DNSSEC

Introduction
Principles
Deployment

NSRC
Network Startup Resource Center

Overview

What we will cover

- The problems that DNSSEC addresses
- The protocol and implementations
- Things to take into account to deploy DNSSEC
- The practical problems tied to real-world deployment

Contents

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

Scope of the problem

So what are the issues?

DNS Cache Poisoning

- Inject forged data into the cache by either:

  a) returning additional (forged) data outside the scope of the origingal query

  b) responding to the caching server with forged data before the authoritative server's answer is received

  - First issue fixed 20 years ago
  - Second issue theoretically very difficult
    - until Dan Kaminsky in 2008

Scope of the problem

What risks ?

- Misdirection of queries for an entire domain
- Response to non-existent domains
- MX hijacking
- Make a large domain (SLD or TLD) domain “disappear” from an ISP's cache – DoS
- Identity theft using SSL stripping attacks (banks, eGovernance)
- More fun stuff...

These have been spotted in the wild, and code IS available... See Dan Kaminsky's slides for a more details & scenarios

- A great illustrated guide
  http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

Refresher

DNS reminders

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

```
zone. SOA nsX.zone. hostmaster.zone.
                  ( 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.
```

RRset

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

RRset

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

RRset

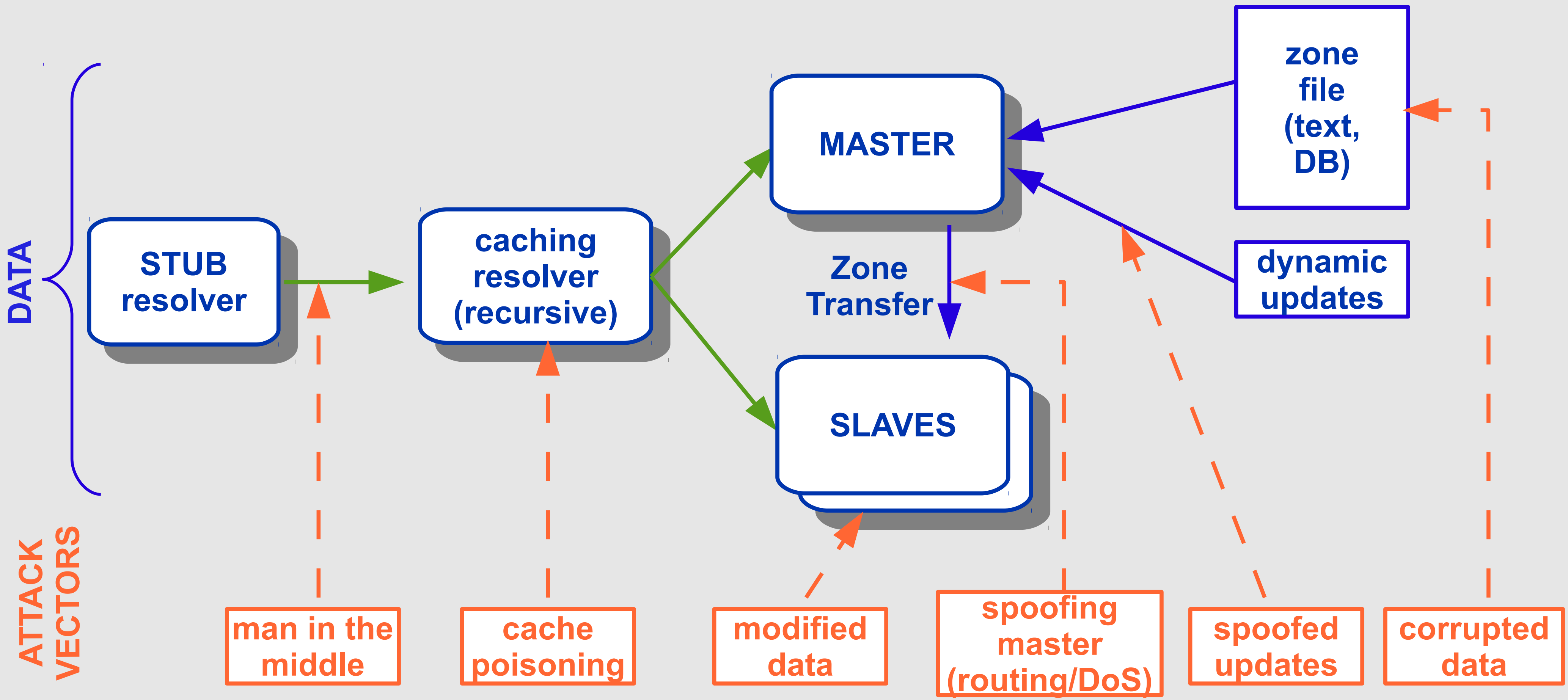
```
server2.zone.   A    11.22.33.44
```

