DNSSEC Deployment: A Tutorial

Phil Regnauld
Hervey Allen

February 2009
Manila, Philippines

http://nsrc.org/tutorials/2009/apricot/dnssec/

Overview

- We will be talking about DNSSEC
- We plan to do a live zone signing demonstration and we will have instructions and tools available so that you may follow along if you have your own laptop with SSH (download Putty if using Windows)
- If you notice anything that may be incorrect, let us know right away. This topic is still fairly dynamic.

Contents

- Scope of the problem
- DNS reminders
- Basics of DNSSEC
- Live demonstration
- Operations
- Issues (what isn't solved) & other aspects
- Status of DNSSEC today

What's the Problem?

Up until recently, DNSSEC looked like a problem looking for a solution

- Thankfully the Kaminsky flaw solved this.

What's the problem?

So what are the issues?

DNS Cache Poisoning

- Forgery: respond before the intended nameserver
- Redirection of a domain's nameserver
- Redirection of NS records to another target domain

DNS Hijacking

- Response to non-existent domains
- Rogue DNS servers

These have been spotted in the wild – code IS available...

What's the problem?

What risks ?

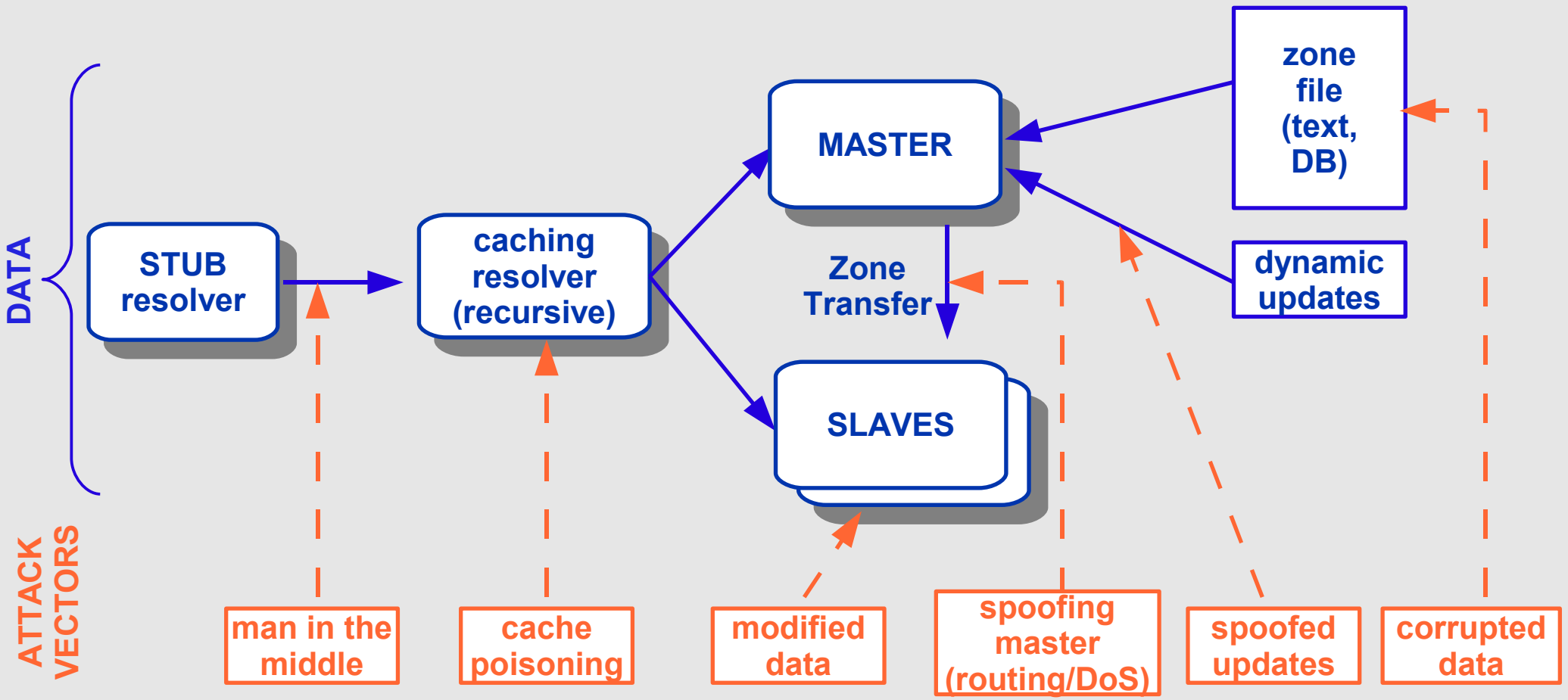
- See Dan Kaminsky's slides for the extent of the risks
  - MANY case scenarios
  - Scary stuff:
    - MX hijacking
    - Entire domain redirection
    - Take a large .COM offline
    - Complete spoofing of a bank's DNS info
    - ...

