LOPSA
the league of professional system administrators
To advance the practice of system administration; to support, recognize, educate, and encourage its practitioners; and to serve the public through education and outreach on system administration issues.

DNS and DNSSEC

Shumon Huque
University of Pennsylvania

Professional IT Community Conference
New Brunswick, New Jersey, May 12th 2012

usenix
ASSOCIATION

Penn
UNIVERSITY *of* PENNSYLVANIA

Version: 2012-05-12-01

DNS and DNSSEC

This tutorial was presented at the PICC 2012 Conference held in New Brunswick, NJ, on May 12th 2012.

Feedback, critique, suggestions on these slides gladly received at <shuque @ upenn.edu>

Reminder: Please fill out the evaluation forms for this course!

Course blurb from PICC conference brochure:

This tutorial will provide system administrators an understanding of the DNS protocol, including advanced topics such as DNSSEC (DNS Security). It will provide practical information about configuring DNS services using examples from the popular ISC BIND DNS software platform.

Topics include: the DNS protocol and how it works, DNS master zone file format, a look at a variety of server configurations and recommendations, DNSSEC (DNS Security Extensions) and how to deploy it, many examples of DNS query and debugging using the "dig" tool, DNS and IPv6, and more.

Who am I?

- An I.T. Director at the University of Pennsylvania
- Have also been:
  - Programmer (C, Perl, Python, Lisp)
  - UNIX Systems Administrator
  - Network Engineer
- Education: B.S. and M.S. (Computer Science) from Penn
- Also teach a Lab course on Network Protocols at Penn's School of Engineering & Applied Science

Course Topics

1. DNS Tutorial
2. Configuring DNS in BIND
3. Live queries using ‘dig’
   [... break ...]
4. DNSSEC Tutorial
5. Configuring DNSSEC in BIND
6. Application uses of DNSSEC
7. DNSSEC deployment status

DNS Tutorial

DNS

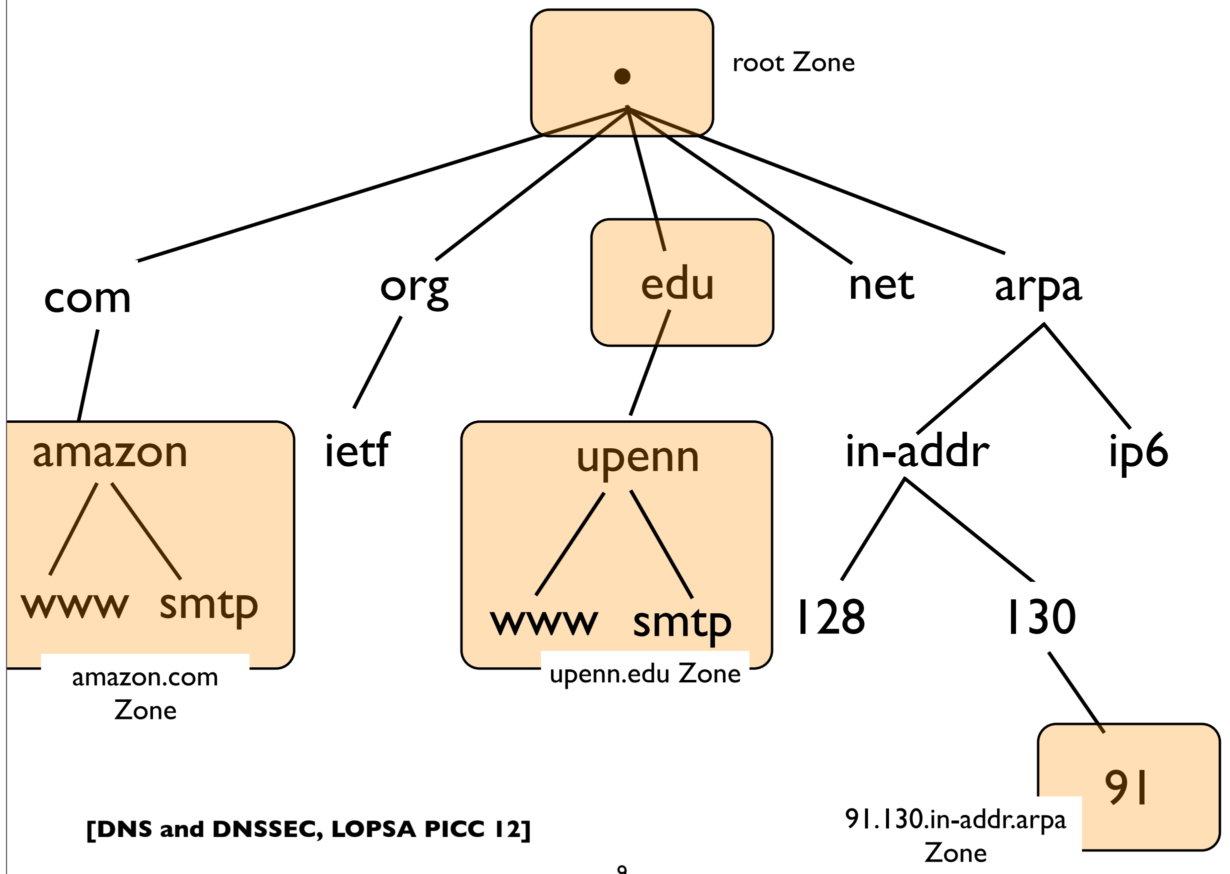
- Domain Name System
- Original specifications in RFCs 1034 and 1035
- Distributed global database
- Indexed by "domain names" (together with a type and class)
- A domain name is a sequence of labels, eg.
  - www.amazon.com.
- Domain Names are case insensitive, but case preserving
- Transport protocol: UDP and TCP port 53

[DNS and DNSSEC, LOPSA PICC 12]

DNS

- (See diagram on next slide)
- DNS can be represented as a tree of labels
- Sibling nodes must have unique labels
- Domain name at a particular label can be formed by the sequence of labels traversed by walking up the tree from that label to the root
- Zone - autonomously managed subtree
- Delegations: boundaries between zones

[DNS and DNSSEC, LOPSA PICC 12]



[DNS and DNSSEC, LOPSA PICC 12]

Root and TLDs

- Root of the DNS ("empty label")
- Next level of names are called Top Level Domains (TLDs)
- Until recently 3 primary classes of TLDs
  - GTLD: Generic Top Level Domains (.com, .net, .edu, .org etc)
  - CCTLD: Country Code TLD (2 letter codes for each country, eg. .us, .fr, .jp, .de, ...)
  - Infrastructure: eg. .arpa etc (uses: reverse DNS e164, etc)
- IDN cctld (Internationalized domain name ccTLD)
- The new gTLDs - the wild west? (newgtlds.icann.org)

[DNS and DNSSEC, LOPSA PICC 12]

DNS main components

- Server Side:
  - Authoritative Servers
  - Resolvers (Recursive Resolvers)
- Client Side:
  - Stub resolvers (usually on DNS client machines)

[DNS and DNSSEC, LOPSA PICC 12]

Authoritative Server

- A server that directly serves data for a particular zone
- Said to be "authoritative" for that zone
- These servers are the ones specified in NS records

[DNS and DNSSEC, LOPSA PICC 12]

Resolver

- Aka "Recursive Resolver", "Cache" etc
- Used by endsystems (stub resolvers) to query ("resolve") arbitrary domain names
- Receives "recursive" queries from these endsystems
- Resolvers query authoritative servers, following DNS delegations until they obtain the answer they need (this process is called "iterative" resolution)
- Resolvers "cache" (remember) query results for the specified "TTL" (also some negative results are cached)

[DNS and DNSSEC, LOPSA PICC 12]

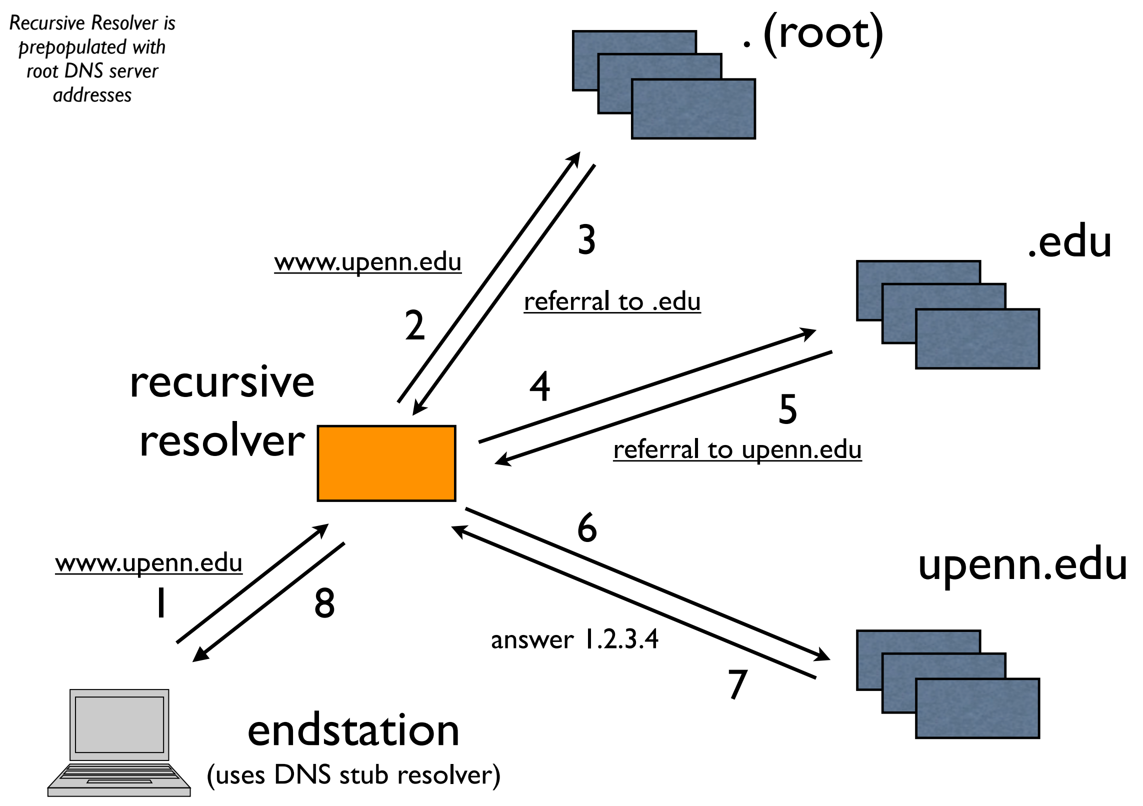
Stub Resolver

- The DNS client software component that resides on most endsystems
- Commonly implemented by the Operating System as a set of library routines
- Has a configured set of addresses of the Recursive Resolvers that should be used to lookup ("resolve") domain names
  - usually by manual configuration, or dynamically learned via DHCP

[DNS and DNSSEC, LOPSA PICC 12]

Stub resolver configuration

```
$ cat /etc/resolv.conf

search          finance.example.com example.com
;;
nameserver      10.12.3.1
nameserver      10.254.23.71
nameserver      10.15.18.9
;;
options timeout:1 attempts:2 rotate
```



Parts of a DNS query

- Each DNS query needs a query name, type, and class
- **qname**: a domain name, eg. www.upenn.edu
- **qtype**: A, AAAA, MX, CNAME, PTR, SRV, TXT, NS, SOA, ...
- **qclass**: IN, CH, HS (only "IN" is commonly used)
- Various flags: QR, RD, EDNS Opt, DO etc

[DNS and DNSSEC, LOPSA PICC 12]

Life of a typical DNS query

- Type "www.amazon.com" into browser
- Browser calls a name lookup function (eg. getaddrinfo())
- DNS may not be the only name lookup service in use. The lookup function might consult a nameservice switch table to figure out what order of services to consult (eg. /etc/nsswitch.conf -- flat file, LDAP, NIS, DNS etc)
- If/when DNS is used, then call DNS specific calls in stub resolver
  - res_ninit(), res_nquery(), res_nsearch()

[DNS and DNSSEC, LOPSA PICC 12]

Life of a typical DNS query

- Stub resolver formulates and makes DNS query:
  - qname www.amazon.com, qtype=A, qclass=IN
  - Note: IPv6 enabled resolvers might try AAAA, then A
- Sends query to DNS servers (resolvers) specified in stub resolver configuration (eg. /etc/resolv.conf) in the order specified until it gets a successful response, failure, or times out
- If a "search" domain list is configured, on lookup failure, the stub retries queries with domain suffixes from this list appended to the original query

[DNS and DNSSEC, LOPSA PICC 12]

Life of a typical DNS query

- DNS resolvers will get the answer:
  - from their authoritative zones if they have any relevant ones
  - from their cache if the answer is already there
  - by iterative queries of the DNS tree, as necessary, eg.
    - root servers, amazon.com servers, ...