RRset

DNS points of attack

DNS Data Flow

Points of attack



DNSSEC concepts

DNSSEC in a nutshell

- Data authenticity and integrity by signing the Resource Records Sets with a private key
- Public DNSKEYs published, used to verify the RRSIGs
- Children sign their zones with their private key
  - Authenticity of that key established by *parent* signing hash (DS) of the *child* zone's key
- Repeat for parent...
- Not that difficult on paper
  - Operationally, it is a bit more complicated

DS_{KEY} KEY –signs zone data

Concepts

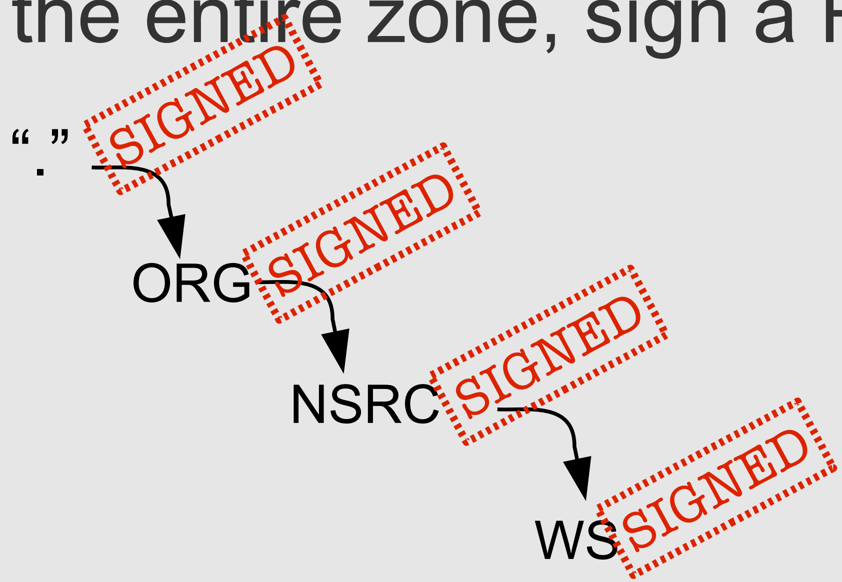
- New Resource Records (DNSKEY, RRSIG, NSEC/NSEC3 and DS)
- New packet options (CD, AD, DO)
- Setting up a Secure Zone
- Delegating Signing Authority
- Key Rollovers

DNSSEC concepts

- Changes DNS trust model from one of ”open” and ”trusting” to one of ”verifiable”
- Use of public key cryptography to provide:
  - Authentication of origin
  - Data integrity
  - Authenticated denial of existence
- No attempt to provide confidentiality (NO encryption)
- DNSSEC does not normally place computational load on the authoritative servers (!= those *signing* the zone)
- No modifications to the core protocol
  - Can coexist with today's infrastructure (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 <u>DOES NOT</u> sign the child zone. The parent signs a *pointer* (hash) to the *key* used to sign the data of *child* zone (DS record)

New Resource Records

DNSSEC: new RRs

Adds five new DNS Resource Records*:

1. **DNSKEY**: Public key used in zone signing operations.
2. **RRSIG**: RRset signature
3. **NSEC &**
4. **NSEC3**: Returned as verifiable evidence that the name and/or RR type does not exist
5. **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 Geoff Huston's discussion at http://ispcolumn.isoc.org/2006-08/dnssec.html

DNSSEC: DNSKEY RR

OWNER TYPE FLAGS PROTOCOL ALGORITHM

```
MYZONE.   600  DNSKEY   256  3  5  (
        AwEAAdevJXb4NxFnDFT0Jg9d/jRhJwzM/YTu
        PJqpvjRl14WabhabS6vioBX8Vz6XvnCzhlAx
        ...) ; key id = 5538
```

KEY ID

PUBLIC KEY (BASE64)

- FLAGS determines the usage of the key (more on this...)
- PROTOCOL is always 3 (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 in validator)
  - 8 – RSA/SHA-256

http://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xml

DNSSEC: Two keys, not one...