DNSSEC Quick Summary

- Data authenticity and integrity by signing the Resource Records Sets with private key
- Public DNSKEYs published, used to verify the RRSIGs
- Children sign their zones with their private key
  - Authenticity of that key established by signature/checksum by the parent (DS)
- Repeat for parent...
- Not that difficult on paper
  - Operationally, it is much more complicated

DNS points of attack

DNS Data Flow
Points of attack



Refresher

DNS reminders

- ISC BIND zone file format is commonly used, and we will use this notation here.

```
zone.       SOA  ( 2009022401  ; serial
                   1d           ; refresh
                   12h          ; retry
                   1w           ; expire
                   1h )         ; neg. TTL

zone.       NS   ns.zone.
            NS   ns.otherzone.

zone.       MX   5 server.otherzone.
www.zone.    A   1.2.3.4
```

DNS reminders

- Record structure:

```
NAME          [TTL]  TYPE  DATA (type specific)
-------------------------------------------------
host.zone.     3600  A     10.20.30.40
sub.zone.     86400  MX    5 server.otherzone.
```

DNS reminders

- Multiple resource records with *same name and type* are grouped into Resource Record Sets (RRsets):

```
mail.zone.      MX    5 server1.zone.
mail.zone.      MX    10 server2.zone.

server1.zone.   A     10.20.30.40
server1.zone.   A     10.20.30.41
server1.zone.   A     10.20.30.42

server1.zone.   AAAA  2001:123:456::1
server1.zone.   AAAA  2001:123:456::2

server2.zone.   A     11.22.33.44
```

RRset

RRset

RRset

RRset

DNSSEC concepts

DNSSEC overview

DNS SECurity extensions

- Concepts
- New Resource Records (DNSKEY, RRSIG, NSEC/NSEC3 and NS)
- New packet options (CD, AD, DO)
- Setting up a Secure Zone
- Delegating Signing Authority
- Key Rollovers

DNSSEC concepts

- Changes DNS trust model from one of ”open” and ”trusting” to one of ”verifiable”
- Extensive use of public key cryptography to provide:
  - Authentication of origin
  - Data integrity
  - Authenticated denial of existence
- No attempt to provide confidentiality
- DNSSEC does not place computational load on the authoritative servers (!= those *signing* the zone)
- No modifications to the core protocol
  - Can coexist with today's infrastructure
    - ... kind of (EDNS0)

DNSSEC concepts

- Build a chain of trust using the existing delegation-based model of distribution that is the DNS
- Don't sign the entire zone, sign a RRset

- Note: the parent DOES NOT sign the child zone.
  - The parent signs a pointer (hash) to the key used to sign the data of child zone (important!)

