Introduction to DNSSEC

ARIN Tutorial

April 1, 2001

Edward Lewis

lewis@tislabs.com

PGP

SECURITY

PROTECTING YOUR PRIVACY

Agenda

- Overall Description
- The easy features
- The complicated features
- The remaining issues

Features of DNSSEC

- Provides protection of host to name server communication
  - remote control, zone transfers, query/response
- Provides server to server protections (*zone*)
  - authoritative-ness can be proven
- Provides means to distribute certificates
  - Not a PKI, but a tool that can be used by a PKI
- Provides a way to secure dynamic update

Components of DNSSEC

- TSIG, SIG(0), and TKEY
  - Close-quarters, shared secret security for messages
- SIG, KEY and NXT
  - Scaleable digital signature protection of data
- CERT
  - Holder of certificate (PGP, X.509) data
- Secure Dynamic Update
  - Uses message security to identify the requestor

Some basics

- Technology Status
- Terminology
- How it fits together

Protocol and Software Status

- Protocol specified in a collection of IETF RFCs
  - First of three levels of standardization
  - Rewrites of major documents to happen

- ISC's BIND software implements most of DNSSEC
  - Still in "bleeding edge" state
- Microsoft and Lucent are implementing parts
  - Software hasn't been distributed yet

IETF Working Groups

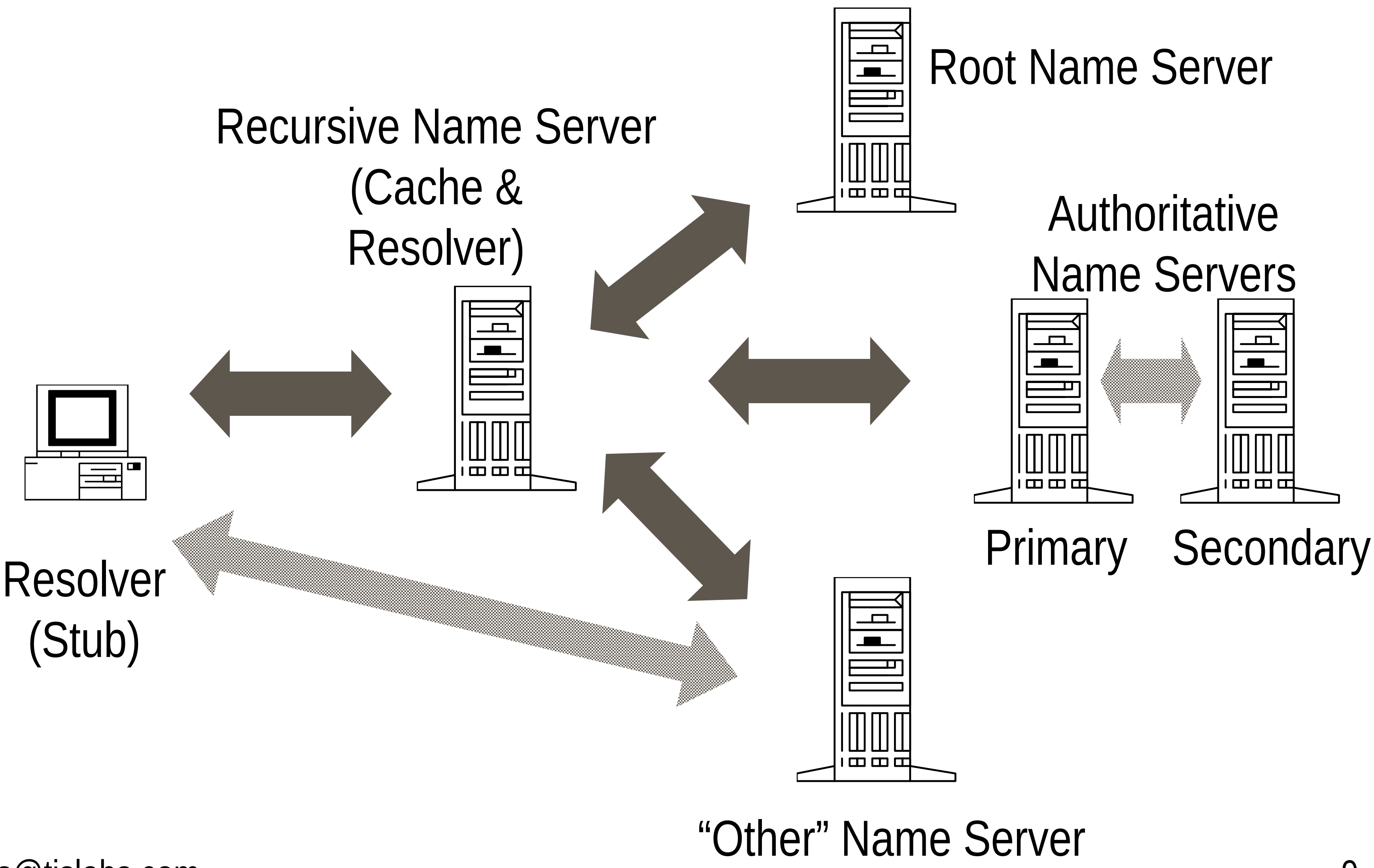
- Work is progressing to refine protocol
  - IETF WG on DNS Extensions (DNSEXT)
  - Much work remains to progress to "Full Standard"
  - Internet Drafts document the work in progress

- Operational experience is limited but growing
  - IETF WG on DNS Operations (DNSOP)
  - Many DNSSEC workshops have been held
  - “How to operate” and “policy” questions abound

Deployment Plans

- A major push is in Europe
  - Three ccTLD's plan to have signed zones as soon as possible
  - CENTR has a DNSSEC WG in action
- Root Servers
  - Looking into adoption, sooner rather than later
  - Recommended the adoption of TSIG
- Other recent activity - ENUM, Asian TLD's

Some Terminology



Resource Record "Sets"

- <owner> <ttl> <class> <type> <rdatalen> <rdata>
  - myname.xy. 14400 IN A 123.123.123.123
  - myname.xy. 14400 IN A 203.123.245.123
- In old DNS
  - Records with common owner, type, class are treated together, but still are singular entities
- For DNSSEC
  - The RR set is formalized
  - No longer are records singular, always treated as a set

