DS Updates and Multi-Signer Coordination – A Continuing Series ICANN 72, “San Juan” – Episode 7

Steve Crocker & Shumon Huque

steve@shinkuro.com

shuque@gmail.com

Two gaps in the DNSSEC protocol specs

- Automation of DS updates
  - Periodic key changes
  - New key in the child's zone requires new parent DS record
  - Registrar has access to parent
    - If Registrar is providing signed DNS service, conveying new DS to parent is easy
  - **But 3rd party DNS provider does not have access to the Registry**

- Multiple DNS Providers
  - Each DNS provider signs with its own keys (RFC 8901 Model 2)
  - Each must include ZSKs from the other providers
  - No defined way to share the keys
  - Needed for:
    - **Capacity and high reliability**
    - **Glitch-free transfer of a signed zone from one DNS Provider to another (Disruptions can be worse than expected)**

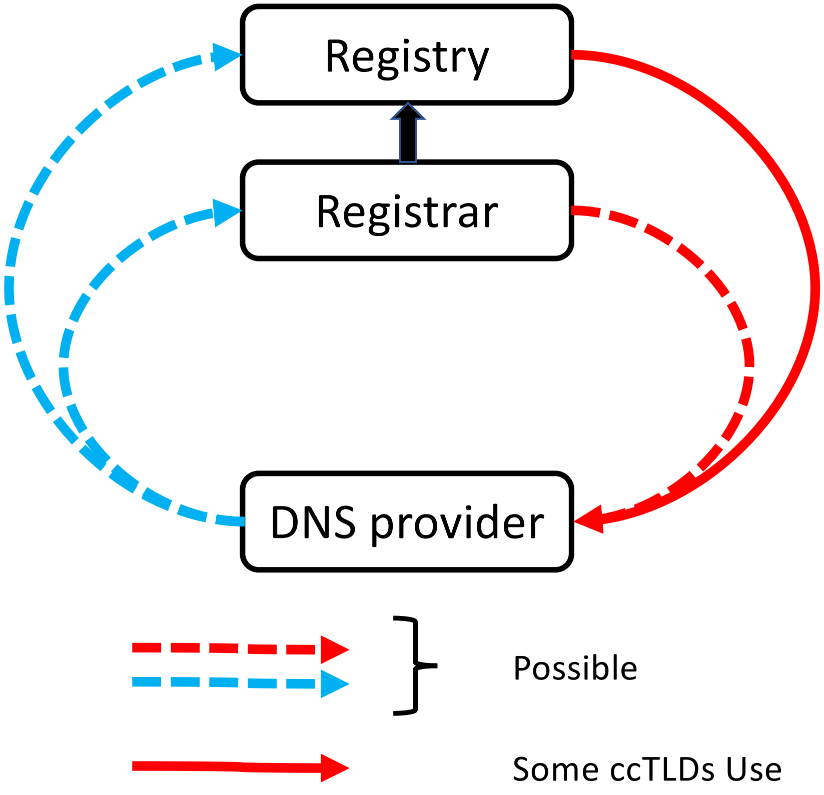
Agenda

#	Title	Speaker
3.1	Overview: DNSSEC Provisioning Automation	Steve Crocker, Shinkuro, Inc.
3.2	GoDaddy CDS Support Update	Brian Dickson, GoDaddy
3.3	CSYNC implementation	Ulrich Wisser, Swedish Internet Foundation
3.4	Authenticated Bootstrapping of DNSSEC Delegations	Nils Wisiol, deSEC, Technische Universität Berlin
3.5	SSAC DS Automation Work Party	Steve Crocker, Shinkuro, Inc.
3.6	Making MUSIC with DNSSEC	Johan Stenstam, Roger Murray, Swedish Internet Foundation
3.7	RFC Adjustments for Multi-Signer	Shumon Huque, Salesforce
3.8	DNS(SEC) Views	P.F. Tehrani, E. Osterweil, T.C. Schmidt, M. Wählisch, Weizenbaum Institute / Fraunhofer FOKUS
3.9	Q & A	Everyone

DS Updates

Possible Ways to Convey the DS key from 3rd party DNS Provider

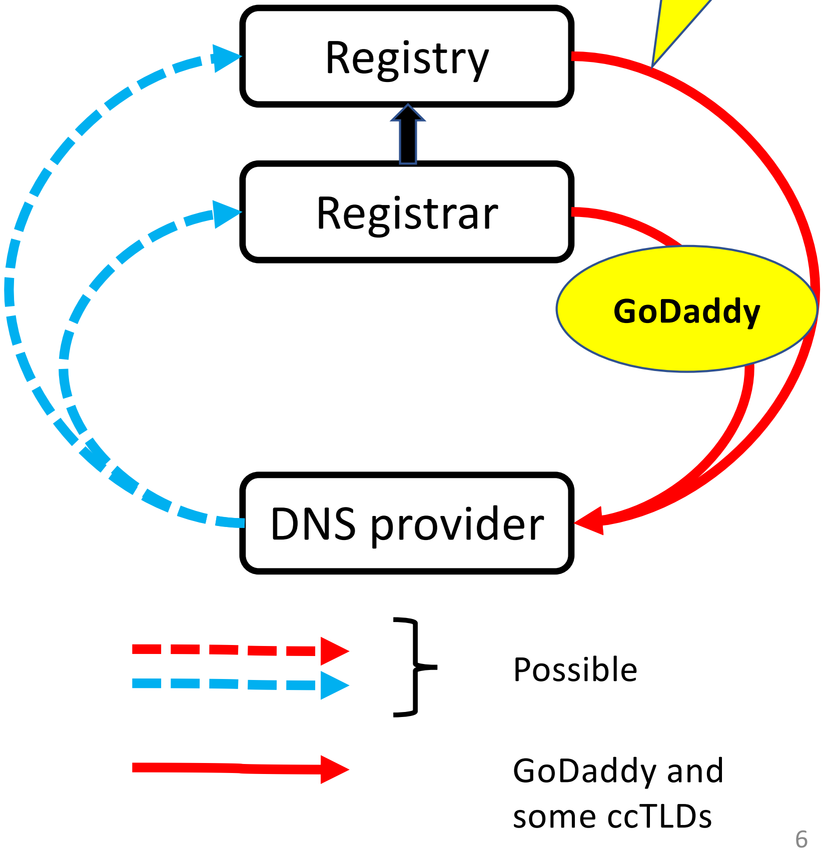
	Direction	
Upper Side	Push (Calling) DNS Provider calls API at Ry, Rr	Pull (Polling) DNS Provider publishes CDS and/or CDNSKEY
Registry	1. Requires API	3. RFC 8078
Registrar	2. Requires API	4. RFC 8078



Possible Ways to Convey the DS key from 3rd party DNS Provider

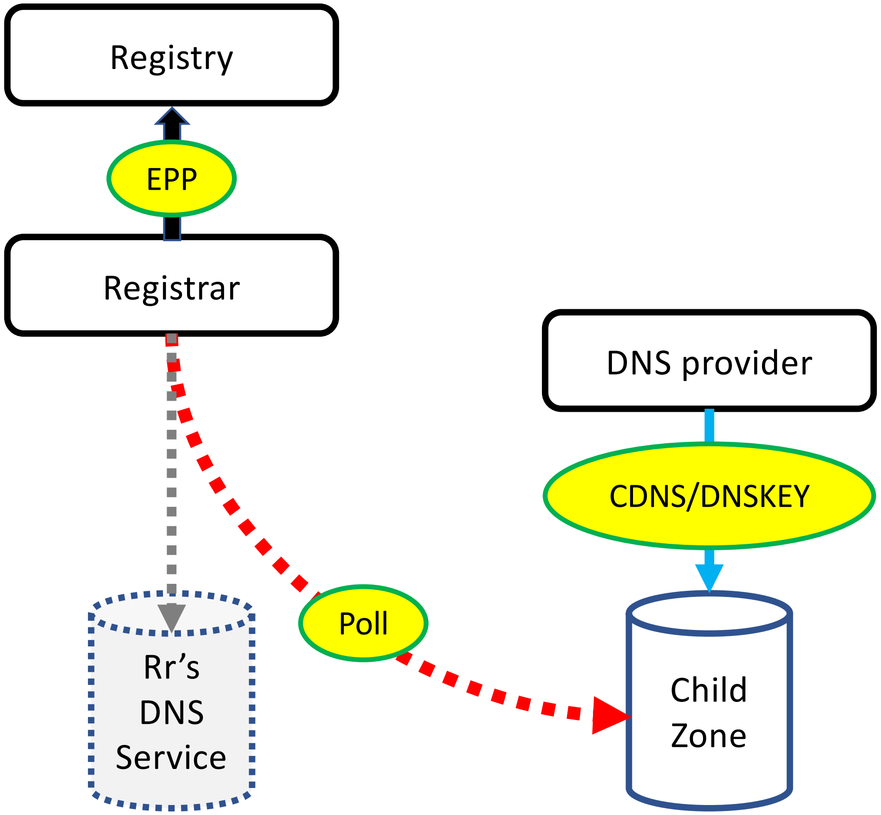
	Direction	
Upper Side	Push (Calling) DNS Provider calls API at Ry, Rr	Pull (Polling) DNS Provider publishes CDS and/or CDNSKEY
Registry	1. Requires API	3. RFC 8078
Registrar	2. Requires API	4. RFC 8078

Now on the maps



Possible Ways to Convey the DS key from 3rd party DNS Provider

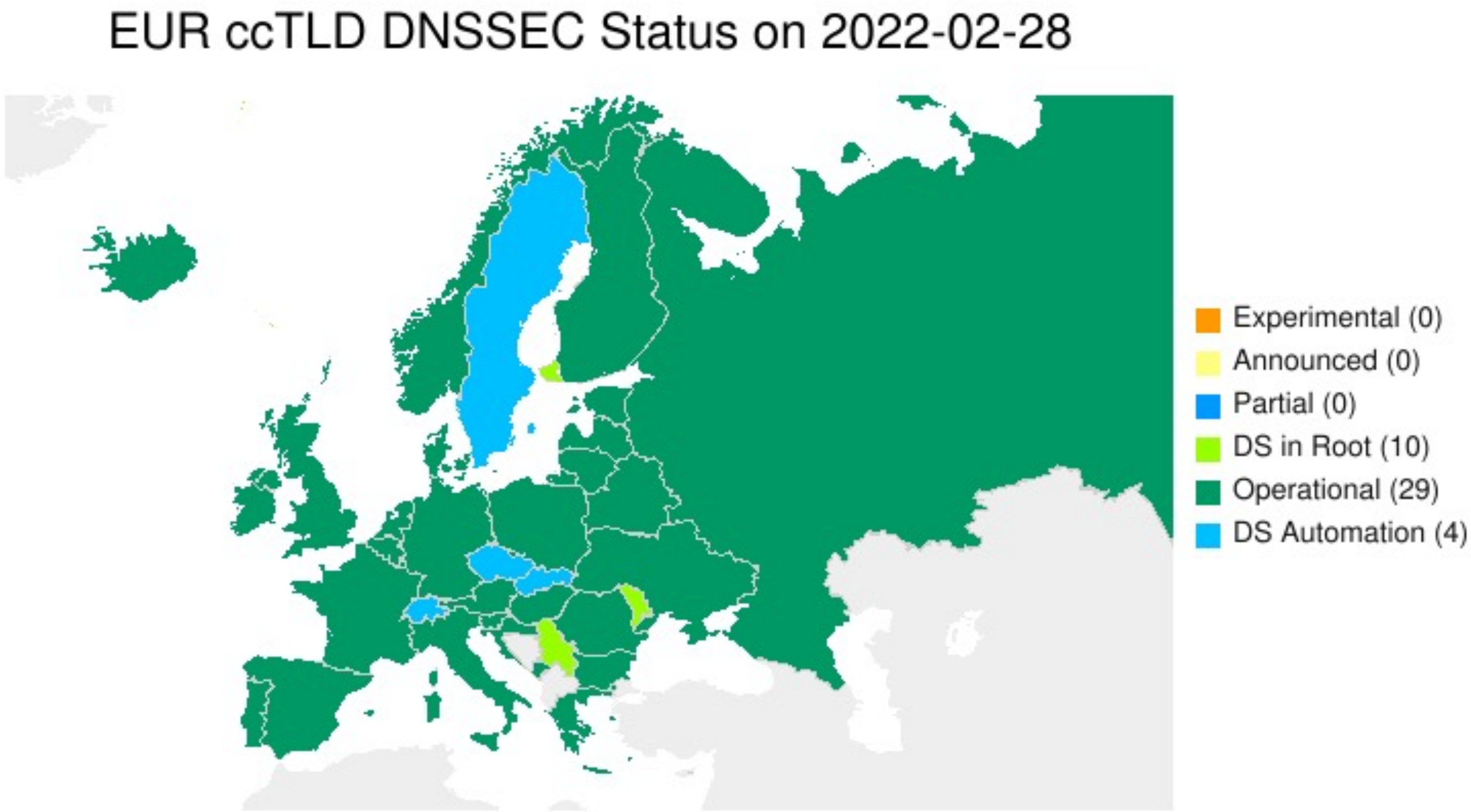
	Direction	
Upper Side	Push (Calling) Call Rr or Rt API	Pull (Polling) Publish CDS/ CDNSKEY
Registry		
Registrar		4. RFC 8078



Registrar polls for CDS/CDNSKEY records.

