

DNSSEC Tutorial: How it Works, and Where to Start

Scott Rose, NIST

scottr@nist.gov

2011 Winter JointTechs Meeting

Jan 30, 2011

Clemson University

Special Thanks to RIPE NCC who provided the base slides for this tutorial.

Why DNSSEC?

- DNS is not secure
 - Known vulnerabilities to spoofing
 - So easy, even a script kiddie can do it!
 - People depend more and more on DNS
 - Sure you can memorize a few IPv4 addresses, but how about IPv6?
- DNSSEC protects against data spoofing and corruption

DNS Background

- Created as scaleable method to replace original ARPANet host & address file:
 - ‘hosts.txt’ file distributed by SRI-NIC
- DNS was the first standards based name system:
 - Proved viability of interoperable infrastructure service based on specifications rather than single vendor implementation
- Originally, no attempt to identify DNS security requirements
- DNS quickly became a critical service of the infrastructure

Why Worry About DNS Security?

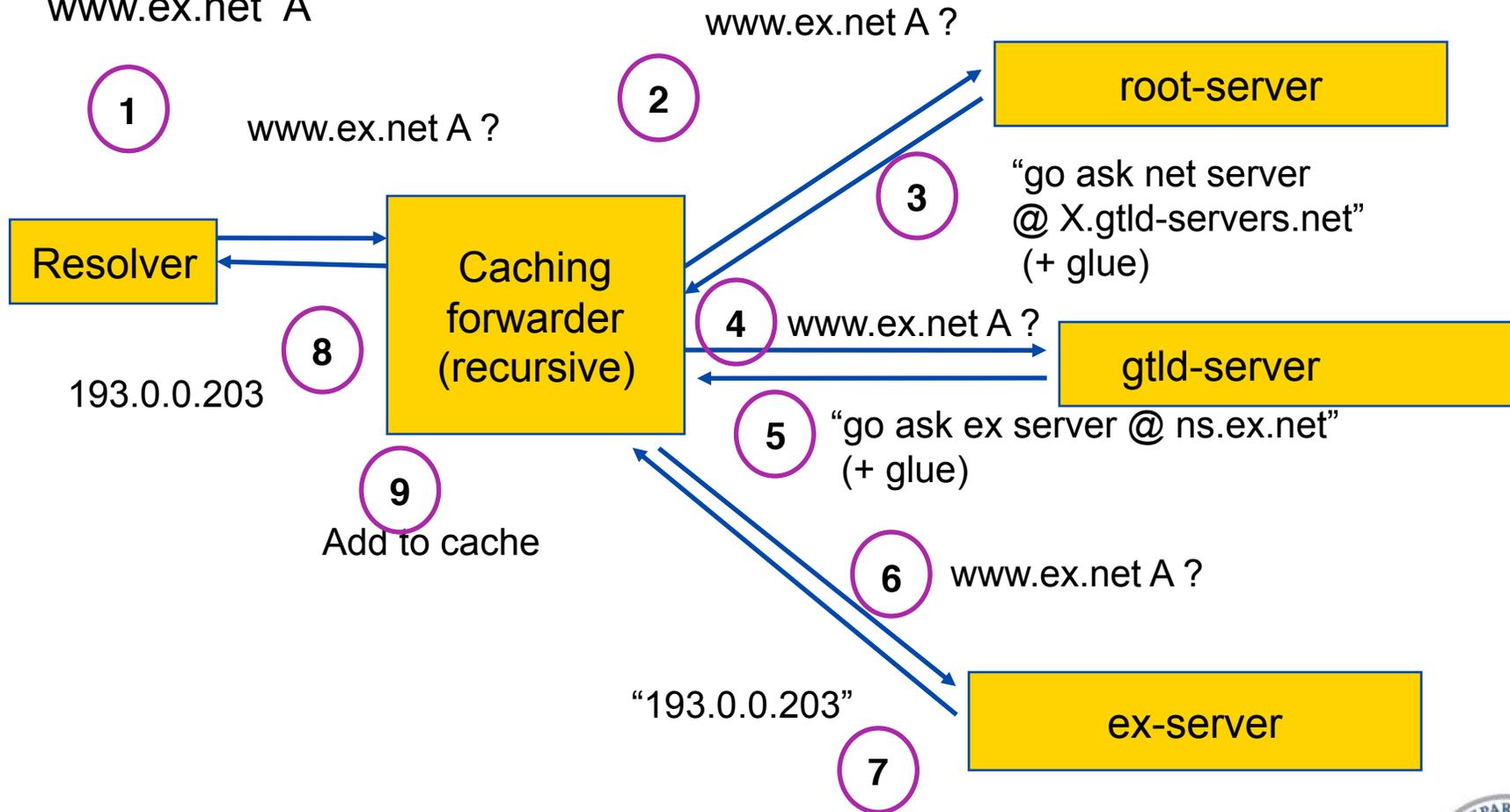
- Forged DNS data subverts most applications:
 - Web site can be replaced with a false site without ever touching the victim site
 - EMail can be re-routed or mis-delivered
 - Login compromise through man-in-the-middle attack
 - *See D. Kaminsky's Black Hat 2008 talk for the gory details*
- DNS attacks 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