✉So, I will be talking about “sets” of data

Zones vs. Servers

- Zone is an administrative cut of the name space
- Name server is a host dispensing information
- Relationship
  - A zone is served by name servers (1 or more)
  - A name server may serve many zones (0 or more)
  - Authoritative servers have the original zone data
  - Primary master server has the data in a source text file

☒SIG/KEY secures on the basis of zones

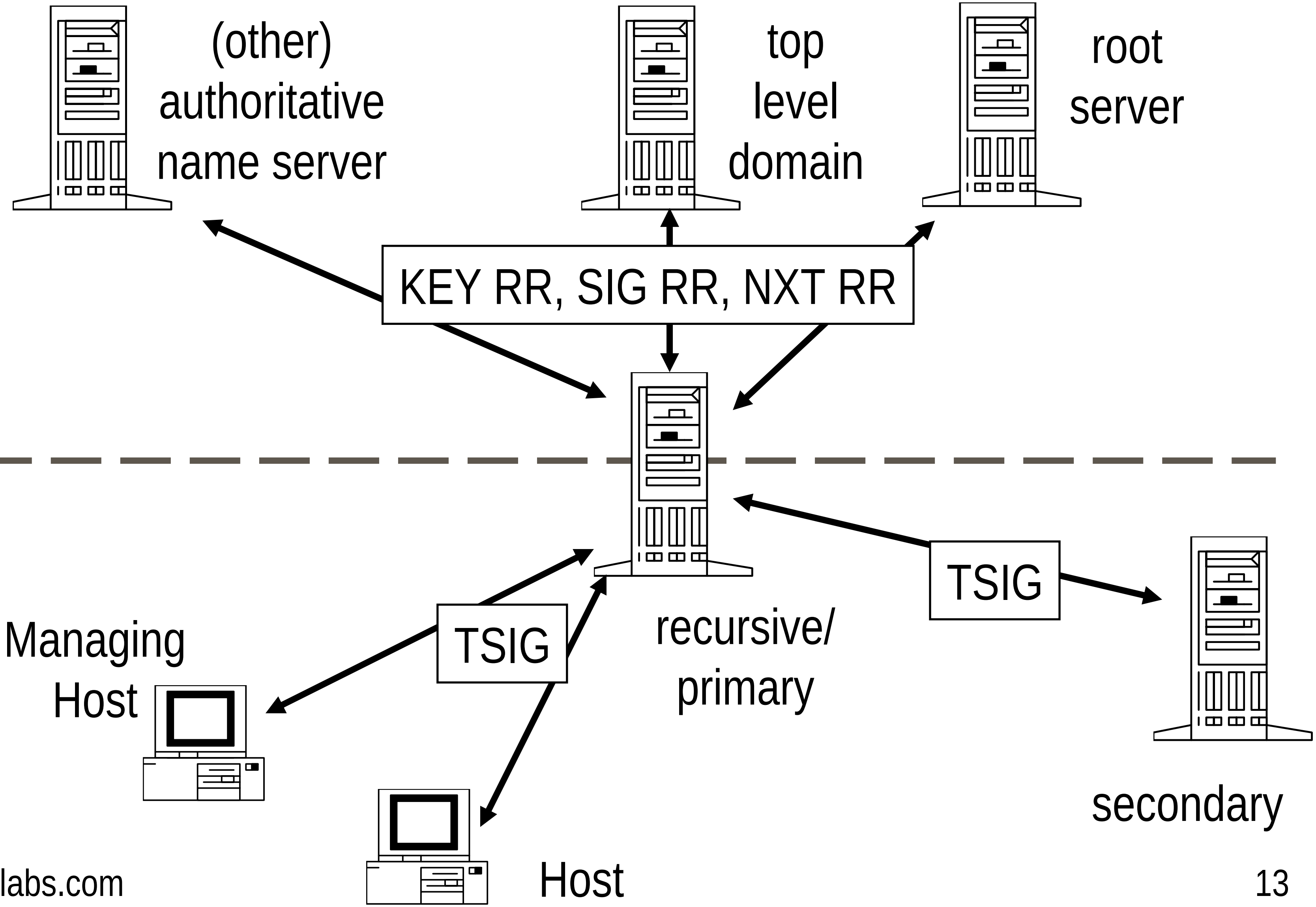
☒Query/Response secures between a resolver and a server

  - Or, in the case of zone transfers, between two servers

Cryptography

- Symmetric keys (aka shared secret)
  - One key, encrypts and decrypts/signs and verifies
  - Problem: distributing the secret secretly, storing the secret secretly
- Asymmetric keys (aka public key)
  - Pair of keys, one encrypts/signs, other decrypts/verifies
  - Problem: slower than symmetric
- Optimization
  - Use asymmetric keys to agree upon a symmetric key
- Other issues: patents and export control

