DNS/DNSSEC Deployment Workshop

As part of bdNOG 2 Conference and Workshop

11 November 2014

APNIC

Presenter

Sheryl Hermoso (Shane)

Training Officer, APNIC

Sheryl has had various roles as a Network and Systems Administrator prior to joining APNIC. Starting her career as a Technical Support Assistant while studying at the University of the Philippines. Sheryl later finished her degree in Computer Engineering and continued to work in the same university as a Network Engineer, where she managed the DILNET network backbone and wireless infrastructure.

Areas of interests:

Wireless/wifi, DNS/DNSSEC, IPv6, and security.

Contact:
Email: sheryl@apnic.net

APNIC

Agenda

- DNS Overview
- BIND DNS Configuration
- Recursive and Forward DNS
- Reverse DNS
- Troubleshooting
- DNS Security Overview
- Transaction Signature (TSIG)
- DNS Security Extensions (DNSSec)
- DNSSec Key Management and Automation

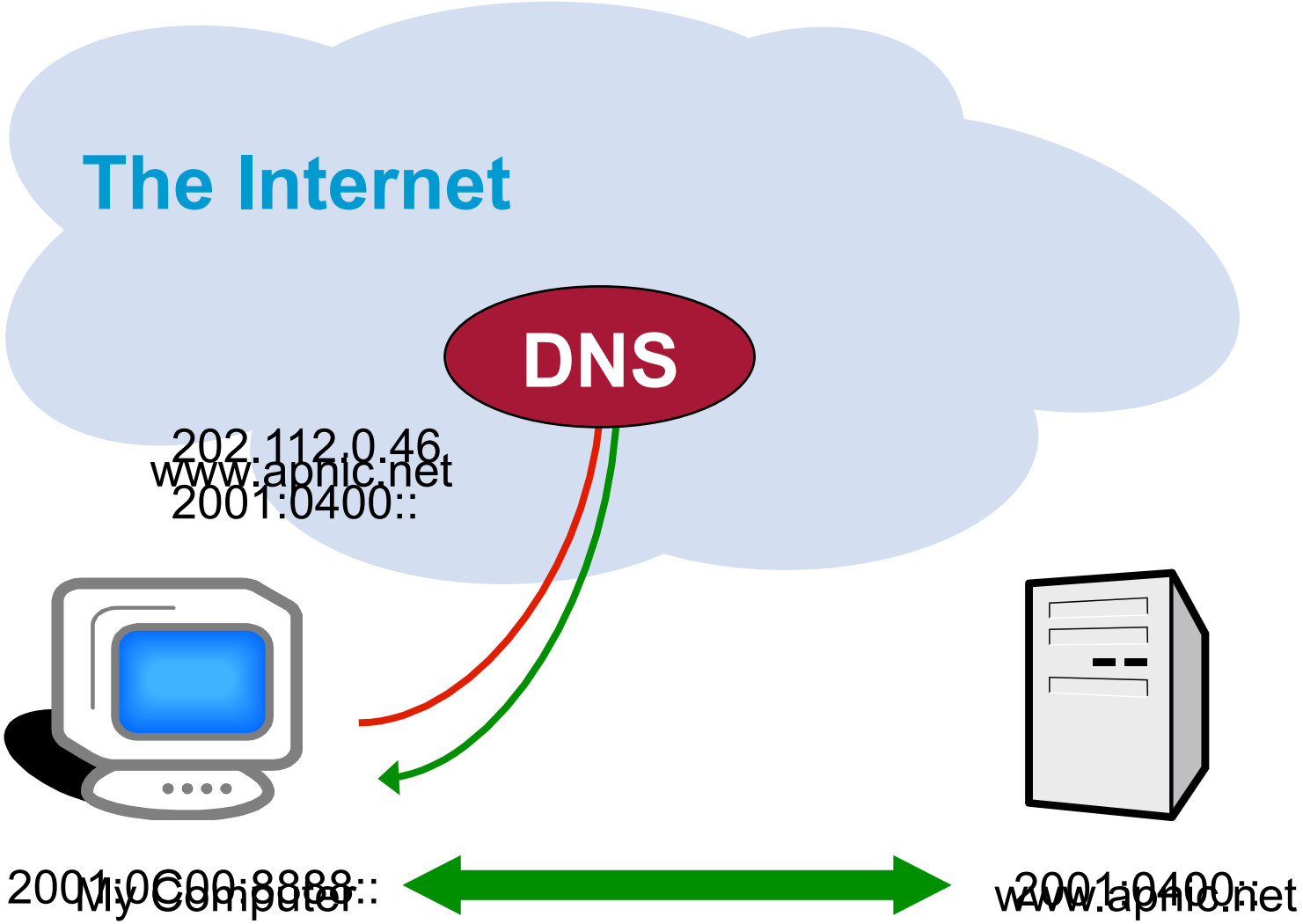
DNS Overview

APNIC

Domain Name System

- A lookup mechanism for translating objects into other objects
  - Mapping names to numbers and vice versa
- A globally distributed, loosely coherent, scalable, reliable, dynamic database
- Comprised of three components
  - A “name space”
  - Servers making that name space available
  - Resolvers (clients) which query the servers about the name space
- A critical piece of the Internet infrastructure

IP Addresses vs Domain Names



Old Solution: hosts.txt

- A centrally-maintained file, distributed to all hosts on the Internet
- Issues with having just one file
  - Becomes huge after some time
  - Needs frequent copying to ALL hosts
  - Consistency
  - Always out-of-date
  - Name uniqueness
  - Single point of administration

```
// hosts.txt
SERVER1         128.4.13.9
WEBMAIL         4.98.133.7
FTPHOST         200.10.194.33
```

This feature still exists:
[Unix] /etc/hosts
[Windows] c:\windows\hosts

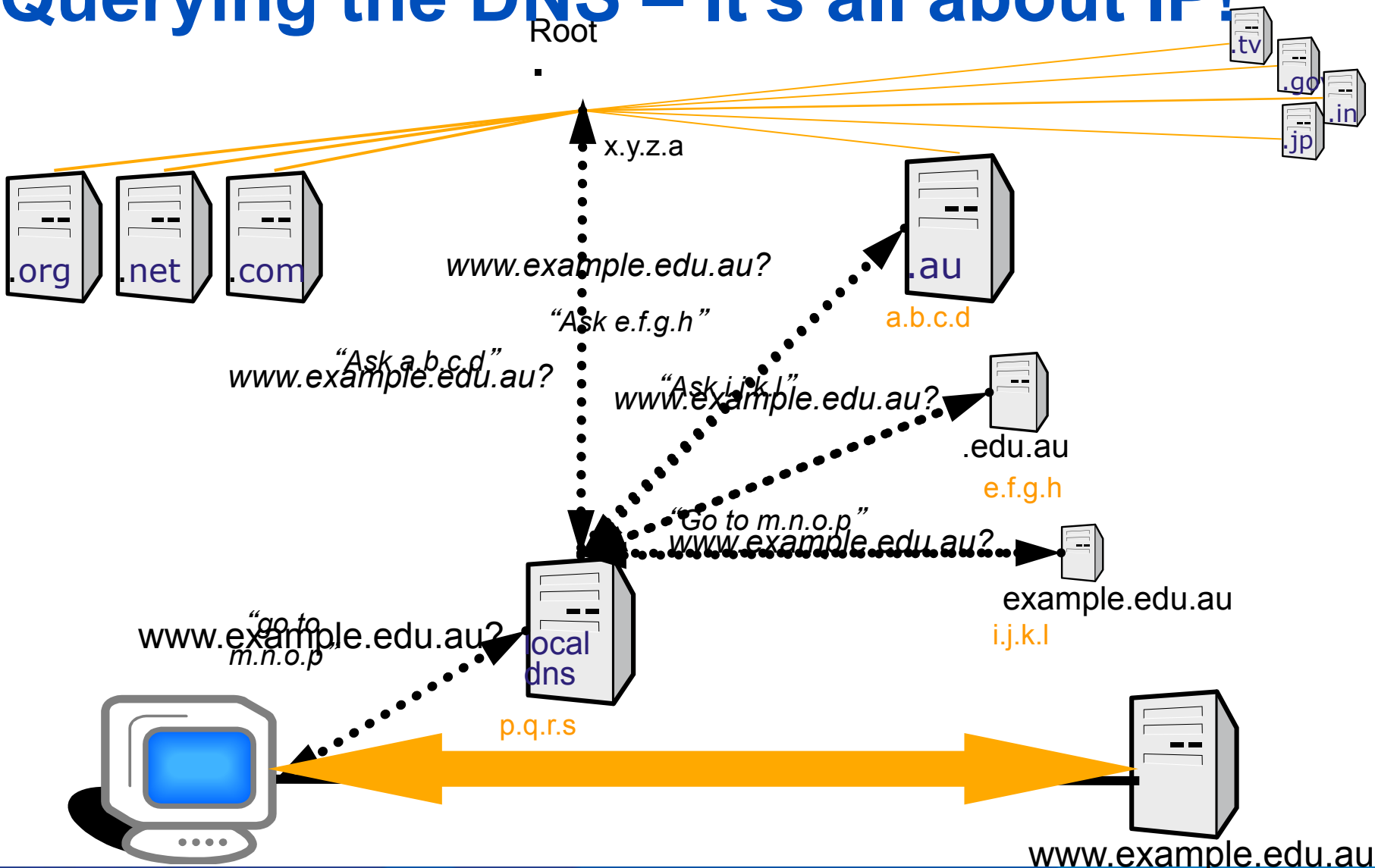
DNS Features

- Global distribution
  - Shares the load and administration
- Loose Coherency
  - Geographically distributed, but still coherent
- Scalability
  - can add DNS servers without affecting the entire DNS
- Reliability
- Dynamicity
  - Modify and update data dynamically

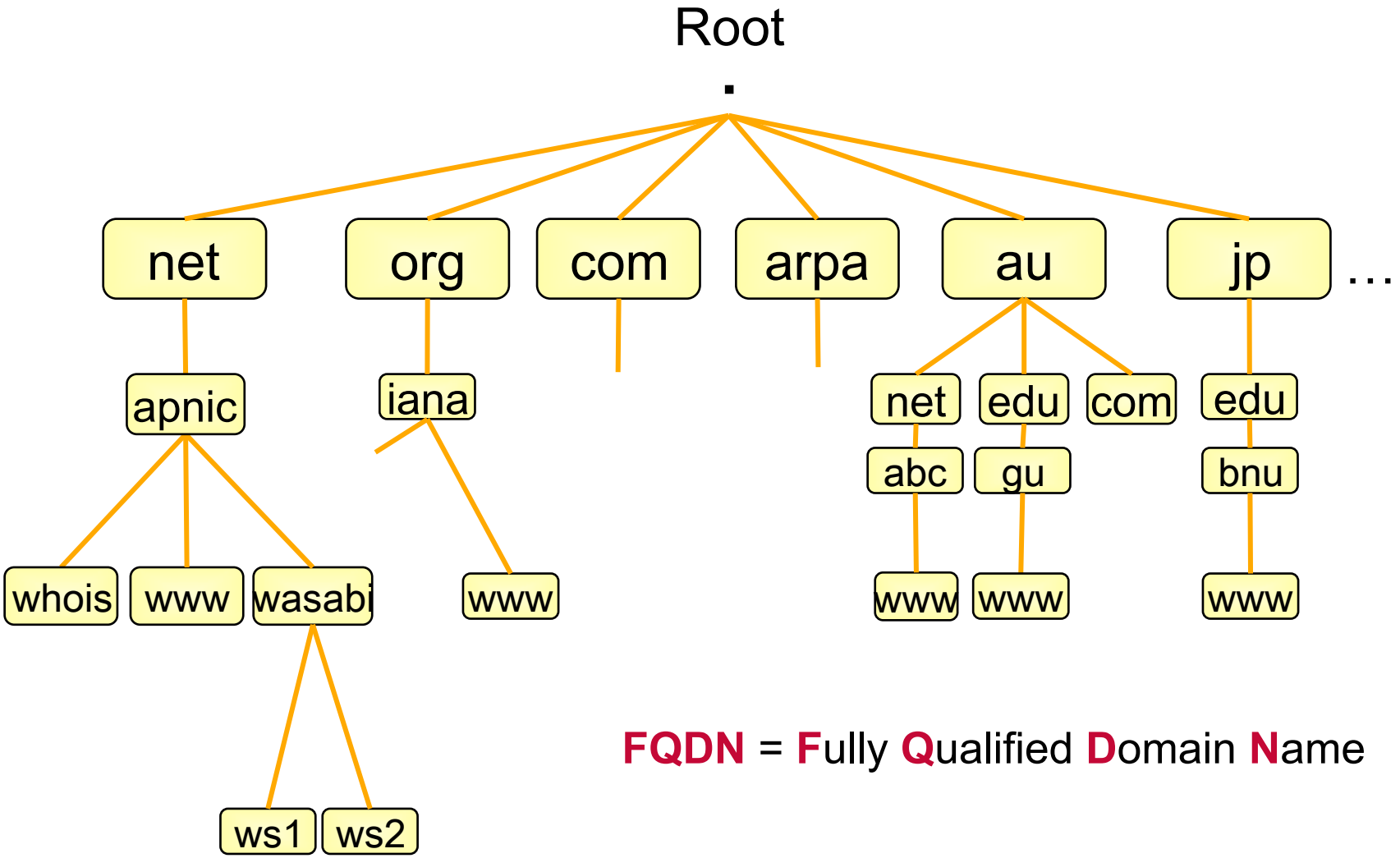
DNS Features

- DNS is a client-server application
- Requests and responses are normally sent in UDP packets, port 53
- Occasionally uses TCP, port 53
  - for very large requests, e.g. zone transfer from master to slave

Querying the DNS – It's all about IP!



The DNS Tree Hierarchy

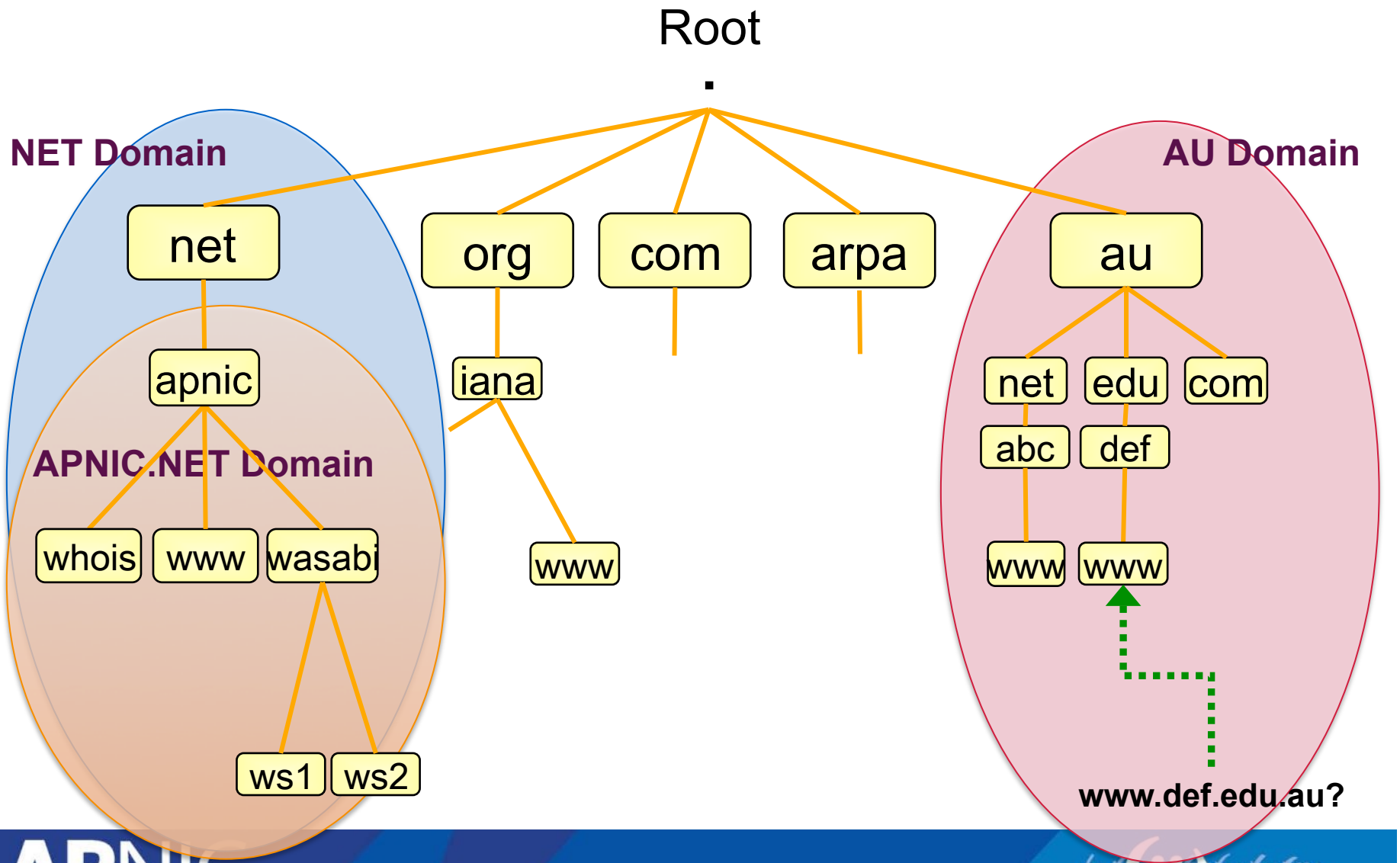


FQDN = **F**ully **Q**ualified **D**omain **N**ame

Domains

- Domains are “namespaces”
- Everything below .com is in the com domain
- Everything below apnic.net is in the apnic.net domain and in the net domain

Domains


Root
.
NET Domain
AU Domain
net
org
com
arpa
au
apnic
iana
net
edu
com
APNIC.NET Domain
abc
def
whois
www
wasabi
www
www
www
ws1
ws2
www.def.edu.au?

