DNS Security

In Conjunction with

PacNOG

22-26 Nov 2011

Noumea, New Caledonia

Introduction

- Presenters
  - Champika Wijayatunga
    - Training Unit Manager
    - champika@apnic.net

DNS Security :
DNSSEC Deployment

Overview

- Introduction
  - – DNSSEC support in BIND
  - – Why DNSSEC?
- DNSSEC mechanisms
  - – To authenticate servers (TSIG)
  - – To establish authenticity and integrity of data
    - Quick overview
    - New RRs
    - Using public key cryptography to sign a single zone
    - Delegating signing authority ; building chains of trust
    - Key exchange and rollovers
- Steps

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

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) knows it is talking to the proper master (primary)?

Why DNSSEC?

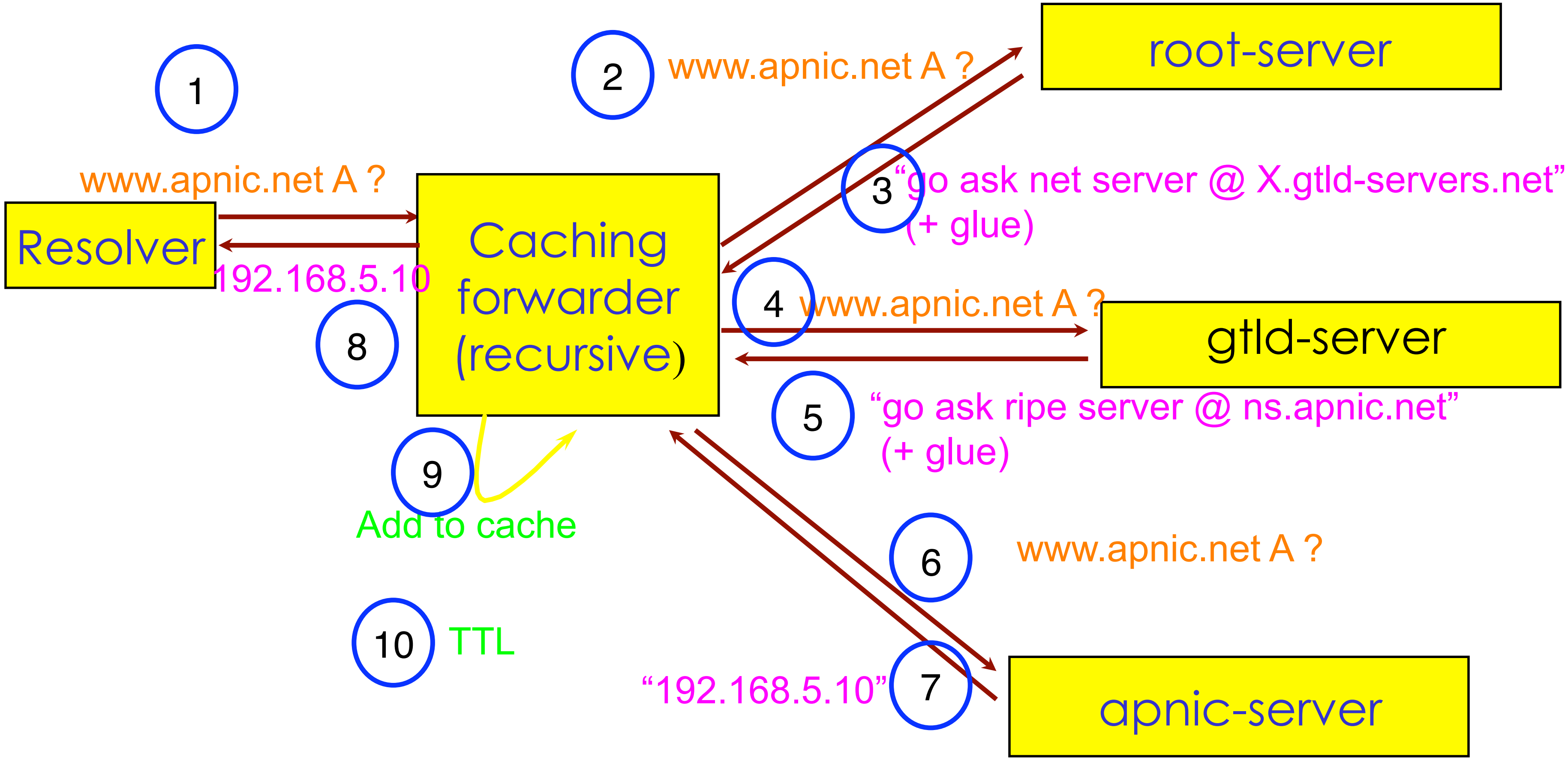
- DNS is not secure
  - Applications depend on DNS
    - Known vulnerabilities

