



DNSSEC

An Introduction to Concepts

February 2003



Why DNSSEC?

- DNS is not secure
 - ◆ Applications depend on DNS
 - _ Known vulnerabilities
- DNSSEC protects against data spoofing and corruption

Outline

- Introduction
- DNSSEC mechanisms
 - ◆ to authenticate servers (TSIG / SIG0)
 - ◆ 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
- Conclusions



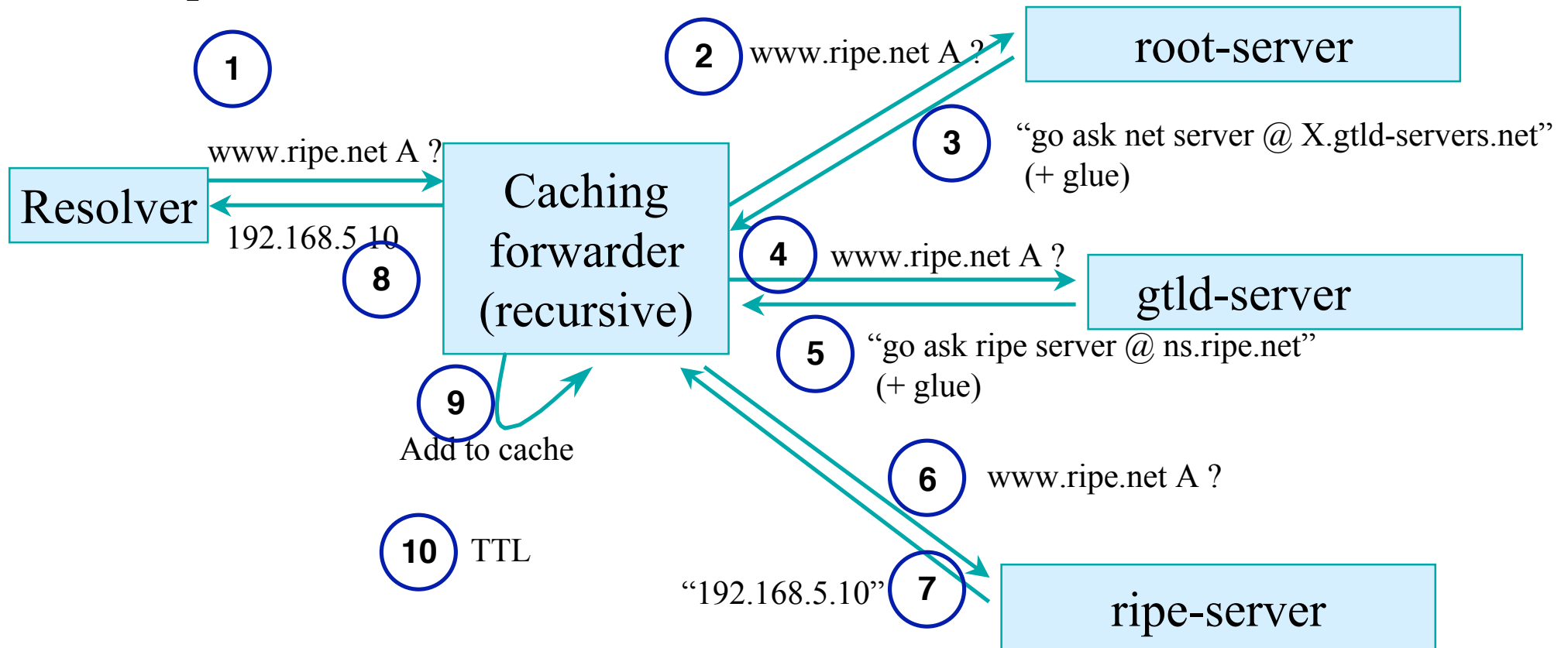
DNS: Known Concepts

- Known DNS concepts:
 - ◆ Delegation, Referral, Zone, RRs, label, RDATA, authoritative server, caching forwarder, stub and full resolver, SOA parameters, etc
 - ◆ Don't know? Do ask!

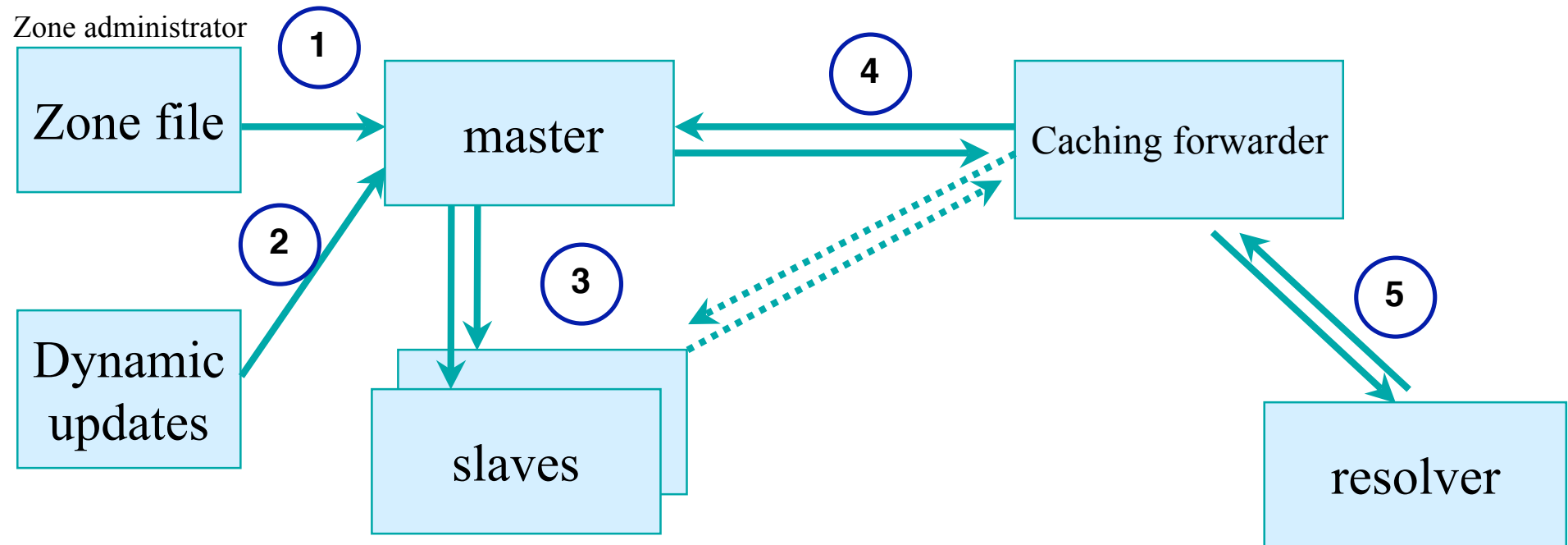
Reminder: DNS Resolving

Question:

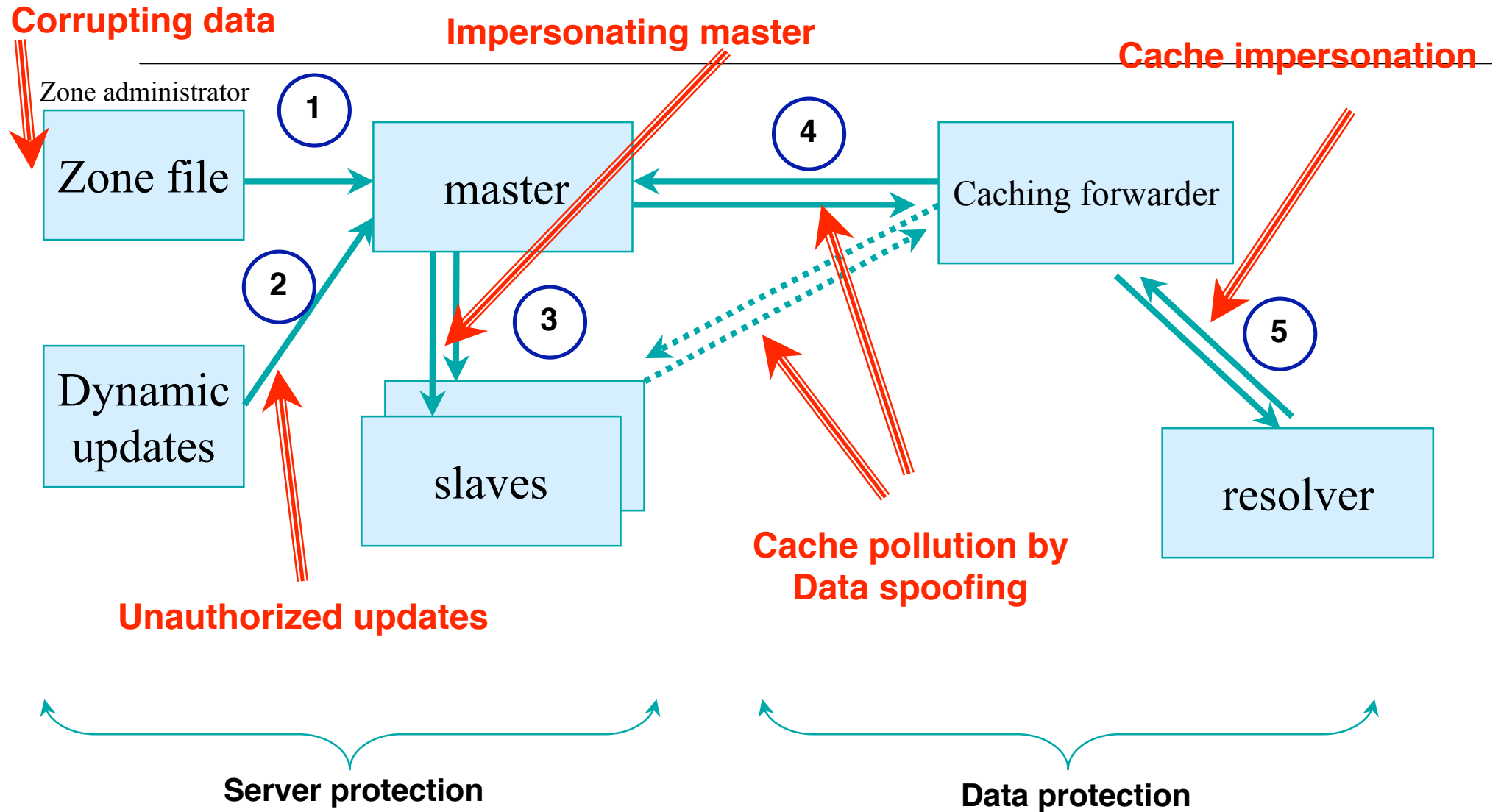
www.ripe.net A



DNS: Data Flow



DNS Vulnerabilities



DNS Protocol Vulnerability

- DNS data can be spoofed and corrupted on its way between 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)?

Motivation for DNSSEC

- DNSSEC protects against data spoofing and corruption
- DNSSEC (TSIG) provides mechanisms to authenticate servers
- DNSSEC (KEY/SIG/NXT) provides mechanisms to establish authenticity and integrity of data
- A secure DNS will be used as a public key infrastructure (PKI)
 - ◆ However it is **NOT** a PKI

DNSSEC Current State

- This tutorial is based on the ‘current’ RFC2535 with modifications
- Changes to the specs that are now going through the IETF:
 - ◆ Rewrite of the specs; mainly an editing job;
 - ◆ Incorporation of operational experiences;
 - ◆ Changes not backward compatible with current specs!
 - ◆ E.g. introduction of DS, NXT, NXT opt-in, AD bit, etc

