CLOUDFLARE

DNSSEC ~~DNS~~ A Review of CloudFlare Development and Deployment

Martin J. Levy @ CloudFlare
(based on work by Filippo Valsorda, Jono Bergquist & Ólafur Guðmundsson)

Introduction

CloudFlare DNS (the background)

- How big?
  - 2+ million domains
  - Authoritative for 40% of Alexa top 1 million
  - 43+ billion DNS queries/day
    - Second to only Verisign
- 63+ Anycast datacenters globally

CloudFlare DNS offerings

- DNS for customers
  - UI based access; heavily linked to CDN/DDoS services
- DNS for partners
  - API based access; heavily linked to resold CDN/DDoS services
- DNS as a secondary service (vDNS offering)
  - Operates as an authoritative NS for TLDs (or significant domains)
    - Looks like a classic secondary service

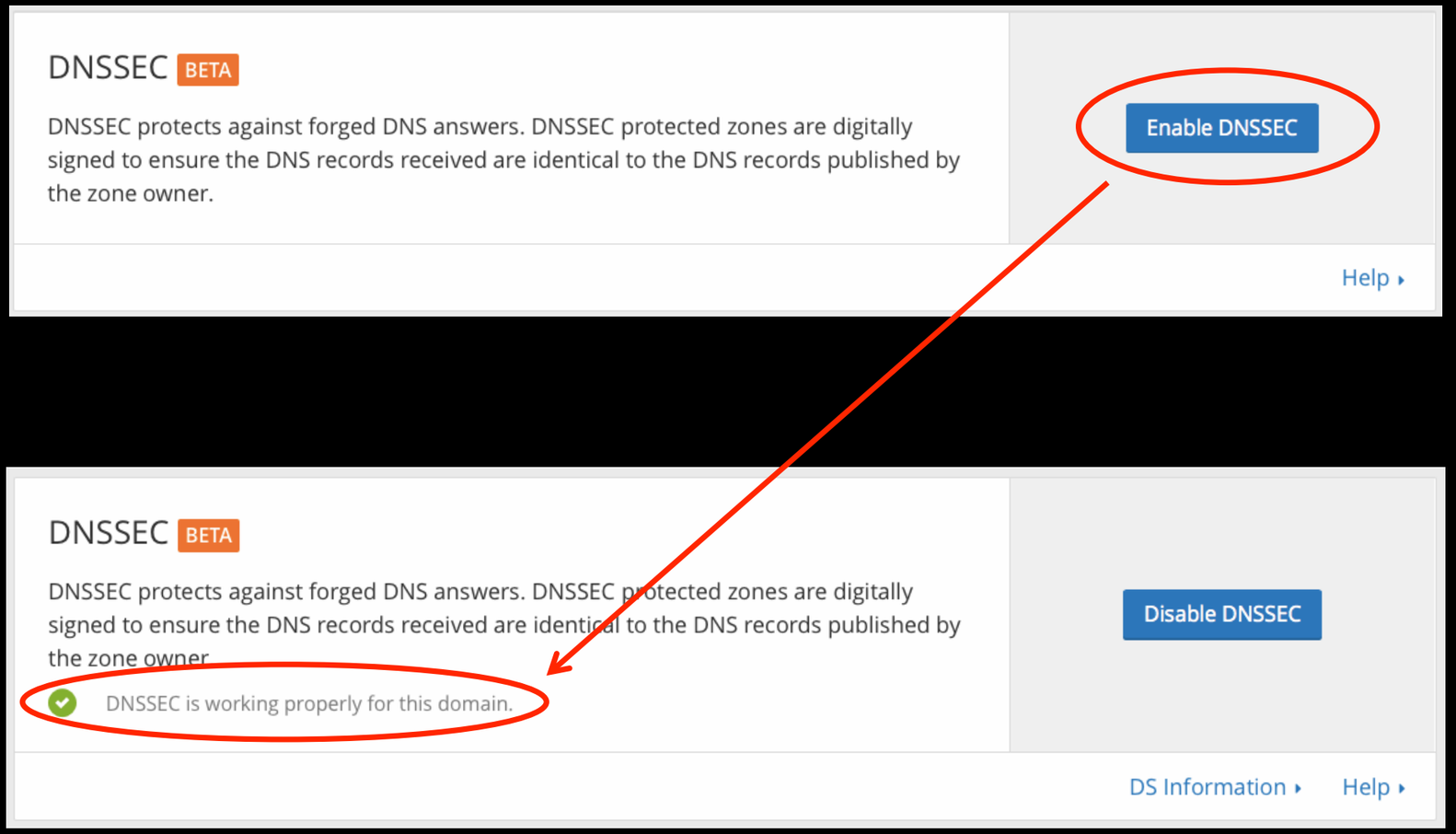
CloudFlare Goals & Solution

- DNSSEC at web scale
  - Scalable // DNSSEC for entire CloudFlare customer base
  - Simple // make it easy to consume
  - DNSSEC shouldn’t be for power users only! It should be for everyone!
- DNS & DNSSEC software structure for this large scale deployment
  - CloudFlare wrote our own DNSSEC systems (scale & speed dictated this)
  - CloudFlare uses modern crypto and sign-on-the-fly at the edge