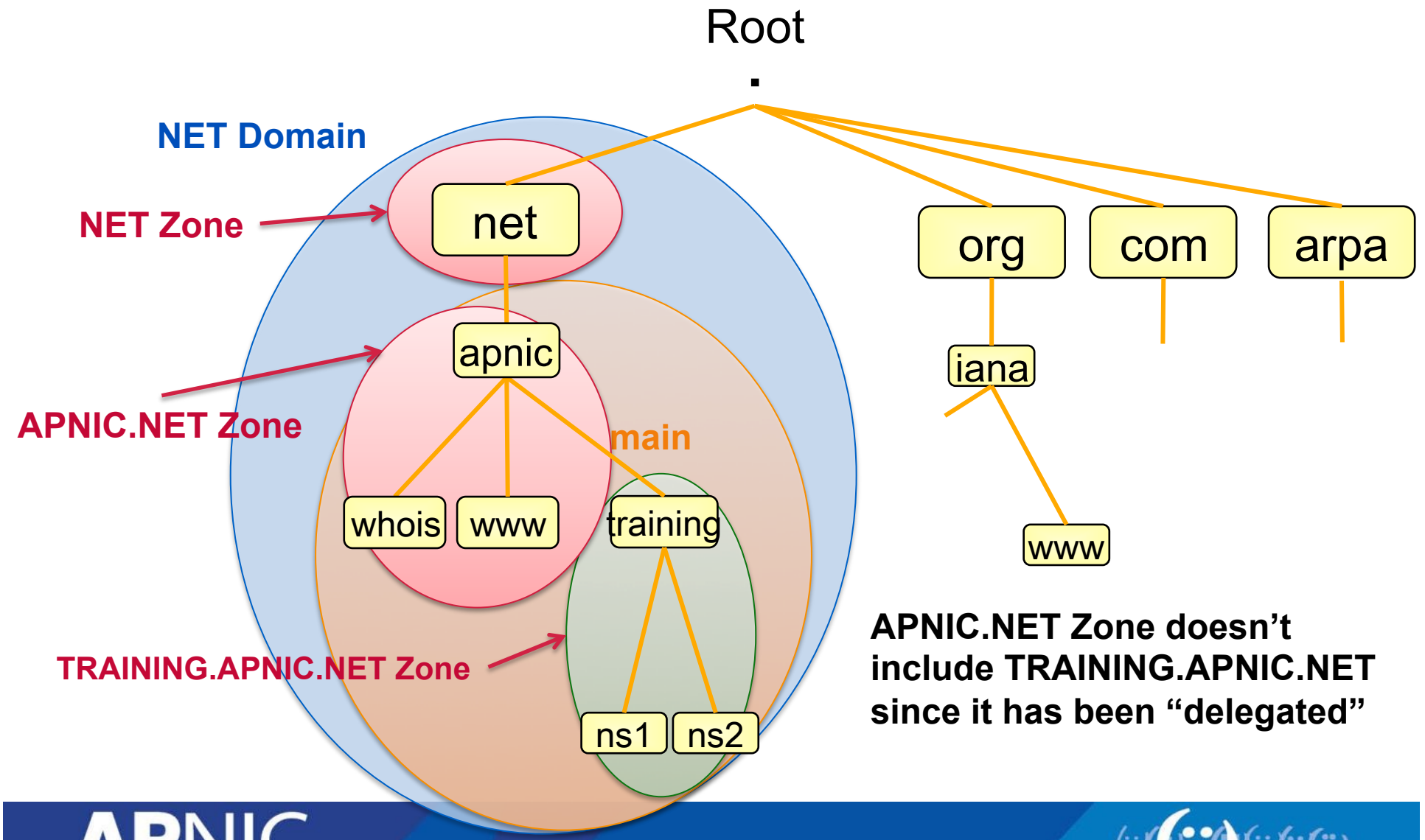
Delegation

- Administrators can create subdomains to group hosts
  - According to geography, organizational affiliation or any other criterion
- An administrator of a domain can delegate responsibility for managing a subdomain to someone else
  - But this isn't required
- The parent domain retains links to the delegated subdomain
  - The parent domain "remembers" who it delegated the subdomain to

Zones and Delegations

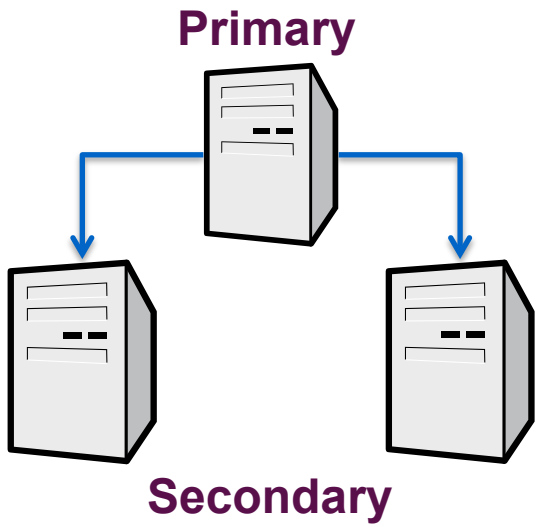
- Zones are “administrative spaces”
- Zone administrators are responsible for portion of a domain’s name space
- Authority is delegated from parent to child

Zones


Root
.
NET Domain
NET Zone
net
org
com
arpa
apnic
iana
APNIC.NET Zone
main
whois
www
training
www
TRAINING.APNIC.NET Zone
ns1
ns2
APNIC.NET Zone doesn't include TRAINING.APNIC.NET since it has been "delegated"

Name Servers

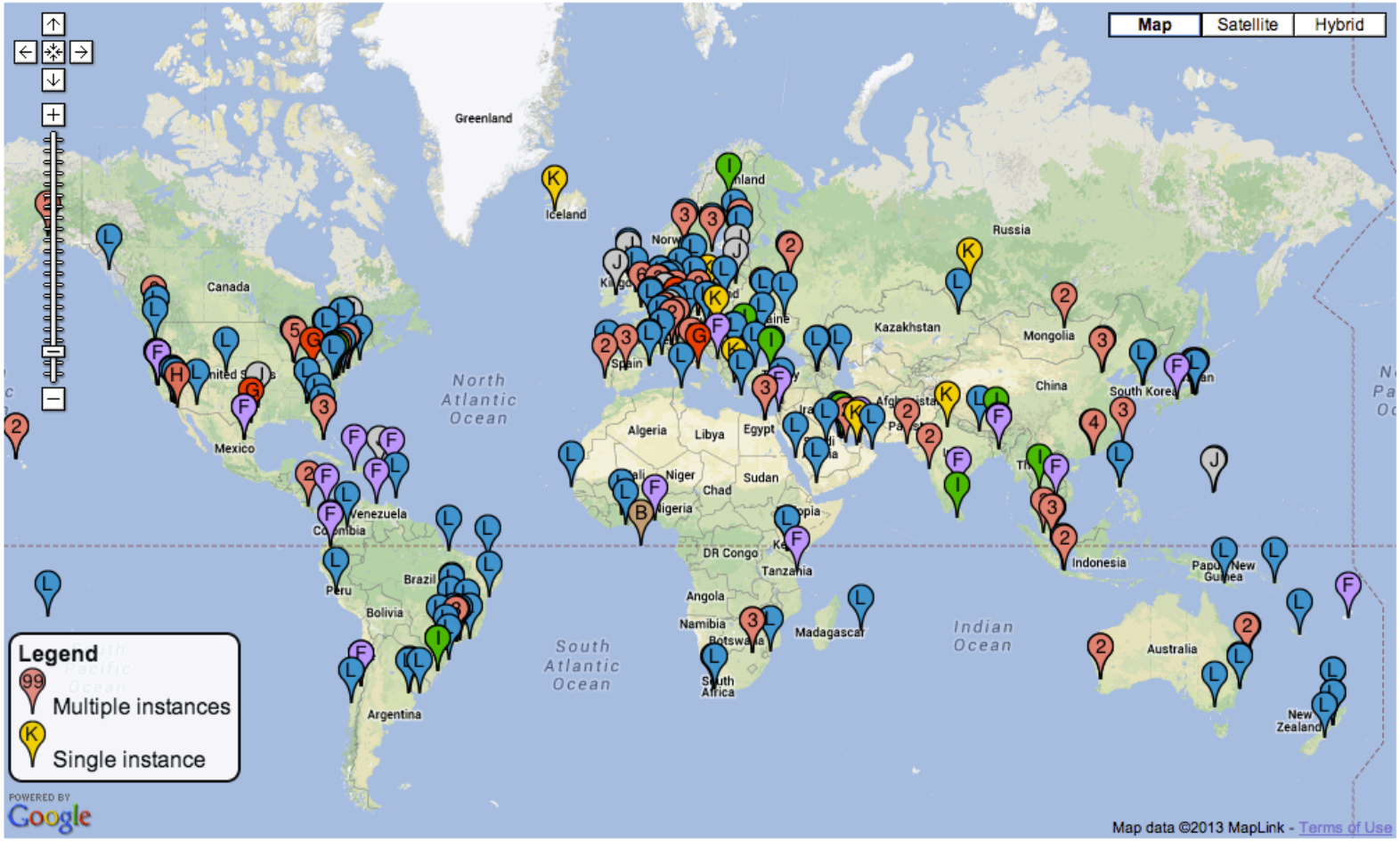
- Name servers answer ‘DNS’ questions
- Several types of name servers
  - Authoritative servers
    - master (primary)
    - slave (secondary)
  - Caching or recursive servers
    - also caching forwarders
- Mixture of functions



Root Servers

- The top of the DNS hierarchy
- There are 13 root name servers operated around the world, with names from `[a-m].root-servers.net`
- There are more than 13 physical root name servers
  - Each rootserver has an instance deployed via anycast
- Root hints file come in many names (db.cache, named.root, named.cache, named.ca)
  - Get it from ftp.rs.internic.net
- See root-servers.org for more detail

Root Servers

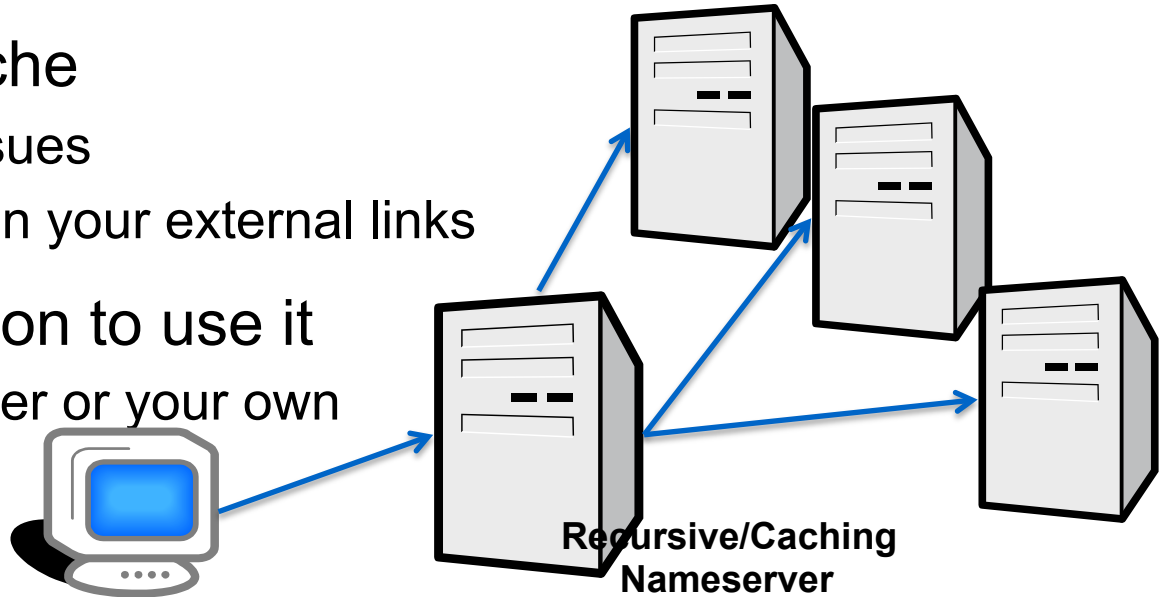


Resolver

- Or “stub” resolver
- A piece of software (usually in the operating system) which formats the DNS request into UDP packets
- A stub resolver is a minimal resolver that forwards all requests to a local recursive nameserver
  - How to find a local nameserver? The IP address should be explicitly configured in the resolver.
- Every host needs a resolver
  - In Linux, it uses /etc/resolv.conf
- Note that it is always a good idea to configure more than one nameserver

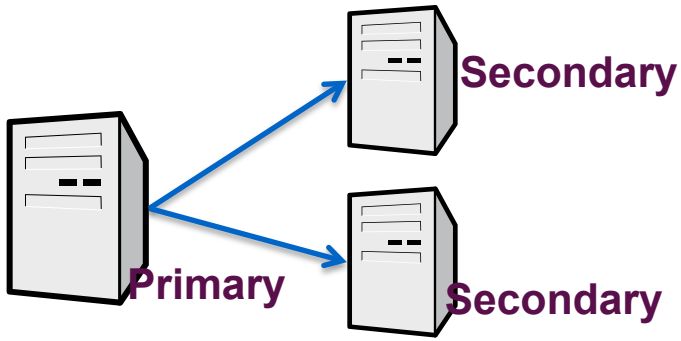
Recursive Nameserver

- The job of the recursive nameserver is to locate the authoritative nameserver and get back the answer
- This process is iterative – starts at the root
- Recursive servers are also usually caching servers
- Prefer a nearby cache
  - Minimizes latency issues
  - Also reduces traffic on your external links
- Must have permission to use it
  - Your ISP's nameserver or your own



Authoritative Nameserver

- A nameserver that is authorised to provide an answer for a particular domain.
  - Can be more than one auth nameserver
- Two types based on management method:
  - Primary (Master) and Secondary (Slave)
- Only one primary nameserver
  - All changes to the zone are done in the primary
- Secondary nameserver/s will retrieve a copy of the zonefile from the primary server
  - Slaves poll the master periodically
- Primary server can “notify” the slaves



Resource Records

- Entries in the DNS zone file
- Components:

Resource Record	Function
Label	Name substitution for FQDN
TTL	Timing parameter, an expiration limit
Class	IN for Internet, CH for Chaos
Type	RR Type (A, AAAA, MX, PTR) for different purposes
RDATA	Anything after the Type identifier; Additional data

Common Resource Record Types

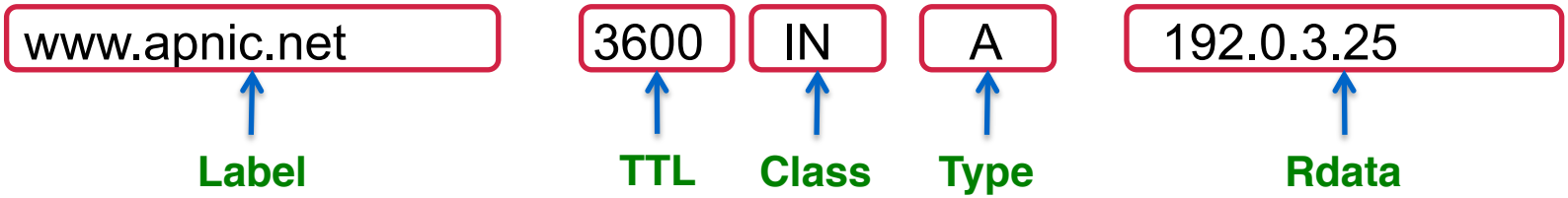
RR Type	Name	Functions
A	Address record	Maps domain name to IP address `www.apnic.net. IN A 203.176.189.99`
AAAA	IPv6 address record	Maps domain name to an IPv6 address `www.apnic.net. IN AAAA 2001:db8::1`
NS	Name server record	Used for delegating zone to a nameserver `apnic.net. IN NS ns1.apnic.net.`
PTR	Pointer record	Maps an IP address to a domain name `99.189.176.203.in-addr.arpa. IN PTR www.apnic.net.`
CNAME	Canonical name	Maps an alias to a hostname `web IN CNAME www.apnic.net.`

MX	Mail Exchanger	Defines where to deliver mail for user @ domain
		`apnic.net. IN MX 10 mail01.apnic.net.`
		`IN MX 20 mail02.apnic.net.`

Example: RRs in a zone file

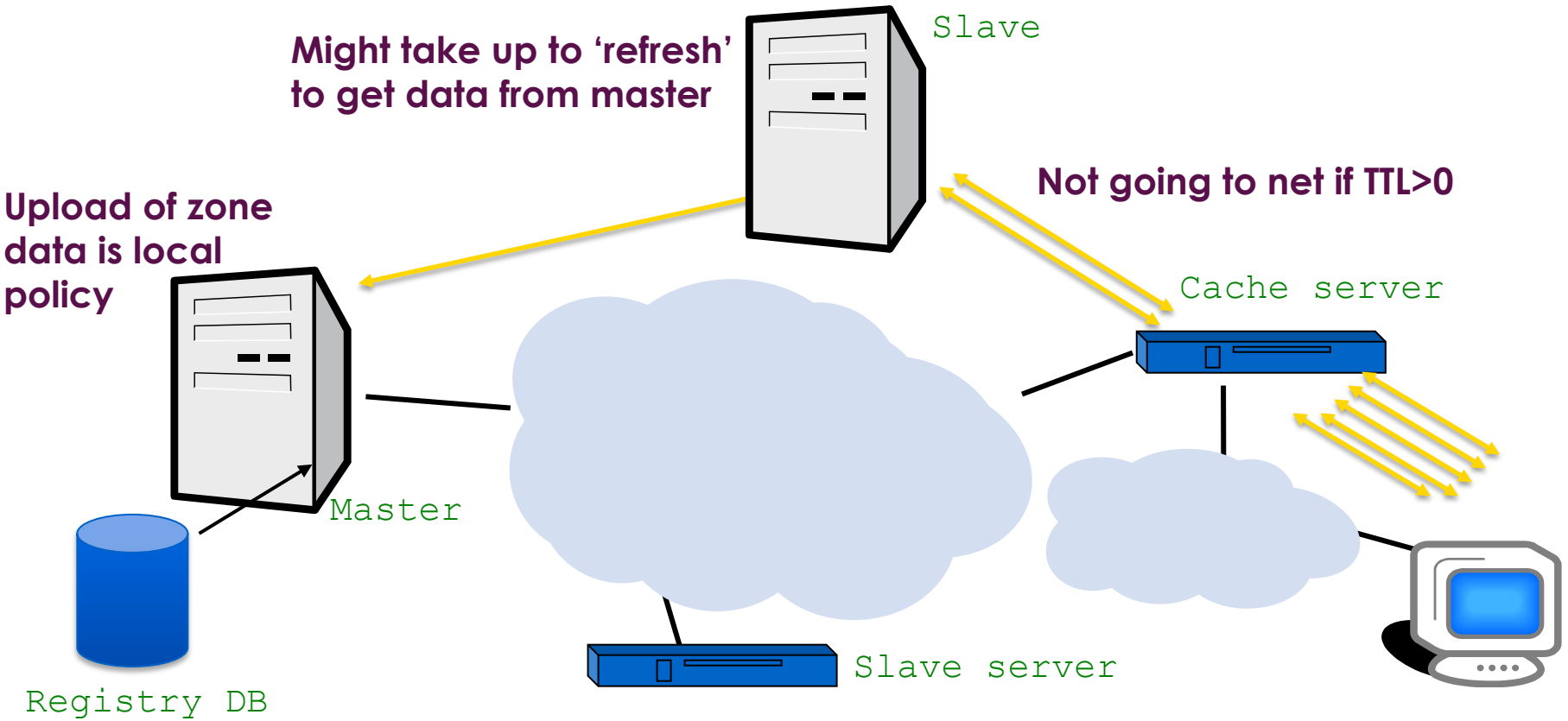
```
apnic.net. 7200 IN      SOA     ns.apnic.net. admin.apnic.net. (
                                2013071001          ; Serial
                                12h                 ; Refresh 12 hours
                                4h                  ; Retry 4 hours
                                4d                  ; Expire 4 days
                                2h                  ; Negative cache 2 hours )
```

```
apnic.net.              7200    IN      NS      ns.apnic.net.
apnic.net.              7200    IN      NS      ns.ripe.net.
whois.apnic.net.        3600    IN      A       193.0.1.162
www.apnic.net           3600    IN      A       192.0.3.25
```



Places where DNS data lives

Changes do not propagate instantly



Delegating a Zone

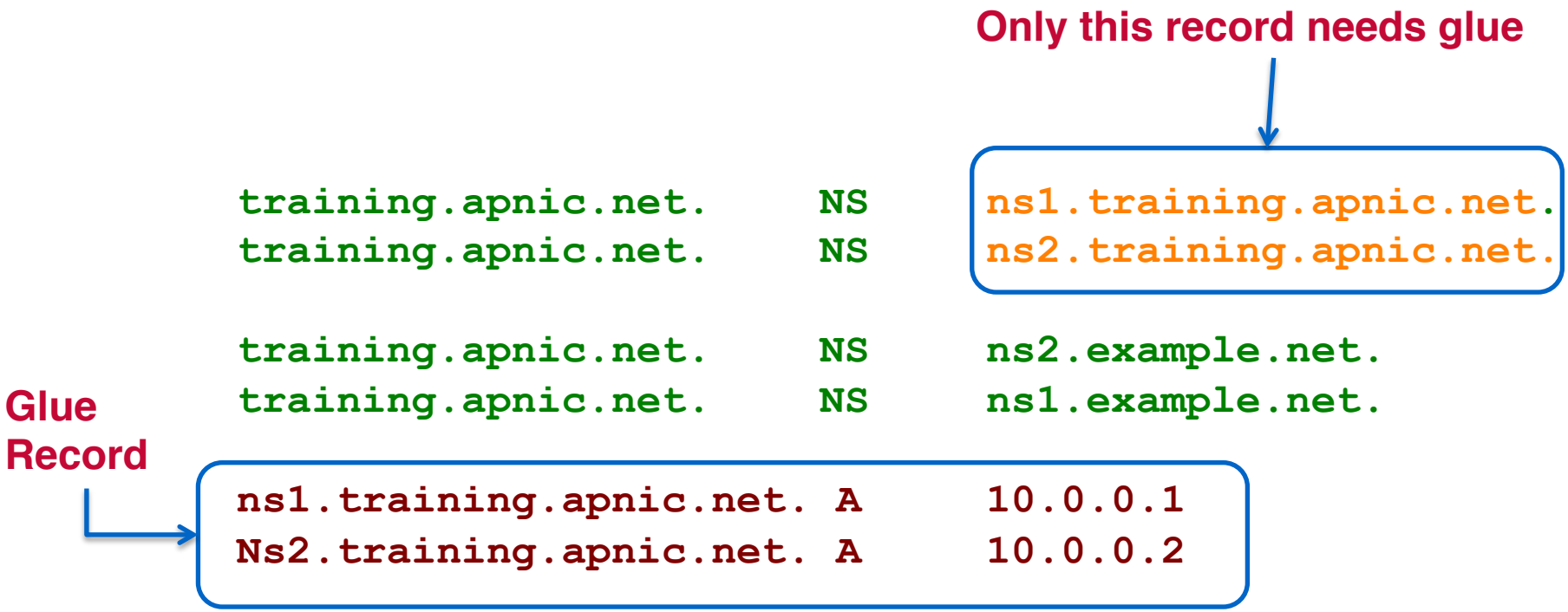
- Delegation is passing of authority for a subdomain to another party
- Delegation is done by adding NS records
  - Ex: if APNIC.NET wants to delegate TRAINING.APNIC.NET

```
training.apnic.net.        NS ns1.training.apnic.net.
training.apnic.net.        NS ns2.training.apnic.net.
```

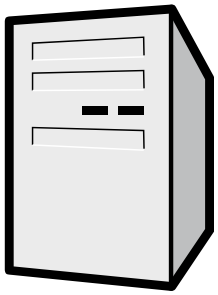
- Now how can we go to ns1 and ns2?
  - We must add a **Glue Record**

Glue Record

- Glue is a ‘non-authoritative’ data
- Don’t include glue for servers that are not in the sub zones

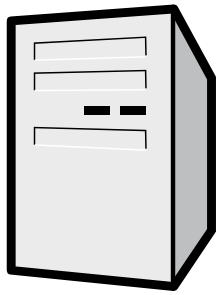


Delegating training.apnic.net. from apnic.net.



ns.apnic.net

1. Add NS records and glue
2. Make sure there is no other data from the training.apnic.net. zone in the zone file



ns.training.apnic.net

1. Setup minimum two servers
2. Create zone file with NS records
3. Add all training.apnic.net data

Remember ...

