DNSSEC Training Workshop

Why DNSSEC?

- **DNS is not secure**
  - Known vulnerabilities
  - People depend more and more on DNS

- **DNSSEC protects against data spoofing and corruption**

- **Why this course:**
  - To raise awareness on DNSSEC
  - To provide handles for deployment to meet Federal mandates

Why Worry About DNS Security?

- **Forged DNS data breaks most applications:**
  - Web site can be replaced with a false site without ever touching the victim site
  - EMail can be re-routed or mis-delivered
  - Login compromise through man in the middle attack
- **DNS attacks are often a precursor to other attacks**
- **DNS attack tools are readily available on the Internet**
- **All parts of the DNS hierarchy are vulnerable to attack, i.e., root level to lowest lever resolver and client**

Why is This Important?

- **Infrastructure problems present a unique challenge**
  - New capabilities must not disrupt old implementations
    - ‘Backward Compatibility’ essential to successful fielding
  - Difficult for applications to counter infrastructure attack
    - Typically, there is no alternative if DNS fails
- **For most applications and end users, when their DNS service is not working correctly, the “INTERNET IS DOWN“**

What Does DNSSEC Do?

- **Provides an approach so DNS users can:**
  - Validate that data they receive came from the correct originator, i.e., Source Authenticity
  - Validate that data they receive is the data the originator put into the DNS, i.e., Data Integrity
- **Approach integrates with existing server infrastructure and user clients**
- **Maximize benefit when application software can determine if DNS data was received with authenticity and integrity**

Course Outline

- **Introduction**
- **DNSSEC mechanisms**
  - New RRs
  - Signing a single zone
  - Building chains of trust
  - Key exchange and key rollovers
  - Show how each of these things can be simplified with new tools.
    - References to NIST Guide 800-81 will be listed.
- **DNSSEC-Capable Applications**
- **Operational concerns**
- **Conclusions**

DNS: Known Concepts

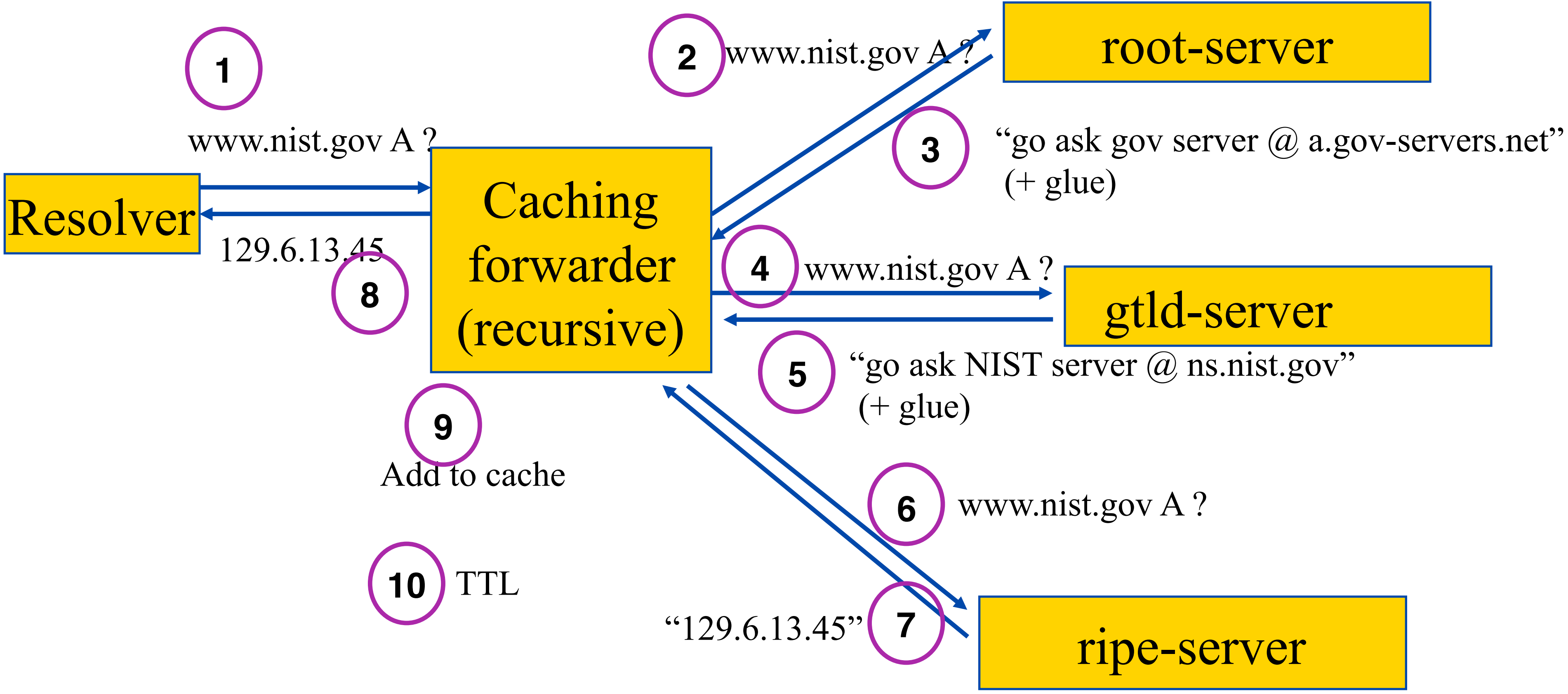
- **Known DNS concepts:**
  - Delegation, Referral, Zone, RRs, label, RDATA, authoritative server, caching forwarder, stub and full resolver, SOA parameters, etc
  - Don’t know? Do ask!