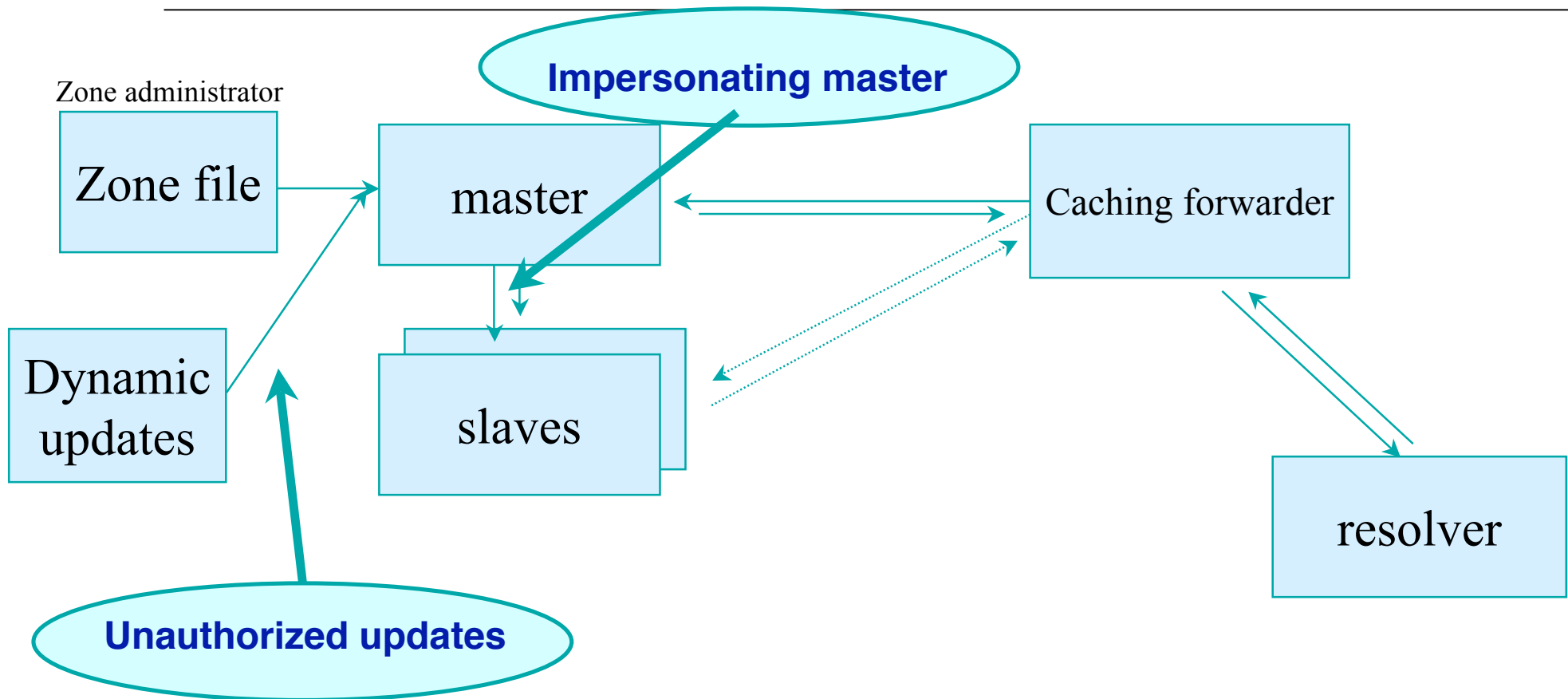


DNSSEC Mechanisms to Authenticate Servers

- TSIG
- SIG0



TSIG Protected Vulnerabilities

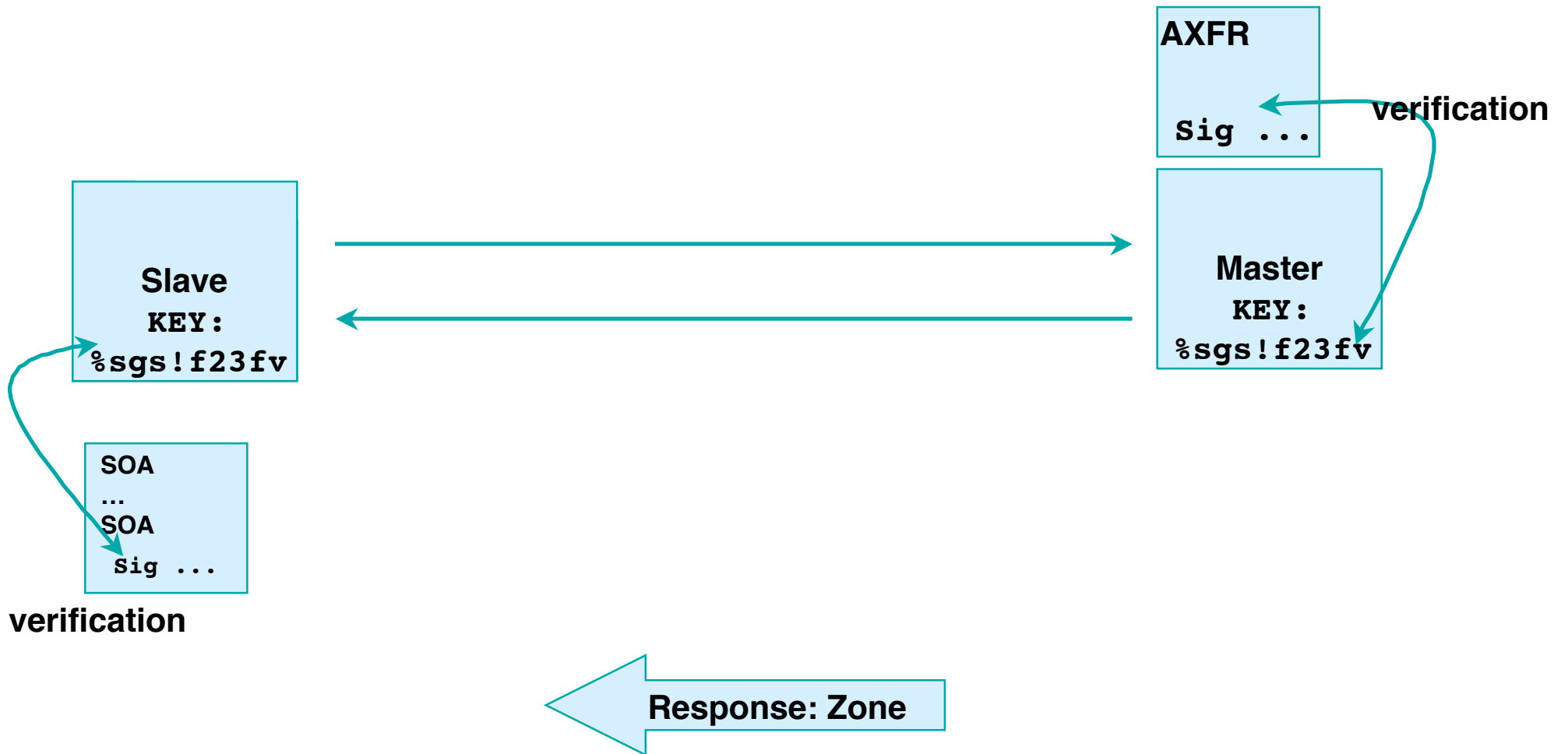


Transaction Signature: TSIG

- TSIG (RFC 2845)
 - ◆ authorizing dynamic updates & zone transfers
 - ◆ authentication of caching forwarders
 - ◆ can be used without deploying other features of DNSSEC
- One-way hash function over:
 - ◆ DNS question or answer
 - ◆ & the timestamp
- Signed with “shared secret” key
- Used in server configuration, not in zone file

TSIG example

Query: AXFR





Authenticating Servers Using SIG0

- Alternatively its possible to use SIG0
 - ◆ Not widely used yet
 - ◆ Works well in dynamic update environment
- Public key algorithm
 - ◆ Authentication against a public key published in the DNS



Summary: Steps to TSIG Configuration

- Configuring secure transfers between servers with TSIG
 1. Generate a key using “DNSSEC-keygen”
 2. Communicate key with your partner (off-band, PGP...)
 3. Configure your server to require the key for zone transfers
 - ♦ “**key**” statement to configure the key
 - ♦ “**allow-transfer**” statement in the “**zone**” statement
 - ♦ tip: use “**include <file_name>**”
 4. Have your partners configure their servers to use the key when talking to you
 - ♦ Using the “**server**” statement



Importance of the Time Stamp

- TSIG/SIG0 signs a complete DNS request / response with time stamp
 - ◆ to prevent replay attacks
 - ◆ 'seconds since epoch'
- Operational problems when comparing times
 - ◆ Make sure your local time zone is properly defined
 - ◆ **date -u** will give UTC time, easy to compare between the two systems
- Use NTP synchronization!!!