- There are in practice at least two DNSKEY pairs 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
- DNSSEC works fine with a single key pair...
- Problem with using a single key:
  - Every time the key is updated the, DS record corresponding to the key must be updated in the parent zone as well
    - ➔Introduction of Key Signing Key (flags = 257)

DNSSEC: KSK and ZSK

- Key Signing Key (KSK)
  - pointed to by parent zone in the form of DS (Delegation Signer). Also called Secure Entry Point
  - 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 – less administrative interaction.

DS_{KSK} KSK –signs ZSK –signs RRsets

DNSSEC: Resource Record SIGnature
RRset signed using ZSK

```
test.myzone. 600 A  1.2.3.4
test.myzone. 600 A  2.3.4.5
```

2 1

TYPE COVERED

TYPE ALGO # LABELS ORIG. TTL SIG. EXPIR.

```
test.myzone. 600 RRSIG A 5 2 600 20090317182441 (
  20090215182441 5538 myzone.
  rOXjsOwdIr576VRAoIBfbk0TPtxvp+1PI0XH
  p1mVwfR3u+ZuLBGxkaJkorEngXuvThV9egBC
  ...
)
```

SIGNER NAME

KEY ID

SIG. INCEP.

SIGNATURE = SIG(RRset + RRSIG-DATA – SIG)

DNSSEC: RRSIG

- Typical default values (not a standard, but BP):
  - Signature inception time is *1 hour before*
  - Signature expiration is *30 days from now*
  - Proper timekeeping (NTP) is required
- What happens when the signatures run out ?
  - SERVFAIL...
  - Your domain effectively disappears from the Internet for validating resolvers
- Note that the *keys* do ***not*** expire.
- Therefore, *regular* re-signing is part of the operations process (not only when changes occur)
- Not all RRsets need be resigned at the same time

DNSSEC: NSEC/NSEC3

- Proof of non-existence using NSEC & NSEC3
- Remember, the authoritative servers are serving precalculated records. No on-the-fly generatio
  - 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: illustrate

```
myzone.          NS      ...
ace.myzone.      A       ...
bob.myzone.      CNAME   ...
cat.myzone.      A       ...
eel.myzone.      MX      ...
```

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 the first.
- Problem:
  - Zone enumeration (walk list of NSEC records)
  - Public DNS shouldn't be used to store sensitive information
    - ➔ But policy requirements vary.

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 details here:
    - 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 long!
  - Also introduces the concept of “opt-out” (see section 6 of the RFC) for delegation-centric zones
  - Don't bother signing RRsets for delegations which you know don't implement DNSSEC.

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/pointer)

```
                                 Digest type 1 = SHA-1, 2 = SHA-256
myzone. DS 61138 5 1
 F6CD025B3F5D0304089505354A0115584B56D683
myzone. DS 61138 5 2
 CCBC0B557510E4256E88C01B0B1336AC4ED6FE08C826
 8CC1AA5FBF00 5DCE3210
```

```
digest = hash( canonical FQDN on KEY RR | KEY_RR_rdata)
```

DNSSEC: DS

- Two hashes generated by default:
  - 1 SHA-1 Mandatory support for validator
  - 2 SHA-256 Mandatory support for validator
- New algorithms are being standardised upon
- This will happen continually as algorithms are broken/proven to be unsafe

DNSSEC: new fields/flags

- 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

Demo: the new records

Security Status of Data
(RFC4033 § 5 & 4035 § 4.3)

- 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
  - No trust anchor to indicate if the zone and children should be secure. 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
    - ➔ Configure trust anchors manually (or DLV)
  - Authoritative servers
    - ➔ Enable DNSSEC code (if required)
      - Signing & serving need not be performed on same machine
      - Signing system can be offline

Signing the zone (using the BIND tools)

1. Generate keypairs
2. Include public DNSKEYs in zone file
3. Sign the zone using the secret key ZSK
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 (NSEC is default)
    - RRSIG records (signature of each RRset)
    - DS records from child keyset files (for parent: -g option)
  - Generate key-set and DS-set files, to be communicated to the parent