- DNSSEC protects against data spoofing and corruption

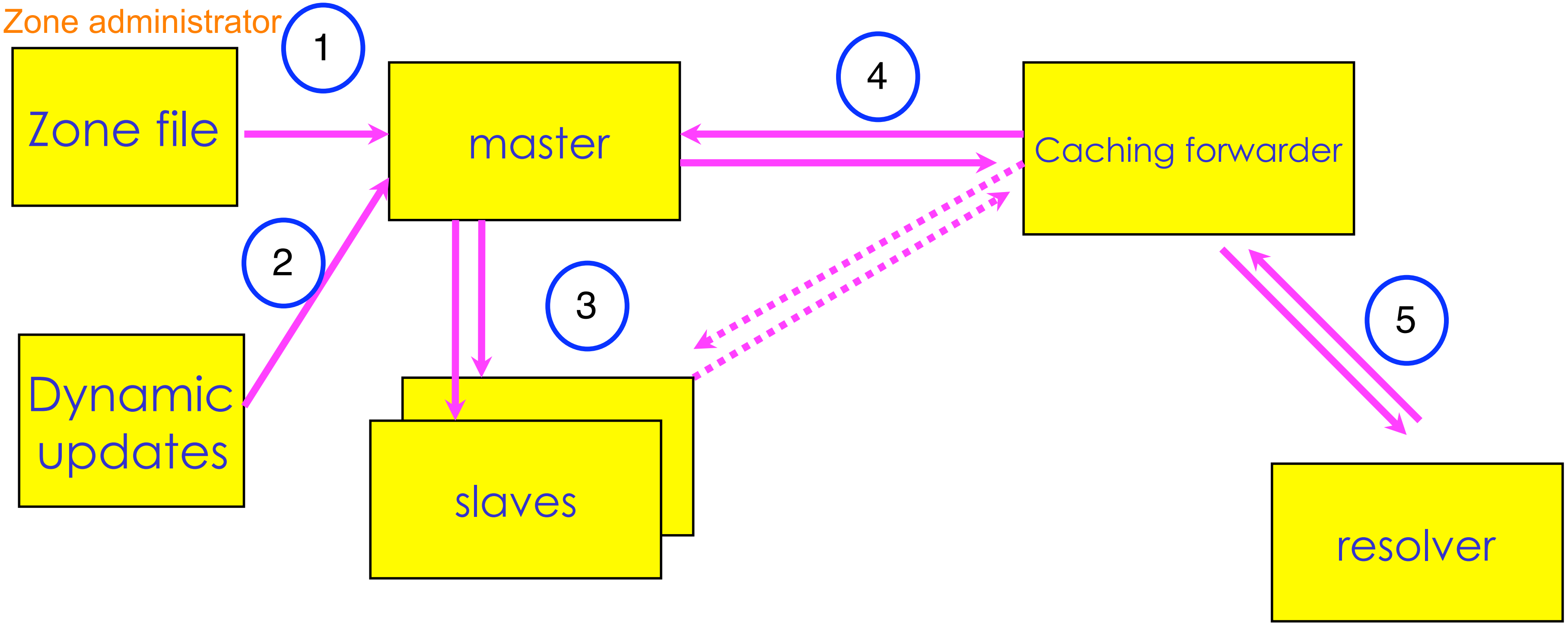
Reminder: DNS Resolving

Question:

