DNSSEC Deployment for Resolver Operators

Jim Reid
jim.reid@uninett.no

OBJECTIVES

- An understanding of how Secure DNS works
  - Explain the key components of DNSSEC
    - Core concepts & jargon
- Configuring DNSSEC validation
- Understanding the main technical & business impacts
- Overview of tools & debugging

WHAT IS NOT COVERED

- Detailed configuration and management of Secure DNS software
- Choosing DNSSEC solutions
  - Hardware/software/services/tools/vendors
- In-depth DNSSEC debugging and troubleshooting
- How to design the DNSSEC solution for your environment
  - ... or deploy it

AGENDA

- Vulnerabilities in the DNS and how Secure DNS (DNSSEC) solves them
- Overview of DNSSEC
- How to configure DNSSEC validation
- Technical & operational considerations
- Business & policy considerations

THE DNS IN ACTION

This Section outlines shows what happens when a DNS lookup is made:

What actors are involved and what they do
The processes that take place
DNS weaknesses/vulnerabilities
How Secure DNS addresses these concerns
What Secure does and how it does it

What does the DNS do?

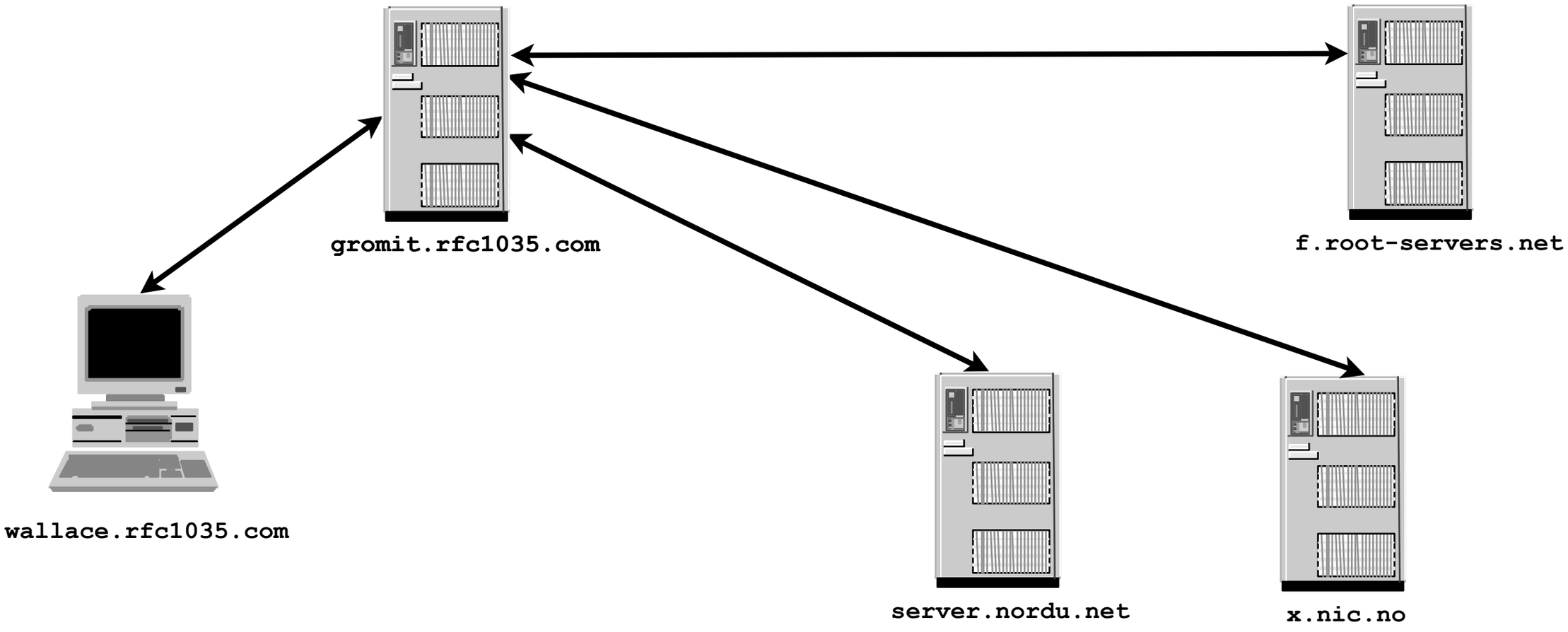
- What happens when someone clicks on a link or types in a domain name into their web browser?
  - DNS finds the IP address of the web server
  - Browser makes a connection to that IP address
  - Web pages fetched over that connection
- But what's going on behind the scenes?

DNS In Action

- Web browser on **`wallace.rfc1035.com`** wants to connect to Norid's web server, **`www.norid.no`**
- DNS maps the domain name (**`www.norid.no`**) into its current IP address, 158.38.130.37
  - Something on **`wallace`** makes a DNS query to find out which IP address its web browser needs to contact
  - **`wallace`** sends a DNS lookup to its local DNS server, **`gromit.rfc1035.com`**
- What actually happens?

Remember this?

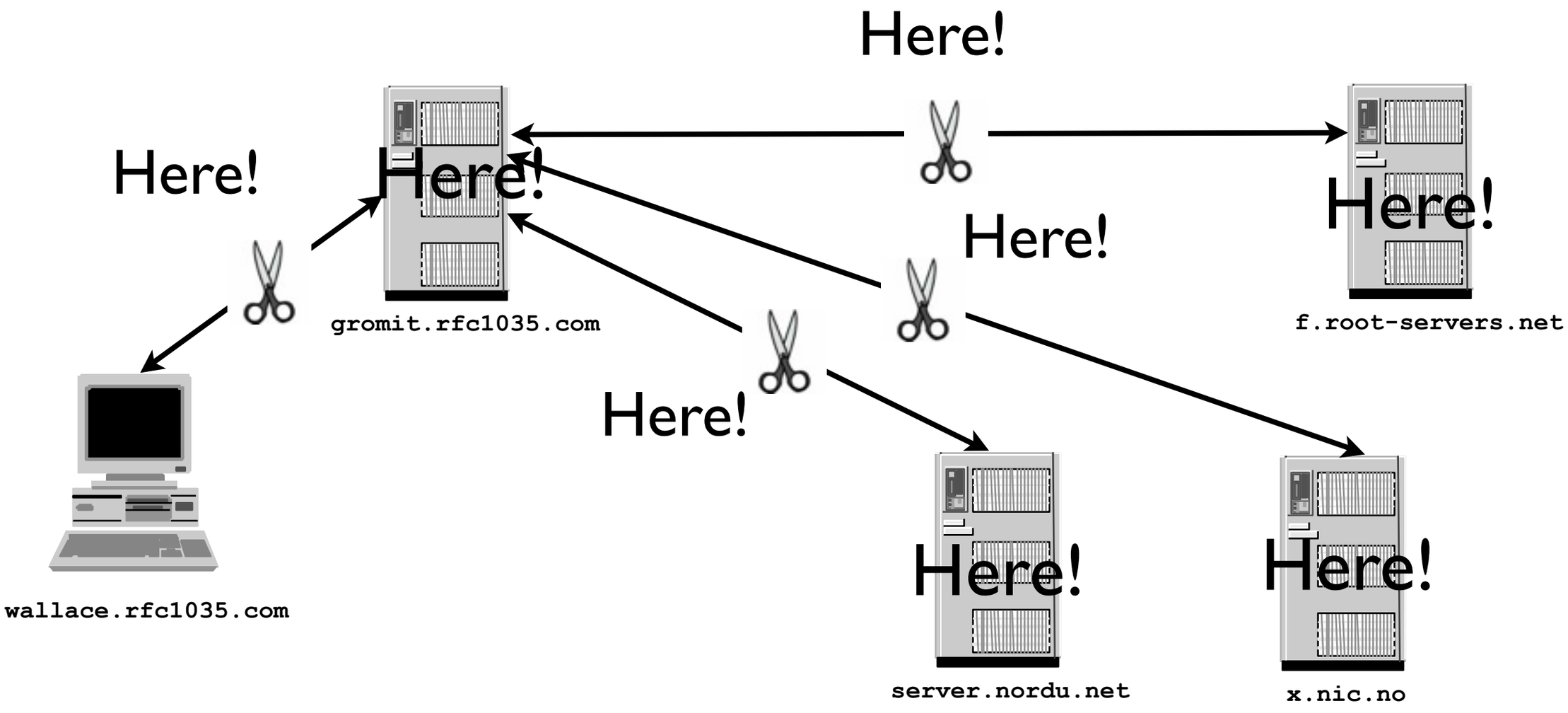
`gromit` returns **`www.norid.no`**'s address to **`wallace`**, who has been patiently waiting for an answer to the query it made



What's Wrong With That?

- Nothing: it all works just fine…..
- BUT there's no authentication at all!
- A client can't tell:
  - Where an answer **really** came from
  - If the server that replied is telling the truth or not
  - If it received **exactly** what the server sent
    - This applies to **`wallace.rfc1035.com`**'s query and the lookups **`gromit.rfc1035.com`** performed to resolve that query

So where are the vulnerabilities?



Attacking the DNS - 1

- Bombard client or resolving server with forged answers
  - Guess what the outgoing query might be
  - Successful Kaminsky attack "predicts" Query IDs
  - Brute force might well be viable
- Intercept a response packet & modify it
  - Tends to only work well if adjacent to client or server
- Set up a fake name server for some zone
  - Trick other name servers into querying the fake one
- Inject bogus data into caches
  - Cache poisoning attacks

Attacking the DNS - 2

- Take control of the name server(s) for some zone
  - Make them answer with false data
- Compromise the registry
  - Gain unauthorised access to registrar account and change the victim zone's delegation to point at bogus name servers
  - Several prominent examples recently:
    - New York Times, twitter, google.*ccTLD*
- Evil routing/peering tricks to hi-jack traffic
  - Introduce bogus routes for the root servers (or the name servers for any other "interesting" zone)

What Does This Mean?

- A regular DNS client really can't be sure of anything:
  - Did a lookup for **`www.norid.no`** really get answered by the **`norid.no`** name servers?
  - Did it get what a real **`norid.no`** name server actually sent?
  - Is the name server that answered telling the truth, the whole truth and nothing but the truth?

OK, What Does This Really Mean?

- Did the DNS provide the actual address of Norid's web/mail/*whatever* server?
  - Is my web browser talking to the One True **`norid.no`** web site?
  - Can I be sure my email is going to the **`norid.no`** mail server?
- Feel free to replace **`norid.no`** with your favourite domain name….
  - **`amazon.no`**, **`ebay.com`**, **`google.no`**

Don’t Panic!

- DNS is only now emerging as a target for attackers
  - Plenty of easier victims elsewhere
- DNS problems have been known about for a long time
  - IETF started working on this in the late 1990s
- The solution is now being deployed, Secure DNS
  - Sometimes called DNSSEC: DNS Security Extensions

What Secure DNS Proves

- Data integrity
  - Verify what was received was exactly what the name server sent
- Non-repudiation
  - Authenticate who/what signed the data
- Name server authenticity (in theory anyway)
  - An answer for **`foo.example.com`** comes from the genuine name servers for **`example.com`**
  - Should be a chain of trust to the root

What DNSSEC Can't Do

- Prevent/thwart denial-of-service attacks
- Stop name server compromises
  - Buffer overflows
  - Environment variable leakages
- Provide confidentiality of DNS data
  - The DNS is public after all...

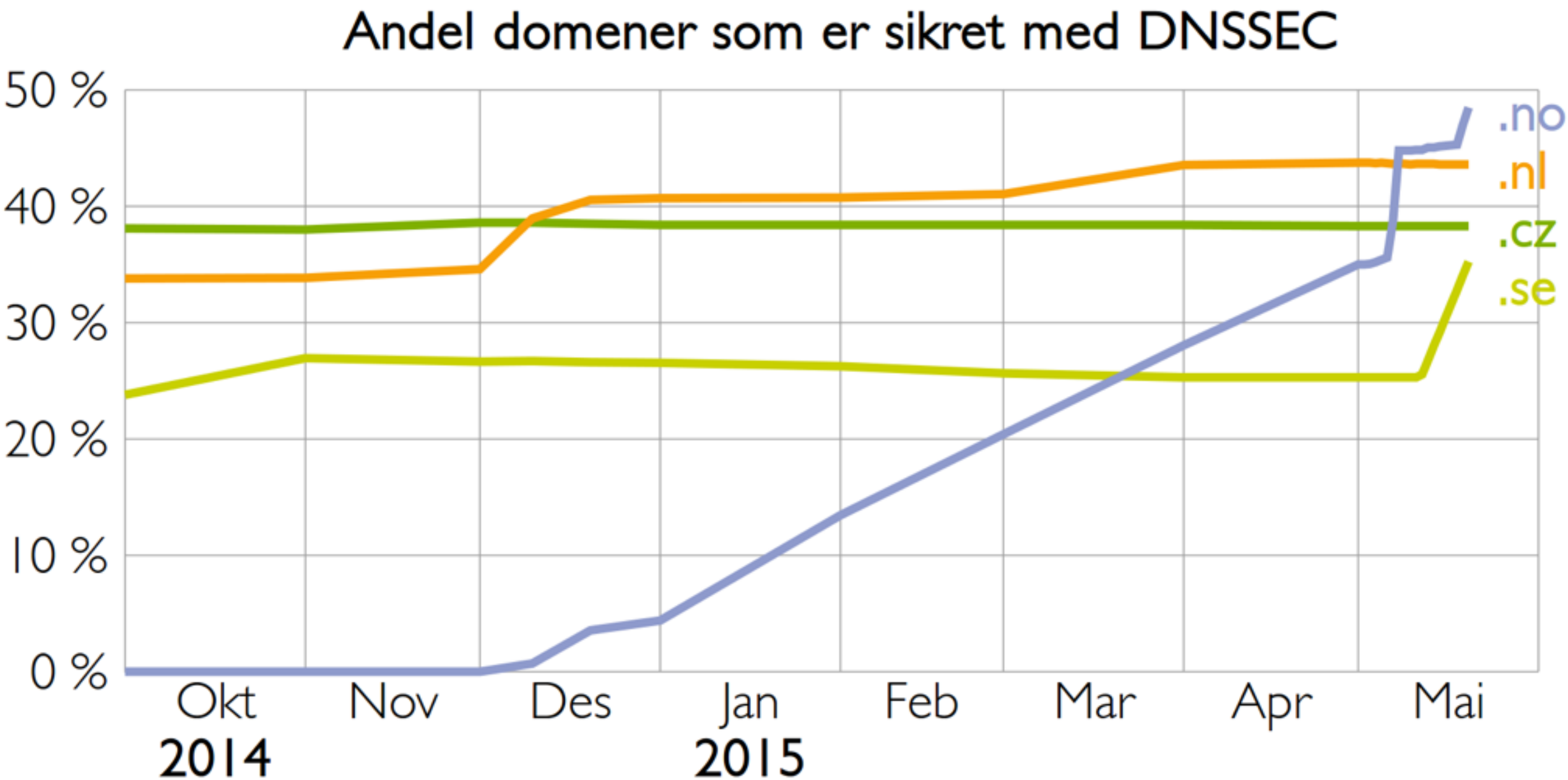
DNSSEC Overview

- Underlying technology is cryptography and digital signatures
  - Cryptographic hash functions (SHA family, MD5)
    - Public key crypto: RSA, DSA, ECC
- New resource records
- New tools
- New admin procedures