3. Signing the zone (2)

- ISC BIND

- Since version 9.7.0, automated zone signing
  - Makes life much easier
  - Key generation, management & rollover still needs to be done separately

- Version 9.8.0 introduces inline signing
  - Easier integration in existing chain of production

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

- Need to securely communicate the KSK derived DS record set to the parent
  - RFCs 4310, 5011
- ... but what if your parent *isn't* DNSSEC-enabled ?
  - DLV

Enabling DNSSEC in the resolver

- Configure forwarding resolver to validate DNSSEC
- Test...
- Remember, validation is only done in the resolver
- Others need to enable DNSSEC validation – it doesn't help if you are the only one doing it!

Summary

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

Questions so far ?

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
  - Use of *jitter* to avoid having to resign all expiring RRsets at the same time
- The keys themselves do NOT expire...
  - But they may 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
- RFC5011 + BIND support
- See http://www.potaroo.net/ispcol/2010-02/rollover.html

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**

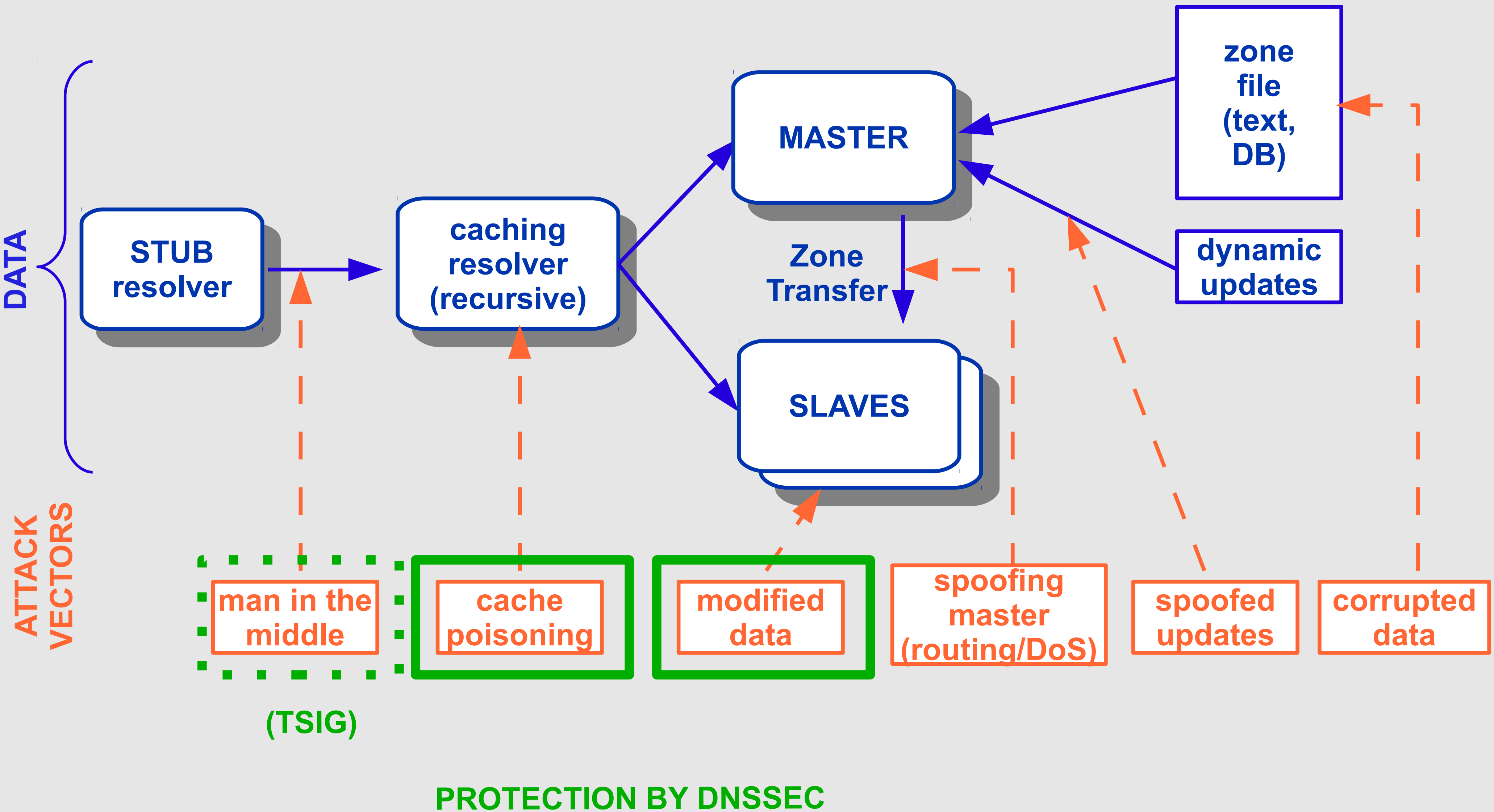
1. wait for old zone data to expire from caches
2. generate a new (published) KSK
3. wait for the old DNSKEY RRset to expire from caches
4. roll the KSKs
5. transfer new DS keyset to the parent
6. wait for parent to publish the new DS record
7. reload the zone