How this fits together



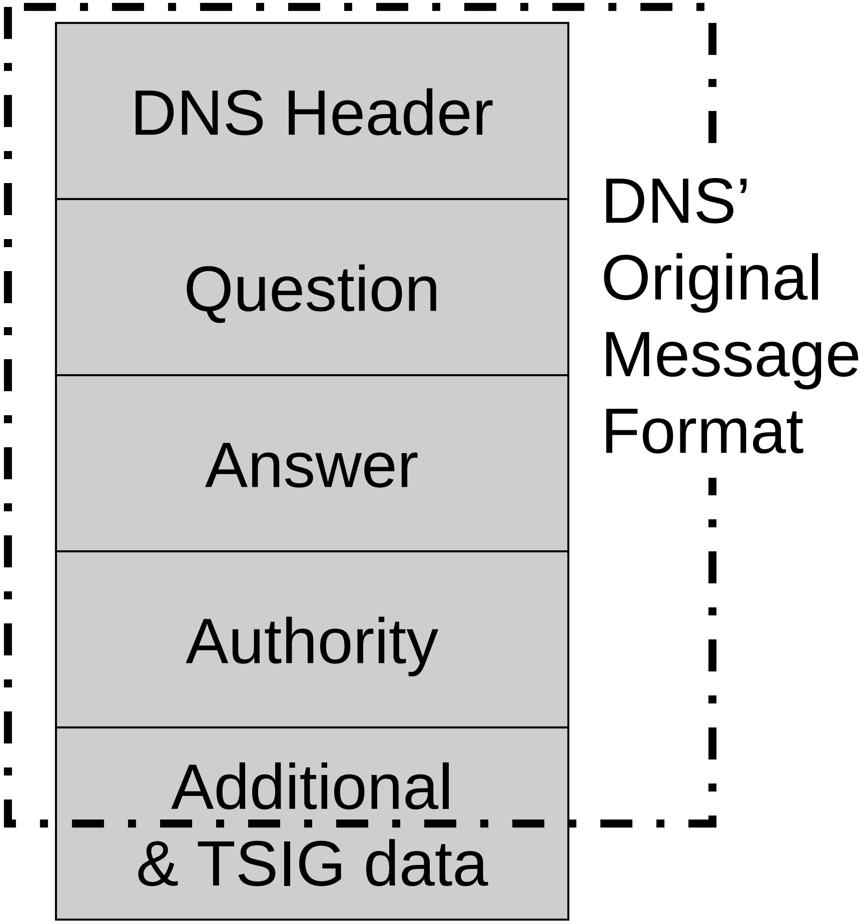
Easy vs. Complicated Features

- The components of DNSSEC have been developed somewhat independent of each other
- Through workshops it is apparent that some parts of DNSSEC are ready for use, others are harder to understand, some need more work
- For the first time, I'll be organizing this tutorial by "ready to use" instead of chronological development

Easy Features

- TSIG - From "Transaction Signature"
  - Uses "keyed hashes" to protect messages
  - Messages are time stamped, but clock synchronization is not part of the process
  - Basic role in DNS - to identify a user or host to another host
- CERT records
  - Basic "holder" for certificates

TSIG in the Message



What Does TSIG Do?

- TSIG is a keyed-hash computed over the entire message
  - Provides proof that an arriving message has not been changed in transit
  - That the message was sent recently (not replayed)
  - And that it was sent from someone who shares the secret
- The querier selects a secret, sends the name of the secret and hash in message
  - but not the secret itself

TSIG in the named.conf file

```
key "test" {
  algorithm hmac-md5;
  secret "qarW1YvJ3OO+f/ToV6ORGw==";
};
```

- This is a BIND-specific topic
- key statements must appear before use, except for rndc

Making Use of TSIG

- Remote Name server Daemon Controller
- Zone transfers
- Dynamic Updates
- Queries and Responses

rndc

- Name server permits this when a "controls" section is in the .conf file
  - Note, key is defined **after** controls statement

```
controls {
   inet 127.0.0.1 allow {127.0.0.1;} keys { rndc_key;};
};
key rndc_key {
   algorithm HMAC-MD5;
   secret "QaRw1Yvj300+f/ToV6ORGw==";
};
```

rndc client configuration

- client program uses /etc/rndc.conf or command line arguments

```
key rndc_key {
     algorithm "HMAC-MD5";
     secret "QaRw1Yvj300+f/ToV6ORGw==";
};
options {
     default-server "127.0.0.1";
     default-key rndc-key;
};
server "127.0.0.1" {
     key rndc_key;
};
```

Zone transfers

- Primary server
  - 10.33.40.46

```
key "test" {
  algorithm hmac-md5;
  secret "ThePlaceToBe";
};
server 10.33.40.35 {
  keys {test;};
};
```

- Secondary server
  - 10.33.40.35

```
key "test" {
  algorithm hmac-md5;
  secret "ThePlaceToBe";
};
server 10.33.40.46 {
  keys {test;};
};
```

Dynamic Update

- An advanced feature, not yet complete
- Securing it relies on TSIG
  - Two forms of security
  - But there is still an issue

Securing Dynamic Update

- Marking a zone as dynamic is done by specifying how the updates are secured

- Access control based on IP address
  - Weak, I'll ignore this
- Coarse-grained access control
  - A secret enables changes to any part of the zone
- Fine-grained access control
  - A secret can make restricted changes

allow-update

- Provides coarse control

```
key "keyto.39.171.199" {
  algorithm hmac-md5;
  secret "ThePlaceToBe";
};
zone "39.171.199.in-addr.arpa." {
   type master;
   file "reversemap.zone";
   allow-update {key keyto.39.171.199;};
};
```

- This says that any update signed by the key called "keyto.39.171.199" can update any part of the zone

update-policy

- Allows fine-grained control

```
key key1. {...};
key key2. {...};
zone "39.171.199.in-addr.arpa." {
  type master;
  file "reverse-map.zone";
  update-policy {
   grant key1. name 1.39.171.199.in-addr.arpa. PTR;
   grant key2. name 2.39.171.199.in-addr.arpa. PTR;
  };
};
```

- This permits the specified keys to change just parts of the zone

Remaining Issue

- Dynamic Update zones that are signed suffer from "signature rot"
  - Haven't covered signatures yet
  - Suffice it to say, this issue is being worked upon
  - Time permitting, this will be covered later in presentation
- Dynamic Update with DNSSEC is *almost* ready for prime time

Other queries and responses

- Using TSIG for all queries and responses is not ready for prime time
  - One issue is storing a secret on a multi-user machine
  - There isn't an easy way to configure a secret for a resolver
  - There also needs to be coordination with DHCP as TSIG secrets are server specific
- But, TSIG can be used with dig, which is useful for testing configurations

Supplying a secret to dig

- dig can be passed a secret
  - Via the command line, meaning the secret is momentarily vulnerable (via the ps command)
  - For testing, this is acceptable
- dig option is "**`-y name:secret`**"

```
dig @0 1.39.171.199.in-addr.arpa. PTR -y \
test:qarW1YvJ3OO+f/ToV6ORGw==
```

- For testing, mnemonic secrets are advantageous, or a working cut-n-paste.

One last comment on TSIG

- When a query arrives with a TSIG
  - The responder must know the secret to verify the message
  - The responder will attach a TSIG to the response using the same secret
- "Server" statements are used by name servers to know when to use a secret on "outgoing" messages
  - AXFR query, NOTIFY, lookups
  - "Server" statements are not needed for stub resolvers

What about SIG(0) and TKEY?