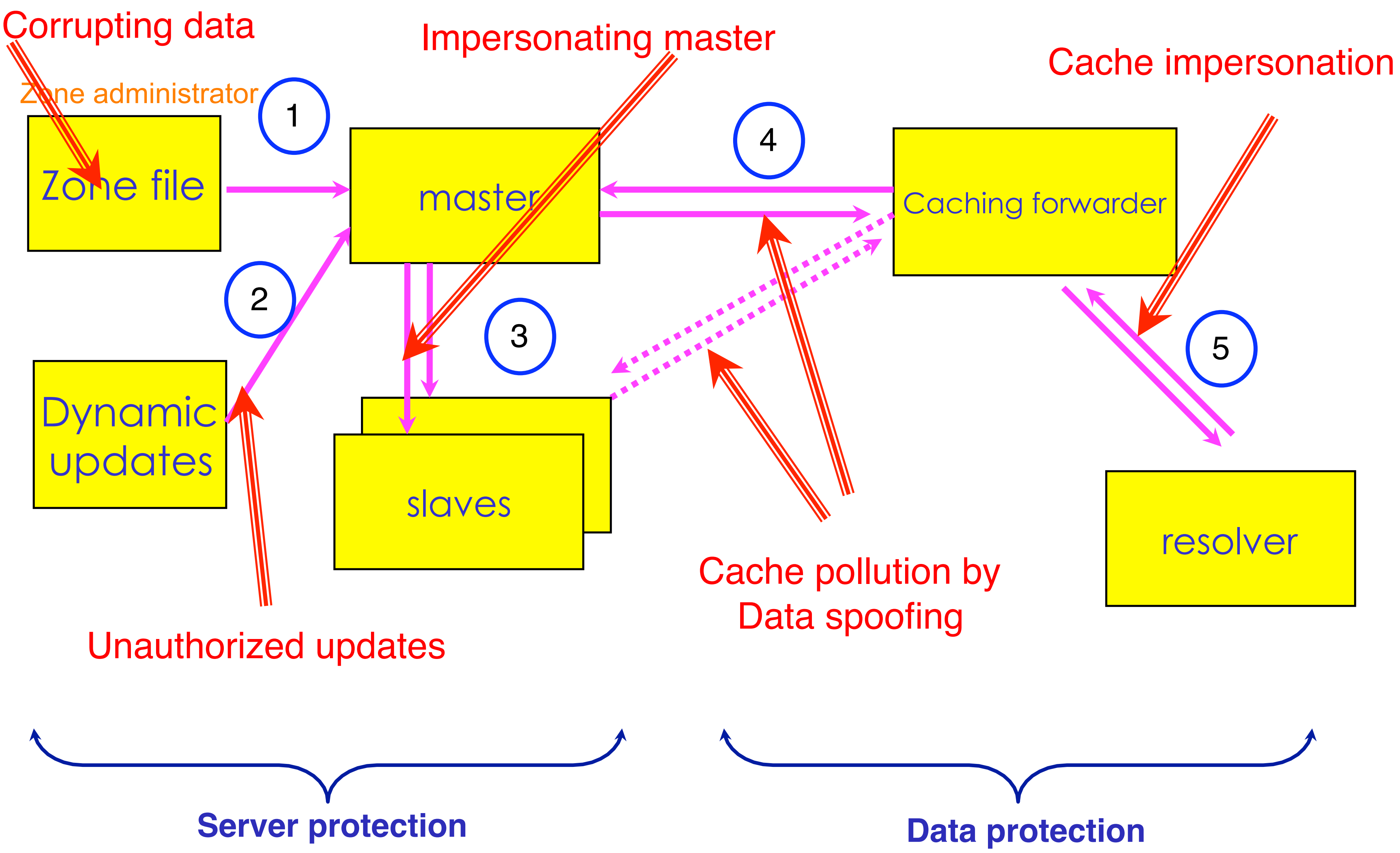
www.apnic.net A



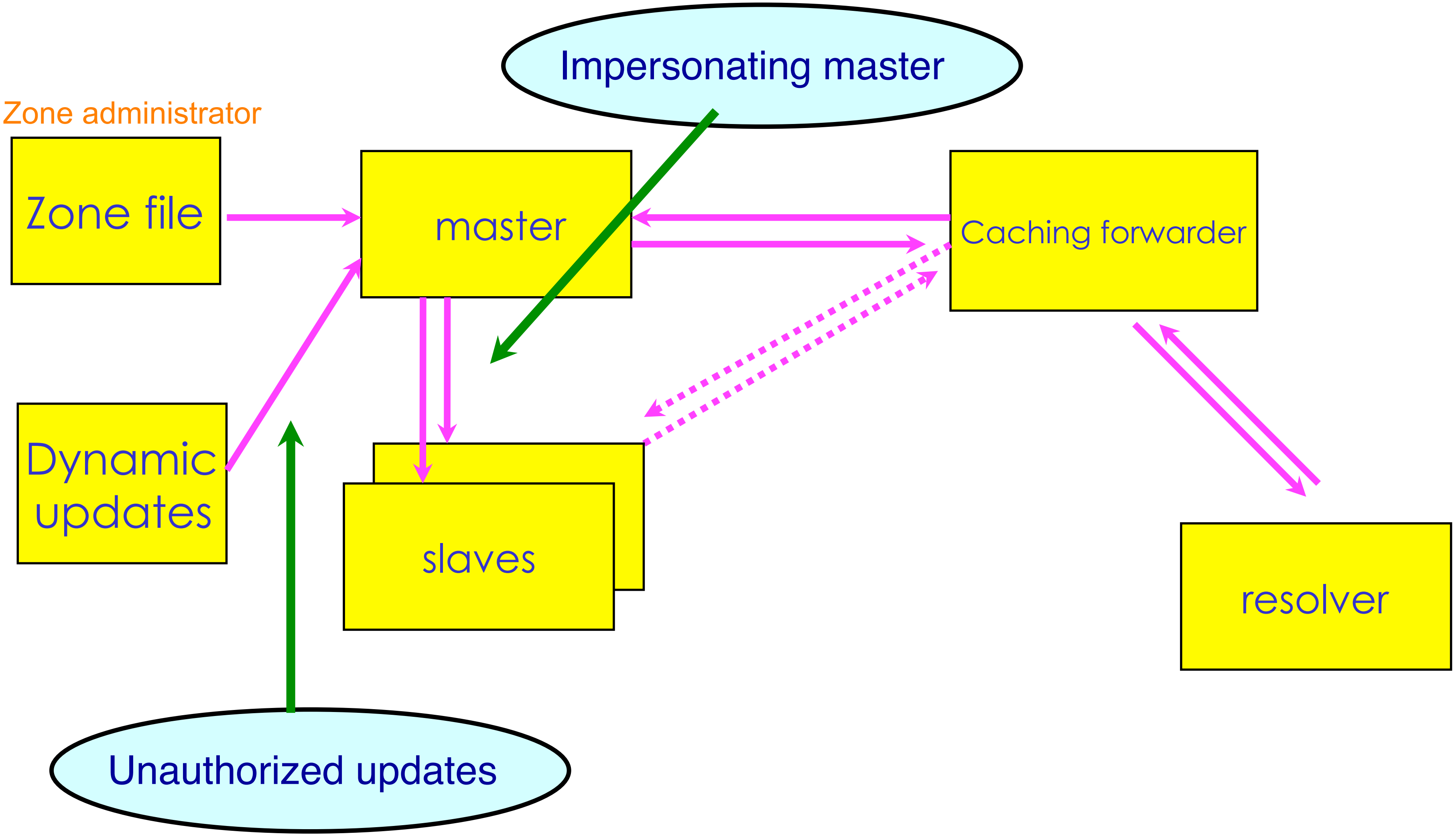
DNS: Data Flow



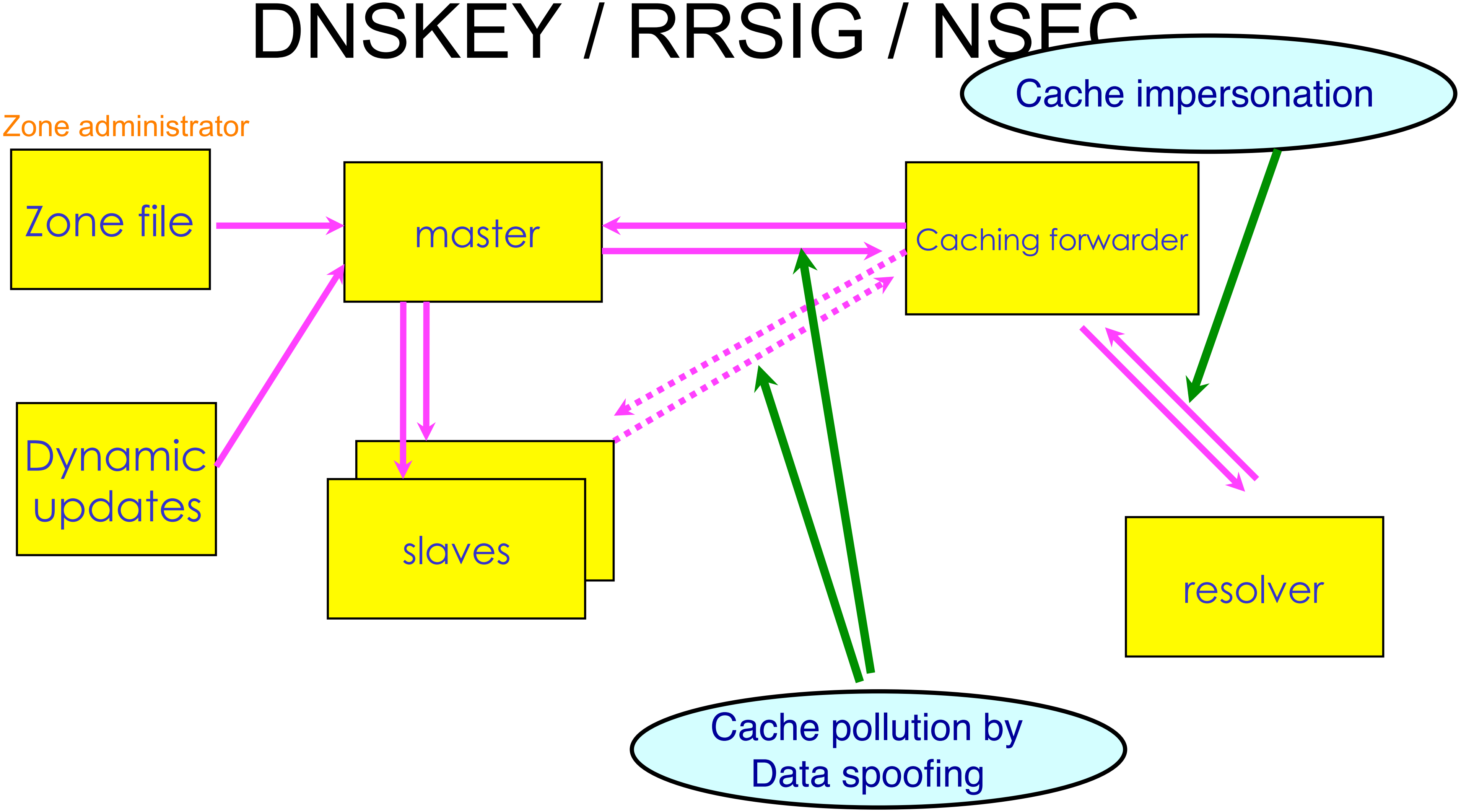
DNS Vulnerabilities



TSIG Protected Vulnerabilities



Vulnerabilities protected by DNSKEY / RRSIG / NSEC



What is TSIG - Transaction Signature?

- A mechanism for protecting a message from a primary to secondary and vice versa