- **Operational knowledge with BIND**
  - BIND 9 named.conf, writing zone files
  - All examples based on IPv4

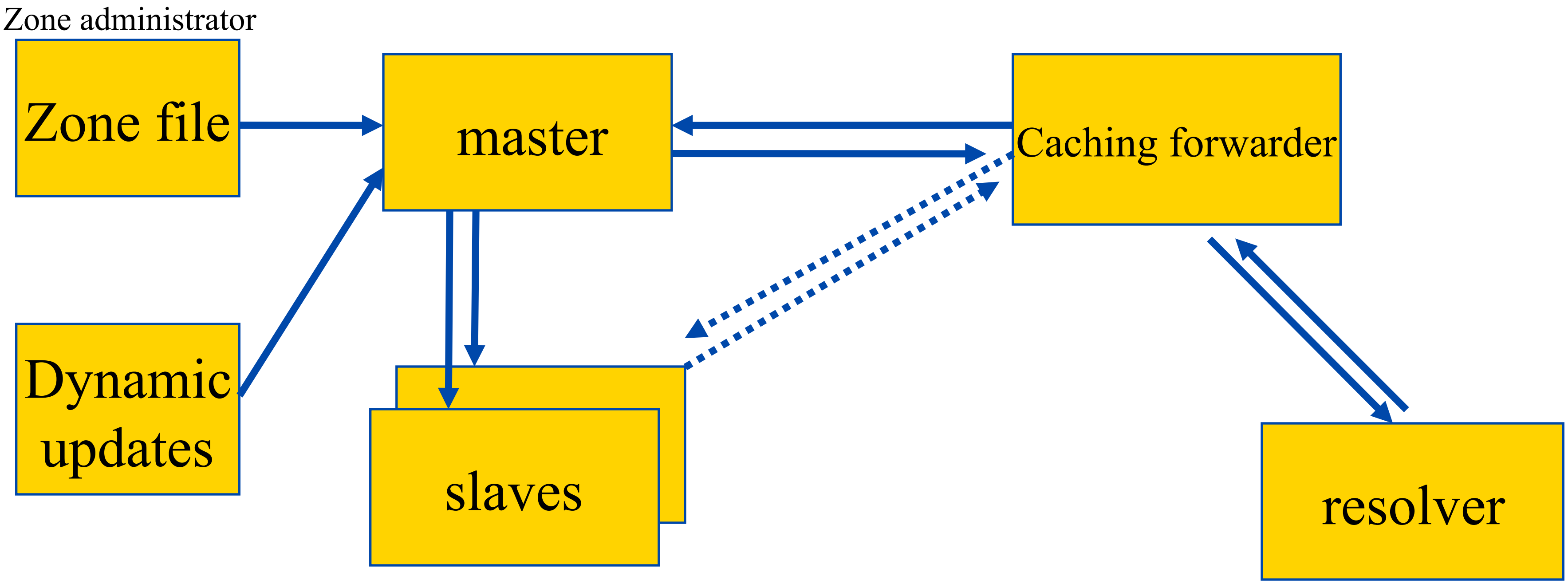
Reminder: DNS Resolving

Question:

www.nist.gov A



DNS: Data Flow



DNS Protocol Vulnerability

- **DNS data can be spoofed and corrupted between authoritative servers and resolvers or forwarders**
- **The DNS protocol does not allow you to check the validity of DNS data**
  - Polluted caching forwarders can cause harm for quite some time (TTL)
  - Corrupted DNS data might end up in caches and stay there for a long time

DNSSEC protects..

- **DNSSEC protects against data spoofing and corruption**
  - TSIG/SIG0: provides mechanisms to authenticate communication between servers
  - DNSKEY/RRSIG/NSEC: provides mechanisms to establish authenticity and integrity of data
  - DS: provides a mechanism to delegate trust to public keys of third parties
- **A secure DNS will be used as an infrastructure with public keys**
  - However it is **NOT** a PKI

DNSSEC Current State

- **RFC 4033**
  - DNS Security Introduction and Requirements
- **RFC 4034**
  - Resource Records for the DNS Security Extensions
- **RFC 4035**
  - Protocol Modifications for the DNS Security Extensions

- **NIST SP 800-81 (Deployment guide for DNSSEC)**
  - Reference for FIMSA controls on DNS Security

Configuration & Installation

- BIND 9.3 or later supports current DNSSEC
  - BUT: BIND 9.4.1 has support for TSIG with SHA-1
  - ftp://ftp.isc.org/isc/bind9/
- TSIG requires servers to sync time (time zone!)
- Openssl libraries required for crypto parts
  - http://www.openssl.org/
- Any FIPS 140-2 compliant libraries should be acceptable

Bind DNSSEC Tools

- **named**
- **dnssec-keygen**
  - Generate keys of various types
- **dnssec-signzone**
  - Sign a zone
- **dig**
  - Troubleshoot: Usage: dig +dnssec @…
- **named-checkzone & named-checkconf**
  - syntax check for zone files and named.conf

Server/Named Configuration

- **The configuration file is called “named.conf”**
- **Documentation in <src>/doc/arm/Bv9ARM.html**