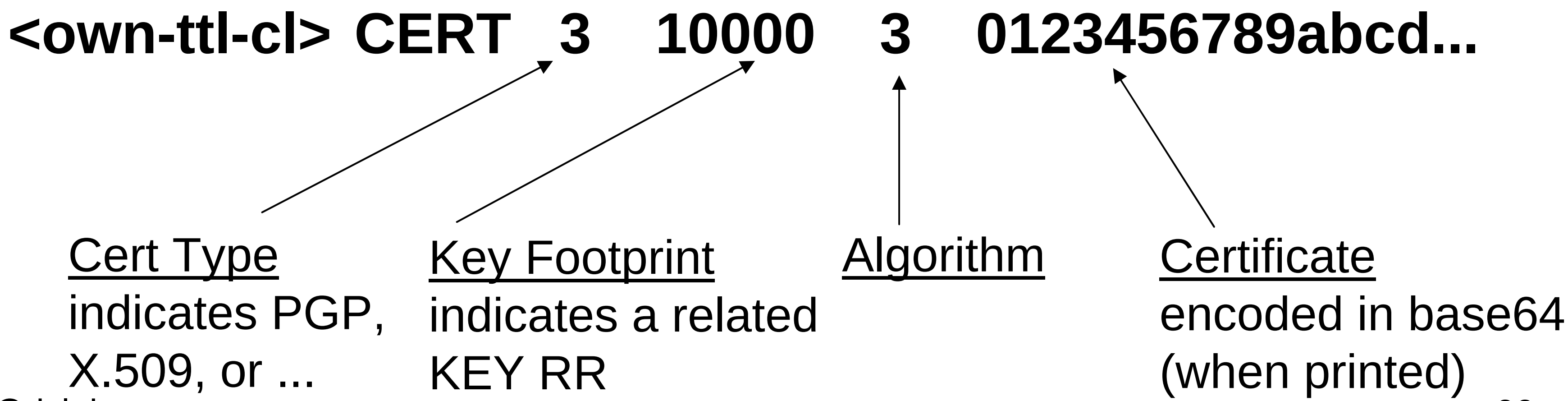
- SIG(0) is a public-key alternative to TSIG and predates TSIG
  - I don't know of anyone using it
  - Instead of a secret value, a private key is needed, which is still an issue on a multiuser machine
- TKEY is a mechanism to negotiate a TSIG on the fly
  - 4 modes, two are not used and not mentioned
  - SIG(0) initiated
  - GSSAPI, used by Microsoft and Lucent

CERT Records

- Now for a completely different, but also straightforward, topic
- The CERT RR is a container for certificates
  - X.509
  - PGP
  - Others
- The certificate can be standalone, like a TXT record for a comment
- The certificate can reference a key in a KEY RR

CERT RR Syntax

- The first RDATA element indicates the kind of certificate
- The second element points to a KEY RR
- The third element indicates the KEY algorithm
- The final element is the binary certificate



<own-ttl-cl> CERT 3 10000 3 0123456789abcd...

Cert Type
indicates PGP,
X.509, or ...

Key Footprint
indicates a related
KEY RR

Algorithm

Certificate
encoded in base64
(when printed)

Limitations on CERT

- This is **not** a PKI
- DNS is used to make a PKI's certificates available
- Relying on DNS signatures to secure the certificate chain is risky
- Instead, rely on the certificate's built in chain of trust
  - With this, it is reasonable to use the CERT record even in unsigned zones
- What's a "signed zone"
  - Answer: a good segue...

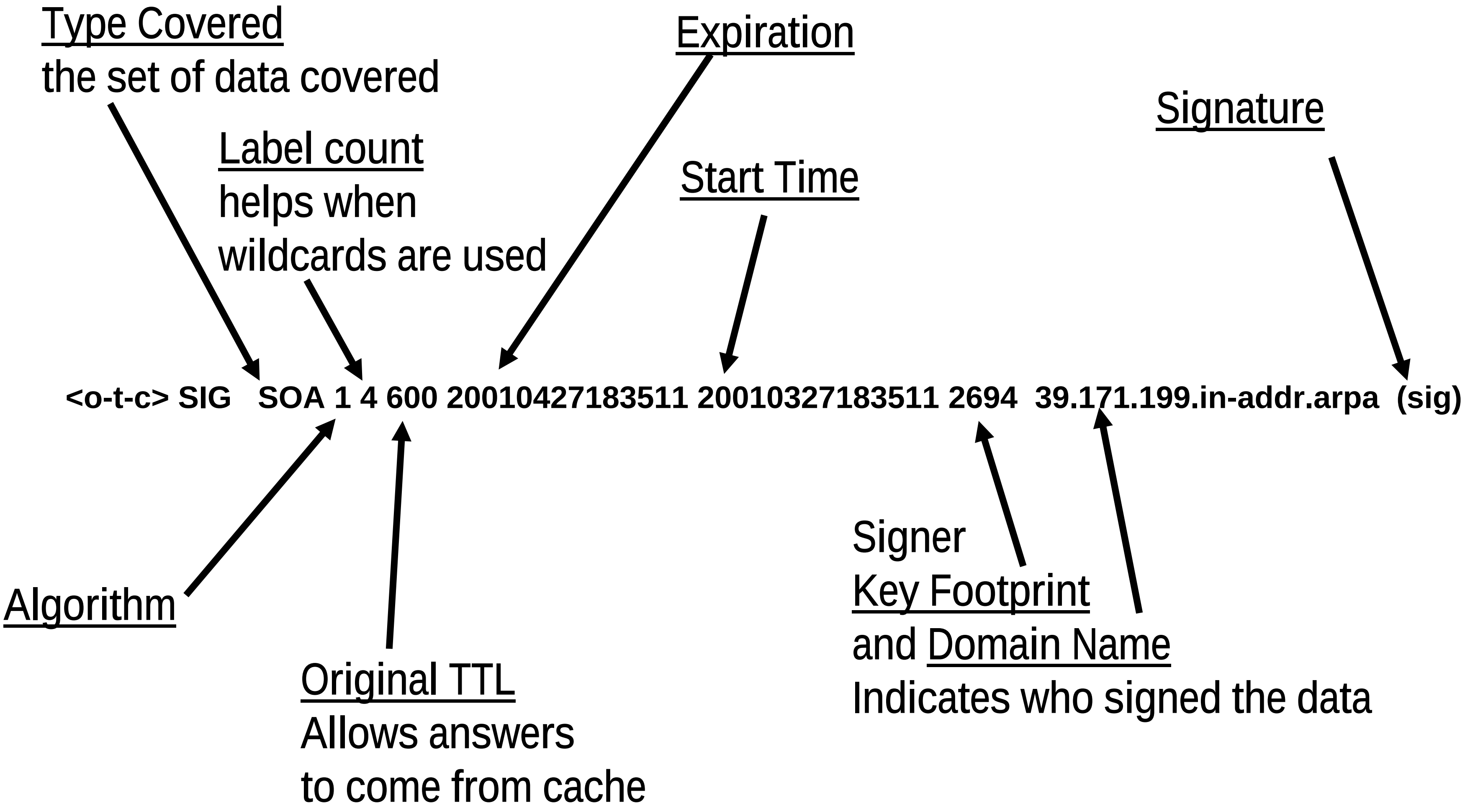
The Complicated Features

- The SIG, KEY, and NXT records
- How they impact zone files and queries
- Tools available to manipulate the records