DNSSEC Deployment

- Swedish ccTLD was first, September 2005
- Internet root got signed July 15th, 2010
- A very, very cautious roll-out for obvious reasons
  - Awkward political problems too
  - No one organisation has the "master key"
- Nice animation here:
- **https://www.dnssec-deployment.org/wp-content/uploads/2013/09/cctld-2013-09-10.gif**
- Now it's Norway's turn :-)

DNSSEC in Norway Today



DNSSEC: The Big Picture - 1

- Cryptographic hash functions:
  - MD5, SHA-1, SHA-2, etc.
- Two key properties
  - Different inputs cannot generate the same hash value
  - Can't compute original input from the generated hash
- MD5 now deprecated and SHA-1 is on the way out
  - Researchers have found hash collisions
- SHA-2 (SHA256) is preferred/recommended
  - ... for now

DNSSEC: The Big Picture - 2

- Public key (or asymmetric) cryptography
  - Uses public/private key pairs:
    - RSA, DSA, Diffie-Hellman, ECC
    - Something encoded with private key can be decoded with public key and *vice versa*
- Concept of digital signatures
  - "Sign" something with a private key and verify it with the corresponding public key
  - Can be used to authenticate stuff: transactions, data, people, etc.

DNSSEC: The Big Picture - 3

- Public key crypto can be slow
  - Not good for large data sets or volumes of data
- DNS data can be large (e.g. 64KB TXT records)
- Compute secure hash of each RRset
- Sign hash with a private key
- Publish signature and public key in the DNS
  - => New resource records

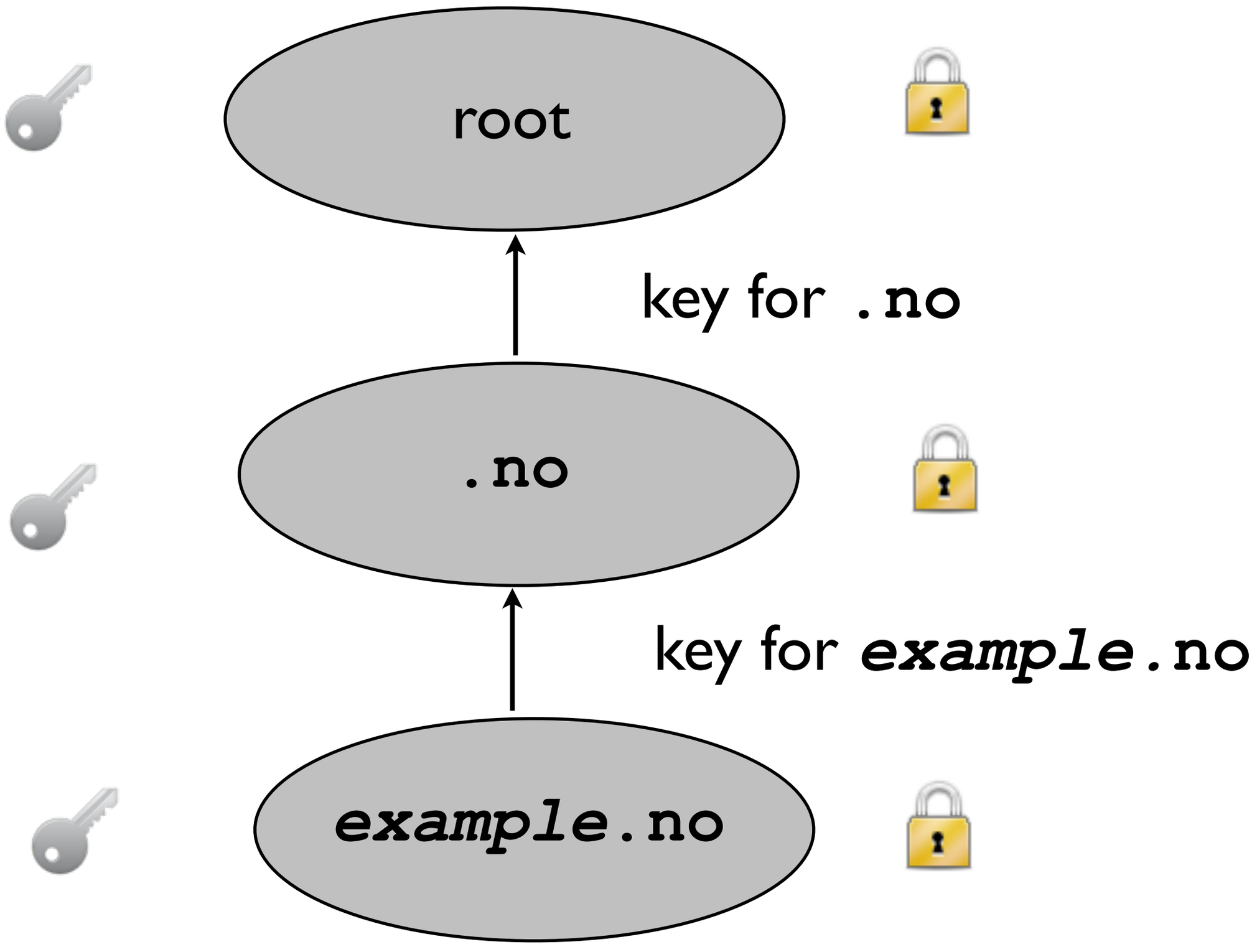
DNSSEC Signatures

- Don't explicitly sign the actual DNS data
  - Sign a hash of the data instead
  - Less data to sign
- Names must be normalised to a canonical form:
  - All in lower-case
  - Fully qualified domain names
  - Handled automatically by signing tools
- Validators need to know what was the original input to the cryptographic hash function!

The Chain of Trust

- Zone administrator generates public/private key pair
  - Uses this to sign RRsets in the zone
  - Publishes signed zone
- Parent zone signs child's key
  - Parent's signature of child's key is itself signed by the parent's private key
- Process repeats all the way up to the root
  - The one (and only?) Trust Anchor
  - Everyone knows and trusts the root's public key

DNSSEC in Pictures



Validation

- Validating resolver computes hash value for the returned RRset (i.e. the response from the DNS)
- Response also includes the signature for that RRset
- Validator gets public key for that signature from DNS
  - It has to validate that key too...
- Applies public key to the signature to get the hash that had been signed
- If that hash matches the one it calculated itself, all is well
- If not, Something Bad has happened

Validation Failures - I

- If the two hashes are not the same, it probably means one of the following:
  - The wrong key(s) were used somewhere
  - What's been received was not what was orginally signed
    - Something tampered with the DNS packets en route
    - Something is sending spoofed answers
- Implementation errors:
  - Bugs in crypto code or hash algorithms
  - Broken DNS software

Validation Failures - 2

- There's no special error code in the DNS protocol to signal that DNSSEC validation failed
  - This is **very** annoying and stupid
- Standard DNS **`SERVFAIL`** error code is returned for all DNSSEC validation failures
  - "An unrecoverable error occured. Give up."
- How does a resolver operator tell the difference between regular **`SERVFAIL`**s and DNSSEC ones?

Secure DNS Standards

- Defined in RFCs 4033, 4034 & 4035 (DNSSEC-bis)
  - Obsoleted & cleaned up a raft of earlier specifications, enhancements & extensions:
    - RFC 2535, RFC 3008, RFC 3090, RFC 3445, etc.
      - RFC 2535 was the first, doomed attempt
- Further refinement in RFC 5155 (DNSSEC-ter)
- Operational practices & clarifications defined in RFCs 4986, 6781, 6840, 6841, 6944, 6975, & 7344

A Never Ending Task?

SUMMARY

This Section has decribed:

How the DNS works and its vulnerabilities

An outline of what Secure DNS does

Why DNSSEC is a continually moving target

DNSSEC in DETAIL

It's now time to look at Secure DNS in more detail.
This Section explains:

The Resource Records needed for Secure DNS
- DNSKEY, RRSIG, NSEC, DS
- NSEC3, NSEC3PARAM

The two versions of DNSSEC in use:
- DNSSEC-bis (RFC 4033, 4034 & 4035)
- DNSSEC-ter (RFC 5155)

Ancient History

- First specification for Secure DNS was completed in 1999: RFC2535
- Advantage: it worked!
- Disadvantage: it was beyond impractical to deploy
  - Changing the root key meant **everything** had to re-sign
- Initial design influenced development of successor Secure DNS protocols
  - Similar concepts and resource records

DNSSEC-bis

- Second attempt to define Secure DNS
  - Fixed the problems in RFC2535
- Results documented in RFCs 4033, 4034 & 4035
  - Published in 2005
- Last minute problem emerged during standardisation
  - Caused further uncertainty and delays
    - Use current standard or wait for the next one?
    - When would work on the next version be done?

The DNSKEY Record - I

- The public key component
- Format:

name **DNSKEY** ***flags proto algorithm pubkey***

The DNSKEY Record - 2

- ***`flags`***
  - What the key can be used for
  - Now always zone: DNSSEC
- ***`proto`***
  - Protocol identifier: must now only be DNSSEC
  - Keys for other applications should use other RR types
- ***`algorithm`***
  - Crypto/hash algorithm: RSA/MD5, DSA/SHA-1, ECC, etc.
- ***`pubkey`***
  - Base-64 public key

An Example DNSKEY Record

```
rfc1035.se. 86400 IN DNSKEY 256 3 5
   AQPLoPeC9mnoh6xhosTwYPAdNMsjjRZNsvFAYq/aXLl
   +mwWsqFg8S95k1ikOMmgZcqjyPd2tza76osS9xb/
   uS4Ll6JlWTMv9T3UwulOeVeABtxHyUEuxLWeVX9bgeRUhBcry001Cs
   8hgnxGA0y66M18RYMT80u8KZVxLtZNJMp5ooRNQpQM60BdQCmi5jCf
   FfLojuzmNj/Y9yreELLh7pNoxSxoWw/Lu+mcmp93RaZ2Yvk/jUeTYn
   +6EyhTgh1Vw353SofkfM1To1xOdPIqicmuu
   +exqzIEKnsOeZVdJVoSGIOG2J6LWxgCsymz72Oqq5JUZjFWNtxr5Lu
   lJBa80jqZ5
```

Public RSA/SHA-1 key for **`rfc1035.se`**

It's a zone signing key (ZSK) - flags field is 256

The RRSIG Record - I

- A digital signature for some RRset
  - RRset: resource records with same name, class, type and TTL
  - Horribly, horribly complicated
  - Format:

```
name RRSIG type alg labels ottl \
sig-exp sig-inc key-tag signer sig
```

The RRSIG Record - 2

- ***type***
  - The RRset type that the RRSIG record signs
    - A, MX, SOA, etc
- ***alg***
  - Crypto algorithm
    - Same as in the DNSKEY record
- ***labels***
  - Number of labels in the name that are signed
  - Ugly kludge for wildcards: **`*.example.com`**

The RRSIG Record - 3

- ***ottl***
  - Original TTL of signed RRset
- ***sig-exp***
  - Time when the signature expires
  - **NOT** the same thing as the TTL
- ***sig-inc***
  - Time when the signature is valid from

The RRSIG Record - 4

- ***key-tag***
  - Short-cut to identify the key
    - Helps when there are 2 or more keys
- ***signer***
  - Name of the public key to validate the signature
- ***sig***
  - Base-64 encoding of signature

An Example RRSIG Record

```
rfc1035.se. 86400  IN RRSIG   SOA 5 2 86400
   ( 20060502155359 20060402155359 50252 rfc1035.se.

  xztN9UMGjQugvgOydMi+o/UreONZPKN00Ft3FYwwi0J7LgwbMFE9i
   +nyKkW7cI/ifmSCMNbYXj37NyoKeCbRVolAD33ZWKwCTkHNbdM
   +rZSz0+Zc+NAqQNYlPdMASG5MpdHnSYCkP82kui4aYVX2lbyy0fLyae
   +EalQMuI/GtSabi7/
   bUY027i7cX8NOe0ALr4Yzmv7nAFnnfla0+a6oXgd4sfkjnfD1m
   TIfVCXhzJYvAeahMSiVhk4nmB/Ys+iulsLqqRpN0HZGyPzzMxam/MJhtb
   +dS199sxxFw/oHw1Hd2e01MntateRwafe
   5FlUsb8d7MTjCmIUzQQqpwof1mg== )
```

RSA/SHA-1 RRSIG record for **`rfc1035.se`**'s SOA record signed with the RSA key for **`rfc1035.se`** that has “footprint” 50252