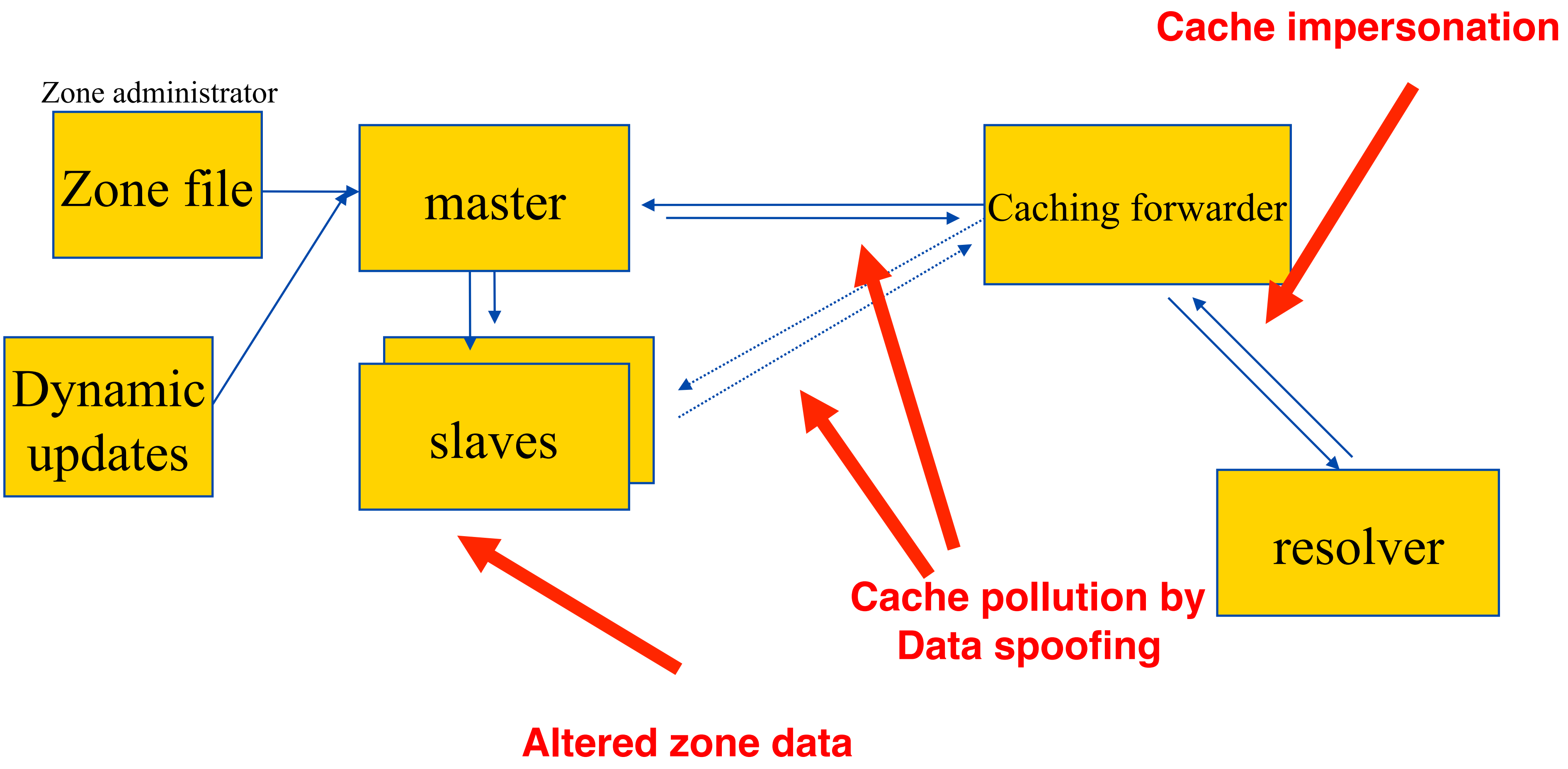
- **Turn on DNSSEC in “options” statement**
  - dnssec-enable yes;

- **Turn on logging for troubleshooting**
  - Several categories
  - Categories are processed in one or more channels
  - Channels specify where the output goes

DNSSEC Mechanisms

- New Resource Records
- Setting up a Secure Zone
- Delegating Signing Authority
- Key Rollovers

Vulnerabilities protected by DNSKEY / RRSIG / NSEC



DNSSEC hypersummary

- **Data authenticity and integrity by signing the Resource Records Sets with private key**

- **Public DNSKEYs 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)

- **Ideal case: one public DNSKEY distributed**

The DNS is not a Public Key Infrastructure (PKI)

- **All key procedures are based on local policy**
- **A PKI is as strong as its weakest link**
  - Certificate Authorities control this by SLAs
- **The DNS does not have Certificate Revocation Lists**
- **If the domain is under one administrative control you might be able to enforce policy**

Zone Status Terminology

- **Verifiably Secure**
  - RRset and its RRSIG can be verified with a DNSKEY that can be chased back to a trusted key, the parent has a DS record
- **Verifiably Insecure**
  - RRset sits in a zone that is not signed and for which the parent has no DS record
- **BAD (or BOGUS)**
  - RRset and its RRSIG can not be verified (somebody messed with the sig, the RRset, or the RRSIG expired)
  - A zone and its subzones are BAD when the parent’s signature over the Child’s key (DS) is BAD

New Resource Records

- **3 Public key crypto related RRs**
  - RRSIG: Signature over RRset made using private key
  - DNSKEY: Public key, needed for verifying a RRSIG
  - DS: Delegation Signer; 'Pointer' for building chains of authentication

- **One RR for internal consistency**
  - NSEC: Indicates which name is the next one in the zone and which typecodes are available for the current name
  - NSEC3: NSEC RR using hashed names (variant)

RR's and RRsets

- **Resource Record:**
  - name   TTL   class   type rdata

```
www.dnsops.gov.  7200  IN A  129.6.100.200
```