TSIG: Questions?





DNSSEC Mechanisms to Establish Authenticity and Integrity of Data

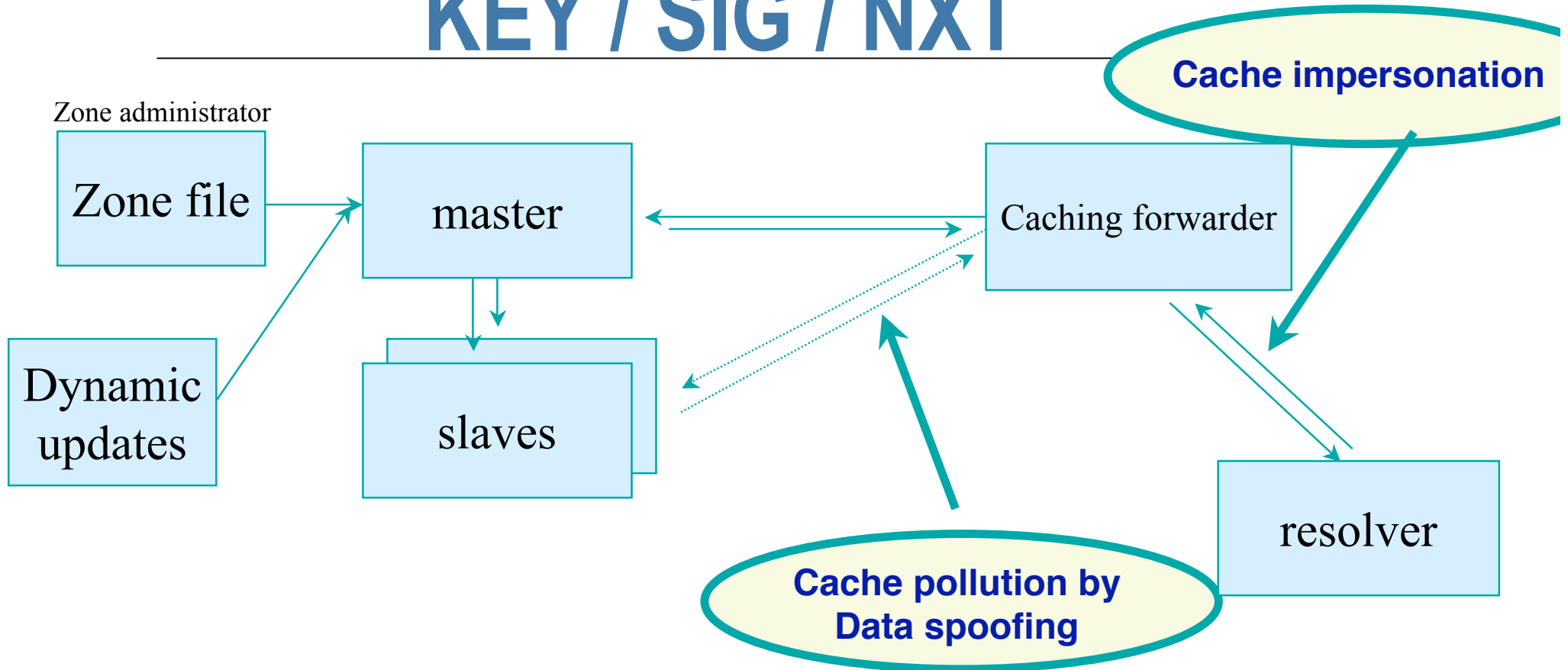
=> **Quick overview**

- New RRs
- Using public key cryptography to sign a single zone
- Delegating signing authority ; building chains of trust
- Key exchange and rollovers





Vulnerabilities protected by KEY / SIG / NXT





DNSSEC Summary on 1 page

- Data authenticity and integrity by SIGning the resource records
- Public KEYs used to verify the SIGs
- Children sign their zones with their private key; The authenticity of their KEY is established by a SIGNature over that key by the parent (DS)
- In the ideal case, only one public KEY needs to be distributed off-band

Authenticity and Integrity of Data

- Authenticity: Is the data published by the entity we think is authoritative?
- Integrity: Is the data received the same as what was published?
- Public Key cryptography helps to answer these questions
 - ◆ signatures to check both integrity and authenticity of data
 - ◆ verifies the authenticity of signatures

Public Key Crypto Reminder

- Key pair: a secret (or private) key and a public key
- Simplified:
 - ◆ If you know the public key, you can decrypt data encrypted with the secret key
 - ◆ Usually an encrypted hash value over a published piece of information; the owner is the only person who can construct the secret. Hence this a signature
 - ◆ If you know the secret key, you can decrypt data encrypted with the public key
 - ◆ data is usually an encrypted key for symmetric cipher
- PGP uses both, DNSSEC only uses signatures

Public Key Crypto Issues

- Public keys need to be distributed
- Secret keys need to be kept secret
- Public key cryptography is 'slow'
- Math:
 - ◆ The security of the cryptosystem is based on a set of mathematical problems for which guessing a solution requires scanning a huge solution space (e.g. factorization)
 - ◆ Algorithms e.g.: DSA, RSA, elliptic curve
 - ◆ RSA/SHA1 is a good choice
 - ✦ Better than RSA/MD5



New Resource Records for DNSSEC



DNSSEC New RRs

- 3 Public key crypto related RRs
 - ◆ SIG Signature over RRset made using private key
 - ◆ KEY Public key, needed for verifying a SIG over a RRset
 - ◆ DS Delegation Signer; 'Pointer' for building chains of trust

- One RR for internal consistency
 - ◆ authenticated non-existence of data
 - ◆ NXT Indicates which RRset is the next one in the zone



Other Keys in the DNS

- For non DNSSEC, public keys can appear in the DNS
- CERT
 - ◆ For x509 certificates
- Under discussion/development are application keys
 - ◆ IP-SEC
 - ◆ SSH

Recap: RRs and RRsets

- Resource Record:

◆ name	TTL	class	type	rdata
<code>www.ripe.net.</code>	<code>7200</code>		<code>IN</code>	<code>A</code>
<code>192.168.10.3</code>				