[DNS and DNSSEC, LOPSA PICC 12]

Resource Records (RR)

- The fundamental unit of data in the DNS database
- A grouping of a {domain name, type, class}, a TTL (time-to-live), and the associated "resource data"
- Has a defined text "presentation format"

```
www.example.com.       86400 IN   A  10.253.12.7
```

name, or owner name	*ttl*	*class*	*type*	*rdata*

Resource Record Sets

- A set of RRs with the same name, class, and type
- The rdata (resource data) associated with each RR in the set must be distinct
- The TTL of all RRs in the set also must match
- RR sets are treated atomically when returning responses

```
www.ucla.edu.       300   IN   A   169.232.33.224
www.ucla.edu.       300   IN   A   169.232.55.224
www.ucla.edu.       300   IN   A   169.232.56.224
```

Resource Record types

Type	Description
SOA	marks **S**tart **O**f a zone of **A**uthority
NS	NameServer record
A	IPv4 Address record
AAAA	IPv6 Address record
CNAME	Canonical name (ie. an alias)
MX	Mail Exchanger record
SRV	Service Location record
PTR	Pointer (most commonly for reverse DNS)
TXT	Text record (free form text with no semantics)
NAPTR	Naming Authority Pointer Record

for full list, see
www.iana.org/assignments/dns-parameters

Other special RRtypes

Type	Description
TSIG	Transaction Signature (RFC 2845)
TKEY	Transaction Key (RFC 2930) - estab secret keys
AXFR	Zone Transfer
IXFR	Incremental Zone Transfer (RFC 1995)
OPT	Opt pseudo RR (RFC 2671 - EDNS0)

for full list, see
www.iana.org/assignments/dns-parameters

SOA record

- Defines the start of a new zone; and important parameters for the zone
- Always appears at the apex of the zone
- Serial number should be incremented on zone content updates

```
google.com.     86400 IN SOA ns1.google.com. (
                            dns-admin.google.com.
                             2012042000 ; serial number
                             7200       ; refresh (2 hours)
                             1800       ; retry (30 minutes)
                             1209600    ; expire (2 weeks)
                             300        ; minimum (5 minutes)
                             )
```

[DNS and DNSSEC, LOPSA PICC 12]

NS record

- Name Server record: owner is the zone name
- Delegates a DNS subtree from parent (ie. create new zone)
- Lists the authoritative servers for the zone
- Appears in both parent and child zones
- rdata contains hostname of the DNS server

```
upenn.edu.    86400 IN NS noc3.dccs.upenn.edu.
upenn.edu.    86400 IN NS noc2.dccs.upenn.edu.
upenn.edu.    86400 IN NS dns2.udel.edu.
upenn.edu.    86400 IN NS dns1.udel.edu.
upenn.edu.    86400 IN NS sns-pb.isc.org.
```

[DNS and DNSSEC, LOPSA PICC 12]

A record

- IPv4 Address Record
- rdata contains an IPv4 address

```
www.example.com.   86400   IN  A  192.0.43.10
```

AAAA record

- IPv6 Address Record
- rdata contains an IPv6 address
- Note: there was another record called A6, which didn’t catch on, and which has now been declared historic (RFC 6563)

```
www.example.com.  86400  IN  AAAA  2001:500:88:200::10
```

CNAME record

- An "alias", ie. maps one name to another (regardless of type)
- Put another way, "this is another name for this name"
- rdata contains the mapped domain name ("canonical name")
- CNAME records have special rules

```
www.example.com.  86400   IN CNAME  worf.example.com.
```

CNAME special rules

```
[from RFC 1034, Section 3.6.2]

>>> CNAME and no other data rule:
A CNAME RR identifies its owner name as an alias, and
specifies the corresponding canonical name in the RDATA section of the
RR.  If a CNAME RR is present at a node, no other data should be
present; this ensures that the data for a canonical name and its aliases
cannot be different.  This rule also insures that a cached CNAME can be
used without checking with an authoritative server for other RR types.

[Note: there is now an exception to this because of DNSSEC metadata records,
 which are allowed to appear with CNAMEs]

>>> CNAME special action processing:
CNAME RRs cause special action in DNS software.  When a name server
fails to find a desired RR in the resource set associated with the
domain name, it checks to see if the resource set consists of a CNAME
record with a matching class.  If so, the name server includes the CNAME
record in the response and restarts the query at the domain name
specified in the data field of the CNAME record.  The one exception to
this rule is that queries which match the CNAME type are not restarted.
```

CNAME special rules

```
Illustration of special action processing of CNAMEs:


$ dig www.sas.upenn.edu A

;; QUESTION SECTION:
;www.sas.upenn.edu.     IN A

;; ANSWER SECTION:
www.sas.upenn.edu.     300   IN   CNAME  virgo.sas.upenn.edu.
virgo.sas.upenn.edu.   900   IN   A      128.91.55.21
```

PTR record

- Pointer record
- The most common use is to map IP addresses back to domain names (reverse DNS mappings)
- IPv4 uses in-addr.arpa, and IPv6 uses ip6.arpa subtrees

IPv4 PTR records

- Uses “**in-addr.arpa**” subtree
- The LHS of the PTR record (“owner name”) is constructed by the following method:
  - Reverse all octets in the IPv4 address
  - Make each octet a DNS label
  - Append “in-addr.arpa.” to the domain name

IPv4 PTR example

```
host1.example.com.   IN   A    192.0.2.17

192.0.2.17                         (orig IPv4 address)

17.2.0.192                         (reverse octets)

17.2.0.192.in-addr.arpa.           (append in-addr.arpa.)
```

```
Resulting PTR record:

17.2.0.192.in-addr.arpa. IN PTR host1.example.com.
```

IPv6 addresses

- 128-bits (four times as large)
- 8 fields of 16 bits each (4 hex digits) separated by colons (:)
- [Hex digits are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f]
- 2^{128} possible addresses (an incomprehensibly large number)

```
2001:0db8:3902:00c2:0000:0000:0000:fe04
```

(2^{128} = 340,282,366,920,938,463, 463,374,607,431,768,211,456)

IPv6 addresses

- Zero suppression & compression for more compact format
  - Suppress (omit) leading zeros in each field
  - Replace consecutive fields of all zeros with a double colon (::) - only one sequence of zero fields can be compressed this way

```
2001:0db8:3902:00c2:0000:0000:0000:fe04
```

$\Downarrow$

```
2001:db8:3902:c2::fe04
```

IPv6 PTR records

- Uses "**ip6.arpa**" subtree
- The LHS of the PTR record ("owner name") is constructed by the following method:
  - Expand all the zeros in the IPv6 address
  - Reverse all the hex digits
  - Make each hex digit a DNS label
  - Append "ip6.arpa." to the domain name (note: the older "ip6.int" was formally deprecated in 2005, RFC 4159)

IPv6 PTR example

```
host1.example.com. IN AAAA 2001:db8:3902:7b2::fe04

2001:db8:3902:7b2::fe04                 (orig IPv6 address)

2001:0db8:3902:07b2:0000:0000:0000:fe04   (expand zeros)

20010db8390207b2000000000000fe04          (delete colons)

40ef0000000000002b7020938bd01002          (reverse digits)

4.0.e.f.0.0.0.0.0.0.0.0.0.0.0.0.2.b.7.0.2.0.9.3.8.b.d.
0.1.0.0.2                               (make DNS labels)

4.0.e.f.0.0.0.0.0.0.0.0.0.0.0.0.2.b.7.0.2.0.9.3.8.b.d.
0.1.0.0.2.ip6.arpa.                     (append ip6.arpa.)

4.0.e.f.0.0.0.0.0.0.0.0.0.0.0.0.2.b.7.0.2.0.9.3.8.b.d.
0.1.0.0.2.ip6.arpa. IN PTR host1.example.com.
```

MX record

- Mail Exchanger: defines the host receiving mail
- rdata consists of a preference field and the hostname of the mail receiver
- Lower preference = higher priority

```
example.com.    86400 IN MX  10 mail1.example.com.
example.com.    86400 IN MX  20 mail2.example.com.
```

preference mailserver name

SRV record

- Service Location record (RFC 2782)
- Allows designation of server(s) providing service for a particular application and transport at a domain name
- Owner name has special form: _service._transport.<domain>
- rdata contains priority, weight, port and server hostname
- Some applications using SRV records include: LDAP, Kerberos, XMPP, SIP, Windows AD, ...

SRV record

```
service name   transport                     priority  weight  port      server name

_ldap._tcp.example.com  600 IN SRV 1 0 389 ldap1.example.com
_ldap._tcp.example.com  600 IN SRV 2 1 389 ldap2.example.com
_ldap._tcp.example.com  600 IN SRV 2 2 389 ldap3.example.com
_ldap._tcp.example.com  600 IN SRV 2 1 289 ldap4.example.com
```

- Priority defines the order in which to query servers (lower number = higher priority)
- Weight defines the proportion in which to send queries to servers at the same priority level (load distribution)

TXT record

- free form descriptive text strings, with no defined semantics
- Although some applications have defined standardized meanings (eg. DKIM)
- rdata: one or more character strings

```
blah.example.com.   300  IN  TXT   "Hello World" "Goodbye"
```

NAPTR record

- Naming Authority Pointer Record (RFC 3403 - DDDS)
- Very complex record, and induces additional complex processing on resolver (lookup and rewrite)
- Uses: URL resolver discovery service, E164, SIP, ...

```
*.freenum       IN      NAPTR   (100 10 "u" "E2U+sip"
        "!^\\+*([^\\*]*)!sip:\\1@sip.magpi.org!" .)
```

Wildcards

- RRs with owner names starting with the label "*" (asterisk)
- When the wildcard is *matched*, the DNS server returns a response with:
    - query name returned as owner name
    - rest of RR content taken from the wildcard record

```
mail.example.com.     300  IN  A    10.1.1.1
www.example.com.      300  IN  A    10.1.1.2
*.example.com.        300  IN  A    10.1.1.7

Here, query for blah.example.com returns:
blah.example.com.     300  IN  A    10.1.1.7
```

ANY query type

- A pseudo record type used in DNS queries only
- Used to match any record type for the queried domain name
- Server will return all records of all types for that domain name that it possesses (note: caches may return incomplete data; to obtain all data for the name, you need to issue ANY query to authoritative servers)
- For debugging and troubleshooting purposes only; do not use in production code

Master Zone file format

- RFC 1035, Section 5 for details
- Entries in the master zone file are DNS resource records in their textual "presentation format"

Zone file example

```
Zone: example.com

@           3600 IN SOA master.example.com. hostmaster.example.com. (
                              1001514808 ; serial
                              10800      ; refresh (3 hours)
                              3600       ; retry (1 hour)
                              604800     ; expire (1 week)
                              3600       ; minimum (1 hour)
                              )
                86400   IN      NS      ns1.example.com.
                86400   IN      NS      ns2.example.com.
                86400   IN      MX      10 mail1.example.com.
                86400   IN      MX      20 mail2.example.com.
ns1             86400   IN      A       10.1.1.1
ns2             86400   IN      A       10.1.1.2
www               900   IN      A       10.1.2.2
mail1            3600   IN      A       10.3.3.3
mail2            3600   IN      A       10.3.3.4
```

[TCOM 504, Spring 2012]