- **RRset: RRs with same name, class and type:**

```
dnsops.gov.   7200   IN  NS  snip1.dnsops.gov
                      IN NS  snip2.dnsops.gov
```

- **RRsets are signed, not the individual RRs**

DNSKEY RDATA

- 16 bits: FLAGS
- 8 bits: protocol
- 8 bits: algorithm
- N*32 bits: public key

Example:

```
dnsops.gov. 3600 IN DNSKEY   256  3  5 (
                  AQOvhvXXU61Pr8sCwELcqqq1g4JJ
                  CALG4C9EtraBKVd+vGIF/unwigfLOA
                  O3nHp/cgGrG6gJYe8OWKYNgq3kDChN)
```

RRSIG RDATA

- 16 bits - type covered
- 8 bits - algorithm
- 8 bits - nr. labels covered
- 32 bits - original TTL

```
dnsops.gov.  3600 IN  RRSIG   A  5  2  3600  (
        20031104144523 20031004144523  3112 dnsops.gov.
             VJ+8ijXvbrTLeoAiEk/qMrdudRnYZM1VlqhN
             vhYuAcYKe2X/jqYfMfjfSUrmhPo+0/GOZjW
             66DJubZPmNSYXw== )
```

- 32 bit - signature expiration
- 32 bit - signature inception
- 16 bit - key tag
- signers name

Delegation Signer (DS)

- **Delegation Signer (DS) RR indicates that:**
  - delegated zone is digitally signed
  - indicated key is used for the delegated zone
- **Parent is authoritative for the DS of the child’s zone**
  - Not for the NS record delegating the child’s zone!
  - DS **should not** be in the child’s zone

DS RDATA

- 16 bits: key tag
- 8 bits: algorithm
- 8 bits: digest type
- 20 bytes: SHA-1 Digest

```
$ORIGIN .gov.
dnsops.gov.   3600 IN   NS   snip1.dnsops.gov.
dnsops.gov.   3600 IN   DS   3112  5 1 (
         239af98b923c023371b11g23b92da12f42162b1a9
                                )
```

NSEC RDATA

- **Points to the next domain name in the zone**
  - also lists what are all the existing RRs for “name”
  - NSEC record for last name “wraps around” to first name in zone
- **N*32 bit type bit map**
- **Used for authenticated denial-of-existence of data**
  - authenticated non-existence of TYPEs and labels
- **Example:**

```
www.dnsops.gov. 3600 IN   NSEC dnsops.gov. A RRSIG
  NSEC
```

NSEC & NSEC3 Records

- **If your query for data does not exist in a zone, the NSEC RR provides proof of non-existence**
- **If after a query the response is:**
  - NXDOMAIN: One or more NSEC RRs indicate that the name or a wildcard expansion does not exist
  - NOERROR and empty answer section: The NSEC TYPE array proves that the QTYPE did not exist
- **More than 1 NSEC may be required in response**
  - wildcards
- **NSEC records are generated by tools**
  - they also lexicographically order the zone
- **NSEC3 replaces domain names with hashed names.**
  - To reduce the threat of zone enumeration

Signing a Zone

Getting the Working Files

- **http://www.dnsops.gov/downloads/configfiles.zip**

- **Extract in a working directory that will be easy to remember**
  - named.conf A sample BIND configuration file
  - root.hint   The root hints file (used by recursive servers)
  - zonefile The zone database file we will use

Securing a Zone

1. Generate keypair

- include public key (DNSKEY) in zone file

2. Sign your zone; signing will:

- sort the zone
- Insert:
  - NSEC records
  - RRSIG records (signature over each RRset)
  - DS records (optional)
- generate key-set file (can be used later)

Securing a Zone - continued

3. Publish Signed Zone

4. Configure Resolvers (Recursive Servers)

5. Test

6. Distribute your public key (DNSKEY) to those that need to be able to trust your zone
   - Key-set or DS-set for Parent

Toolbag: dnssec-keygen

- **dnssec-keygen to generate keys**

```
 dnssec-keygen -a alg -b bits -n type [options]
name
```

- **algorithm:** RSASHA1
- **Bitsize:** size of the key (1024 bits for example)
- **type:** “zone”
- **Name:** zone name – the name for the key