- All RRs of a given name, class, type make an RRset:

<code>www.ripe.net.</code>	<code>7200</code>	<code>IN</code>	<code>A</code>	<code>192.168.10.3</code>
			<code>A</code>	<code>10.0.0.3</code>

- In DNSSEC the RRsets are signed, not the individual RRs

KEY RDATA

- 16 bits: FLAGS
- 8 bits: protocol
- 8 bits: algorithm
- N*32 bits: public key

Example:

ripe.net. 3600 IN **KEY** 256 3 3 (

AQOvhvXXU61Pr8sCwELcqqq1g4JJ
CALG4C9EtraBKVd+vGIF/unwigfLOA
O3nHp/cgGrG6gJYe8OWKYNgq3kDChN)

SIG RDATA

- 16 bits - type covered
- 8 bits - algorithm
- 8 bits - nr. labels covered
- 32 bits - original TTL

www.ripe.net. 3600 IN SIG A 1 3 3600 (

20010504144523 20010404144523 3112 ripe.net.

VJ+8ijXvbrTLeoAiEk/qMrdudRnYZM1VlqhN
vhYuAcYKe2X/jqYfMfjFSUrmhPo+0/GOZjW
66DJubZPmNSYXw==) signature field

- 32 bit - signature expiration
- 32 bit - signature inception
- 16 bit - key tag
- signers name

NXT RDATA

- Points to the next domain name in the zone
 - ◆ also lists what are all the existing RRsets for “name”
- N*32 bit type bit map
- Used for authenticated denial-of-existence of data
 - ◆ authenticated non-existence of TYPEs and labels
- Example:

```
www.ripe.net. 3600 IN      NXT ripe.net. A SIG NXT
```

NXT Record

```

$ORIGIN ripe.net.
@      SOA      ....
      NS      NS.ripe.net.
      KEY      ....
      NXT      mailbox.ripe.net. SOA NS NXT KEY SIG
mailbox A      192.168.10.2
      NXT      www.ripe.net.  A NXT SIG
WWW A      192.168.10.3
      NXT      ripe.net.  A NXT SIG
  
```



'popserver' is missing

- ♦ query for popserver.ripe.net would return:
aa bit set RCODE=NXDOMAIN
authority: mailbox.ripe.net. NXT www.ripe.net. A NXT SIG
- ♦ query for www.ripe.net MX would return: an empty answer section and the www NXT record in the authority section

Meaning of NXT

- If you query for data does not exist in a zone, the NXT RR provides proof of non-existence
- If after a query the response is:
 - ◆ NXDOMAIN: One, and maybe many more, NXT RRs indicate that the name or a wildcard expansion does not exist
 - ◆ NOERROR and empty answer section: The NXT TYPE array proves that the QTYPE did not exist
- NXT records are generated by tools
 - ◆ You do not have to generate NXT RRs by hand

FYI: NXT opt-in Variant

- New variety of the NXT resource record
 - ◆ Introduced to cope with the problem that in a secure zone each name is accompanied by a NXT RR with a SIG
- Instead of authenticated denial of existence it indicates authenticated denial of security
- The change in semantic is indicated by leaving the NXT from the bitmap
- Only at delegation points

NXT opt-in Variant

- Still under discussion in the IETF
 - ◆ First implementations have been tested

a.com	ns	ns.a.com
a.com	NXT	SIG NS w.com
	SIG	NXT
b.com	NS	ns.b.com
c.com	NS	ns.c.com
w.com	NS	ns.w.com
	NXT	SIG NS z.com
	SIG	NXT
z.com	NS	ns.z.com
	NXT	SIG NS .com
	SIG	NXT

Question for non-existent ba.com will return:

NXDOMAIN

Auth: A.COM NXT SIG NS w.com

One can not be sure ba.com does not exist.

New DNS RRs: Questions?





