



Update of the HU Registry DNS Zone Generation and Signer Systems

ISZT.hu

nic.hu - HU cctld hostmaster@nic.hu

**Pásztor Miklós
Zsakó János
Csillag Tamás**

RIPE89, Prague, 2024-10

Disclaimer

My main employer is PCH.net and this is a project based work at the Hungarian cctld with permission from my employer.

my PCH email address: tom@pch.net

my HU cctld mail address: cstamas@nic.hu

Goals - hidden primary

Operate both in a secure and a resilient manner.
Use a kind of "cluster" to handle single failures.

Try to follow best practices

Try to keep it simple (to operate and to debug)

Do checks and interrupt (in case of issues)
with a clear error message before loading the zone.

Summary

Our previous system was based on [OpenDNSSEC](#).
The old signer served us well since [2015](#).

There are newer practices and tools under active development available today and we decided to switch.

We are sharing a few details about the old system and how it got better after the upgrade.

We also learned during the process that is worth sharing with the community.

Old signer

Based on OpenDNSSEC

Zone distribution with Bind9

Hot/cold 2 nodes cluster with the active having the shared IP address

Custom scripts syncing opendnssec internal-state between the 2 sides

The 2 sides were using the same keys

- KSK - was stored on a hardware token (Nitrokey HSM)

- ZSKs - for soft-signing, stored on disk (softsm)

Only the hot side running OpenDNSSEC

To switch between the sides you needed a recent sync between the sides

The cold node needed manual intervention to become active

Old signer

NodeIP: 192.0.2.1

Shared: 192.0.2.6



Source: 192.0.2.6

NodeIP: 192.0.2.2



Locations

Both located at Budapest
5-6 kilometers apart

At 2 BIX locations:

Victor Hugo street

Dataplex (Asztalos Sándor street, by T-Systems)

Safety mechanisms (old signer)

There were scripts used to verify the zone integrity with NotifyCommand in OpenDNSSEC before the zone is loaded and distributed.

Perl/shell and awk scripts calling each other.

Parsing the raw zone file.

Pregenerated signatures for the DNSKEY rrset
(in case any issues with the hardware token)
with custom scripts, modified Idns-tools...

Check and prevent (sanity checks)

For detectable anomalies an error should be a logged and the last known good zone should remain in use.

Check and prevent (sanity checks...)

Existence of so called "golden" records:
delegations that we do not expect to go away
and they were registered a long time ago

For significant: 1% change in the number of delegations
to the previously generated zone (1% is ~9000 in 2024)

Syntax checks: named-checkzone (bind9),
(new with knot) kzonecheck

New check added with the new signer:
DNSKEY, SOA, NS records
RRSIGs verification using the parent DS

Check and alert

There is another case when only an alert is warranted and the zone should be loaded.

If any RRSIG record would expire within the refresh period, which means either software bug, misconfiguration or operational issue.

New signer system setup

Using knot as a signer

Using nsd as distribution and verification point

Hot/warm 2 nodes setup, shared nothing

No shared IP address, only node IPs, secondaries configured to use both
One of them generates odd the other even serials

19 minutes of difference between the 2 sides epoch based SOA serial (examples):

Hidden primary#1: 1729077002 (translates 13:10:02)

Hidden primary#2: 1729075861 (translates 12:51:01) <--- this side is staggered

New zone generated every 10 minutes this means if the hot side dies the secondaries switch over to the other node after 2 cycles as that serial becomes greater.



