DNS Security : DNSSEC Deployment

July 15, 2012, Karachi, Pakistan

In conjunction with

APNIC

Presenters

- Champika Wijayatunga
  - Training Unit Manager, APNIC
  - champika@apnic.net

- Vivek Nigam
  - Internet Resource Analyst, APNIC
  - vivek@apnic.net

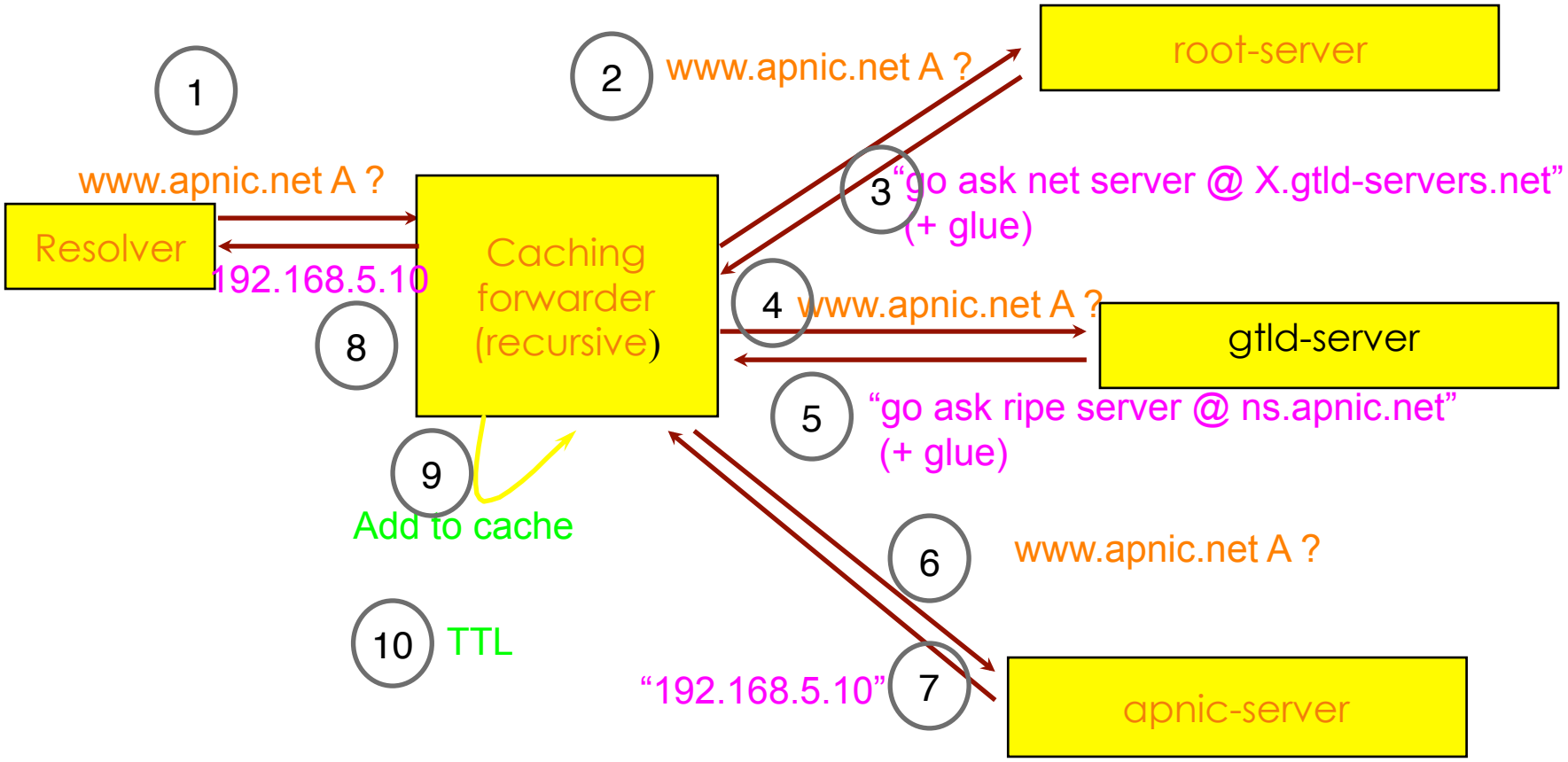
Background - Why DNSSEC?