What Does DNSSEC Do?

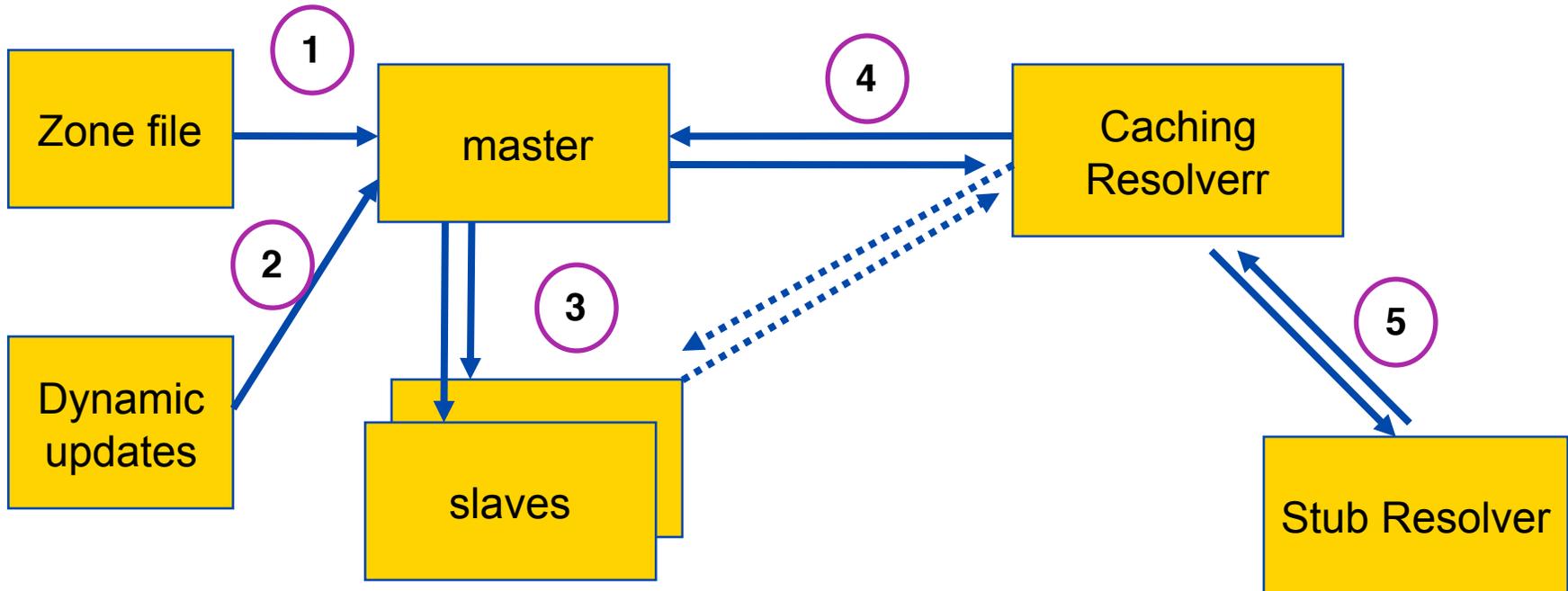
- Provides an approach so DNS users can:
 - Validate that data they receive came from the correct originator, i.e., Source Authenticity
 - Validate that data they receive is the data the originator put into the DNS, i.e., Data Integrity
- Approach integrates with existing server infrastructure and user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