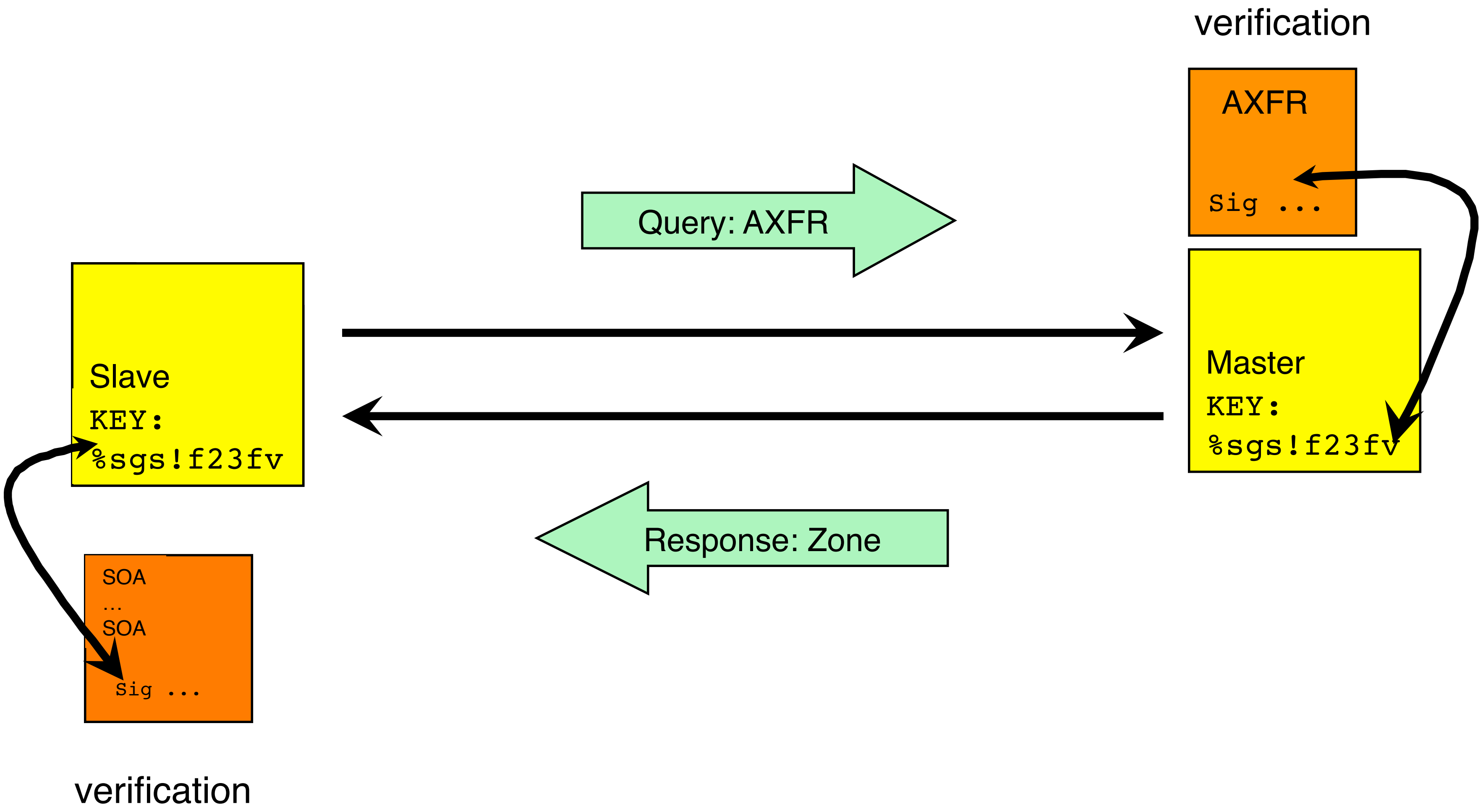
- A keyed-hash is applied (like a digital signature) so recipient can verify message
  - DNS question or answer
  - & the timestamp

- Based on a shared secret - both sender and receiver are configured with it

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



verification

AXFR

Sig ...