CloudFlare Goals & Solution

- Changing the rules in order to deploy DNSSEC at large scale
  - Modifying and extending existing protocols to automate registrar interactions
    - Necessary to enable ease of use and deployment
    - Documented in RFCs or drafts (and code provided on github)
  - CloudFlare operates as a third-party DNS operator
    - i.e. Do not exit is many registration models
    - We are not the registrar or registry for most of these zones

It should be this simple to secure DNS



Scale

Why CloudFlare needs live signing

- Lots (lots!) of small, light traffic zones
- Heavily distributed network (45+ datacenters)
- **Dynamically generated records**
- Zone walking protection

Issues with live signing – Our solutions!

- Speed!
- Negative answers
- Key management

Our Constraints

- Keep size small, and don’t require full zonefiles

Speed

CloudFlare's DNS(SEC) overview

- RRDNS is our in-house DNS server written in Go
- Resilient against attacks and abuse
- No zonefiles, records are pulled from a global distributed database
- Full featured (dynamic answers, CNAME flattening, ...)
- DNSSEC is just a "filter" applied to the answer

Solving speed (and size): ECDSA P256

- ECDSA (Elliptic Curve Digital Signature Algorithm) P256 signatures
  - > 3x faster than RSA1024
  - Measured on OpenSSL 1.0.2 on our servers
- We (Vlad Krasnov) ported OpenSSL ASM to Go
- 21x speedup for the sign: https://go-review.googlesource.com/#/c/8968/
- Bonus: small signatures, small keys, modern crypto!
- Supported by most validators, working on registrars

https://tools.ietf.org/html/rfc6605

Solving speed (and size): ECDSA P256

RSA:
1,181 BYTES

```
$ dig +nocomments +nostats +nocmd +noquestion +dnssec DNSKEY ietf.org @8.8.8.8
ietf.org. 1572 IN DNSKEY 257 3 5 AwEAAavjQ1H6pE8FV8LGP0wQBFVL0EM9BRfqxz9p/sZ+8AByqyFHLdZc
HoOGF7CgB5OKYMvGOgysuYQloPlwbq7Ws5WywbutbXyG24lMWy4jijlJ UsaFrS5EvUu4ydmuRc/TGnEXnN1XQkO+waIT4cLtrmcWjoY8Oqud6lDa
Jdj1cKr2nX1NrmMRowIu3DIVtGbQJmzpukpDVZaYMMAm8M5vz4U2vRCV ETLgDoQ7rhsiD127J8gVExjO8B0113jCajbFRcMtUtFTjH4z7jXP2ZzD
cXsgpe4LYFuenFQAcRBRlE6oaykHR7rlPqqmw58nIELJUFoMcb/BdRLg byTeurFlnxs=
ietf.org. 1572 IN DNSKEY 256 3 5 AwEAAdDECajHaTjfSoNTY58WcBah1BxPKVIHBz4IfLjfqMvium4lgKtK ZLe97DgJ5/
NQrNEGGQmr6fKvUj67cfrZUojZ2cGRizVhgkOqZ9scaTVX NuXLM5Tw7VWOVIceeXAuuH2mPIiEV6MhJYUsW6dvmNsJ4XwCgNgroAmX hoMEiWEjBB
+wjYZQ5GtZHBFKVXACSWTiCtddHcueOeSVPi5WH94Vlubh HfiytNPZLrObhUCHT6k0tNE6phLoHnXWU+6vpsYpz6GhMw/R9BFxW5Pd PFIWBgoWk2/
XFVRSKG9Lr61b2z1R126xeUwvw46RVy3hanV3vNO7LM5H niqaYclBbhk=
ietf.org. 1572 IN RRSIG DNSKEY 5 2 1800 20161012164053 20151013154322 40452 ietf.org. ly4cVZGR0ITgmPy7ldU8mRqKl5glM5oKH+AA5hVfpqDfWnI5N
+jSlCQp k8T/t1TMUTGuV1ZyCmxrxPmi3lUSL0Bj3v3Y5G3aADsKRS8BDEHFPyX+ W8sHc2M/WhqOJunTEcwbBzcKw4J9osRErQ9TWb/HmE38LHtaoCpH+ott
+14UyVWpiR0s5STAM3leMllJgjxmKZZO1KiE0gJo3cg/x7Wg+OHtjdtg 6C1pYdcn/IZsiESNc4yVTJ/FobYa9gE0SnCoztYkDPb9lsMWxVDXScX6 oxjaUBLCBnftlmTlPOGMT
+lK4hYMaRnMl6UhpVk33PVggY6oq3Cs7KWe 4gaE9A==
ietf.org. 1572 IN RRSIG DNSKEY 5 2 1800 20161012164135 20151013154322 45586 ietf.org. IzE2grjgweI6NsfEoRVoqzqAk3JeWfKJ9aa0rEAV/
2o40VyHyiEpzTs2 DVznTFUfy3iiepMBHPjbP+fL1s5mE6ARJYLHsP2gc2TVD3eBTycmIzIV 4UE1nDzxzus9aXULwwaA3e8pI9FWmrdCp4kE1y4u9wlkELP/QAVCdiRm
Ppah0oPwdaeXvdXkZweTQJFI+BpDJR/nfPCK6p2oXh0qz4ojRMXcPuBL wW2l4zushmysI/u59MzO39bmVg/tiBV/pliBtAUBzfWBpOKiDGSrhXEp oMKuyG+9d8+b9qVvgETv/
sAg52KoGCunHIR6hm95KdQxFMcKz+JpWdHO tkcMGw==
$
```