- **‘-r /dev/urandom’ might be needed**

Generating keys

```
$dnssec-keygen -a RSASHA1 -b 1024 -n zone
  dnsops.gov.
Kdnsops.gov.+005+20704
$
```

- **2 files are created:**
  - `Kdnsops.gov.+005+20704.key`
    - contains the public key
    - should go into the zone file
  - `Kdnsops.gov.+005+20704.private`
    - contains the private key
    - **should be kept secret!!!**

Now Create 2 Keys

- **The Zone Signing Keys (ZSK)**
  - Used to sign the zone
  - -a RSASHA1 –b 1024 bits –n ZONE name = your domain
- **The Key Signing Key (KSK)**
  - Used to sign the ZSK (less frequently used)
  - Used to link security from parent to you
  - -a RSASHA1 -b 2048 bits -n ZONE key -f KSK name= your domain
- **Write down the footprints of each key so you can tell them apart later!**

Signing and Serving a Zone

Only Authoritative Records are Signed

- **NS records for the zone itself are signed**
  - I “own” them, I sign them
- **NS records for delegations are not signed**
  - DS RRs are signed!
- **Glue is not signed**

Preparing the Zonefile

- **Include the public keys in the zone file:**
  - `cat Kdnsops.gov.+005+20704.key >> dnsops.gov`
  - `NOTE – might want to edit zone file to add in footprint as a comment (helpful)`
- **Use named-checkzone (optional)**
- **Increase the SOA serial number**
  - **Always increase the SOA serial before signing!**

Sign the Zone

```
dnssec-signzone [options] zonefile [ZSK’s]
```

- **If zone file name is not zone name:**
  - use `–o <origin>` option
- **Signed zone file is called “*zonefilename*.signed”**
- **Keyset and DSset files are created as a bonus…**
  - ready to go to parent

Now Sign the Zone

- **Example**

```
dnssec-signzone -k <KSK>  “zone” <ZSK>
```

  - KSK is the 2048 bit key generated with the “-f KSK”
  - ZSK is the 1024 bit key

- **Often done on separate machine, then transferred to servers.**

Publishing the signed zone

- **Edit named.conf:**

```
zone ”dnsops.gov." {
          type master;
          file "zones/dnsops.gov.signed";
          allow-transfer { 10.1.2.3 ;
                         key mstr-slave.; };
          notify yes;
};
```

- **Use named-checkconf (optional)**
- **Reload zone**
- **Test**

Testing the Signed Zone

```
dig +dnssec [@server] record [TYPE]
```

- **Answer Flags are relevant**
- **Example query to an authoritative name server**

```
; <<>> DiG 9.1.1 <<>> +dnssec @127.0.0.1
  www.dnsops.gov
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
  1947
;; flags: qr aa rd; QUERY: 1, ANSWER: 4, AUTHORITY: 3,
  ADDITIONAL: 4
```

- **Authoritative Answer (not Validated)**

Testing Through a Validator

```
dig +dnssec [@server] record [TYPE]
```

- **Answer Flags are relevant**
- **Example query to a recursive name server**

```
; <<>> DiG 9.1.1 <<>> +dnssec @127.0.0.1
  www.dnsops.gov
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
  1947
;; flags: qr ad rd ra; QUERY: 1, ANSWER: 4,
  AUTHORITY: 3, ADDITIONAL: 4
```

- **AD bit indicates it has been validated**

Troubleshooting (Client Side)

- Dig returns status: SERVFAIL
- First try without `+dnssec`
- Also try with `+dnssec +cdflag`
  - Checking is disabled. Data directly forwarded
- Be ready for some interesting troubleshooting

Troubleshooting (Server Side)

- Turn on logging. Category “dnssec” with severity debug 3 gives you appropriate hints
- Debug output is a little detailed ☺