It is also possible to use dual DS in the parent zone

Automated toolkits

- Luckily, a number of toolkits already exist to make DNSSEC operations as smooth as possible
- Doesn't solve all problems yet, such as interaction with parent and children (DS management, …), but take care of all the rough edges of running a PKI (yes, that's what it is...)
- http://www.dnssec.net/software
  - www.opendnssec.org
  - www.dnssec-tools.org
  - http://www.hznet.de/dns/zkt/
  - …

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 would have to use TSIG, SIG(0), or you will have to trust your resolver
  - It performs all validation on your behalf
- Still need to do validation yourself if you don't trust your upstream's nameservers

Validating the chain of trust

Why the long timeframe ?

Many different reasons...

- Lack of best practice. Ops experience scarce
- Risks of failure (failure to sign, failure to update) which will result in your zone disappearing
- Specification has changed several times
  - ➔NSEC allows for zone enumeration
- Until 2008, DNSSEC “a solution w/o problem”
- Delay in getting the root signed (politics)
- Increased fragility – resolution less tolerant to brokenness!
- Failed validation penalizes client, not owner

Walking the Chain of Trust (slide courtesy RIPE)

Locally Configured

```
Trusted Key . 8907
```

(root) .

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

                   RRSIG  DNSKEY (…)  8907 .  69Hw9…

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

org.

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

                   RRSIG  DNSKEY (…)  7834 org.  cMas…

 nsrc.org.         DS   4252 3 1ab15…
                   RRSIG  DS (…) org. 5612
```

nsrc.org.

```
 nsrc.org.         DNSKEY (…) rwx002…  (4252) ; KSK
                   DNSKEY (…) sovP42…  (1111) ; ZSK

                   RRSIG  DNSKEY (…) 4252 nsrc.org.  5t...