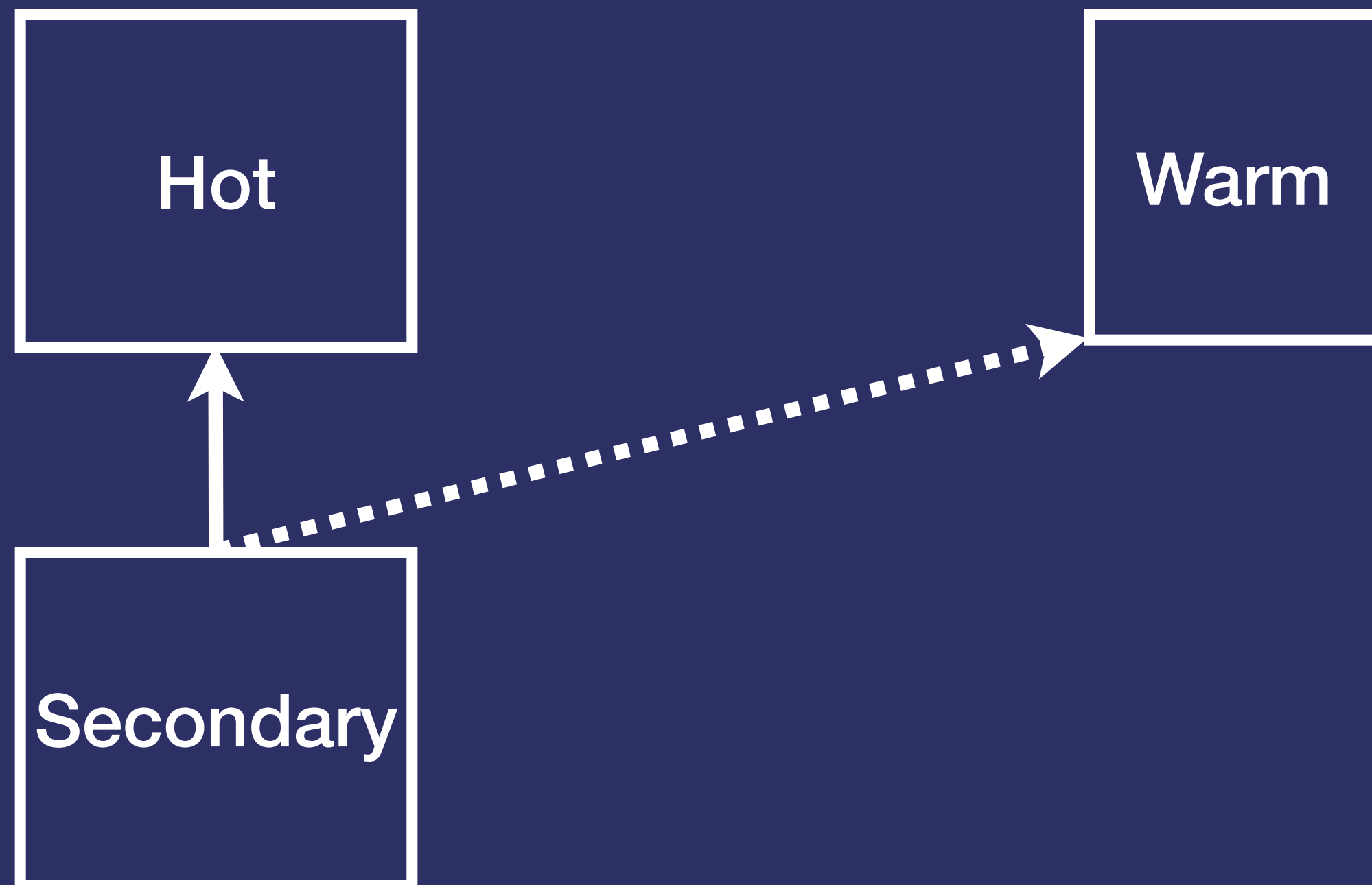
New signer

NodeIP: 192.0.2.1

Serial: 1729077002

NodeIP: 192.0.2.2

Serial: 1729075861



Source: 192.0.2.1, 192.0.2.2

nsd verification

This feature was added in 2020 (nsd v4.6)

A user provided program/script is started.
You either get the zone on STDIN
or you can query a temporary nsd instance.

To prevent zone loading the verification script
need to return with nonzero exit code.