Master Zone file format

```
@           Denotes current origin; defaulting to zone name
            Appended to any domain name not ending in a period.
()          Parens used to group data that crosses a line boundary
;           Starts a comment
$ORIGIN     Resets the origin for subsequent relative names

RRs beginning with whitespace implicitly inherit last owner name.
TTL and Class fields are optional (default to last explicitly stated)

Extensions usable in BIND master files:
$TTL        Define TTL parameter for subsequent records
$GENERATE   Programmatically generate records, eg.
            eg. $GENERATE 10-90 client-$ A 10.4.4.$
                $GENERATE 0-62 blah-${0,3,x}  A 192.168.154.${+64,0,d}
```

[DNS and DNSSEC, LOPSA PICC 12]

Size restrictions

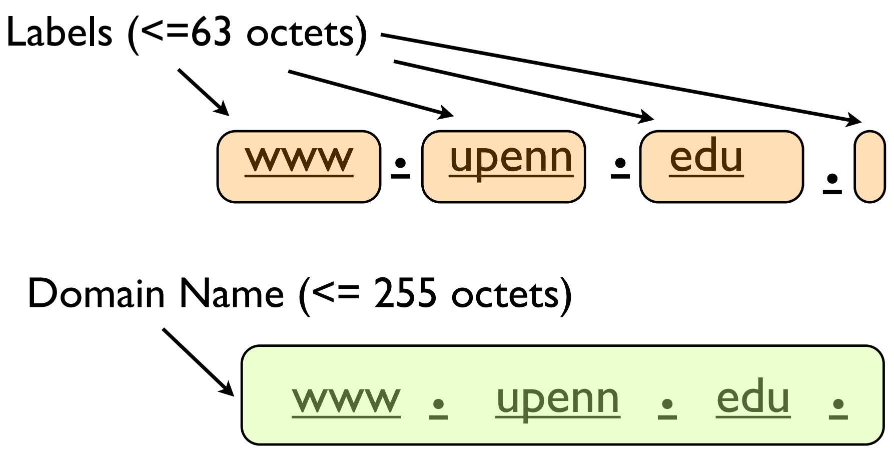
- Label: 63 octets max
- Domain Name: 255 octets max
- TTL: positive signed 32-bit integer
- Entire DNS message: 512 bytes (UDP) - plain DNS
- Messages larger than 512 bytes requires:
  - Use of TCP (often truncated UDP response followed by TCP retry)
  - EDNS0 - a DNS extension mechanism allowing negotiation of larger UDP message buffers

[DNS and DNSSEC, LOPSA PICC 12]

Textual vs wire format

- The human readable "textual representation" or "presentation format" of a domain name is different from the the domain name as it actually appears in DNS protocol messages ("on the wire" or "wire format")
- Text format: labels written in ASCII delimited by periods
- Wire format: label bytes one after the other, always ending with the empty label. each label is composed of a label length followed by the label bytes

[DNS and DNSSEC, LOPSA PICC 12]



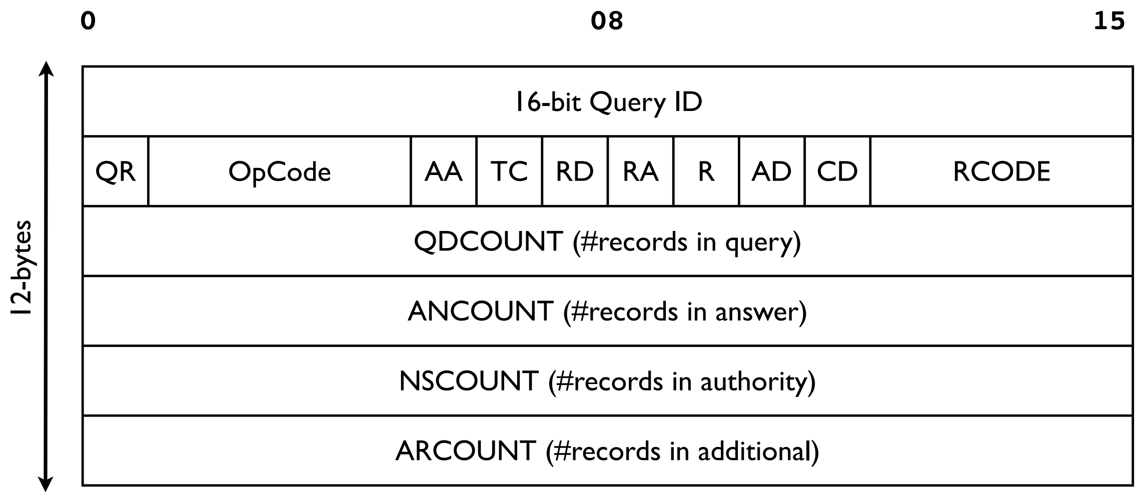
```
Wire format of this domain name:

(hex) 03777777057570656e6e0365647500

4 Component labels:
www      0x 03 77 77 77
upenn    0x 05 75 70 65 6e 6e
edu      0x 03 65 64 75
.        0x 00
```

label length in 1st octet (lower 6-bits)

DNS Packet Format

DNS Header (12 bytes)
Question Section
Answer Section
Authority Section
Additional Section

DNS Header



0
08
15
12-bytes
16-bit Query ID
QR
OpCode
AA
TC
RD
RA
R
AD
CD
RCODE
QDCOUNT (#records in query)
ANCOUNT (#records in answer)
NSCOUNT (#records in authority)
ARCOUNT (#records in additional)

DNS Header

QR: set to 1 in DNS response messages

OpCode:

0	Standard Query
1	Inverse Query (deprecated)
2	Status request (undefined and unused?)
4	Notify
5	Update
3,6-15	Undefined

AA = Authoritative answer (ie. not from cache)
TC = message was truncated (exceeded 512 byte UDP limit)
RD = Recursion desired
RA = Recursion available
R = Reserved/Unused
AD = Authenticated Data (DNSSEC)
CD = Checking Disabled (DNSSEC)

DNS Response Codes

Common Response codes:

0	NOERROR	No Error
1	FORMERR	Format Error
2	SERVFAIL	Server Failure
3	NXDOMAIN	Not existent domain name
4	NOTIMPL	Function not implemented
5	REFUSED	Query Refused, usually by policy

Used by DNS Dynamic Update (RFC 2136):

6	YXDomain	Name Exists when it should not
7	YXRRSet	RR Set Exists when it should not
8	NXRRSet	RR Set that should exist does not
9	NotAuth	Server not authoritative for zone
10	NotZone	Name not contained in zone
11-15	Unassigned	

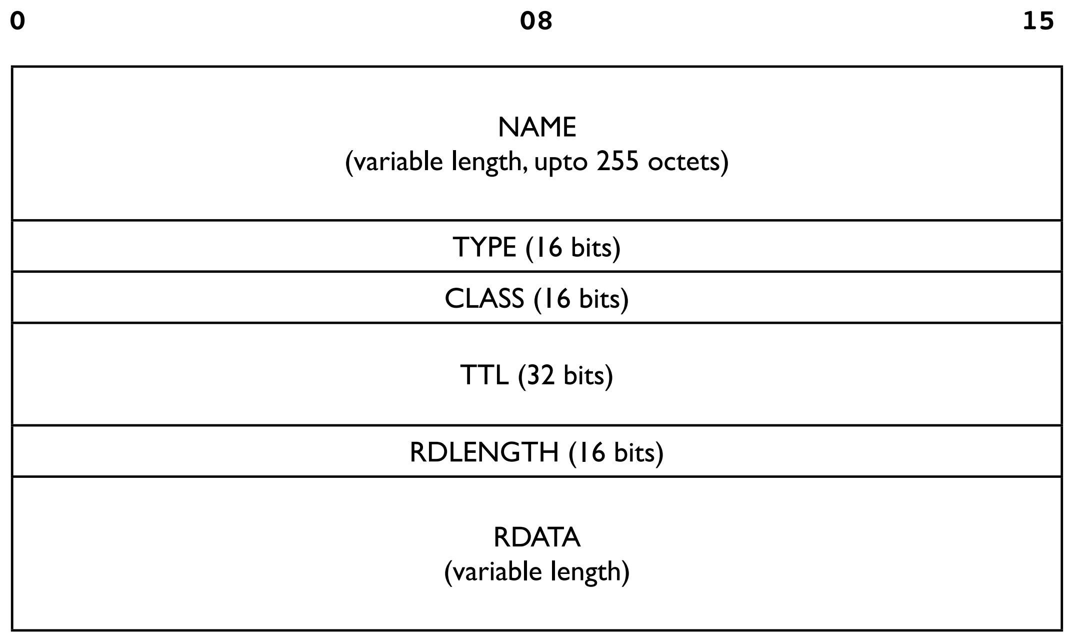
Extended RCodes

Extended RCODES do not appear in the DNS header (since there isn't enough space there). They instead appear in the OPT pseudo RR, which has a special format designed to accommodate them.

Extended RCodes used by EDNS0, TSIG, TKEY, etc:

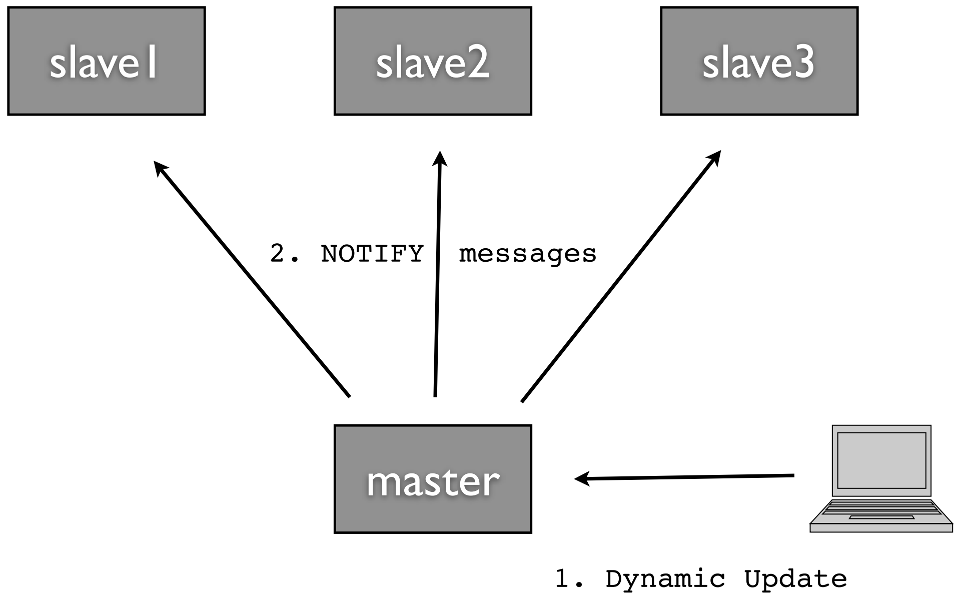
16	BADVERS	Bad OPT version
16	BADSIG	TSIG Signature Failure
17	BADKEY	Key not recognized
18	BADTIME	Signature out of time window
19	BADMODE	Bad TKEY Mode
20	BADNAME	Duplicate Key Name
21	BADALG	Algorithm not supported
22	BADTRUNK	Bad Truncation