GoDaddy now testing

ccTLDs now implementing CDS/CDNSKEY Scanning



Actions and Issues

- GoDaddy now testing scanning of customer zones
- SSAC exploring recommendation of DS automation support

- Issue: Scanning is time-consuming. Doesn’t scale well

DS Management Score Card

24 Feb 2022	CDS/CDNSKEY Scanning	DS Bootstrapping
Designed	✓	✓
Specifications	RFC 8078	draft-thomassen-dnsop-dnssec-bootstrapping
In Progress	.CL, GoDaddy	.CL, GoDaddy, CoCCA and others
Done	Several ccTLDs	

DNSSEC: Multi-DNS Provider Coordination & Glitch-Free Provider Change

“Glitch-Free” = No loss of resolution AND no loss of validation

Multi-Signer Software Project

The Swedish Internet Foundation

deSEC

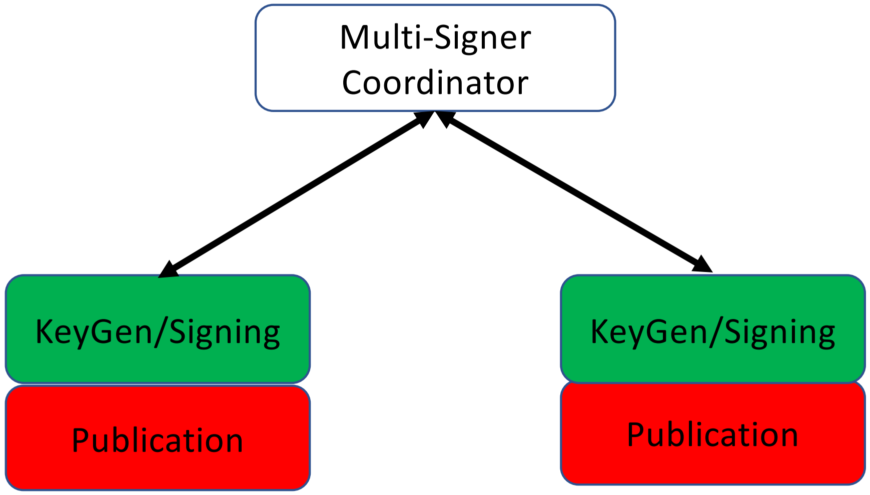
Salesforce

George Mason University

Neustar Security Services

Shinkuro, Inc.

Cross-Signing: Communicating ZSKs & KSKs



Registrant coordinates using a Multi-signer Coordinator

Multi-Signer Operational* Demonstrations

* Operational = Repeatable

- Adding a DNS operator
- Key rollover in one of the operations
- (Concurrent key rollover – will it work?)
- Removal of an operator

▪ Observation of glitch-free operation for each of the above

- Repeat of each, violating the timing constraints

▪ Observation of glitches when timing constraints are violated

Multi-Signer Big Picture

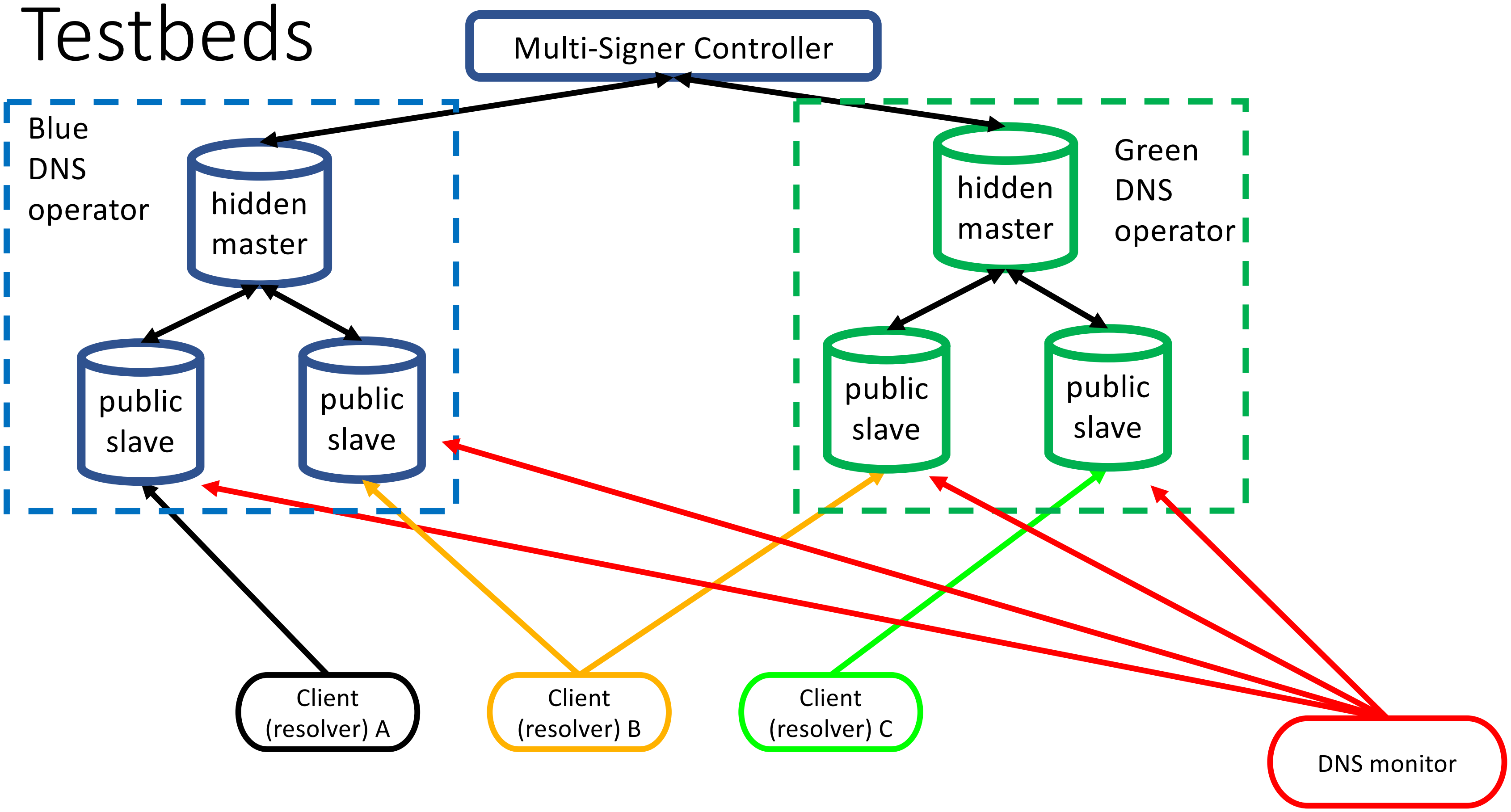
✓ Done
❑ In progress
○ Future
• Unspecified/Mixed

✓ Protocol (RFC 8901)

- Software
  - Multi-Signer Controller
    - ❑ Design
    - ❑ Implementation
  - DNS Server Interfaces
    - ❑ BIND, PowerDNS, …
  - Services/Operations
    - ❑ deSEC, NS1, Neustar …

- Analysis
  - ✓ Text
  - ○ Proof
- Observation
  - Longitudinal
  - Real-time
    - ○ System Design
    - ○ Deployment
    - ○ Experiments
      - ○ Positive
      - ○ Negative

Testbeds


Multi-Signer Controller
Blue
DNS
operator
hidden
master
public
slave
public
slave
hidden
master
Green
DNS
operator
public
slave
public
slave
Client
(resolver) A
Client
(resolver) B
Client
(resolver) C
DNS monitor

Multi-Signer Controller Components

- Interfaces to authoritative DNS servers
- Scenario sequencer
- User interface
  - Identities of authoritative servers
  - Credentials for access to the servers
  - Control to start, stop, undo transitions
- Module to check success of transitions
- Reporting
- Statistics

Multi-Signer Score Card

3 Mar 2022	Designed	In Progress	Done
Specifications	✓	**draft-wisser-dnssec-automation**	**RFC 8901** **RFC 8078**
Multi-Signer Controller	✓	✓	
Name Server Software Capabilities	✓	**Knot**	**PowerDNS, BIND**
DNS Service Provider Capabilities	✓	**NS1, Neustar, Cloudflare**	**deSEC**
Documents			
Observation & Analysis	✓	✓	
Demonstrations			

Name Server Software Capabilities

14 Oct 2021	BIND			Knot			PowerDNS			(Others TBD)					
	C	D	R	C	D	R	C	D	R	C	D	R	C	D	R
Add DNSKEY records	✓	✓	▨	✓	❑	▨	✓	✓	✓						
Remove DNSKEY records	✓	✓	▨	✓	❑	▨	✓	✓	✓						
Add CDS/CDNSKEY records	✓	✓	▨	?	❑	▨	✓	✓	✓						
Remove CDS/CDNSKEY records	✓	✓	▨	✓	o	▨	✓	✓	✓						
Add CSYNC record	✓	✓	▨	✓	❑	▨	✓	✓	✓						
Remove CSYNC record	✓	✓	▨	✓	❑	▨	✓	✓	✓						

C = Command Line Interface – not usable

D = Dynamic DNS

R = Rest API

✓	**Complete**
❑	**In progress**
o	**Planned but not started**
▨	**Not Planned**

DNS Service Provider Capabilities

4 Mar 2022	deSEC			NS1			Neustar			Cloudflare			(Others)					
	C	D	R	C	D	R	C	D	R	C	D	R	C	D	R	C	D	R
Add DNSKEY records	▨	✓	✓	▨	❑	❑	▨	❑	❑	▨	▨	✓						
Remove DNSKEY records	▨	✓	✓	▨	❑	❑	▨	❑	❑	▨	▨	✓						
Add CDS/CDNSKEY records	▨	✓	✓	▨	❑	❑	▨	❑	❑	▨	▨	❑						
Remove CDS/CDNSKEY records	▨	✓	✓	▨	❑	❑	▨	❑	❑	▨	▨	❑						
Add CSYNC record	▨	✓	✓	▨	❑	❑	▨	❑	❑	▨	▨	o						
Remove CSYNC record	▨	✓	✓	▨	❑	❑	▨	❑	❑	▨	▨	o						

C = Command Line Interface – not usable

D = Dynamic DNS

R = Rest API

✓ **Complete**

❑ **In progress**

o **Planned but not started**

▨ **Not Planned**

References

DNSSEC Provisioning Automation “Episodes” Standing Panel at ICANN DNSSEC Workshops

Episode	Date	Meeting	DNSSEC Provisioning Automation Sessions
1	11 Mar 2020	ICANN 67 “Cancún”	https://tinyurl.com/5dwxfz2v
2	22 Jun 2020	ICANN 68 “Kuala Lumpur”	https://tinyurl.com/m8eraezu
3	21 Oct 2020	ICANN 69 “Hamburg”	https://tinyurl.com/f8ma6347
4	24 Mar 2021	ICANN 70 “Cancún”	https://tinyurl.com/bj69sn87
5	14 Jun 2021	ICANN 71 “The Hague”	https://tinyurl.com/t2fcefr6
6	27 Oct 2021	ICANN 72 “Seattle”	https://tinyurl.com/32aeptd3
7	9 Mar 2022	ICANN 73 “San Juan”	https://tinyurl.com/yzyb29s9

Internet Society DNSSEC Maps

https://www.internetsociety.org/deploy360/dnssec/maps/

Episode 1: 20 March 2020 “Cancún”

#	Title	Speaker	TinyURL
	Steve Crocker will outline the problems and the space of possible solutions	Steve Crocker, Shinkuro, Inc	https://tinyurl.com/4w2eck8j
	DS Automation		
	Registry:	James Galvin, Afilias; Erwin Lansing, DK; and Gavin Brown, CentralNic for SK	
	Multisigner Project		
	Registrar	Brian Dickson, GoDaddy; Jothan Frakes, PLISK; and Ólafur Guðmundsson, Cloudflare	
	DNS Provider	Ólafur Guðmundsson, Cloudflare	