Delegation Signer (DS) RR’s

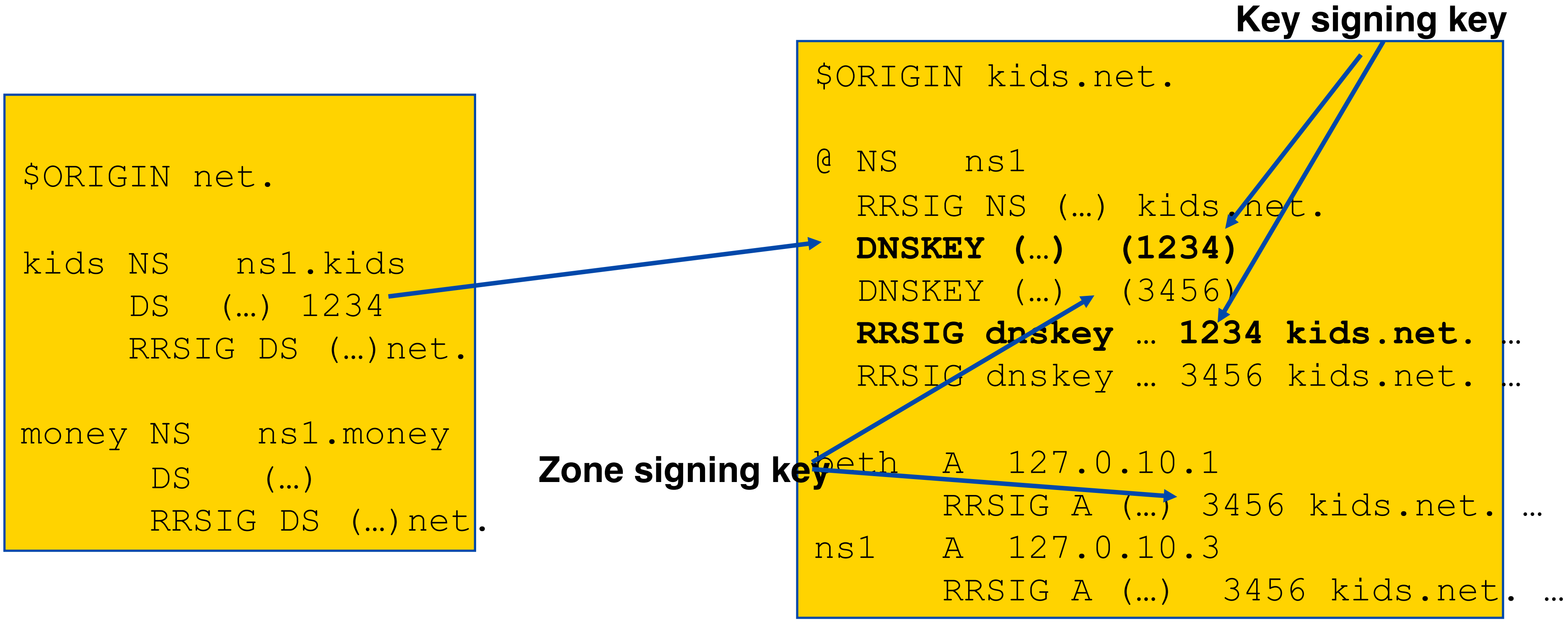
Securing the delegation

DS RRs for Delegations

- **Parent is authoritative for the DS record**
  – It should not appear in the child's zone file
- **DS resource records are used for Delegation of Security**
  – Parent zone can indicate absence of DS RR as well
- **Eases resigning**
  – parent can sign often => short signature lifetime => shorter impact time when key gets compromised
- **Simplifies key distribution problem**
  – Push problem up to parent and reduce the number of keys needed to validate entire DNS tree.

Delegating Signing Authority

- **Parent signs the DS record pointing to the key signing key**



```
$ORIGIN net.

kids NS    ns1.kids
     DS   (…) 1234
     RRSIG DS  (…)net.

money NS    ns1.money
      DS    (…)
      RRSIG DS  (…)net.
```

```
$ORIGIN kids.net.

@ NS    ns1
  RRSIG NS  (…)  kids.net.
  DNSKEY  (…)   (1234)
  DNSKEY  (…)   (3456)
  RRSIG dnskey … 1234 kids.net.  …
  RRSIG dnskey … 3456 kids.net.  …

beth   A  127.0.10.1
       RRSIG A  (…)  3456 kids.net.  …
ns1    A  127.0.10.3
       RRSIG A  (…)   3456 kids.net.  …
```