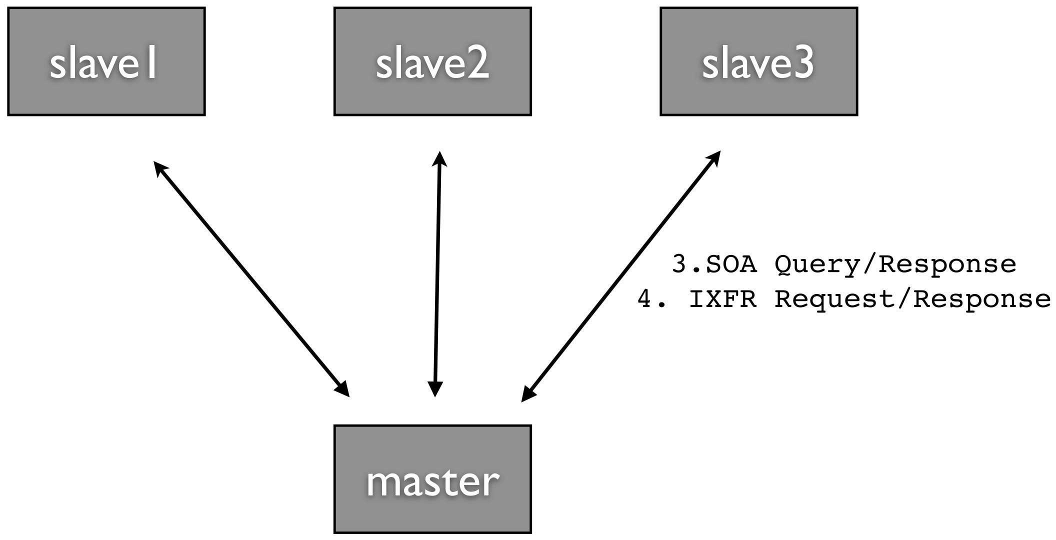
DNS RR common format



Zone Data Synchronization

- Authoritative server operators can synchronize zone data on their servers in a number of ways
- However, DNS provides a way to do this using the DNS protocol itself: **Zone Transfers**, and it's widely used
- Full zone transfers: **AXFR**: slaves send period transfer requests to masters (SOA refresh interval)
- Incremental zone transfers: **IXFR**, usually in combination with the NOTIFY mechanism (see RFC 1995 and 1996)
  - Commonly used in conjunction with Dynamic Update
- A good idea to authenticate zone transfers with TSIG


slave1
slave2
slave3
2. NOTIFY messages
master
1. Dynamic Update


slave1
slave2
slave3
3.SOA Query/Response
4. IXFR Request/Response
master

Zone Delegation

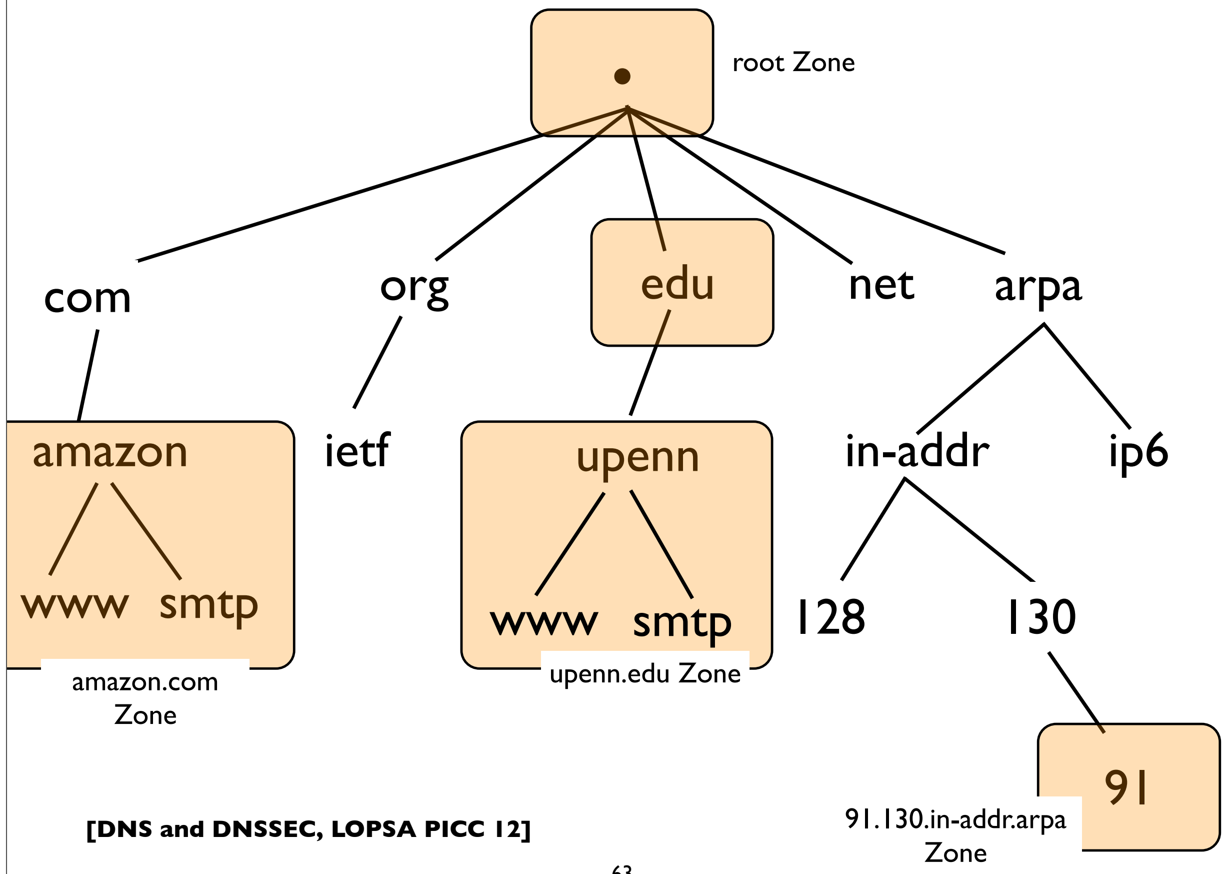
- Decentralized administration of DNS subtrees
- Delegations cause new zones to be created, that are (typically) served by different servers, run by different people
- Boundaries between zones (sometimes called zone cuts)
- An NS record set is needed in both the parent and child zones; these indicate the delegation, and the set of new nameservers involved in serving the child zone
- "Glue records" may be needed in the parent zone in order to find the addresses of the servers

Zone Delegation

Example of delegation of google.com in .com zone:

```
;; NS Record Set for google
google.com.     172800  IN NS  ns2.google.com.
google.com.     172800  IN NS  ns1.google.com.
google.com.     172800  IN NS  ns3.google.com.
google.com.     172800  IN NS  ns4.google.com.

;; Glue records for google nameservers
ns2.google.com.    172800 IN  A   216.239.34.10
ns1.google.com.    172800 IN  A   216.239.32.10
ns3.google.com.    172800 IN  A   216.239.36.10
ns4.google.com.    172800 IN  A   216.239.38.10
```

The glue records in the .COM zone are needed because the google DNS servers are inside the child google.com zone, otherwise they couldn't be found.



Configuring BIND

Simple zone file

Zone: example.com

```
$TTL 6h
@ IN SOA master.example.com. hostmaster.example.com.(
                     1001       ; Serial
                     10800      ; Refresh (3h)
                     3600       ; Retry (1h)
                     604800     ; Expire (1w)
                     3600 )     ; Min/ncache (1h)
;
             IN NS  ns1.example.com.
             IN NS  ns2.example.com.
             IN MX  10 mail.example.com.
;
ns1          IN A   192.168.1.1
ns2          IN A   192.168.2.2
www          IN A   192.168.4.4
mail         IN A   192.168.5.1
smtp         IN CNAME mail.example.com.
```

Recursive Resolver

```
# named.conf Recursive resolver example

acl trusted {
        192.0.2.0/24;           # my clients IPv4 address block
        2001:db8:f470::/48;     # my clients IPv6 address block
}

options {
        max-cache-size 1024M;
        listen-on-v6 { any; };
        allow-query-cache {
                trusted;
        };
        allow-recursion {
                trusted;
        };
};

zone "." {
        type hint;
        file "named.root";
};
```

who's allowed to use the recursive resolver.
note: some people run open servers

root nameserver addresses. latest version at
www.internic.net/domain/named.root

Authoritative Server

The master (primary master) authoritative server should define an access control list to limit the servers (usually only its slave servers) which can perform zone transfers of the DNS database. Note however, that this is a policy decision. Some folks allow anyone to transfer the contents of their zone.

```
# List of authorized secondary/slave servers
acl transferlist {
        192.0.2.2/32;
        192.0.2.3/32;
        2001:db8:f470:1234:2/128;
        2001:db8:f470:1234:3/128;
}

options {
        [...]
        allow-transfer {
                transferlist;
        };
       [...]
};
```

[DNS and DNSSEC, LOPSA PICC 12]

Authoritative Server

Authoritative Servers, need **zone definitions** for the zones they are serving. They should also disable recursion if not also providing recursive resolver service to endusers.

```
options {
        [ ... various options ...];
        recursion no;                  <-------------------  if authoritative only
};

zone "example.com" {
        type master;                   <-------------------  on master server
        file "zone.example.com";
};


zone "example.com" {
        type slave;                     <-------------------  on slave server
        file "zone.example.com";
        masters { 10.2.2.2; };
};
```

[DNS and DNSSEC, LOPSA PICC 12]

Reverse zones

```
# Reverse zone for 10.74.213.0/24
zone "213.74.10.in-addr.arpa" {
        type master;
        file "zone.213.74.10.in-addr.arpa";
};
```

(Delegation on non-octet boundaries is a bit trickier. Need zones per address or the trick described in RFC 2317)

```
# Reverse zone for 2001:db8:c472::/48
zone "2.7.4.c.8.b.d.0.1.0.0.2.ip6.arpa" {
        type master;
        file "zone.2.7.4.c.8.b.d.0.1.0.0.2.ip6.arpa";
};
```

zone xfr with TSIG

Authenticating Zone Transfers with TSIG:

On primary master server:

Generate TSIG key with (example):

```
   $ dnssec-keygen -a HMAC-MD5 -b 128 -n HOST slave1.example.com.
```

File: zonetransfer.key:

```
key "slave1.example.com." {
        algorithm "hmac-md5";
        secret "xjlsjdlfdfhfhdfldfljdflsjdljsdlfjdlkf=";
};
```

secret key taken from K* files produced by dnssec-keygen

File: named.conf:

```
include "/usr/local/bind/zonetransfer.key"

options {
        [...]
        allow-transfer { key slave1.example.com.; };
        [...]
};
```

can also be used within individual zone stanzas

zone xfr with TSIG

```
Authenticating Zone Transfers with TSIG (continued):

On secondary (slave) server (use same key as configured on master):

File: named.conf:
include "/usr/local/bind/zonetransfer.key"

zone “example.com” {
        type slave;
        masters { 10.12.7.26 key slave1.example.com.;  };
        [...]
};

It is also possible to sign and authenticate all transactions with a master
server (not just AXFR/IXFR) with a “server” statement:

server 10.12.7.26 {
        keys { slave1.example.com.; };
};
```

rndc

```
rndc: Name Server Control Utility

  reload
  reload zone [class [view]]
  refresh zone [class [view]]
  retransfer zone [class [view]]
  freeze
  freeze zone [class [view]]
  thaw
  thaw zone [class [view]]
  sync [-clean]
  sync [-clean] zone [class [view]]
  notify zone [class [view]]
  reconfig
  stats
  dumpdb [-all|-cache|-zones] [view ...]
  stop
  halt
  flush
  flush [view]
  flushname name [view]
  flushtree name [view]
  status
```

More examples

[Placeholder: Show fuller examples online]

Additional details

- The BIND ARM (Administrator's Reference Manual)
- http://www.isc.org/software/bind/documentation
- For latest BIND version (9.9):
  - http://ftp.isc.org/isc/bind9/cur/9.9/doc/arm/Bv9ARM.html
- Essential reading for the BIND DNS operator

Live DNS queries with dig

In this section, we'll look at some live DNS queries with the "**dig**" tool, widely available on most UNIX/Linux platforms.

```
Common invocations:
dig <qname>
dig <qname> <qtype>
dig @server <qname> <qtype>
dig -x <ipaddress>
dig +trace <qname> <qtype>
```

DNSSEC Tutorial

DNSSEC at a glance

- “DNS Security Extensions”
- A system to verify the authenticity of DNS “data” using public key signatures
  - Specs: RFC 4033, 4034, 4035, 5155 (and more)
- Helps detect DNS spoofing, misdirection, cache poisoning ..
- Recall the “Kaminsky attack”
- Additional benefits:
  - Ability to store and use cryptographic keying material in the DNS, eg. SSHFP, IPSECKEY, CERT, DKIM, TLSA, etc ..