In this case nsd keeps serving the previous zone.

knot key handling

OpenDNSSEC can use a different keystore for the KSKs and the ZSKs.

knot can use either, but not both.

To emulate the same with knot we use offline-ksk functionality

This means we have 2 config files

- one with the KSK(s) pkcs11 backend

- one with the ZSKs file backend (the default)

We pre-generate ZSKs and sign the resulting DNSKEY rrsets for a year.

We use knot's tooling to do this with a minimal shell script.

As a result the tokens are rarely used.

nsd verification

During verification:

```
# nsd-control zonestatus hu
```

```
zone: hu
```

```
state: ok
```

```
served-serial: "1729543802 since 2024-10-21T22:50:30"
```

```
commit-serial: "1729544402 since 2024-10-21T23:00:28"
```

```
wait: "3226 sec between attempts"
```


nsd verification...

We have rewritten our checks to multiple short Perl script Net::DNS, example (excerpt) this checks if records listed in a file have existing NS records:

```
my $errcount;
while (<$fh>) {
    next if m/^#/;
    m/^(\\S+)/;
    my( $ret ) = query( $ENV{VERIFY_IPV4_ADDRESS},
                       $ENV{VERIFY_IPV4_PORT},
                       $1, 'NS' );
    $errcount++ unless $ret;
}
exit 1 if $errcount;
say "All golden records found";
```

nsd ixfr issue

The nsd version available in stable Debian 12 is at 4.6.1 which have a bug
It corrupts outgoing ixfr messages when multiple ixfr messages are received

knot sent such messages once it started re-signing about to expire records
In our case that was 2 weeks into production

Backporting 4.9.1 (available in Debian sid) resolved the issue
<https://github.com/NLnetLabs/nsd/pull/300>

Incident - nsd verification issue

date: 2024-10-22

Node1 (preferred side): nsd hanged after successful verification and no new zone was loaded at 03:51


(Possibly an issue with during nsd IPC communication)

Node2's serial "won" 20 minutes later and secondaries switched over automatically recovering at 04:11.

(Node1 "woke up" an hour later at 04:51 and resumed normal operations.)

NLnet Labs is working on a fix and the next release should have it:

<https://github.com/NLnetLabs/nsd/issues/338>



Thanks,
and questions?

