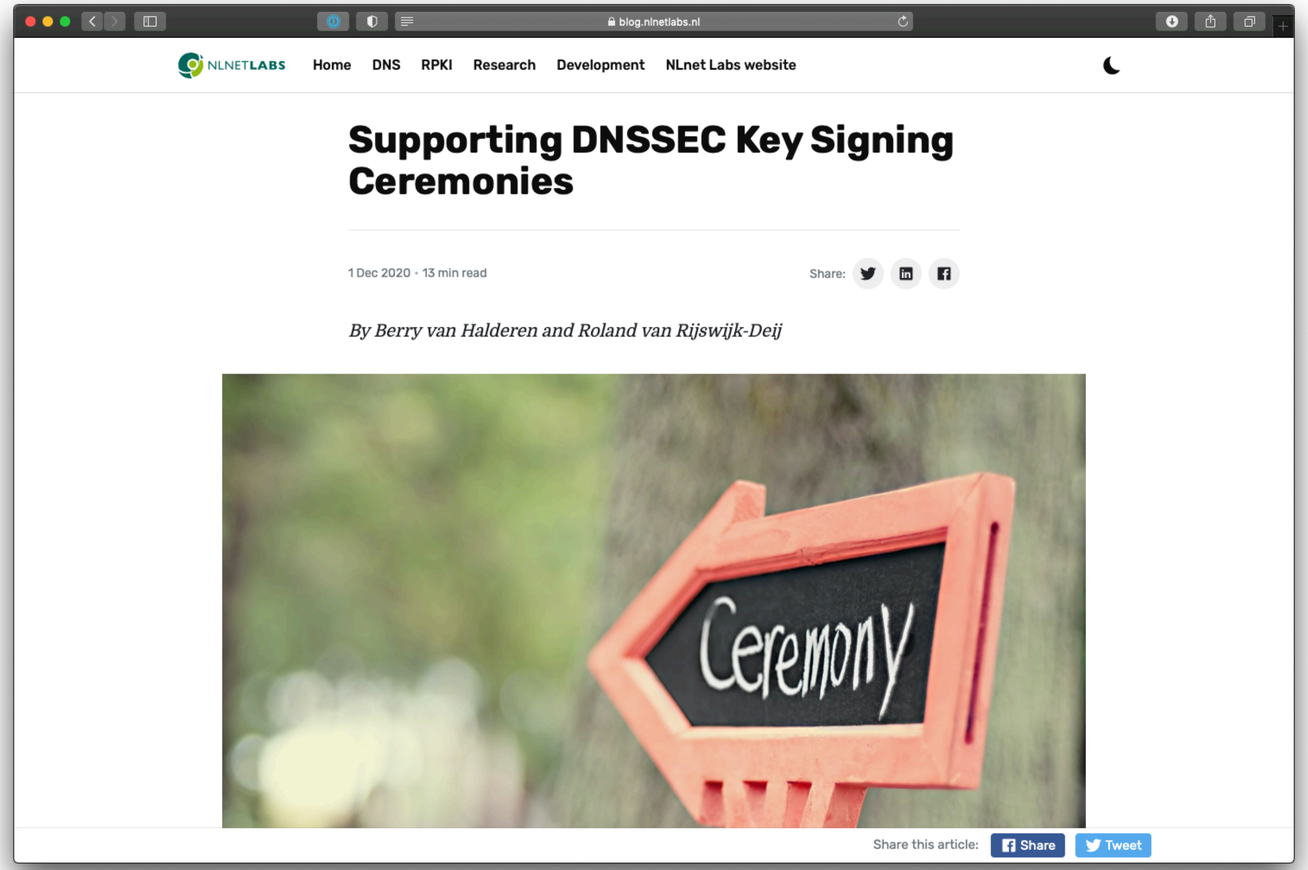
- **Get community feedback!**
- If an interest exists, **take recipe API to IETF** for standardisation
- **Adoption by other** OSS **DNSSEC implementers**



NLNETLABS

Further reading

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



Thank you! Questions?

https://nlnetlabs.nl/

@nlnetlabs

labs@nlnetlabs.nl

NLNETLABS