Query: AXFR

Slave

KEY:

%sgs!f23fv

Master

KEY:

%sgs!f23fv

Response: Zone

SOA

...

SOA

Sig ...

verification

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>`

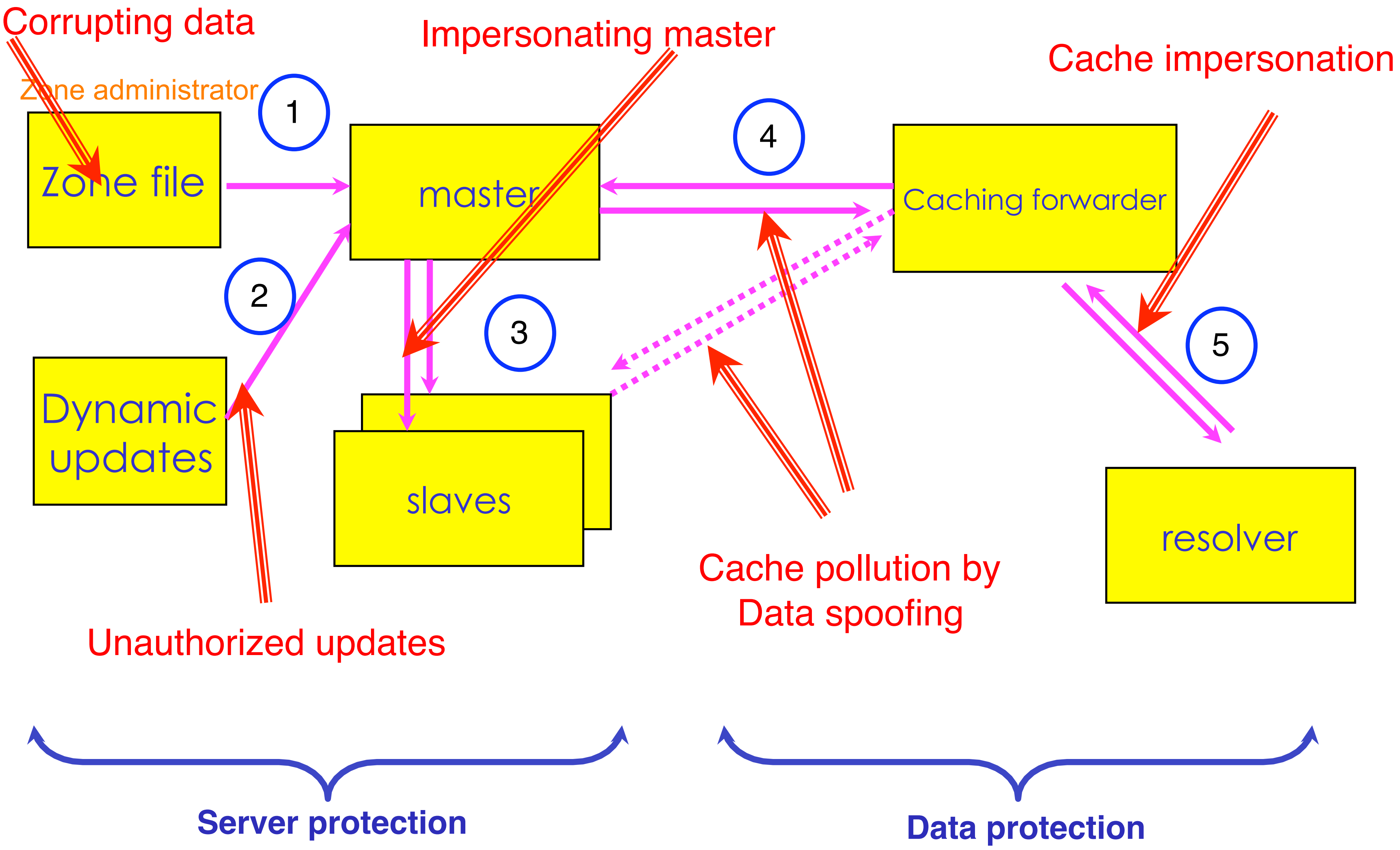
```
$ dig @127.0.0.1 example.net AXFR \
    -k Kns1-ns2.pcx.net.+157+15921.key
```