The NSEC Record

- For proving a name or RR type does not exist
  - Can't just sign NULL string!
- Format:

name **NSEC** ***next-name types***

- ***next-name***
  - Name of alphabetically next record in the zone
  - Last name points back to zone's SOA record
- ***types***
  - Resource record types that exist for the name

An Example NSEC Record

```
gromit.rfc1035.se. 86400 IN NSEC \
  hutch.rfc1035.se.   A RRSIG NSEC
```

Next name in zone after **`gromit.rfc1035.se`** is **`hutch.rfc1035.se`**

A, RRSIG and NSEC records exist for **`gromit.rfc1035.se`**

The DS Record

- Delegation Signer (DS)
- Format:

name **DS** ***keytag alg type digest***

- ***name*** - name of child key
- ***keytag*** - hint for child key
- ***alg*** - crypto algorithm of key
- ***type*** - crypto hash algorithm used to create ***digest***
- ***digest*** - hash of child's KEY

An Example DS Record

```
rfc1035.se. 86400 IN DS 33840 5 1 \
  7B9802EA312D73D0EB9E499975796B598AD6165E
```

SHA-1 hash of an RSA/SHA-1 key-signing key for **`rfc1035.se`** that has an id-tag (footprint) of 33840

DS record goes in the parent zone of the signed delegation, **not** the child

How DS Records Are Used

- DS record goes in the parent zone and gets signed by parent's zone key(s)
  - Think of it as a sort of meta-NS record
  - Proves a delegation exists and is "secure" (DNSSEC-aware)
  - Essentially a signature of a key for a child zone
  - Child zone's key is actually a key-signing key:
    - Signs the child's key(s) that sign the child's zone

Zone- & Key-Signing Keys

- Every zone is expected to use two keys
  - A zone-signing key (ZSK) which signs the zone data
  - A key-signing key (KSK) which signs the ZSK(s)
- Can have multiple ZSKs and/or KSKs
  - May need two or more keys when phasing in/out a key
- Parent needs to know about child zone KSKs
  - Doesn't care about child zone ZSKs

Zone Signing With DS

- Child only needs to contact parent when they have a new key signing key (KSK)
  - Parent could pick up this automatically - RFC5011 - but few do
- Signing becomes a lot simpler
  - Child signing or change of ZSK can be done at will
    - ZSK will be signed by the KSK which is "signed" by the DS record in the parent zone
  - Parent can change its keys and re-sign without informing its children
    - DS record does not change with a new parent key

Zone Enumeration

- Some TLDs were/are concerned about NSEC walking:
  - Just follow the NSEC chain to get a complete copy of the zone
    - Each NSEC record has the name of the alphabetically next domain name in the zone
    - Last NSEC record points back to start of the zone
- Showstopper for deployment in some TLDs
  - Largely privacy & copyright concerns, not technical objections

DNSSEC-ter

- IETF dnsext Working Group developed another version of the DNSSEC protocol, DNSSEC-ter
  - Defined in RFC5155
- Fixes the zone enumeration problem
  - Adds even more complexity
- Likely to mainly get used by TLDs
  - `.no` uses DNSSEC-ter
  - Does not mean delegations in `.no` need to use it
- Both DNSSEC versions are now in use and can be mixed

The NSEC3 Record - 1

- Spectacularly ugly:

```
hstring.example.com. NSEC3 alg flag iter salt \
   next rrtypes
```

hstring - hash of owner-name, record types + "stuff"

alg - hash algorithm

flags - for signalling opt-in

iter - number of iterations of hash algorithm

salt - initial salt as input to hash function

next - name of next NSEC3 record: ***hstring2*.example.com**

rrtypes - resource records that exist for ***owner-name*.example.com**

The NSEC3 Record - 2

- ***`hstring`*** is truly disgusting
  - RDATA as part of the RR's owner-name
    - => Can't use RFC5155 on zones with very long domain names
    - Maximum length of a domain name is too small for ***`hstring.verylongdomainname`***
- Computed hash string could clash with an existing name in the zone: e.g. ***`looks-like-a-hash`*****`.example.no`**
  - Hence awful kludges like applying the hash algorithm multiple times, different salts, etc.

The NSEC3 Record - 3

- Signing tools sort the NSEC3 records
  - Clients can compute ***hstring*** from the query they made and the NSEC3PARAM info but can't figure out what input (owner-name) generated ***next***
    - => Can't enumerate a DNSSEC-ter signed zone - allegedly
- Authoritative servers sometimes have to compute hashes for the names of inbound queries
  - Very Icky
  - NXDOMAIN responses get computed on the fly
  - Vector for DoS attacks: burning server CPU cycles

The NSEC3PARAM Record

- Intended as common hash parameters for a zone using NSEC3
  - Helps authoritative server generate NXDOMAIN replies
  - Validators need this to do hash computations too...

`example.com NSEC3PARAM` ***`alg flags iter salt`***

`alg` - hash algorithm

`flags` - for signalling opt-in

`iter` - number of iterations of hash algorithm

`salt` - initial salt as input to hash function

Deploying RFC5155

- Some TLDs (`.no`, `.uk`, etc.) will **only** use RFC5155
- Both DNSSEC protocols have to be supported
  - Both versions work
    - Some DNSSEC-bis adopters won't use DNSSEC-ter
      - See no need or consider DNSSEC-ter irrelevant
    - Some DNSSEC-ter adopters won't use DNSSEC-bis
      - DNSSEC-bis doesn't meet their needs
- Horrible code complexity for implementers, particularly validating resolvers

SUMMARY

The previous Section explained:

The resource records needed for Secure DNS and what they are used for

The two styles of DNSSEC

DNSSEC Software

This section gives an overview of DNSSEC validation software & solutions:

Open source software

Commericial products and services

Open Source Software

- BIND
  - Authoritative name server and validating resolver
  - Signing tools
  - Considered the reference implementation
- Unbound
  - Validating resolver server from NLnetLabs
- Patches enable Secure DNS validation for:
  - postfix, sendmail, SSH, wget, jabberd, Firefox & Thunderbird
  - See **`https://www.dnssec-tools.org`**

Windows and DNSSEC - 1

- Surprisingly good story here
- Microsoft DNS Server 2012 fully supports DNSSEC
  - Windows 2008 & Windows 7 R2 DNS servers only supported DNSSEC-bis
    - Combination of GUI and command line tools
- IPsec secures path between client and validating resolver
  - IPsec keys managed using Active Directory's security features
- Microsoft probably forced to fully support DNSSEC to win business from US Government

Windows and DNSSEC - 2

- Fairly helpful documents - well worth reading
- DNSSEC step-by-step setup guide:
  - **http://technet.microsoft.com/en-us/library/hh831411.aspx**
- Useful DNSSEC checklist/readiness document:
  - **http://www.microsoft.com/en-us/download/details.aspx?id=15204**

DNSSEC Vendors

- Bluecat Networks
- Infoblox
- Men & Mice
- Nominum
- Secure64
- Xelerance
- 2010 product survey:
  - **http://www.iis.se/docs/DNSSEC-Admin-tools-review-Final.pdf**

DNSSEC as a Service

- A few DNS providers offer DNSSEC validation
  - Google's 8.8.8.8, Dyn
- OpenDNS has a DNSCrypt product
  - Complementary to DNSSEC
  - Uses crypto to provide a "VPN" between DNS client and resolving name server
  - Analogous to IPsec channel in Windows DNSSEC offering
- Some large ISPs have enabled DNSSEC validation
  - Comcast, TeliaSonera

SUMMARY

Details of DNSSEC-aware software and services were described in this Section

Open source & commercial products

TECHNICAL CONSIDERATIONS

The engineering and operational considerations of DNSSEC deployment are discussed in this section:

Zone size
CPU load
Traffic levels, need for rate-limiting & traffic shaping
Random number generation
HSMs & Crypto Hardware
Key rotation (rollover)

Zone & Cache Size

- Signed zones are typically 5-10 times larger than unsigned ones
  - RRSIGs are big
  - More (and larger) RRs for resolving servers to cache
- So what?
- Should not be an issue for resolver operators
  - Commodity RAM is cheap: ~$30 for 4GB of DDR3
  - Disk space is cheap too: 1TB costs < $100 (Q4 2013)

CPU Load

- Signing & validating DNS data need more CPU cycles
  - No big deal on today's hardware
  - Might be an issue when handling huge volumes of validations
    - But large resolver loads are a problem without DNSSEC!
- NSEC3 means more work for name servers
  - Computing hashes for queries and responses
- Busy validating resolvers should use a multi-threaded name server (on a multi-core CPU?) if possible
  - Many threads can be validating and resolving queries simultaneously

Performance Considerations

- 1024-bit RSA/SHA-1 keys, 3GHz Xeon, 1 CPU
- Signing:
  - ~1800 signatures/second
- Validation:
  - ~24,000 validations/second
- Off-the-shelf hardware & software should be good enough
  - Hardware crypto accelerators for massive zones or very, very busy resolving servers?

Network Considerations

- DNSSEC **requires** EDNS0
  - DO bit, larger payloads, etc
  - Want to avoid truncation in both DNS response and the underlying MTU for packets/frames
- Lots of broken stuff assumes DNS only goes over UDP and always has packets < 512 bytes
  - DNSSEC kills these false assumptions
  - Firewalls sometimes get misconfigured like this
  - CPE is notorious for getting this wrong too

Traffic Levels

- Signed DNS responses are much bigger
  - Typically 4-6 times larger than unsigned responses
- => More DNS traffic
- Potential problems with fragmentation or truncation
  - Could mean an increase in TCP-based DNS traffic
  - Check firewalls, routers, etc.
- Might get more queries too
  - Validating resolvers explicitly query for RRSIGs and DNSKEYs from time to time

Rate Limiting & Traffic Shaping

- DNSSEC is a vector for DDoS amplification and reflector attacks
  - 50-60 byte query generates 1-2KB of response
- Use Response Rate Limiting (RRL)
  - Provided in BIND9.10 - patches available for BIND9.9
  - Also in recent versions of other DNS implementations
- Traffic shaping might also help
  - Routers and/or kernels

Secure Hardware

- If performance **really, really** matters, consider buying crypto accelerators for signing & verification
- Probably overkill for resolver operators
- Should be cheaper/better to just throw more commodity hardware at the problem:
  - How many multicore ~3GHz Intel boxes can be bought for the price of a decent crypto card?

Key Rollover - 1

Key Rollover - 2

- DNSSEC keys will need to be changed from time to time
  - Sensible cryptographic practice
- This should happen at regular, planned intervals
  - Say once a year
- Might have to happen sooner in an emergency
  - Current key(s) get compromised somehow or are considered tainted
- How is this best done?

Key Rollover - 3

- Principle is simple enough, doing it right isn't
- Adopt a very cautious approach:
  - Introduce the new key at least one *zone signing interval* before it is to be used
    - Zone size doubles: signed once with current key and once with the new key
  - Allow at least one *zone signing interval* with both keys in use (maybe)
  - Retire old key at least one *zone signing interval* after the new key is in production
  - Delete old key one *zone signing interval* after retirement

Key Rollover - 4

- Why be cautious?
  - Want to be sure the zone always validates
  - Avoid corner cases and validation failures where wrong keys or signatures might get cached somewhere
- Want a seamless and orderly transition from one key to the other
  - May need/want to back out the new key in a hurry
- Secure DNS signing tools may take care of this
  - Zone admins must carefully plan how to integrate these with their processes for provisioning DNS

Key Rollover: Not my Problem?

Yes and no

Validating resolvers don't need to manage key rollovers, signers do.

But if/when the signers get this wrong, validation fails.

How do detect this and deal with the problems? i.e. CEO's email can't be sent to an important customer, users can't get to their favourite web site(s)

What if **`www.skatteetaten.no`** fails to validate on tax deadline day?

SUMMARY

Technical considerations for DNSSEC deployment were described in this Section:

Disk, RAM and CPU Usage
Performance and bandwidth issues
Rate limiting/traffic shaping
Key rollover (rotation)

ZONE SIGNING

An overview of how to sign a zone using the BIND toolset is presented in this Section. Note that this is not the only way to do sign a zone, not even with the BIND tools. It just illustrates the main steps in the process.