- The original DNS protocol wasn't designed with security in mind
- It has very few built-in security mechanism
- As the Internet grew it was realized that DNS spoofing was to easy
- DNSSEC and TSIG were develop to help address this problem

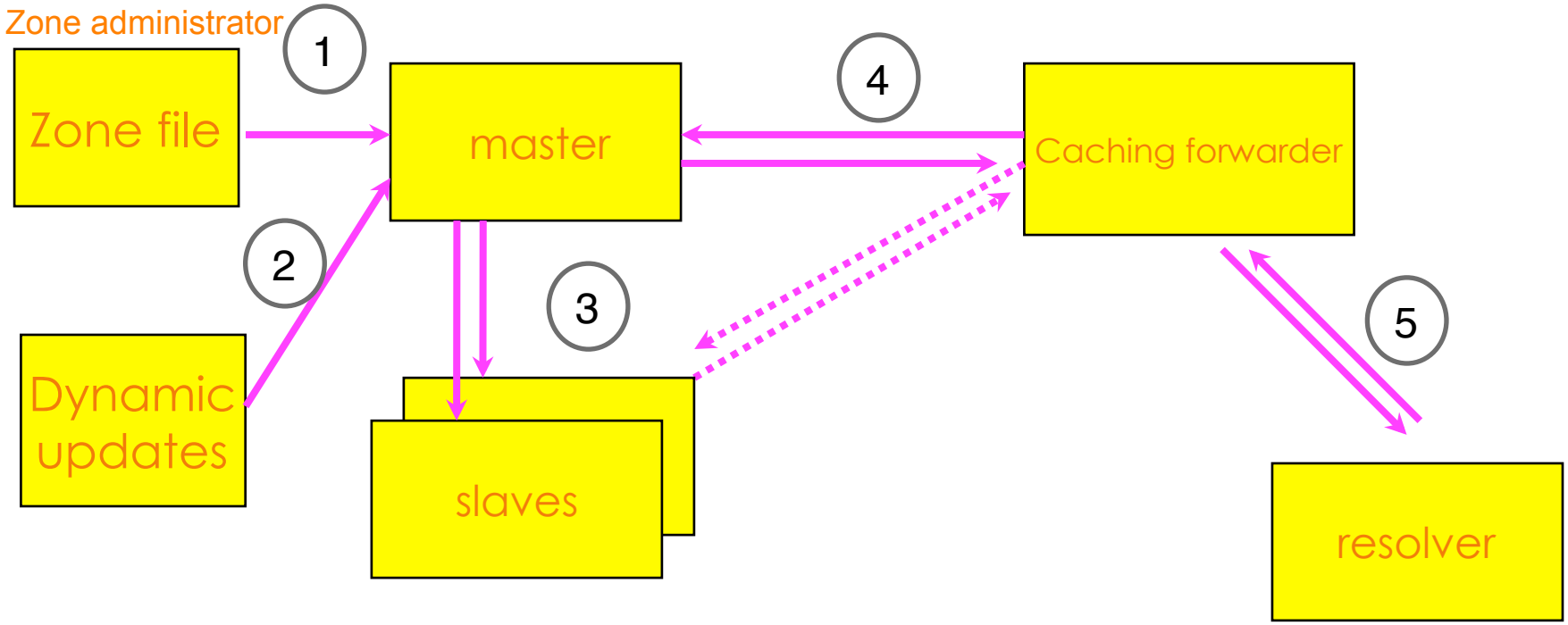
Recap: DNS Resolving

Question:

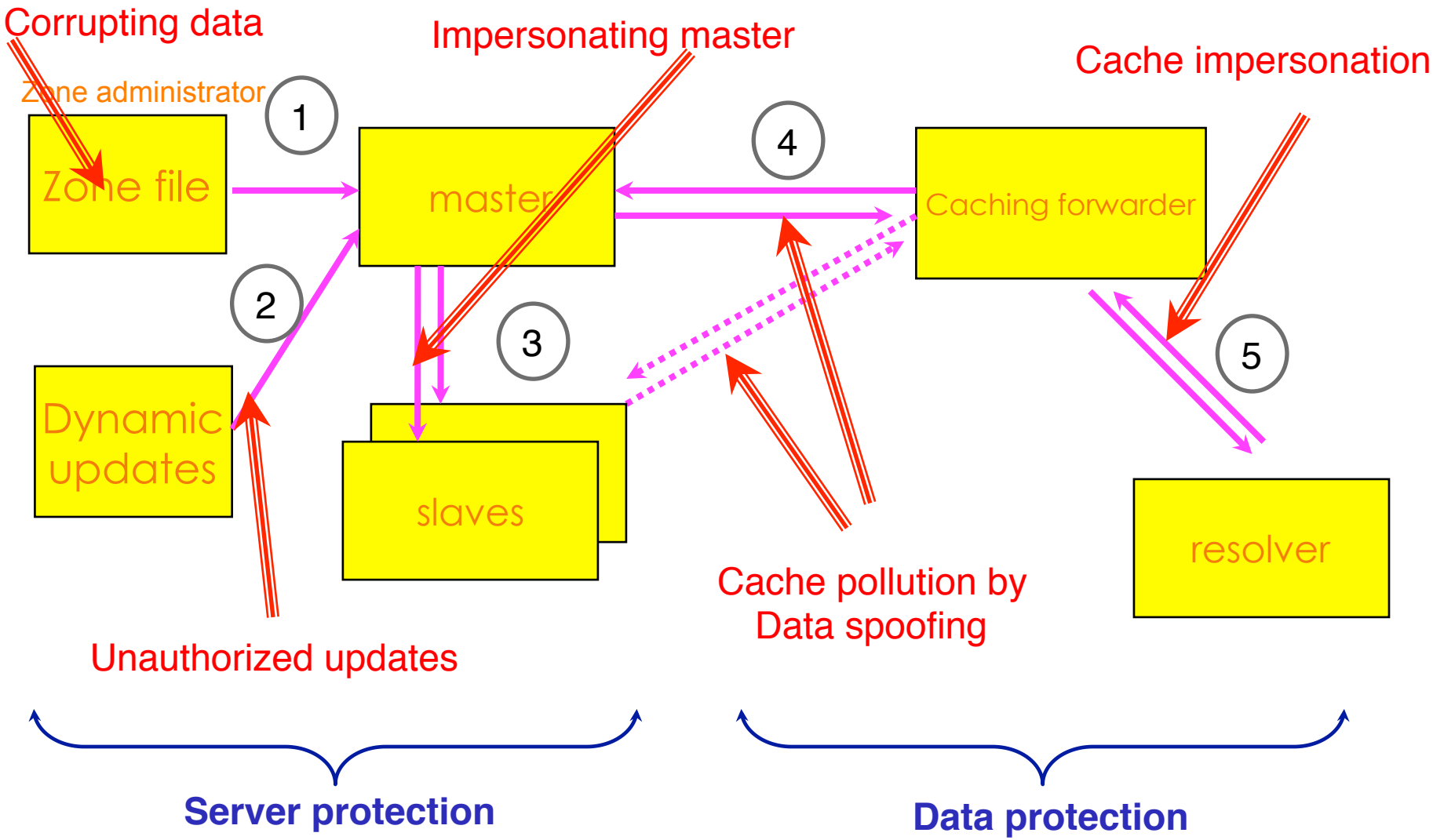
www.apnic.net A



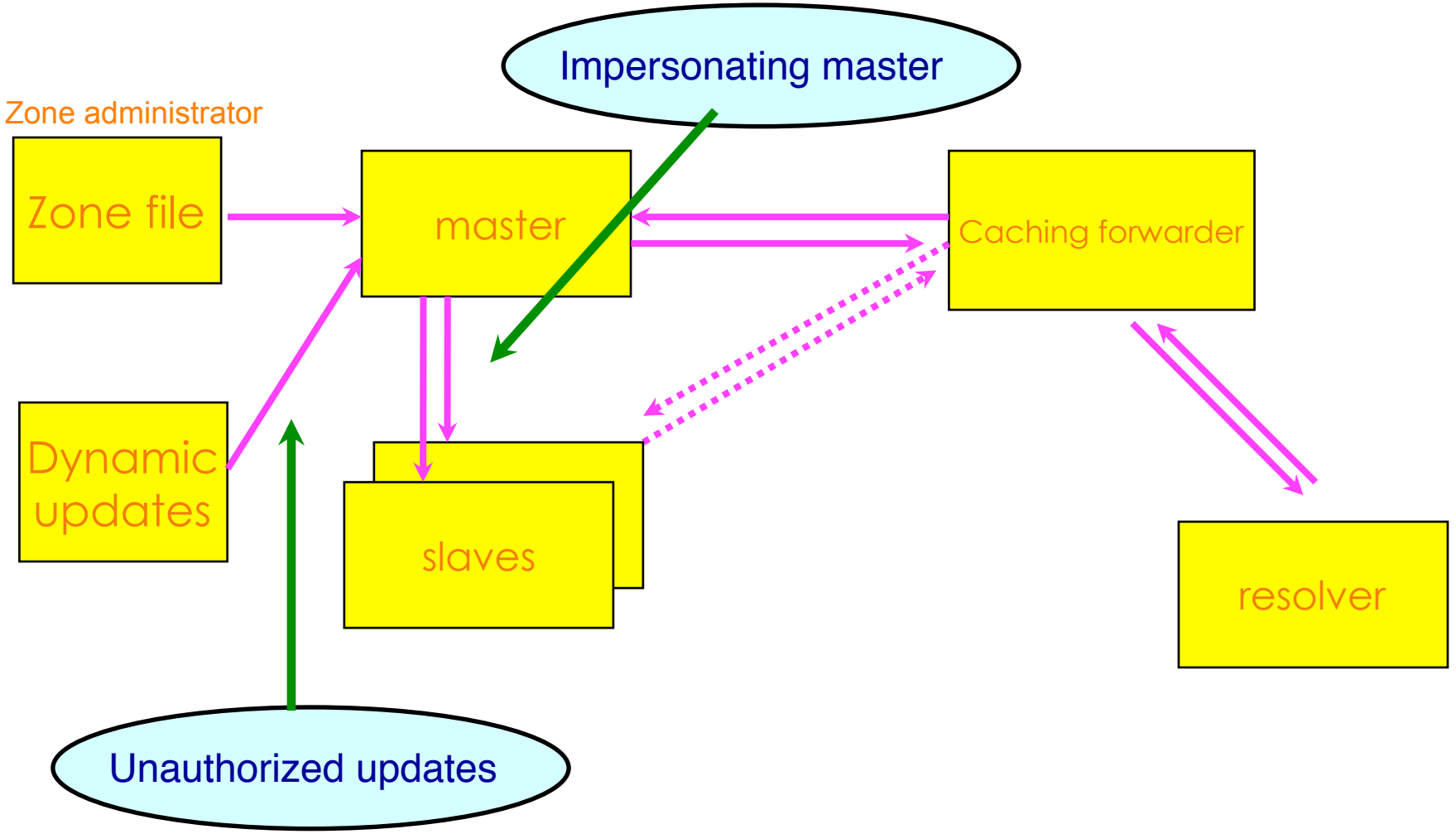