New Resource Records

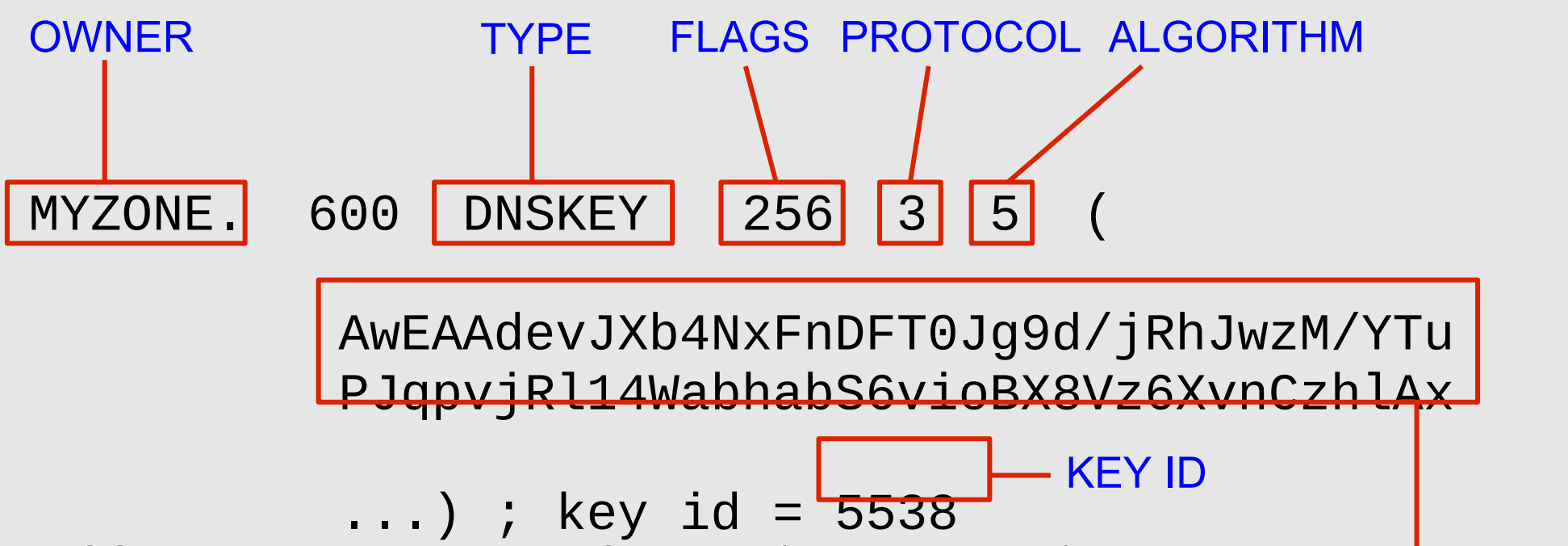
DNSSEC: new RRs

Adds four new DNS Resource Records*:

1. **DNSKEY**: Public key used in zone signing operations.
2. **RRSIG**: RRset signature
3. **NSEC/NSEC3**: Returned as verifiable evidence that the name and/or RR type does not exist
4. **DS**: Delegation Signer. Contains the hash of the public key used to sign the key which itself will be used to sign the zone data. Follow DS RR's until a ”trusted” zone is reached (ideally the root).

*See Geof Huston's excellent discussion at http://ispcolumn.isoc.org/2006-08/dnssec.html

DNSSEC: DNSKEY RR



- FLAGS determines the usage of the key (more on this...)
- PROTOCOL is always 3 in the current version of DNSSEC
- ALGORITHM can be:
  - 0 – reserved
  - 1 – RSA/MD5 (deprecated)
  - 2 – Diffie/Hellman
  - 3 – DSA/SHA-1 (optional)
  - 4 – reserved
  - 5 – RSA/SHA-1 (mandatory)