- Multiple authoritative servers to distribute load and risk:
  - Put your name servers apart from each other

- Caches to reduce load to authoritative servers and reduce response times

- SOA timers and TTL need to be tuned to needs of the zone
  Stable data: higher numbers

Performance of DNS

- Server hardware requirements
- OS and the DNS server running
- How many DNS servers?
- How many zones are expected to load?
- How large are the zones?
- Zone transfers
- Where are the DNS servers located?
- Bandwidth

Performance of DNS

- Are these servers Multihomed?
- How many interfaces are to be enabled for listening?
- How many queries are expected to receive?
- Recursion
- Dynamic updates?
- DNS notifications

Questions

DNS BIND

APNIC

DNS Software

- DNS BIND – authoritative + recursive server
- Unbound - caching DNS resolver
- NSD – authoritative only nameserver
- Microsoft DNS – provided with the Windows Server
- Knot DNS – authoritative only nameserver
- PowerDNS – data storage backends

BIND

- Berkeley Internet Name Domain
- The most widely-used open source DNS software on the Internet
- Maintained by the Internet Systems Consortium (ISC)
- Bind 10 is in development
  - New architecture
  - Bind 10.1.1 released on June 06 2013

Where to Get BIND

- From the ISC website
  - http://www.isc.org
  - ftp://ftp.isc.org/isc/bind9
- Other packages that are necessary
  - OpenSSL (for DNSSEC)

Unpacking BIND9

- When installing BIND from source, decompress the gzip file

```
tar xvfz bind-9.9.3-P1.tar.gz
cd bind-9.9.3-P1
```

- What's in there?
  - A lot of stuff (dig, libraries etc)
  - configure (script)
  - Administrator's Reference Manual
    - doc/arm/Bv9ARM.html

Building BIND9

- must be in the BIND 9.9.2 directory
- Determine the appropriate includes and compiler settings

```
./configure --with-openssl
```

- Build and compile

```
make
```

- Install the BIND package

```
make install
```

- Verify the installation

```
which named
named -v
```

Location of Executables

- Executables
  - `/usr/local/sbin`
    - named
    - dnssec-keygen, dnssec-makekeyset, dnssec-signkey, dnssec-signzone
    - lwresd, named-checkconf, named-checkzone
    - rndc, rndc-confgen
  - `/usr/local/bin`
    - dig
    - host, isc-config.sh, nslookup
    - nsupdate
- And libraries included
- Documentation in <src>/doc/arm/Bv9ARM.html

Named Configuration

- The BIND configuration file is called “named.conf”
  - Default location is in /etc/named.conf
- Defines the zones and corresponding zonefile
- Turn on logging for troubleshooting
  - Several categories
  - Categories are processed in one or more channels
  - Channels specify where the output goes

Named Configuration

- BIND Configuration file
- Option statement contains all global configuration options to be used as defaults by named.

```
options {
    directory "/var/named/recursive"; };
```

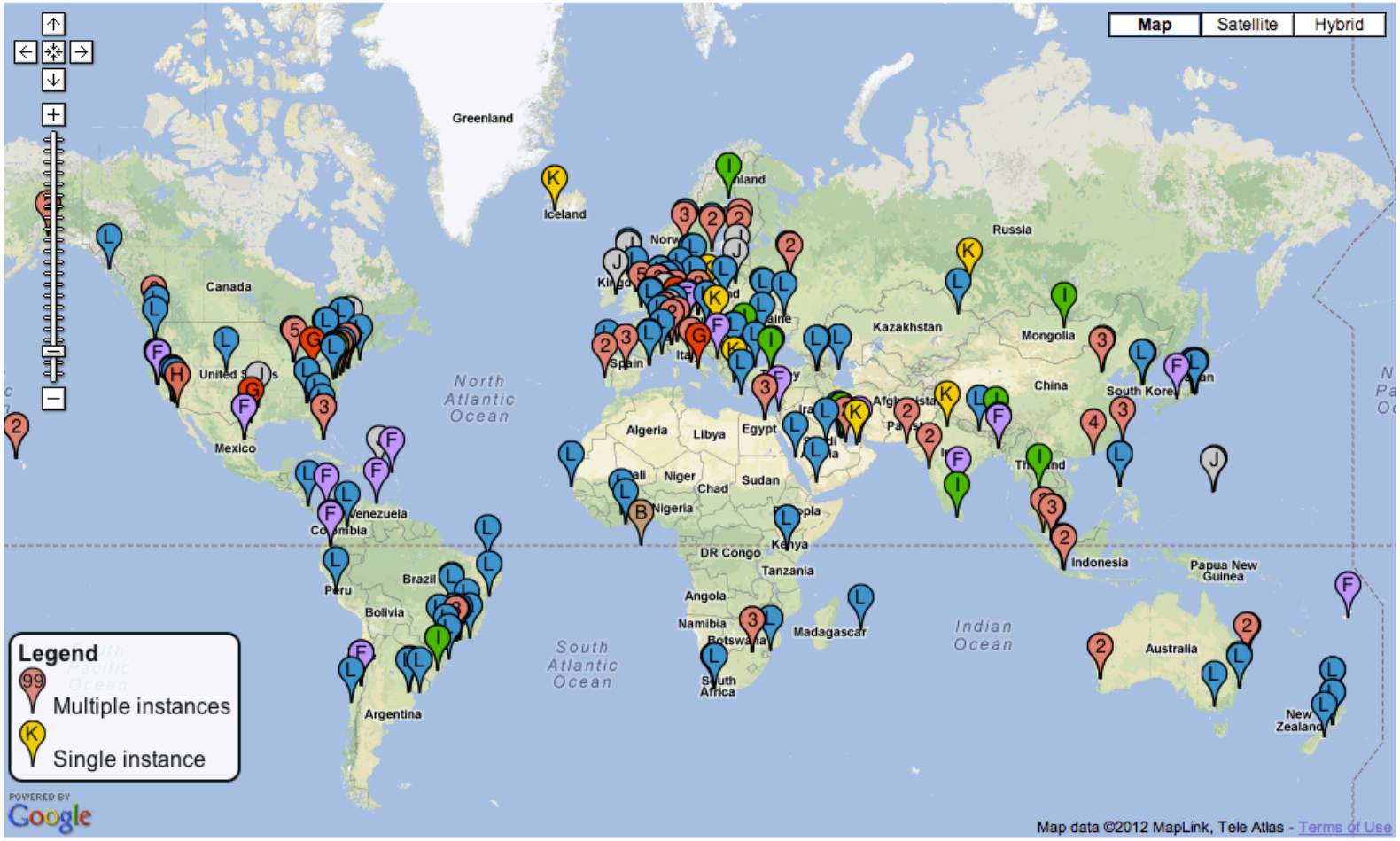
- Zone statement defines the zones
  - For a simple caching server, the zone statement defines
    - Root Hints
    - Forward dns for localhost
    - Reverse dns for loopback zones

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

Root Servers

- The top of the DNS hierarchy
- There are 13 root name servers operated around the world, with names from `[a-m].root-servers.net`
- There are more than 13 physical root name servers
  - Each rootserver has an instance deployed via anycast
- Root hints file come in many names (db.cache, named.root, named.cache, named.ca)
  - Get it from ftp.rs.internic.net
- See root-servers.org for more detail

Root Servers



What it looks like

ftp://ftp.rs.internic.net/domain/

```
.                       3600000   IN   NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000        A     198.41.0.4
A.ROOT-SERVERS.NET.     3600000        AAAA  2001:503:BA3E::2:30
; operated by WIDE
.                       3600000        NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.     3600000        A     202.12.27.33
M.ROOT-SERVERS.NET.     3600000        AAAA  2001:dc3::35
```

Named Configuration (Recursive Server)

- The recursive server needs to know how to reach the top of the DNS hierarchy
- It should also stop some queries such as those for localhost (127.0.0.1)
- The following files are required to run a recursive/caching server:
  - named.conf
  - root.hints
  - localhost zone (db.localhost)
  - 0.0.127.in-addr.arpa zone (db.127.0.0.1)
  - ::1 IPv6 reverse zone (db.ip6)

Zones Defined in a Recursive Server

- Loopback name in operating systems
  - Queries for this shouldn't use recursion
  - So we will configure a file to define the localhost zone
  - Localhost will map to 127.0.0.1 and ::1
  - **db.localhost**

```
zone "localhost" {
    type master;
    file db.localhost; };
```

- Reverse zone for the loopback
  - We need a reverse zone that maps 127.0.0.1 (and ::1) to localhost
  - **db.127.0.0.1**

```
zone "0.0.127.in-addr.arpa" {
    type master;
    file db.127.0.0.1;
    };
```

Example named.conf

```
options {
        directory "/var/named/
recursive";
        recursion yes;
};
zone "." {
        type hint;
        file "named.root";
};
```

```
zone "localhost." {
        type master;
        file "localhost";
};
zone "0.0.127.in-
addr.arpa." {
        type master;
        file "0.0.127.in-
addr.arpa";
};
```

Zone Files

- Contain the resource records defined in a particular zone
- A zone file begins with a Start of Authority Record (SOA)

```
@       SOA   localhost.  root.localhost.   (
                          20121115 ;serial no.
                          30m      ;refresh
                          15m      ;retry
                          1d       ;expire
                          30m      ;negative cache ttl )
```

- Common Zone File directives
  - $ORIGIN
  - $INCLUDE
  - $TTL
  - @ represents the current origin

Start of Authority (SOA) record

```
Domain_name. CLASS  SOA  hostname.domain.name. mailbox.domain.name (
                                                Serial Number
                                                Refresh
                                                Retry
                                                Expire
                                                Minimum TTL )
```

- **Serial Number** – must be updated if any changes are made in the zone file
- **Refresh** – how often a secondary will poll the primary server to see if the serial number for the zone has increased
- **Retry** - If a secondary was unable to contact the primary at the last refresh, wait the retry value before trying again
- **Expire** - How long a secondary will still treat its copy of the zone data as valid if it can't contact the primary.
- **Minimum TTL** - The default TTL (time-to-live) for resource records

TTL Time Values

- The right value depends on your domain
- Recommended time values for TLD (based on RFC 1912)

Refresh	86400 (24h)
Retry	7200 (2h)
Expire	2592000 (30d)
Min TTL	345600 (4d)

- For other servers – optimize the values based on
  - Frequency of changes
  - Required speed of propagation
  - Reachability of the primary server
  - (and many others)

localhost file

```
$TTL 86400
@       IN      SOA localhost. root.localhost. (
                      20121115   ; serial
                      1800       ; refresh
                      900        ; retry
                      69120      ; expire
                      1080       ; negative ttl
                           )
                NS    localhost.
                A     127.0.0.1
```

0.0.127.in-addr.arpa file

```
$TTL 86400
@         IN      SOA localhost.  root.localhost.  (
                        20121115   ; serial
                        1800       ;refresh
                        900        ;retry
                        69120      ;expire
                        1080       ;negative ttl
                        )

                  NS   localhost.
               1      PTR  localhost.
```

Assembling the files

- Create a directory in /var/named/

```
ls
0.0.127.in-addr.arpa   localhost   root.hints
```

- The directory name and file names will be defined in named.conf
- Now create a named.conf file in the same directory

Running the server

- From the directory

```
named -g -c named.conf

where:
      -c  path to the configuration file
      -g  run in the foreground
```

Testing the server

- **dig @127.0.0.1 www.google.com**

```
; <<>> DiG 9.8.3-P1 <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 213
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         156     IN      A       74.125.237.115
www.google.com.         156     IN      A       74.125.237.113
www.google.com.         156     IN      A       74.125.237.116
www.google.com.         156     IN      A       74.125.237.114
www.google.com.         156     IN      A       74.125.237.112

;; Query time: 27 msec
;; SERVER: 127.0.0.1#53(203.119.98.119)
;; WHEN: Thu Jul 11 13:46:29 2013
;; MSG SIZE  rcvd: 112
```

Questions

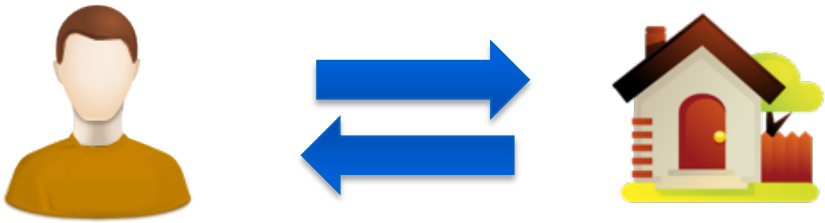
Reverse DNS

APNIC

What is ‘Reverse DNS’?

- ‘Forward DNS’ maps names to numbers
  - svc00.apnic.net ➔202.12.28.131

- ‘Reverse DNS’ maps numbers to names
  - 202.12.28.131 ➔ svc00.apnic.net

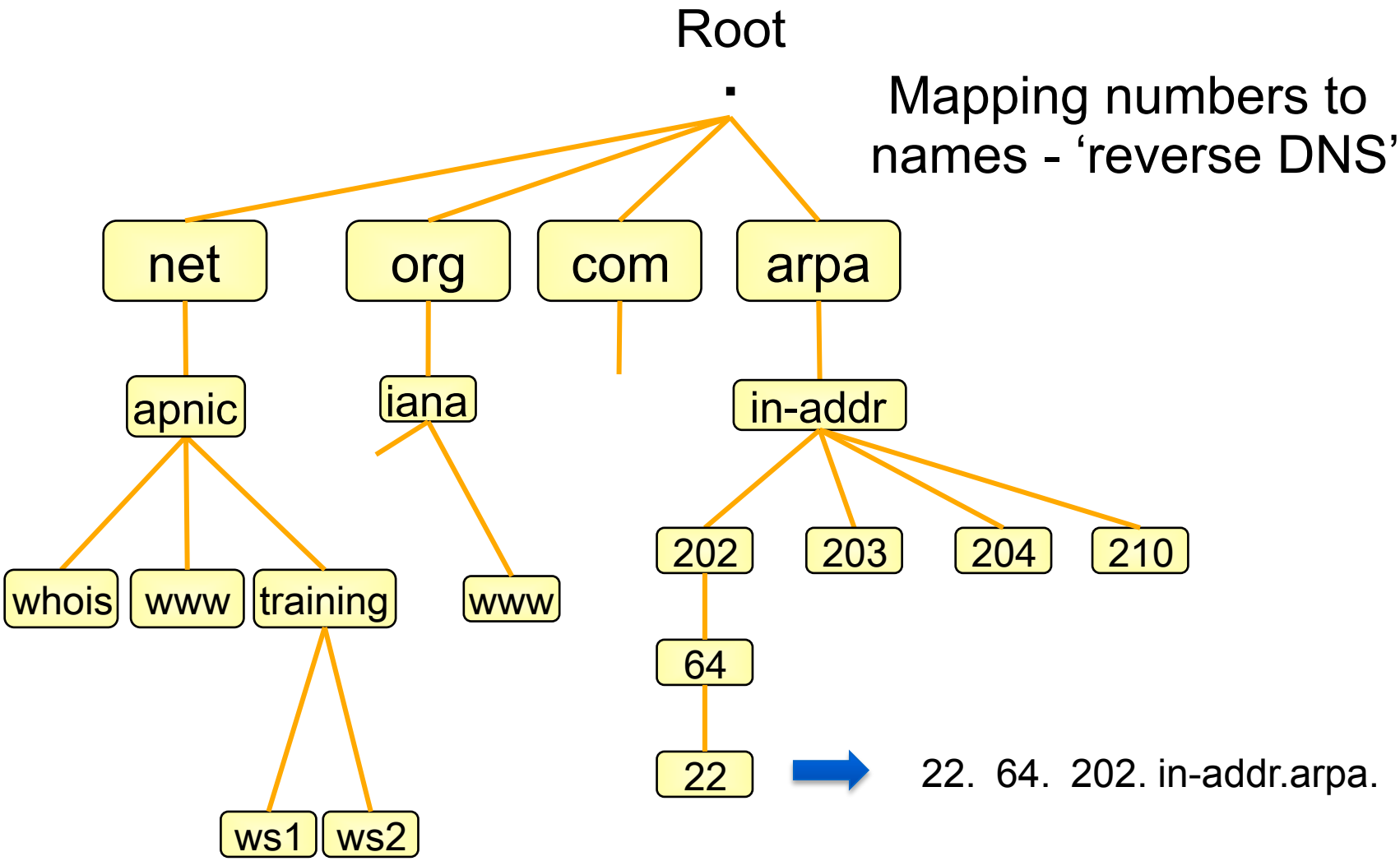


Person (Host) Address (IPv4/IPv6)

Reverse DNS - why bother?

- Service denial
  - only allow access when fully reverse delegated
  - Example: anonymous ftp
- Diagnostics
  - Assisting in trace routes etc
- SPAM identifications
  - Failed reverse lookup results in a spam penalty score
- Registration responsibilities
  - APNIC members must make sure that all their address space are properly reverse delegated

Principles – DNS Tree



Creating Reverse Zones

- Same as creating a forward zone file
  - SOA and initial NS records are the same as normal zone
- Main difference
  - need to create additional PTR records
- Can use BIND or other DNS software to create and manage reverse zones
  - Details can be different
- In addition to the forward zone files, you need the reverse zone files
  - Ex: for a reverse zone on a 203.176.189.0/24 block, create a zone file and name it as “db.203.176.189” (make it descriptive)

Pointer (PTR) Records

- Create pointer (PTR) records for each IP address

```
131.28.12.202.in-addr.arpa. IN PTR svc00.apnic.net.
```

or

```
131             IN      PTR             svc00.apnic.net.
```

Reverse Zone Example

```
$ORIGIN 1.168.192.in-addr.arpa.
@       3600   IN SOA test.company.org. (
                      sys\.admin.company.org.
                      2002021301    ; serial
                      1h            ; refresh
                      30M           ; retry
                      1W            ; expiry
                      3600 )        ; neg. answ. ttl

        NS      ns.company.org.
        NS      ns2.company.org.

1       PTR     gw.company.org.
                router.company.org.

2       PTR     ns.company.org.
```

Reverse Delegation Requirements

- /24 Delegations
  - Address blocks should be assigned/allocated
  - At least two name servers
- /16 Delegations
  - Same as /24 delegations
  - APNIC delegates entire zone to member
- < /24 Delegations
  - Read “Classless IN-ADDR.ARPA delegation” (RFC 2317)