DNS: Data Flow



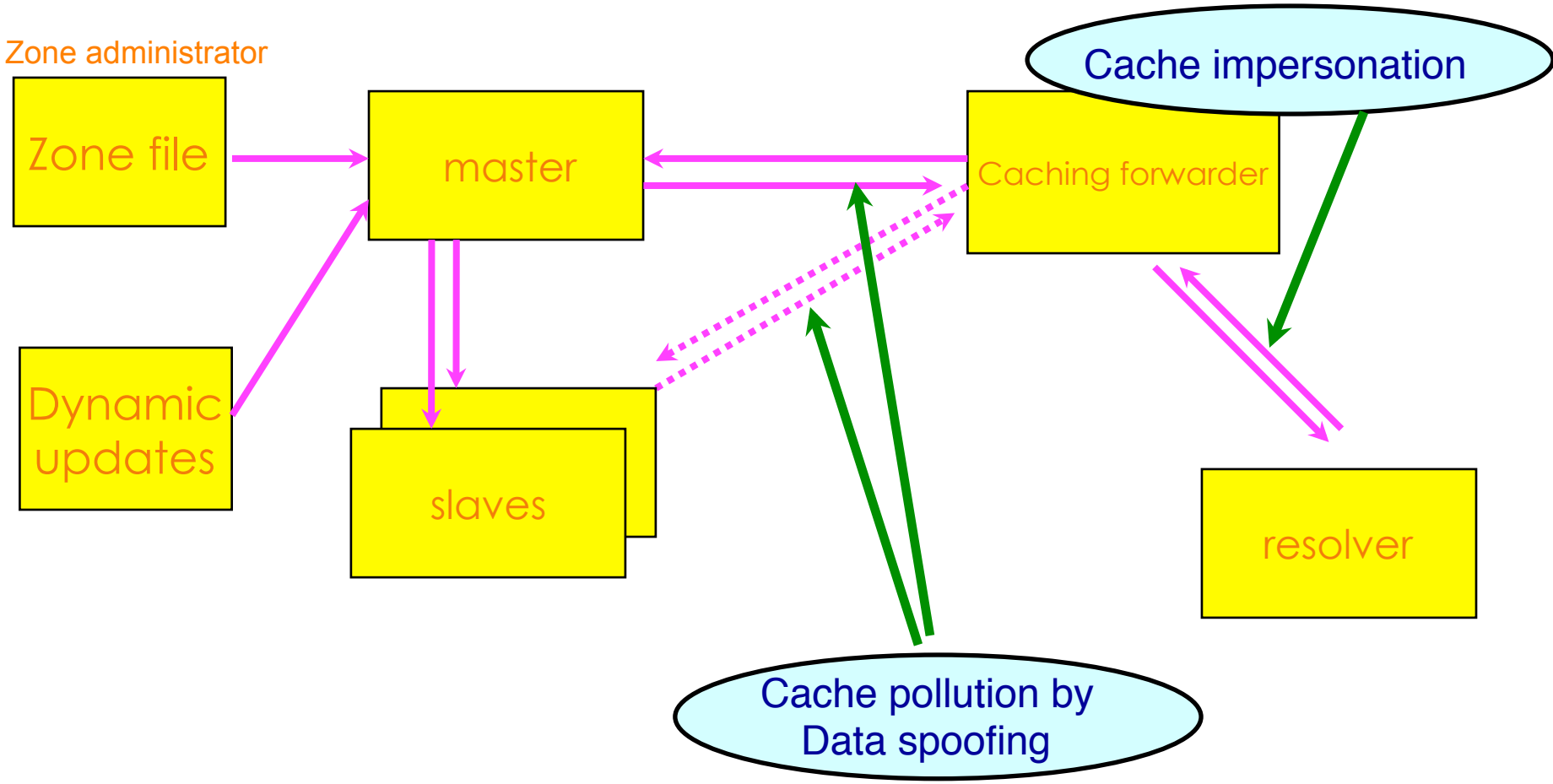
DNS Vulnerabilities