Episode 2: 22 June 2020 “Kuala Lumpur”

#	Title	Speaker	TinyURL
	DS Updates and Multi-Signer Coordination	Steve Crocker, Shinkuro, Inc	https://tinyurl.com/vzu58xzv
DS Automation			
	Multi-Signer DNSSEC	Shumon Huque, Salesforce, Inc	https://tinyurl.com/6sche46m
Multisigner Project			
	Support for Multi-Signer DNSSEC	Paul Ebersman, Neustar	https://tinyurl.com/4kmcxmfw
	GoDaddy DNSSEC Signing and DS Updates	Brian Dickson, GoDaddy	https://tinyurl.com/bev24h6u
	Managing DNSSEC via API	Jothan Frakes, PLISK	https://tinyurl.com/w6ce9mu9
	Automated DNSSEC in CZ	Jaromír Talíř, CZ.NIC	https://tinyurl.com/dphwhby4
	Support for and adoption of CDS in .CH and .LI	Oli Schacher, SWITCH	https://tinyurl.com/22c6t6sn

Episode 3: 21 October 2020 “Hamburg”

#	Title	Speaker	TinyURL
I.	Overview: Framing the Issues	Shumon Huque and Steve Crocker	https://tinyurl.com/44dttx7p
II.	• SE DNSSEC History Present Future	Ulrich Wisser, SIF*	https://tinyurl.com/35m44a67
	• Deploying DNSSEC in a Large Enterprise	Han Zhang & Allison Mankin, Salesforce	https://tinyurl.com/jn8d9cv8
		DS Automation	
III.	• DS Automation	Shumon Huque, Salesforce	https://tinyurl.com/nnma8aau
	• DS Automation: Non-technical Considerations	James Galvin Ph.D., Afilias, Inc	https://tinyurl.com/p692jjzu
	• GoDaddy DNSSEC DS – Current and Proposed DS Update Methods	Brian Dickson, GoDaddy	https://tinyurl.com/8d695va9
	• Gathering the Childrens DS’	Mark Elkins, Posix	https://tinyurl.com/59697hm5
	• Evolving the DNSSEC Deployment Maps	Dan York, Internet Society	https://tinyurl.com/ytz9xw8k
		Multisigner Project	
IV.	• DNSSEC Census: Are DNSKEY Transitions Working?	Eric Osterweil, George Mason Univ	https://tinyurl.com/7tzwr6hr
	• Automating Multiple Signers	Shumon Huque, Salesforce	https://tinyurl.com/va53mwy8
V.	• Action Items:	Steve Crocker	https://tinyurl.com/2zykj7zs

*SIF = The Swedish Internet Foundation

Episode 4: 24 March 2021 “Cancún”

#	Title	Speaker	TinyURL
4.1	Panel Overview	Steve Crocker, Shinkuro, Inc	https://tinyurl.com/msaakbud
DS Automation			
4.2	DS Automation at GoDaddy	Brian Dickson, GoDaddy	https://tinyurl.com/hwx6hy52
Multisigner Project			
4.3	Intro to Multisigner Project Foundations	Shumon Huque, Salesforce	https://tinyurl.com/4cwcndrr
4.4	Multisigner Protocols	Ulrich Wisser, SIF*	https://tinyurl.com/v4y727sj
4.5	Multisigner Testbed	Ulrich Wisser, SIF*	https://tinyurl.com/cm3uuhk3
4.6	Multisigner Multisigner support at deSEC	Peter Thomassen, Secure Systems Engineering	https://tinyurl.com/eyymfh2z
4.7	DNSKEY Transition Observatory	Ravichander, Osterweil, GMU	https://tinyurl.com/vdwpj4wp
4.8	Anatomy of DNSSEC Transitions	Osterweil, Tehrani, Schmidt, Waehlisch	https://tinyurl.com/ssfxwr3x

*SIF = The Swedish Internet Foundation

Episode 5: 14 June 2021 “The Hague”

#	Title	Speaker	TinyURL
3.1	DNSSEC Provisioning Automation Overview	Steve Crocker, Shinkuro, Inc	https://tinyurl.com/5a66kvpx
	DS Automation		
3.2	CDS scanning at RIPE NCC	Ondřej Caletka, RIPE NCC	https://tinyurl.com/t673a7px
3.3	The State of DNSSEC Automated Provisioning	Wilco van Beijnum, University of Twente	https://tinyurl.com/ntv5um3k
	Multisigner Project		
3.4	Multi-Signer Project Overview and Status	Ulrich Wisser, SIF*	https://tinyurl.com/4uyvps4u
3.5	BIND DNSSEC Provisioning Interfaces	Matthijs Mekking, Internet Systems Consortium	https://tinyurl.com/56p3pye7
3.6	PowerDNS DNSSEC Provisioning Interfaces	Peter van Dijk, PowerDNS	https://tinyurl.com/vracytyp

*SIF = The Swedish Internet Foundation

Episode 6: 27 October 2021 "Seattle"

#	Title	Speaker	TinyURL
6.1	DNSSEC Provisioning Automation Overview	Steve Crocker, Shinkuro, Inc	
6.2	Recent DNSSEC Automation Developments in .CZ	Jaromír Talíř, CZ.NIC	
6.3	CDS & CDNSKEY Verification in Zonemaster	Mats Dufberg, SIF	
6.4	Authentication Bootstrapping of DNSSEC Delegations	Peter Thomassen, deSEC	
6.5	DNS Resolver Observatory	Pouyan Tehrani, Freie Universität Berlin	
6.6	Introduction to CSYNC	Ulrich Wisser, SIF	

*SIF = The Swedish Internet Foundation

Episode 7: 9 March 2022 “San Juan”

#	Title	Speaker	TinyURL
3.1	Overview: DNSSEC Provisioning Automation	Steve Crocker, Shinkuro, Inc.	
3.2	GoDaddy CDS Support Update	Brian Dickson, GoDaddy	
3.3	CSYNC implementation	Ulrich Wisser, SIF	
3.4	Authenticated Bootstrapping of DNSSEC Delegations	Nils Wisiol, deSEC, Technische Universität Berlin	
3.5	SSAC DS Automation Work Party	Steve Crocker, Shinkuro, Inc.	
3.6	Making MUSIC with DNSSEC	Johan Stenstam, Roger Murray, SIF	
3.7	RFC Adjustments for Multi-Signer	Shumon Huque, Salesforce	
3.8	DNS(SEC) Views	P.F. Tehrani, et al, Weizenbaum Institute / Fraunhofer FOKUS	

*SIF = The Swedish Internet Foundation

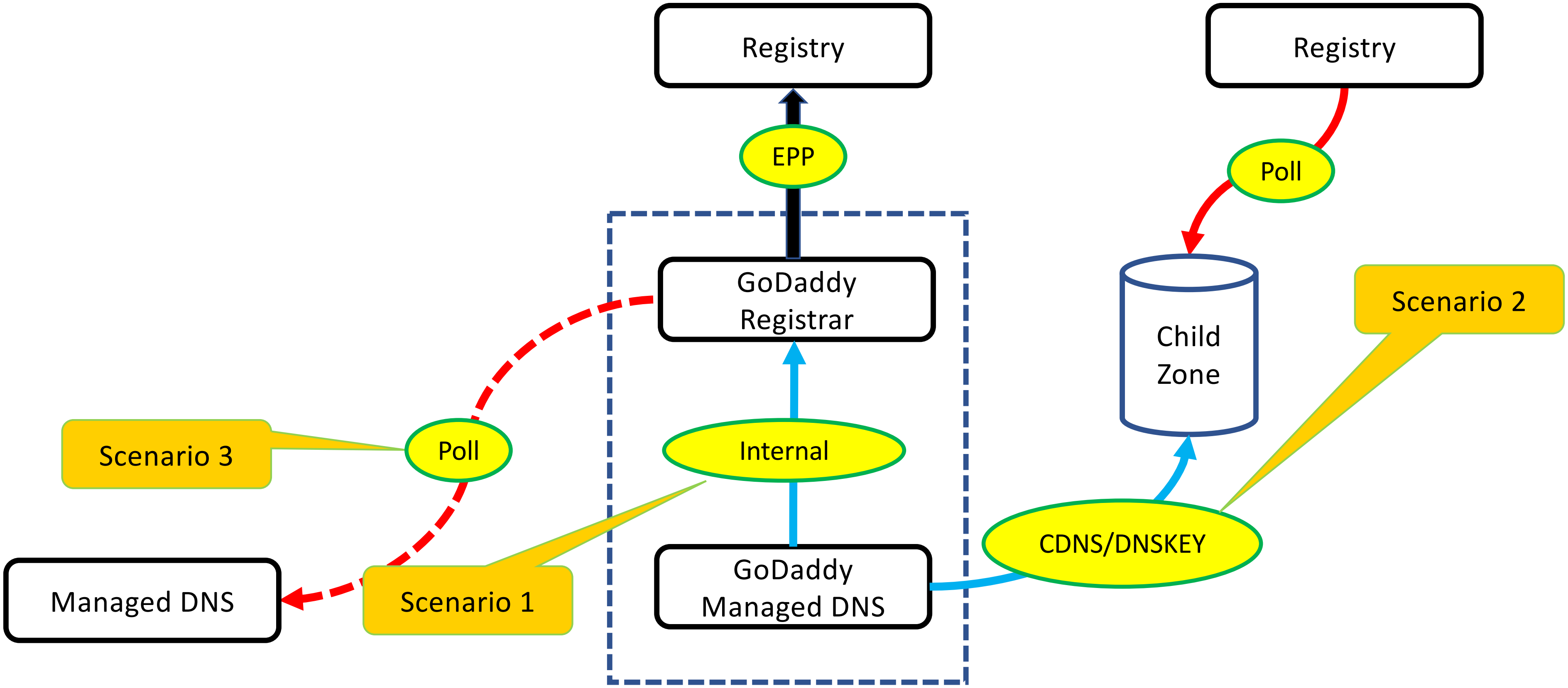
Thanks!

GoDaddy CDS Support Update

Current and Proposed DS Update Methods for KSK Rollovers

Brian Dickson, GoDaddy

GoDaddy DNS – Three Scenarios



6 March 2022

GoDaddy DNSSEC DS – KSK Rollover

- Managed DNS (Scenarios 1 & 2):
  - KSK rolls automated
  - EPP Update(s) Sent to Registry
  - Regardless of parent (TLD), CDS and CDNSKEY are always published
  - Managed DNS performs initial DS when DNSSEC is enabled
- Third Party DNS (Scenario 3)(Current Enhancement):
  - Require initial DS registration, to authorize and initiate polling
  - Poll CDS and/or CDNSKEY periodically
  - EPP Update(s) Sent to Registry
  - **Testing has started! Contact Brian Dickson, bdickson@godaddy, to participate.**

DS Update Scenarios

Scenario	Registrar	DNS Provider	Update DS	Status
1	GD	GD	EPP, GD also publishes CDS/CDNSKEY record.	Fully Operational
2	Not GD	GD	GD publishes CDS/CDNSKEY record. Registry polls zone.	Fully Operational
3	GD	Not GD	DNS provider publishes CDS/CDNSKEY record. GD polls zone and submits DS and/or DNSKEY records	Implemented and now being tested