...

ECDSA:
305 BYTES

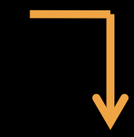
```
$ dig +nocomments +nostats +nocmd +noquestion +dnssec DNSKEY filippo.io @8.8.8.8
filippo.io. 3390 IN DNSKEY 257 3 13 DGpDkudNu/XQT1KmQkXFtKCfZPxHGV07qSTIcDXS33/WtT8UUG7LyxAg KznsRSFEhiQVR53E69/E57IFm8b6Zw==
filippo.io. 3390 IN DNSKEY 256 3 13 koPbw9wmYZ7ggcjnQ6ayHyhHaDNMYELKTqT+qRGrZpWSccr/lBcrm10Z 1PuQHB3Azhii+sb0PYFkH1ruxLhe5g==
filippo.io. 3390 IN RRSIG DNSKEY 13 2 3600 20151124213345 20150925213345 42 filippo.io. tvVoLbw4WEEAQDJYzioxfl+me2mPlvq6kWToLqnd/
2slz26ndLN3CZgn TBmKgRz9qZ19Sasus57NU7gnGjzCQA==
$
```

https://blog.cloudflare.com/ecdsa-the-digital-signature-algorithm-of-a-better-internet/

Solving speed (and size): ECDSA P256

Standard Go crypto:

BenchmarkSingleSignECDSA	832,295 ns/op
BenchmarkSingleSignRSA	6,003,261 ns/op



13x
speed up

Go with Vlad's changes:

BenchmarkSingleSignECDSA	60,806 ns/op
BenchmarkSingleSignRSA	3,124,274 ns/op

https://blog.cloudflare.com/go-crypto-bridging-the-performance-gap/

Negative Answers

Solving negatives: “Black Lies”

- To answer a NXDOMAIN normally we need:
  - Database lookups for previous and next name
  - 2 or 3 signatures (NSEC/NSEC3) - slow and big!
  - Previous and next name disclosure

Solving negatives: “Black Lies”

```
mipappstg.comcast.com.  3600    IN      NSEC    mmgr.comcast.com. CNAME RRSIG NSEC
mipappstg.comcast.com.  3600    IN      RRSIG   NSEC 5 3 3600 20150508165102 2015050
1134602 39162 comcast.com. OjKZ/h3bkK/AXs0kkg2Cbd13+aabCnCnp0sW9QHSrX8xcDO4+SdxYx+E
F6PtFUUYhOKA8u9dcir7nkqI2Et326oAPuV8gbY6cLB8sFTceK6Fz0VO /cIXrZyggy/VPf82FuBcoZsQnAb
erV0sI6RRbwjatPW65Wlo1bqKBrr9 Z7Q=
comcast.com.            3600    IN      NSEC    208.20.10.201.comcast.com. A NS SOA
MX TXT RRSIG NSEC DNSKEY
comcast.com.            3600    IN      RRSIG   NSEC 5 2 3600 20150508165102 2015050
1134602 39162 comcast.com. TdPdnLkg5Zf12/rgskPWGl94L+WigPn4AUD59p0qaX/T1fDmXU0g7WXH
38ROROuUGmBmu7HSqzCekxJf1S//4ohw07NP3gSTz5dtW6co0Hvw1E5n0 XaW+5nQC7pSBBjxa99DrUtPtpk6
2WACXuug/6A6lFcIovOppknsUl/12 fsQ=

;; Query time: 344 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu May 07 14:05:56 BST 2015
;; MSG SIZE  rcvd: 736
```

Solving negatives: “Black Lies”