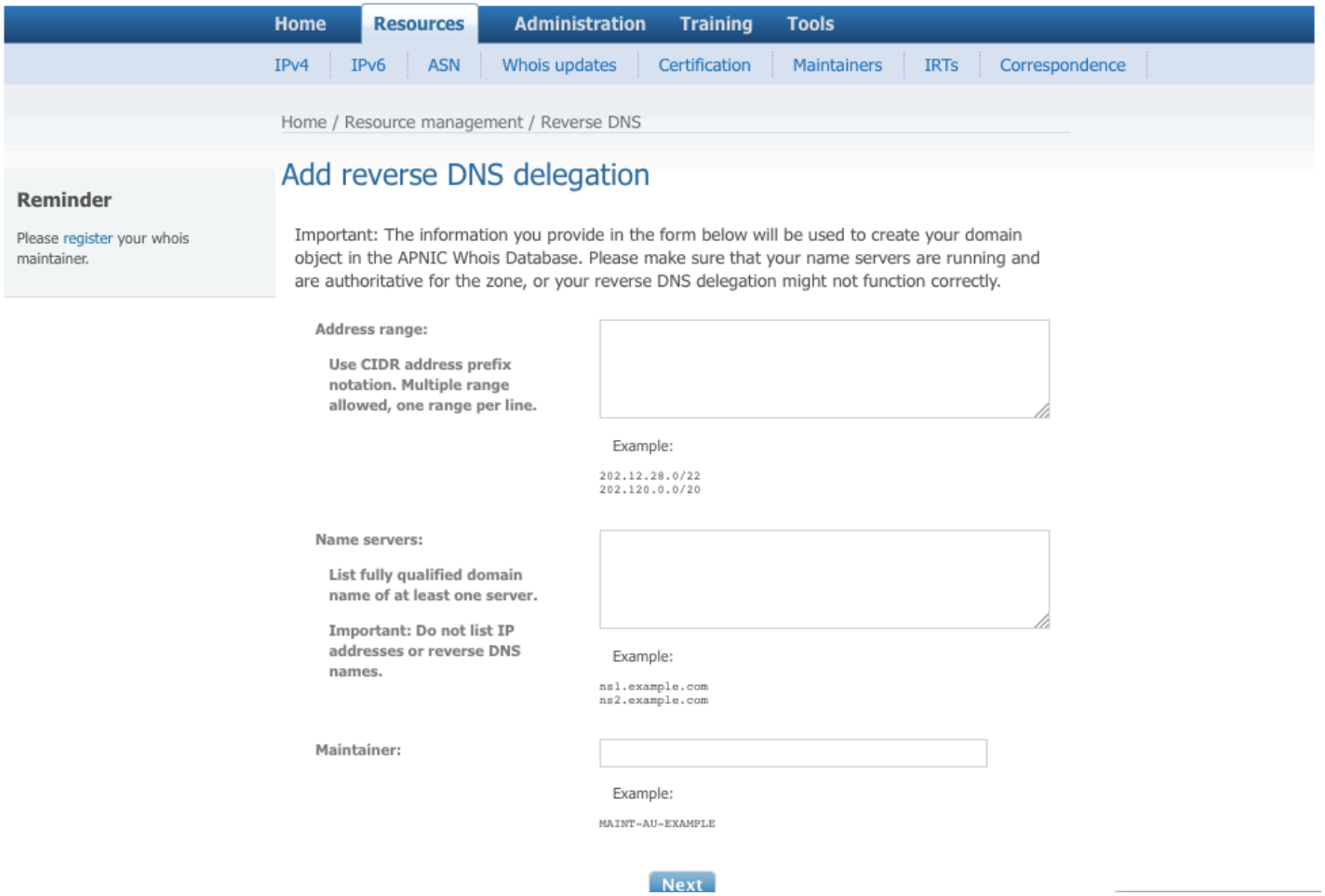
APNIC & ISPs responsibilities

- APNIC
  - Manage reverse delegations of address block distributed by APNIC
  - Process organisations requests for reverse delegations of network allocations
- Organisations
  - Be familiar with APNIC procedures
  - Ensure that addresses are reverse-mapped
  - Maintain nameservers for allocations
  - Minimise pollution of DNS

Reverse Delegation Procedures

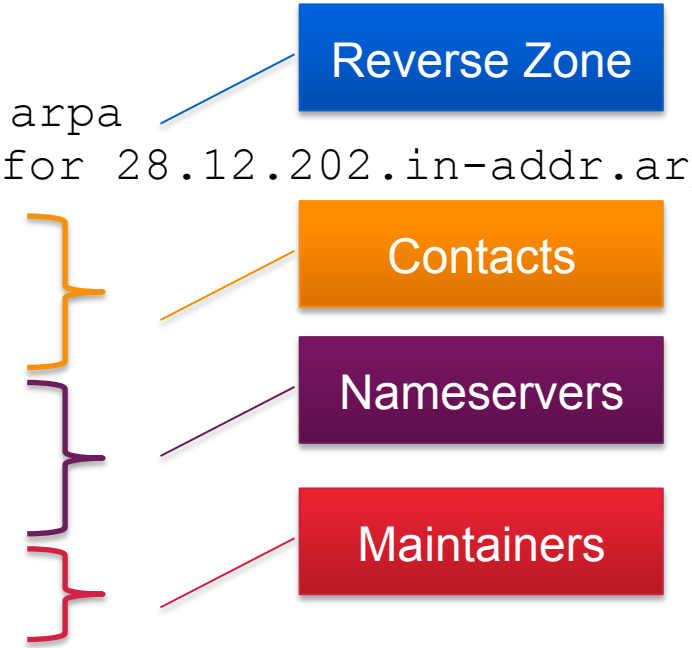
- Standard APNIC database object
  - can be updated through MyAPNIC
- Nameserver/domain set up verified before being submitted to the database.
- Protection by maintainer object
  - (current auths: CRYPT-PW, PGP).
- Any queries
  - Contact helpdesk@apnic.net

Reverse Delegation Procedures



Whois domain object

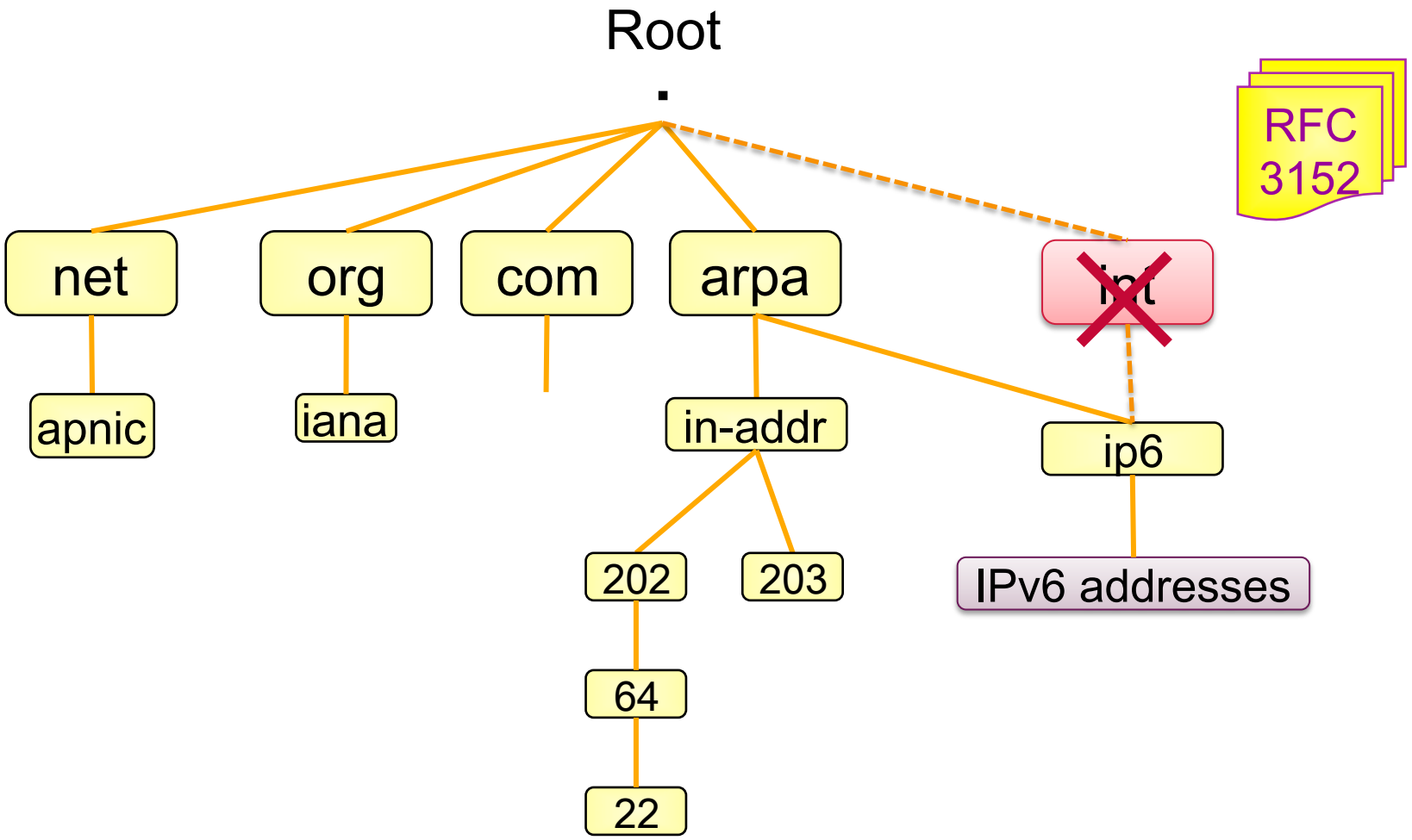
```
domain:     28.12.202.in-addr.arpa
Descr:      in-addr.arpa zone for 28.12.202.in-addr.arpa
admin-c:    NO4-AP
tech-c:     AIC1-AP
zone-c:     NO4-AP
nserver:    cumin.apnic.net
nserver:    tinnie.apnic.net
nserver:    tinnie.arin.net
mnt-by:     MAINT-APNIC-AP
mnt-lower:  MAINT-AP-DNS
changed:    inaddr@apnic.net 20021023
changed:    inaddr@apnic.net 20040109
changed:    hm-changed@apnic.net 20091007
changed:    hm-changed@apnic.net 20111208
source:     APNIC
```



IPv6 Reverse Delegations

APNIC

Reverse DNS Tree – with IPv6



IPv6 Representation in the DNS

- Forward lookup support: Multiple RR records for name to number
  - AAAA (Similar to A RR for IPv4)

- Reverse lookup support:
  - Reverse nibble format for zone ip6.arpa

IPv6 Reverse Lookups – PTR records

- Similar to the IPv4 reverse record

```
b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.3.4.ip6.arpa.
        IN      PTR test.ip6.example.com.
```

- Example: The reverse name lookup for a host with address 3ffe:8050:201:1860:42::1

```
$ORIGIN 0.6.8.1.1.0.2.0.0.5.0.8.e.f.f.3.ip6.arpa.

1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.4.0.0  14400  IN PTR
host.example.com.
```

IPv6 forward and reverse mappings

- Existing A record will not accommodate the 128 bit addresses for IPv6
- BIND expects an A record's record-specific data to be a 32-bit address (in dotted-octet format)
- An address record
  - AAAA (RFC 1886)
- A reverse-mapping domain
  - ip6.arpa

IPv6 forward lookups

- Multiple addresses possible for any given name
  - Ex: in a multi-homed situation
- Can assign A records and AAAA records to a given name/ domain
- Can also assign separate domains for IPv6 and IPv4

Sample forward lookup file

```
;; domain.edu
$TTL            86400
@     IN       SOA     ns1.domain.edu. root.domain.edu. (
                20121115        ; serial - YYYYMMDDXX
                21600           ; refresh - 6 hours
                1200            ; retry - 20 minutes
                3600000         ; expire - long time
                86400)          ; minimum TTL - 24 hours
;; Nameservers
                IN      NS      ns1.domain.edu.
                IN      NS      ns2.domain.edu.

;; Hosts with just A records
host1           IN      A       1.0.0.1

;; Hosts with both A and AAAA records
host2           IN      A       1.0.0.2
                IN      AAAA    2001:468:100::2
```

Sample reverse lookup file

```
;; 0.0.0.0.0.0.1.0.8.6.4.0.1.0.0.2.rev
;; These are reverses for 2001:468:100::/64)
;; File can be used for both ip6.arpa and ip6.int.
$TTL            86400
@     IN      SOA     ns1.domain.edu. root.domain.edu. (
                            2002093000              ; serial - YYYYMMDDXX
                            21600           ; refresh - 6 hours
                            1200            ; retry - 20 minutes
                            3600000         ; expire - long time
                            86400)          ; minimum TTL - 24 hours
;; Nameservers
                    IN      NS      ns1.domain.edu.
                    IN      NS      ns2.domain.edu.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0    IN      PTR     host1.ip6.domain.edu
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0    IN      PTR     host2.domain.edu
;;
;; Can delegate to other nameservers in the usual way
;;
```

Questions

Troubleshooting

APNIC

Why Troubleshoot?

- What Can Go Wrong?
  - Misconfigured zone
  - Misconfigured server
  - Misconfigured host
  - Misconfigured network

Tools

- BIND Logging Facility
- named's built-in options
- ping and traceroute
- tcpdump and wireshark
- dig and nslookup

The Best Way To Handle Mistakes

- Assume You Will Make Them
- Prepare The Name Server via Logging

BIND Logging

- Telling named which messages to send
  - category specification
- Telling named where to send messages
  - channel specification

BIND channels

- BIND can use syslog
- BIND can direct output to other files
  - Example:

```
channel my_dns_log {
file "seclog" versions 3 size 10m;
print-time yes;
print-category yes;
print-severity yes;
severity debug 3;
};
```

BIND Categories

- BIND has many categories
- Short descriptions of each can be found in the Administrator's Reference Manual (ARM)
  - Example:

```
category queries { my_dns_log; };
```

So You've Set Up A Server

- What testing should be done?
- From Basic liveness
  - Is the (right) server running?
  - Is the machine set up correctly?
- To data being served
  - Has the zone loaded?
  - Have zone transfers happened?

Checking the Configuration

- To see named start, use the -g flag
  - Keeps named process in the foreground
  - Prints some diagnostics
  - But does not execute logging
- When satisfied with named's start, kill the process and start without –g flag
- Other option
  - `% named-checkconf`
  - checks syntax only

Is the Server Running?

- Once the name server is thought to be running, make sure it is

```
% dig @127.0.0.1 version.bind chaos txt
```

- This makes the name server do the simplest lookup it can - its version string
- This also confirms which version you started
  - Common upgrade error: running the old version, forgetting to 'make install'

Is the Server Data Correct?

- Now that the server is the right one (executable)

```
% dig @127.0.0.1 <zone> soa
```

- Check the serial number to make sure the zone has loaded
- Also test changed data in case you forgot to update the serial number
- When we get to secondary servers, this check is made to see if the zone transferred

Is the Server Reachable?

- If the dig tests fail, its time to test the environment (machine, network)

```
% ping <server machine ip address>
```

- This tests basic network flow, common errors
  - Network interface not UP
  - Routing to machine not correct

- Pinging 'locally' is useful, believe it or not
  - Confirms that the IP address is correctly configured

Is the Server Listening?

- If the server does not respond, but machine responds to ping
  - look at system log files
  - telnet server 53
  - firewall running?

- Server will run even if it can't open the network port
  - logs will show this
  - telnet opens a TCP connection, tests whether port was opened at all

Using the Tools

- named itself
- dig/nslookup
- host diagnostics
- packet sniffers

Built in to named

- named -g to retain command line
  - named -g -c <conf file>
  - keeps named in foreground
- named -d <level>
  - sets the debug output volume
  - <level>'s aren't strictly defined
  - -d 3 is popular, -d 99 gives a lot of detail

dig

- domain internet groper
  - already used in examples
  - best tool for testing
  - shows query and response syntax
  - documentation

```
% man dig
% dig -help
```

- Included in named distribution

Flags

Flags	Meaning
AA	Authoritative answer
RD	Recursion desired
RA	Recursion Available
AD	Authenticated Data (DNSSEC only)
CD	

Status	Response Code
0 - NOERR	No error
1 - FORMERR	Format error
2 - SERVFAIL	Nameserver unreachable
3 - NXDOMAIN	Domain name not existing
4 - NOTIMPL	Not implemented
5 - REFUSED	Request refused

Non-BIND Tools

- Tools to make sure environment is right
  - Tools to look at server machine
  - Tools to test network
  - Tools to see what messages are on the network

ifconfig

- InterFace CONFIGuration

  ```
  % ifconfig -a
  ```

  – shows the status of interfaces
  – operating system utility
- Warning, during boot up, ifconfig may configure interfaces after named is started
  – named can't open delayed addresses
- Documentation

  ```
  % man ifconfig
  ```

ping

- Checks routing, machine health
  - Most useful if run from another host
  - Could be reason "no servers are reached"
  - Can be useful on local machine - to see if the interface is properly configured

traceroute

- If ping fails, traceroute can help pinpoint where trouble lies
  - the problem may be routing
  - if so - it's not named that needs fixing!
  - but is it important to know...

tcpdump and wireshark

- Once confident in the environment, problems with DNS setup may exist
- To see what is happening in the protocol, use traffic sniffers
- These tools can help debug "forwarding" of queries

Address Match Lists

APNIC

Elements in an address match list

- Individual IP addresses
- Addresses/netmask pairs
- Names of other ACLs
- In some contexts, key names

Purposes in Bind

- Restricting queries & zone xfer
- Authorizing dynamic updates
- Selecting interfaces to listen on
- Sorting responses

*Address match lists are always enclosed in curly braces.

Notes on Address Match list

- Elements must be separated by “ ; ”
- The list must be terminated with a “ ; ”
- Elements of the address match list are checked sequentially.
- To negate elements of the address match list prepend them with “!”
- Use *acl* statement to name an address match list.
- *acl* must be define before it can be used elsewhere.

Example: Address match lists

- For network 192.168.0.0 255.255.255.0
  { 192.168.0.0/24; }
- For network plus loopback
  { 192.168.0.0/24; 127.0.0.1; }
- Addresses plus key name
  { 192.168.0.0/24; 127.0.0.1; tequila.apnic.net;}

The *acl* Statement

- Syntax:

  acl <acl name> { address match list>};

- Example:

  acl internal { 127.0.0.1; 192.168.0/24; };

  acl dynamic-update { key dhcp.apnic.net; };

Notes on the *acl* Statement

- The acl name need not be quoted.
- There are four predefined ACLs:

any	(Any IP address)
none	(No IP address)
localhost	(loopback, 127.0.0.1)
localnets	(all networks the name server is directly connected to)

Blackhole

```
options   {

  blackhole { ACL-name or itemized
 list; };

          };
```

Allow-transfer

```
zone "myzone.example." {
type master;
file "myzone.example.";
allow-transfer { ACL-name or
itemized list; };
     };
```

Allow-Query

```
zone "myzone.example." {
type master;
file "myzone.example.";
allow-query { ACL-name or
itemized list; };
      };
```

Listen-on

```
options {
        listen-on port # { ACL-
        name or itemized list;};
        };
```

Summary

- ACLs and Configuration options can be used to create simple split DNS.
- It is cumbersome and difficult to maintain.
- Good operational practice suggests that ACLs and configuration options be reviewed regularly to ensure that they accurately reflect desired behaviour

Views

The view statement is a powerful new feature of BIND 9 that lets a name server answer a DNS query differently depending on who is asking. It is particularly useful for implementing split DNS setups without having to run multiple servers.