In outline the process is:

Add two keys (ZSK & KSK) to the zone
Sign it
Load the signed zone file
Send DS record to the parent for it to sign

Step 1: Generate Keys

- Generate two keys:
  - 1024-bit RSA/SHA-1 to sign zone contents (ZSK)
  - 1024-bit RSA/SHA-1 as a key-signing key (KSK)
- For illustrative purposes
  - Choice of algorithms and key lengths is somewhat arbitrary
    - SHA-1 likely to be deprecated soon...
  - KSKs should be > 1024 bits (maybe)
  - ZSKs might not need to be as long as 1024 bits

Generating a key using **dnssec-keygen**

Generate a 1024-bit RSA key for **rfc1035.se** using an RSA/SHA-1 hash:

```
% dnssec-keygen -a RSASHA1 -b 1024 -n \
ZONE rfc1035.se
Krfc1035.se.+005+50252
```

Produces two files in current directory:

Krfc1035.se.+005+50252.key

- Public key as a DNSKEY record & some metadata

Krfc1035.se.+005+50252.private

- Private key

File Naming Convention

- **K*name*+*alg*+*tag*.{key,private}**

  name: domain name

  alg: algorithm number

  tag tag: key tag

- File naming convention helps signing tools to locate keys
- **.private** file mode 600 (obviously)
- **.key** file mode 644 (public key)

Generate the KSK

Extra argument to **`dnssec-keygen`**:

```
% dnssec-keygen -f KSK -a RSASHA1 \
  -b 1024 -n ZONE rfc1035.se
Krfc1035.se.+005+33840
```

- **`.key`** and **`.private`** files generated as before
- The **`-f KSK`** option tells **`dnssec-signzone`** to create a key-signing key
  - Flag field of DNSKEY RR will be 257, indicating a KSK

Step 2 - Signing The Zone

- Add public keys to unsigned zone file
  - Insert DNSKEY records into zone file or just **`$INCLUDE`** them:

```
$INCLUDE Krfc1035.se.+005+33840.key

$INCLUDE Krfc1035.se.+005+50252.key
```

- Run **`dnssec-signzone`**

Signing a Zone With **dnssec-signzone**

- Run **dnssec-signzone**:

```
% ZSK=Krfc1035.se.+005+50252
% KSK=Krfc1035.se.+005+33840
% dnssec-signzone -k $KSK rfc1035.se $ZSK
rfc1035.se.signed
```

Resulting **rfc1035.se.signed** file is the signed version of **rfc1035.se** zone

Original (unsigned) zone file is unchanged

Example Unsigned Zone

```
$TTL 86400
;
rfc1035.se. IN SOA ns0.rfc1035.com. hostmaster.rfc1035.com. (
   2006040100    ; serial number
   10800         ; refresh interval
   3600          ; retry
   2592000       ; expire
   86400         ; negative cache interval
   )

rfc1035.se. IN TXT  "$Id: rfc1035.se,v 1.6 2006/02/27 19:08:15 jim Exp jim $"
; Include the public keys for DNSSEC
$INCLUDE Kexample.com.+005+33840.key
$INCLUDE Kexample.com.+005+50252.key

; Here is a comment about NS records

rfc1035.se. IN      NS      ns0.rfc1035.com.

rfc1035.se. IN      NS      ns3.rfc1035.com.

; Here is a comment about A records

wallace.rfc1035.se.  IN A   195.54.233.66

gromit.rfc1035.se.   IN A   195.54.233.69

hutch.rfc1035.se.    IN A   195.54.233.70

; This is the zone's MX record for email

rfc1035.se.          IN MX  10 hutch.rfc1035.se.
```

Example Signed Zone File - I

```
; File written on Sun Apr  2 17:53:59 2006

; dnssec_signzone version 9.3.2

rfc1035.se. 86400   IN  SOA  ns0.rfc1035.com.
   ( hostmaster.rfc1035.com. 2006040100 10800 3600 2592000 86400 )

rfc1035.se. 86400   IN  RRSIG   SOA 5 2 86400 ( 20060502155359
   20060402155359 50252 rfc1035.se. xztN9UMGjQugvgOydMi+o/
   UreONZPKN00Ft3FYwwi0J7LgwbMFE9i+nyKkW7cI/
   ifmSCMNbYXj37NyoKeCbRVolAD33ZWKwCTkHNbdM+rZSz0+Zc
   +NAqQNYlPdMASG5MpdHnSYCkP82kui4aYVX2lbyy0fLyae+EalQMuI/GtSabi7/
   bUY027i7cX8NOe0ALr4Yzmv7nAFnnfla0+a6oXgd4sfkjnfD1mTIfVCXhzJYvAea
   hMSiVhk4nmB/Ys+iulsLqqRpN0HZGyPzzMxam/MJhtb+dS199sxxFw/
   oHw1Hd2e01MntateRwafe5FlUsb8d7MTjCmIUzQQqpwof1mg== )
```

Example Signed Zone File - 2

```
rfc1035.se.     86400   IN  NS   ns0.rfc1035.com.
rfc1035.se.     86400   IN  NS   ns3.rfc1035.com.

rfc1035.se.     86400   IN  RRSIG NS 5 2 86400 ( 20060502155359
   20060402155359 50252 rfc1035.se. JVGpX4CnNUYg0TlW/
   x1jMUXLKTJ8Ui/1dEi+3d+illlUOiHOiv25iDnZ7489A1o8h3TJtx/
   eWXaR45YjaZiGV9tcAY502AHW+h69BJGzxS4O4xnupA3sbGO/
   F175MBXmJJZZLBLI110eikNnahVvG99jNJl9+3MO7Y3rQMwb/jNywlN957ZEU
   +yBZgbi9Emd20RqOPfIKya+05gM9w1xy/TS/yEsrVMR3ATM05TGJcY/
   qsfhZUof14gnU3njoRvAk6KYbakC/
   DM0T5HwdfSOmuZTEIF4Ie4S6qTijnd8ryt6Lzl0Z8AB22qMd5dq1LpPX1miIaLQ
   hRCwdZXHtXnK3g== )
```

Example Signed Zone File - 3

```
rfc1035.se. 86400    IN    MX       10 hutch.rfc1035.se.

rfc1035.se. 86400    IN    RRSIG    MX 5 2 86400 ( 20060502155359
    20060402155359 50252 rfc1035.se. DnLhp8VzCuRJRBQf6w4BJm/
    Nsf3wG/qToUghzzrL44bFn4J83ZowtZUCXLeZufgugs0JYQzK/
    orEXxEOJ6Mwep2Zc9a1lJaA8lbRMbCnVIF5C5kOC2wO8YJjMhUCxL13rBnhliV
    e3buO68jRns147WAzlalT+2Mvx6kBYklXdYO1aNvJQpMrxiLaOi8Ui9l/
    8wvx5FUdOruYCPuzohSo+xL6EU52cQJBqcMGHm0xZHOshnkH7yuuVks
    +ZF5b1KEMFehTy5iQ1XKTjA9uk31/
    Hcv6cX3soJ84wKXNb2ushTJjmAlK8xGf0C/pA4dN3ZmRFk2x7psX7gZxUg
    +Q0UyCkA== )
```

Example Signed Zone File - 4

```
rfc1035.se. 86400  IN  TXT  "$Id: rfc1035.se,v 1.6 2006/02/27
   19:08:15 jim Exp jim $"

rfc1035.se. 86400  IN  RRSIG  TXT 5 2 86400 ( 20060502155359
   20060402155359 50252 rfc1035.se. ko7Vspl5h7/td96p4x2n6cm
   +jvTwhOn//B/foQ8P6gPH8TXinKmtav2nkjKUv4mHvX
   +b2tPJOEsenX6P31uEYd4EDaVfuvuufdwZ6/
   ROcVbKrh4AnaS3bd0Lf0YMttmirdQ5VRCzXiVNuzSkVagvlpW4QW2fuS9HD3fi
   SaFAroplS4veElG4Fc39ezWkD8NZPBUby0TC1y0j3MjCmDlz5cXtSShyaFfKh/
   pyt
   +nOl6ZR4T4WRYP9vOdIdzx46xZeoVUQKY8YOoWyF7tg6+dF9O6kcwjkX4n8lq0
   3bwYxvBqeX8vXK2YQCAqMA1oliYrDFJ1oBSdMsAZdmsMrl9fvkQ== )
```

Example Signed Zone File - 5

```
rfc1035.se. 86400   IN      DNSKEY  256 3 5
   ( AQPLoPeC9mnoh6xhosTwYPAdNMsJjRZNsvFAYq/aXLl
   +mwWsqFg8S95k1ikOMmgZcqjyPd2tza76osS9xb/
   uS4Ll6JlWTMv9T3UwulOeVeABtxHyUEuxLWeVX9bgeRUhBcry001Cs8hgnxGA0
   y66M18RYMT80u8KZVxLtZNJMp5ooRNQpQM60BdQCmi5jCfFfLojuzmNj/
   Y9yreELLh7pNoxSxoWw/Lu+mcmp93RaZ2Yvk/jUeTYn
   +6EyhTgh1Vw353SofkfM1To1xOdPIqicmuu
   +exqzIEKnsOeZVdJVoSGIOG2J6LWxgCsymz72Oqq5JUZjFWNtxr5LulJBa80jq
   Z5 ); key_id = 50252

rfc1035.se. 86400   IN      DNSKEY  257 3 5
   ( AQPCn7jr8I2ATUP4uujGQrJZaJo4aYOrHtBM/9S8siwioM7z
   +gHLHjNYX810lTLw4bz8EsuAgzw0iDNv4PYrobhzsjlVVTMH9LzFzo7bJnep2Z
   8mn7p6eb1316jNRYyAiMKLuf3CkS3qf4ozV2/
   AVQcFnRYR5yA2I0kVhEPlLJCwZ5GiGse1zSzco2lEyu7/
   DiAOdvFb0C2UxNuF3jRSPRQ3xVicAJ3YecMYEx6OII+zDyT/hB5PyQAMcYFHQX
   +7sCPrL/WblzsotlKJ6EMWw/g8FS1MMr9DHdCrf3t0EdNKTDxUAzU5UWFLOmw/
   lGdhLtLHRoD4v6pZ81R+EiIHUJgz ) key_id = 33840
```

Example Signed Zone File - 6

```
rfc1035.se. 86400   IN      RRSIG   DNSKEY 5 2 86400
   ( 20060502155359 20060402155359 33840 rfc1035.se.
   jDT7Uh8Q6TUZ0lIKi42kcD5GTt29PHi32eYKakopBG99xxrFV3BGkpBubqFGy0
   a8hzYm8AhSL4Ilcah7KY3yCpaLVkalWMgX0GCKwpNAiFt
   +YJ5JBTgDNkVgdy1oQUn2fwr7OLkFb3k+cDJu6AnagoRNrV3wt3B1JuPy/
   VyBYggR8EoPub+6HDFu9W/X/LJZqWpbyjUHqo5S
   +QsdsC4lIxW8CWvcjVzmJCLzv9GYgiRX6l4M0OO/sDxcbtQuAKUVdlnpTrT92/
   eIcdlPim8vvT0gntR3SNPUEESV5Wi/
   C2BNx9yZZcTf4TtAHyClqZfN19Km6ij0Nf2w8GE1eIzXSA== )

rfc1035.se. 86400   IN      RRSIG   DNSKEY 5 2 86400
   ( 20060502155359 20060402155359 50252 rfc1035.se.
   LOKgjWKRkxoRCQlwsLe10n5kYGv0pW8KvBzfQfE91iDZnCGrqYXd6qhtUtufjR
   r4MT76FLYK4l4MeyA29S+WN8K7I9OfXF/hnFZdm8O666jB4cUbj
   +y8byJCWPiErWjQpYKD4jS1zCQYpSatb2aRgoE3wnEzFftbiwk8wD9KNFnKtvr
   YibIlqXIuu/SP5VFoN7GjCnG1VMy6mpllcUQFM9qbhJ79vOTYIZnnaiJ/
   pPNyiz8y9Qxg/lIRzQnb8LuUj4ayikezJp6Le9ahNN/ECF
   +YjDgiZyWviDmi1U/BGVjzk1//XJ6ED8cXqC/E/sUWSE/
   fNgl768kIN5qteBUA7A== )
```

Example Signed Zone File - 7

```
rfc1035.se. 86400   IN      NSEC    gromit.rfc1035.se. NS SOA MX
   TXT RRSIG NSEC DNSKEY

rfc1035.se. 86400   IN      RRSIG   NSEC 5 2 86400
   ( 20060502155359 20060402155359 50252 rfc1035.se.
   vlM1A38wz3fSlLn8Jid450W8WfclL0e5lM9qR3ckNlsVOHaS4ak26dNGN3KH2o
   oJdLbklOczsOuaXddlDtggeVXknd1kKSfPIYRf8wGn6QOI6NInPtdj9Y39Lee9
   dhaIpjKyT6o/cSdTuiekQvmkiRR0SfLBZslEgkY/
   0OfPgdUi1GMfAJGXRSLnEWrU5AL+WvIbvvU5FjlpFGOw3FDi2jYXJb1u3a/
   DgNDfY4WVNo4ebKs1KiqwvyKBbiFvWEEzDOwc7hTlCMnRkxwoa0g2GAN6g4Afi
   +fkaAcW35Y5ijAYLZ6sMiSkWI0FwbKSIi9pyV9e+s5Kkq7Mqq6n5V+elg== )
```

Example Signed Zone File - 8

```
gromit.rfc1035.se. 86400 IN  A 195.54.233.69

gromit.rfc1035.se. 86400 IN  RRSIG A 5 3 86400 ( 20060502155359
20060402155359 50252 rfc1035.se.
EMKczUrdSileCSil6ZCK8uV5enJE28akCm4hQ+VYzC/wMdq2qBh0/kVkV
+fCnEmq8wPQfShSEUIlifuMovqDx3YRxQbVIGHsZqsbCK/
OzwishqzzA9TncYTD6yTwJtT4PRW+U7qtD2tgv/
dLlEDDsrFdlLMb5rxjxlMDkh7WCFsqKYZHtp4+XxuTCmPr8EvKRKKbd+IYI9l/
HJMCcPna/prM+aCW4N9lf9xIf9fzTDEJHeuENICxpELCfVdYt7Y2U/
RI6u5TqrqluiAwRFzdm3qL3glRS4cbc3TOFXGgRTHxz9/vU9xAbVToKWvbfq/
59yUcyn73hhcQv8PSL2c9Ow== )
```

Example Signed Zone File - 9

```
gromit.rfc1035.se. 86400 IN NSEC hutch.rfc1035.se.     A RRSIG
   NSEC

gromit.rfc1035.se. 86400 IN RRSIG  NSEC 5 3 86400
   ( 20060502155359 20060402155359 50252 rfc1035.se.
   TPhe7ylqAJY9hXc1H7Jl+9pmQ+/JD6fPX4/T3b8r3sCQ8rJ
   +babahmulm5WF46aEjazo7WWkTUAqxYyjTRI6IMkav3pPMs3AEg1OhHimGCV7a
   rAJLfWSeGOFyPuvXy5DM55ZCWhher8ekmML9HbP/Yw8fQ3ck
   +lL6I92vFMqALBjOXOXNW+p5jqCeUMMeQj
   +nTDQlnY29DhyZgILGcgxAYKQ7CYjdDJopSE5689kfva5Rotj9XfPOYACuoSXy
   QB9aw3Mw1c6HtVxi88jhaLmyB/
   fhOZNztP1hkXiIf7TxHREvPuQNDXAU5fXexPpNSQpVhlgGRtqW7WUyQDMQaMG/
   w== )
```