DNSSEC: DNSKEY RR

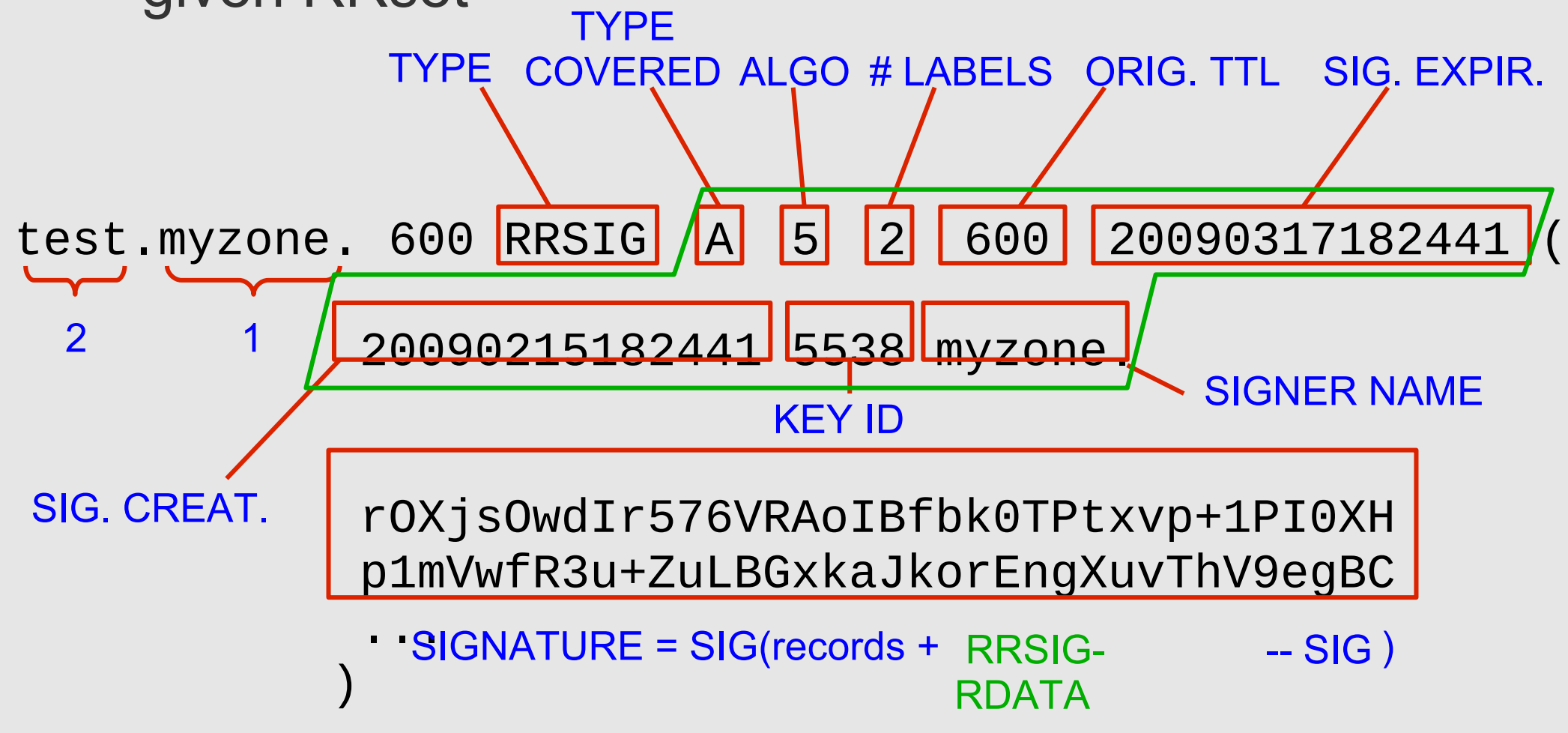
- There are in practice at least two DNSKEYs for every zone:
  - Originally, one key-pair (public, private) defined for the zone:
    - private key used to sign the zone data (RRsets)
    - public key published (DNSKEY) in zone
    - DS record (DNSKEY hash) published in parent zone, and signed in turn with rest of data
- Problem:
  - to update this key, DS record in parent zone needs to be updated...
    - Introduction of Key Signing Key (flags = 257)

DNSSEC: KSK and ZSK

- To allow for key updates (“rollovers”), generate two keys:
  - Key Signing Key (KSK)
    - pointed to by parent zone (Secure Entry Point), in the form of DS (Delegation Signer)
    - used to sign the Zone Signing Key (ZSK)
  - Zone Signing Key (ZSK)
    - signed by the Key Signing Key
    - used to sign the zone data RRsets
- This decoupling allows for independent updating of the ZSK without having to update the KSK, and involve the parent.

DNSSEC: RRSIG

- Resource Record Signature
  - lists the signatures performed using the ZSK on a given RRset



DNSSEC: RRSIG

- By default:
  - Signature creation time is *1 hour before*
  - Signature expiration is *30 days from now*
  - Needless to say, proper timekeeping (NTP) is strongly recommended
- What happens when the signatures run out ?
  - SERVFAIL...
  - Your domain effectively disappears from the Internet
  - ... more on this later
- Note that the *keys* do not expire.
- Therefore, *regular* re-signing is part of the operations process (not only when changes occur)
  - the entire zone doesn't have to be resigned...