DNSSEC at a glance

- Each zone has a public and private key pair
- The zone owner uses the private key to sign the zone data, producing digital signatures for each resource record set
- Public key is used by others (DNS resolvers) to validate the signatures (proof of authenticity)
- Public key is published in the zone itself so that resolvers can find it
- Zone public keys are organized in a chain of trust following the normal DNS delegation path

[DNS and DNSSEC, LOPSA PICC 12]

DNSSEC Records

DNSKEY	Contains zone public key
RRSIG	Contains DNSSEC signature
NSEC	Points to next name in zone (used for authenticated denial of existence)
DS	Delegation Signer (certifies public key for subordinate zone)
NSEC3	Enhanced version of NSEC (provides zone enumeration protection and opt-out)
NSEC3PARAM	NSEC3 parameters

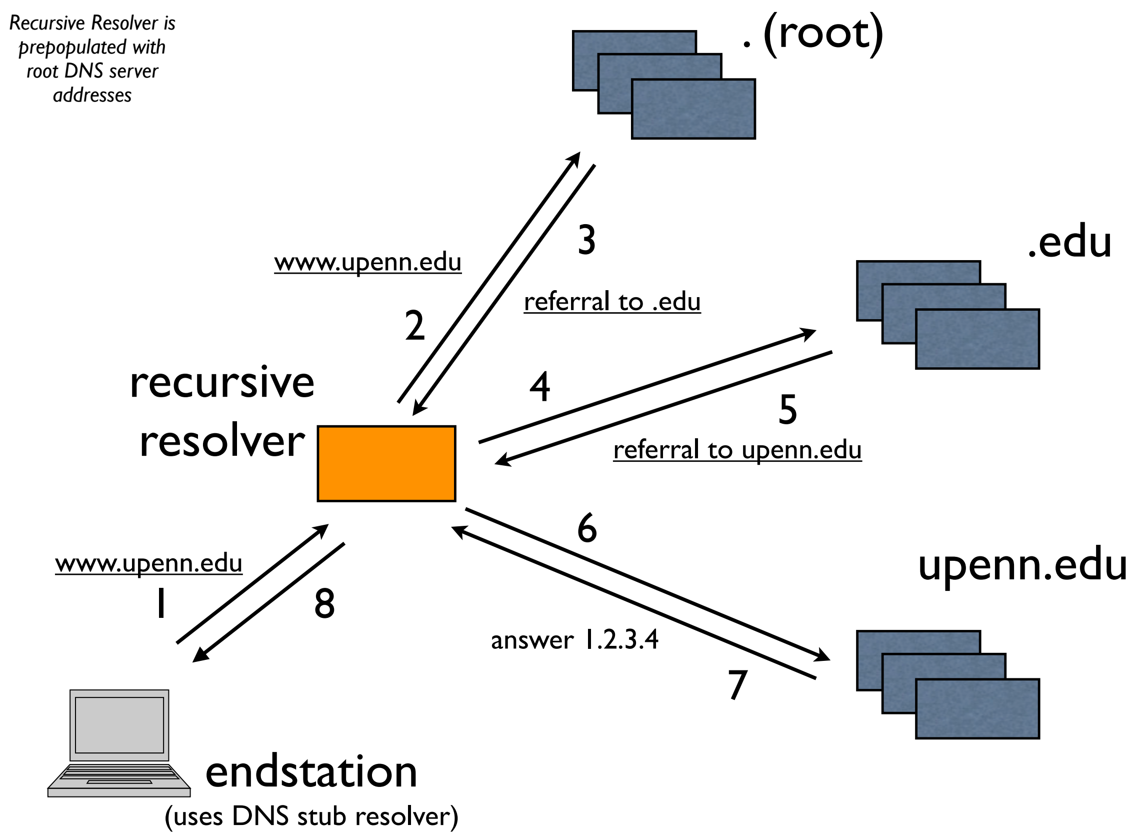
[DNS and DNSSEC, LOPSA PICC 12]

Signed zone additions

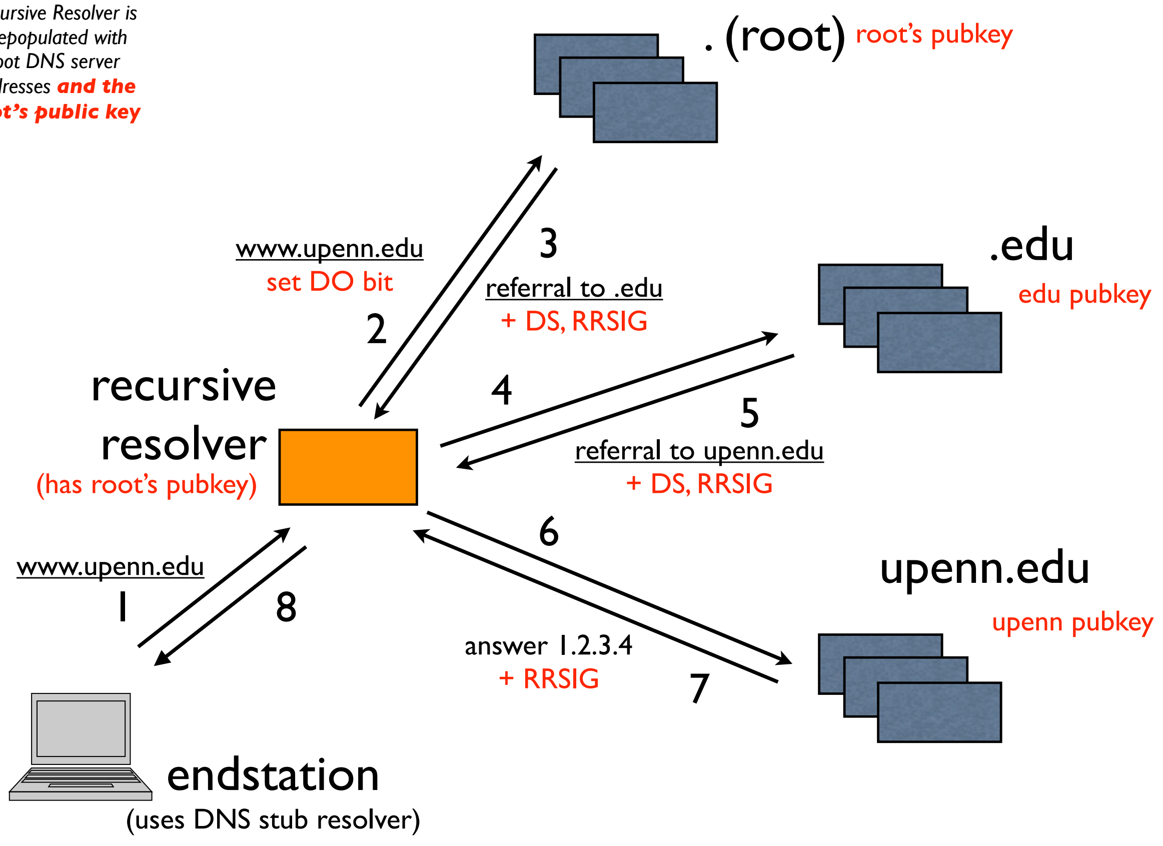
- One or more DNSKEY at the zone apex
- One or more NSEC for every DNS name
- One or more RRSIG for every RR set
- One or more DS records for every secure delegation

- Exceptions: non-authoritative data like delegation NS records and glue have no signatures (RRSIG)

[DNS and DNSSEC, LOPSA PICC 12]



Recursive Resolver is prepopulated with root DNS server addresses ***and the root's public key***



Multiple DNSKEYs

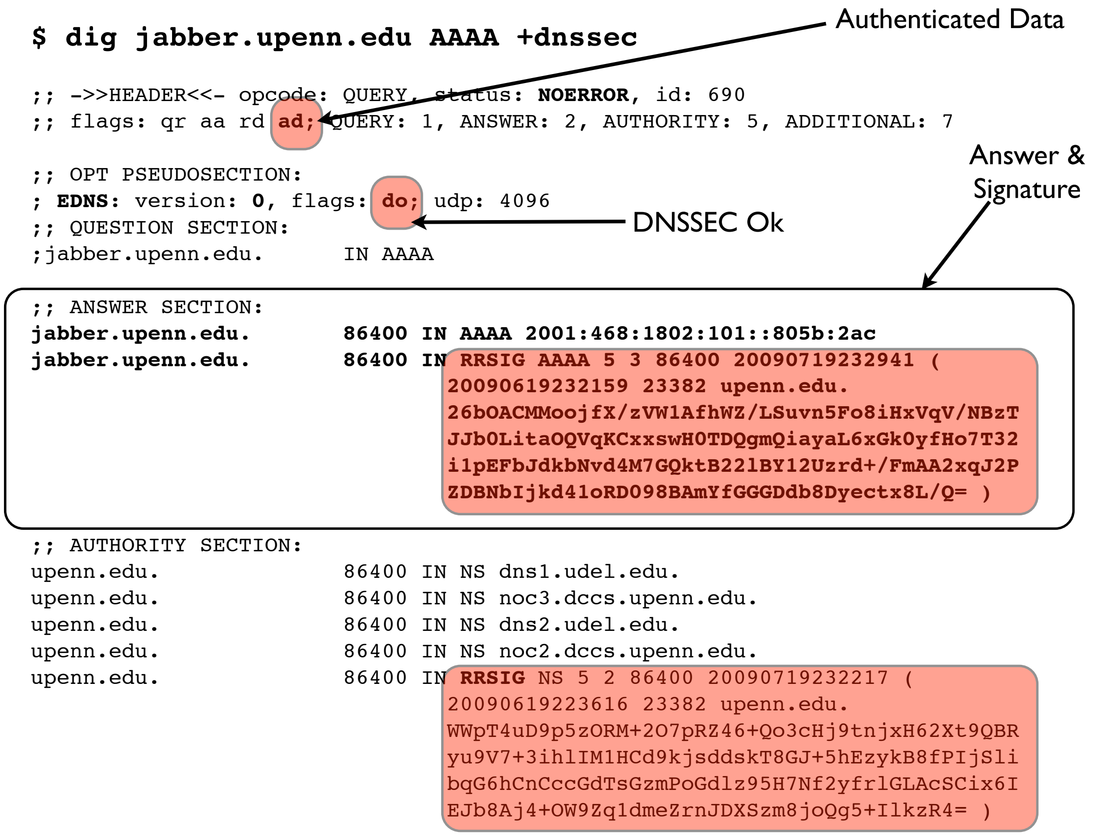
- Typically, a 2-level hierarchy of DNSKEYs is employed
- KSK: Key Signing Key
    - Signs other keys (can be larger, ie. stronger, and kept offline; used as the trust anchor and certified by the parent zone in the DS)
- ZSK: Zone Signing Key
    - Signs all data in the zone (can be lower strength and impose less computational overhead; can be changed without co-ordination with parent zone)

[DNS and DNSSEC, LOPSA PICC 12]