Syntax

- view view_name
  [class] {
  match-clients { address_match_list } ;
  match-destinations {

  address_match_list } ;
  match-recursive-only yes_or_no ;
  [view_option; ...]
  [zone_statement; ...]
  };

Example Config

```
view "internal" {
        // This should match our internal networks.
        match-clients { 10.0.0.0/8; };

        // Provide recursive service to internal clients only.
        recursion yes;

        // Provide a complete view of the example.com zone
        // including addresses of internal hosts.
        zone "example.com" {
                type master;
                file "example-internal.db";
        };
};
```

Continued

```
view "external" {
        // Match all clients not matched by the previous view.
        match-clients { any; };

        // Refuse recursive service to external clients.
        recursion no;

        // Provide a restricted view of the example.com zone
        // containing only publicly accessible hosts.
        zone "example.com" {
            type master;
            file "example-external.db";
        };
    };
```

Questions

DNS Security

APNIC

Cryptography

- Cryptography deals with creating documents that can be shared secretly over public communication channels
- Other terms closely associated
  - Cryptanalysis (breaking an encoded data without the knowledge of the key)
  - Cryptology (combination of cryptography and cryptanalysis)
- Cryptography is a function of plaintext and a cryptographic key

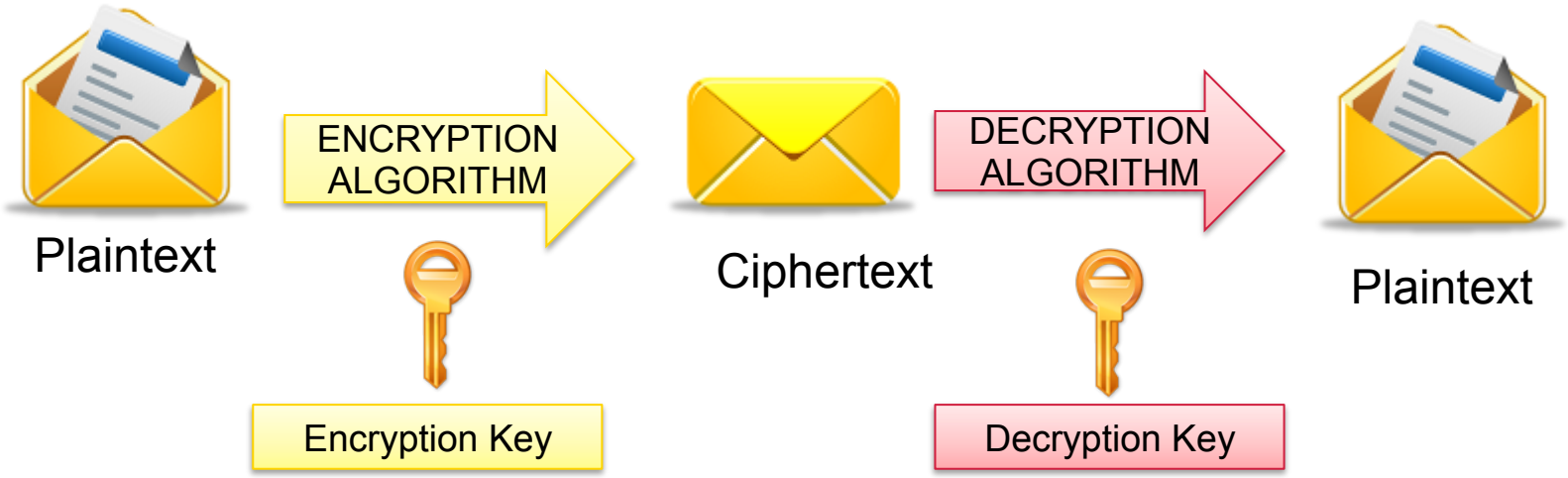
$$C = F(P, k)$$

Notation:
Plaintext (P)
Ciphertext (C)
Cryptographic Key (k)

Encryption

- process of transforming plaintext to ciphertext using a cryptographic key
- Used all around us
  - In Application Layer – used in secure email, database sessions, and messaging
  - In session layer – using Secure Socket Layer (SSL) or Transport Layer Security (TLS)
  - In the Network Layer – using protocols such as IPSec
- Benefits of good encryption algorithm:
  - Resistant to cryptographic attack
  - They support variable and long key lengths and scalability
  - They create an avalanche effect
  - No export or import restrictions
- Two general types:
  - Symmetric and Asymmetric

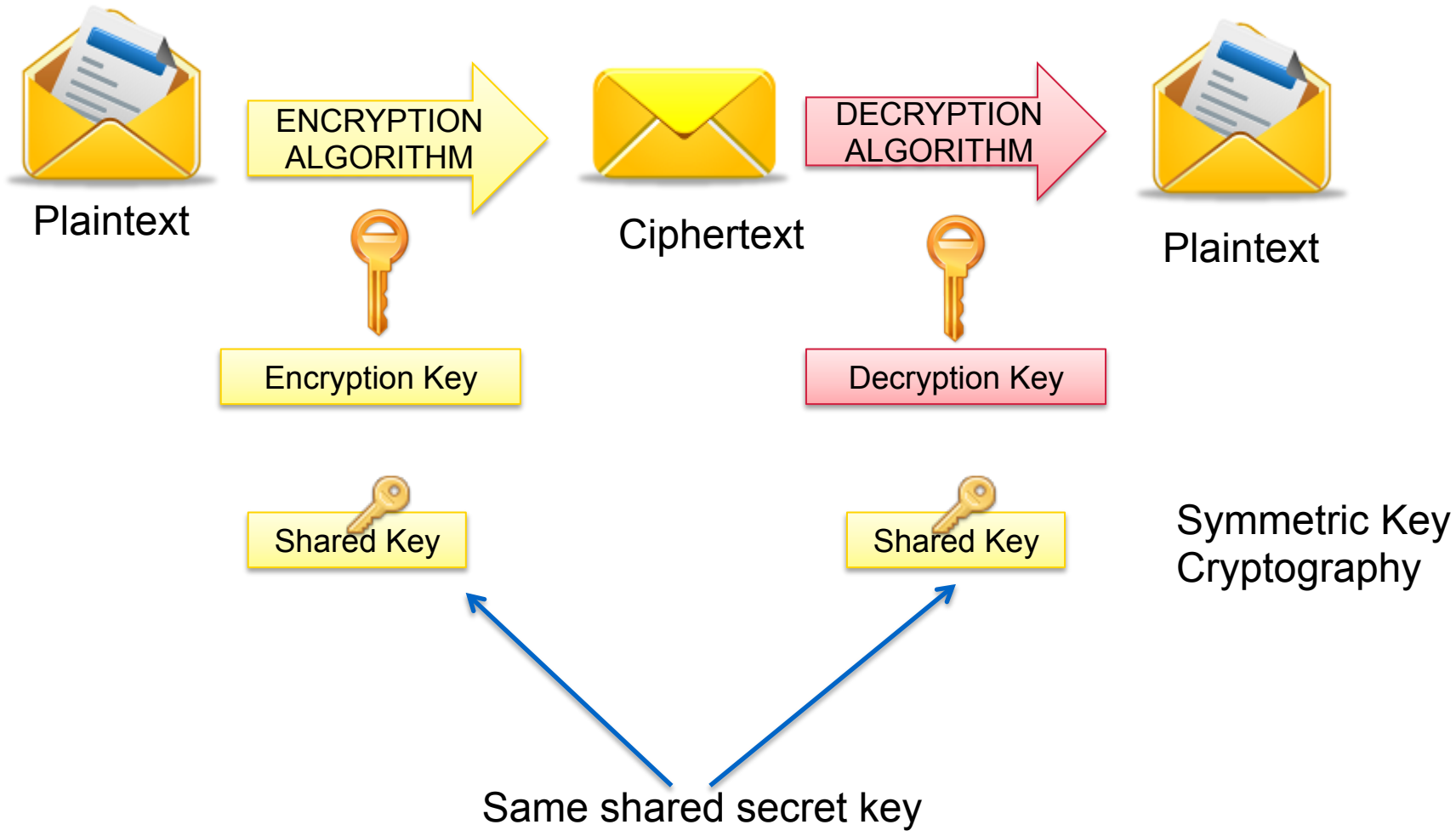
Encryption and Decryption



Symmetric Key Algorithm

- Uses a single key to both encrypt and decrypt information
- Also known as a secret-key algorithm
  - The key must be kept a “secret” to maintain security
  - This key is also known as a private key
- Follows the more traditional form of cryptography with key lengths ranging from 40 to 256 bits.
- Examples of symmetric key algorithms:
  - DES, 3DES, AES, IDEA, RC5, RC6, Blowfish

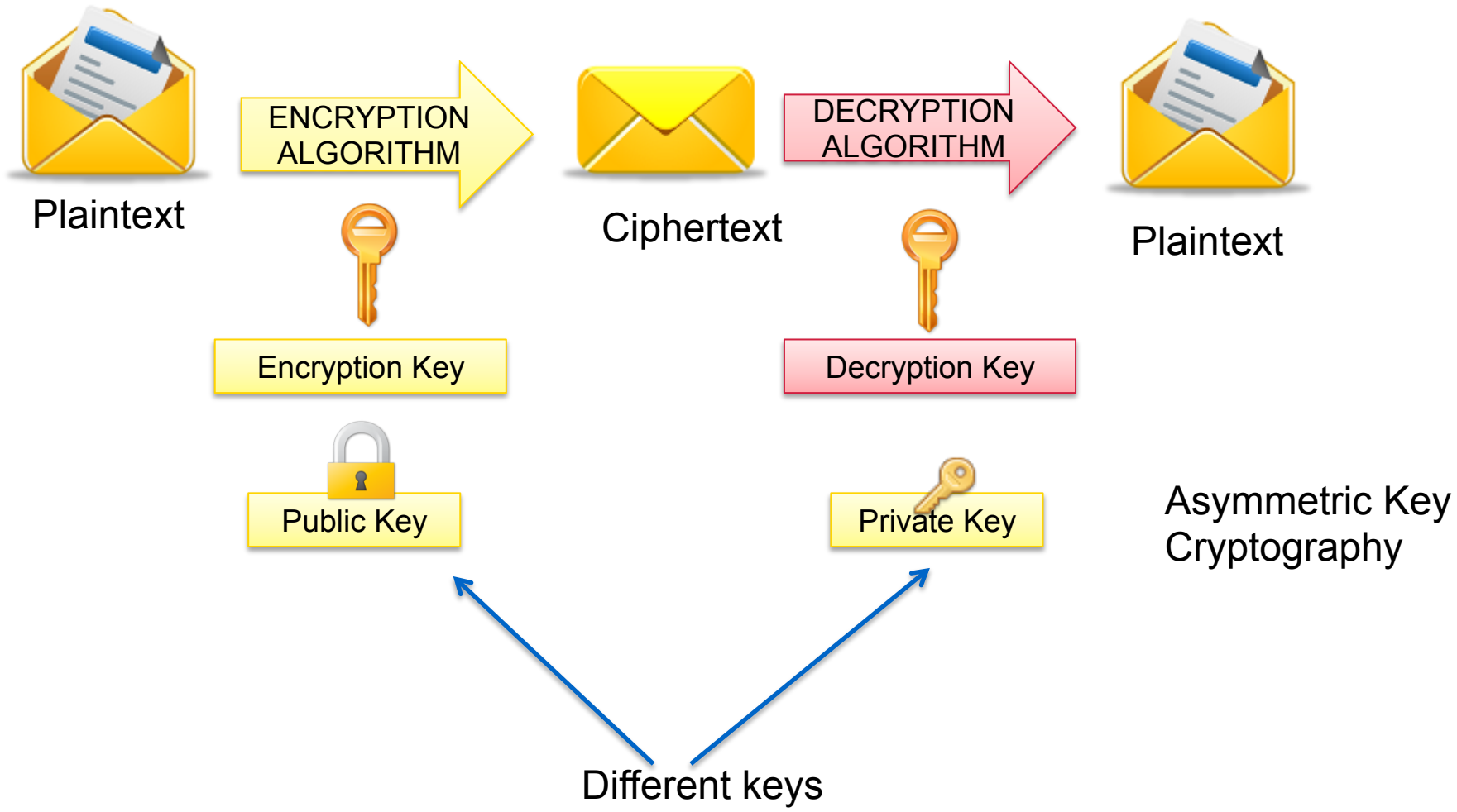
Symmetric Encryption



Symmetric Key Algorithm

- DES – block cipher using shared key encryption, 56-bit
- 3DES (Triple DES) – a block cipher that applies DES three times to each data block
- AES – replacement for DES; it is the current standard
- RC4 – variable-length key, “stream cipher” (generate stream from key, XOR with data)
- RC6
- Blowfish

Asymmetric Encryption



Asymmetric Key Algorithm

- RSA – the first and still most common implementation
- DSA – specified in NIST's Digital Signature Standard (DSS), provides digital signature capability for authentication of messages
- Diffie-Hellman – used for secret key exchange only, and not for authentication or digital signature
- ElGamal – similar to Diffie-Hellman and used for key exchange
- PKCS – set of interoperable standards and guidelines

Symmetric vs. Asymmetric Key

Symmetric	Asymmetric
generally fast Same key for both encryption and decryption	Can be 1000 times slower Uses two different keys (public and private) Decryption key cannot be calculated from the encryption key Key lengths: 512 to 4096 bits Used in low-volume

Hash Functions

- produces a condensed representation of a message (hashing)
- The fixed-length output is called the hash or message digest
- A hash function takes an input message of arbitrary length and outputs fixed-length code. The fixed-length output is called the hash, or the message digest, of the original input message.
- A form of signature that uniquely represents the data
- Uses:
  - Verifying file integrity - if the hash changes, it means the data is either compromised or altered in transit.
  - Digitally signing documents
  - Hashing passwords

Hash Functions

- Message Digest (MD) Algorithm
  - Outputs a 128-bit fingerprint of an arbitrary-length input
  - MD4 is obsolete, MD5 is widely-used
- Secure Hash Algorithm (SHA)
  - SHA-1 produces a 160-bit message digest similar to MD5
  - Widely-used on security applications (TLS, SSL, PGP, SSH, S/MIME, IPsec)
  - SHA-256, SHA-384, SHA-512 are also commonly used, which can produce hash values that are 256, 384, and 512-bits respectively
- RIPEMD
  - Derived from MD4, but performs
  - RIPEMD-160 is the most popular version

Digital Signature

- A digital signature is a message appended to a packet
- The sender encrypts message with own private key instead of encrypting with intended receiver's public key
- The receiver of the packet uses the sender's public key to verify the signature.
- Used to prove the identity of the sender and the integrity of the packet

Digital Signature

- Two common public-key digital signature techniques:
  - RSA (Rivest, Shamir, Adelman)
  - DSS (Digital Signature Standard)
- Used in a lot of things:
  - Email, software distribution, electronic funds transfer, etc
- A common way to implement is to use a hashing algorithm to get the message digest of the data, then use an algorithm to sign the message

Message Authentication Code

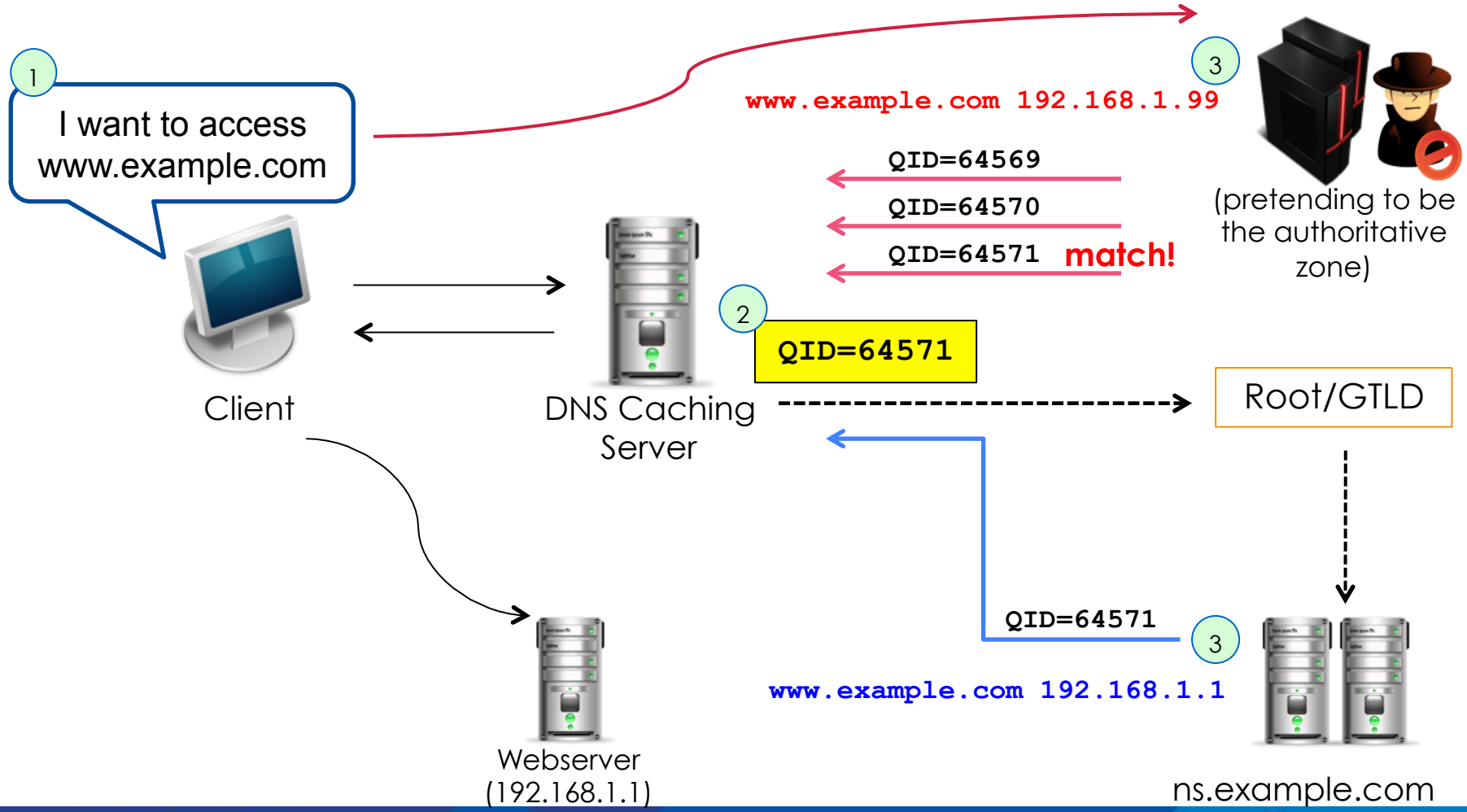
- Message authentication code provides
  - Integrity (checks that data has not been altered)
  - Authenticity (verifies the origin of data)
- In the sender side, the message is passed through a MAC algorithm to get a MAC (also called Tag)
- In the receiver side, the message is passed through the same algorithm. The output is compared with the received tag and should match
- Sender and receiver uses the same secret key
- Hash-based Message Authentication Code (RFC2104)
  - Uses hash function to generate the MAC
  - “HMACs are less affected by collisions than their underlying hashing algorithms alone.”

DNS Security - Background

- The original DNS protocol wasn't designed with security in mind
  - It has very few built-in security mechanism
- As the Internet grew wilder & wollier, IETF realized this would be a problem
  - For example DNS spoofing was to easy
- DNSSEC and TSIG were develop to help address this problem
- Some security problems:
  - Using reverse DNS to impersonate hosts
  - Software bugs (buffer overflows, bad pointer handling)
  - Bad crypto (predictable sequences, forgeable signatures)
  - Cache poisoning (putting inappropriate data into the cache)

https://wiki.tools.isoc.org/DNSSEC_History_Project

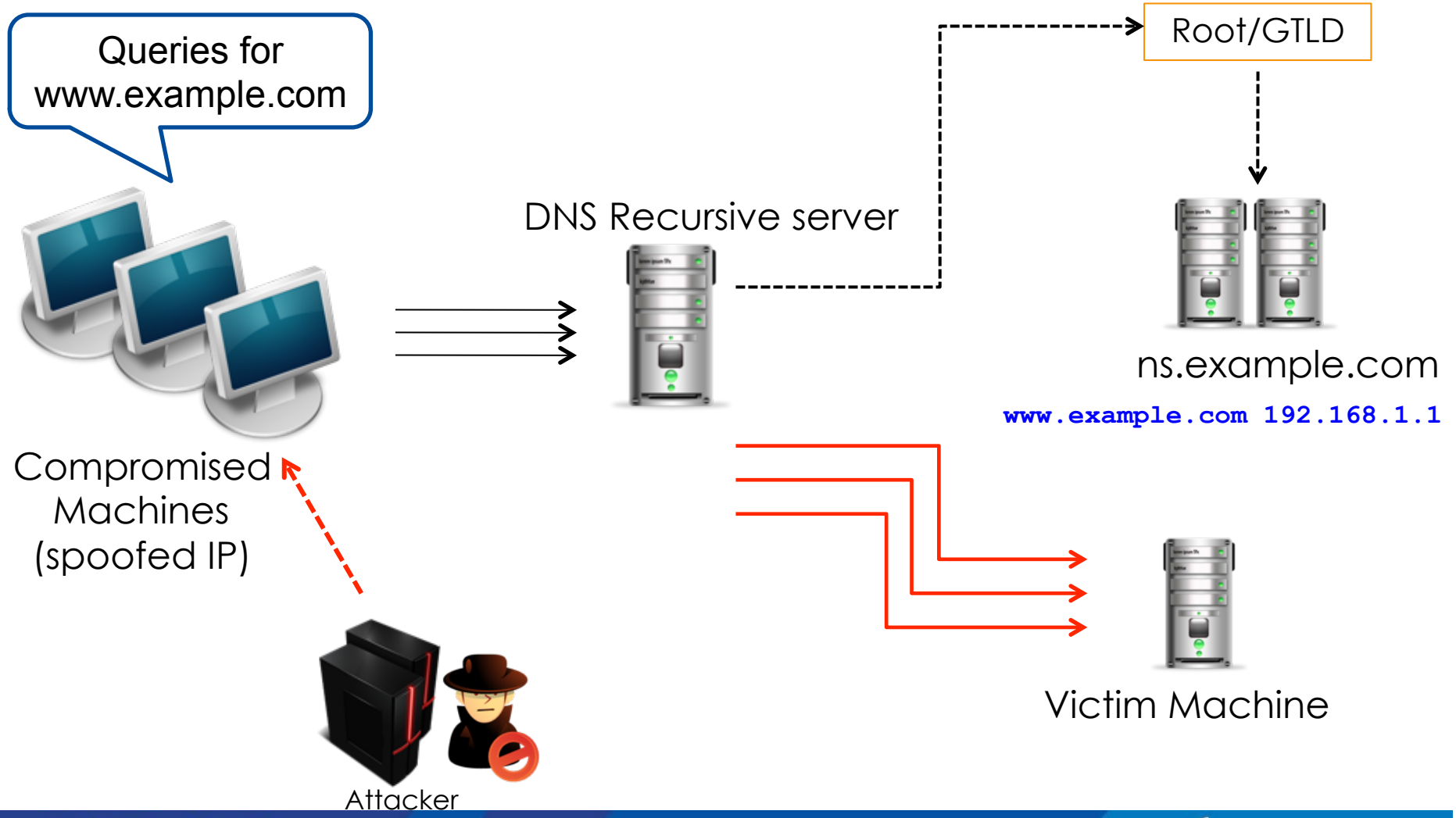
DNS Cache Poisoning



DNS Amplification

- A type of reflection attack combined with amplification
  - Source of attack is reflected off another machine
  - Traffic received is bigger (amplified) than the traffic sent by the attacker
- UDP packet's source address is spoofed