DNSSEC: NSEC/NSEC3

- NSEC – proof of non-existence
- Remember, the authoritative servers are serving precalculated records. No on-the-fly generation is done.
  - NSEC provides a pointer to the **N**ext **SEC**ure record in the chain of records.
    - "there are no other records between this one and the next", signed.
  - The entire zone is sorted lexicographically:

```
myzone.
sub.myzone.
test.myzone.
```

DNSSEC: NSEC/NSEC3

```
myzone. 10800 NSEC test.myzone. NS SOA RRSIG NSEC DNSKEY

myzone. 10800 RRSIG NSEC 5 1 10800 20090317182441 (
                    20090215182441 5538 myzone.

                    ZTYDLeUDMlpsp+IWV8gcUVRkIr7KmkVS5TPH
                    KPsxgXCnjnd8qk+ddXlrQerUeho4RTq8CpKV
                      ...
                    )
```

- Last NSEC record points back to first.
- Problem:
  - Zone enumeration (walk list of NSEC records)
  - Yes, DNS shouldn't be used to store sensitive information, but future uses may require this “feature”

DNSSEC: NSEC/NSEC3

- If the server responds NXDOMAIN:
  - One or more NSEC RRs indicate that the name (or a wildcard expansion) does not exist
- If the server's response is NOERROR:
  - And the answer section is empty
  - The NSEC proves that the TYPE did not exist

DNSSEC: NSEC/NSEC3

- What about NSEC3 ?
  - We won't get into this here, but the short story is:
    - Don't sign the name of the Next SECure record, but a ***hash*** of it
      - Still possible to prove non-existence, without revealing name.
    - This is a simplified explanation. RFC 5155 covering NSEC3 is *53* pages long.
  - Also introduces the concept of “opt-out” (see section 6 of the RFC) which has uses for so-called delegation-centric zones with unsigned delegations.

DNSSEC: DS

- Delegation Signer
- Hash of the KSK of the child zone
- Stored in the parent zone, together with the NS RRs indicating a delegation of the child zone
- The DS record for the child zone is signed *together* with the rest of the parent zone data
  NS records are *NOT* signed (they are a hint)

```
myzone. DS 61138 5 1
 F6CD025B3F5D030408950535 4A0115584B56D683
myzone. DS 61138 5 2
 CCBC0B557510E4256E88C01B0B1336AC4ED6FE08C826
 8CC1AA5FBF00 5DCE3210
```

DNSSEC: DS

- Two hashes generated by default:
  - 1 SHA-1 MANDATORY
  - 2 SHA-256 MANDATORY

DNSSEC: new fields

- Updates DNS protocol at the packet level
- Non-compliant DNS recursive servers *should* ignore these:
  - **CD**: Checking Disabled (ask recursing server to not perform validation, even if DNSSEC signatures are available and verifiable, i.e.: a Secure Entry Point can be found)
  - **AD**: Authenticated Data, set on the answer by the validating server if the answer could be validated, and the client requested validation
- A new EDNS0 option
  - **DO**: DNSSEC OK (EDNS0 OPT header) to indicate client support for DNSSEC options

Live demo using dig

Security Status of Data
(RFC4035)

- Secure
  - Resolver is able to build a chain of signed DNSKEY and DS RRs from a trusted security anchor to the RRset
- Insecure
  - Resolver knows that it has no chain of signed DNSKEY and DS RRs from any trusted starting point to the RRset
- Bogus
  - Resolver believes that it ought to be able to establish a chain of trust but for which it is unable to do so
  - May indicate an attack but may also indicate a configuration error or some form of data corruption
- Indeterminate
  - Resolver is not able to determine whether the RRset should be signed

Signing a zone...

Enabling DNSSEC

- **Multiple systems involved**
  - Stub resolvers
    - Nothing to be done... but more on that later
  - Caching resolvers (recursive)
    - Enable DNSSEC validation
  - Authoritative servers
    - Enable DNSSEC logic (if required)
      - Signing & serving need not be performed on same machine
      - Signing system can be offline

Signing the zone

1. Generate keypair
2. Include public DNSKEYs in zone file
3. Sign the zone using the secret keys
4. Publishing the zone
5. Push DS record up to your parent
6. Wait...