- 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

DNS Vulnerabilities



DNSSEC mechanisms

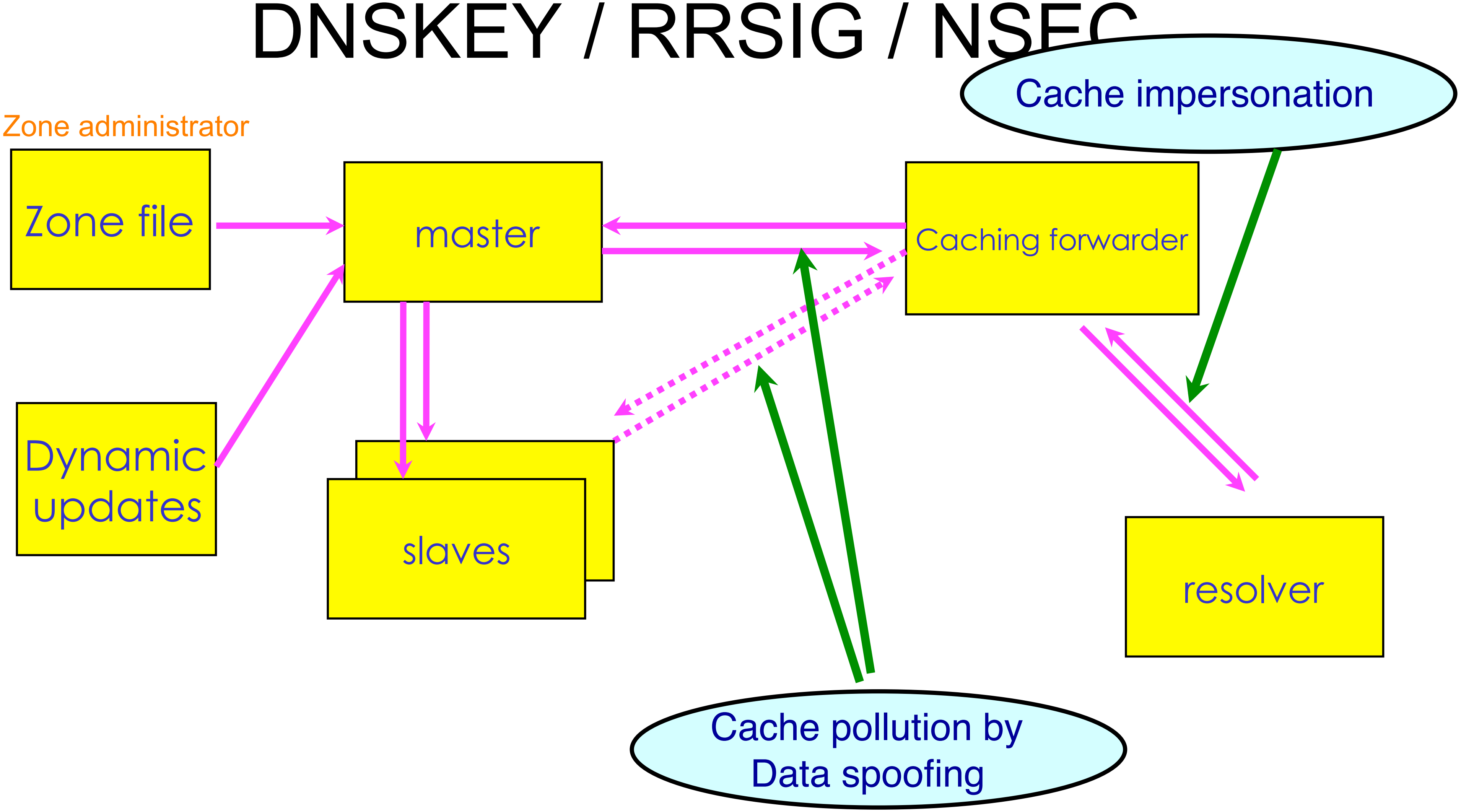
- TSIG: provides mechanisms to authenticate communication between servers

- DNSKEY/RRSIG/NSEC: provides mechanisms to establish authenticity and integrity of data

- DS: provides a mechanism to delegate trust to public keys of third parties

- A secure DNS will be used as an infrastructure with public keys
  - However it is **NOT** a PKI

Vulnerabilities protected by DNSKEY / RRSIG / NSEC



DNSSEC RRs

- Data authenticity and integrity by signing the Resource Records Sets with private key
- Public DNSKEYs used to verify the RRSIGs
- Children sign their zones with their private key
  - Authenticity of that key established by signature/checksum by the parent (DS)
- Ideal case: one public DNSKEY distributed

New Resource Records

- 3 Public key crypto related RRs
  - RRSIG
    - Signature over RRset made using private key
  - DNSKEY
    - Public key, needed for verifying a RRSIG
  - DS
    - Delegation Signer; ‘Pointer’ for building chains of authentication
- One RR for internal consistency
  - NSEC
    - Indicates which name is the next one in the zone and which typecodes are available for the current name
    - authenticated non-existence of data

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 RDATA

```
Example:
example.net. 3600 IN DNSKEY  256  3  5   (
    AQOvhvXXU61Pr8sCwELcqqq1g4JJ
    CALG4C9EtraBKVd+vGIF/unwigfLOA
    O3nHp/cgGrG6gJYe8OWKYNgq3kDChN)
```

RRSIG RDATA

```
example.net.  3600 IN  RRSIG   A  5  2  3600  (
  20081104144523 20081004144523  3112  example.net.
  VJ+8ijXvbrTLeoAiEk/qMrdudRnYZM1VlqhNvhYuAcYKe2X/
  jqYfMfjfSUrmhPo+0/GOZjW66DJubZPmNSYXw== )
```

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

DS RDATA

```
$ORIGIN .net.

example.net.       3600 IN   NS   ns.example.net

ns.example.net.    3600 IN   DS   3112  5 1 (
                                    239af98b923c023371b52

                                  1g23b92da12f42162b1a9

                                  )
```

NSEC RDATA

- Points to the next domain name in the zone
  - also lists what are all the existing RRs for “name”
  - NSEC record for last name “wraps around” to first name in zone

- 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
```

Setting up a secure zone

Enable dnssec

- In the named.conf,

```
Options {
    directory “….”
    dnssec-enable yes;
    dnssec-validation yes;
    };
```

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

Keys

- Two 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)

Creating key pairs

- To create ZSK
  - > ```
    > dnssec-keygen -a rsasha1 -b 1024 -n zone champika.net
    > ```
- To create KSK
  - > ```
    > dnssec-keygen -a rsasha1 -b 1400 -f KSK -n zone champika.net
    > ```

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