- RFC 4470 introduces “white lies” for online signing:
  - Generate a NSEC on the name’s immediate predecessor, covering up to the successor (RFC4471)
  - Same with the wildcard
  - Solves: zone walking, database lookups
  - Still, 2 signatures to say one thing :(

Solving negatives: “Black Lies”

- Our solution: true lies. Just sign a NOERROR.
- Place a NSEC on the name, cover until the successor, set only the NSEC and RRSIG bits

```
$ dig +nocomments +nocmd +noquestion +dnssec missing.filippo.io @8.8.8.8
missing.filippo.io. 3599 IN NSEC \000.missing.filippo.io. RRSIG NSEC
Xnq2mlWUeUS1io2Kvps1v4H0TMBRl0 Dm9V4L9xgMHuU5nFrWT2BH92aPr2ug==
missing.filippo.io. 3599 IN RRSIG NSEC 13 3 3600 20151030081916 20151028061916
35273 filippo.io. fruM9IlupZDM7n3UAG8iB3XPFWj59jRS5Rn2PcepalCMAbr/pLi86gcP
zf9BGcXV+ncWA3uHWWExYjLkFRar5A==
...
;; MSG SIZE  rcvd: 359
$
```

Solving negatives: “Black Lies”

```
missing.filippo.io.     3587    IN      NSEC    \003.missing.filippo.io. RRSIG NSEC
missing.filippo.io.     3587    IN      RRSIG   NSEC 13 3 3600 20150507190048 201505
05170048 35273 filippo.io. Fb/xInfArVCMJWBDBqsbBPxiKsC1ueUyBFGi5lAHbjRBGAGm8sKDJx/l
YAO1bKYzJep3dRgQw5hS89JukD+m8w==

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed May 06 19:01:01 BST 2015
;; MSG SIZE  rcvd: 363
```

Solving negatives: “Black Lies”

- 1 signature op, no db lookup or zone walking
- The entire answer fits 512 bytes (actually, < 400!)
- End-user behavior is unchanged

```
missing.filippo.io.     3587    IN      NSEC    \003.missing.filippo.io. RRSIG NSEC
missing.filippo.io.     3587    IN      RRSIG   NSEC 13 3 3600 20150507190048 201505
05170048 35273 filippo.io. Fb/xInfArVCMJWBDBqsbBPxiKsC1ueUyBFGi5lAHbjRBGAGm8sKDJx/l
YAO1bKYzJep3dRgQw5hS89JukD+m8w==
```

Solving negatives: the “NSEC shotgun”

- But. To answer a missing type on an existing name, we still need to query the database for the NSEC bitmap
- That’s not even always possible! (Dynamic answers)

```
filippo.io. 3600 IN NSEC \003.filippo.io. A NS SOA MX TXT AAAA RRSIG
NSEC DNSKEY
```

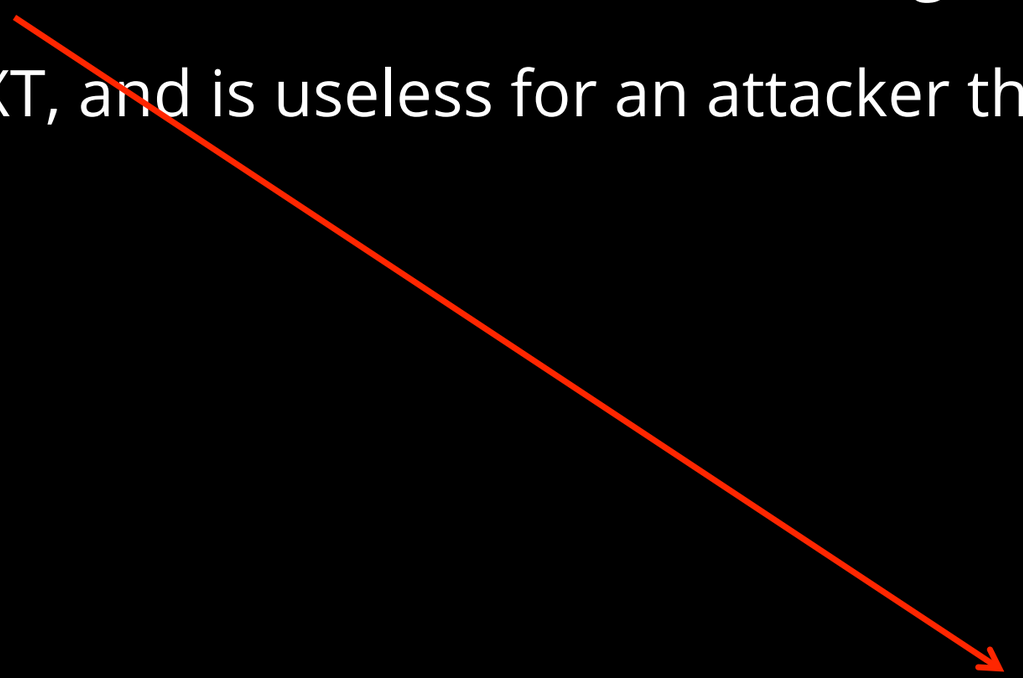
Solving negatives: the “NSEC shotgun”