Example Signed Zone File - 10

```
hutch.rfc1035.se. 86400   IN  A    195.54.233.70

hutch.rfc1035.se. 86400   IN  RRSIG   A 5 3 86400
   (  20060502155359 20060402155359 50252 rfc1035.se.
   vRjvewsuWjIOCY0jckHjUZT5MBBLMWq0wcJsjfElS7CWre/8I7gXldh+QhH
   +xKUZgWQkYAwy8c9xxMNyoV3Bn9T2spQ2CRy06A387oS+YANIHZftbU/
   UmRyJCygrBJ7aQhh+JJDVIRlfhYeA619GMUTEvd+
   +0Wjj1vAXk73AA3pGh4sQaCZ7qiZphW1cVx7Q
   +sWear0uSGAQxb5FdtzpLxlB964ynK2oBeGnSojT3LtgEof3Te2aK20cS3FYgj
   xAi4bBCuEouS06sKDq7kB9HulfLmWwR9vUXlvqSwq8QXkYVuXj1zgHiSr0S8jM
   1VPRfpJRaXue4jhqE3T7CoXKaA== )
```

Example Signed Zone File - I I

```
hutch.rfc1035.se. 86400  IN  NSEC wallace.rfc1035.se. A RRSIG
   NSEC

hutch.rfc1035.se. 86400  IN  RRSIG  NSEC 5 3 86400
   ( 20060502155359 20060402155359 50252 rfc1035.se.
   wRlLFUKeNgdiwpLq6vv336UMQuMJ7TZmZURlqV+RLunpGfND37eM5rgKOTxEuS
   +H4RYcuAXgic9RfTAqHBIxWLOM1viHmCf9pebgqDXzlbF
   +HAQdaOOSz8KtNfOHyBMIpUN6HGjjRhyEgLC0QS
   +Pt3Mg3g9QYNjqupshdRguKL2foRtwgSq/3B6gyfagZ
   +OMtbp0lbMDBrSGhsQLo/
   3rZudbUpOyzYjIhTAAgkFk1RXyuKSWpLN9bK9jRm9/
   COX0Ouzy3abci5emWXUbSeyBUmFMCDV4nLb20nkZrGGmWZYeHIlPuRQtFbvhif
   jtliaHxmziVDm8dT0Yd5pF5ffnQg== )
```

Example Signed Zone File - 12

```
wallace.rfc1035.se. 86400 IN A     195.54.233.66

wallace.rfc1035.se. 86400 IN RRSIG  A 5 3 86400 ( 20060502155359
   20060402155359 50252 rfc1035.se. MiPh1xrudSGRHu
   +VIaKabc3UJO4W6YRrJKS9KJ+rZFNCpOz
   +YSkf4d6wRxefP5X19nqyZiRtF128Kb1MpQexjlDFLLSA7cfACgALYWGpLdb6C
   D4kRAeG+DYFREeIekNhYtMMPj
   +isVscblvLQfMtvssrRRTBvpDatkGYVgsSB1v3+yPskLvR328HRYKScZ3bZhNw
   hxKtdRrJNZ7Mj/KYnhv2weJHHLAdoSwih4+aMpk/
   vGsBeB0vGg3yI3yEMT6TQvg7SPswEZOElbLtOSy2amAhjr4UXe/
   sXOxLeKDgUPXjj2pThA6ml77u9ZI+k+8zvrwC7naIih5tW6zGQjlsXw== )
```

Example Signed Zone File - 13

```
wallace.rfc1035.se. 86400  IN  NSEC rfc1035.se.        A RRSIG
   NSEC

wallace.rfc1035.se. 86400  IN  RRSIG NSEC 5 3 86400
   ( 20060502155359 20060402155359 50252 rfc1035.se. b6HrR2dyT
   +ebAB8eBvyjwtrwFoAby74FkPkxYh5vDSo3F5lH5/
   PmPU2JDrCj6gBOi2tFFMI/xbZVAjrK52e3ndWJYt6D9OSp3cKQyvG05szXFpLK
   +zk7NMkv0PfHjJWs+vHhU5cQ/Aj3HifSQqKNiguYBmbV
   +f3ZOl2Ls7wA8InDRoQHArHiH4hJBNXY9UzzzSjS3a0If
   +KZgzuWQdcjqpi9xwcwcsuJ9WCsCJno6Nyd3/
   E25VdIsuu8fZh4q5l1+CnuyMr1wGx44s17+erYWSZ
   +Qpv8DQtrGswYDzYNhAFTz2sFiWlYOdyAsoDnrrqweTIjsW
   +uJSfzzzMofIycpA== )
```

Comments on Signed Zone - 1

- Signed zone file written in a canonical style by **`dnssec-signzone`**
- Original ordering is lost
  - So are any comments in the unsigned zone file
- Signed zone files are not human-readable
  - Should not be modified by hand or conventional DNS administration tools and wizards
  - Might not even be text files (BIND's raw format)

Comments on Signed Zone - 2

- Signed zone file has 4 times as many resource records
  - Each RRset has an RRSIG record
    - And an NSEC record which also has a RRSIG record
- Zone files may be even bigger than that
  - Signatures are often larger than the RRset they sign
- For example:

```
% wc rfc1035.se rfc1035.se.signed
  36     139    1516 rfc1035.se
 200     440    8091 rfc1035.se.signed
```

Comments on Signed Zone - 3

- The signed zone file contains the two keys
  - Zone signing key (ZSK) and key signing key (KSK)
  - DNSKEY RRset signed with both keys
- NSEC records cycle through the zone file
  - First NSEC record points to **`gromit.rfc1035.se`**
  - Next one points to **`hutch.rfc1035.se`**
  - Final one at **`wallace.rfc1035.se`** points back to the zone apex (**`rfc1035.se`**)

Step 3: Load the Signed Zone

As always, check the zone before loading it:

- `named-checkzone rfc1035.se rfc1035.se.signed`
- `dnssec-verify rfc1035.se.signed`

Excerpt from **`named.conf`** file:

```
zone rfc1035.se {
  type master;
  file "rfc1035.se.signed";
  };
```

Run **`rndc reload rfc1035.se`** and check the name server logs

Step 4: Send DS To Parent

- Public part of KSK or its DS record should be sent to DNSSEC-aware parent
  - Need a secure out-of-band transfer mechanism
  - EPP is the communication path used at Norid
- Many security and authentication issues here!
  - Child might need to identify themself to parent and/or registrar
  - Parent should ensure child zone is signed and KSK is present before accepting/publishing child's DS record(s)
- Use **`dnssec-dsfromkey`** to generate DS record(s):
  - e.g.: **`dnssec-fromkey Krfc1035.se.+005+33840`**

Excerpt from Parent Zone

`.se` zone is DNSSEC-aware and signed the DS record for **`rfc1035.se`**'s KSK, so here's what was in the **`.se`** zone

```
rfc1035.se. 86400   IN      NS      ns0.rfc1035.com.

rfc1035.se. 86400   IN      NS      ns3.rfc1035.com.

rfc1035.se. 86400   IN      DS      33840 5 1 \
   7B9802EA312D73D0EB9E499975796B598AD6165E

rfc1035.se. 86400   IN      DS      33840 5 2 \
   61D6C12705309730149 0A41C681FAF183D58AD082DCD329B7EFE21D3904D84BE

rfc1035.se. 86400   IN      RRSIG   DS 5 2 86400 ( 20060414121444
   20060409030554 29432 se.
   sgYv7eolfPkq6AynsyeP1X6c7OEX4dFDed79wRFn9mqs
   +IADankCqPlAbww2zrRWil4sZUA1KQOD2jvuVj8uXowLrr1RuIeaEO2l8jfB0ILrNl
   ESqM+IUJPf8CedbTFNWfk+YWBGOgqt3JKxVEWc0ZWzZ0nSMc4T/lEsRtk7obk= )
```

Notes on `.se` Zone Excerpts - 1

- DS RRset for **`rfc1035.se`** is signed with **`.se`** ZSK
  - Validates the DS RRset's authenticity
  - Note two DS records for different hash algorithms
    - SHA-1 (flag field is 1)
      - No longer present in **`.se`** zone
    - SHA-2 (flag field is 2)
- DS records for **`rfc1035.se`** "sign" KSK for **`rfc1035.se`**

Notes on `.se` Zone Excerpts - 2

- The NS records for **`rfc1035.se`** are not signed inside **`.se`** zone
  - These records are below the zone cut
  - Can only be signed inside the **`rfc1035.se`** zone
  - Parent zone can't be authoritative for any delegations and glue for its child zones
  - Parent **is** authoritative for child's DS records
    - Seems "wrong" to DNS protocol purists

The Chain of Trust - Again!

- Validation involves checking parent zone's DS record and its RRSIG for a DNSSEC-aware child zone's KSK
  - KSK signs the ZSK which signs the zone data
- In principle, the process iterates all the way up to the root zone
  - Root zone has a DS record for `.no`'s KSK which signs the ZSK `.no` uses for the RRSIGs over the DS records for its DNSSEC-aware child zones
    - And so on….
- Public KSK for root zone published everywhere:
  - Essentially hard-wired into validating resolver configurations

Zone Signing with NSEC3

- Subtle but important differences from "classical" zone signing
- Need extra arguments/options when invoking **`dnssec-keygen`** and **`dnssec-signzone`**
  - Need to use NSEC3-flavour keys
    - **`dnssec-keygen -3 -a NSEC3RSASHA1 ....`**
  - Generate NSEC3 RRs
    - **`dnssec-signzone -3 `*`salt`*` ....`**

An NSEC3-signed Zone - I

```
; File written on Fri Sep 25 19:17:36 2009
; dnssec_signzone version 9.7.0a3
rfc1035.se.             86400   IN SOA  ns0.rfc1035.com. hostmaster.rfc1035.com. (
                                        2009092500 ; serial
                                        10800      ; refresh (3 hours)
                                        3600       ; retry (1 hour)
                                        2592000    ; expire (4 weeks 2 days)
                                        86400      ; minimum (1 day)
                                        )
                        86400   RRSIG   SOA 7 2 86400 20091124171736 (
                                        20090925171736 26312 rfc1035.se.
                                        Kwf+ZOPB3ee2HMAmLA1KacfSLC2P9Cyb8AmsNKeQ3WFLvYysL2Ojo/n9QuxQ5vTt+1bCQ6DN
                                        Slk4d0E2hcm+Ncuww5zyrX+6qkbcDFh0NASj1qwZ8tV+rW5IOKGJb5R33zPnBEglNu6bKQuM
                                        CK9LwsOwcH5T+x0Py4fVwMdu3Z0= )
                        86400   NS      ns0.rfc1035.com.
                        86400   NS      ns3.rfc1035.com.
                        86400   RRSIG   NS 7 2 86400 20091124171736 (
                                        20090925171736 26312 rfc1035.se.
                                        UxWXwydtbTgqQUpoH1L8If5py2i/1xdVORNUHAAy93qGp8KXP4Z0PCvwgnsaxSYh7NXB77FN
                                        XfbBNjSArU92pd4wqKd8x6nRKQkdzc8PJSpff9XScXmKV+inD5//qeRAoU+xY0s6omowaToG
                                        GN/uCGiyzGPOrg0PPQecALNsaOk= )
```

An NSEC3-signed Zone - 2

```
86400   MX      10 hutch.rfc1035.se.
86400   RRSIG   MX 7 2 86400 20091124171736 (
                20090925171736 26312 rfc1035.se.
                LxEbZ0EWW+6CzHVYyjeE3T8F2aQ8ibTsdz9K4xm5PTYF/5tt0jZnIgXmEqYPxrSjW7TPq7hD
                5HtVxJHMF9OzQzXJ17uxh7iPtp36uTssoW3AUsP4XpX+1IXvpoCoQQdyygf2Rg5gCWvotT43
                ZTamEcESYd9Zh12qWdQ0XDDGmbM= )
86400   TXT     "$Id: rfc1035.se,v 1.8 2006/04/22 21:28:19 jim Exp jim $"
86400   RRSIG   TXT 7 2 86400 20091124171736 (
                20090925171736 26312 rfc1035.se.
                c4V8I/VYk06qvfQwuY4qODYNK0e5DEH1joF5m9Tg248/wCtBuluKUqHO1Da4p7jo6YccyHVH
                YgOXGl2TWrVG7Qu5/ukn16znVyhf1wet5hdo8uuiMXw+AURvzWb0PLdMXgfPNs3idpTwNhwe
                KImRBriAQ/MGI7UVhfBxGTaOBkg= )
```

An NSEC3-signed Zone - 3

```
86400   DNSKEY  256 3 7 (
                AwEAAdrTTaUWBZzC+j/EEQoLZ8X4hAKKhPEU6zTJHDSM2nQd6YwJUUA0159aqlaxBLLAkZiN
                N2yxyeFuEKhk2kS2H6GdG6WrnYEJ5QFrgNT37uFy/wS1QEEXytRhKBRmRDjXzuioVYJhff7C
                yADf9vcQ1wOWBbY50Y40VVE0IM6KD7TZ
                ) ; key id = 26312
86400   DNSKEY  257 3 7 (
                AwEAAcb8KlZiw3XNdgycwe9KqVRbBaorxWQ3dLCftMk7Vxncp7vX4yYAI/WZEMz323MpIfY9
                Cq7GV8bNT+MqJcHVUmtOJllQCiqqlV2tlOGCQ4bJfXWNpMcTsOjsf++CjucEwd+0teOL5eoW
                nWUENbEJHmM6zIWD43TdCar2AGFrZOt3
                ) ; key id = 54210
86400   RRSIG   DNSKEY 7 2 86400 20091124171736 (
                20090925171736 26312 rfc1035.se.
                JiOAmms/nv6015zKv/7QzTiH9KAoIvKMsSyhGsu/7G6bjuej/YXzaKwM06UN7rAGqlXLkiyT
                +lFlVqDd2dz+kStDPhMz/90FZoaxvkxVreucDCxOWzjyM3v6Zgif8cM3qUoBFOOsYIhlv4pC
                MX0QXFsqssEkHxu14wKi/BCP73w= )
86400   RRSIG   DNSKEY 7 2 86400 20091124171736 (
                20090925171736 54210 rfc1035.se.
                SFZOfcRkZlBu6/jAHbfnxukkxH6DvLZ/vN5qb/DP2nrVr2HWW64SKNdeb0VHrZI1BKxpRYOn
                +LS7MRXkBYV7fXIvU5gvp0lWbU5UQCvMVH6lD2aw5dHPIW0LZZdR5vV5BLqt8CAFQAuXsrJB
                OKvKBAI8+bUlFzEfjTvo+rtQw60= )
```