1. Generating the keys

```
# Generate ZSK
dnssec-keygen -a rsasha1 -b 1024 -n ZONE  myzone

# Generate KSK
dnssec-keygen -a rsasha1 -b 2048 -n ZONE -f KSK myzone
```

This generates 4 files:

- Kmyzone.+005+*id_of_zsk*.key
- Kmyzone.+005+*id_of_zsk*.private
- Kmyzone.+005+*id_of_ksk*.key
- Kmyzone.+005+*id_of_ksk*.private

2. Including the keys into the zone

Include the DNSKEY records for the ZSK and KSK into the zone, to be signed with the rest of the data:

```
cat Kmyzone*key >>myzone
```

or add to the end of the zone file:

```
$INCLUDE “Kmyzone.+005+id_of_zsk.key”
$INCLUDE “Kmyzone.+005+id_of_ksk.key”
```

3. Signing the zone

Sign your zone

```
# dnssec-signzone myzone
```

- dnssec-signzone will be run with all defaults for signature duration, the serial will not be incremented by default, and the private keys to use for signing will be automatically determined.
- Signing will:
  - Sort the zone (lexicographically)
  - Insert:
    - NSEC records
    - RRSIG records (signature of each RRset)
    - DS records from child keyset files (for parent)
  - Generate key-set and DS-set files, to be communicated to the parent

4. Publishing the signed zone

- Publish signed zone by reconfiguring the nameserver to load the signed zonefile.
- ... but you still need to communicate the DS RRset in a secure fashion to your parent, otherwise no one will know you use DNSSEC

5. Pushing DS record to parent

Securely communicate the KSK derived DS record set to the parent

- ... but what if your parent *isn't* DNSSEC-enabled ?
  - manually distributing your public keys is too complicated
  - could there be an easier mechanism Until The Root Is Signed™ ?

Enabling DNSSEC in the resolver

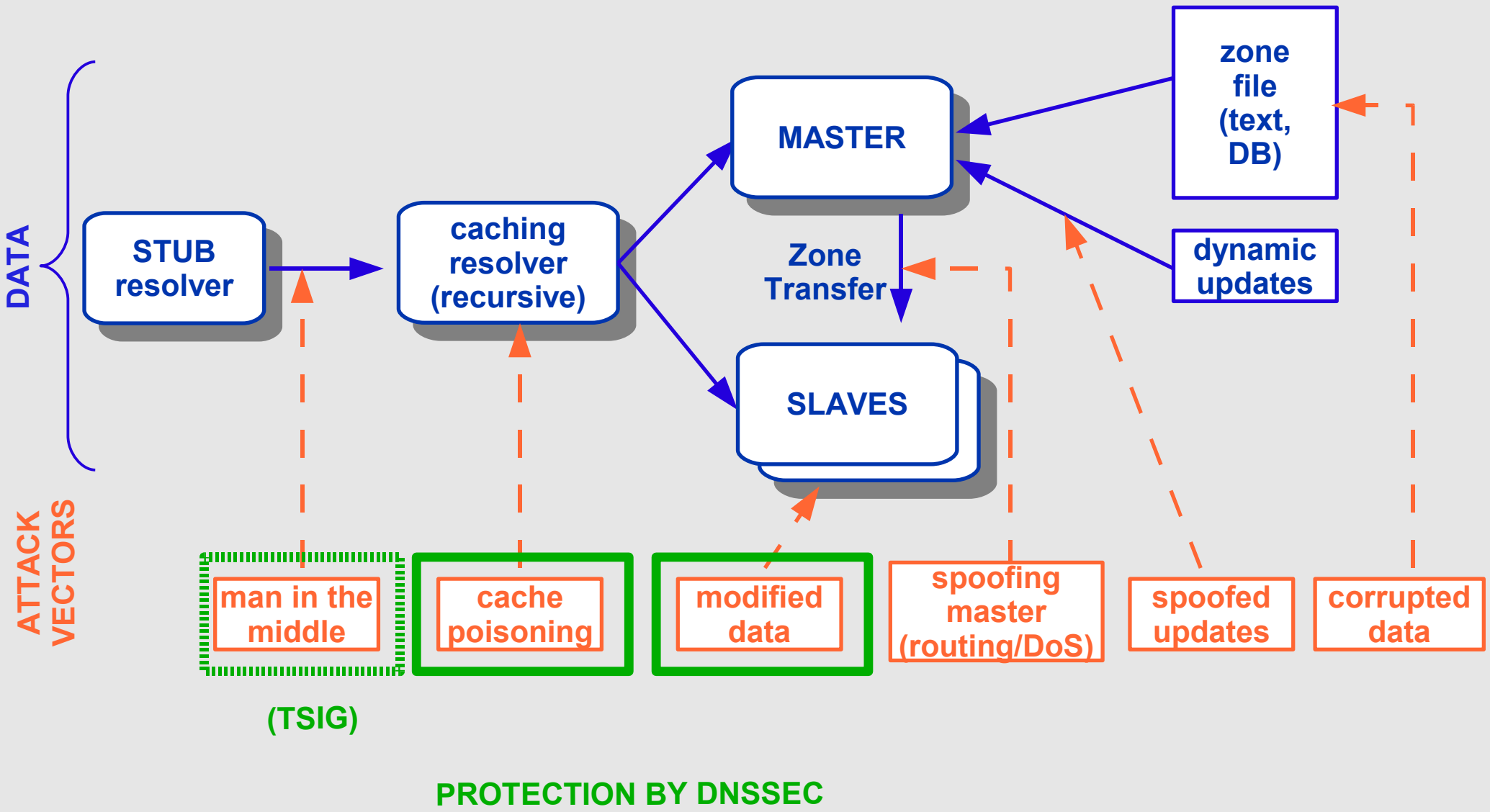
- Configure forwarding resolver to validate DNSSEC
  - not strictly necessary, but useful if only to verify that your zone works
- Test...
- Remember, validation is only done in the resolver.