```
> dnssec-signzone –o champika.net -t -k
  Kchampika.net.+005+00478 db.champika.net
  Kchampika.net.+005+33633
```

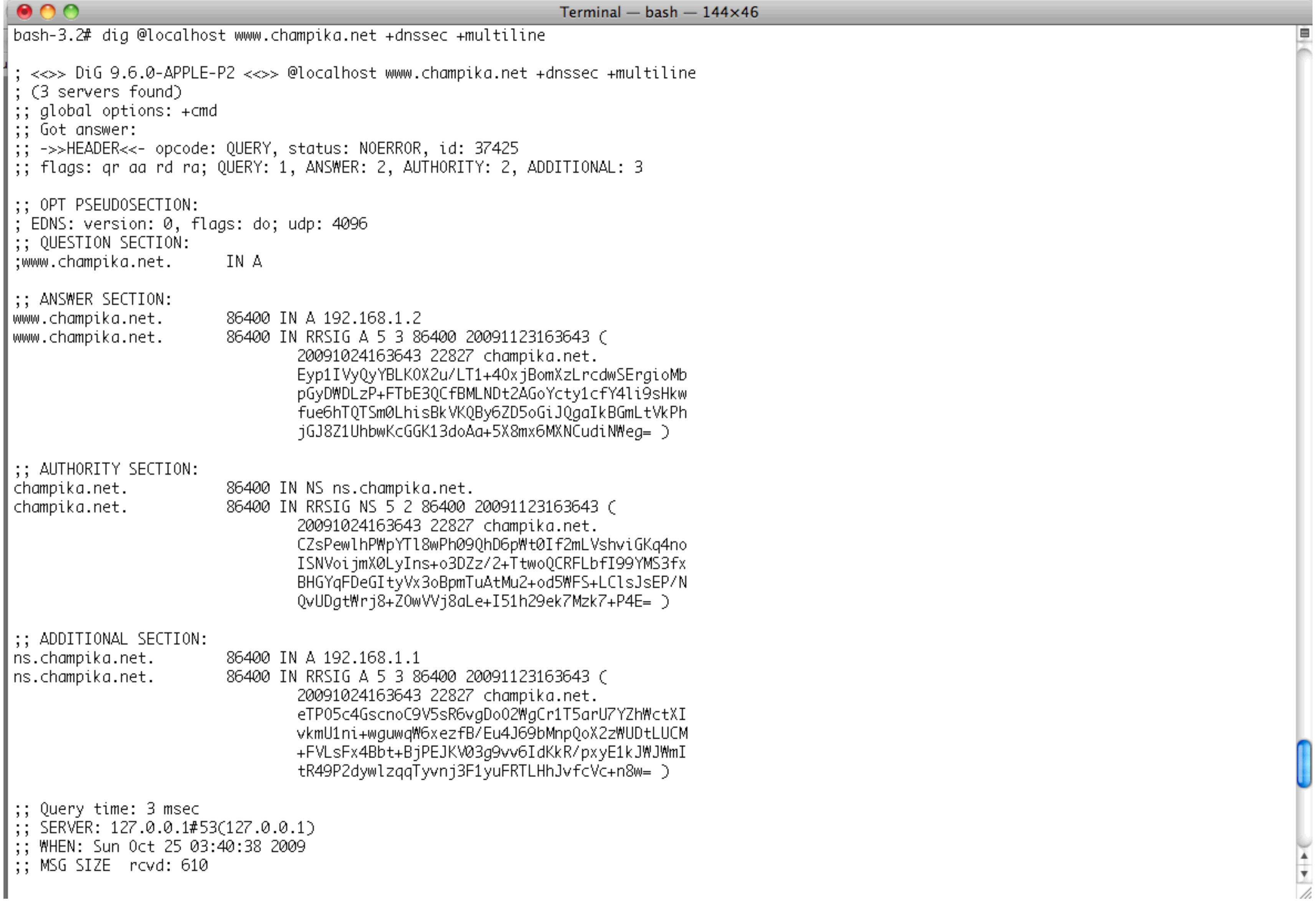
- Once you sign the zone a file with a .signed extension will be created
  - db.champika.net.signed

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