- Step back: what is a NSEC? A denial of existence.
- “The types not in the bitmap don’t exist”
- So, let’s make a “minimally covering” one.
  By setting all possible bits in the bitmap!

```
filippo.io. 3600 IN NSEC \003.filippo.io. A NS SOA WKS HINFO MX TXT
AAAA LOC SRV CERT SSHFP IPSECKEY RRSIG NSEC DNSKEY TLSA HIP OPENPGPKEY
SPF
```

Solving negatives: the “NSEC shotgun”

- Asked for TXT and there’s no TXT? Set all the other bits that might exist.
- The NSEC is a valid denial for TXT, and is useless for an attacker that wants to replay it for other queries.



```
filippo.io. 3600 IN NSEC \003.filippo.io. A NS SOA WKS HINFO MX ~~TXT~~
AAAA LOC SRV CERT SSHFP IPSECKEY RRSIG NSEC DNSKEY TLSA HIP OPENPGPKEY
SPF
```

Key Management

Solving keys: centralized DNSKEY sets

- It’s live-signing, you need the ZSK at the edge (for now)
- Protect the KSK: keep it in a safe central auditable machine, distribute the signed DNSKEY sets to edges
- Short regular RRSIG validity, longer for DNSKEY
- Prepared to roll the ZSK fast at any time

https://blog.cloudflare.com/dnssec-complexities-and-considerations/
https://blog.cloudflare.com/dnssec-an-introduction/

Solving keys: global ZSK and KSK

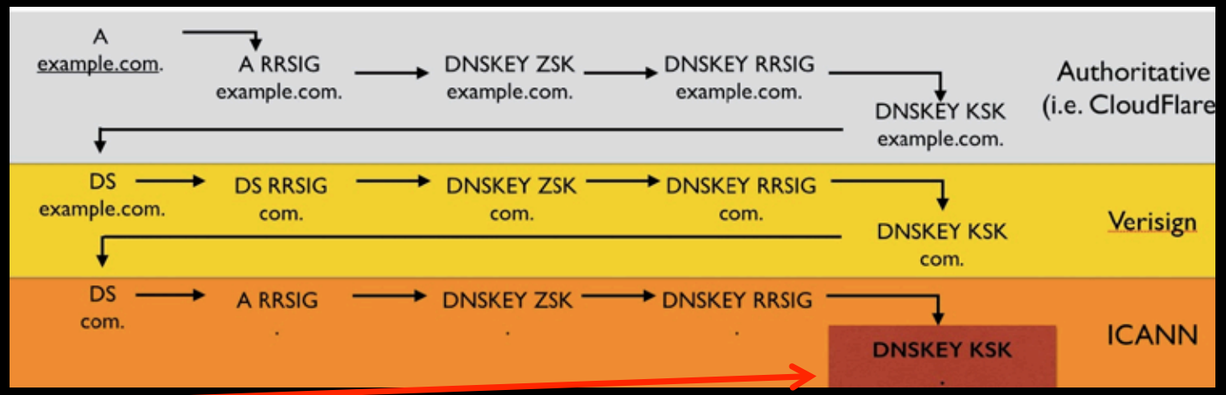
- No reason to have millions of ZSKs and KSKs:
  - all would be used/stored/rolled together
- Use a single KSK and a single ZSK with multiple names

```
$ dig +short DNSKEY filippo.io
256 3 13 koPbw9wmYZ7ggcjnQ6ayHyhHaDNMYELKTqT+qRGrZpWSccr/lBcrm10Z 1PuQHB3Azhii+sb0PYFkH1ruxLhe5g==
$

$ dig +short DNSKEY cloudflare-dnssec-auth.com
256 3 13 koPbw9wmYZ7ggcjnQ6ayHyhHaDNMYELKTqT+qRGrZpWSccr/lBcrm10Z 1PuQHB3Azhii+sb0PYFkH1ruxLhe5g==
$
```

“DS” – Simplify

ZSK's, KSK's (DNSKEY & DS records)

- The RRset of DNSKEYs are signed with the key signing key (KSK)
- Trust is conferred from the DNSKEY to the record though the RRSIG
  - if you trust a DNSKEY, then you can trust the records that are correctly signed by that key
- The DS is a hash of the DNSKEY
  - It belongs in the parent zone
- Repeat, all the way to root (.)



https://blog.cloudflare.com/dnssec-an-introduction/

How long does it take to ?