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

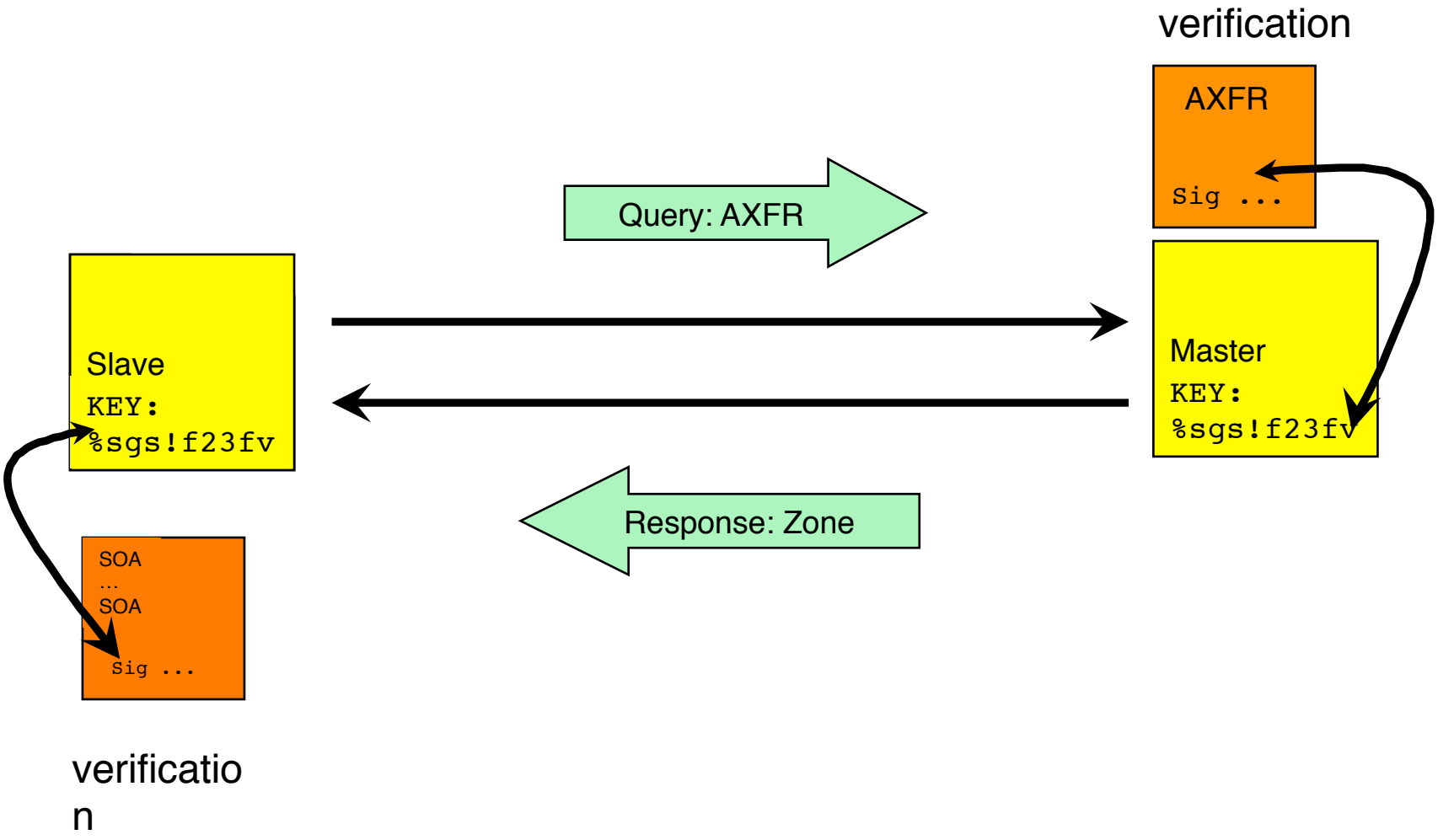
- Based on a shared secret - both sender and receiver are configured with it