- The parent is authoritative for the DS RR of its children

Chain of Trust Verification, Summary

- Data in zone can be trusted if signed by a Zone-Signing-Key
- Zone-Signing-Keys can be trusted if signed by a Key-Signing-Key
- Key-Signing-Key can be trusted if pointed to by trusted DS record
- DS record can be trusted
  - if signed by the parents Zone-Signing-Key

  or

  - DS or DNSKEY records can be trusted if exchanged out-of-band and locally stored (Secure entry point)

Setting up a Secure Resolver

Setup and Configuration

Setting up a Validating Name Server

- **To verify the content of a zone:**
  - Get the public (key signing) key and check that this key belongs to the zone owner

- **Configure the keys you trust as secure entry points in named.conf**

```
trusted-keys {
        "dnsops.gov." 257 3 1 "AQ…QQ==";
        };
```

End User Side

- **Local verifying/recursive server trusted?**
  - TSIG for queries?
  - IPSec?
  - Use local validation?
- **How much information needed?**
  - AD bit enough?
  - Local validation using the (draft) validator API is able to return detail information about the validation results to the end application.

Server Operational Considerations

Key Rollovers

Key Rollover

- **Try to minimize 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 2nd party or parties:**
  - State to be maintained during rollover
  - operationally expensive
  - Refer to NIST SP 800-81r1 for full details

ZSK Rollover

1. **Generate new ZSK**
   - 1024 bit RSASHA1 key
2. **Add key to zone**
   - Remember to increase the serial number
3. **Re-sign zone (using old key and KSK)**

```
dnsops.gov   SOA
             RRSIG  (old-zsk)

             DNSKEY  old-zsk
             DNSKEY  new-zsk
             DNSKEY  KSK
             RRSIG  (old-zsk)
             RRSIG  (KSK)
```

ZSK Rollover

4. Time passes… (TTL)
5. Now re-sign (again) with the new key

```
dnsops.gov  SOA
         RRSIG (new-zsk)

         DNSKEY  old-zsk
         DNSKEY  new-zsk
         DNSKEY  KSK
         RRSIG (new-zsk)
         RRSIG (KSK)
```

ZSK Rollover

6. Remove old key*
7. Resign one last time

```
dnsops.gov  SOA
         RRSIG (new-zsk)

         DNSKEY  new-zsk
         DNSKEY  KSK
         RRSIG (new-zsk)
         RRSIG (KSK)
```

*** Wouldn’t it be nice to add a new future key here (like in step 1-2 before)? ☺**

KSK Rollover

1. **Generate new KSK**
   - 2048 bit with the “-f KSK” option
2. **Add new KSK to the zone and resign keyset/zone**

```
dnsops.gov  SOA
           RRSIG (ZSK)

           DNSKEY  KSK
           DNSKEY  new-KSK
           DNSKEY ZSK
           RRSIG (new-KSK)
           RRSIG (KSK)
```

3. **Wait TTL of the zone**

KSK Rollover

4. **Upload new DS or Keyset to parent zone.**

5. **When new DS RR appears in the zone, wait TTL of the DS RRset (to be safe)**

6. **Pull old KSK and resign zone**

```
dnsops.gov  SOA
          RRSIG (zsk)

          DNSKEY  new-KSK
          RRSIG (new-KSK)
```

Conclusions

What Did We Cover

- **DNSSEC provides a mechanism to protect DNS**
- **DNSSEC implementation:**
  - RRSIG, DNSKEY and NSEC/NSEC3 for data
  - DS for delegating trust
- **DNSSEC main difficulties:**
  - Key distribution
  - Time now a factor in operations

When You Get Back Home

- **Have a plan before deployment**
  - Make sure everything is documented
  - Identify areas where upgrades are needed
- **Consider testing with a non-production zone (using the SNIP for example)**
- **Chose one zone as the pilot program**
  - Expand from there
- **Look at entire network for weak points**
  - older Firewalls and routers may not handle larger DNS packets

Additional Resources

- **NIST Secure Naming Infrastructure Pilot (SNIP)**
  - http://www.dnsops.gov/
    - Contains links to NIST Guidance documents and DNSSEC testbed
- **DNSSEC Deployment Initiative**
  - http://www.dnssec-deployment.org/
- **DNSSEC.NET**
  - http://www.dnssec.net/