- Post a new selfie on Facebook and all your friends to be notified
  - few seconds (this is INTERNET SPEED)
- For a new domain to appear in the DNS?
  - less than 5 minutes in ICANN TLD's, random in others
- Move domain from one DNS operator to another?
  - long time limited by MAX(Parent NS TTL, Child NS TTL)
- Transfer a domain from one registrar to another one?
  - 1 sec … 5 days
- DNSSEC key rollover
  - many DAYS (your-mileage-may-vary)

Recent example: HBOnow.com

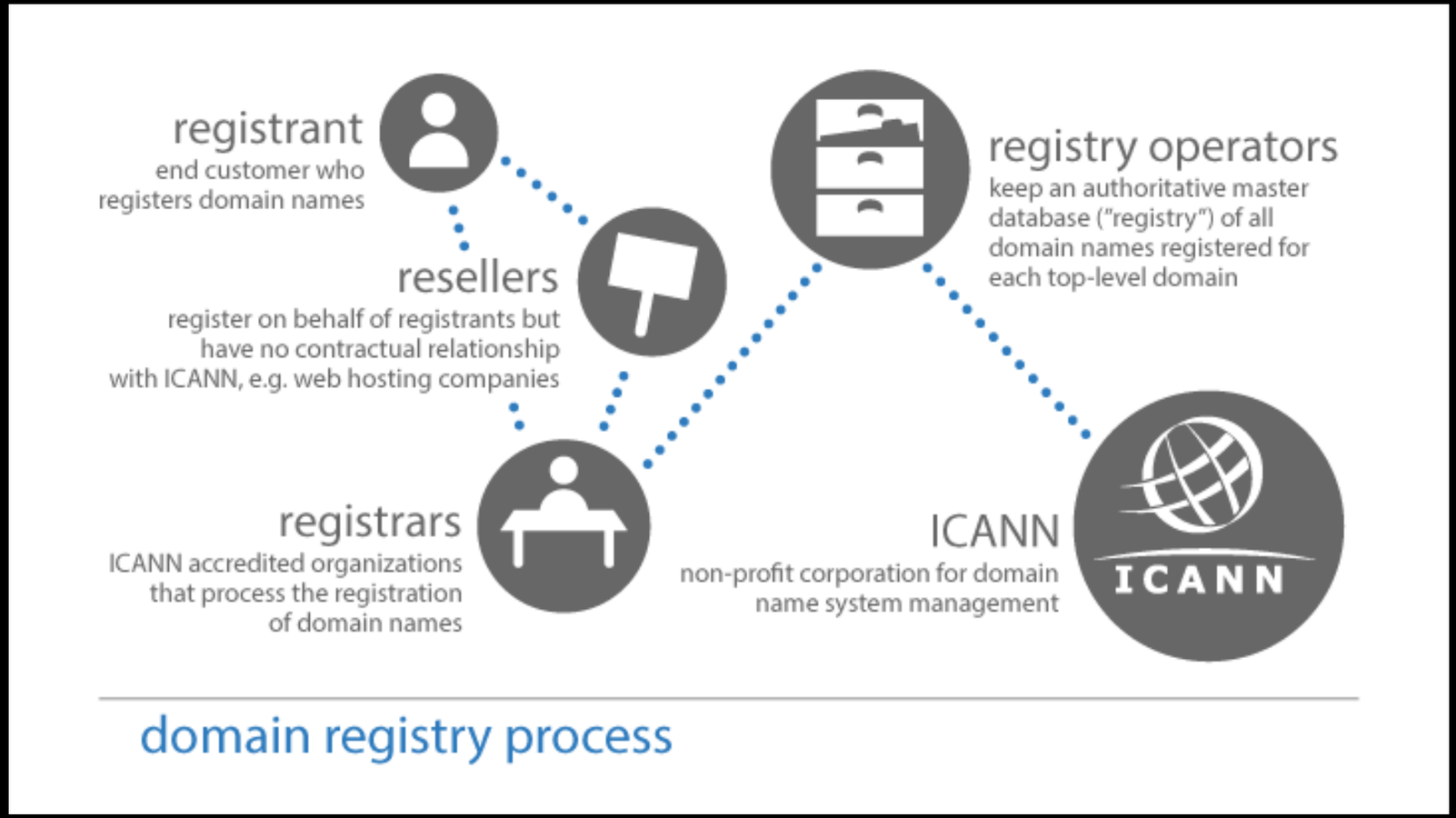
- Affected: Customers behind DNSSEC validating DNS resolvers
- Blamed: Comcast and ISP's for resolution failure i.e. blocking
- Root cause: HBO for not checking the domain was DNSSEC bogus
- Time to full recovery:
  - 1 day to purge DS from all caches after HBO made a change in .com registration system
- Mitigation: Temporary enable negative trust anchor by resolvers operators
- Side effect: Lots of non-polite Facebook and Twitter posts