DNS Amplification

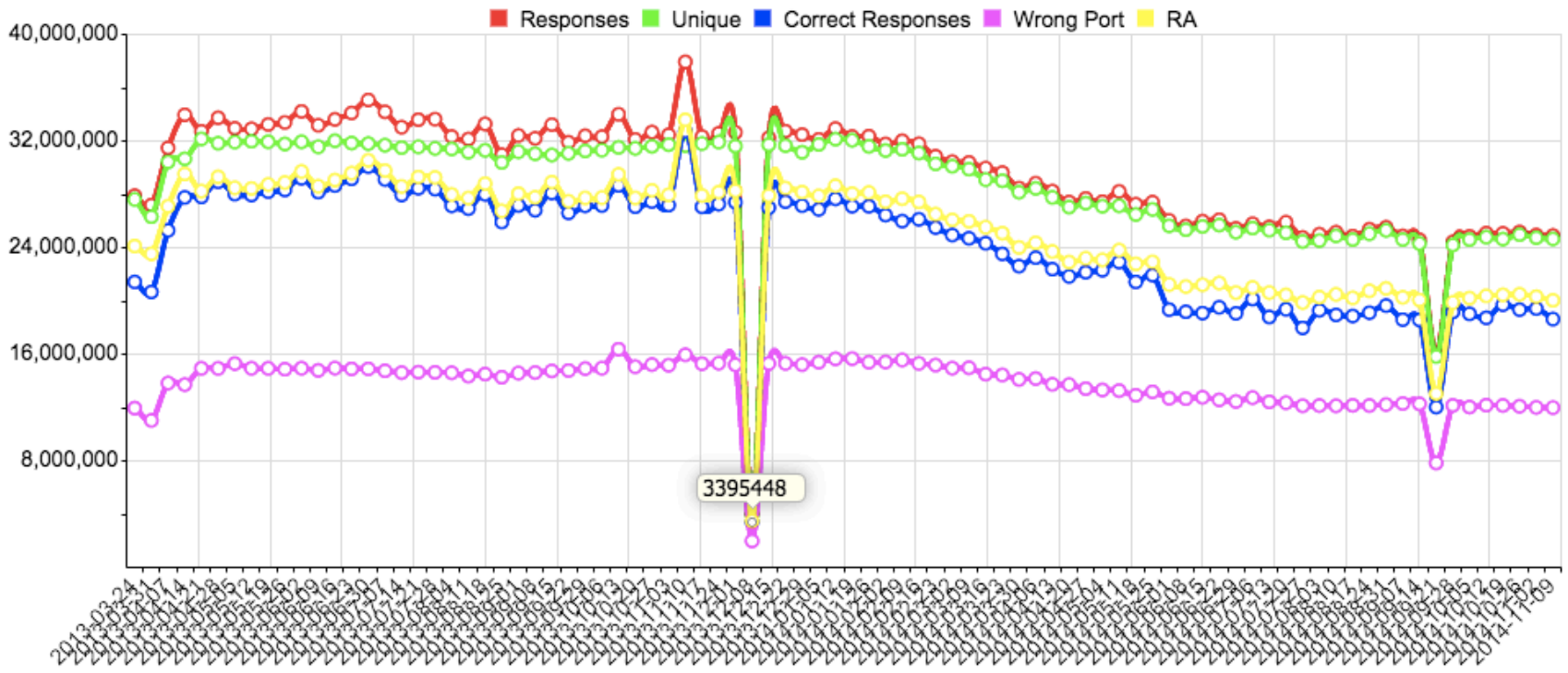


Open Resolvers

- DNS servers that answer recursive queries from any host on the Internet
- http://openresolverproject.org/
- Check if you're running open resolvers
  - http://dns.measurement-factory.com/cgi-bin/openresolvercheck.pl
- More statistics at
  - http://dns.measurement-factory.com/surveys/openresolvers/ASN-reports/latest.html

Open Resolvers

As of 27 Oct 2013:
24,910,951 servers responded to udp/53 probe
19,834,410 returned OK



Reference: http://openresolverproject.org/

Response Rate Limiting (RRL)

- Protects against DNS amplification attack
- Implemented in CZ-NIC Knot (v1.2-RC3), NLNetLabs NSD (v3.2.15), and ISC BIND 9 (v9.9.4) release

```
rate-limit {
                responses-per-second 5;
                log-only yes;
        };
```

- If using older versions, a patch is available from
  - http://ss.vix.su/~vjs/rrlrpz.html
  - `patch -p0 -l`

DNS Changer

- “Criminals have learned that if they can control a user’s DNS servers, they can control what sites the user connects to the Internet.”
- How: infect computers with a malicious software (malware)
- This malware changes the user’s DNS settings with that of the attacker’s DNS servers
- Points the DNS configuration to DNS resolvers in specific address blocks and use it for their criminal enterprise
- For more: see the NANOG presentation by Merike

Rogue DNS Servers

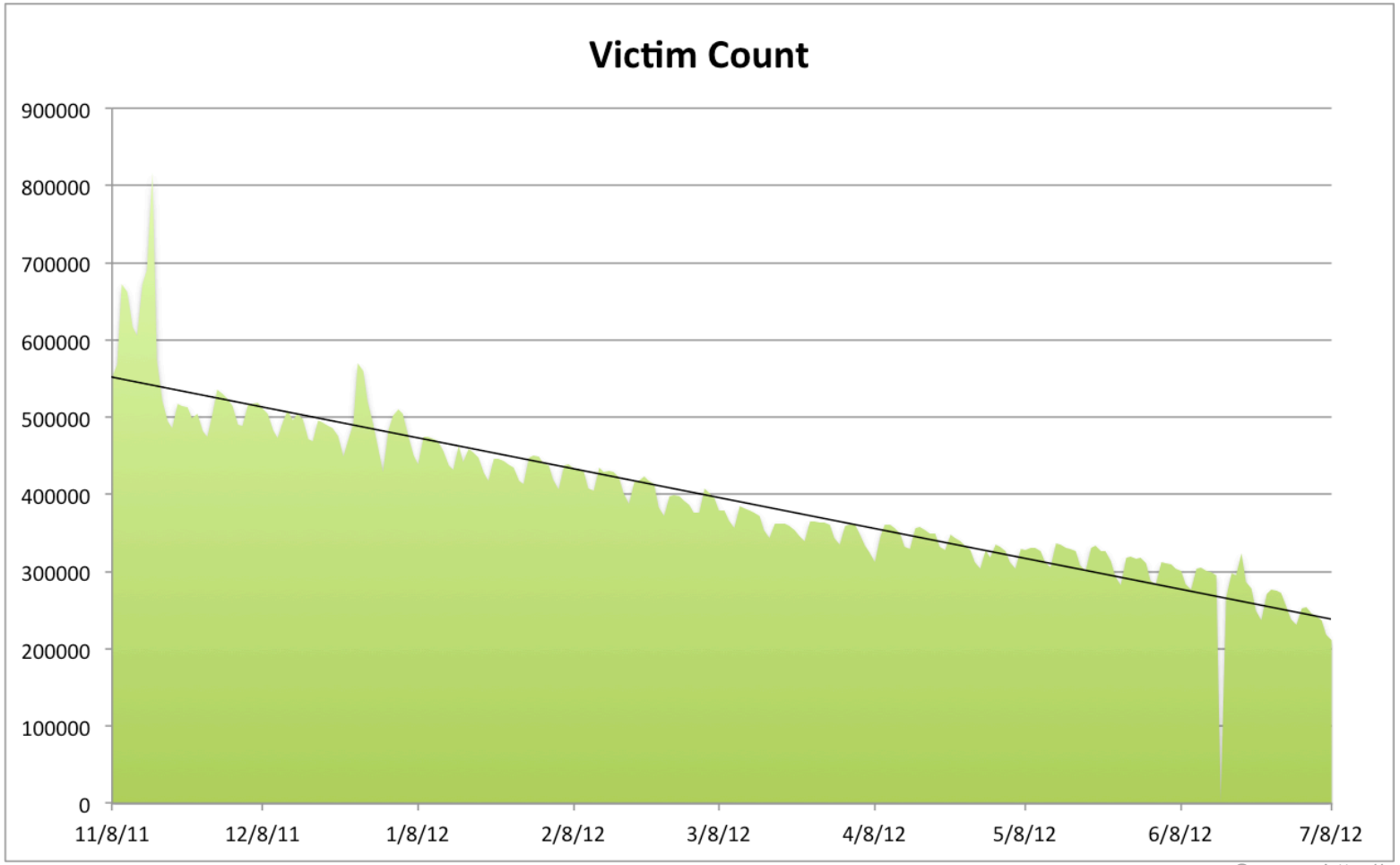
- 85.225.112.0 through 85.255.127.255
- 67.210.0.0 through 67.210.15.255
- 93.188.160.0 through 93.188.167.255
- 77.67.83.0 through 77.67.83.255
- 213.109.64.0 through 213.109.79.255
- 64.28.176.0 through 64.28.191.255

- If your computer is configured with one of these DNS servers, it is most likely infected with DNSChanger malware

Top DNS Changer Infections

- By country (as of 11 June 2012):
  - USA - 69517
  - IT – 26494
  - IN – 21302
  - GB – 19589
  - DE – 18427
- By ASNs
  - AS9829 (India) – 15568
  - AS3269 – 13406
  - AS7922 – 11964
  - AS3320 – 9250
  - AS7132 – 6743
- More info at http://dcwg.org/

DNS Changer – Victim Count



Source: http://www.dcwg.org

DNS Changer (News)

Operation Ghost Click

International Cyber Ring That Infected Millions of Computers Dismantled

11/09/11

Six Estonian nationals have been arrested and charged with running a sophisticated Internet fraud ring that infected millions of computers worldwide with a virus and enabled the thieves to manipulate the multi-billion-dollar Internet advertising industry. Users of infected machines were unaware that their computers had been compromised—or that the malicious software rendered their machines vulnerable to a host of other viruses.

Details of the two-year FBI investigation called Operation Ghost Click were announced today in New York when a federal indictment was unsealed. Officials also described their efforts to make sure infected users' Internet access would not be disrupted as a result of the operation.

The indictment, said Janice Fedarcyk, assistant director in charge of our New York office, "describes an intricate international conspiracy conceived and carried out by sophisticated criminals." She added, "The harm inflicted by the defendants was not merely a matter of reaping illegitimate income."

Beginning in 2007, the cyber ring used a class of malware called DNSChanger to infect approximately 4 million computers in more than 100 countries. There were about 500,000 infections in the U.S., including computers belonging to individuals, businesses, and government agencies such as NASA. The thieves were able to manipulate Internet advertising to generate at least $14 million in illicit fees. In some cases, the malware had the additional effect of preventing users' anti-virus software and operating systems from updating, thereby exposing infected machines to even more malicious software.

"They were organized and operating as a traditional business but profiting illegally as the result of the malware," said one of our cyber agents who worked the case. "There was a level of complexity here that we haven't seen before."

FBI Statement:
Janice Fedarcyk, New York
Assistant Director in Charge

"Today, with the flip of a switch, the FBI and our partners dismantled the Rove criminal enterprise. Thanks to the collective effort across the U.S. and in Estonia, six leaders of the criminal enterprise have been arrested and numerous servers operated by the criminal organization have been disabled. Additionally, thanks to a coordinated effort of trusted industry partners, a mitigation plan commenced today, beginning with the replacement of rogue DNS servers with clean DNS servers to keep millions online, while providing ISPs the opportunity to coordinate user remediation efforts." More

DNS—Domain Name System—is a critical Internet service that converts user-friendly domain names, such as www.fbi.gov, into numerical addresses that allow computers to talk to each other. Without DNS and the DNS servers operated by Internet service providers, computer users would not be able to browse websites or send e-mail.

Securing the Nameserver

- Run the most recent version of the DNS software
  - Apply the latest patches
- Hide version
- Restrict queries
  - `Allow-query { acl_match_list; };`
- Prevent unauthorized zone transfers
  - `Allow-transfer { acl_match_list; };`
- Run BIND with the least privilege (use `chroot`)
- Randomize source ports
  - don't use `query-source` option
- Secure the box
- Use TSIG and DNSSEC

Sender Policy Framework (SPF)

- Using DNS for email validation
- Checks the sender IP address
- Defined in RFC 4408 with updates in RFC 6652

```
apnic.net.          3600   IN     TXT    "v=spf1 mx
a:clove.apnic.net a:asmtp.apnic.net ip4:203.119.93.0/24
ip4:203.119.101.0/24 ip4:203.89.255.141/32
ip4:203.190.232.30/32 ip4:122.248.232.184/32
include:_spf.google.com -all"
```

DANE

- DNS-Based Authentication of Named Entities
- RFC 6698 (proposed standard)
- “secure method to associate the certificate that is obtained from the TLS server with a domain name using DNS”
- Adds a TLSA resource record

DNS RPZ

- Resource Policy Zone
- Developed for ISC Bind. Built in from version 9.8
- Turns a recursive DNS server into a “DNS firewall”
- “reputation-based” zones
- Like creating a reputation server for recursive DNS servers
  - Function is similar to DNSBL for email SMTP servers
- Blocks DNS resolution to malicious hosts

Questions

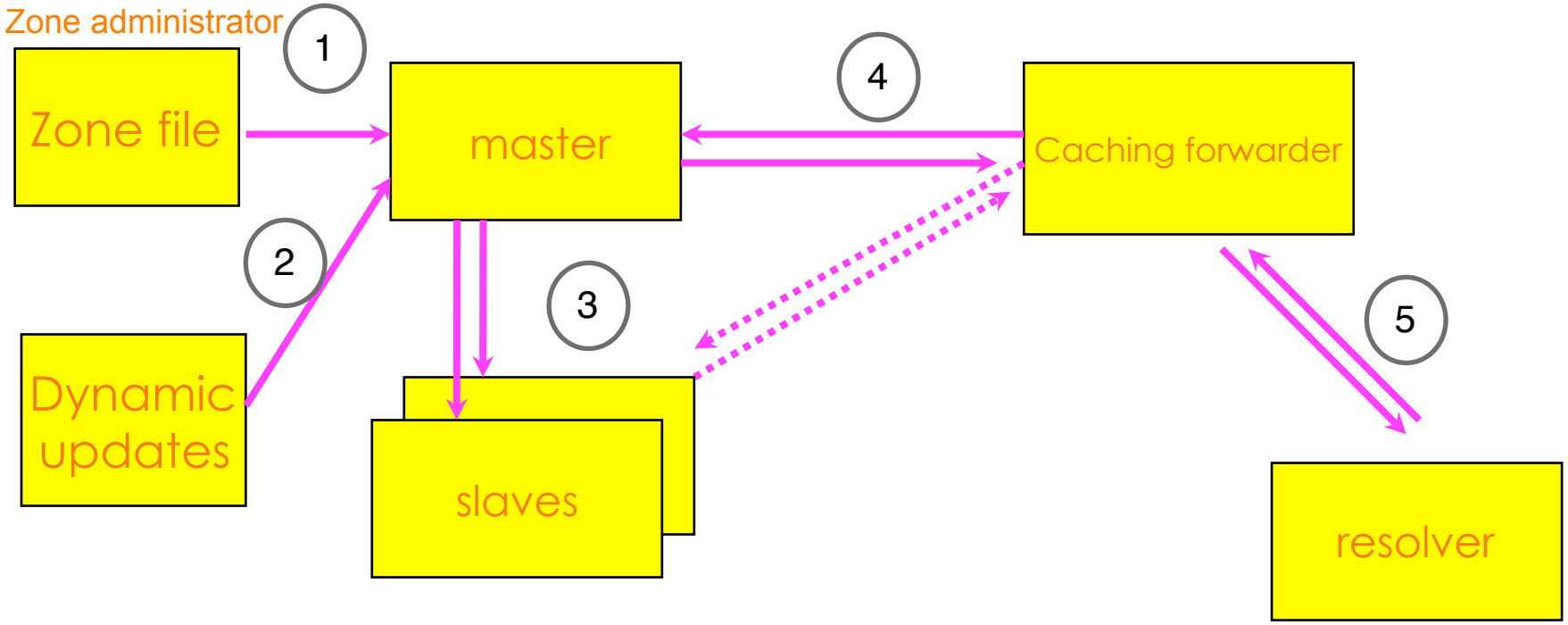
Transaction Signature (TSIG)

APNIC

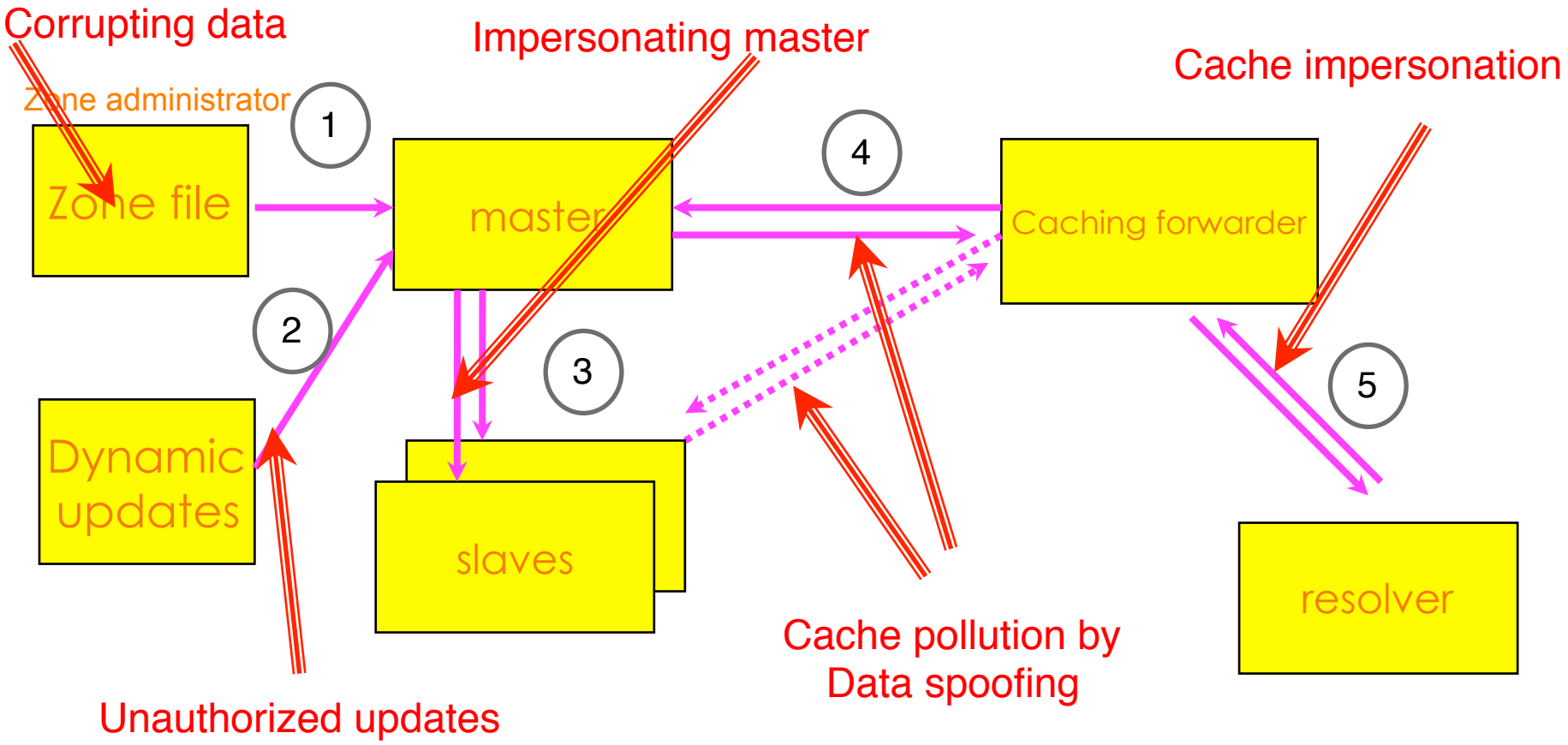
DNS Protocol Vulnerability

- DNS data can be spoofed and corrupted between master server and resolver or forwarder
- The DNS protocol does not allow you to check the validity of DNS data
  - Exploited by bugs in resolver implementation (predictable transaction ID)
  - 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
- How does a slave (secondary) know it is talking to the proper master (primary)?