Reminder: DNS Resolving

Question:
www.ex.net A

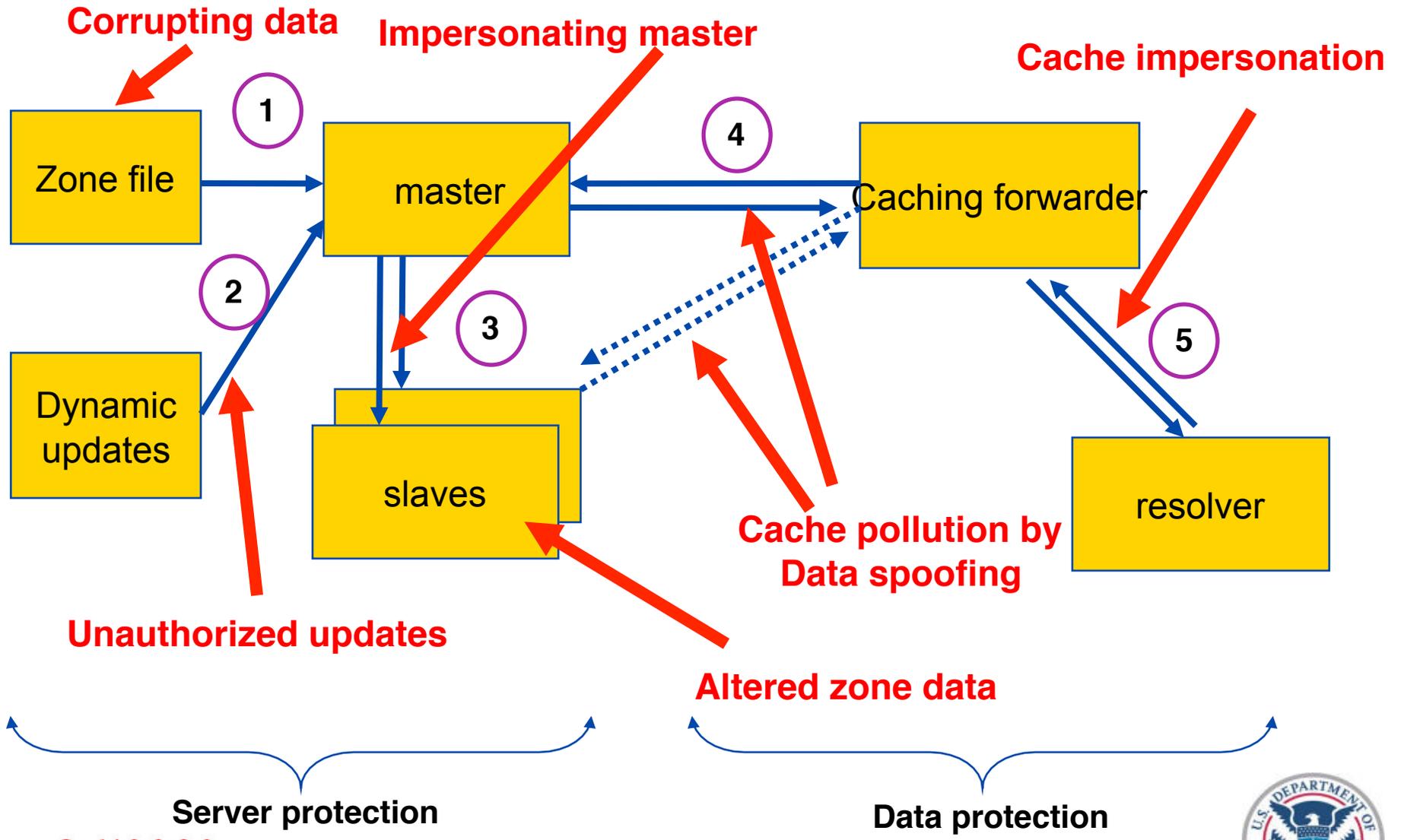


DNS: Data Flow



DNS Vulnerabilities

Cyber and Network Security Program



DNSSEC protects..