DNSSEC Signing of a Local Zone

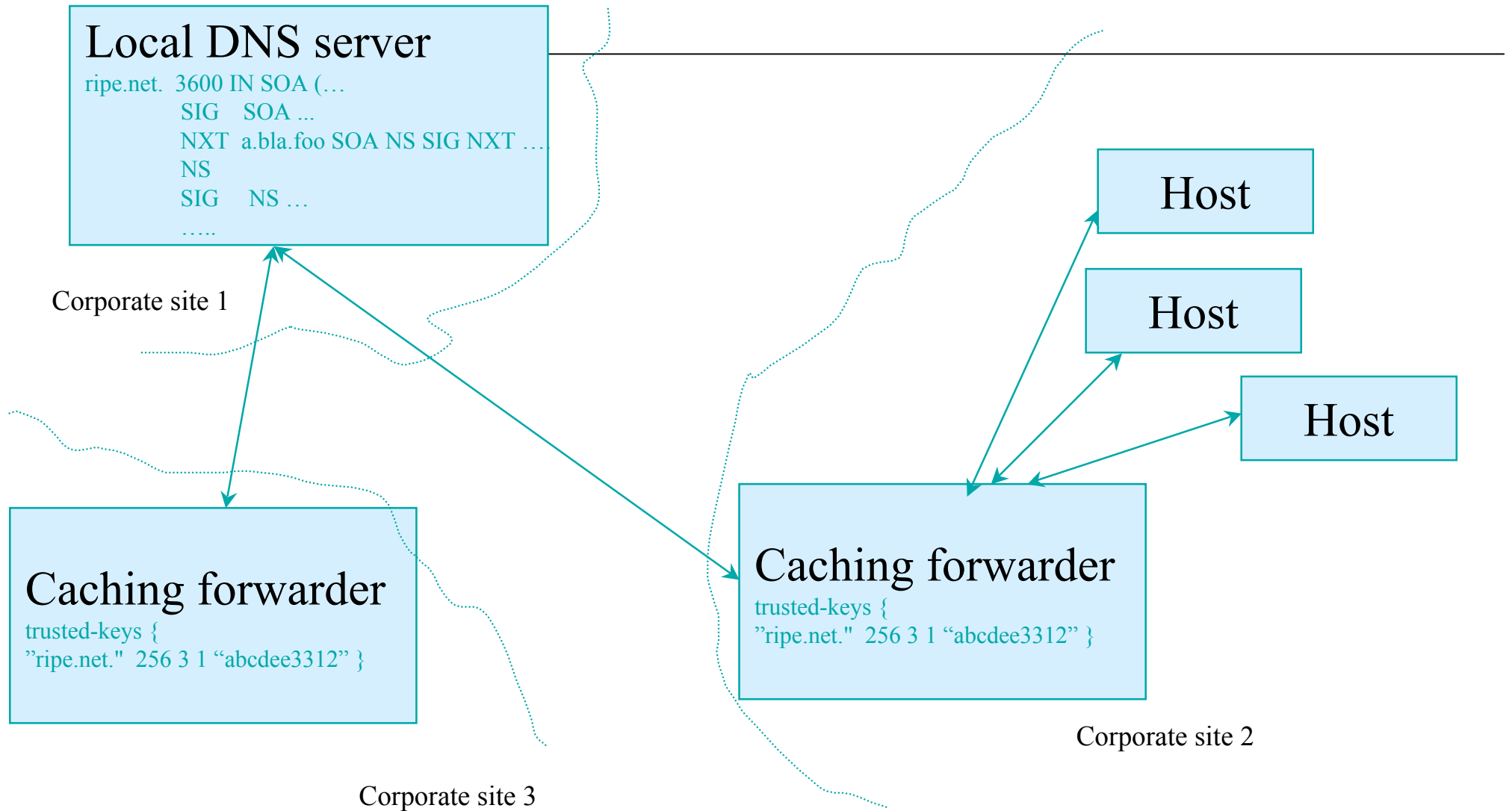




DNSSEC Signing of a Local Zone

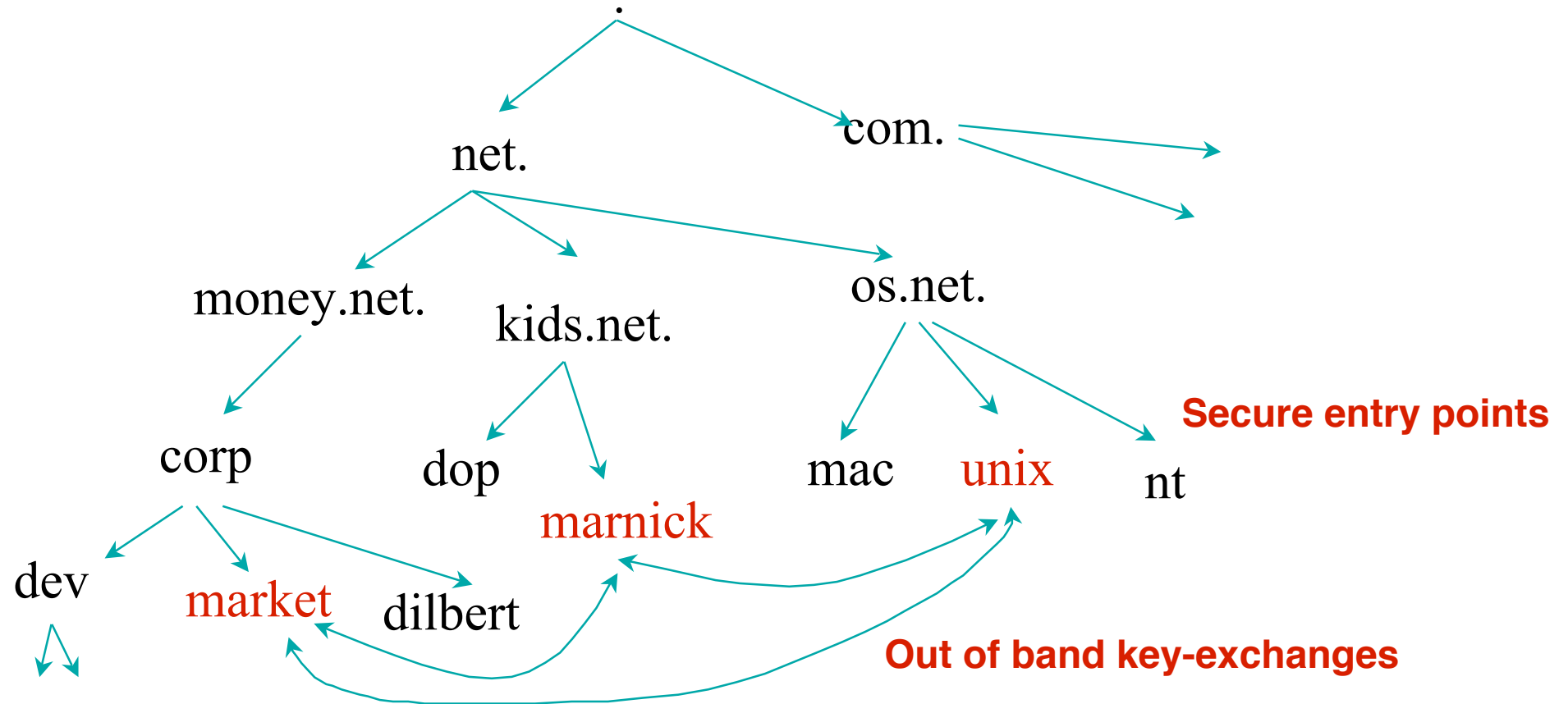
1. Generate keys and include them in the zone file
2. Sign your zone; signing will:
 - ◆ sort the zone
 - ◆ insert the NXT records
 - ◆ insert SIG-s containing a signature over each RRset
 - ◆ made with your private key
 - ◆ generate key-set file (used later)
3. Distribute the Public KEY to those that need to be able to trust your zone
 - ◆ they configure your key in their resolver
 - ◆ thus configuring “secure entry point” in the tree

Locally Signed Zone



Locally Secured Zones

- Key distribution problem for distributing keys
 - ◆ It would be better if the whole tree would be secured!



Signing Local Zone: Questions?





Delegating Signing Authority

building chains of trust





Using the DNS to Distribute Keys

- Securing a DNS zone tree
- Building chains of trust from the root down
- Tools: KEY, SIG and DS records
- This material is based on new developments
 - ◆ Only in bind9.3.0 November 15 snapshot or later !

Chain of Trust

- The goal is to build a chain of trust from the root down the DNS tree
- You need to verify the public keys with which signatures over other keys are made
- Parents need to sign the keys of their children
- *Outline:*
 - ◆ *Which key is used to make a SIG*
 - ◆ *How do parents sign children keys*
 - ◆ *Walking the chain of trust*



SIG RDATA

Recap for next slides

```
www.ripe.net.      3600 IN  SIG      A 1 3 3600  
20010504144523 (    20010404144523 3112 ripe.net.  
                  VJ+8ijXvbrTLeoAiEk/qMrduRnYZM1VlqhN  
                  vhYuAcYKe2X/jqYfMfjfSUrmhPo+0/GOZjW  
                  66DJubZPmNSYXw== )
```

This field indicates the signer.