Looking Ahead

- Automation of bootstrap of DS record
- Multi-signer support

INTERNET
STIFTELSEN

CSYNC

Ulrich Wisser, The Swedish Internet Foundation

CSYNC

- Domain Is DNSSEC signed

- CSYNC updates NS records and possibly glue information

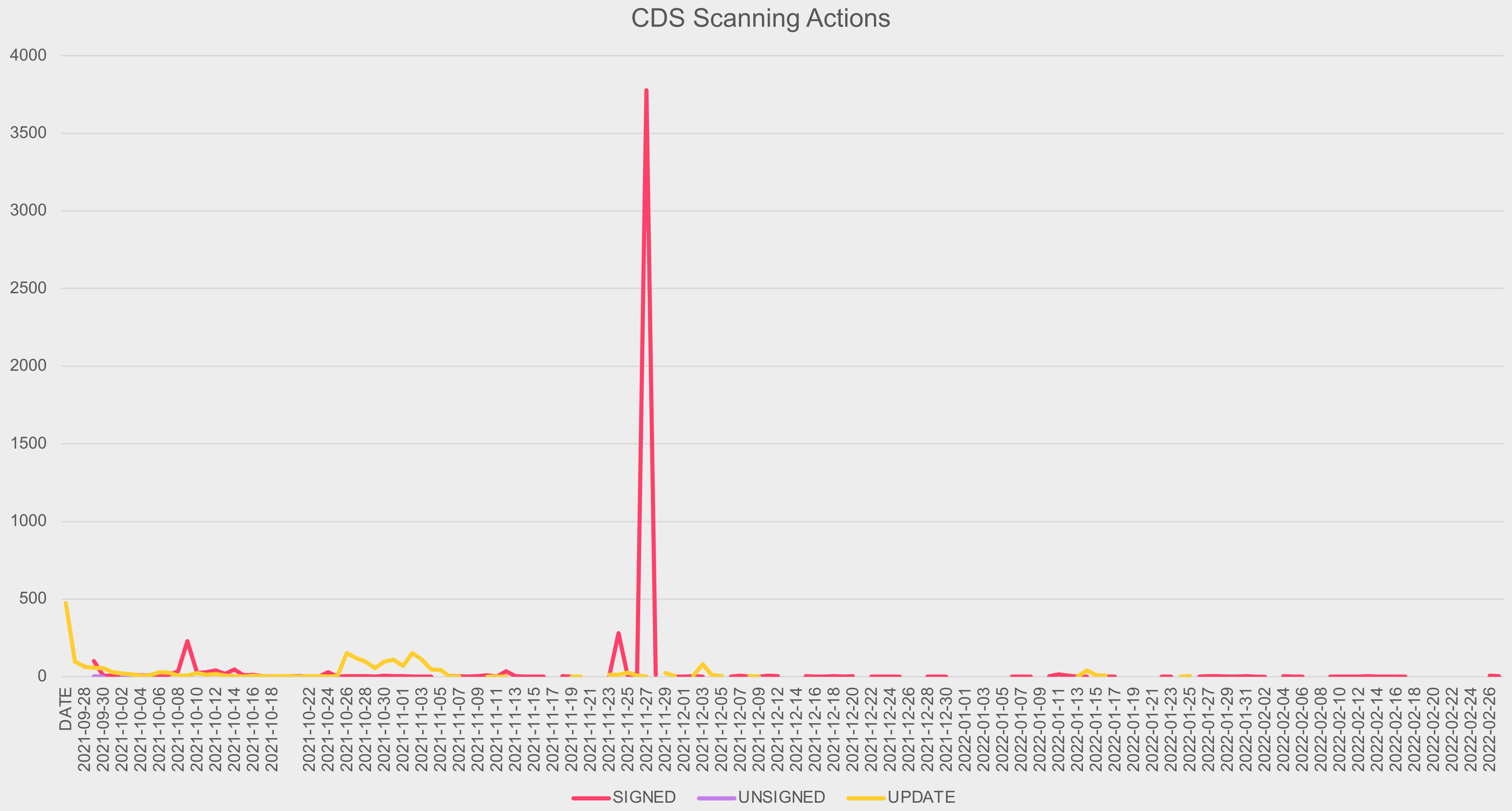
.SE

2021-10 CDS Scanning

2022-02 CSYNC Scanning

CDS / CSYNC

- Registry – Registrar Model has no place for DNS Operators
- Combination of CDS and CSYNC Scanning gives DNS Operators the influence they need
- Allows infrastructure updates
- Allows DNSSEC automation


CDS Scanning Actions
4000
3500
3000
2500
2000
1500
1000
500
0
DATE
2021-09-28
2021-09-30
2021-10-02
2021-10-04
2021-10-06
2021-10-08
2021-10-10
2021-10-12
2021-10-14
2021-10-16
2021-10-18
2021-10-22
2021-10-24
2021-10-26
2021-10-28
2021-10-30
2021-11-01
2021-11-03
2021-11-05
2021-11-07
2021-11-09
2021-11-11
2021-11-13
2021-11-15
2021-11-17
2021-11-19
2021-11-21
2021-11-23
2021-11-25
2021-11-27
2021-11-29
2021-12-01
2021-12-03
2021-12-05
2021-12-07
2021-12-09
2021-12-12
2021-12-14
2021-12-16
2021-12-18
2021-12-20
2021-12-22
2021-12-24
2021-12-26
2021-12-28
2021-12-30
2022-01-01
2022-01-03
2022-01-05
2022-01-07
2022-01-09
2022-01-11
2022-01-13
2022-01-15
2022-01-17
2022-01-19
2022-01-21
2022-01-23
2022-01-25
2022-01-27
2022-01-29
2022-01-31
2022-02-02
2022-02-04
2022-02-06
2022-02-08
2022-02-10
2022-02-12
2022-02-14
2022-02-16
2022-02-18
2022-02-20
2022-02-22
2022-02-24
2022-02-26
SIGNED
UNSIGNED
UPDATE

Automatic DNSSEC Bootstrapping

using Authenticated Signals from the Zone's Operator

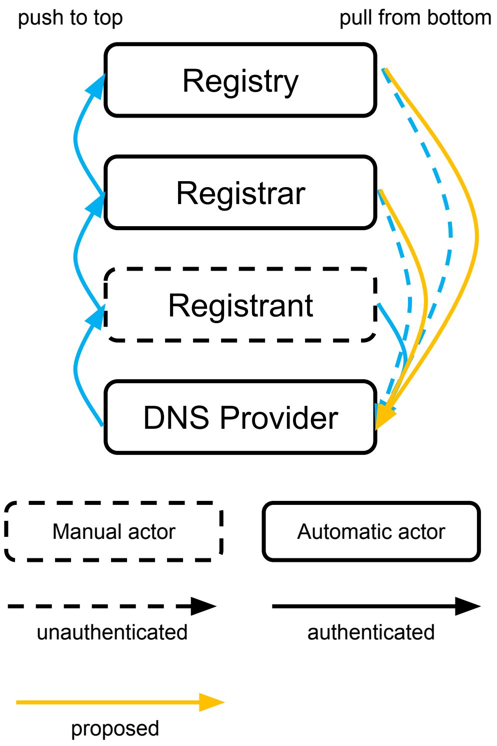
draft-thomassen-dnsop-dnssec-bootstrapping

ICANN 73 – DNSSEC and Security Workshop
9 March 2022

Peter Thomassen (deSEC, Secure Systems Engineering)
Nils Wisiol (deSEC, Technische Universität Berlin)

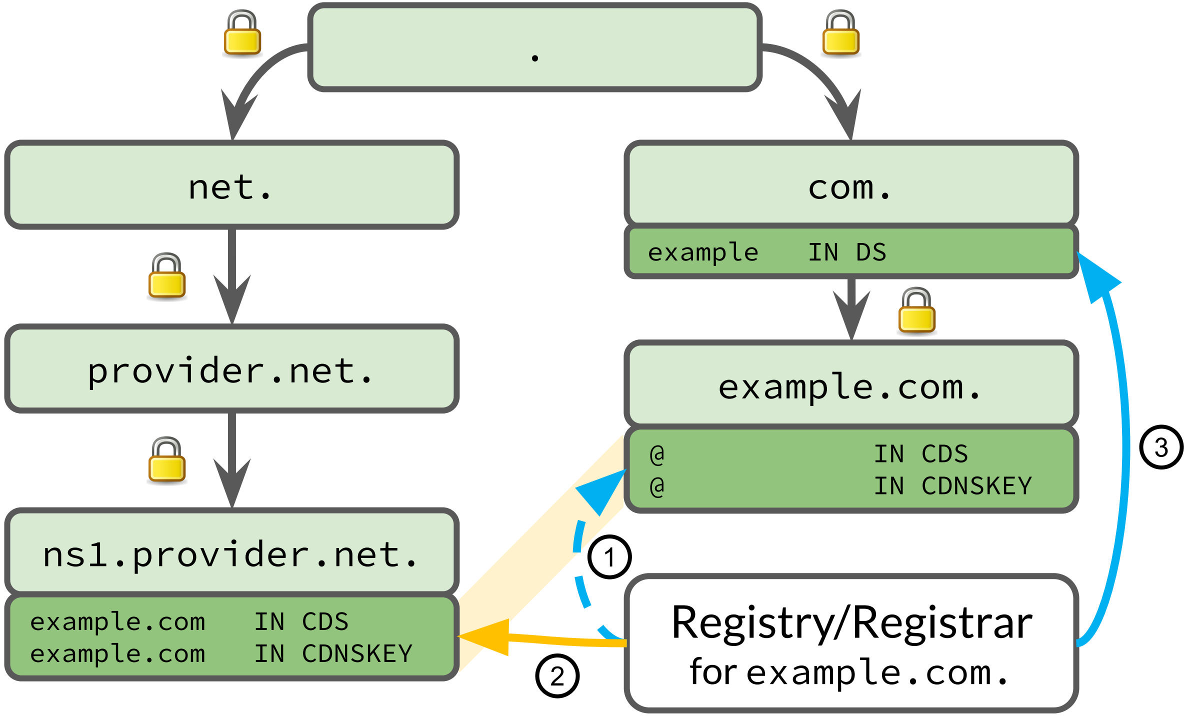
DS Bootstrapping and Why It Needs Improvement

- Various methods have emerged
- Each suffers from one or more downsides
  - **Authenticated workflow involves too many steps**
- RFC 8078: **direct pull from DNS operator**
  - **in-band** (via CDS / CDNSKEY)
  - not secure for bootstrapping
- Proposal: co-publish CDS/CDNSKEY records with authentication
  - In child zones of the name server names
  - Name server zones must be secure



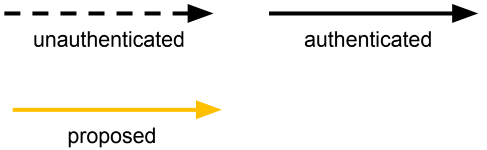
* ICANN 54 (2015), draft-ietf-regext-dnsoperator-to-rrr-protocol (2018)

CDS Authentication: Co-Publish under Trusted Hostname



💡 Use an **established chain of trust** (left) to take a detour

- authenticated, immediate
- no active on-wire attacker



Technical Considerations

- No collision with primary use of CDS/CDNSKEY (those are apex-only)

- Add extra label: `example.com._dsboot.ns1.provider.net.`
  - to enable delegation of signaling data to separate zone
  - Update: no hashing of any part of the zone name (enables online generation/signing)
  - allows splitting off DNS operations (e.g. online-signing with different key; delegate by parent)
  - reduces churn on nameserver zone
  - allows discovery of bootstrappable domains using XFR (if allowed)
  - Do you like “`_dsboot`”?

Is this useful?

Deployment Requirements

- Name server names are in secure zones
- Zone not yet secure

25%

Fulfill requirements in Tranco Top 1M

Current Status

- Implementation
  - Prototype implementation: github.com/desec-io/dsbootstrap
  - CoCCA: implementation underway for 59 ccTLDs
  - GoDaddy: implementation planned after CDS scanning
  - Cloudflare: experimental implementation planned
  - .cl: implementation finished, waiting for internal approval
  - .ch, .cz: interested
- Adoption of draft by IETF DNS WG in sight
- Post at APNIC Blog to get the word out

Backup

Open Questions

- Should we support sharding, by splitting Signaling Names into several labels?
    - How exactly would that work? Should that be configurable? (How to store configuration?)
- Should the hash(ancestor) label have a PTR record pointing to ancestor?
    - This would allow full enumeration of bootstrappable domains
- For an operator supporting the protocol: is it REQUIRED for all domains?
    - Probably no, as it won’t work with secondary providers?

Closed Questions (I)

- If a DNS operator deploys DS bootstrapping, parents may like bulk processing. How is that best achieved?
  - allow NSEC walking of signaling zone (thanks to Brian Dickson)
  - allow public AXFR of signaling zone (thanks to John R. Levine)

- Should an extra layer be inserted in the Signaling Name to allow parent-specific bulk processing? (thanks to John R. Levine)
  - Yes
  - compatible with both NSEC walking
  - also compatible with AXFR (but benefit gained only when using subzones for large parents)

- Do we need hash collision mitigation (salt) and/or hash algo upgrade path?
  - No: due to child apex check, collisions don’t affect key integrity
  - In case of collision, bootstrapping fails (for this parent) → fallback to conventional DS init

- Do we want hashing at all?
  - No

Closed Questions (II)

- Should the proposal be rephrased as a new mode of operation for RFC 8078?
  - cf. RFC 8078 Section 3.1
  - done

- Drop requirement that all NS responses must agree?
  - No. Otherwise, multihoming with different signers will break the zone.
  - Deployment effort is manageable: 95% of delegations with at least one securely delegated NS target in fact have *all* NS targets securely delegated. Also, dropping this requirement would be inconsistent with requiring records at the child apex to match. It's also unclear what should happen in case of contradictory signaling records, if they are not required to agree.

- Registries/registrars can select which TLDs to trust in the chain. Desirable?
  - No (at least in the spec). One could say that you can't trust a DNS operator anyway if its NS hostnames are not trusted. (That doesn't prevent parents from deciding locally to ignore or reject certain signaling names.)