The SIG record

- The SIG record holds a digital signature
- This record is only intended for use within DNS
  - It is not a general purpose signature holder
- Data held in the SIG RR (Highlights)
  - Validity period
  - The identity of the verifying key
  - The signature

SIG RR syntax



Type Covered
the set of data covered

Label count
helps when
wildcards are used

Expiration

Start Time

Signature

<o-t-c> SIG SOA 1 4 600 20010427183511 20010327183511 2694 39.171.199.in-addr.arpa (sig)

Algorithm

Original TTL
Allows answers
to come from cache

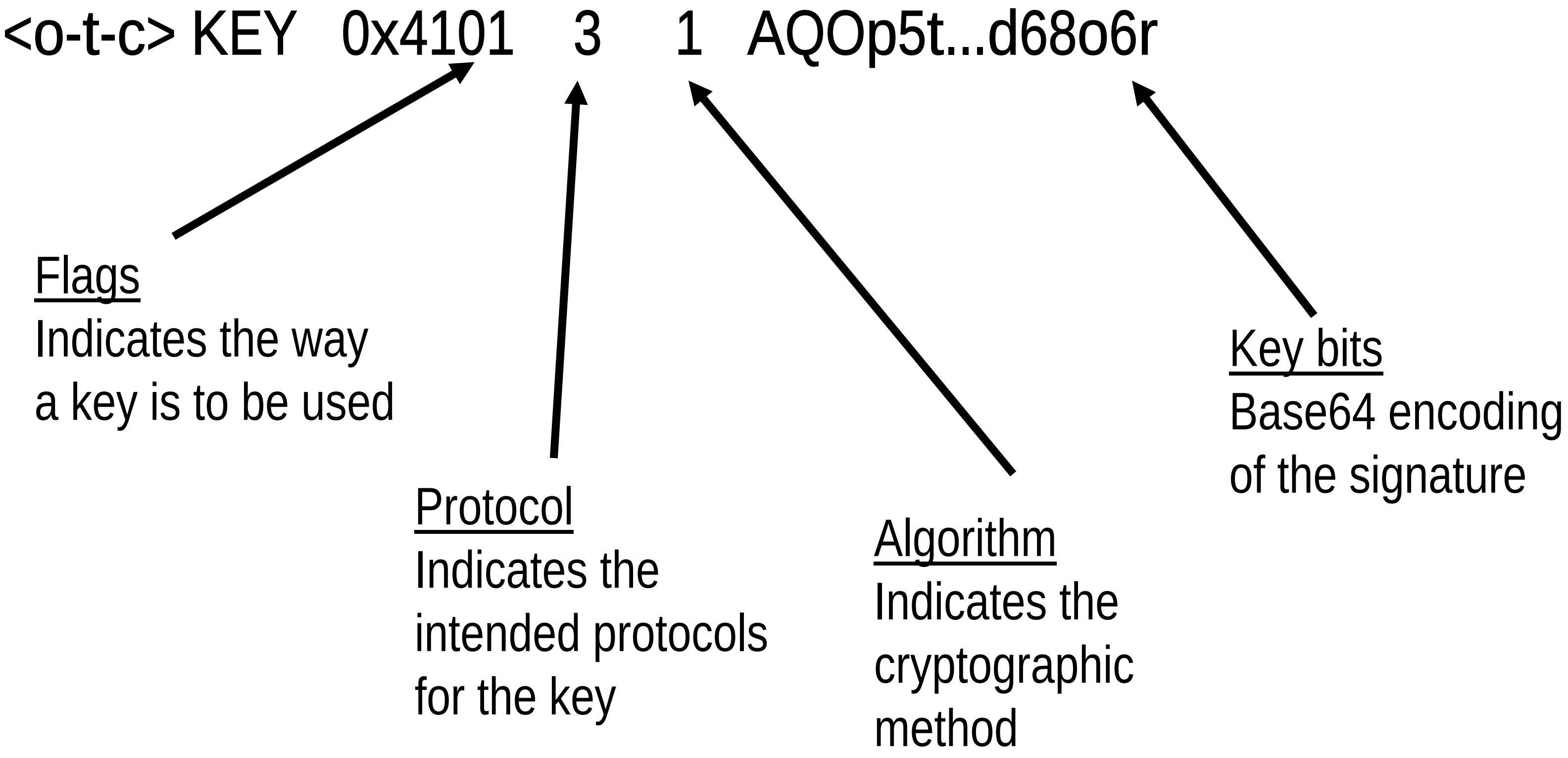
Signer
Key Footprint
and Domain Name
Indicates who signed the data

The KEY record

- The KEY record is a general purpose holder of public keys
  - E.g., an RSA key pair, a DSA key pair
  - Not a TSIG nor other shared secret!
  - The KEY may or may not be DNS specific
    - unlike the SIG RR

KEY RR Syntax

<o-t-c> is short hand for owner-ttl-class



<o-t-c> KEY 0x4101 3 1 AQOp5t...d68o6r

Flags
Indicates the way
a key is to be used

Protocol
Indicates the
intended protocols
for the key

Algorithm
Indicates the
cryptographic
method

Key bits
Base64 encoding
of the signature

The NXT record

- The NXT record is used to deny existence of data
  - With authentication (proof)
  - Kind of like signing the NXDOMAIN response
- There is one nit against the NXT record
  - The method it uses exposes the entire zone's contents to a determined querier
  - There is an option under consideration

NXT RR Syntax

Type Bit Map
sets at the owner,
other sets absent

(owner is 39.171.199.in-addr.arpa.)
<o-t-c> NXT 1.39.171.199.in-addr.arpa. NS SOA TXT SIG KEY NXT

Next name
No name fits between 39.171.199.in-addr.arpa. and
1.39.171.199.in-addr.arpa.