Delegation Signer (DS)

- The parent delegates authority to sign DNS RRs to the child using this RR
- DS is a pointer to the next key in the chain of trust
 - ◆ You may trust data that is signed using a key that the DS points to
- New RR to solve problems with key-rollovers
 - ◆ More on that later

DS RDATA

- 16 bits: key tag
- 8 bits: algorithm
- 8 bits: digest type
- 20 bits: SHA-1 Digest

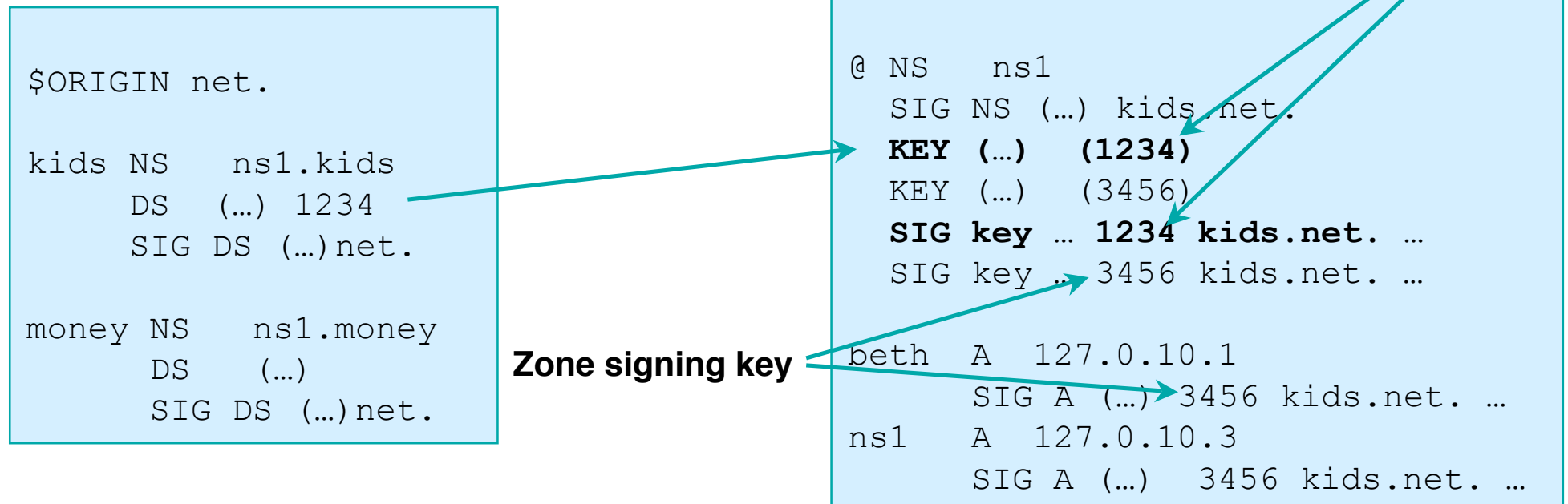
This field indicates which key is the next in the chain of trust

```
$ORIGIN ripe.net.
```

```
disi.ripe.net      3600 IN      NS      ns.disi.ripe.net
disi.ripe.net.     3600 IN      DS      3112 1 1 (
                                     239af98b923c023371b52
                                     1g23b92da12f42162b1a9
                                     )
```

Delegating Signing Authority

- Parent signs the DS record pointing to the key signing key



- The parent is authoritative for the DS RR of its children

Key / Zone Signing Keys

Only an administrative distinction, you cannot tell from the KEY record itself!

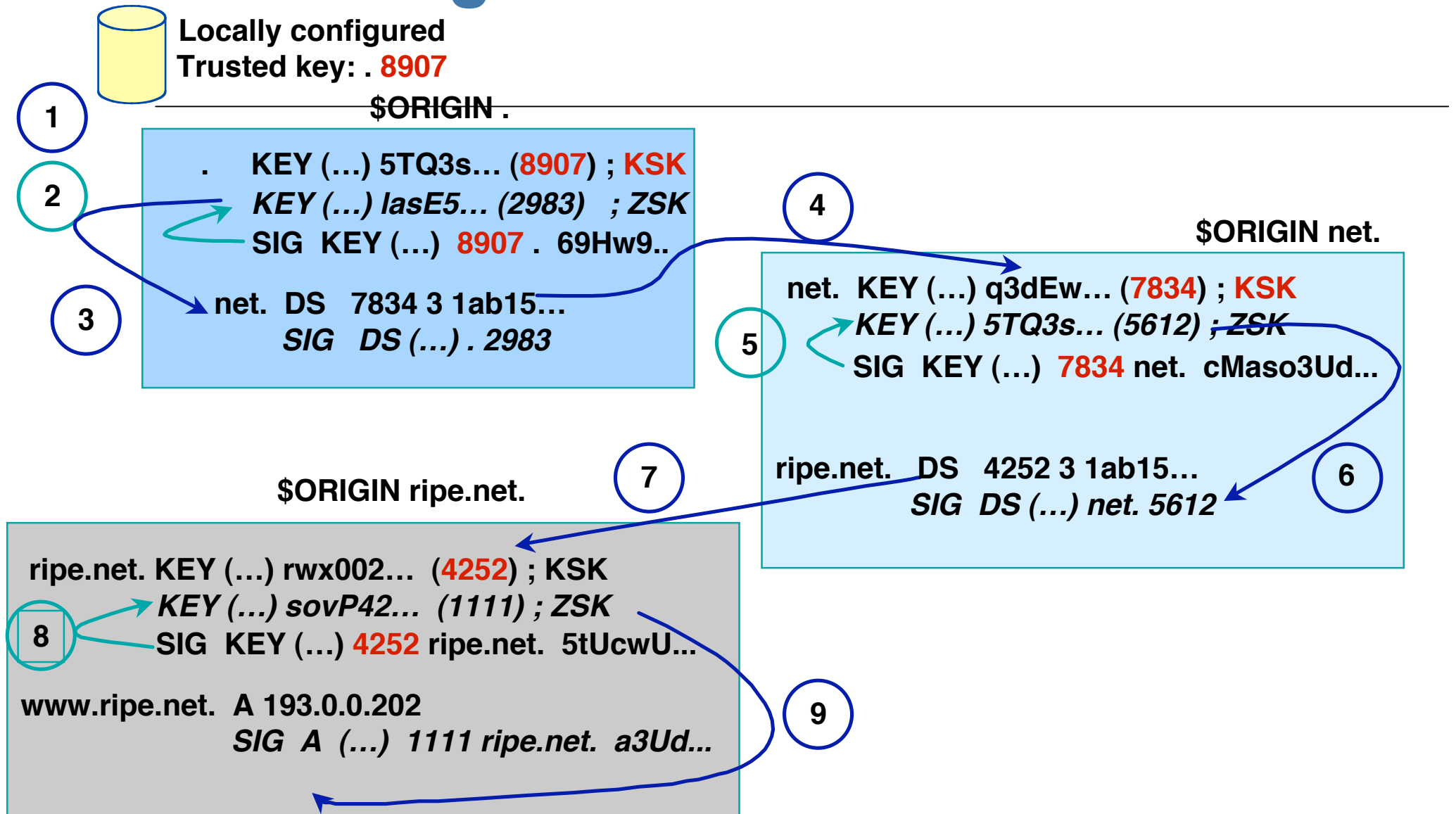
- DS points to a key signing key (KSK)
- The zone is signed with a zone signing key (ZSK)
 - ◆ (these keys may be the same)
- Key signing key may be long lived, and “bigger”
- Zone signing key may be short lived
 - ◆ can be “smaller” = “faster”