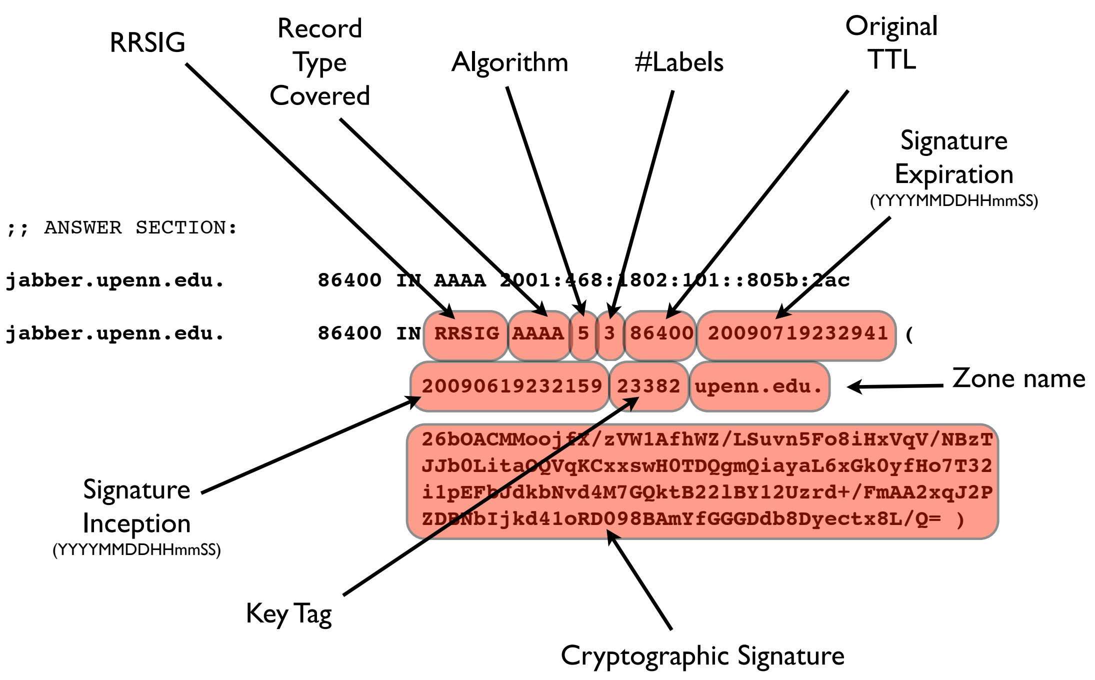
Protection of signing keys

- Keep offline? Problems with dynamic signing
- Keep only KSK offline? But need to bring them online for key rollovers (even only ZSK rollovers)
- If keeping online, lock down housing server rigorously, as you might do a critical authentication server, like a KDC
- Physically secured machine room & racks
- Tamper resistant HSM (Hardware Security Module)

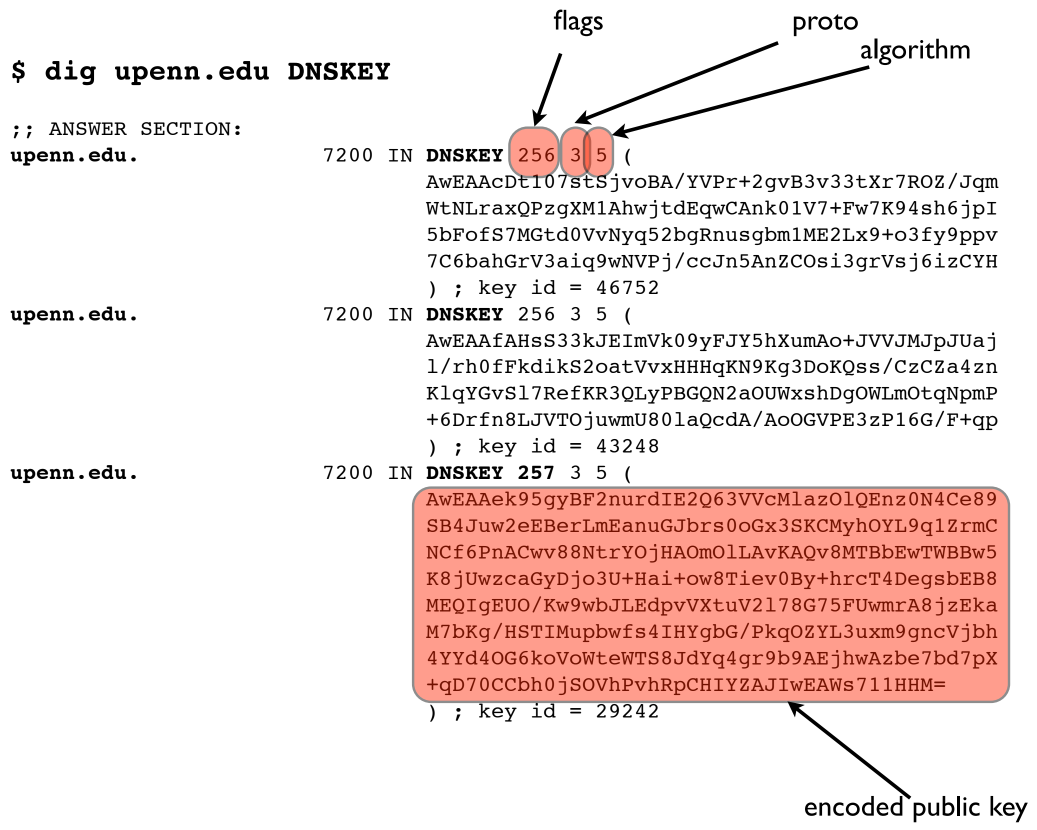
```
$ dig jabber.upenn.edu AAAA

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 337

;; QUESTION SECTION:
;jabber.upenn.edu.              IN      AAAA

;; ANSWER SECTION:
jabber.upenn.edu.       86400   IN      AAAA  2001:468:1802:101::805b:2ac

;; AUTHORITY SECTION:
upenn.edu.              86400   IN      NS      dns2.udel.edu.
upenn.edu.              86400   IN      NS      noc2.dccs.upenn.edu.
upenn.edu.              86400   IN      NS      noc3.dccs.upenn.edu.
upenn.edu.              86400   IN      NS      dns1.udel.edu.

;; ADDITIONAL SECTION:
noc2.dccs.upenn.edu.    86400   IN      A       128.91.254.1
noc2.dccs.upenn.edu.    86400   IN      AAAA    2001:468:1802:102::805b:fe01
noc3.dccs.upenn.edu.    86400   IN      A       128.91.251.158
dns1.udel.edu.          86400   IN      A       128.175.13.16
dns2.udel.edu.          86400   IN      A       128.175.13.17
```



```
$ dig jabber.upenn.edu AAAA +dnssec

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 690
;; flags: qr aa rd ad; QUERY: 1, ANSWER: 2, AUTHORITY: 5, ADDITIONAL: 7

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;jabber.upenn.edu.           IN AAAA

;; ANSWER SECTION:
jabber.upenn.edu.       86400 IN AAAA 2001:468:1802:101::805b:2ac
jabber.upenn.edu.       86400 IN RRSIG AAAA 5 3 86400 20090719232941 (
                                20090619232159 23382 upenn.edu.
                                26bOACMMoojfX/zVW1AfhWZ/LSuvn5Fo8iHxVqV/NBzT
                                JJb0LitaOQVqKCxxswH0TDQgmQiayaL6xGk0yfHo7T32
                                i1pEFbJdkbNvd4M7GQktB22lBY12Uzrd+/FmAA2xqJ2P
                                ZDBNbIjkd41oRD098BAmYfGGGDdb8Dyectx8L/Q= )

;; AUTHORITY SECTION:
upenn.edu.              86400 IN NS dns1.udel.edu.
upenn.edu.              86400 IN NS noc3.dccs.upenn.edu.
upenn.edu.              86400 IN NS dns2.udel.edu.
upenn.edu.              86400 IN NS noc2.dccs.upenn.edu.
upenn.edu.              86400 IN RRSIG NS 5 2 86400 20090719232217 (
                                20090619223616 23382 upenn.edu.
                                WWpT4uD9p5zORM+2O7pRZ46+Qo3cHj9tnjxH62Xt9QBR
                                yu9V7+3ihlIM1HCd9kjsddskT8GJ+5hEzykB8fPIjSli
                                bqG6hCnCccGdTsGzmPoGdlz95H7Nf2yfrlGLAcSCix6I
                                EJb8Aj4+OW9Zq1dmeZrnJDXSzm8joQg5+IlkzR4= )
```



```
;; ANSWER SECTION:

jabber.upenn.edu.       86400 IN AAAA 2001:468:1802:101::805b:2ac

jabber.upenn.edu.       86400 IN RRSIG AAAA 5 3 86400 20090719232941 (
                               20090619232159 23382 upenn.edu.
                               26bOACMMoojfX/zVW1AfhWZ/LSuvn5Fo8iHxVqV/NBzT
                               JJb0LitaOQVqKCxxswH0TDQgmQiayaL6xGk0yfHo7T32
                               i1pEFbJdkbNvd4M7GQktB22lBY12Uzrd+/FmAA2xqJ2P
                               ZDBNbIjkd41oRD098BAmYfGGGDdb8Dyectx8L/Q= )
```



```
$ dig upenn.edu DNSKEY

;; ANSWER SECTION:
upenn.edu.              7200 IN DNSKEY 256 3 5 (
                                AwEAAcDt107stSjvoBA/YVPr+2gvB3v33tXr7ROZ/Jqm
                                WtNLraxQPzgXM1AhwjtdEqwCAnk01V7+Fw7K94sh6jpI
                                5bFofS7MGtd0VvNyq52bgRnusgbm1ME2Lx9+o3fy9ppv
                                7C6bahGrV3aiq9wNVPj/ccJn5AnZCOsi3grVsj6izCYH
                                ) ; key id = 46752
upenn.edu.              7200 IN DNSKEY 256 3 5 (
                                AwEAAfAHsS33kJEImVk09yFJY5hXumAo+JVVJMJpJUaj
                                l/rh0fFkdikS2oatVvxHHHqKN9Kg3DoKQss/CzCZa4zn
                                KlqYGvSl7RefKR3QLyPBGQN2aOUWxshDgOWLmOtqNpmP
                                +6Drfn8LJVTOjuwmU80laQcdA/AoOGVPE3zP16G/F+qp
                                ) ; key id = 43248
upenn.edu.              7200 IN DNSKEY 257 3 5 (
                                AwEAAek95gyBF2nurdIE2Q63VVcMlazOlQEnz0N4Ce89
                                SB4Juw2eEBerLmEanuGJbrs0oGx3SKCMyhOYL9q1ZrmC
                                NCf6PnACwv88NtrYOjHAOmOlLAvKAQv8MTBbEwTWBBw5
                                K8jUwzcaGyDjo3U+Hai+ow8Tiev0By+hrcT4DegsbEB8
                                MEQIgEUO/Kw9wbJLEdpvVXtuV2l78G75FUwmrA8jzEka
                                M7bKg/HSTIMupbwfs4IHYgbG/PkqOZYL3uxm9gncVjbh
                                4YYd4OG6koVoWteWTS8JdYq4gr9b9AEjhwAzbe7bd7pX
                                +qD70CCbh0jSOVhPvhRpCHIYZAJIwEAWs711HHM=
                                ) ; key id = 29242
```

Negative answers

- "Authenticated Denial of Existence"
- NSEC or NSEC3 records (and their signatures)
- Chain together DNS records in a zone; can think of them and their signatures as spanning the gaps between names in the zone
- Canonical ordering of names in signed zones needed (RFC 4034, Section 6.1)
- Needed because of the pre-computed signature model of DNSSEC (computational concerns & signing key security)

[DNS and DNSSEC, LOPSA PICC 12]

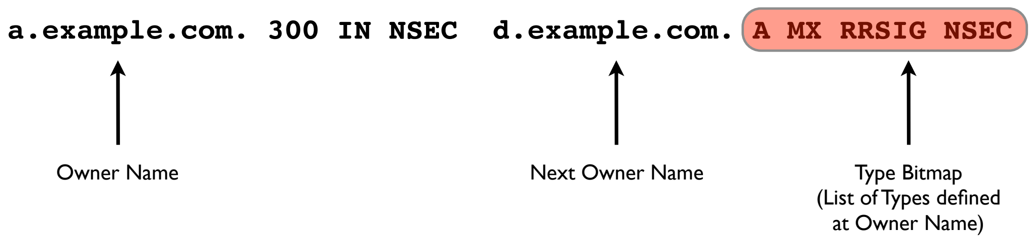
NSEC3 differences

- NSEC3 instead of NSEC records
- Owner name is a cryptographic hash of the name (flattened) rather than the actual name - <u>provides zone enumeration defense</u>
- Some names may not have an NSEC3 (the "opt-out" feature)
- Additional apex record: NSEC3PARAM
- Increased CPU usage implications
- See RFC 5155 (Hashed Authenticated Denial of Existence) for details

[DNS and DNSSEC, LOPSA PICC 12]