Discussion Point: **Do we want the hashed label?**

Do the benefits justify the added complexity?

Pros: ... yes, please, hash please!

- Helps **stay within limits**
  - length / no. of labels → less edge cases
- **Prevents CDS ambiguity** at zone cut
  - What does foo.bar.net._boot.[...] mean?
  - It's possible that bar.net is not delegated
- **Improves privacy** during discovery
  - must know ancestor to begin NSEC walk
- **Flat structure**
  - simplifies scanning logic
  - facilitates adding prefixes → **"properties"** ... like: _cds.**example.h(co.uk)**._signal.[...]

Cons: ... no, smash the hash!

- **Complicates implementation**
  - all tooling needs to be able to hash
- Makes **debugging more difficult**
  - standard tools should suffice (dig etc.)
- Makes **synthesis more difficult**
  - How to dynamically associate an incoming query with a target domain? → **mapping needed (ancestors only!)**
  - h(co.uk)._boot DNAME co.uk._boot (cacheable per parent!)

Securing the `example.com` delegation (no existing DS)

__Assumption__: The **NS targets** (e.g. *ns1.provider.net*) **live in securely delegated zones** (e.g. *provider.net*).

(I) __On the DNS provider side__:

Publish `example.com`'s CDS/CDNSKEY records at a **"signaling name"** under the **nameserver zone**:
example.com.*ns1.provider.net*

(II) __On the registrar / ccTLD registry side__:

When receiving a new NS record set,

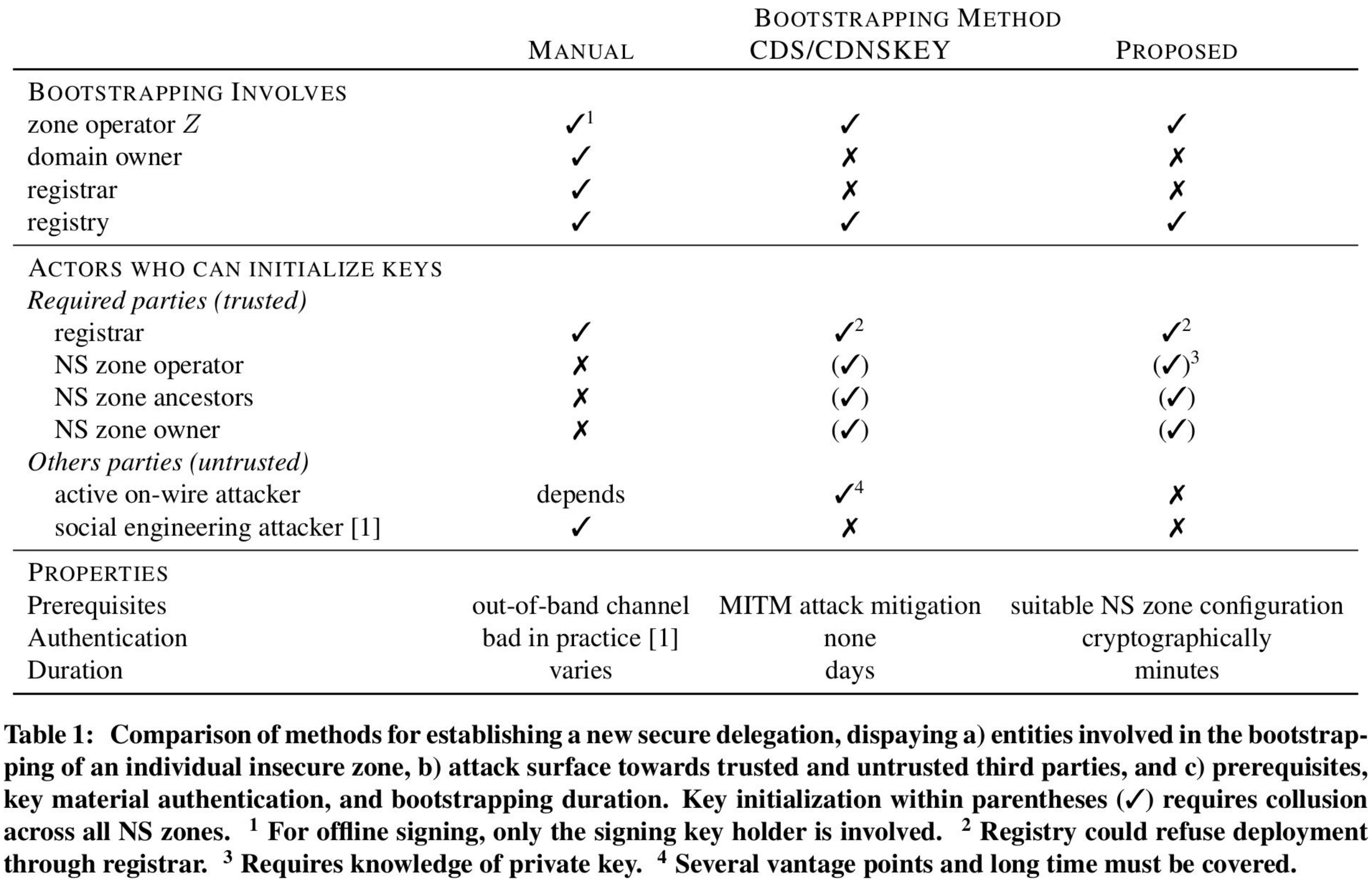
1. **query** CDS/CDNSKEY records **from DNS provider** (using all NS names):
   - example.com.*ns1.provider.net*, …;
2. **validate**
   - **DNSSEC signatures** of responses,
   - **sanity check** (consistency with target zone);
3. **publish** example.com's **DS records** in the parent zone → **done!** ✅

Security Model

- We use an established chain of trust to take a detour
    - authenticated, immediate
    - no active on-wire attacker

- Actors in the chain of trust can undermine the protocol
    - can also undermine CDS / CDNSKEY from insecure
    - but: known point in time / window of opportunity much smaller

- Further mitigations exist, e.g:
    - monitor delegation
    - diversify NS TLDs
    - multiple vantage points



		BOOTSTRAPPING METHOD	
	MANUAL	CDS/CDNSKEY	PROPOSED
BOOTSTRAPPING INVOLVES			
zone operator Z	✓[1]	✓	✓
domain owner	✓	✗	✗
registrar	✓	✗	✗
registry	✓	✓	✓
ACTORS WHO CAN INITIALIZE KEYS			
Required parties (trusted)			
registrar	✓	✓[2]	✓[2]
NS zone operator	✗	(✓)	(✓)[3]
NS zone ancestors	✗	(✓)	(✓)
NS zone owner	✗	(✓)	(✓)
Others parties (untrusted)			
active on-wire attacker	depends	✓[4]	✗
social engineering attacker [1]	✓	✗	✗
PROPERTIES			
Prerequisites	out-of-band channel	MITM attack mitigation	suitable NS zone configuration
Authentication	bad in practice [1]	none	cryptographically
Duration	varies	days	minutes

Table 1: Comparison of methods for establishing a new secure delegation, dispaying a) entities involved in the bootstrapping of an individual insecure zone, b) attack surface towards trusted and untrusted third parties, and c) prerequisites, key material authentication, and bootstrapping duration. Key initialization within parentheses (✓) requires collusion across all NS zones. [1] For offline signing, only the signing key holder is involved. [2] Registry could refuse deployment through registrar. [3] Requires knowledge of private key. [4] Several vantage points and long time must be covered.

SSAC DS Automation Work Party

Steve Crocker

steve@shinkuro.com

Who, Why and What

Who: SSAC has initiated a “work party” to review the state of DS automation and formulate recommendations. Several guests are included.

Why: One of the reasons DS automation hasn’t been part of DNSSEC deployment is registrars are officially recognized in the ICANN contractual and political structure but DNS providers are not.

What: Raise visibility regarding the issues. Provide support at the policy level to accompany the technical developments

Goals

- Remove Provisioning Roadblock
- Facilitate DNSSEC Deployment and Operation

Specifically:

- Determine if additional technical developments are needed
- Raise Awareness of the problem and the solution(s)
- Recommendations for Registries, Registrars, DNS Signers, and DNS software vendors

Tasks/Activities

- Survey technical developments
- Survey deployment
- Engage in discussions with RrSG, RySG, ccNSO, DNS Providers
- Draft recommendations
- Etc.

Thanks!

Making Music With DNSSEC

Getting the pesky details of a complex DNSSEC corner right through automation

Johan Stenstam Roger Murray

Internetstiftelsen

February 28, 2022

To many people DNSSEC sounds a bit like...

To many people DNSSEC sounds a bit like. . .

...nails scratching against a black board

To many people DNSSEC sounds a bit like. . .

...nails scratching against a black board

- Unfortunately, it isn't hard to understand why.
- There are lots of details, lots of things to get wrong.
- In particular there are things that may go wrong **over time** even if one successfully turns on DNSSEC for a zone.

DNSSEC Progress and Failures over the Years

Compared to the initial versions, we've come a long way

- Key generation and zone signing is mostly automated.
- Software quality and robustness has improved dramatically.
- There are multiple quality providers of both primary and secondary services for DNSSEC signed zones.
- Knowledge of what DNSSEC does and why is now widespread .

DNSSEC Progress and Failures over the Years

Compared to the initial versions, we've come a long way

- Key generation and zone signing is mostly automated.
- Software quality and robustness has improved dramatically.
- There are multiple quality providers of both primary and secondary services for DNSSEC signed zones.
- Knowledge of what DNSSEC does and why is now widespread .

But there are also things that still aren't quite sorted

- Failure scenarios are still painful, both to experience and to debug.
- Some zone owners avoid rolling their keys, in particular their KSKs, as that involves communicating with the parent zone.
- Changing DNS operator for a signed zone is so utterly difficult that the normal case is to "go unsigned" during transition.

DNSSEC Progress and Failures over the Years

Let's focus here

Compared to the initial versions, we've come a long way

- Key generation and zone signing is mostly automated.
- Software quality and robustness has improved dramatically.
- There are multiple quality providers of both primary and secondary services for DNSSEC signed zones.
- Knowledge of what DNSSEC does and why is now widespread .

But there are also things that still aren't quite sorted

- Failure scenarios are still painful, both to experience and to debug.
- Some zone owners avoid rolling their keys, in particular their KSKs, as that involves communicating with the parent zone.
- Changing DNS operator for a signed zone is so utterly difficult that the normal case is to "go unsigned" during transition.

Focusing on the Issue of Communicating With the Parent

In the DNSSEC case the parent communication is primarily updating the **DS** (Delegation Signer) record in the parent zone.

- To be fair, it is not just an issue with communication it is just as much an issue about the perceived risks of making a change to critical DNSSEC stuff that may break.

However, this is not the only case when it is necessary to communicate with the parent.

- There is also the old classic of keeping the **NS** records and **A** and **AAAA** glue records in sync.

We all know the importance of this, but for some reason it seems that there is always a certain percentage of delegations that are not in sync and therefore, while they may still work, are not as robust as they think they are.

Communicating With the Parent, cont'd

There are several reasons for this communication failure but among them a clear contender for primary cause is the lack of communication path between the DNS operator and the registry (i.e. the parent).

- The registrar has that communication channel (usually via EPP) and this has led to a prevalence of the registrar also being the DNS operator, to the detriment of third party DNS operators.

What is really needed is a mechanism for the DNS operator to **signal to the parent** that certain changes to the data in the parent zone are needed. The good news is that there now exists such a mechanism

- It is implemented via special signalling records in the child zone, called **CDS** and **CSYNC**.

Enter the DNS Record Types **CDS** and **CSYNC**

To update the **DS** records in the parent zone it should be sufficient that:

- The child zone publishes a **CDS** RRset ("Child DS") containing the data it wants the parent to publish as a **DS** RRset.

Likewise, to update the **NS** RRset and/or the **A** and **AAAA** glue records in the parent it should be sufficient that:

- The child zone publishes a **CSYNC** record that signals to the parent what data in the child zone to sync into the parent zone (out of relevant **NS**, **A** and **AAAA** RRsets)

In both cases a requirement is that the child zone is DNSSEC signed. This is needed to be able to verify the data that is copied from the child zone (like the **CDS** RRset).

CDS and **CSYNC**, cont'd

An important caveat is the part about "should be sufficient". **CDS** and **CSYNC** publication only works (as in cause updates in the parent) for parent zones that have chosen to "scan" the child zones to look for these records.

- At this time most parent zones do not scan for **CDS** and **CSYNC** but the expectation is that this will change quite rapidly.