An unsigned zone

```
$ORIGIN myhome.zone4.sec.test.
@     450 IN SOA ns1.myhome.zone4.sec.test.  root.ns1.myhome... (
                        100001 21600 3600 604800 300 )
      450 IN NS  ns1.myhome.zone4.sec.test.
      450 IN NS  ns2.myhome.zone4.sec.test.
dynup 450 IN NS  ns1.myhome.zone4.sec.test.
      450 IN NS  ns2.myhome.zone4.sec.test.
host1 450 IN A   10.53.53.101
host2 450 IN A   10.53.53.102
... and more...
```

- Not a reverse map zone, sorry

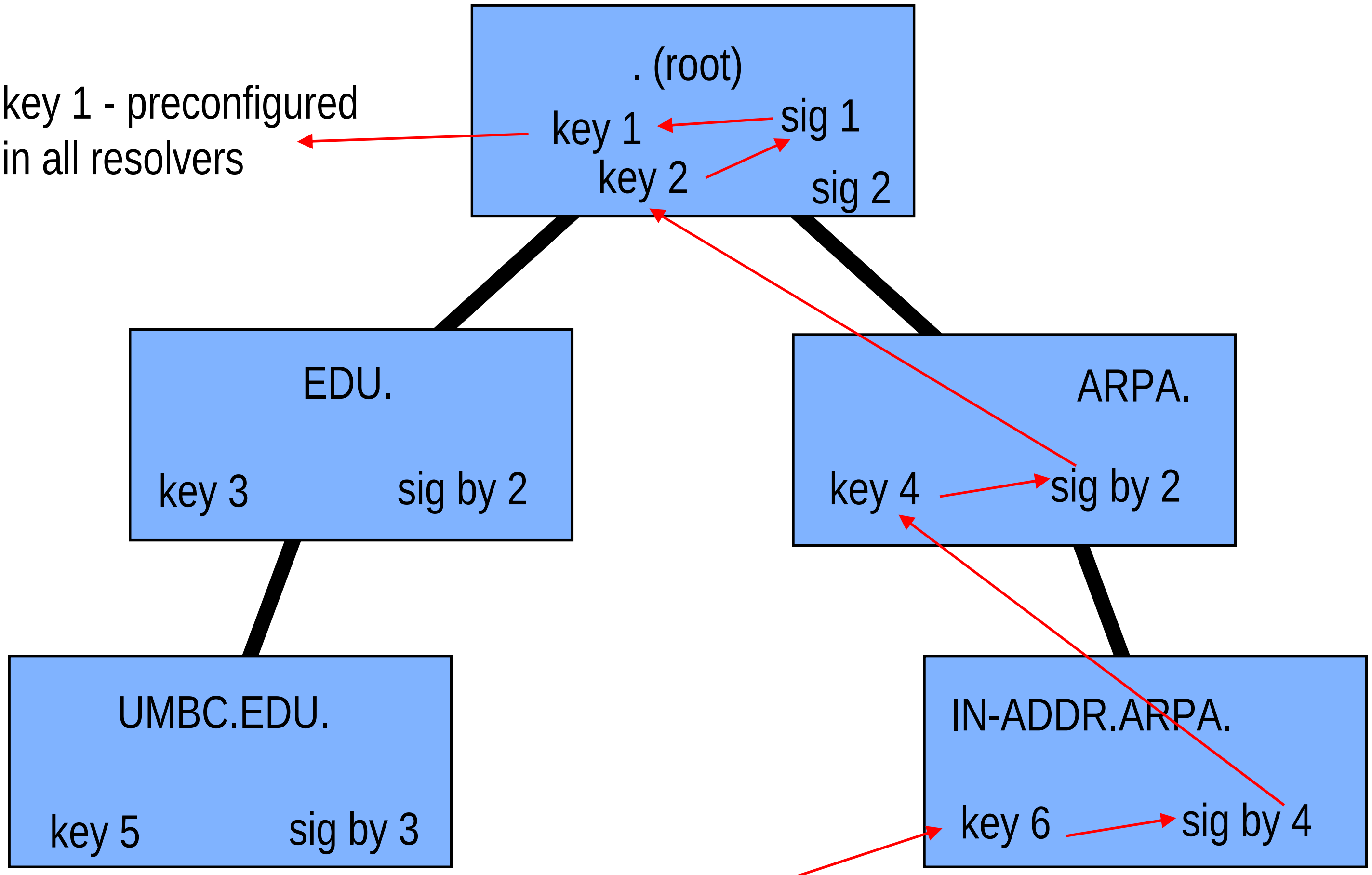
The same zone signed (part 1)

```
; File written on Thu Feb 15 16:11:38 2001
; dnssec_signzone version 9.1.0-modified
$ORIGIN myhome.zone4.sec.test.
@  450   IN SOA   ns1.myhome.zone4.sec.test. root.ns1.myhome.zone4.sec.test. (
                  100001 21600 3600 604800 300 )
   450   SIG      SOA 1 4 450 20010317211138 20010215211138 (
                  7721 myhome.zone4.sec.test.
                  LOUkhBghJB+516jUvqmS7z19DNazUKRxmz
                  JaQAR3lPmm7sW6Hu0RElr39uRxKkySarfM
                  XD/uIZijbsZfwYcL+Q== )
   450   NS       ns1.myhome.zone4.sec.test.
   450   NS       ns2.myhome.zone4.sec.test.
   450   SIG      NS 1 4 450 20010317211138 20010215211138 7721 (
                  myhome.zone4.sec.test. zYFJ+on0oR/NB9OEsPe...l6QQCrgSf+q
                  PDwPMa0qTQuwQw== )
   450   KEY      256 3 1 AQPPXEoG9mWfEG0jEk/TR...V3q5IA8Hinn ) ; key id = 7721
   450   SIG      KEY 1 4 450 20010416204257 20010215204257 7721 (
                  myhome.zone4.sec.test. G+t8TThil757pp9CVZR...mJvzC/AVmSdzQQ== )
   450   SIG      KEY 1 4 450 20010416204257 20010215204257 31512 (
                  zone4.sec.test. LSQn44NYAeeLSUWDms...TJQyq6NxTfsjsiTdQ31
                  +doQ8fUASqvMQQ== )
...continues on next slide...
```