Third party DNS operator (3-DNS)

- Definition: An entity contracted by “owner” of the domain to operate DNS on their behalf.
- Who: 3-DNS Operators include CDNs, DNS specialists, appliance vendors, friends, etc.
- Millions of domains are operated by 3-DNS
  - Many “important” domains are operated by 3-DNS
  - Some domains use vanity DNS server names, but routing/traceroute do not lie :-)

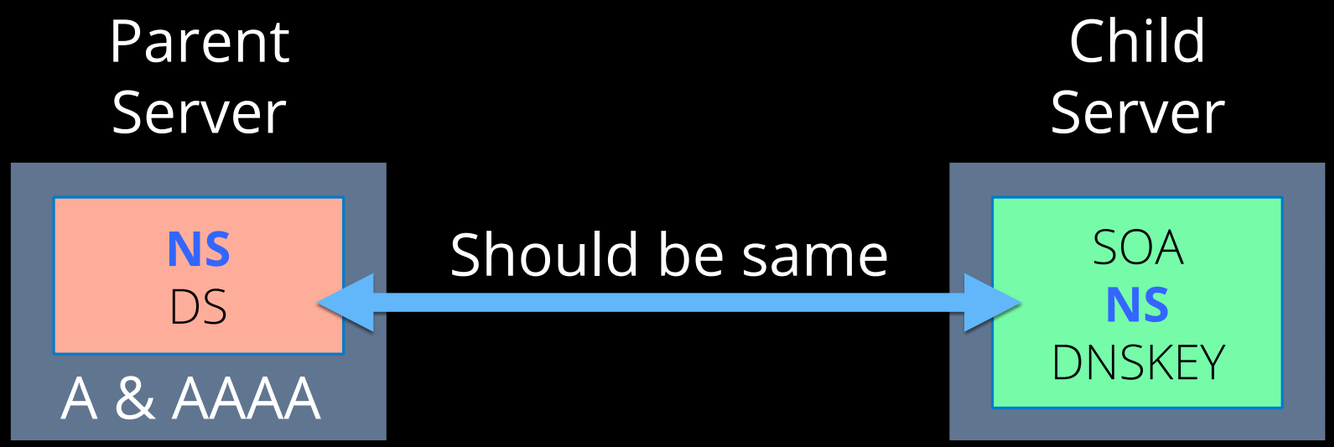
Domain Registry model:

- Includes Registries, Registrars, Resellers and Registrants.
- When developed did not envision 3-DNS



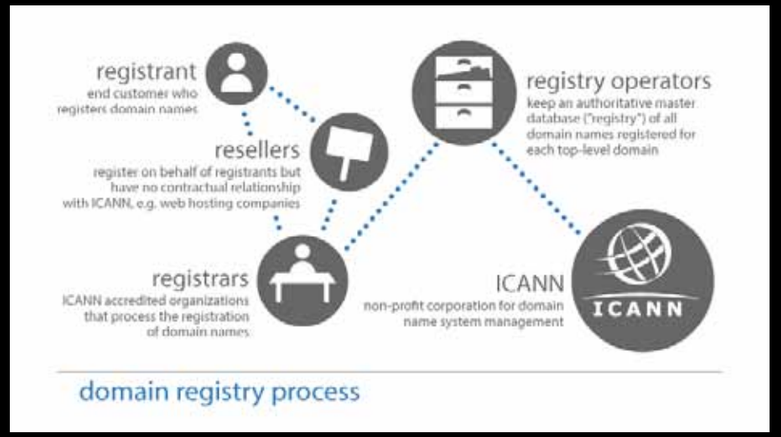
What info does 3-DNS want to maintain?

- NS records
- DS records
- A/AAAA records
  - need to be able to look up if glue is registered, add and delete.



What happens today?

- To change information in parent Registrant has to be in the loop
  - Not reliable, registrant may or may not take action
  - Not timely
  - Cut & Paste errors happen.
- Registrant can give access to registration account to 3-DNS
  - BAD idea !!



3-DNS as registars?

- Addresses part of the problem
  - Hard to become registrar in all ccTLD’s
  - Registrars/resellers are frequently partners with 3-DNS

What is desired by 3-DNS?

- Ability to gain authenticated permission to maintain delegation information for customers
- Ability to learn where to change information and connect there
  - WHOIS has last century contact information when it has any, frequently unusable

How can this be done?

- #1 In-band signaling
  - When DNSSEC is enabled
    - Child zone can advertise what the contents of NS and DS should be
      - via NS and CDS/CDNSKEY records when DNSSEC is present [RFC7344]
      - Not specified how to tickle right parental agent.
      - Not possible to say do it NOW!!