 www.nsrc.org.     A 202.12.29.5
                   RRSIG  A  (…)  1111 nsrc.org.  a3...
```

DNSSEC Deployment & Operations

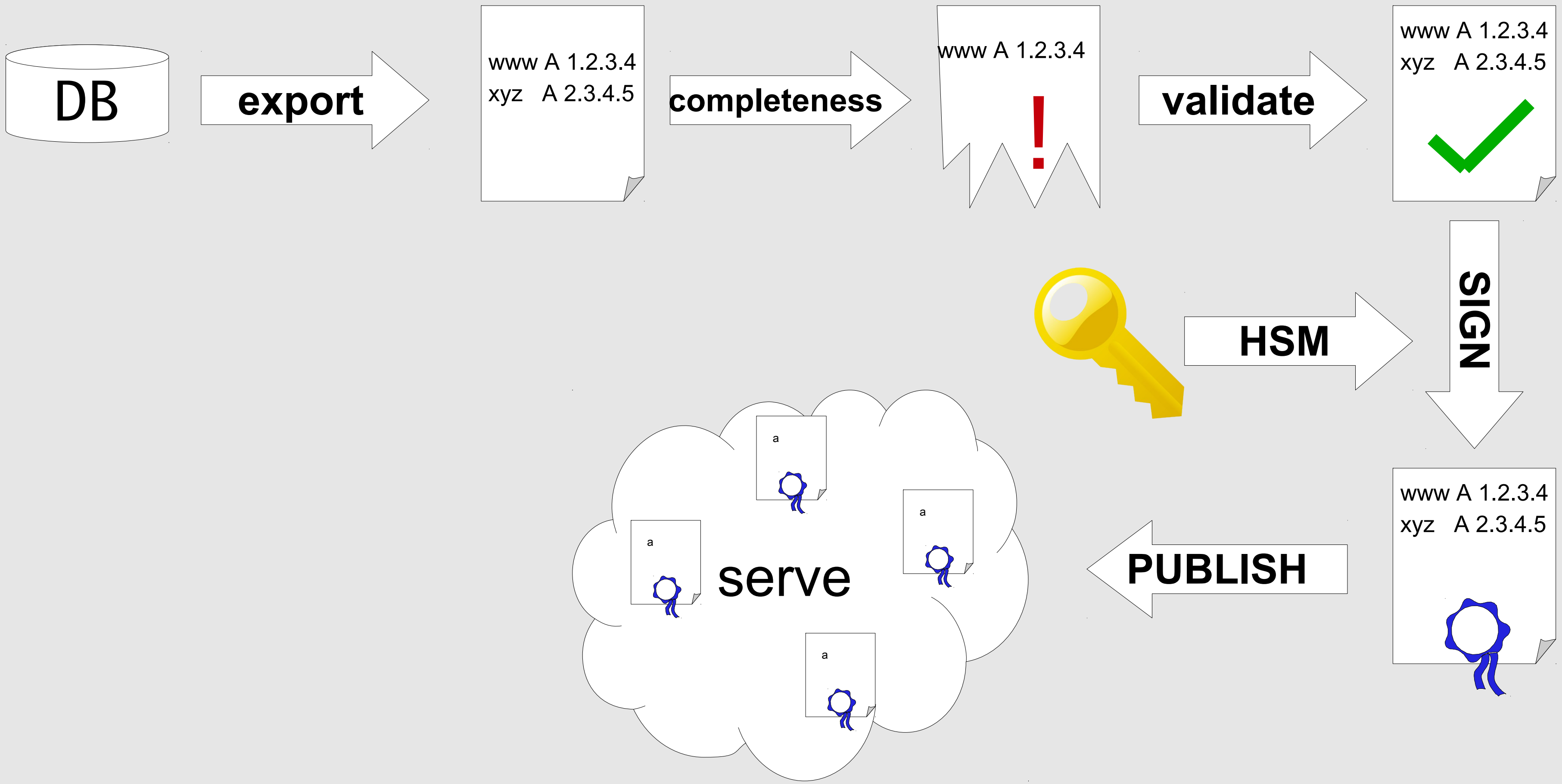
Deploying DNSSEC the boring bits

- A DPS (DNSSEC Policy & Practice Statement)
  http://tools.ietf.org/html/draft-ietf-dnsop-dnssec-dps-framework-03
  - ___
  - Details the design, implementation, methods and practices governing the operation of a DNSSEC signed zone
  - Helps external parties review/scrutinize the process and evaluate the trustworthiness of the system.

- Existing operational framework in which to insert the DNSSEC process
  - much larger chance of shooting one self in foot if the organisation doesn't have proper operational procedures in the first place.

What does it take to deploy DNSSEC ? (2)

- Monitoring



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
  - no longer signing the zone isn't enough
  - parent needs to stop publishing DS record + signatures
- Failure modes are fairly well known, but recovery procedures cumbersome and need manual intervention

DS publication mechanisms

Standardized way to communicate DS to parent, but not widely deployed, or different method used

- SSL upload ?
- PGP/GPG signed mail ?
- EPP extension (RFC4310)

- Remember, this should happen securely
- Redelegation or change of registrant when the zone is signed
  - Share the key during the transition ?
  - Turn off DNSSEC for the time ?
  - What if the original administrator is not cooperative ?
    - → Policy issues

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...

- Many hotel network infrastructures (maybe this one as well) do not allow DNSSEC records through, or interfere with DNS resolution
  - Captive portals, redirections

Application awareness

- Applications don't know about DNSSEC, mostly
  - 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, Chrome support
  - ➔ What if applications explicitly set +CD ?

Securing the last link

- Stub resolvers remain open to man in the middle attacks
  - Not many ways around this
  - Either trust your resolver, use TSIG or validate yourself
- Work is being done to address these issues
  - DNS over other transport protocols to work around excessive filtering
  - dnssec-trigger project (http://www.nlnetlabs.nl/projects/dnssec-trigger/)

OPCODE=0

?