Summary

- Generating keys
- Signing and publishing the zone
- Resolver configuration
- Testing the secure zone

Questions so far ?

So, what does DNSSEC protect ?



What doesn't it protect ?

- Confidentiality
  - The data is not encrypted
- Communication between the stub resolver (i.e: your OS/desktop) and the caching resolver.
  - For this, you will need TSIG, or you will have to trust your resolver 100%
  - ... it performs all validation on your behalf

So why isn't it implemented ?

Many different reasons...

- It's ”complicated”. Requires more work. Tools will help with this. Operational experience is the keyword.
- Risks of failure (failure to sign, failure to update) what will result in your zone disappearing
- Specification has changed several times since the 90s
- NSEC Allow(ed|s) for zone enumeration.
- Until Kaminsky, maybe not obvious enough why we needed DNSSEC.
- *The root (.) is not yet signed* - it's political...

Delegating Signing Authority

Using the DNS to Distribute Keys

- Secured islands make key distribution problematic
- Distributing keys through DNS:
  - Use one trusted key to establish authenticity of other keys
  - Building chains of trust from the root down
  - Parents need to sign the keys of their children
- Only the root key needed in ideal world
  - Parents always delegate security to child
  - ... but it doesn't help to sign if your parent doesn't sign, or isn't signed itself...

Walking the Chain of Trust (thank you RIPE :)

Locally Configured

```
Trusted Key . 8907
```

(root) .

```
.                   DNSKEY (…) 5TQ3s… (8907) ; KSK
                    DNSKEY (…) lasE5… (2983) ; ZSK

                    RRSIG  DNSKEY (…)  8907 .  69Hw9…

net.                DS   7834 3 1ab15…
                    RRSIG   DS (…) . 2983
```

net.

```
net.                DNSKEY (…) q3dEw… (7834) ; KSK
                    DNSKEY (…) 5TQ3s… (5612) ; ZSK

                    RRSIG  DNSKEY (…)  7834 net.  cMas…

apricot.net.        DS   4252 3 1ab15…
                    RRSIG  DS (…) net. 5612
```

apricot.net.

```
apricot.net.        DNSKEY (…) rwx002…  (4252) ; KSK
                    DNSKEY (…) sovP42…  (1111) ; ZSK

                    RRSIG  DNSKEY (…) 4252 apricot.net.  5t...

www.apricot.net.    A 202.12.29.5
                    RRSIG  A  (…)  1111 apricot.net.  a3...
```

Ok, but what do we do Until The Root Is Signed[tm] ?

- Use of Trust Anchors
  - *A DNS resource record store that contains SEP keys for one or more zones.*
- Two initiatives exist to provide these Trust Anchor Repositories.
  - for TLDs
  - for other domains
- Note: this is our interpretation of the current situation, and does not necessarily reflect the position of the parties involved.