extra slides

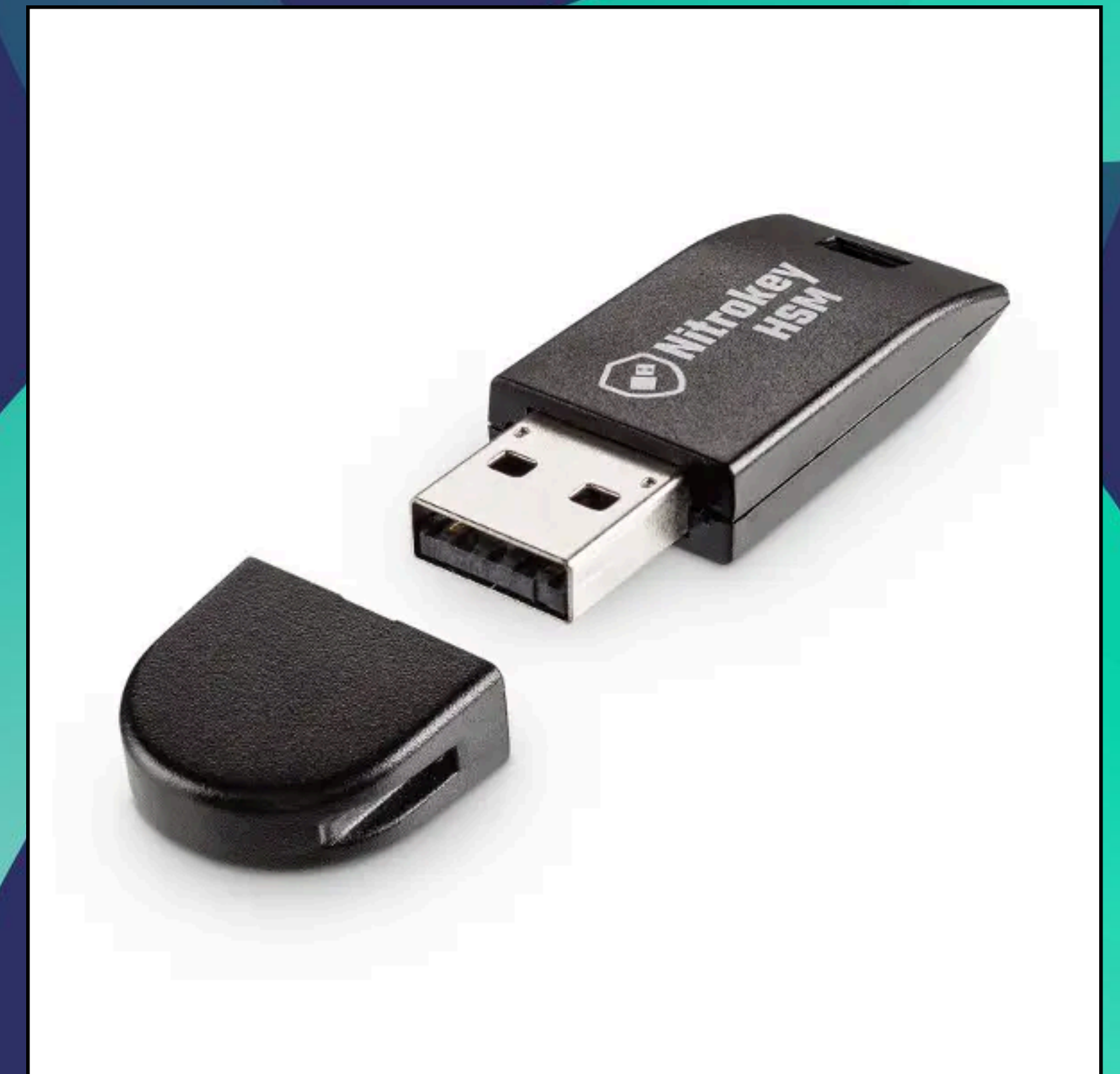
Nitrokey HSM 2

They are slow, but okay for KSKs.

As it only needs to sign the DNSKEY rrset.

Protects against someone walking away with a private key.

We have 2 of these with the same keys loaded.



KSK rollover to the new signer (DS)

Depending on the registry or registrar you may need to add the new KSK to the DNSKEY rrset first

IANA standard process (this means: TLDs) requires this before adding a new DS
TLDs (registries and in turn registrars) might ask for a published KSK the same way

We removed the old DNSKEY after the DS was added to keep the DNSKEY rrset as small as possible.

KSK rollover to the new signer (transition)

Choose a time window when no ZSK rollover is scheduled (or freeze rollovers)

Add the new signer's ZSK to the DNSKEY rrset on the old signer

To enable rollback to the old signer you should also

add the old signer's ZSK to the DNSKEY rrset on the new signer

You can freely move to the new (and even back to the old signer) as necessary

Once the transition is over the old DS and the extra DNSKEY record can be removed

Zonemd generation and check

We configured knot to generate a zonemd and on our request the knot developers added an option to kzonecheck to check the zonemd of a zonefile.

https://gitlab.nic.cz/knot/knot-dns/-/merge_requests/1645

We added this to our monitoring to AXFR from secondaries (those that we operate) once an hour.

systemd controversy

we do not use systemd because of various interferences
(unit file security "features", systemd-resolved taking port 53)

because stability and operational simplicity

systemd makes running dns software more complicated reducing stability

pkcs11 use for example would require tweaks

apparmor implements most "security" features without systemd

on a debian server sysvinit is viable

(on desktop it is more complicated...)