Vision – #2 Registry System interface

- If 3-DNS gets authenticated and authorized to make changes to NS/DS/ glue for specific domain, these changes can be injected into registration systems via
  - Registars/Resellers
  - Registries
- Hence: Updates can take place at Internet speed

Goal: DNS operators change < 4 hours

- Assume Changes in parent take less than 1 hour
- Operations:
  - provision new operator
  - change NS in parent and old operator (if possible)
  - wait for resolvers
- Precondition: Child and Parent NS
  - TTL <= 2 hours

Goal: DNSSEC KSK rollover in 6 hours

- Assume changes in TLD’s take less than 1 hour
- Operations:
  - update DNSKEY and/or DS;
  - switch KSK signing key;
  - purge old DS and DNSKEY records (Not in critical path)
- Child DNSKEY set < 1 hour TTL
- Child and Parent NS + DS sets TTL <= 2 hours

Call for Action

- Start discussion on what the right goals and policies are
- Proposed goals:
  - Get TLD’s to adopt lower TTL <= 2H
  - Give 3-DNS access to maintain Delegation information
- Bonus: get registries and registrars to support new DNSSEC algorithms by default in particular algorithm 13 ECDSA

ANY queries

Deciding to Neuter “ANY” queries

- An ANY query is a bad idea
  - Amplification, Information leaks, Non reliable responses, Expensive
- Applications (and people) assume ANY returns ALL records of all types
  - Firefox had a version that used ANY to retrieve A & AAAA in one query

https://tools.ietf.org/html/draft-ogud-dnsop-any-notimp-00
https://blog.cloudflare.com/deprecating-dns-any-meta-query-type/

CLOUDFLARE

Responses to neutering “ANY” queries

- Positive!
  - “We have this problem”
  - “We spend too much on bandwidth because of ANY queries”
  - “Yes stop this information leak”
- Negative!
  - “You are hurting Firefox and Qmail”
  - “you are idiots !!!!”
  - “I use ANY to debug my systems all the time!!!”

The qmail issue

- On DNSOP mailing list D. J. Bernstein (djb) wrote an explanation as to what Qmail is doing
  - Translation: Qmail uses ANY as a probabilistic optimization
  - Will fall back to normal resolution if ANY does not yield “useful” answer
- Hence: CloudFlare will not break qmail

https://mailarchive.ietf.org/arch/msg/dnsop/kXSApuM4i0WLoIo3_OhrCcAZ-cc

Why does CloudFlare care about “ANY”

- Expensive and complex to enumerate all RR Type for a name
  - We hate big answers
  - Sometimes not even available => incomplete answers
- Deploying DNSSEC with on-line signing on the edge at massive scale
  - Waste of effort to sign all the RR types the query origin does not care about

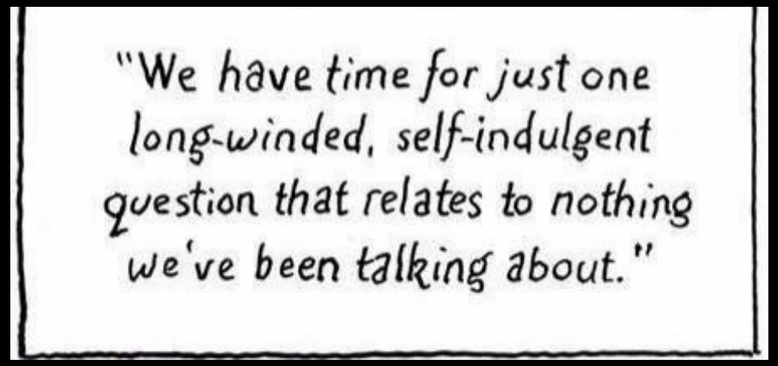
CloudFlare implemented “ANY”

```
$ dig +nocmd +nostats ANY cloudflarestatus.com @fred.ns.cloudflare.com
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56815
;; flags: qr rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;cloudflarestatus.com.    IN  ANY

;; ANSWER SECTION:
cloudflarestatus.com. 3789 IN HINFO "Please stop asking for ANY" "See draft-jabley-dnsop-refuse-any"

$
```

https://tools.ietf.org/html/draft-jabley-dnsop-refuse-any-01

CloudFlare implemented “ANY”

- No customers use HINFO in their zones → No need for new type
- We can generate this on the fly early in the processing
  - No need for multiple database lookups, discovery of all types, or multiple signatures
  - Simplified our code as we can remove ANY processing from various parts
- Cached as-is by resolvers → stops retries
- Accepted by resolvers → doesn’t break … applications



Summary – Questions & Answers

IXP peering information at PeeringDB

AS13335

Martin J. Levy, Network Strategy
@mahtin / @cloudflare
http://www.cloudflare.com/