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



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>`

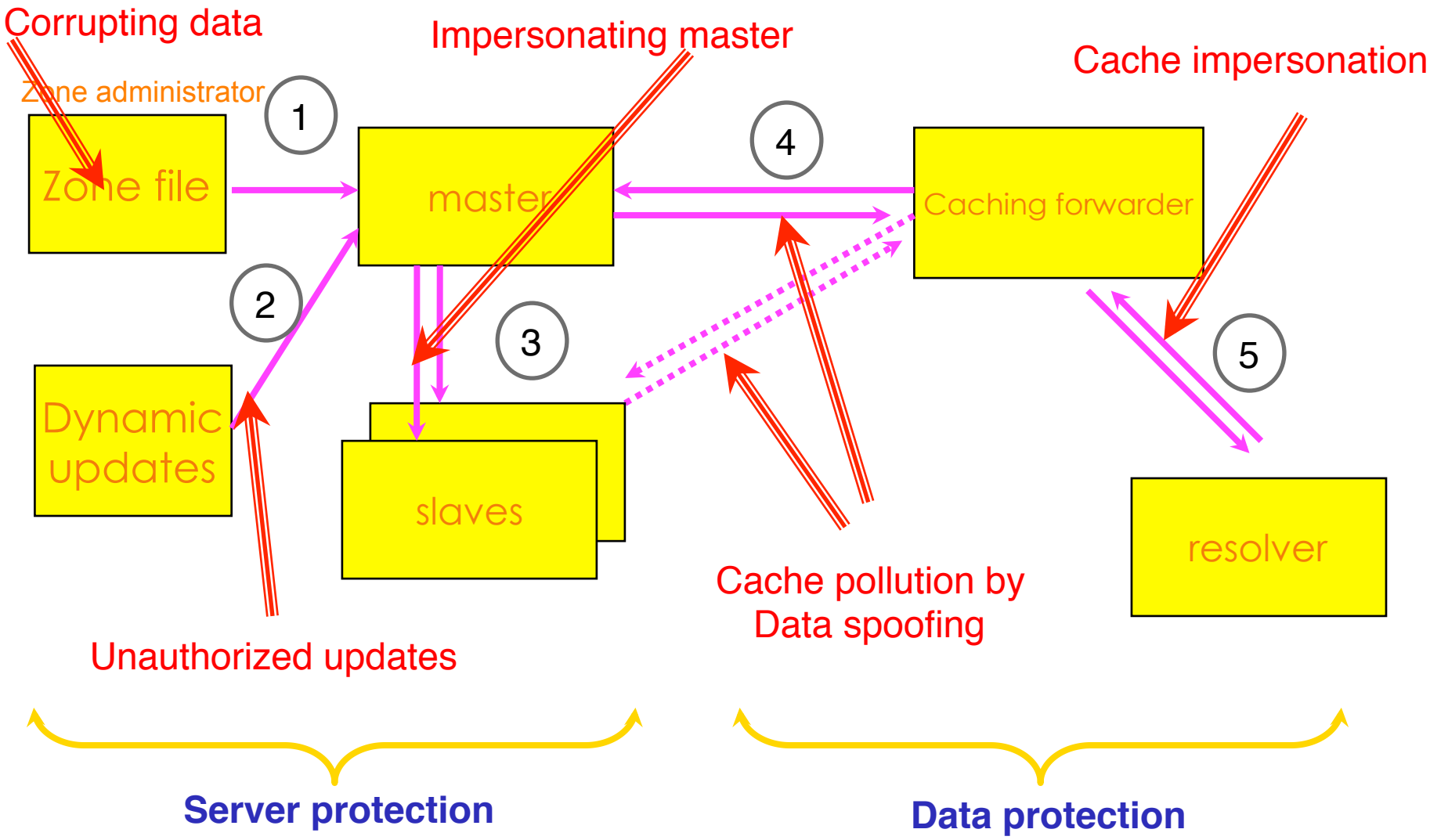
```
$ dig @127.0.0.1 example.net AXFR \
    -k Kns1-ns2.pcx.net.+157+15921.key
```