An NSEC3-signed Zone - 4

```
0       NSEC3PARAM 1 0 100 -

0       RRSIG   NSEC3PARAM 7 2 0 20091124171736 (

                20090925171736 26312 rfc1035.se.

                cD7l+Q+3naFrkhXVDpxVT+FgJt2++nTDCI1OJ+EurxwDmbJa+mSheyIdoxcoIFjAYnOFQPYy

                sl6rZgkhHoj6n3X2YYuQqyr/ZZwnmePYqrr6kzLEJ1qvwiJAOMMIUxZoPUhVpp8LR2sSgd25

                +y6gTwCyJr6pRx7GOTZknYS8IJs= )
```

An NSEC3-signed Zone - 5

```
gromit.rfc1035.se.      86400   IN A    195.54.233.69
                        86400   RRSIG   A 7 3 86400 20091124171736 (
                                        20090925171736 26312 rfc1035.se.
                                        XDjzqZQqW5guUYhlNLM08eqzBA7AKCwRf6GmoWBeNu6/ADZ2S4dGANd3/2lfb1mhZVapScgG
                                        WCOM/6vN5A7YdN2nQiJ6Yj1rpjrbqXuQHisdID0kuvHYLmMyntaWThs+I2pW1CLlUMRRMOOv
                                        rd6redzH7eqhGTDxKC5ARpks9zY= )
hutch.rfc1035.se.       86400   IN A    195.54.233.70
                        86400   RRSIG   A 7 3 86400 20091124171736 (
                                        20090925171736 26312 rfc1035.se.
                                        TKKeC1dTHvt/ji51TkfnlXV3R+ivVd69Tq5YpJWF29lbhpBHq73znOvz2ADzJD+25B4PNfqW
                                        IWlAgcPNU9vjXXSafpiQeApSE7WV9TQxjJWTTUD7j6IlwSXHQ4kPLCjAo91lDC0DsSP7Iu4I
                                        CZbfv4DR0dxlXxZg1gVwG0Pgk5I= )
wallace.rfc1035.se.     86400   IN A    195.54.233.66
                        86400   RRSIG   A 7 3 86400 20091124171736 (
                                        20090925171736 26312 rfc1035.se.
                                        YUJW0Pt9D6EAtuXxkxrG3Rz9KAJdSLiu5v4KENH6MeT6ufgFUV9oBmL0hQ55zpw5imT2aNFq
                                        2WI7y8/NOMNTYNjBTfKN0L4Md3fZCymQnU1P5W47CX20NZc21Au/9DQZYEMx7j627fdqLR/4
                                        qy9P9U7gTihEvCgQX/KhZwtMbjw= )
```

An NSEC3-signed Zone - 6

```
0OSUK3IK82BNIJLPOGS7G77CTD8EALN1.rfc1035.se. 86400 IN NSEC3 1 0 100 - 2QRSUKC76N2HH2T5H45C3TFEPV8HNH4F A RRSIG

                    86400   RRSIG   NSEC3 7 3 86400 20091124171736 (

                                    20090925171736 26312 rfc1035.se.

                                    H/XGR+Rcry2a6acVhIh6tF8QXhPIxyPHNpfZ

                                    F6Z1v2imwm8nNVBFqK0ahM+ke5e46kL0pKVT

                                    8rGnQo0ugvXmAVYW9bOHVZoU625XCOrZsbh0

                                    BZyOruDwaGRbDQKVbp/2SsNm0qfs4BkRcCy4

                                    p/JF7xRcEepoMWS4xgSd1xksLq0= )

2QRSUKC76N2HH2T5H45C3TFEPV8HNH4F.rfc1035.se. 86400 IN NSEC3 1 0 100 - FO8ROOFUT2LE4FO034P84CL21QPQM7HI A RRSIG

                    86400   RRSIG   NSEC3 7 3 86400 20091124171736 (

                                    20090925171736 26312 rfc1035.se.

                                    v7eOhDOF/KBSe7yJUuntzPg8wiNhy10tjFxT

                                    +5+KLMi2AhKYM+wW3k8BqTzqcXTcZFumCdLT

                                    s8GTaK9byeHMz8RUHr5ezhH4pckLkuUqyNcC

                                    jETHmvHPtw2P3RsDZ5v+CIoEk6sclenvCOjq

                                    yACahmd8Q5fifjyEh4tHC+iGY1E= )
```

An NSEC3-signed Zone - 7

```
FO8ROOFUT2LE4FO034P84CL21QPQM7HI.rfc1035.se. 86400 IN NSEC3 1 0 100 - KOSQ3UI0BNSVI6RUK3FMQOIT08T5MQG9 A RRSIG

                        86400   RRSIG   NSEC3 7 3 86400 20091124171736 (

                                        20090925171736 26312 rfc1035.se.

                                        eSPwqab179fv0lfnk1chqecKU7JxVFnivUaW

                                        CLzzxNlRuwXoK2HG++N5vx3LIAJecMJ5E1hd

                                        jUki5QA7Jk3qbYyBVaNSaujjVb7WSTmYWrGM

                                        LETAiCq69e5M7VnXiAxmRM5+qY7n8tQZQ4mc

                                        dX1rKssuNJM2RYx215ZoMowNPds= )

KOSQ3UI0BNSVI6RUK3FMQOIT08T5MQG9.rfc1035.se. 86400 IN NSEC3 1 0 100 - 0OSUK3IK82BNIJLPOGS7G77CTD8EALN1 NS SOA \

MX TXT RRSIG DNSKEY NSEC3PARAM

                        86400   RRSIG   NSEC3 7 3 86400 20091124171736 (

                                        20090925171736 26312 rfc1035.se.

                                        eB48yxjR6CFgfh/E4SaX8oIJYjscT3D5fbQc

                                        Tl/BjwMnHoYe3SaYDfw1PZ7KruKUSDLOT7j8

                                        AoW0cp7h3me78O/bEzUB3K6FhVPGRgEKFrjo

                                        Ezw6eCf2jrkQ4yheT+umonXEykuKW7/2dC0h

                                        Q27ueMzqQGwL021or8WNVw+hCQk= )
```

Notes on NSEC3-Signed Zone

- **`dnssec-signzone`** usually auto-generates NSEC3PARAM record:
  - **`-A`** and **`-H`** options
- NSEC3PARAM record & signature have 0 TTL
  - Not to be cached
- Owner-names of NSEC3 records are sorted into a linked list, similar to NSEC records

NSEC3 NXDOMAIN Responses

- Server computes hash of query-name
- i.e. **`printer.rfc1035.se`** hashes to a label somewhere between

`2QRSUKC76N2HH2T5H45C3TFEPV8HNH4F.rfc1035.se`

`FO8ROOFUT2LE4FO034P84CL21QPQM7HI.rfc1035.se`

- Server returns relevant NSEC3 RR and its RRSIG
- Client computes the hash of **`printer.rfc1035.se`** to confirm it lies between the hash values in the returned NSEC3 record

SUMMARY

This Section showed how to sign a zone using the BIND signing tools

Generating keys
Using DNSSEC-bis & DNSSEC-ter
The importance of the DS record

AUTHENTIC DATA

How can signed zones/DNS data be checked?
This Section explains what is done in the DNS protocol

The AD Header Bit

- AD (authentic data) bit in DNS header
  - Set by DNSSEC-aware server when it has validated RRSIG records
    - Client should still check RRSIGs to be 100% sure
      - What if someone sets AD bit as reply crosses the net?
- BIND9 doesn't validate RRSIGs at load time
  - AD bit not set for answers for authoritative signed zones
- **`dnssec-signzone`** can validate the signatures it generates
  - So does **`dnssec-verify`** (BIND9.10)

The “Last Mile” Issue

- AD bit is set by a validating resolver
- Is the path from some stub resolver to that server trusted?
  - Windows uses IPsec to protect this
- If the local net can’t be considered secure, run a validating resolver on the device itself
- Who standardises a Secure DNS API?
  - IETF does not do APIs
  - No apparent interest from POSIX or IEEE

A Dirty DNSSEC Secret

- If something **really** cares about DNSSEC validation, it can't rely on anything else to do that for them
  - Beware of Stupid DNS Tricks™ in hotel/coffee shop/airport type networks
    - Can't trust local network or its resolver(s)
- Maybe run a validating resolver on your laptop or tablet or smart phone?
  - Validating resolver on CPE - DSL/cable boxes?
- VPN (or equivalent) to a trusted validator is OK
  - Offered in Windows environments

Verifying with `dig -t`

```
% dig @auth-server rfc1035.se soa

; <<>> DiG 9.3.2 <<>> rfc1035.se soa

; (1 server found)

;; global options:  printcmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1509

;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2,
   ADDITIONAL: 2

;; QUESTION SECTION:

;rfc1035.se.                    IN      SOA

;; ANSWER SECTION:

rfc1035.se.             86400   IN      SOA     ns0.rfc1035.com.
   hostmaster.rfc1035.com. 2006040100 10800 3600 2592000 86400
```

Verifying with `dig` - 2

```
;; AUTHORITY SECTION:

rfc1035.se. 86400   IN      NS      ns0.rfc1035.com.

rfc1035.se. 86400   IN      NS      ns3.rfc1035.com.


;; ADDITIONAL SECTION:

ns0.rfc1035.com. 86400   IN    A       195.54.233.67

ns3.rfc1035.com. 86400   IN    A       192.71.80.122


;; Query time: 4 msec

;; SERVER: 195.54.233.69#53(195.54.233.69)

;; WHEN: Sun Apr  9 17:00:26 2006

;; MSG SIZE  rcvd: 154
```

Verifying with `dig` - 3

- DNSSEC-aware query:

```
% dig @validating-resolver rfc1035.se soa +dnssec

; <<>> DiG 9.3.2 <<>> rfc1035.se soa +dnssec

; (1 server found)

;; global options:  printcmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 493

;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 3,
   ADDITIONAL: 3

;; OPT PSEUDOSECTION:

; EDNS: version: 0, flags: do; udp: 4096

;; QUESTION SECTION:

;rfc1035.se.                            IN      SOA
```

Verifying with `dig` - 4

```
;; ANSWER SECTION:

rfc1035.se. 86400   IN      SOA     ns0.rfc1035.com.
   hostmaster.rfc1035.com. 2006040100 10800 3600 2592000
   86400

rfc1035.se. 86400   IN      RRSIG   SOA 5 2 86400
   20060502155359 20060402155359 50252 rfc1035.se.
   xztN9UMGjQugvgOydMi+o/UreONZPKN00Ft3FYwwi0J7LgwbMFE9i
   +nyKkW7cI/ifmSCMNbYXj37NyoKeCbRVolAD33ZWKwCTkHNbdM
   +rZSz0+Zc+NAqQNYlPdMASG5MpdHnSYCkP82kui4aYVX2lbyy0fLyae
   +EalQMuI/GtSabi7/
   bUY027i7cX8NOe0ALr4Yzmv7nAFnnfla0+a6oXgd4sfkjnfD1mTIfVC
   XhzJYvAeahMSiVhk4nmB/Ys+iulsLqqRpN0HZGyPzzMxam/MJhtb
   +dS199sxxFw/
   oHw1Hd2e01MntateRwafe5FlUsb8d7MTjCmIUzQQqpwof1mg==
```

Verifying with `dig` - 5

```
;; AUTHORITY SECTION:
rfc1035.se. 86400   IN      NS      ns3.rfc1035.com.
rfc1035.se. 86400   IN      NS      ns0.rfc1035.com.

rfc1035.se. 86400   IN      RRSIG   NS 5 2 86400
  20060502155359 20060402155359 50252 rfc1035.se.
  JVGpX4CnNUYg0TlW/x1jMUXLKTJ8Ui/1dEi+3d
  +illlUOiHOiv25iDnZ7489A1o8h3TJtx/
  eWXaR45YjaZiGV9tcAY502AHW+h69BJGzxS4O4xnupA3sbGO/
  F175MBXmJJZZLBLI110eikNnahVvG99jNJ19+3MO7Y3rQMwb/
  jNywlN957ZEU+yBZgbi9Emd20RqOPfIKya+05gM9w1xy/TS/
  yEsrVMR3ATM05TGJcY/qsfhZUof14gnU3njoRvAk6KYbakC/
  DM0T5HwdfSOmuZTEIF4Ie4S6qTijnd8ryt6Lzl0Z8AB22qMd5dq1LpP
  X1miIaLQhRCwdZXHtXnK3g==
```

Verifying with `dig` - 6

```
;; ADDITIONAL SECTION:
ns0.rfc1035.com. 86400   IN   A 195.54.233.67
ns3.rfc1035.com. 86400   IN   A 192.71.80.122

;; Query time: 10 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Apr  9 17:03:52 2006
;; MSG SIZE  rcvd: 761
```

DNSSEC-aware queries

- Note use of EDNS0 protocol
  - Bigger DNS payloads/buffers
  - Standard DNS query only has 512 byte payload
    - Prevents truncated responses and TCP retries
- DNSSEC-aware answer is **much** bigger
  - All the crypto stuff: RRSIGs, DNSKEYs, etc.
  - Exceeds standard 512-byte limit
    - Examples with modest key sizes and 1 or 2 keys