```
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+4OxjBomXzLrcdwSErgioMb
                                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+ZOwVVj8aLe+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.
                                eTPO5c4GscnoC9V5sR6vgDoO2WgCr1T5arU7YZhWctXI
                                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
```

Questions ?

Reverse DNS

Overview

- Principles
- Creating reverse zones
- Setting up nameservers
- Reverse delegation procedures

What is ‘Reverse DNS’?

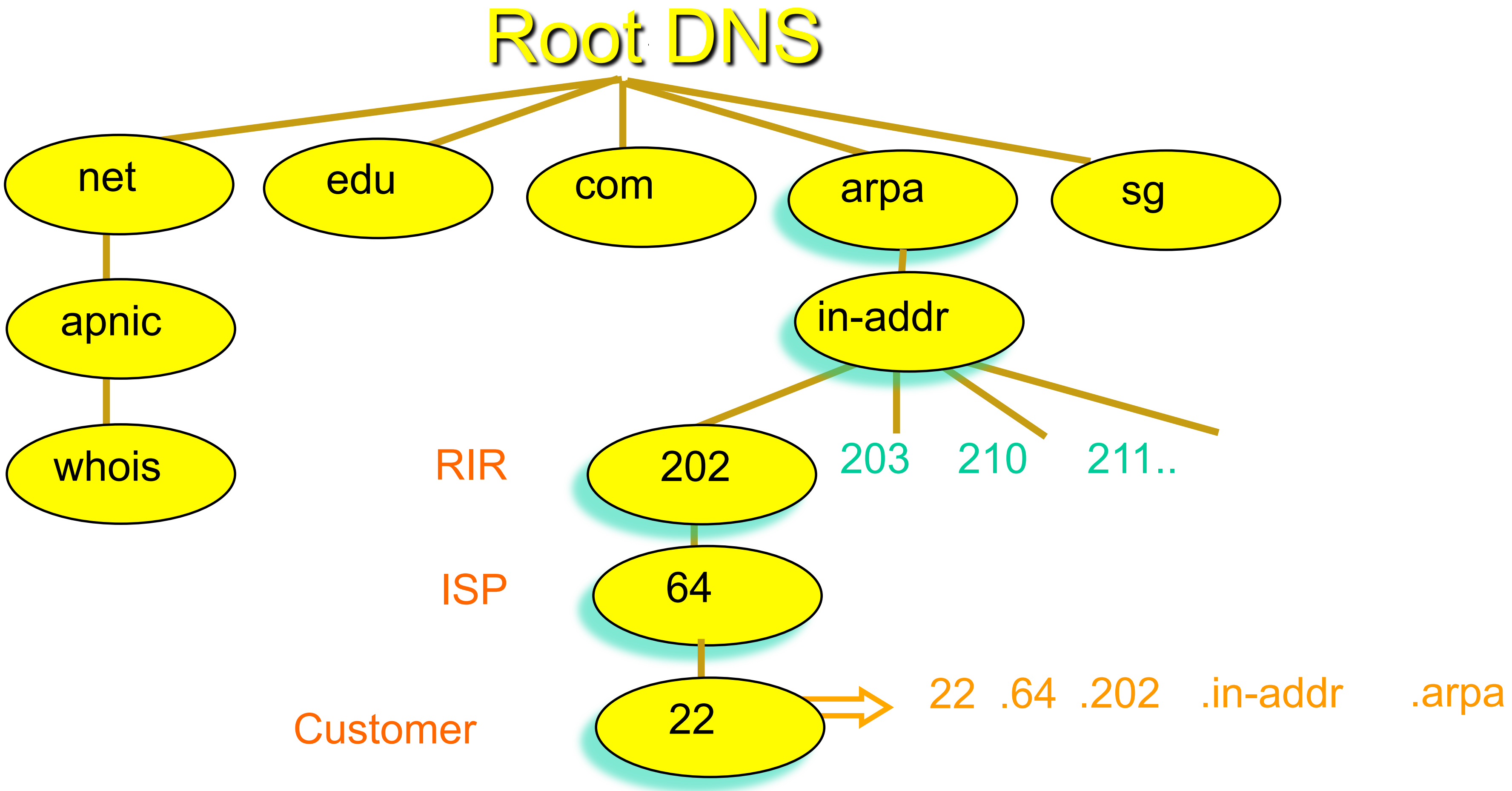
- ‘Forward DNS’ maps names to numbers
  - svc00.apnic.net -> 202.12.28.131
- ‘Reverse DNS’ maps numbers to names
  - 202.12.28.131 -> svc00.apnic.net

Reverse DNS - why bother?

- Service denial
  - That only allow access when fully reverse delegated eg. anonymous ftp
- Diagnostics
  - Assisting in trace routes etc
- SPAM identifications
- Registration responsibilities

Principles – DNS tree

- Mapping numbers to names - 'reverse DNS'



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

Creating reverse zones - contd

- Files involved
  - Zone files
    - Forward zone file
      - e.g. db.domain.net
    - Reverse zone file
      - e.g. db.192.168.254
  - Config files
    - <named.conf>
  - Other
    - Hints files etc.
      - Root.hints

Start of Authority (SOA) record

```
<domain.name.>          CLASS   SOA     <hostname.domain.name.>
<mailbox.domain.name> (
                                               <serial-number>
                                               <refresh>
                                               <retry>
                                               <expire>
                                               <negative-caching> )
```

253.253.192.in-addr.arpa.

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.
```

A 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.
;auto generate:  65 PTR host65.company.org
$GENERATE 65-127 $ PTR host$.company.org.
```

Setting up the primary nameserver

- Add an entry specifying the primary server to the ***named.conf*** file

```
zone "<domain-name>" in {
type master;
file "<path-name>"; };
```

- <domain-name>
  - Ex: 28.12.202.in-addr.arpa.
- <type master>
  - Define the name server as the primary
- <path-name>
  - location of the file that contains the zone records

Setting up the secondary nameserver

- Add an entry specifying the primary server to the ***named.conf*** file

```
zone "<domain-name>" in {
type slave;
file "<path-name>";
Masters { <IP address> ; }; };
```

- <type slave> defines the name server as the secondary
- <ip address> is the IP address of the primary name server
- <domain-name> is same as before
- <path-name> is where the back-up file is

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
  - Recommend APNIC secondary zone
- < /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

Subdomains of in-addr.arpa domain

- Example: an organisation given a /16
  - 192.168.0.0/16 (one zone file and further delegations to downstreams)
  - 168.192.in-addr.arpa zone file should have:

```
0.168.192.in-addr.arpa.      NS ns1.organisation0.com.
0.168.192.in-addr.arpa.      NS ns2.organisation0.com.
1.168.192.in-addr.arpa.      NS ns1.organisation1.com.
1.168.192.in-addr.arpa.      NS ns2.organisation1.com.
2.168.192.in-addr.arpa.      NS ns1.organisation2.com.
2.168.192.in-addr.arpa.      NS ns2.organisation2.com.
        :
```

Subdomains of in-addr.arpa domain

- Example: an organisation given a /20
  - 192.168.0.0/20 (a lot of zone files!) – have to do it per /24)
  - Zone files

0.168.192.in-addr.arpa.

1.168.192.in-addr.arpa.

2.168.192.in-addr.arpa.

:

:

15.168.192.in-addr.arpa.

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>

Whois domain object

```
domain:       28.12.202.in-addr.arpa
descr:        in-addr.arpa zone for 28.12.202.in-addr.arpa
admin-c:      DNS3-AP
tech-c:       DNS3-AP
zone-c:       DNS3-AP
nserver:      ns.telstra.net
nserver:      rs.arin.net
nserver:      ns.myapnic.net
nserver:      svc00.apnic.net
nserver:      ns.apnic.net
mnt-by:       MAINT-APNIC-AP
mnt-lower:    MAINT-DNS-AP
changed:      inaddr@apnic.net 19990810
source:       APNIC
```

Reverse Zone

Contacts

Name Servers

Maintainers (protection)

Removing lame delegations

- Objective
  - To repair or remove persistently lame DNS delegations
- DNS delegations are lame if:
  - Some or all of the registered DNS nameservers are unreachable or badly configured
- APNIC has formal implementation of the lame DNS reverse delegation procedures

Questions ?

Thank you ☺!

<champika@apnic.net>