- 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



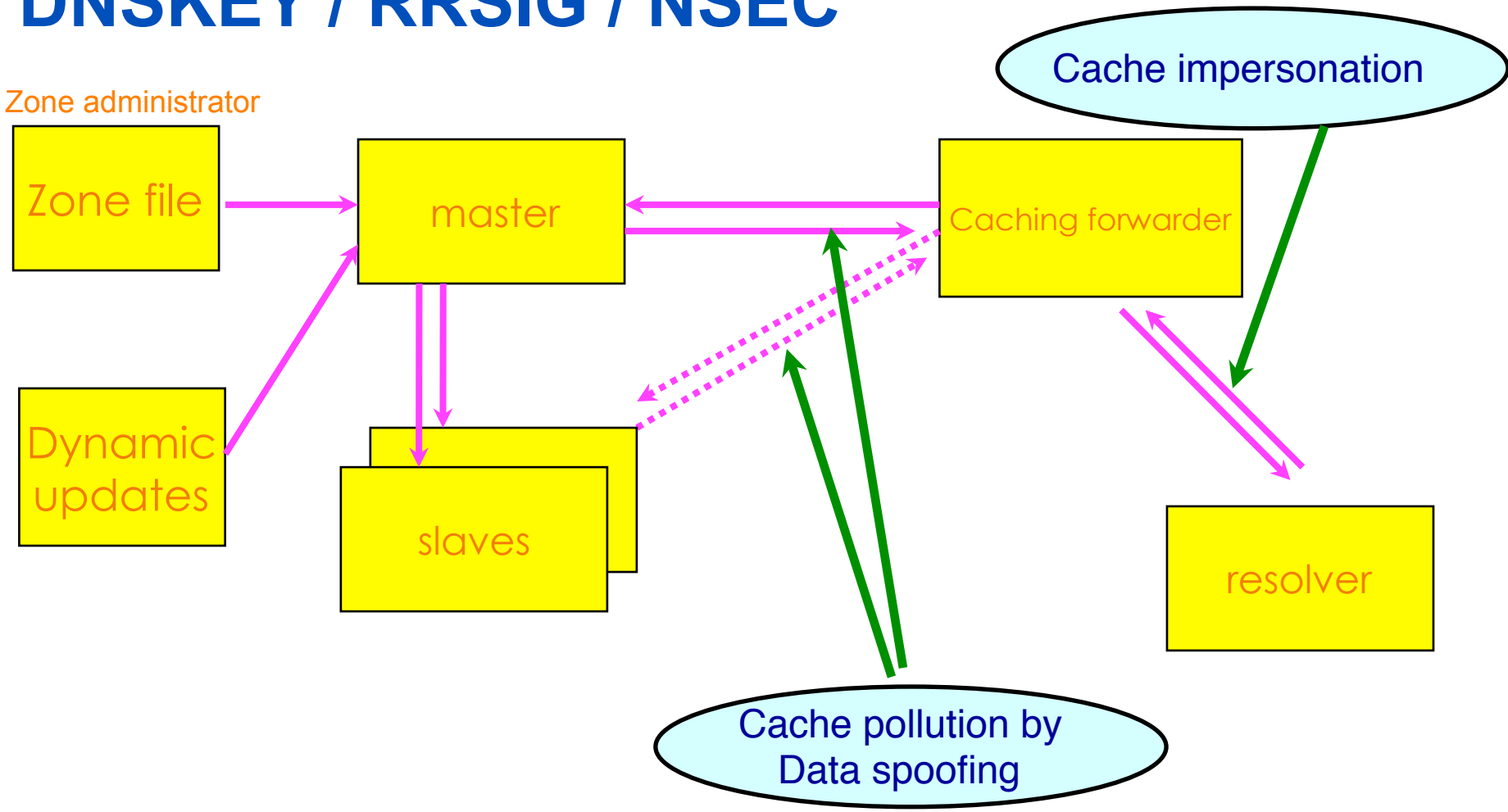
DNSSEC mechanisms

- 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

DNSSEC mechanisms

- Key pair
  - A private(secret) key and a corresponding public key
- In DNSSEC,
  - If you know the public key, you can verify a signature created with the private key
  - Only uses signatures
- Public Key Crypto
  - If you know the public key, you can encrypt data that can only be decrypted with the private key

Vulnerabilities protected by DNSKEY / RRSIG / NSEC



Authenticity and Integrity

- Authenticity
  - Is the data published by the entity we think is authoritative
- Integrity
  - Is the data received the same as what was published?
- Islands of security
  - We cannot expect that every name server in the world would configure to support DNSSEC and every zone is secured
  - Security aware name servers and Security not aware name servers

Publishing keys

- A zone is signed using its private key
- Receiving name server must have access to zone's public key in order to perform the security verification
- How to obtain public key
  - Publish the public key using DNSKEY RR in the zone file
  - Obtain the key using out of band process
    - Trusted anchor (defined using *trusted-keys* statement in config file)

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

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+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?

Thank You! ☺

<champika@apnic.net>

APNIC