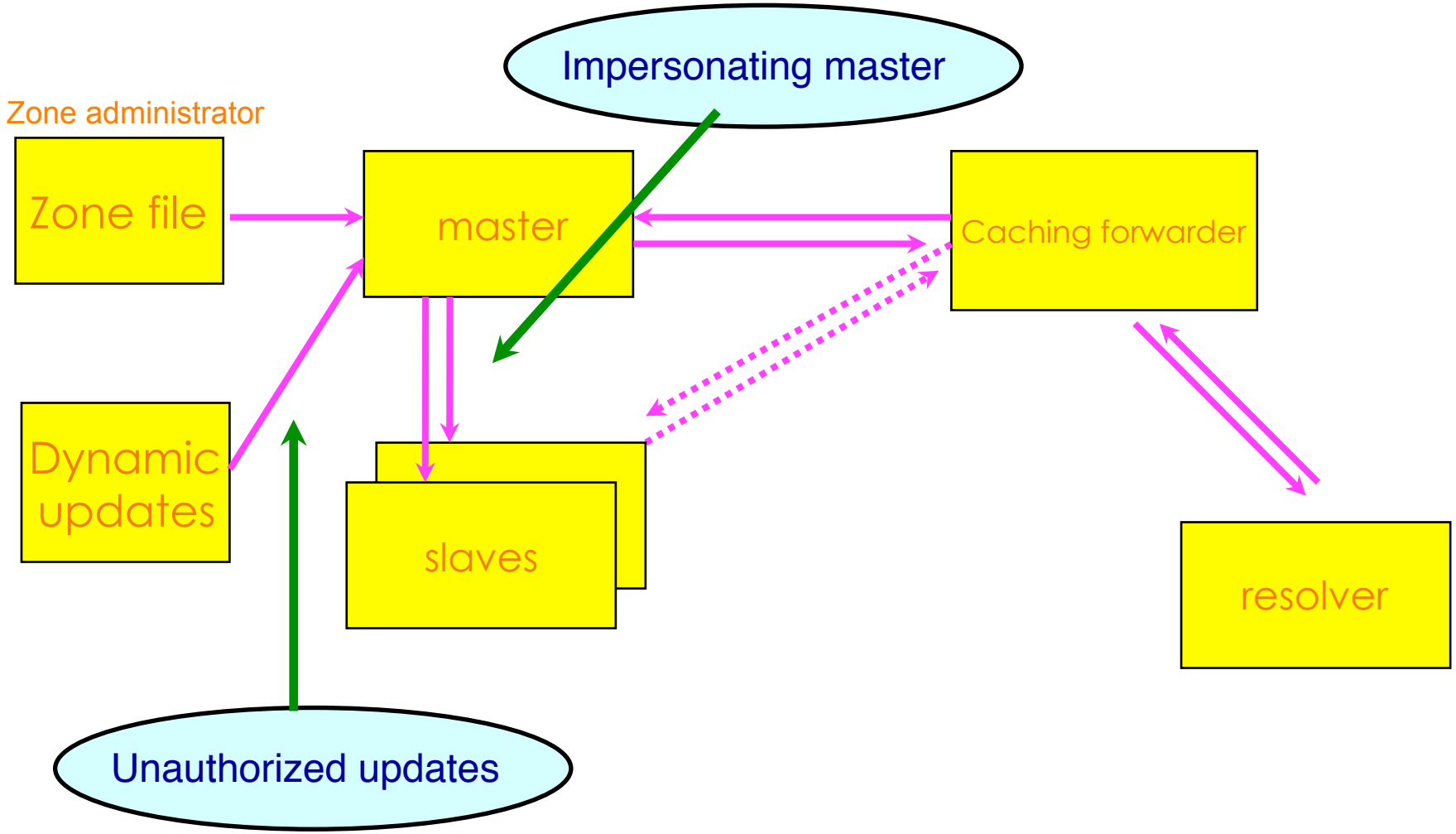
DNS: Data Flow



DNS Vulnerabilities



TSIG Protected Vulnerabilities



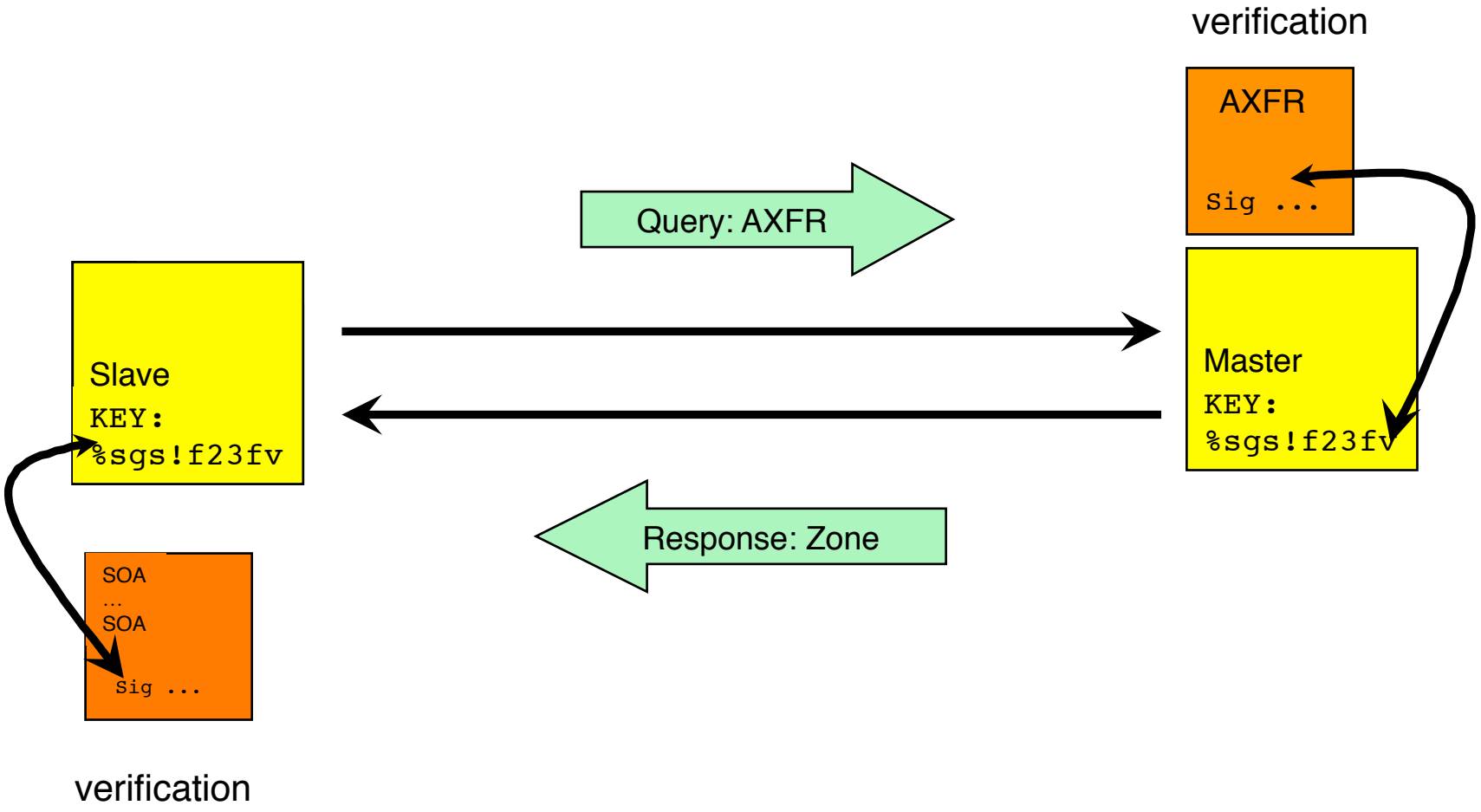
What is TSIG - Transaction Signature?

- A mechanism for protecting a message from a primary to secondary and vice versa
- A keyed-hash is applied (like a digital signature) so recipient can verify the message
  - DNS question or answer
  - & the timestamp
- Based on a shared secret - both sender and receiver are configured with it
  - TSIG/TKEY uses DH, HMAC-MD5, HMAC-SHA1, HMAC-SHA224, HMAC-SHA512 among others

What is TSIG - Transaction Signature?

- TSIG (RFC 2845)
  - authorizing dynamic updates & zone transfers
  - authentication of caching forwarders
- Used in server configuration, not in zone file

TSIG example



TSIG steps

1. Generate secret
2. Communicate secret
3. Configure servers
4. Test

TSIG - Names and Secrets

- TSIG name
  - A name is given to the key, the name is what is transmitted in the message (so receiver knows what key the sender used)

- TSIG secret value
  - A value determined during key generation
  - Usually seen in Base64 encoding

TSIG – Generating a Secret

- dnssec-keygen
  - Simple tool to generate keys
  - Used here to generate TSIG keys

```
> dnssec-keygen -a <algorithm> -b <bits> -n host
  <name of the key>
```

TSIG – Generating a Secret

- Example

```
> dnssec-keygen -a HMAC-MD5 -b 128 -n HOST ns1-
  ns2.pcx.net

This will generate the key
> Kns1-ns2.pcx.net.+157+15921

>ls
Kns1-ns2.pcx.net.+157+15921.key
Kns1-ns2.pcx.net.+157+15921.private
```

TSIG – Generating a Secret

- TSIG should never be put in zone files
  - might be confusing because it looks like RR:

```
ns1-ns2.pcx.net. IN KEY 128 3 157 nEfRX9…bbPn7lyQtE=
```

TSIG – Configuring Servers

- Configuring the key
  - in named.conf file, same syntax as for rndc
  - `key { algorithm ...; secret ...;}`
- Making use of the key
  - in named.conf file
  - `server x { key ...; }`
  - where 'x' is an IP number of the other server

Configuration Example – named.conf

Primary server 10.33.40.46

```
key ns1-ns2.pcx. net {
        algorithm hmac-md5;
        secret "APlaceToBe";
};
server 10.33.50.35 {
        keys {ns1-ns2.pcx.net;};
};
zone "my.zone.test." {
        type master;
        file “db.myzone”;
        allow-transfer {
        key ns1-ns2.pcx.net ;};
};
```

Secondary server 10.33.50.35

```
key ns1-ns2.pcx.net {
        algorithm hmac-md5;
        secret "APlaceToBe";
};
server 10.33.40.46 {
  keys {ns1-ns2.pcx.net;};
};
zone "my.zone.test." {
        type slave;
        file “myzone.backup”;
        masters {10.33.40.46;};
};
```

You can save this in a file and refer to it in the named.conf using ‘include’ statement:

```
include “/var/named/master/tsig-key-ns1-ns2”;
```

TSIG Testing : dig

- You can use dig to check TSIG configuration
  - `dig @<server> <zone> AXFR -k <TSIG keyfile>`

```
$ dig @127.0.0.1 example.net AXFR \
    -k Kns1-ns2.pcx.net.+157+15921.key
```

- Wrong key will give “Transfer failed” and on the server the security-category will log this.

TSIG Testing - TIME!

- TSIG is time sensitive - to stop replays
  - Message protection expires in 5 minutes
  - Make sure time is synchronized
  - For testing, set the time
  - In operations, (secure) NTP is needed

TSIG steps

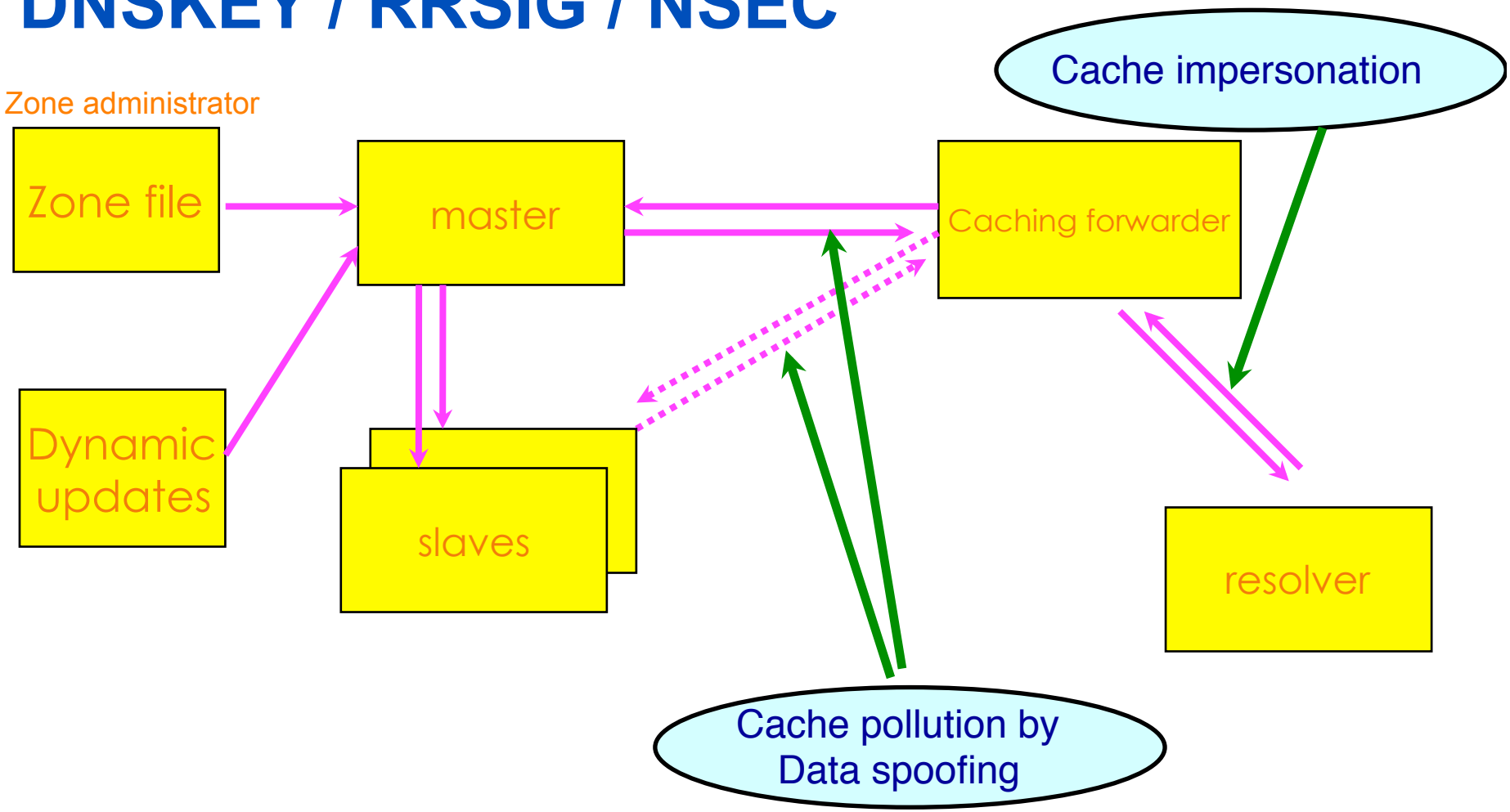
1. Generate secret
   - `dnssec-keygen -a <algorithm> -b <bits> -n host <name of the key>`
2. Communicate secret
   - `scp <keyfile> <user>@<remote-server>:<path>`
3. Configure servers
   - `key { algorithm ...; secret ...;}`
   - `server x { key ...; }`
4. Test
   - `dig @<server> <zone> AXFR -k <TSIG keyfile>`

Questions

DNSSEC

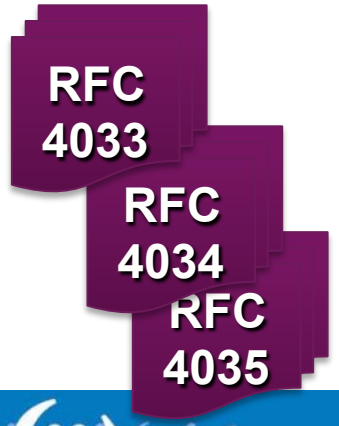
APNIC

Vulnerabilities protected by DNSKEY / RRSIG / NSEC



DNS Security Extensions (DNSSEC)

- Protects the integrity of data in the DNS by establishing a chain of trust
- Uses public key cryptography – each link in the chain has a public/private key pair
- A form of digitally signing the data to attest its validity
- Standard is defined in RFC4033, RFC4034, and RFC4035
- Guarantees
  - Authenticity
  - Integrity
  - Non-existence of a domain



DNSSEC History

- **1990**: Steven Bellovin discovers a major flaw in the DNS
- **1995**: Bellovin publishes his research; DNSSEC (as it became later known) becomes a topic within IETF
- **1997**: RFC 2065 (adding security extensions) was published
- **1998**: Dan Kaminsky discovers some security flaw
- **1999**: RFC 2535, the DNSSEC protocol, is published; BIND 9 developed to be DNSSEC-capable
- **2001**: key handling in RFC2535 is causing operational problems
- **2005**: Three new RFCs published to update RFC2535
  - RFC 4033 (DNS Security Introduction and Requirements)
  - RFC 4034 (Resource Records for DNS Security Extensions)
  - RFC 4035 (Protocol Modifications)

https://wiki.tools.isoc.org/DNSSEC_History_Project

DNSSEC History

- **2005**: In October, Sweden (.SE) becomes the first ccTLD to deploy DNSSEC
- **2008**: new DNSSEC record created to address privacy concerns (RFC 5155)
- **2010**
  - In July 15, the root zone was signed
  - In July 29, .edu was signed
  - In December 9, .net was signed
- **2011**: In March 31, `.com` was signed

https://wiki.tools.isoc.org/DNSSEC_History_Project

DNSSEC 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** = Next Secure; indicates which name is the next one in the zone and which typecodes are available for the current name
    - authenticated non-existence of data

DNSSEC Resource Records

- DNSKEY, RRSIG, and NSEC records provide mechanisms to establish authenticity and integrity of data
- DS record provides a mechanism to delegate trust to public keys of third parties

DNSSEC RRs

- Data authenticity and integrity by signing the Resource Records Sets with private key
- Public DNSKEY is used to verify the RRSIG
- 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

RR's and RRsets

- Resource Record:

Name	TTL	class	type	rdata
www.example.net.	7200	IN	A	192.168.1.1

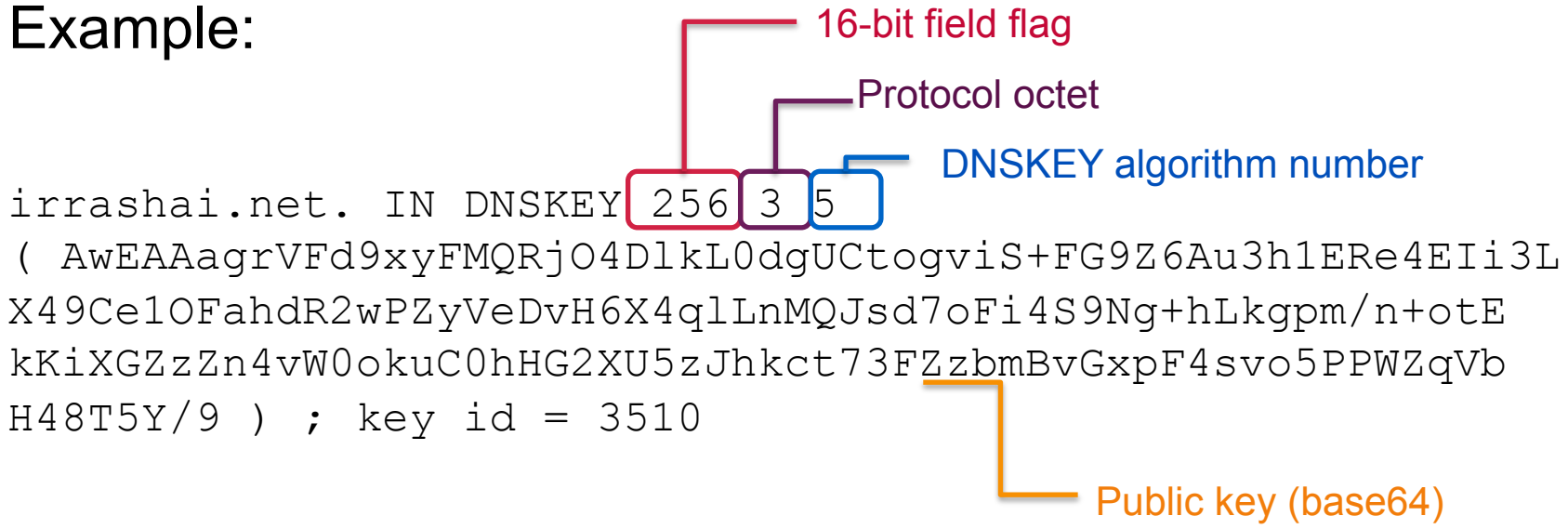
- RRset: RRs with same name, class and type:

www.example.net.	7200	IN	A	192.168.1.1
			A	10.0.0.3
			A	172.10.1.1

- RRsets are signed, not the individual RRs

DNSKEY

- Contains the zone's public key
- Uses public key cryptography to sign and authenticate DNS resource record sets (RRsets).
- Example:



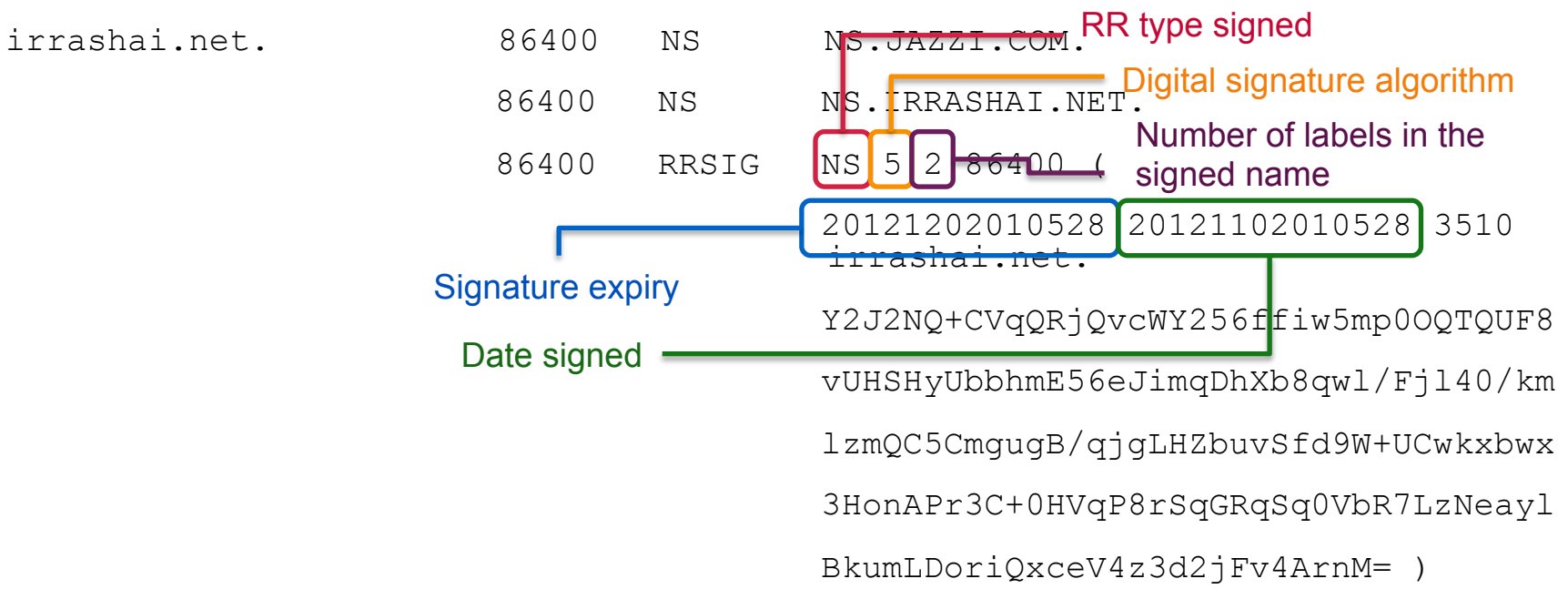
DNSKEY

- Also contains some timing metadata – as a comment in the key file

```
; This is a key-signing key, keyid 19996, for myzone.net.
; Created: 20121102020008 (Fri Nov  2 12:00:08 2012)
; Publish: 20121102020008 (Fri Nov  2 12:00:08 2012)
; Activate: 20121102020008 (Fri Nov  2 12:00:08 2012)
```

RRSIG

- The private part of the key-pair is used to sign the resource record set (RRset) per zone
- The digital signature per RRset is saved in an RRSIG record



NSEC / NSEC3

- Next Secure
- Forms a chain of authoritative owner names in the zone
- Lists two separate things:
  - Next owner name (canonical ordering)
  - Set of RR types present at the NSEC RR's owner name
- Also proves the non-existence of a domain

```
irrashai.net.  NSEC    blog.irrashai.net. A NS SOA MX
                       RRSIG NSEC DNSKEY
```

NSEC / NSEC3

- “The last NSEC wraps around from the last name in the ordered zone to the first”
- Each NSEC record also has a corresponding RRSIG

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

- Used for authenticated denial-of-existence of data
  - authenticated non-existence of TYPEs and labels

NSEC Record example

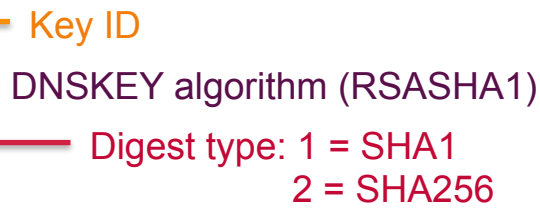
```
$ORIGIN example.net.
@ SOA        …
       NS     NS.example.net.
       DNSKEY…
       NSEC    mailbox.example.net. SOA NS NSEC DNSKEY        RRSIG

mailbox       A      192.168.10.2
                     NSEC  www.example.net.  A NSEC RRSIG
 WWW          A      192.168.10.3
                     TXT    Public webserver
                     NSEC  example.net. A NSEC RRSIG TXT
```

Delegation Signer (DS)

- Establishes the chain of trust from parent to child zones
- Found in the parent's zone file
- In this example, irrashai.net has been delegated from .net. This is how it looks like in .net zone file

```
irrashai.net.          IN NS   ns1.irrashai.net.
                       NS      ns2.irrashai.net.
                       IN DS   19996 5 1 (
                               CF96B018A496CD1A68EE7
                               C80A37EDFC6ABBF8175 )
                       IN DS   19996 5 2 (
                               6927A531B0D89A7A4F13E11031
                               4C722EC156FF926D2052C7D8D70C50
                               14598CE9 )
```



Delegation Signer (DS)

- Delegation Signer (DS) RR indicates that:
  - delegated zone is digitally signed
  - indicated key is used for the delegated zone

- Parent is authorative for the DS of the childs zone
  - Not for the NS record delegating the childs zone!
  - DS **should not** be in the childs zone

Types of Keys

- Zone Signing Key (ZSK)
  - Sign the RRsets within the zone
  - Public key of ZSK is defined by a DNSKEY RR
- Key Signing Key (KSK)
  - Signed the keys which includes ZSK and KSK and may also be used outside the zone
- Trusted anchor in a security aware server
- Part of the chain of trust by a parent name server
- Using a single key or both keys is an operational choice (RFC allows both methods)

Creation of keys

- Zones are digitally signed using the private key
- Can use RSA-SHA-1, DSA-SHA-1 and RSA-MD5 digital signatures
- The public key corresponding to the private key used to sign the zone is published using a DNSKEY RR

Chain of Trust

- DNSSEC is based on trust
- Root is on top of the chain of trust.
  - Root servers were signed on July 15, 2010.

Implementing DNSSEC

APNIC

DNSSEC - Setting up a Secure Zone

- Enable DNSSEC in the configuration file (named.conf)
  - `dnssec-enable yes; dnssec-validation yes;`
- Create key pairs (KSK and ZSK)
  - `dnssec-keygen -a rsasha1 -b 1024 -n zone champika.net`
- Publish your public key
- Signing the zone
- Update the config file
  - Modify the zone statement, replace with the signed zone file
- Test with dig

Updating the DNS Configuration

- Enable DNSSEC in the configuration file (named.conf)

```
options {
        directory "…."
        dnssec-enable yes;
        dnssec-validation yes;
  };
```

- Other options that can be added later
  - `auto-dnssec { off | allow | maintain} ;`
  - These options are used to automate the signing and key rollover

Creating key pairs

- To create ZSK

```
dnssec-keygen -a rsasha1 -b 1024 -n zone
<myzone>
```

- To create KSK

```
dnssec-keygen -a rsasha1 -b 1400 -f KSK -n
zone champika.net
```

Generating Key Pair

- Generate ZSK and KSK

```
dnssec-keygen -a rsasha1 -b 1024 -n zone <myzone>
```

Default values are RSASHA1 for algorithm, 1024 bits for ZSK and 2048 bits for KSK

The command above can be simplified as:

```
dnssec-keygen -f KSK <myzone>
```

This generates four files.

Note: There has to be at least one public/private key pair for each DNSSEC zone

Publishing your public key

- Using $INCLUDE you can call the public key (DNSKEY RR) inside the zone file

```
$INCLUDE /path/Kchampika.net.+005+33633.key ; ZSK
$INCLUDE /path/Kchampika.net.+005+00478.key ; KSK
```

- You can also manually enter the DNSKEY RR in the zone file

Signing the Zone

- Sign the zone using the secret keys:

```
dnssec-signzone –o <zonename> -N INCREMENT -f
<output-file> -k <KSKfile> <zonefile> <ZSKfile>
```

```
dnssec-signzone –o champika.net db.champika.net
Kchampika.net.+005+33633
```

- Once you sign the zone a file with a .signed extension will be created
  - `db.champika.net.signed`

Signing the Zone

- Note that only authoritative records are signed
  - NS records for the zone itself are signed
  - NS records for delegations are not signed
  - DS RRs are signed!
  - Glue is not signed
- Difference in the file size
  - db.champika.net vs. db.champika.net.signed

Smart Signing

- Searches the key repository for any keys that will match the zone being signed

```
options {
keys-directory { "path/to/keys"; };
```

- Then the command for smart signing is

```
dnssec-signzone -S db.myzone.net
```

Publishing the Zone

- Reconfigure to load the signed zone. Edit named.conf and point to the signed zone.

```
zone "<myzone>" {
     type master;
     # file "db.myzone.net";
     file "db.myzone.net.signed";
};
```

Pushing the DS record

- The DS record must now be published by the parent zone.
- Contact the parent zone to communicate the KSK to them.

KSK Key Rollover

- Perform scheduled zone maintenance.
- KSK rollover using Double signing
- When you change the KSK keys, the DS record in the parent zone must also be updated.

```
dnssec-signzone –o myzone.net –N increment –f <output- \
file> -k Kmyzone.net.+005+11111 db.myzone.net \
Kmyzone.net.+005+67890
```

- Send the new DS record to the parent, and wait for it to propagate.
- Remove the old key and resign.

KSK Key Rollover

- Using Pre-publication
- In this method, the new key will be published but will not be used for signing yet.

```
dnssec-keygen -K keydir -f ksk -A none <myzone.net>
rndc loadkeys <myzone.net>
```

- Publish both keys, but use only the old one for signing
- Wait for the propagation time and TTL of the DNSKEY RR to expire.

KSK Key Rollover

- Then use dnssec-settime once you are ready to sign the zone. Use the new key for zone signing, leaving the old one published.

```
dnssec-settime -K keydir -A now Kexample.com.+005+12345
rndc loadkeys example.com
```

- Wait for the propagation and TTL in the old zone. Set the old key to no longer sign with the key, but leaves it in the zone.

```
dnssec-settime -K keydir -I now Kexample.com.+005+12345
rndc loadkeys example.com
```

- Now remove the old keys. This completely removes the keys.

```
dnssec-settime -K keydir -D now Kexample.com.+005+12345
rndc loadkeys example.com
```

Automated Signing

- Using RNDC

- Add the option to named.conf

```
auto-dnssec allow;
```

- Then you can use the commands:

```
rndc loadkeys zone
rndc sign zone
```

Testing the server

- Ask a dnssec enabled question from the server and see whether the answer contains dnssec-enabled data
  - Basically the answers are signed

```
dig @localhost www.champika.net +dnssec
+multiline
```

Testing with dig: an example

Terminal — bash — 144×46

```
bash-3.2# dig @localhost www.champika.net +dnssec +multiline

; <<>> DiG 9.6.0-APPLE-P2 <<>> @localhost www.champika.net +dnssec +multiline
; (3 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37425
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;www.champika.net.      IN A

;; ANSWER SECTION:
www.champika.net.       86400 IN A 192.168.1.2
www.champika.net.       86400 IN RRSIG A 5 3 86400 20091123163643 (
                              20091024163643 22827 champika.net.
                              Eyp1IVyQyYBLKOX2u/LT1+40xjBomXzLrcdwSErgioMb
                              pGyDWDLzP+FTbE3QCfBMLNDt2AGoYcty1cfY4li9sHkw
                              fue6hTQTSm0LhisBkVKQBy6ZD5oGiJQgaIkBGmLtVkPh
                              jGJ8Z1UhbwKcGGK13doAa+5X8mx6MXNCudiNWeg= )

;; AUTHORITY SECTION:
champika.net.           86400 IN NS ns.champika.net.
champika.net.           86400 IN RRSIG NS 5 2 86400 20091123163643 (
                              20091024163643 22827 champika.net.
                              CZsPewlhPWpYTl8wPh09QhD6pWt0If2mLVshviGKq4no
                              ISNVoijmX0LyIns+o3DZz/2+TtwoQCRFLbfI99YMS3fx
                              BHGYqFDeGItyVx3oBpmTuAtMu2+od5WFS+LClsJsEP/N
                              QvUDgtWrj8+Z0wVVj8aLe+I51h29ek7Mzk7+P4E= )

;; ADDITIONAL SECTION:
ns.champika.net.        86400 IN A 192.168.1.1
ns.champika.net.        86400 IN RRSIG A 5 3 86400 20091123163643 (
                              20091024163643 22827 champika.net.
                              eTP05c4GscnoC9V5sR6vgDo02WgCr1T5arU7YZhWctXI
                              vkmU1ni+wguwqW6xezfB/Eu4J69bMnpQoX2zWUDtLUCM
                              +FVLsFx4Bbt+BjPEJKV03g9vv6IdKkR/pxyE1kJWJWmI
                              tR49P2dywlzqqTyvnj3F1yuFRTLHhJvfcVc+n8w= )

;; Query time: 3 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Oct 25 03:40:38 2009
;; MSG SIZE  rcvd: 610
```

DNSSEC Key Management

APNIC

Ways to Deploy DNSSEC

- As part of the DNS software used
  - Manual key management
  - Can be quite complex
  - For static environment
  - Some means of automation using
    - option commands and scripts

DNSSEC tools for BIND, NSD, PowerDNS, etc

- Use with a hardware security module (HSM)
  - Semi-automatic
  - Good for dynamic environment

HSM, OpenDNSSEC

- Using an external appliance
  - 'dnssec-in-a-box'
  - Fully automates key generation, signing and rollover

DNS Appliance

Hardware Security Module

- Cryptographic devices used for storage of the encryption keys
  - Smart cards, PCI cards, USB tokens
- It also speeds up the cryptographic key generation
- Implements PKCS#11 (Cryptographic Token Key Interface)
  - A standard interface or API to cryptographic tokens

DNSSEC Signer Appliance

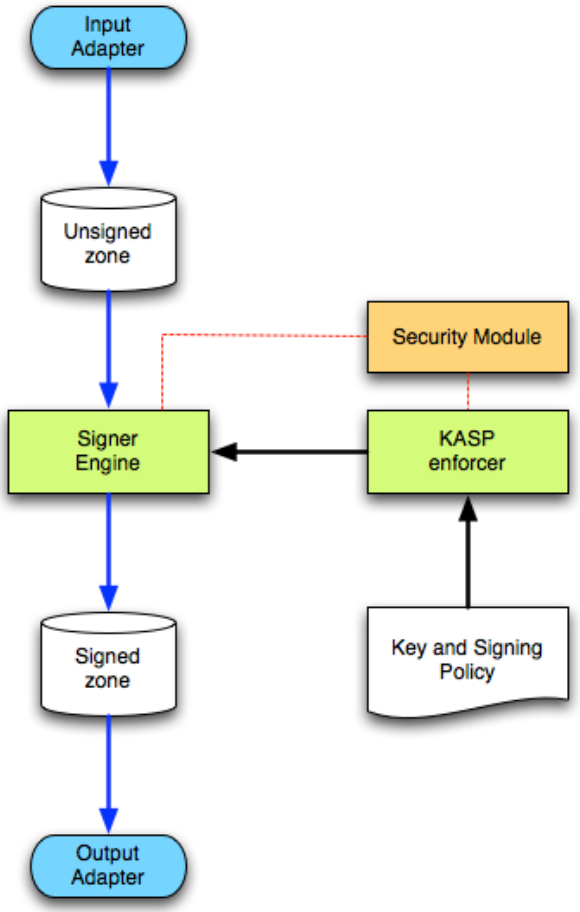
- Can be a pure signer or packaged with an IPAM or a DNS server
- In pure signer, the hardware appliance interfaces between the master/slave servers
- Examples: Secure64, Xelerance, SolidDNS, etc

OpenDNSSEC

APNIC

What is OpenDNSSEC?

- An open source turn-key solution for managing DNSSEC
- The goal is to simplify the signing process and minimize the workload on the admin
- Uses PKCS#11 for key storage
- Check out the documentation
  - https://wiki.opendnssec.org/display/DOCS



Reference: http://www.opendnssec.org/about/

Installation (on CentOS 6.1)

- Install dependencies
  - LDS (DNS programming library)
  - libxml2 (libxml2, libxml2-dev, libxml2-utils)
  - Java
  - SQLite (sqlite3, libsqlite3, libsqlite3-dev)
  - MySQL – optional
- Install a Hardware Security Module (HSM)
  - Or an equivalent software emulation
- Install OpenDNSSEC

SoftHSM Installation

- Software-only implementation of an HSM
- Dependencies
  - At least Botan 1.10.0 and OpenSSL 0.9.8
- Install from a repository or source

```
tar xvzf softhsm-<version>.tar.gz
cd softhsm-<version>
./configure
make
sudo make install
```

- Edit the configuration file and specify the slots to be used

```
vi /etc/softhsm.conf
0:/var/softhsm/slot0.db
```

SoftHSM

- Although the HSM has been defined, it has to be initialized

```
softhsm --init-token –slot 0 –label “opendnssec”
```

- Check that this HSM repository is configured in OpenDNSSEC’s conf.xml

```
<Repository name="SoftHSM”>
    <Module>/usr/local/lib/libsofthsm.so</Module>
    <TokenLabel>OpenDNSSEC</TokenLabel>
    <PIN>1234</PIN>
</Repository>
```

OpenDNSSEC

- Install from repository

  ```
  yum -y install opendnssec
  ```

- Important files
  - conf.xml – overall configuration
  - kasp.xml – policies used to sign zones; key and signature policy
  - zonelist.xml – list of zones that opendnssec will sign
  - addns.xml – dns adapter configuration
- The config folder is set to /etc/opendnssec/ by default

Conf.xml

- overall configuration

  `<RepositoryList>` - defines the HSM

  `<Common>` - common config

  `<Enforcer>` - deals with key rollover and generation

  `<Signer>` - the part that constructs the signature records to include in the zone file

Note the repository name

Reference to kasp.xml and zonelist.xml

Timing Parameters

- P[n]Y[n]M[n]DT[n]H[n]M[n]S
- P1Y6MT12H – 1 year, 6 months, and 12 hours
- P1Y – 1 year (always 365 days)
- P1M – 1 month (always 31 days)

Running OpenDNSSEC

- To begin with, run the command

  ```
  ods-ksmutil setup
  ```

- Two daemons must be started – **ods-signerd** and **ods-enforced**. Use the command to start them

  ```
  ods-control start
  ```

- Add zones

  ```
  ods-ksmutil zone add --zone <zonename>
  ```

- This can also be added manually by editing zonelist.xml. If you do this, run the command after edit:

  ```
  ods-ksmutil update zone list
  ```

Running OpenDNSSEC

- To check if the config files are valid

  ```
  ods-kaspcheck
  ```

- To generate the DS record from KSK

  ```
  ods-ksmutil key list -v (to get the keytag)
  ods-ksmutil key export -z example.com -e publish -x
  <keytag>
  ```

Error Logging

- Uses the system's syslog for logging. Define a log facility to where the messages will go

```
<Logging>
<Syslog><Facility>local0</Facility></Syslog>
</Logging>
```

- Check your logfile – /var/log/messages

```
tail -f /var/log/messages | grep ods*
```

- If you have syslog, you can edit /etc/rsyslog.conf to

Troubleshooting

- Some common problems encountered:
- Starting opendnssec

```
# ods-control start
Starting enforcer...
OpenDNSSEC ods-enforcerd started (version 1.4.1), pid
5601
Could not start enforcer
```

- Database not updated (when adding zones)
- Typo / error in the configuration files. Run

```
ods-kaspcheck
```

Questions

Further Readings

- DNS Operational Practices
  - http://tools.ietf.org/html/rfc6781
  - Informational, published December 2012
- A Framework for DNSSEC Policies and DNSSEC Practice Statements
  - http://tools.ietf.org/html/rfc6841
  - Informational, published January 2013

Thank you!

End of Session

APNIC