NSEC record

- "Next Secure" record
- Describes interval between consecutive names in a zone
- Type-bitmap defines RRtypes available at owner
- Side Effect: allows enumeration of zone contents



```
a.example.com. 300 IN NSEC  d.example.com. A MX RRSIG NSEC
```

[DNS and DNSSEC, LOPSA PICC 12]

An authenticated negative answer

```
$ dig +dnssec +multi bozo.upenn.edu

;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 7708

;; ;; AUTHORITY SECTION:
upenn.edu.          3600 IN   SOA assailants.net.isc.upenn.edu. hostmaster.upenn.edu. (
                    1002349189 ; serial
                    10800      ; refresh (3 hours)
                    3600       ; retry (1 hour)
                    604800     ; expire (1 week)
                    3600       ; minimum (1 hour)
                    )
upenn.edu.          3600 IN   RRSIG SOA 5 2 3600 (
                    20120527220138 20120427210138 50475 upenn.edu.
                    rlYYgnl1J0hOVItgX616TJQ5aqG7Se5zoacp+k5neiIC
                    rAKXKWDY5uVRCaJzci4kmYnHNksR0Z+Dtn2HJBB51gaG
                    JksqIp3PqvKZcX75UfpER4TjMYfmawhykESnR29jfm7w
                    k9onXJkYkVDVaimGFakGlAmuWQlvHBLQFSjnwVQ= )
upenn.edu.          3600 IN   NSEC _kerberos.upenn.edu. NS SOA MX RRSIG NSEC DNSKEY TYPE65534
upenn.edu.          3600 IN   RRSIG NSEC 5 2 3600 (
                    20120508051318 20120408042226 50475 upenn.edu.
                    ZzTYjeHECy5xLo+wrXq1VwmtNI3Wz7cpNLBdg+3xM9ph
                    H9jOndAViCKwsDa4uLBYBcKss9qbbYjVtMp5w0lmVpwm
                    cwxYheAyEN+w2VPBhLZ9qjfib8Q6Lfi3r3lC8EDJciL0
                    1LSQwP2gyFx7V6VG08z11W6fuB6A/6/3/55xwW0= )
cagrid.bmif.upenn.edu.  3600 IN   NSEC BRYNMAWR-GW.upenn.edu. CNAME RRSIG NSEC
cagrid.bmif.upenn.edu.  3600 IN   RRSIG NSEC 5 4 3600 (
                    20120507190845 20120407181400 50475 upenn.edu.
                    yn4Au0Q4EViYu0LonWlWviTUn6kLYfyMMERajl2Jdaob
                    CYLfwnWMrXYPh6IZu03dYSkIRg7WEoyEGckk5J5Gudok
                    ikdFEEuuBjV4cdUCMp67lvUjCGVclFWnKhb5ni/FmieH
                    q7yFeztBt/IsKxtbcFSX0Isjt5mtNqt5is+UNpY= )
```

*.upenn.edu would have been between upenn.edu and _kerberos.upenn.edu

bozo.upenn.edu would have been between cagrid.bmif.upenn.edu & brynmawr-gw.upenn.edu

An authenticated negative answer

NOERROR (nodata) responses can be authenticated with one signed NSEC record, which just reports all available RRTYPEs at that name

```
$ dig +dnssec +multi upenn.edu

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44529

;; AUTHORITY SECTION:
upenn.edu.          3600 IN   SOA assailants.net.isc.upenn.edu. hostmaster.upenn.edu. (
                    1002349284 ; serial
                    10800      ; refresh (3 hours)
                    3600       ; retry (1 hour)
                    604800     ; expire (1 week)
                    3600       ; minimum (1 hour)
                    )
upenn.edu.          3600 IN   RRSIG SOA 5 2 3600 (
                    20120528003143 20120427233143 50475 upenn.edu.
                    lK7/1MnNo2ox+cAJAMbX+it0tEpJiQJ+DorWcYvPUTek
                    rr5efJ/yUSxK7bZ+LHqiBj4vSVT3EDnlu7Di7RlyxCAB
                    LfRUhbNb29ToCQB/HDWjyjtWKjY8rWehWUeXqcJ0PCP1
                    6J8FH3g41lxjcOLGom0opalOqtLxPEjqwtlm8BY= )
upenn.edu.          3600 IN   NSEC _kerberos.upenn.edu. NS SOA MX RRSIG NSEC DNSKEY TYPE65534
upenn.edu.          3600 IN   RRSIG NSEC 5 2 3600 (
                    20120508051318 20120408042226 50475 upenn.edu.
                    ZzTYjeHECy5xLo+wrXq1VwmtNI3Wz7cpNLBdg+3xM9ph
                    H9jOndAViCKwsDa4uLBYBcKss9qbbYjVtMp5w0lmVpwm
                    cwxYheAyEN+w2VPBhLZ9qjfib8Q6Lfi3r3lC8EDJciL0
                    1LSQwP2gyFx7V6VG08z11W6fuB6A/6/3/55xwW0= )
```

NSEC3 record

- New version of NSEC that provides defense against zone enumeration (see RFC 5155 for details)
- Hashed owner names, and optional “opt-out” feature
- rdata: nsec3 parameters (hash alg, flags, iterations), hashed next owner name, type bitmap

hashed owner

```
N3O3HKFAKV0FDU4OMHFEHT0FROFG6RQL.example.com. 3600 IN
NSEC3 1 0 5 9EBA4228 Q9T0VRM5S6EEF2N72RPCC5ENOF4IGV3O A
MX RRSIG
```

nsec3 params
hashalg, flags, iterations, salt

next hashed owner

NSEC3PARAM record

- NSEC3PARAM record at zone apex also holds the parameters
- Hash algorithm, Flags, #Iterations, Salt
- This is used by secondary nameservers slaving the zone

```
lsu.edu.     0   IN NSEC3PARAM 1 0 10 6F772A6B
```

An authenticated negative answer (NSEC3)

placeholder

Secure Delegations

- Indicated by DS (Delegation Signer) record
- Appears in the delegating (ie. parent) zone
- Contains a hash of the public key of the child zone's
- Validating resolvers use the presence of the DS record *and* its corresponding signature (RRSIG) to securely authenticate the delegation

DS contains hash of the public key of delegated domain. 2 DS records are shown here because 2 different hashing algorithms were used

```
magpi.net.      3587 IN      DS 15462 5 2 (
                             9EFD691150378921179A5408F04E6EA93CBA2488B221
                             96493142E47D1AD24C3A )
magpi.net.      3587 IN      DS 15462 5 1 (
                             C020FB9E09EE30568F250E2086D52E62F2B4FA17 )
magpi.net.      3587 IN      RRSIG DS 5 5 3600 20090812170009 (
                             20090713170009 64263 dlv.isc.org.
                             M+09bX9XP79yfDhWDUNuDEg9KOEHV2eV33/dEYnutVpD
                             iZYGqJ6BWLhWZYE8Y8megYozfa5UJv/AVcdIZ51JCPI4
                             k/jlRDj60kRaWRlfCBgqOR2WPL+F20vhg3wS57bIjmRW
                             To0r/HpXemnJVdXLbrzWD5WdpYGFy1UVX+15N4o= )
```

Signature of DS record set

DNSSEC Lookaside Validation (DLV)

- A mechanism to securely locate DNSSEC trust anchors "off path"
- Intended as an early deployment aid until top-down deployment of DNSSEC is completed
- DLV Registry operated by Internet Systems Consortium:
  - https://www.isc.org/solutions/dlv
- If you can't find a DS record for example.com, look for a DLV record for example.com.<dlv-domain>

Configuring DNSSEC in BIND

General advice

- Use the latest possible version of BIND (current is v9.9)
- Many more features that make DNSSEC configuration much much easier, and almost automated ...

Validating Resolver

In named.conf (this will use BIND’s built-in keys for the root and the ISC DLV registry, and will automatically rollover keys as they are detected:

```
options {
        [...]
        dnssec-enable yes;
        dnssec-validation auto;
        dnssec-lookaside auto;
        [...]
};
```

Validating Resolver

Manually configured keys (if needed):

```
# manually configured static key
trusted-keys {
   . 257 3 8 “AwE...jlsdjfld=”;
};

# managed keys (with automated rollover)
managed-keys {
    “.” initial-key 257 3 8 “Awlsdjflkdjfl”;
};
```

Signing zones

```
Generating Keys:

  **dnssec-keygen** <zone>
  dnssec-keygen -f KSK <zone>
  dnssec-keygen -3 <zone>          # NSEC3 zone

  Creates K<zone>+mmm+nnnn.key and K<zone>+mmm
+nnnn.private files

Signing Zone:

  dnssec-signzone -o zone -S <zonefile>

   -S: smart signing
```

Authoritative Server

```
options {
        [...]
        dnssec-enable yes;
        [...]
};
```

Dynamic Update + DNSSEC

```
The easiest way, in my opinion.

* Configure dynamic zones (ie. zones updated only
  with the Dynamic Update protocol, eg. with the
  nsupdate program)
* Make DNSSEC keys available to named
* When dynamic updates are made, named will
  automatically sign the records and generate or
  re-generate related DNSSEC metadata

* Latest BIND versions include special options to
  make this really easy.
```

Live example of signing a zone with DNSSEC
(Time permitting!)

Signing a zone

```
Steps for reference. We’ll do this live (hopefully)


# Create zone for “example.com” and configure named
[...]

# Generate KSK and ZSK (in this example RSASHA256 2048/1024bit)
dnssec-keygen -a RSASHA256 -b 2048 -n ZONE -f KSK example.com
dnssec-keygen -a RSASHA256 -b 1024 -n ZONE example.com

# Sign zone (will generate “zonefile.signed”)
dnssec-signzone -o example.com -S zonefile

# Reconfigure named.conf to serve “zonefile.signed”
[...]
```

Signing a zone (dynamic)

```
# Generate KSK and ZSK as before, but don’t use dnssec-signzone
[...]

# Setup named.conf with the “auto-dnssec” option for the zone
zone "example.com" {
     type master;
     update-policy local;                # allow-update for expl key
     auto-dnssec allow;                  # also see “maintain”
     file "zones/example.com/zonefile";
     key-directory "zones/example.com";
};

# Tell named to sign the zone
  rndc sign example.com

# From now, use dynamic update (eg. via nsupdate) to update
# zone contents.
```

Signing a zone (dynamic)

```
# Example of using dynamic update to add an ldap.example.com
# A RR to the zone .. This will cause name to automatically
# compute and add RRSIGs and NSEC/NSEC3s as needed.

$ nsupdate -l
ttl 86400
zone example.com.
update add ldap.example.com. A 10.4.4.4
send
^D
$
```

Other methods

Newest versions of BIND have some other ways that might make it easier to deploy DNSSEC in some environments where it's not easy to modify the master server ...

* **Inline Signing** (BIND 9.9)

This feature greatly simplifies the deployment of DNSSEC by allowing completely automatic, fully transparent signing of zones. Using the new 'inline-signing' option in a master server allows named to switch on DNSSEC in a zone without modifying the original zone file in any way. Using it in a slave server allows a zone to be signed even if it's served from a master database that doesn't support DNSSEC.

Some example configurations may be found at
https://kb.isc.org/article/AA-00626/0/Inline-Signing-in-ISC-BIND-9.9.0-Examples.html

Key Rollover

Key Rollover

- Conventional wisdom is that DNSSEC keys should be changed ("rolled over") at regular intervals. However, not everyone agrees, including some noted security experts
- If you choose strong enough keys, there is no cryptographic reason to routinely roll them
- There are good operational reasons to change keys *after specific events*, eg. turnover of a staff member who had access to the private keys, or a system compromise of the server
- Some argue routine key rollover instills practice & confidence that you'll be able to do it properly when you really need to. However, do we do this for other applications (Kerberos, PKI/CAs, SSL)?