The almost magic part of **CDS** and **CSYNC** is that it suddenly makes synchronization of delegations between parent and child automatic and seamless. And this is enabled by DNSSEC.

- Suddenly DNSSEC is not so much a cause for nail scratching as many people still think.
- Rather, that DNSSEC makes the delegation synchronisation problem that DNS has had for decades just. . . go away is great.

It's almost like. . . MUSIC.

CDS and **CSYNC**, cont'd

An important caveat is the part about “should be sufficient”. **CDS** and **CSYNC** publication only works (as in cause updates in the parent) for parent zones that have chosen to “scan” the child zones to look for these records.

- At this time most parent zones do not scan for **CDS** and **CSYNC** but the expectation is that this will change quite rapidly.

The almost magic part of **CDS** and **CSYNC** is that it suddenly makes synchronization of delegations between parent and child automatic and seamless. And this is enabled by DNSSEC.

- Suddenly DNSSEC is not so much a cause for nail scratching as many people still think.
- Rather, that DNSSEC makes the delegation synchronisation problem that DNS has had for decades just. . . go away is great.

It's almost like. . . $\mathrm{M^{U}S_{I}C}$.

Going back to: DNSSEC Progress and Failures

Compared to the initial versions of DNSSEC, we've come a long way

- Key generation and zone signing is mostly automated.
- Software quality and robustness has improved dramatically.
- There are multiple quality providers of both primary and secondary services for DNSSEC signed zones.
- Knowledge of what DNSSEC does and why is now widespread.

But there are also things that still aren't quite sorted

- Failure scenarios are still painful, both to experience and to debug.
- Some zone owners avoid rolling their keys, in particular their KSKs, as that involves the parent zone.
- Changing DNS operator for a signed zone is so utterly difficult that the normal case is to "go unsigned" during transition.

Going back to: DNSSEC Progress and Failures

Now let's focus here

Compared to the initial versions of DNSSEC, we've come a long way

- Key generation and zone signing is mostly automated.
- Software quality and robustness has improved dramatically.
- There are multiple quality providers of both primary and secondary services for DNSSEC signed zones.
- Knowledge of what DNSSEC does and why is now widespread.

But there are also things that still aren't quite sorted

- Failure scenarios are still painful, both to experience and to debug.
- Some zone owners avoid rolling their keys, in particular their KSKs, as that involves the parent zone.
- Changing DNS operator for a signed zone is so utterly difficult that the normal case is to "go unsigned" during transition.

Focusing On the Issue of Changing the Signer

Why is it so difficult?

Well, it's because:

- It involves keys and in particular exchanging keys between two parties (gaining and losing operator) that have different incentives with the operation.
- Keys are often (for good reason) stored in a way that intentionally makes it difficult to do anything with them other than the "normal operation" (i.e. signing stuff).
- The margins in the DNS services sector are so thin that operators only make money when they don't spend any staff hours on doing something special for certain zones.

Enter the Multi Signer Controller, **MUSIC**

MUSIC is a piece of software that implements the "multi-signer" Internet-Draft by Shumon Huque and Ulrich Wisser.

- That document describes the processes to migrate a signed zone from having one set of "signers" to a new set where either a signer has been added or removed.
- A "signer" is a service that receives the unsigned zone and produces a signed version. A signer is able to generate and manage the necessary keys itself.
- The most common case is that of migrating from one DNS operator to another
  - In multi-signer terms that equals going from one signer via a temporary state of having two signers and then back to a single signer (which is the new one, and the original signer has been removed).

Exactly What Is It That M^{U}S$_{I}$C Does?

M^{U}S$_{I}$C does three things:

- M$_{U}$S^{I}C will get the **DNSKEY** RRsets for the zone in sync between multiple signers by adding the **DNSKEY**s for the ZSKs from each signer to the other signers (the **DNSKEY**s for the KSKs are not needed).

Exactly What Is It That M$_{\text{U}}$S$^{\text{I}}$C Does?

M$_{\text{U}}$S$^{\text{I}}$C does three things:

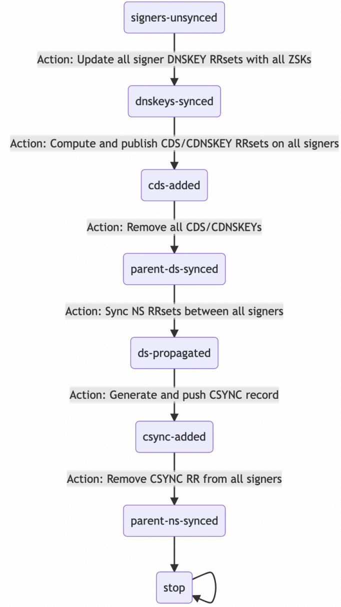
- M$_{\text{U}}$S$^{\text{I}}$C will get the **DNSKEY** RRsets for the zone in sync between multiple signers by adding the **DNSKEY**s for the ZSKs from each signer to the other signers (the **DNSKEY**s for the KSKs are not needed).
- M$^{\text{U}}$S$_{\text{I}}$C will get the **DS** records in the parent zone in sync. It does that by adding instructions to the parent via **CDS** records that are inserted into the zones that the signers maintain.

Exactly What Is It That MUSIC Does?

MUSIC does three things:

- MUSIC will get the **DNSKEY** RRsets for the zone in sync between multiple signers by adding the **DNSKEY**s for the ZSKs from each signer to the other signers (the **DNSKEY**s for the KSKs are not needed).
- MUSIC will get the **DS** records in the parent zone in sync. It does that by adding instructions to the parent via **CDS** records that are inserted into the zones that the signers maintain.
- MUSIC will get the delegation data in the parent zone (**NS** records and also **A** and **AAAA** glue records) in sync. It does that by adding instructions to the parent via **CSYNC** records that are inserted into the zones that the signers maintain.

What Does the MUSIC "Processes" Do?

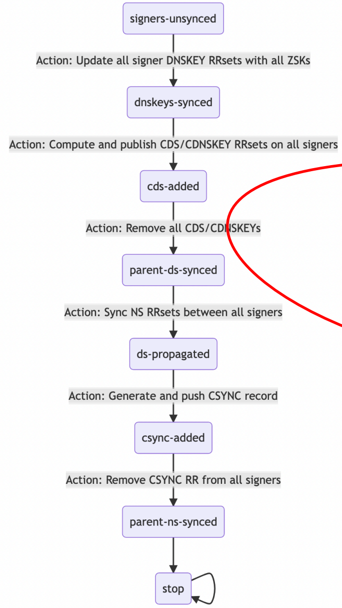

signers-unsynced
Action: Update all signer DNSKEY RRsets with all ZSKs
dnskeys-synced
Action: Compute and publish CDS/CDNSKEY RRsets on all signers
cds-added
Action: Remove all CDS/CDNSKEYs
parent-ds-synced
Action: Sync NS RRsets between all signers
ds-propagated
Action: Generate and push CSYNC record
csync-added
Action: Remove CSYNC RR from all signers
parent-ns-synced
stop

The ADD-SIGNER process

- has a number of "states", all of which are valid
- has a number of "transitions", each with a pre-condition and a post-condition
- starts with getting the signers in sync
- continues with publishing `CDS` records to trigger a `DS` update in the parent
- concludes with syncing remaining delegation data via `CSYNC`

REMOVE-SIGNER is similar, but obviously does things mostly in the opposite order.

What Does the MUSIC “Processes” Do?


signers-unsynced
Action: Update all signer DNSKEY RRsets with all ZSKs
dnskeys-synced
Action: Compute and publish CDS/CDNSKEY RRsets on all signers
cds-added
Action: Remove all CDS/CDNSKEYs
parent-ds-synced
Action: Sync NS RRsets between all signers
ds-propagated
Action: Generate and push CSYNC record
csync-added
Action: Remove CSYNC RR from all signers
parent-ns-synced
stop

This is a crucial point

The ADD-SIGNER process

- has a number of “states”, all of which are valid
- has a number of “transitions”, each with a pre-condition and a post-condition
- starts with getting the signers in sync
- continues with publishing `CDS` records to trigger a `DS` update in the parent
- concludes with syncing remaining delegation data via `CSYNC`

REMOVE-SIGNER is similar, but obviously does things mostly in the opposite order.

How Does **MUSIC** Work?

MUSIC knows three types of entities:

- **signers** are services that are able to sign and publish zones.
- **zones** are normal DNS zones. MUSIC doesn't care how the zone contents are maintained, nor how zone content is is kept in sync with the signers.
- **signergroups** are data structures that connects signers and zones. All zones in a signergroup will be kept in sync across all signers in the group.

Adding a signer to a signergroup automatically cause the zones in the signergroup to go through the "ADD-SIGNER" process to ensure that that the zones get in sync given the new set of signers will cause changes to **DNSKEY**s, and quite likely also to **NS** records and perhaps glue.

- Removing a signer likewise trigger the "REMOVE-SIGNER" process for the zones in the signergroup.

How Does **MUSIC** Work, cont'd

The core requirement to being able to design a “third party” controller (like **MUSIC**) for the synchronization between signers is the availability of an interface that allows the controller to update DNS records directly in the signers.

- The poster child update interface is standard DNS dynamic updates, which work with several open-source nameserver implementations including BIND9 and PowerDNS.
- Commercial DNS services typically provide update access via some sort of API. Here, we use the deSEC API as the proof-of-concept. But it will likely work well with most other APIs, like Google, Route53, etc, although at this time this has not yet been tested.

What **MUSIC** Is

- **MUSIC** is a proof-of-concept implementation. The goal is to prove that it is possible to fully automate the complex processes of adding or removing signers for a signed zone **without “going unsigned”**.
- **MUSIC** uses a server—client design where the client uses a secure RESTful API to communicate with the server.
- The **MUSIC server** manages all zones through the series of state transitions that are needed according to the multi signer algorithms.
- The **MUSIC client** part can be anything. We provide a command-line tool, but it could just as well be a web frontend, or part of a provisioning system managing a large portfolio of zones.
- **MUSIC** is **safe**. Every state that a zone passes through during the process is a valid state where everything is working fine. Every transition has pre- and post-conditions that must be fulfilled for that step to take place.

What Are the Use Cases for **MUSIC**?

The goal was to to address the problem of how to change the DNS operator for a DNSSEC signed zone without "going unsigned".

However, it seems that there are several similar scenarios that differ more in organisational structure than in the operations needed:

- Migrate a zone from a publication pipeline dependent on one HSM to a new pipeline dependent on a new HSM.
- For an important zone that has a (usually in-house) DNSSEC pipeline it is a possibility to be able to have two independent pipelines for redundancy reasons. MUSIC would do this automatically.
- Similar to the previous case, a zone may have outsourced zone signing to the DNS operator, but would like to be able to have multiple independent DNS operators, including the signing part. Without MUSIC or similar software this is close to impossible to achieve today.

Current State of M^{U}S$_{I}$C And Next Steps?

The current system works fine, but is limited to only work with signers that support DDNS as an interface.

That said, there are some improvements that we would like to fix over the coming weeks.

- The interface to the API for the deSEC DNS service is almost done and will be complete soon.
- Today the only user interface is the CLI tool, **`music-cli`**. It would be nice with a web interface to easier present current state.
- For testing purposes we built a simple CDS/CSYNC-scanner to deploy in the parent of test zones. This is not really part of M$_{U}$S^{I}C but we still want to clean that up a bit.
- Although we don't know of any, there are of course some lingering bugs somehere that we need to expose and dispose of.