Part 2

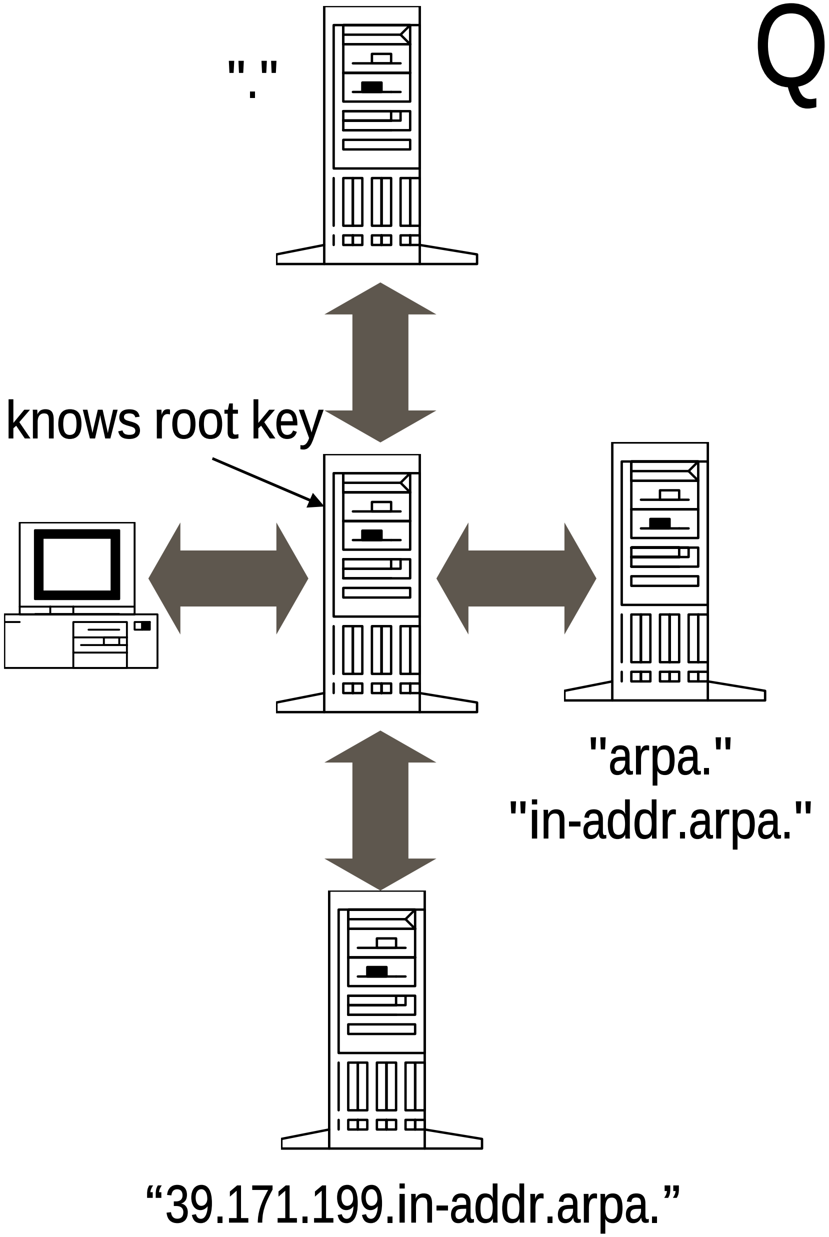
```
         450     NXT      dynup.myhome.zone4.sec.test. NS SOA SIG KEY NXT
         450     SIG      NXT 1 4 450 20010317211138 20010215211138 (
                          7721 myhome.zone4.sec.test.
                          Mdz5r8ouNnj+XYFWo4Qo0R/eCtzZeq8KTjKCG428v
                          PnxMwo+Uq6Xd8x3hmAU1QWVBikRoJG0xgoXnzmdcOCMgg== )
dynup    450     IN NS    ns1.myhome.zone4.sec.test.
         450     IN NS    ns2.myhome.zone4.sec.test.
         450     NXT      host1.myhome.zone4.sec.test. NS SIG NXT
         450     SIG      NXT 1 5 450 20010317211138 (
                          20010215211138 7721 myhome.zone4.sec.test.
                          zzBFfBZjguc9XVKPCsuzlkMc04g1uz6u+JSP
                          f4yF7dCxzJjnI7akJIeaTKsC5j+iQ6i4zkSg
                          Uh7238SWzgO+1w== )
host1    450     IN A     10.53.53.101
         450     SIG      A 1 5 450 20010317211138 (
                          20010215211138 7721 myhome.zone4.sec.test.
                          GiBTjzikKZO5CN2lUJuHUf1thgQfw3V9axT8
                          KnDrhGZM/u6h4lJx7dxA6NILjMQ9hihZYjWB
                          LAKcfDjdF16krA== )
         450     NXT      host2.myhome.zone4.sec.test. A SIG NXT
         450     SIG      NXT 1 5 450 20010317211138 (
                          20010215211138 7721 myhome.zone4.sec.test.
                          Vlfv/rzgWzfc+S0+IEckT5QMRjClpqJLhN0Z
                          MA4UBr+ANujK0ghJdvifdSysAC60FH8Ex33f
                          vuC+jrKum/A7yg== )
....and there's still more to the zone, not shown
```

Chain of trust



```
in-addr.arpa. in soa a.root-servers.net. noc.netsol.com. (
                     2001032115 1800 900 604800 86400), signed by 6
```

Queries



- Query: 1.39.171.199.in-addr.arpa. PTR
  - "." refers to arpa server
  - "in-addr.arpa." refers to 39.171.199 server
  - Answer contains (all or some of)
    - PTR for 1.39.171.199.in-addr.arpa
    - SIG by 39.171.199.in-addr.arpa.
    - KEY of 39.171.199.in-addr.arpa.
    - SIG of that KEY by in-addr.arpa.
- Query for KEY of in-addr.arpa.
  - KEY of in-addr.arpa. and SIG by arpa. KEY
- Query for KEY of arpa.
  - KEY of arpa. and SIG by root key
- Now, can verify chain