Key Rollover

- RFC 4641: DNSSEC Operational Practices
  - Covers general practices, procedures, recommendations
  - Update: http://tools.ietf.org/html/draft-ietf-dnsop-rfc4641bis-11
- Most commonly used:
  - KSK rollover: double signature policy
  - ZSK rollover: pre-publish policy

KSK: Double signature

- Generate new KSK; publish (public part) in zone
- Sign DNSKEY RRset with both keys
- Publish additional DS record in parent for new key
- Wait until DS is propagated and TTL of the old DS record
- Remove the old KSK and re-sign DNSKEY RRset with only new key

ZSK: Pre-publish

- Generate new ZSK, and publish the DNSKEY in the zone, but do not yet sign zone data with it
- Wait zone propagation time + TTL of the DNSKEY RRset
- Use new ZSK for signing zone records instead of old ZSK, but leave the old ZSK published in the zone
- Wait zone propagation time + largest TTL of all records in the zone
- Remove old key & re-sign DNSKEY RRset

Other DNSSEC caveats

General DNSSEC Caveats

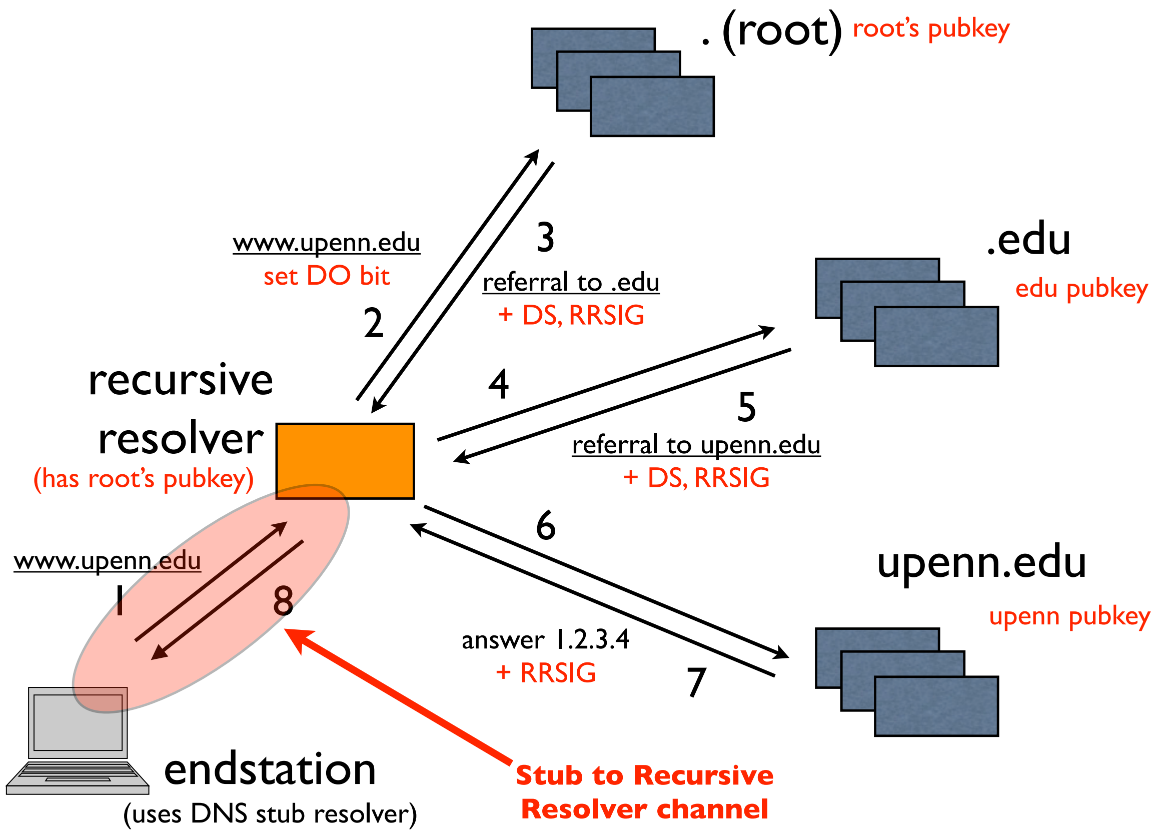
- Zone size increases significantly when signed
- Memory and CPU usage increase
- DNSSEC answers are larger
- Server side & query side impacts
- Interference by firewalls, proxies, and other middlebox, eg. botching EDNS0, large packets, DNSSEC meta data etc
- Fallback to TCP increases
- Many modern resolvers already ask for DNSSEC by default (ie. set the DNSSEC-OK bit in their queries)

[DNS and DNSSEC, LOPSA PICC 12]

Securing the last hop

- How do we protect the stub resolver?
- Employ a channel security mechanism between stub and the upstream recursive resolver:
  - TSIG, SIG(0), IPSEC, etc
- Have the stub validate DNSSEC responses? Set CD bit and authenticate signatures directly?
- Give up, and run a full service DNS Resolver on clients?

[DNS and DNSSEC, LOPSA PICC 12]



Channel Security

- For stub channel security, simple symmetric key TSIG won't work
- Can't distribute same TSIG key to many clients, because that allows any of them to forge answers to all others
- Need per client keys and thus a key management infrastructure
- GSS-TSIG has a chicken-egg problem, because DNS is often used to locate Kerberos servers
- SIG(0) may be better - distribute single public key to clients
- Microsoft supposedly has an implementation of IPsec (GSS authenticated) to protect client to recursive resolver path
- DNSCurve?

[DNS and DNSSEC, LOPSA PICC 12]

Application use of DNSSEC

Application use of DNSSEC

- One of the more exciting prospects for DNSSEC
- DNSSEC allows applications to securely obtain (authenticate) cryptographic keying material stored in the DNS
- A variety of existing and proposed record types have been designed to store crypto material:
  - SSHFP, IPSECKEY, CERT
  - DKIM _domainkey TXT record (p=... public key data)
  - TLSA (upcoming, see IETF DANE working group)

Application use of DNSSEC

- Securely obtaining other assertions from the DNS
  - DKIM/ADSP
  - Route Origination Authorizations (controversial - see RPKI, the standardized mechanism to do this, which will allow BGP path validation also)

SSHFP record

- SSH Host Key Fingerprint (RFC 4255)
- Allows you to validate SSH host keys using DNS (securely using DNSSEC)

algorithm number

fingerprint type (1= SHA-1)

```
grodd.magpi.net. 86400 IN SSHFP (1 1
      F60AE0994C0B02545D444F7996088E9EA7359CBA)
```

fingerprint

IPSECKEY record

- RFC 4025: method for storing IPSEC keying material in DNS
- rdata format: precedence, gateway-type, algorithm, gateway address, public key (base64 encoded)

```
38.2.0.192.in-addr.arpa. 7200 IN  IPSECKEY ( 10 1 2
     192.0.2.38
     AQNRU3mG7TVTO2BkR47usntb102uFJtugbo6BSGvgqt4AQ== )
```

Public CA model problems

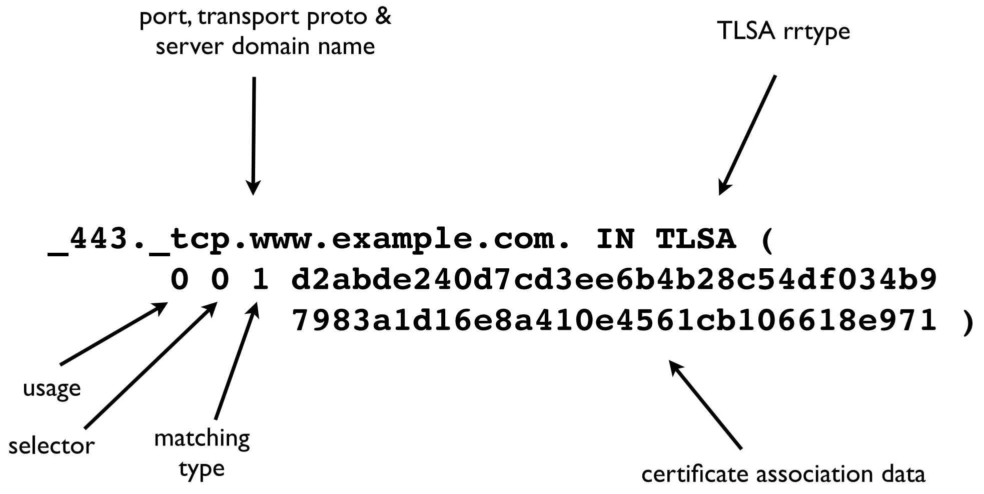
- Applications need to trust a large number of global certificate authorities, and this trust appears to be unfounded
- No namespace constraints! **Any** of them can issue certificates for **any** entity on the Internet, whether you have a business relationship with them or not
- Least common denominator security: our collective security is equivalent to weakest one
- Furthermore, many of them issue subordinate CA certificates to their customers, again with no naming constraints
- Most are incapable of issuing certs with any but the most basic capabilities (eg. alternate name forms or other extensions)

DANE/TLSA record

- The DNS-Based Authentication of Named Entities (DANE) Protocol for Transport Layer Security (TLS)
  - draft-ietf-dane-protocol-20 (almost published as RFC)
  - RR type code already assigned
- Use DNSSEC for better & more secure ways to authenticate SSL/TLS certificates:
  - by specifying authorized public CAs, allowable end entity certs, authorizing new non-public CAs, or even directly authenticating certs without involving CAs!

[DNS and DNSSEC, LOPSA PICC 12]

TLSA record example


port, transport proto & server domain name
TLSA rrtype
usage
selector
matching type
certificate association data

```
_443._tcp.www.example.com. IN TLSA (
      0 0 1 d2abde240d7cd3ee6b4b28c54df034b9
            7983a1d16e8a410e4561cb106618e971 )
```

[DNS and DNSSEC, LOPSA PICC 12]

TLSA rdata parameters

```
Usage field:
    0   CA Constraint
    1   Service Certificate Constraint
    2   Trust Anchor Assertion
    3   Domain Issued Certificate

Selector field:
    0   Match full certificate
    1   Match only SubjectPublicKeyInfo

Matching type field:
    0   Exact match on selected content
    1   SHA-256 hash of selected content
    2   SHA-512 hash of selected content

Certificate Association Data: raw cert data in hex
```

TLSA record example

Usage type 1: Service certificate constraint; match an end-entity certificate

```
_443._tcp.www.example.com. IN TLSA (
       1 1 2 92003ba34942dc74152e2f2c408d29ec
             a5a520e7f2e06bb944f4dca346baf63c
             1b177615d466f6c4b71c216a50292bd5
             8c9ebdd2f74e38fe51ffd48c43326cbc )
```

TLSA record example

Usage type 3: Full certificate association (No CA required)

```
_443._tcp.www.example.com. IN TLSA (
      3 0 0 30820307308201efa003020102020... )
```

DNSSEC Deployment Status

Deployment status

- DNSSEC Root signed (July 2010)
- Many TLDs signed (94 of 313 as of Apr 2012, and others planning):
  - GTLD: edu gov com net org biz info arpa
  - ccTLD: ac ag at bg bg br bz ch cl co cr cz de dk eu fi fr gi gl gr hn in io jp kr la lc li lk lu me mm mn my na nc nl nu nz pl pm pr re sc se sh si su sx tf th tm tw ug uk us wf yt (58 at last count, plus some IDNcctlds)
  - See http://stats.research.icann.org/dns/tld_report/
- Reverse trees: in-addr.arpa ip6.arpa
- Note: not all TLD registrars support DNSSEC yet (ie. ability to install a DS record in the TLD)

SecSpider

- DNSSEC zone monitoring project
- http://secspider.cs.ucla.edu/
- Over 37,000 signed zones as of mid April 2012
- Crawling and user submissions
- Distributed polling
- Also a DLV registry

DNSSEC Tools

Some useful tools

- Checking correct operation/deployment:
  - DNSviz: http://dnsviz.net/
  - http://dnssec-debugger.verisignlabs.com/
  - http://dnscheck.iis.se/
- 3rd party tools that some folks use to deploy/manage DNSSEC with BIND (mostly everything can be done in BIND itself these days):
  - OpenDNSSEC
  - zkt

Thank you!

twitter
@shuque

Shumon Huque
shuque -@- upenn.edu

Please fill out the Trainer Evaluation
http://www.picconf.org/training-survey

Rate PICC'12
http://www.picconf.org/rate-picc-12