- DNSSEC protects against data spoofing and corruption
 - DNSKEY/RRSIG/NSEC/NSEC3: provides mechanisms to establish authenticity and integrity of data
 - DS: provides a mechanism to delegate trust to public keys of third parties
- Related: TSIG/SIG0: provides mechanisms to authenticate communication between servers
 - Uses a shared secret, so you have to have a prior relationship

DNSSEC Hyper-summary

- 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)

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: 'Pointer' for building chains of authentication
- One (Two) RR's 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)
 - NSEC3PARAMS store info about the NSEC3 variables used.

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
			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 (encoded)

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.  
                  3600 IN      DS      3112  5 1  (  
                  239af98b923c023371b52  
                  1g23b92da12f42162b1a9  
                  )
```

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
 - data that can be signed offline.

- Example:

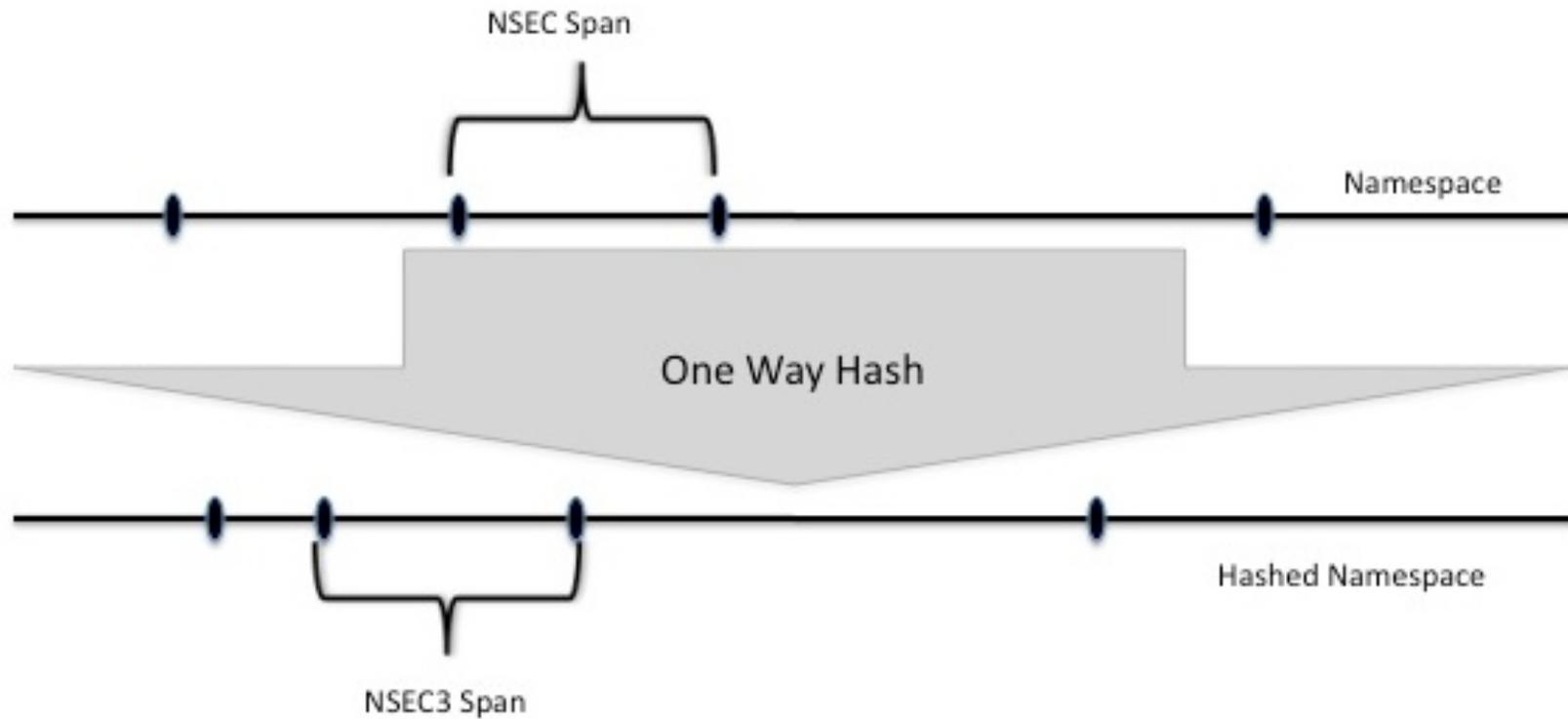
`www.dnsops.gov. IN NSEC dnsops.gov A RRSIG NSEC`



NSEC3 Record

- Format and use the same as the NSEC Record
- Uses hashed names instead of cleartext
 - Hashed names exist in separate “ghost” namespace (so you can't query for them directly)
- Used to protect privacy and combat zone enumeration