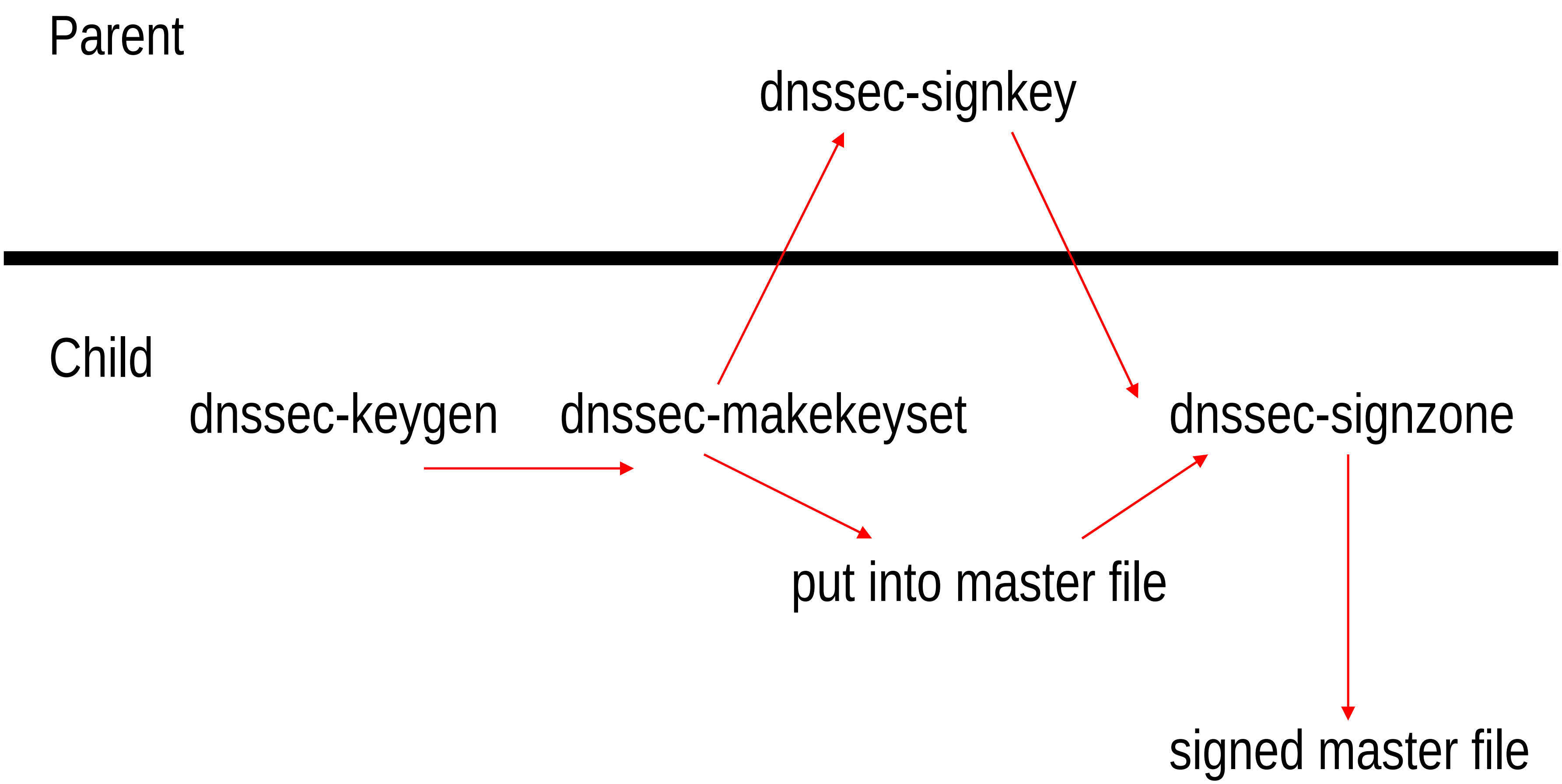
Delegations

- The biggest issue facing DNSSEC is the delegation interaction
  - E.g., how will .edu sign umbc.edu.'s key?
  - How is key 5 signed by key 3? (Previous slide)
- umbc.edu generates a key, ships it to edu., the signature is returned
  - How will each side trust the other?
  - What happens when the .edu key changes?

BIND's DNSSEC tools

- dnssec-keygen
  - Generates public/private keys and shared secrets
- dnssec-signzone
  - Signs master / zonefile
- dnssec-makekeyset
  - Assembles and self-signs keys for validation
- dnssec-signkey
  - Signs a key set (e.g., by parent)

Using the tools



Parent

dnssec-signkey

Child

dnssec-keygen

dnssec-makekeyset

dnssec-signzone

put into master file

signed master file

Wrap-up

- Some parts of DNSSEC are ready for use
  - Generally TSIG-based protections
- Some features of DNS are not mature
  - Dynamic Update and DNSSEC
- Some features of DNSSEC are still progressing
  - Digital Signatures and Delegations
- Remaining Issues & Work
  - Whether the NXT is replaced or not
  - How DNSSEC (keys) will impact operations
  - Writing client software to make use of features

Reference Material

- IETF Sites (http://www.ietf.org/...)
  - DNSEXT: html.charters/dnsext-charter.html
  - DNSOP: html.charters/dnsop-charter.html
  - State of DNS: internet-drafts/draft-lewis-state-of-dnssec-01.txt
- DNSSEC Experiments
  - http://www.sigz.net
  - https://keys.cairn.net
  - http://secnl.nlnetlabs.nl/
- ISC
  - BIND 9 http://www.isc.org/products/BIND/

RFC's

- RFC's defining DNSSEC (available from IETF)
  - 2535 - Current base definition
  - 2536,2537 - Define key and signature processing
  - 2538 - CERT record
  - 2939 - Diffie Hellman keys
  - 2845 - TSIG
  - 2930 - TKEY
  - 2931 - SIG(0)
  - 3007 - Secure Dynamic Update (ignore 2137)
  - 3008 - Signing Authorization Model
  - 3090 - Clarifications