Getting Signed Responses - I

- Client must set the DO bit — DNSSEC OK — in EDNS0 header (RFC 3225) in outbound queries
- DO bit means "I speak DNSSEC"
  - Ensures no crypto gunk sent to legacy implementations
- DO bit gets set with **`+dnssec`** option to **`dig`**
  - Also sets a suitably large EDNS0 buffer for the reply

Getting Signed Responses - 2

- Validating BIND9 server needs compile-time support for DNSSEC:
  - `./configure --with-openssl`
- BIND9 **always** sets the DO bit on outbound queries
  - Even if DNSSEC is disabled or no crypto libraries used at compile time
    - Why ask for DNSSEC data and then just ignore it?
  - No config file option to toggle DO bit setting
    - This is unfortunate

SUMMARY

How to use check for DNSSEC responses was described in this section:

Using `dig`
The AD bit
Last-mile and packet size issues
Poor BIND behaviour

DNSSEC OBSERVATIONS

Some miscellaneous observations and advice are presented in this Section

DNSSEC Problems - I

- Bigger DNS packets
  - Typically break 512-byte payload limit
  - Need EDNS0 to allow bigger packets
    - And prevent truncated responses => TCP retries
      - Idiot firewalls don't understand EDNS0 packets
- Zone files are bigger and unreadable
- Signed zones can't be altered by hand
- Signing means changes to admin procedures:
  - *Check-out, modify, check, check-in,* ***sign*** zone
  - *Add/remove/change* keys, routine zone re-signing

DNSSEC Problems - 2

- Parent zone "signs" child's key signing keys
- Implies closer coupling between parent and child zones
  - No bad thing, but **too** many broken/lame delegations
    - ~25% in tightly controlled registries
    - ??% in `.com`
- High levels of DNS cluelessness
  - "DNS can be 90% broken and still appear to be working OK"
  - DNSSEC may well put a stop to that

DNSSEC Problems - 3

- Root zone key(s) present an obvious single point of failure
  - Compromise of the key(s) is potentially devastating
  - Will need to be changed regularly to prevent cryptanalysis
    - Will secure resolvers pick up new key in time?
  - Root zone keys likely to be a target for attack
    - How to propagate a new key after a compromise
- Who controls the key?
  - Obvious layer-9+ issues

DNSSEC Problems - 4

- DNSSEC knowledge and installed base is low
  - Uptake is improving though...
    - More TLDs are signed and the root is signed
    - Some ISPs have enabled DNSSEC validation

- Better tools and procedures are needed:
  - Zone file maintenance
  - Key management
  - Debugging & troubleshooting
  - Key rollover is tricky and “brittle”

DNSSEC Future

- Securing important Internet infrastructure is obviously a Very Good Thing
- DNSSEC could be used for distributing/publishing keys and identity tokens: SSH & PGP keys, X.509 certificates, VPN keys, etc.
  - IETF DANE working group

Potential DNSSEC Applications

- DNS as a simple PKI?
  - DNS is ubiquitous and works!
  - DNSSEC means answers can be validated
- Plug-in validators for some web browsers
  - Mainly proof-of-concept
- Validation capabilities getting added to commonly used mail systems

SUMMARY

Some of the challenges and potential uses of DNSSEC were discussed in this Section

DNSSEC READINESS

An outline of how to prepare/plan for DNSSEC deployment is explained in this Section:

Checking the network
DNSSEC-capable software
Testbed/evaluation
Documentation & training

Checking the Network

- DNSSEC will generate larger DNS responses, easily more than 1K bytes
- Some network kit (firewalls, routers, etc.) might assume DNS traffic always goes over UDP, never uses EDNS0 and never has packets > 512 bytes
  - Check vendor documentation and upgrade firmware if necessary
- Make DNSSEC-aware queries to check the underlying network path(s):
  - **`dig @i.root-servers.net . ns +dnssec`**

DNS Software

- Recent versions of BIND (9.7+) should be fine
  - Really should run the latest/current ISC release (9.9.7 or 9.10.2 - Feb 2015)
- UNIX & Linux vendors might distribute BIND that does not have DNSSEC support compiled-in
  - Get a special release from supplier?
  - Failing that, download and compile the source code yourself
    - Might not always be viable: support/audit issues

Set Up a Testbed

- Assess DNSSEC tools & software in a "safe" setting
  - Find out which ones work best/worst for you
  - Decide how to integrate them into existing infrastructure: workflows, provisioning, reporting, production systems
  - Determine how DNSSEC will be used in your environment and plan its roll-out
  - Find out what breaks and how for most likely errors like signature expiry, mistakes with keys, DS records/KSK inconsistencies, validation failures, etc.
  - Document lessons learned

Documentation

- Make sure everything gets properly documented
  - Design/architecture
  - Deployment plans & roadmap
  - Familiarise support and technical staff with DNSSEC tools and new processes
  - Describe how to recognise DNSSEC validation failures and the appropriate ways to deal with them
  - White papers and use cases for customers?

Training and Outreach

- How will your staff get trained?
  - Knowledge transfer from in-house experts
  - Commercial training courses, webinars, etc.
- Not just for your DNS team
  - Network operations & system administrators
  - Development & helpdesk
  - Customer relations & support staff
- Talk to customers, suppliers and business partners

SUMMARY

This Section briefly described how to prepare for DNSSEC deployment.

CONFIGURING DNSSEC VALIDATION

Here's how to enable DNSSEC validation for commonly used-implementations:

BIND
unbound
Microsoft

In Outline…

- Tell the validating DNS server which trust anchor(s) to use
  - Should only need to be the root zone's KSK
    - Fetch this from the IANA web site or the DNS itself

      `http://data.iana.org/root-anchors`

    - Might need other trust anchors: reverse zones for RFC1918 address space, internal name spaces on corporate net, etc.
- Add info about trust anchor(s) to name server's configuration

Configuring BIND

```
options {
  ...
 dnssec-enable yes;
 dnssec-validation auto;
 managed-keys-directory "managed-keys";
};

managed-keys {
          . initial-key 257 3 8 "AwEAAag....";
};
```

That's it!

More on BIND Setup

- **`managed-keys{}`** statement tells the validator which trust anchor(s) to use and what keys to trust
  - Add local one(s) to taste
  - BIND will automatically handle key rollovers for these
    - Name server updates **`managed-keys.bind`** in **`managed-keys-directory`**
      - Tricky file permission considerations: file should be "secure", but **`named`** shouldn't run as the super-user
- **`AwEEAag....`** is base-64 encoding of current (May 2015) KSK for the root key

`managed-keys{}` and `trusted-keys{}`

- **`managed-keys{}`** statement
  - Used for zones doing RFC5011-style key rollover
  - For one-time checking when server starts
  - Used to validate whatever's in **`managed-keys.bind`**
- **`trusted-keys{}`** statement
  - Has to be updated by hand when keys are changed
  - Avoid: makes things brittle

BIND's Trust Anchors

- BIND's approach is rather clumsy
- Need to cut & paste DNSKEY records into **`named.conf`**:

```
managed-keys {
   . initial-key 257 3 8 "AwEAA...."
   /* Key for intranet's example.com */
   example.com. initial-key 257 3 5 "BNY4w...."

trusted-keys {
   /* Key for intranet's example.com */
   example.com. 257 3 5 "BNY4w...."
```

Validation with **unbound**

- A no-brainer - just run **`unbound-anchor`**

```
unbound-anchor -a /etc/unbound/run/root-ta
```

- Fetches current root KSK (and checks it)
- Sets up **`unbound`** to automatically deal with rollover of the root key
- Tell unbound to use the created trust anchor

Excerpt from **`unbound.conf`**:

```
auto-trust-anchor-file: "/etc/unbound/run/root-ta"
```

unbound with Other Trust Anchors

- Put DS or DNSKEY record(s) per trust anchor in a **`trust-anchor-file`**

In **`unbound.conf`**:

```
trust-anchor-file: "/etc/unbound/10.in-addr.arpa"
```

In **`10.in-addr.arpa`**:

```
10.in-addr.arpa. IN DS 58534 5 1 EF....
```

DNSSEC Validation for Windows 2012 Server

- Two options:
  - Command line with **`dnscmd.exe`**
  - DNS Manager GUI
- Not yet fully supported in Powershell
  - Can use this to introduce local trust anchors for private domains such as RFC1918 reverse zones

dnscmd.exe

- It's possible to simply fire and forget:

```
dnscmd.exe /RetrieveRootTrustAnchors
```

  - Fetches root trust anchor and enables validation

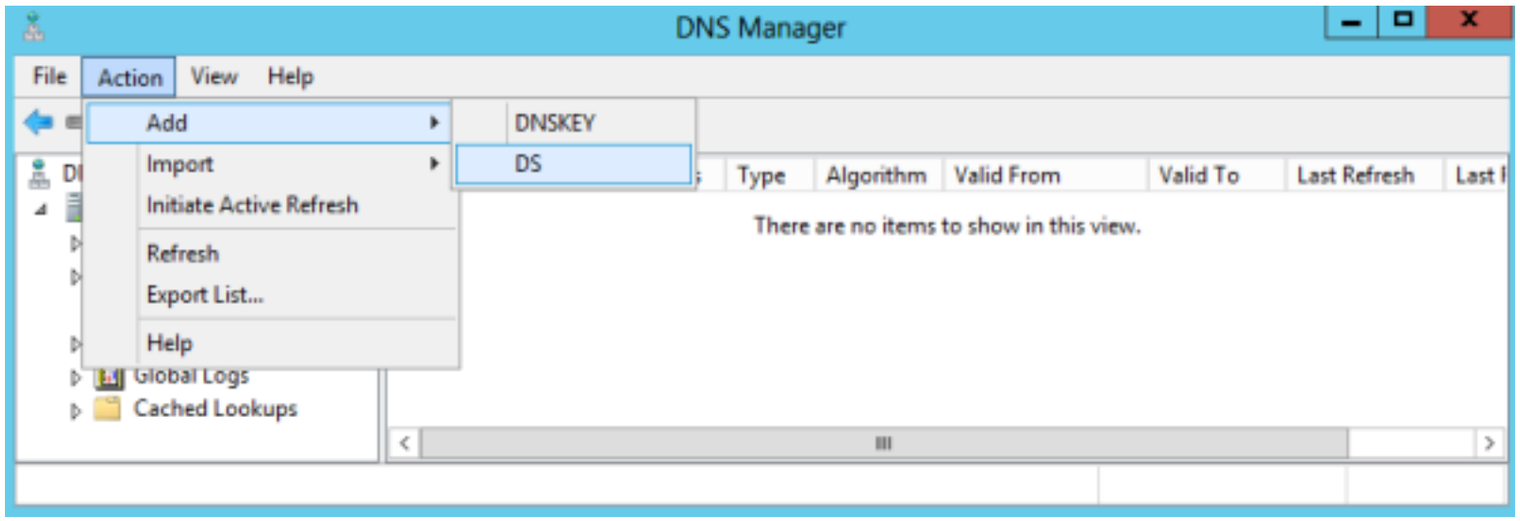
- To enable or disable DNSSEC validation seperately:

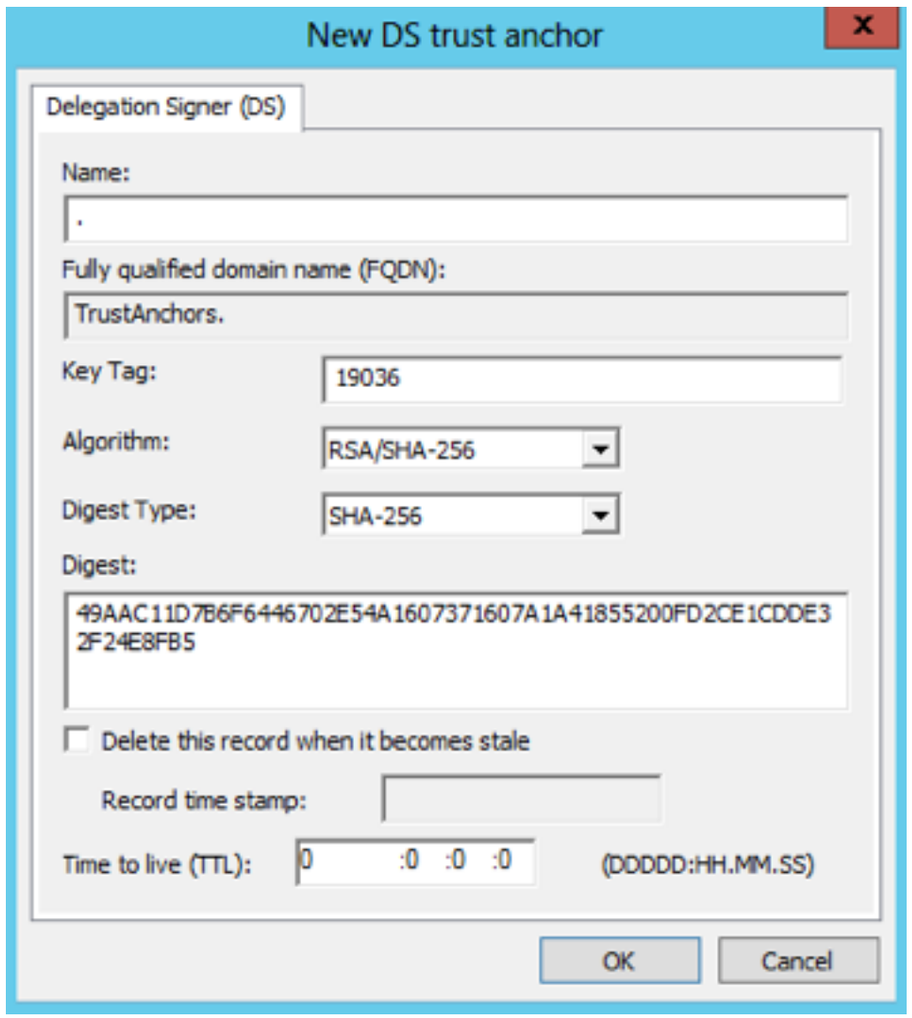
```
dnscmd.exe /EnableDNSSEC [0|1]
```

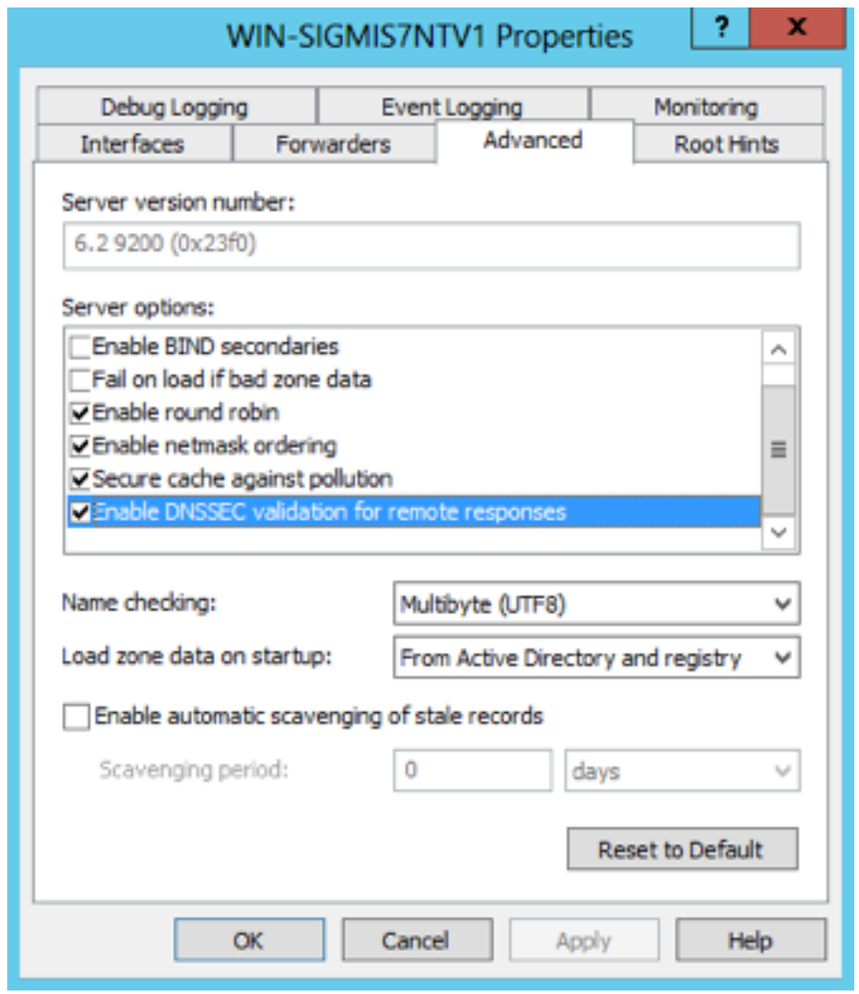
- To add other trust anchors:

```
dnscmd.exe /TrustAnchorAdd \
10.in-addr.arpa DS 58534 5 1 EF…
```

DNS Manager


DNS Manager
File
Action
View
Help
Add
Import
Initiate Active Refresh
Refresh
Export List...
Help
DNSKEY
DS
Type
Algorithm
Valid From
Valid To
Last Refresh
There are no items to show in this view.
Global Logs
Cached Lookups


New DS trust anchor
Delegation Signer (DS)
Name:
.
Fully qualified domain name (FQDN):
TrustAnchors.
Key Tag:
19036
Algorithm:
RSA/SHA-256
Digest Type:
SHA-256
Digest:
49AAC11D7B6F6446702E54A1607371607A1A41855200FD2CE1CDDE32F24E8FB5
Delete this record when it becomes stale
Record time stamp:
Time to live (TTL):
0 :0 :0 :0
(DDDDD:HH.MM.SS)
OK
Cancel


WIN-SIGMIS7NTV1 Properties
Debug Logging
Event Logging
Monitoring
Interfaces
Forwarders
Advanced
Root Hints
Server version number:
6.2 9200 (0x23f0)
Server options:
Enable BIND secondaries
Fail on load if bad zone data
Enable round robin
Enable netmask ordering
Secure cache against pollution
Enable DNSSEC validation for remote responses
Name checking:
Multibyte (UTF8)
Load zone data on startup:
From Active Directory and registry
Enable automatic scavenging of stale records
Scavenging period:
0
days
Reset to Default
OK
Cancel
Apply
Help

A Word about DLV

DON'T, JUST DON'T

OK, that's 3 words…

DLV is going away

Testing & Verification

- Try visiting these web sites:
  - **`dnssec-or-not.org`**
  - **`dnssectest.sidn.nl`**
- If successful, you're done!
- If not, the validating server is probably misconfigured
  - Inspect logs, turn up debugging
  - Use **`dig`** to check DNS responses
    - Do they set the AD bit and have RRSIGs?
  - Use **`drill`** or **`delv`** for further troubleshooting

Validator Crunch Time!

- Major test is just around the corner
  - First ever rollover of root zone's KSK due Real Soon Now
- Validators which don't support/use RFC5011 key rollover will almost certainly break:
  - BIND configurations with **`trusted-keys{}`** statements
  - Old/clunky/buggy DNSSEC implementations
- IANA's giving plenty of advance warning
- Nothing important should fail
  - Famous last words….

Negative Trust Anchors

- How to tell the validator that the key(s) for some domain(s) are broken
  - Don't validate these domains for now, but carry on validating elsewhere
    - Known insecure data will (perhaps) be better than nothing
- Proposal under consideration by the IETF
- Features in BIND and **`unbound`** to support this
  - A necessary evil - unfortunately
  - Workarounds for other people's mistakes

Kludging around Validation Failures in BIND

```
options {
  ...
  dnssec-accept-expired yes;
  dnssec-must-be-secure example.com no;
    ..
   };
```

dnssec-accept-expired

- Return answers despite them having out of date signatures

dnssec-must-be-secure

- Return answers for **example.com** even if they are known not to validate

Kludging around Validation Failures in **unbound**

- **`val-permissive-mode: yes`**
  - Tells **unbound** to return known insecure data instead of a DNSSEC validation **`SERVFAIL`** error
- **`domain-insecure: "example.com"`**
  - Don't validate **`example.com`** names - presumably because its zone admin has messed up:
    - Forgot to re-sign the zone or tell the parent about a new KSK or didn't roll keys correctly

SUMMARY

This section showed how to configure DNSSEC Validation for the common DNS implementations:

BIND
unbound
Microsoft Windows 2012 Server

The importance of automatic key rollover support was explained: new root KSK is due soon
Workarounds for validation failures were also covered

DNSSEC Tools

An overview of the current DNSSEC tools will be presented:

`dig`
`drill`
`delv`
DNSVIZ

DNSSEC Troubleshooting

- Can be **very** painful
- Debugging "vanilla" DNS problems is hard enough
- DNSSEC makes things even harder
  - Checking DNSKEY, RRSIG, NSEC3 and DS records
  - Keeping track of tags, algorithm numbers, etc.
- Tools are getting better, but still too hard for many DNS administrators
  - Lack obvious error messages: "you forgot to re-sign", "that signature has the wrong hash value", "there's no DS record for this KSK", etc

BIND DNSSEC Logging

Use the **`crypto logging{}`** channel

```
logging {
    ....
    channel crypto {
            file "querylogs/dnssec"
            versions 30 size 5M;
            severity debug;
            print-time yes;
    };
```

Give the server a kick to turn up debugging:

`rndc trace N`

Troubleshooting with `dig`

- Hidden compile-time option
  - `-DDIG_SIGCHASE`
- Makes `dig` do DNSSEC validation
  - Can start at an arbitrary trust anchor
  - Top-down
- Awkward to use and hard to understand output
- Not recommended: better tools than that

Troubleshooting with `drill`

- The best DNSSEC debugging tool **by far**
  - Also replicates **`dig`**'s commonly used functionality
  - Open source from NLnetLabs
- Can illustrate validation in action
  - Shows which keys (algorithms & key lengths) are used
  - Work on a single signature or top-down from the root
- Pinpoint stale keys & signatures
- Identify DS record and KSK mismatches
- **`drill -TD`** ***`some-domain`*** is just awesome!

Using `drill`

```
% drill -S www.norid.no CNAME
;; Number of trusted keys: 1
;; Chasing: www.norid.no. CNAME


DNSSEC Trust tree:
www.norid.no. (CNAME)
|---norid.no. (DNSKEY keytag: 409 alg: 8 flags: 256)
    |---norid.no. (DNSKEY keytag: 62984 alg: 8 flags: 257)
    |---norid.no. (DS keytag: 62984 digest type: 2)
        |---no. (DNSKEY keytag: 490 alg: 8 flags: 256)
            |---no. (DNSKEY keytag: 29471 alg: 8 flags: 257)
            |---no. (DS keytag: 29471 digest type: 2)
                |---. (DNSKEY keytag: 48613 alg: 8 flags: 256)
                    |---. (DNSKEY keytag: 19036 alg: 8 flags: 257)
;; Chase successful
```

Troubleshooting with **`delv`**

- ISC's answer to **`drill`**
  - And **`dig`**'s **`-DDIG_SIGCHASE`**
- Distributed in BIND9.10
- Command-line options almost identical to **`dig`**
- Not as chatty as **`delv`**
- Use the **`+vtrace`** option to see the validations

`delv` Examples

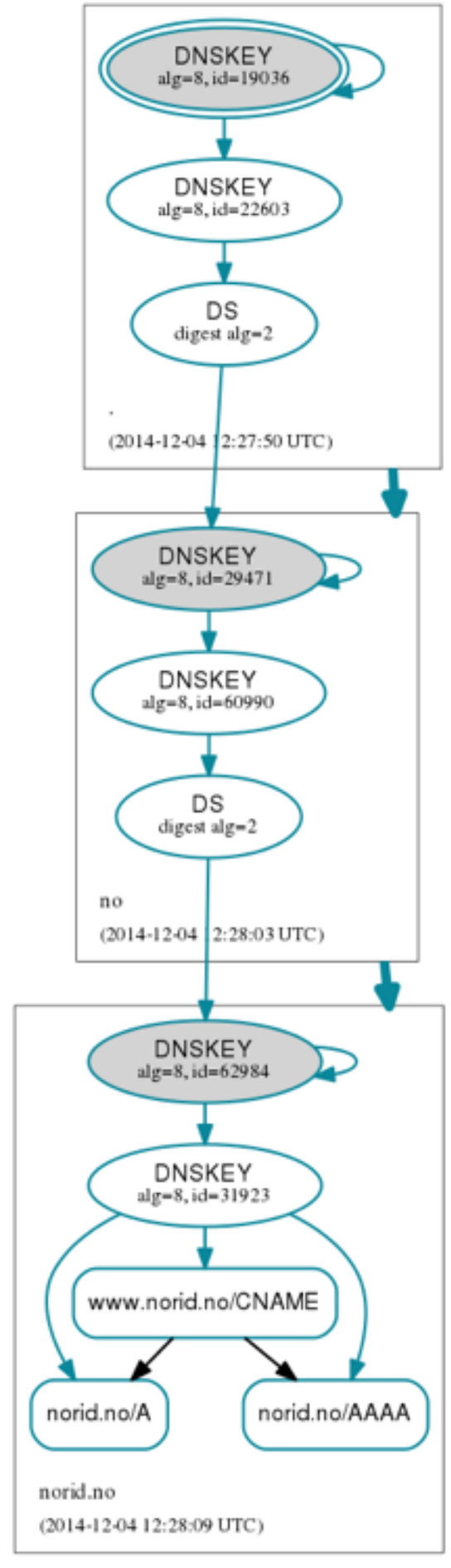
```
% delv norid.no mx
; fully validated
norid.no.    3253   IN MX 10 ismtp.uninett.no.
norid.no.    3253   IN MX 20 ipv4.ismtp.uninett.no.
norid.no.    3253   IN RRSIG MX 8 2 3600 \
   20150612061155 20150522074406 409 norid.no. YKCT....
% delv www.norid.no dname
;; resolution failed: ncache nxrrset
; fully validated
www.norid.no.       86320  IN CNAME norid.no.
www.norid.no.       86320  IN RRSIG CNAME 8 3 86400 \
   20150608070639 20150518061818 409 norid.no. MxwQ....
; norid.no.     208 IN \-DNAME  ;-$NXRRSET
; norid.no. SOA ns.uninett.no. hostmaster.uninett.no. \
   2015052403 14400 3600 1814400 900
; norid.no. RRSIG SOA ...
; 1ksplso7uobn6o2605rgdo14k3tuebf5.norid.no. RRSIG NSEC3 ...
; 1ksplso7uobn6o2605rgdo14k3tuebf5.norid.no. NSEC3 1 0 5 \
   C2D6F7B336F6CD98 1SOMPVTIVER0H2A6TDIUMEO4M60IFFSA \
   A NS SOA MX TXT AAAA RRSIG DNSKEY NSEC3PARAM SPF
```

DNSVIZ

- A web-based DNS visualisation tool
  - Draws nice pictures
  - Can display details of revoked keys
- Visit **`dnsviz.net`** and type in your favourite signed domain name
  - Uses the web site's validating resolver, not yours
  - Doesn't check your validator's configuration
  - Can't check internal-only signed domains
- Could fetch DNSVIZ source code and run it locally

DNSVIZ Example

- Hover over elements to see details of the key, algorithm, TTL, key tags, etc
- Shaded elements are the KSKs
- Double circle around the trust anchor: the root's KSK



DS/DNSKEY Mismatches

- Not easily detected by hand: get **`delv`** or **`drill`** or DNSVIZ to do the hard work instead
- Could extract DNSKEY records from **`dig`** output, edit this, feed to **`dnssec-dsfromkey`** and compare to DS record(s) in the parent
  - Only for masochists
  - Some DNS geeks have **`awk`** or **`perl`** scripts to do this

QUESTIONS?