Code: `https://github.com/DNSSEC-Provisioning/music.git`

Contact: **`musicians@internetstiftelsen.se`**

RFC Adjustments for Multi-Signer

ICANN DNSSEC & Security Workshop
Wednesday, March 9th 2022

Shumon Huque
Email: shuque@gmail.com

Goal

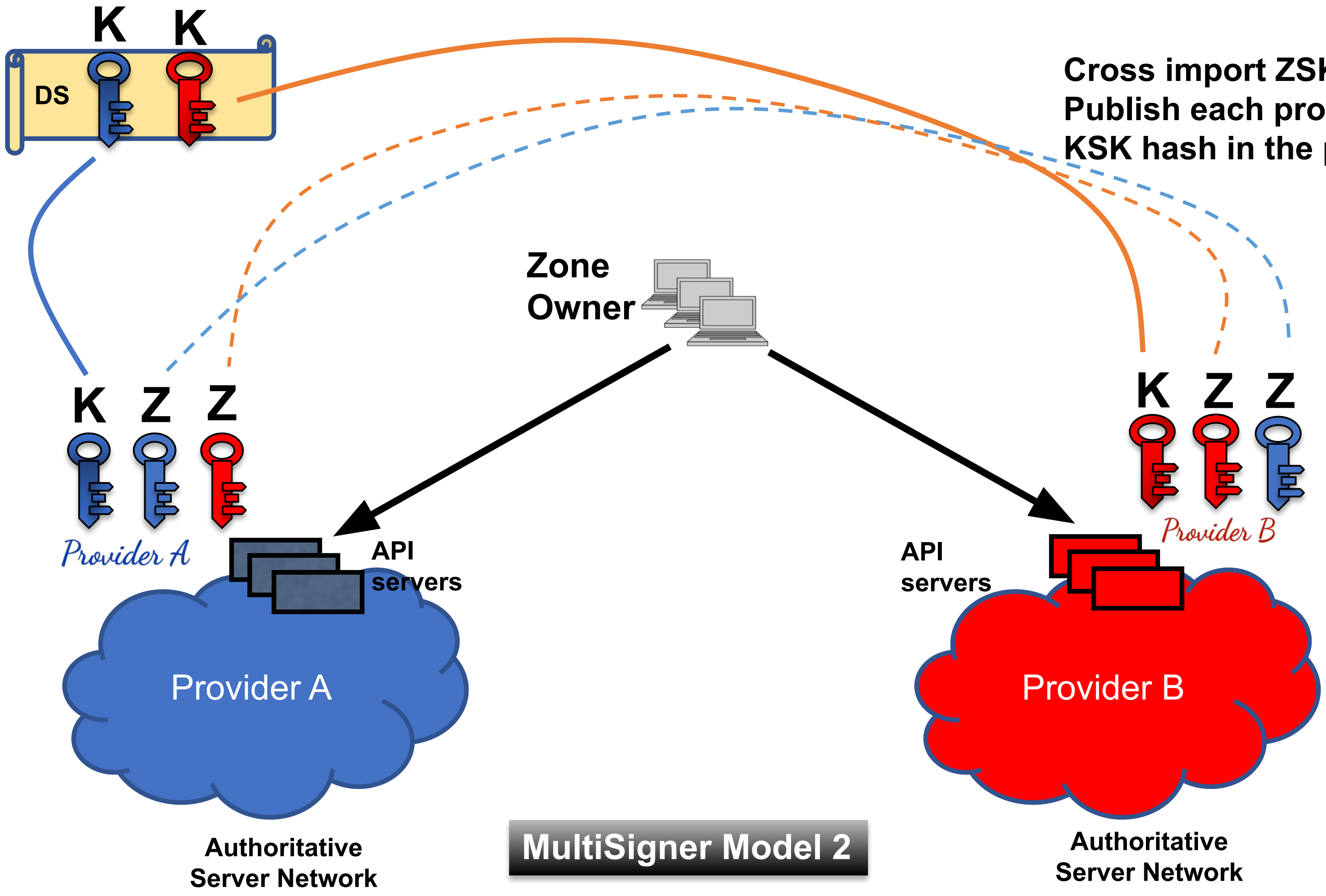
- Address some specific inconsistencies and limitations in the DNS protocol specifications (RFCs) that pose challenges for certain modes of multi-signer operation. And that are unnecessary and fixable.

Recap: RFC 8901: Multi-Signer DNSSEC

- Goal: allow multiple DNS providers to serve the same zone using their own DNSSEC signing keys.
- Introduces new key management mechanisms to make this possible.
- Two Models:
  - 1. Common KSK Set, Unique ZSK Set per Provider
  - 2. Unique KSK Set and ZSK Set per Provider

Model 2 is the most interesting, because it also offers a non-disruptive solution to inter-provider signed zone transfer. A model 2 multi-signer configuration can be viewed as a transitional state of a provider transfer.

Cross import ZSKs;
Publish each provider’s
KSK hash in the parent DS.



MultiSigner Model 2

Challenge: Differing DNSSEC Algorithms

- If providers use different algorithms, they cannot participate in a multi-signer configuration due to restrictions imposed by the current DNSSEC protocol specifications.
- This also prevents the use of the multi-signer protocol to non-disruptively transfer a signed DNS zone to a new provider that uses different algorithm(s).
- **This is an operational gap that should be closed.**
- We expect the presence of providers supporting distinct algorithm sets to be more common over time, since there will be more algorithms (RSASHA256, RSASHA512, ECDSAP256, ECDSAP384, ED25519, ED448, PQC1, PQC2, …)

RFC 4035: Protocol Modifications for DNSSEC

Section 2.2 (last paragraph)

```
   There MUST be an RRSIG for each RRset using at least one DNSKEY of
   each algorithm in the zone apex DNSKEY RRset.  The apex DNSKEY
RRset
   itself MUST be signed by each algorithm appearing in the DS RRset
   located at the delegating parent (if any).
```

RFC 4035: Protocol Modifications for DNSSEC

Section 2.2 (last paragraph)

```
   There MUST be an RRSIG for each RRset using at least one DNSKEY of
   each algorithm in the zone apex DNSKEY RRset.  The apex DNSKEY
RRset
   itself MUST be signed by each algorithm appearing in the DS RRset
   located at the delegating parent (if any).
```

This requirement cannot be satisfied if the DNS providers in a multi-signer configuration are using different signing algorithms.

RFC 6840: DNSSEC Clarifications

Section 5.11

This requirement applies to servers, not validators. Validators SHOULD accept any single valid path. They SHOULD NOT insist that all algorithms signaled in the DS RRset work, and they MUST NOT insist that all algorithms signaled in the DNSKEY RRset work.

RFC 6840: DNSSEC Clarifications

Section 5.11

```
This requirement applies to servers, not validators.  Validators
SHOULD accept any single valid path.  They SHOULD NOT insist that all
algorithms signaled in the DS RRset work, and they MUST NOT insist
that all algorithms signaled in the DNSKEY RRset work.
```

The 2 assertions in 4035 and 6840 are (arguably) not self consistent. If validators should accept ANY single valid path, then why should signers be required to sign zone data with one of EACH algorithm in the DNSKEY set?

RFC 6840: DNSSEC Clarifications

> **Section 5.11**
>
> This requirement applies to servers, not validators. Validators SHOULD accept any single valid path. They **SHOULD NOT** insist that all algorithms signaled in the DS RRset work, and they MUST NOT insist that all algorithms signaled in the DNSKEY RRset work.

The use of “SHOULD NOT” here seems to be part of the problem. If this had been “MUST NOT”, then multi-signer configurations could be deployed across signers with different algorithms (implementations permitting), and we could dismiss validators that attempted to enforce such a requirement as not compliant with the specification.

Simple proposal

- (1) Remove the requirement in RFC 4035, Section 2.2 (last para), and (2) in RFC 6840, Section 5.11 turn the “SHOULD NOT” into “MUST NOT”

Supporting wide range of validators

- Requiring signing by all algorithms allows a wide range of validators that may not have implemented support for all algorithms.
- No serious organization would only deploy an algorithm (such as a relatively new one) that did not yet have a critical mass of support by the deployed field of validators. If they deployed it, they would do so in conjunction with another well known algorithm, and sign with all of them.
- Extending this argument to multi-signer means that an organization would not choose a provider that only supported an algorithm not widely supported. The provider would also have a well supported algorithm, and sign with both.
- But there is no need to impose ‘sign with all algorithms’ *across* multiple distinct providers that all supported a disjoint set of well known algorithms.

But algorithm downgrade protection?

- Requiring signing by all algorithms allows validators to detect algorithm downgrade attacks (e.g. via signature stripping).
- But this rationale is not stated anywhere in the specs.
- And in the general case, only the zone owner knows the intent of their use of multiple algorithms (e.g. for multi-signer operation, for provider transfer, or whether they desire algorithm downgrade protection).
- Validators should not unilaterally impose requirements that interfere with the zone owner’s actual intentions.
- One option would be to add additional signaling to the protocol to allow precise expression & determination of this intent.

Where could this be signaled?

- In the DS record set, but it lacks flags. So, the usual approach is a DS record “hack”: create a new “pseudo” DNSSEC algorithm number, and use a DS record entry that references that algorithm number, but carries in its data field, the required signaling information.
- In the flags of the DNSKEY record(s).
- [Your idea here]

What needs to be signaled?

- Do not enforce requirement for all data to be signed by all available algorithms in the DNSKEY set
  - either because this is a multi-signer configuration, a zone in transition across providers with disjoint algorithms, or simply because the zone owner doesn’t care about algorithm downgrade protection.
- Enforce requirement for all data to be signed by all available algorithms in the DNSKEY set.
  - e.g. because the zone owner wants to provide algorithm downgrade protection, and wants to allow validators to be able to authenticate data using all algorithms, which will include the strongest available algorithm.
- Something else (something will always come up in the future)

Write up a proposal for IETF

- We are planning to write up a specific protocol enhancement proposal for consideration by the IETF DNS Operations Working Group.
- Collaborators welcome.

DNS(SEC) Views

https://dnssecviews.net

Pouyan Fotouhi Tehrani, Eric Osterweil,
Thomas C. Schmidt, Matthias Wählisch

Motivation

- Securing DNS zones is fairly straight-forward
- Authoritative nameservers provide consistent data

We have been monitoring this through SecSpider (`https://secspider.net/`)

HOWEVER

- Users rely on recursive resolvers
- Recursive resolvers follow different policies
- Timing, caching, multiple signers, etc. influence propagation
- Data from multiple sources may be combined to validate signed recods
- Infrastructure providers are interested to know how their services are observed by users

That’s why we built the **DNS(SEC) Views**!

Motivation

- Securing DNS zones is fairly straight-forward
- Authoritative nameservers provide consistent data

We have been monitoring this through SecSpider (https://secspider.net/)

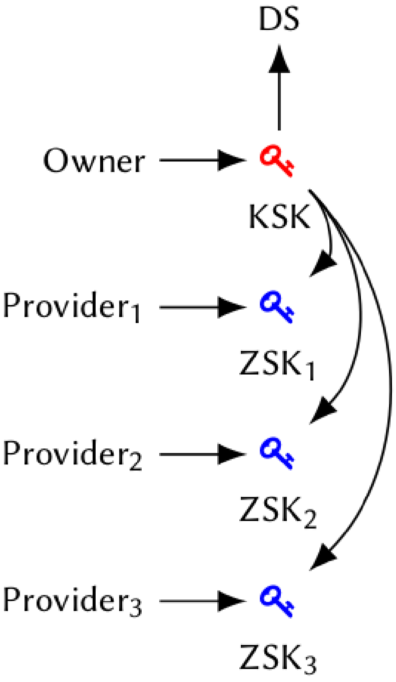
Goal: understand how the **distributed** nature of DNS and its **eventual consistency** (temporal aspect) is observed by and affects users

- Users rely on re
- Recursive resol
- Timing, caching
- Data from multiple sources may be combined to validate signed recods
- Infrastructure providers are interested to know how their services are observed by users

That’s why we built the **DNS(SEC) Views**!