Trust Anchor Repositories… DLV and ITAR

DLV: DNSSEC Lookaside Validation

- Alternative method for chain of trust creation and verification in a disjointed signed space (islands of trust)
- DLV functions automatically (if the resolver is configured to do so) by looking up in a preconfigured "lookaside validation" zone
  - no need to fetch a list of anchors
  - ISC Initiative: https://www.isc.org/solutions/dlv

Trust Anchor Repositories… DLV and ITAR

ITAR: Interim Trust Anchor Repositories

- Interim Trust Anchor Repository
- IANA Trust Anchor Repository (Until The Root Is Signed)
  - Is targeted at TLDs
  - Lookup is not automatic
    - list of anchors must be retrieved (one more operational constraint)
  - Already a beta program, several TLDs have already registered
  - https://itar.iana.org/

Trust Anchor Repositories… DLV and ITAR

- See the summary and discussions here:
  - "Using DNSSEC today" http://www.links.org/?p=542
  - "DNSSEC with DLV" http://www.links.org/?p=562
- … the conclusion seems to be that DLV and ITAR complement each other

Operational Aspects

Signature expiration

- Signatures are per default 30 days (BIND)
- Need for regular resigning
  - To maintain a constant window of validity for the signatures of the existing RRset
  - To sign new and updated RRsets
- Who does this ?
- The keys themselves do NOT expire...
  - But they do need to be rolled over...

Key Rollovers

- Try to minimise impact
  - Short validity of signatures
  - Regular key rollover
- Remember: DNSKEYs do not have timestamps
  - the RRSIG over the DNSKEY has the timestamp
- Key rollover involves second party or parties:
  - State to be maintained during rollover
  - Operationally expensive

Key Rollovers

- Two methods for doing key rollover
  - pre-publish
  - double signature
- KSK and ZSK rollover use different methods (courtesy DNSSEC-Tools.org)

Key Rollovers

- **ZSK Rollover Using the Pre-Publish Method**

1. wait for old zone data to expire from caches (TTL)
2. sign the zone with the KSK and published ZSK
3. wait for old zone data to expire from caches
4. adjust keys in key list and sign the zone with new ZSK

Key Rollovers

- **KSK Rollover Using the Double Signature Method**

1. wait for old zone data to expire from caches
2. generate a new (published) KSK
3. wait for the old DNSKEY RRset to expire from caches
4. roll the KSKs
5. transfer new DS keyset to the parent
6. wait for parent to publish the new DS record
7. reload the zone

Signing the Root

- The current state of things is viewable here:
  - http://www.ntia.doc.gov/DNS/dnssec.html

Deployment hurdles and other issues

Lack of operational experience...

Everyone talks about DNSSEC

- ... but few people have real hands-on experience with day-to-day operations
- One can't just turn DNSSEC on and off
  - stopping to sign a zone isn't enough
  - parent needs to stop publishing DS record + signatures
- Failure modes are fairly well known, but recovery procedures cumbersome and need automated help

DS publication mechanisms

No established procedure exists for communicating DS records to the parent

- SSL upload ?
- PGP/GPG signed mail ?
- EPP extension ?

- Remember, this should happen *automatically* and reliably

EDNS0 and broken firewalls, DNS servers

DNSSEC implies EDNS0

- Larger DNS packets means > 512 bytes
- EDNS0 not always recognized/allowed by firewall
- TCP filtering, overzealous administrators..

- Most hotels (including this one) do not allow DNSSEC records through

Application awareness

This could be a long term pain...

- Application's knowledge of DNSSEC ... is non-existent
  - Users cannot see why things failed
  - Push support questions back to network staff
    - Compare with SSL failures (for users who can read...)

- There are APIs – currently 2
  - http://tools.ietf.org/id/draft-hayatnagarkar-dnsext-validator-api-07.txt
  - http://www.unbound.net/documentation/index.html

    - Firefox plugin example (pullup from DNS layer to user)
    - What if applications explicitly set +CD ?

Corporate environments

- Split DNS anyone ?
  - How do we deal with:

```
www.corp.net.    A    130.221.140.4 ; public
```

and

```
www.corp.net.    A    10.2.4.6      ; private
```

- “Oh but you shouldn't do that, that's Not The Right Way!”
  - ... like NAT ?
  - ... and NSEC enumeration ?