 - Zone enumeration: following an NSEC chain to obtain all the contents of a zone
 - Doesn't stop enumeration (DNS is public, so it's impossible)

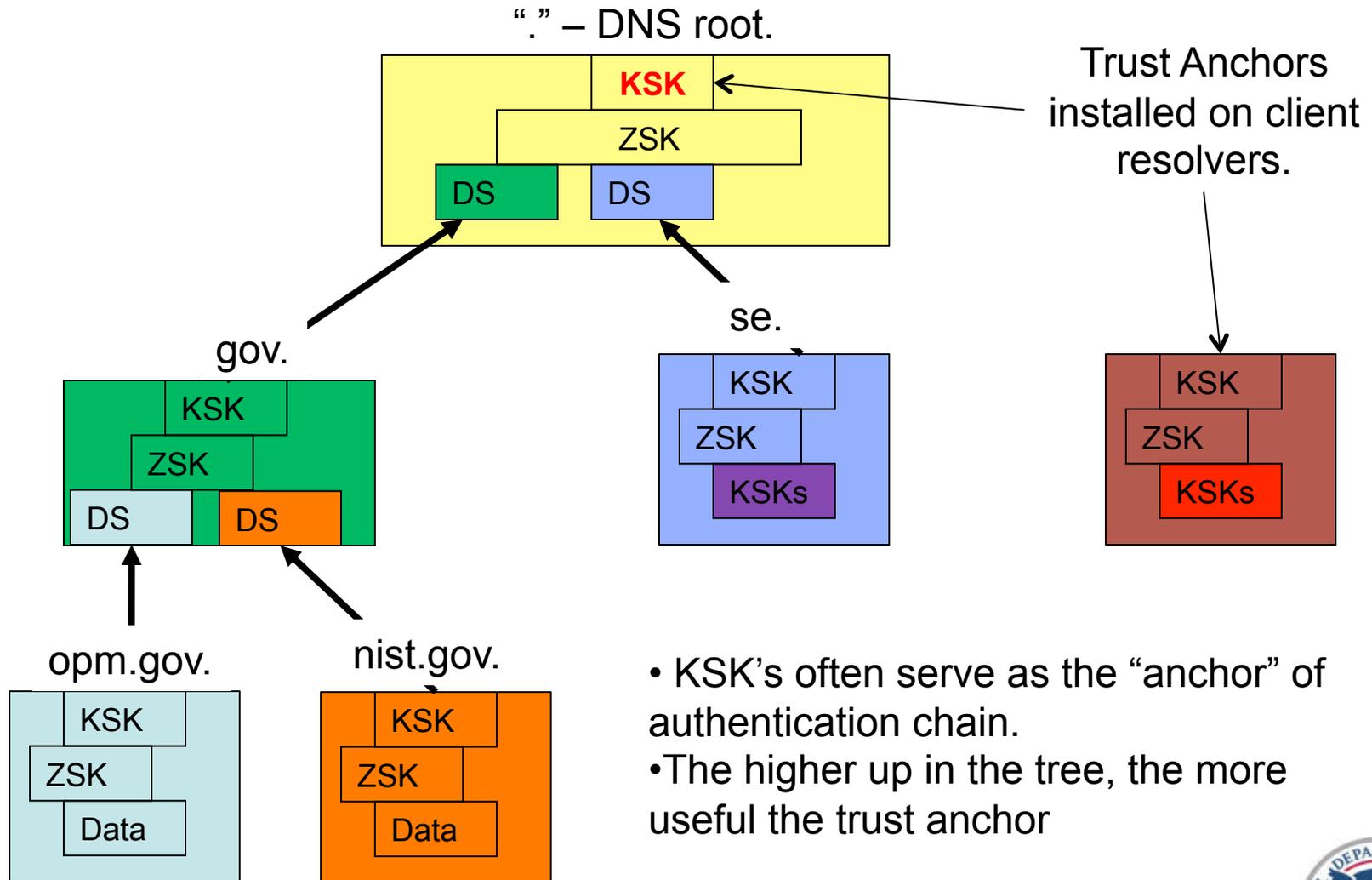
NSEC vs. NSEC3



Authentication Chains

- DS RR's used to link parent and child
- DS points to specific key
 - Signature from that key over DNSKEY RRset transfers trust to all keys in DNSKEY RRset
- Key that DS points to only signs DNSKEY RRset (usually, but not always).
 - Key Signing Key (KSK)
- Other keys in DNSKEY RRset sign entire zone
 - Zone Signing Key (ZSK)

DNSSEC Chain of Trust



- KSK's often serve as the "anchor" of authentication chain.
- The higher up in the tree, the more useful the trust anchor

Signing a Zone

Finally! We get to the Good Stuff...

1. Generate keypair(s)
 - include public key (DNSKEY) in zone file*
 - * This is a zone update! So SOA serial number must be incremented.
2. Sign your zone; signing will:
 - sort the zone
 - Insert:
 - NSEC/NSEC3 records
 - RRSIG records (signature over each RRset)
 - DS records (if signed delegations present)

Securing a Zone - continued

3. Publish Signed Zone
 - May require server configuration changes
4. Configure Resolvers (Recursive Servers) – Optional
5. Establish Secure Chain with parent and/or distribute key for use as a trust anchor

I'll use the default BIND tools for this – it will be ugly, but will show all the steps. There are more advanced tools available, but those hide the details (for good reason, as you'll see).

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