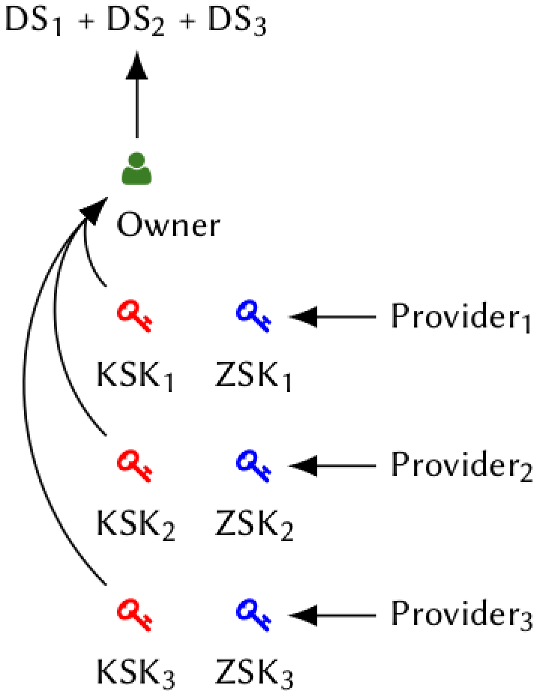
Use Case: Multi-Signer DNSSEC

Common KSK Set, Unique ZSK Set per Provider

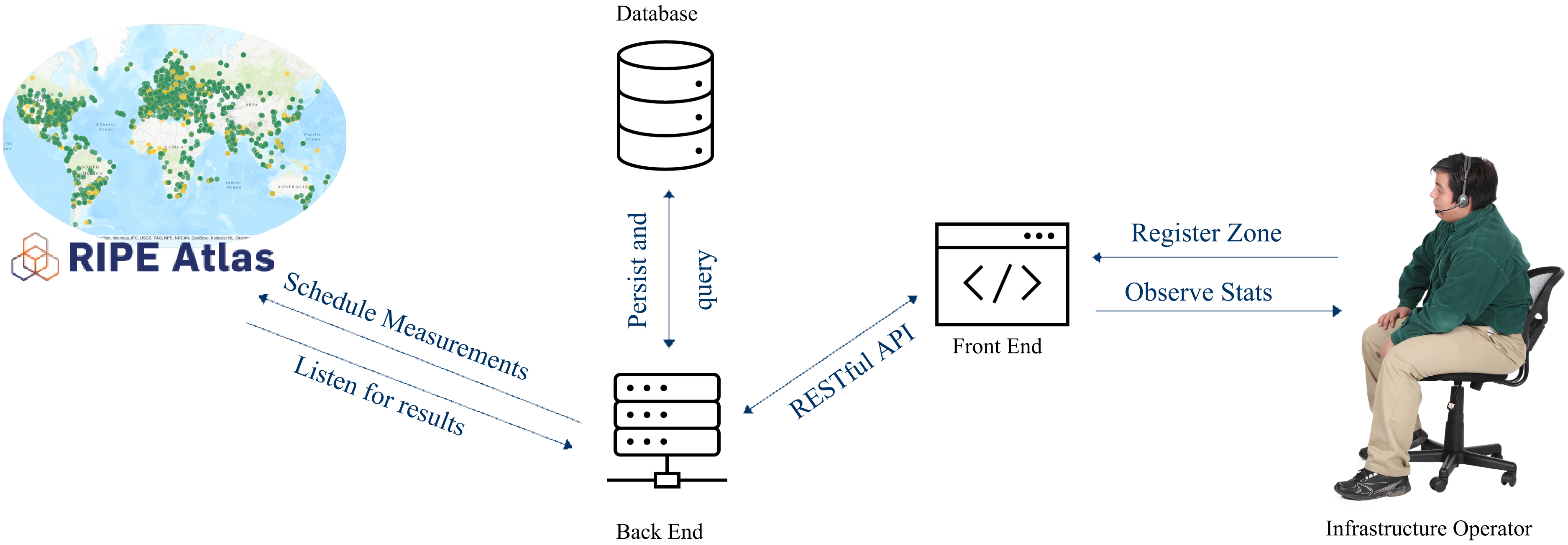


To verify correct deployment observations from various vantage points should simultaneously be collected.

Unique KSK Set and ZSK Set per Provider



System Overview



Database

Persist and query

RIPE Atlas

Schedule Measurements

Listen for results

Back End

RESTful API

Front End

Register Zone

Observe Stats

Infrastructure Operator

Approach: Collect Data

1. Find zone apex
2. Schedule regular measurements via RIPE Atlas for following records:
   - `DNSKEY`
   - `DS`
   - `NS`
   - `SOA`
3. Parse and serialize data into the DB iff:
   - Response is valid
   - Response is signed

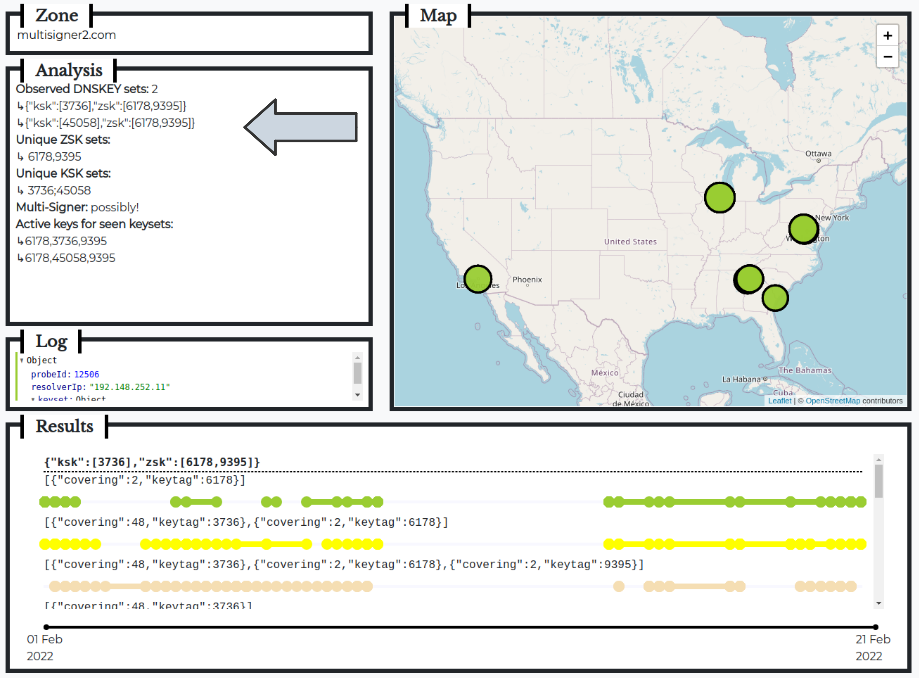
Executed by a set of random probes (currently only US)

Also record when each probes sees which RRSet and RRSIG

Approach: Provide Analysis

For any given zone:

1. Calculate different combinations of *observed* `DNSKEY` sets and active keys in use.



Zone

multisigner2.com

Analysis

Observed DNSKEY sets: 2
↳{"ksk":[3736],"zsk":[6178,9395]}
↳{"ksk":[45058],"zsk":[6178,9395]}
Unique ZSK sets:
↳ 6178,9395
Unique KSK sets:
↳ 3736;45058
Multi-Signer: possibly!
Active keys for seen keysets:
↳6178,3736,9395
↳6178,45058,9395

Log

```
▾ Object
  probeId: 12506
  resolverIp: "192.148.252.11"
  ▾ keyset: Object
```

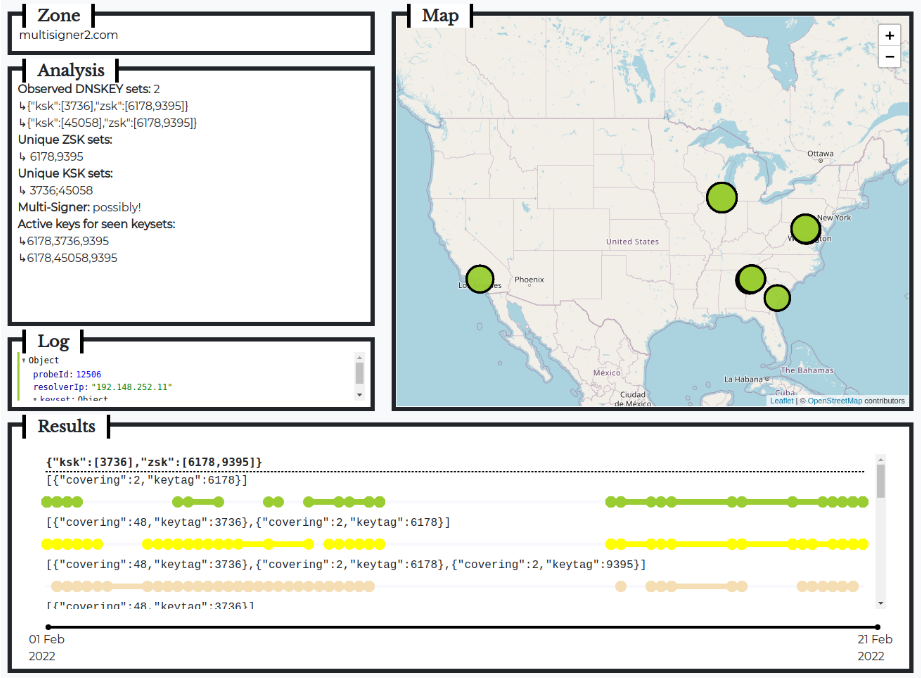
Results

```
{"ksk":[3736],"zsk":[6178,9395]}
[{"covering":2,"keytag":6178}]
[{"covering":48,"keytag":3736},{"covering":2,"keytag":6178}]
[{"covering":48,"keytag":3736},{"covering":2,"keytag":6178},{"covering":2,"keytag":9395}]
[{"covering":48,"keytag":3736}]
```

Approach: Provide Analysis

For any given zone:

1. Calculate different combinations of *observed* `DNSKEY` sets and active keys in use.
2. Color code each combination and calculate when each probe sees which combination.



Zone

multisigner2.com

Analysis

Observed DNSKEY sets: 2
↳{"ksk":[3736],"zsk":[6178,9395]}
↳{"ksk":[45058],"zsk":[6178,9395]}
Unique ZSK sets:
↳ 6178,9395
Unique KSK sets:
↳ 3736;45058
Multi-Signer: possibly!
Active keys for seen keysets:
↳6178,3736,9395
↳6178,45058,9395

Log

```
▾ Object
  probeId: 12506
  resolverIp: "192.148.252.11"
  ▾ keyset: Object
```

Map

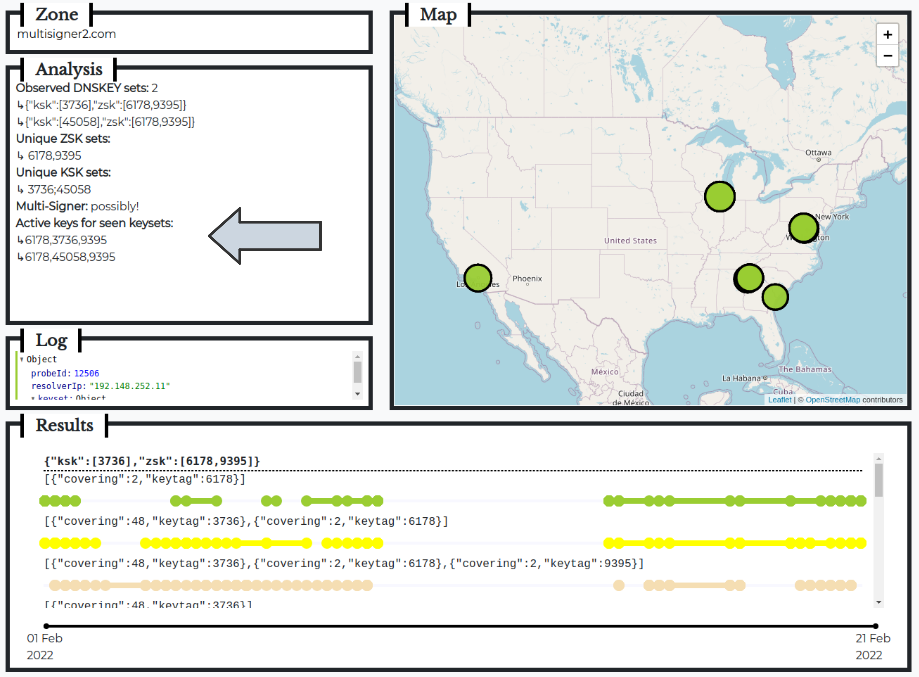
Results

```
{"ksk":[3736],"zsk":[6178,9395]}
[{"covering":2,"keytag":6178}]
[{"covering":48,"keytag":3736},{"covering":2,"keytag":6178}]
[{"covering":48,"keytag":3736},{"covering":2,"keytag":6178},{"covering":2,"keytag":9395}]
[{"covering":48,"keytag":3736}]
```

Approach: Provide Analysis

For any given zone:

1. Calculate different combinations of *observed* `DNSKEY` sets and active keys in use.
2. Color code each combination and calculate when each probe sees which combination.
3. Analyze for specific events or deployment models: ongoing key transitions, multi signer DNSSEC, etc.



Zone

multisigner2.com

Analysis

Observed DNSKEY sets: 2
↳{"ksk":[3736],"zsk":[6178,9395]}
↳{"ksk":[45058],"zsk":[6178,9395]}
Unique ZSK sets:
↳ 6178,9395
Unique KSK sets:
↳ 3736;45058
Multi-Signer: possibly!
Active keys for seen keysets:
↳6178,3736,9395
↳6178,45058,9395

Log

```
▾ Object
  probeId: 12506
  resolverIp: "192.148.252.11"
  ▾ keyset: Object
```

Map

Results

```
{"ksk":[3736],"zsk":[6178,9395]}
[{"covering":2,"keytag":6178}]
[{"covering":48,"keytag":3736},{"covering":2,"keytag":6178}]
[{"covering":48,"keytag":3736},{"covering":2,"keytag":6178},{"covering":2,"keytag":9395}]
[{"covering":48,"keytag":3736}]
```

Conclusion

- There is a measurable discrepancy between records at authoritative name servers and what recursive resolvers deliver
- *DNS(SEC) Views* gives operators the opportunity to follow their DNSSEC deployment from the perspective of clients in real time
- Aggregated data can be used to improve deployment practices and figure out acceptance criteria