Generating Keys

- Using the `dnssec-keygen` command
 - Generate 2 keys (a ZSK and KSK)
 - Size and algorithm depends on local policy
 - KSK as the `-f KSK` option selected
 - Two files are generated
 - *.private has the private data
 - *.key has the public portion in DNS RR format
 - Note that the ZSK has flags of 256 and KSK has 257
- Add the *.key file data to the zone file
 - either copy/paste or use the BIND `$INCLUDE` statement:
`$INCLUDE "K<zonename>+005+<footprint>.key"`
- Don't Forget to Update the SOA Serial Number!

Signing the Zone

- Using the `dnssec-signzone` command
 - The default is NSEC, NSEC3 requires additional option flags (see the help for all of them)
 - Parameters – iterations should be below 250, salt just random hex strings (suggested length: around 18 octets)
 - Might want to include the “-P” command to avoid having the signer validate its own work (not needed usually and often returns an error that stops the process)
 - Default output is the zonefile with the same name plus “.signed” at the end.
 - Two other files: `*.keyset` has the DNSKEY RRset in a different format and `*.dsset` as DS RR’s made of all the keys with flags = 257 (i.e. KSK’s)

Uploading the New Zonefile

- The signed zonefile is just like any other zone file
 - In BIND, edit the named.conf file to point to the new file
 - i.e. “zonefile.signed” instead of “zonefile”
- BIND requires a config change to enable DNSSEC (some other servers don't)
 - in the `options` statement:

```
dnssec-enable yes;
```

Uploading and Testing

- Remember to upload your *.keyset or *.dsset (as necessary) to the right party (for .edu – EDUCAUSE)
- Test using dig:
 - dig @<serverIP> <zonename> DNSKEY +dnssec
 - The “+dnssec” signals dig to ask for RRSIGs
 - The output should have RRSIGs
 - If not, check the configuration of the server and make sure the server has loaded the signed zonefile.

Configure the Clients

- Need to install at least one public key in your recursive servers to anchor the authentication chain.

– In BIND: use the “trusted-keys” statement:

```
trusted-keys {  
  "gov."      257  7 3  "BQEAAApA3pKwuBOv6Cx3gK9yg6fxc  
                      kYuM5lBxHKhgrYFSLbWirnjA4R0QlG0zDj  
                      ...";  
};
```

Remember to restart the server!

Now Test Using dig

- The dig tool is included in the BIND package:
 - general format:
> dig @server <domain name> <type> <flags>
- Relevant flags:
 - '+dnssec' request DNSSEC processing
 - '+cdflag' return raw response, do not do validation
 - '+tcp' use TCP instead of UDP
- Try both a recursive server and authoritative servers.

We're Done! (for a while)

- Remember that RRSIGs have a lifespan and need to be regenerated
 - Before the old RRSIG's expire
 - Every time the zone changes (redo NSEC/NSEC3 chain)
- Some tools allow for making minor changes without having to resign the entire zone
 - Also for zones that use dynamic update

...And we're back.

- Complicated, right? Luckily computers are good at some things:
 - Following a set process
 - Scheduled tasks
 - ~~Go back in time and find Sarah Conner~~
- There are tools to make life easier
 - Open source
 - Turnkey appliances
 - Outsource services

We're not done yet...

- Need to set up processes to perform maintenance tasks:
 - editing zone
 - resigning zone
 - key rollovers
- Keys should be changed on a regular basis
 - Not simple, since caching makes life difficult for zone operators
 - Not impossible either
- See RFC 4641 or NIST SP 800-81r1 for details

Key Rollovers

- 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
- KSK rollover involves 2nd party or parties:
 - State to be maintained during rollover
 - operationally expensive

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...
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)? 😊

ZSK Rollover

- Isn't this sort of a pain?
 - Yes and that's why a lot of companies produce tools to automate this process
- What about KSK?
 - Will require more interaction with EDUCAUSE

USG DNSSEC Experiences

- Lessons Learned
 - Planning for DNSSEC provides opportunity to revisit DNS structure.
 - Many agency level DNS operators were forced to discover and revisit their DNS architectures.
 - Many “new” DNSSEC management processes improve existing DNS operations.
 - DNSSEC requires regular maintenance (e.g. resigning)
 - DNSSEC inspires careful consideration of authentication, notification, and monitoring process to maintain signed zones.

Lessons Learned

- Administrator education should be a major priority during deployment.
 - Admin error the cause of most problems
 - Give administrators time to plan and clear policy guidance about what they need to do.
 - Know who to contact when mistakes occur
 - Establish a help desk/support network to resolve issues.
- For large domains: establish a procedure for your delegations to upload key material to the parent zone

Lessons Learned

- DNSSEC centric crypto policy is important (DNSSEC is not a PKI)
 - US Federal key policy aimed at PKI certificates (i.e. large, long lived keys), not DNSSEC.
 - causes large response sizes and problems in some routers/firewalls
- Look at your other network components for hidden dangers
 - Old routers/switches or firewalls may drop large DNSSEC responses
 - 1500 bytes a reasonable MTU setting
 - Firewall rules may need changed (UDP & TCP port 53)

Looking Ahead

- Now that we have a signed infrastructure, what can we use it for?
 - email certs
 - ID credentials
 - TLS/SSL certs
 - other protocols?
- People are starting to look at the DNS to support cross-enterprise authentication and trust.

So to Wrap Up:

- First, plan your deployment
 - DNS architecture, content management, key policy, etc.
- Second, start small
 - use a test delegation or pilot program
- Third, make sure everyone knows their roles
 - write down procedures for admins, help desk, etc., train them if necessary!
- Fourth, sign first, validate later
 - make sure the infrastructure is solid

Resources

- DNSSEC Deployment Initiative
<http://www.dnssec-deplyment.org/>
- NIST Secure Naming Infrastructure Pilot
<http://www.dnsops.gov/>
- NIST Special Publications
<http://csrc.nist.gov/>
- Good place for all things DNSSEC
<http://www.dnssec.net/>