Chain of Trust Verification, Summary

- Data in zone can be trusted if signed by a Zone-Signing-Key
- Zone-Signing-Keys can be trusted if signed by a Key-Signing-Key
- Key-Signing-Key can be trusted if pointed to by trusted DS record
- DS record can be trusted
 - ◆ if signed by the parents Zone-Signing-Key
 - or
 - ◆ DS or Key records can be trusted if exchanged out-of-band and locally stored (Secure entry point)

Walking the Chain of Trust



RFC3090 Terminology

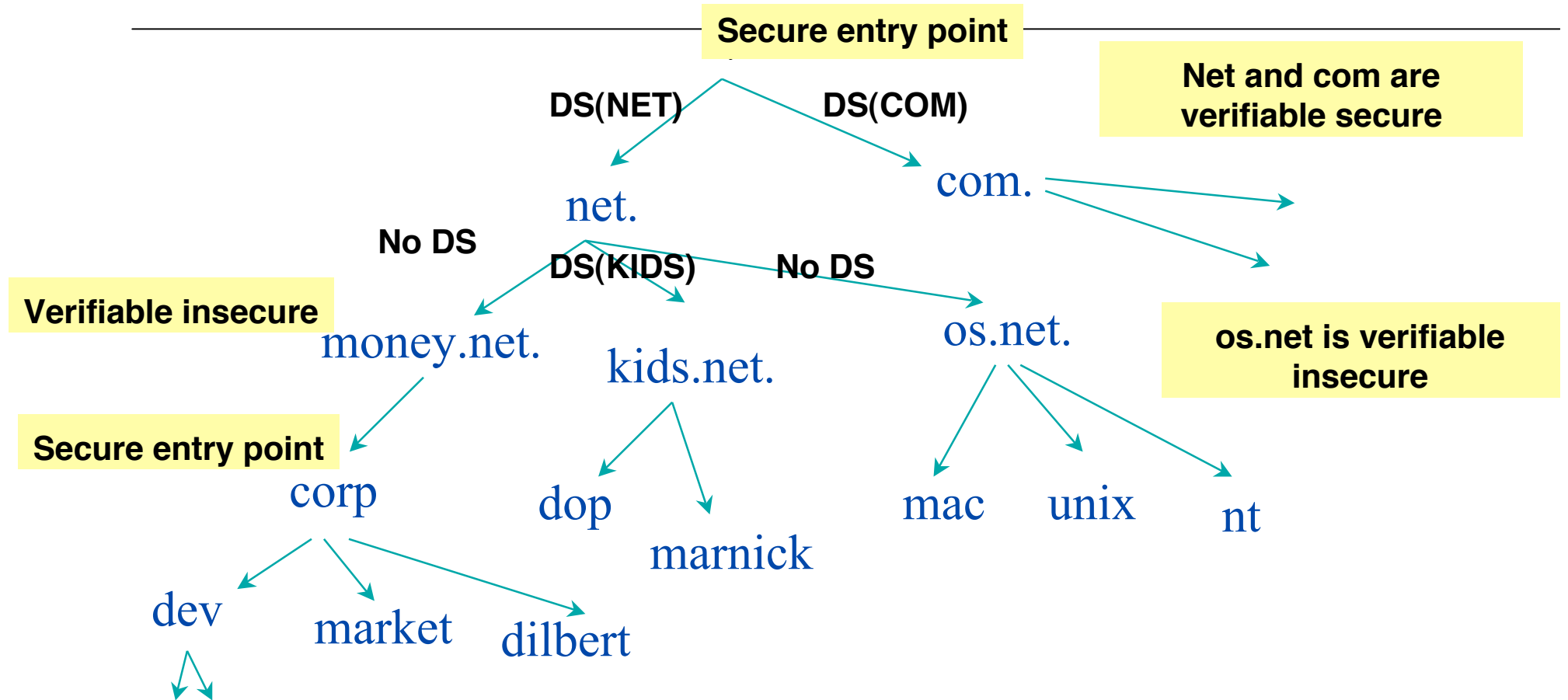
- Verifiable Secure
 - ◆ RRset and its SIG can be verified with a KEY that can be chased back to a trusted key, the parent has a DS record
- Verifiable Insecure
 - ◆ RRset sits in a zone that is not signed and for which the parent has no DS record (more next slide)
- BAD
 - ◆ RRset and its SIG can not be verified (somebody messed with the sig, the RRset, or the SIG expired)
 - ◆ A zone and its subzones are BAD when the parent's SIG over the Child's key is BAD



Insecure Children

- Cryptographic evidence for the verifiably insecure zone status is given by parent
- If there is no DS record as proved by a NXT record with valid signature, the child is not secured
- A child may contain signatures but these will not be used when building a chain of trust
- In RFC2535 the parent has a “NULL” key with a signature

Illustrated Terminology



Resolver has key of root and corp.money.net configured as secure entry points

Building the Chain of Trust

- The child has to:
 - ◆ be secure (see “Signing the local zone”)
 - ◆ upload (off-band) the KSK to the parent
- The parent has to:
 - ◆ generate the DS record from the KSK of the child
 - ◆ sign the DS record with his own ZSK (re-sign his zone)
- Then the parent has to repeat the process, going to his own parent, and so on, till the "." (root)

All of this is done automatically - using tools

Parental signature

adopting orphans carefully...

- Parents needs to check if the child KEY is really their child's... Did you get the KEY from the source authoritative for the child zone?
- This needs an out-of-DNS identification

Open operational issue:

- How do you identify the KEY comes from an authoritative source?
 - ◆ Billing information?
 - ◆ Phone call?
 - ◆ Secret token exchange via surface mail?



The DNS is not a Public Key Infrastructure (PKI)

- All procedures on the previous slide are based on local policy i.e. policy set by the zone administrator
- A PKI is as strong as it's weakest link, we do not know the strength of the weakest link
 - ◆ Certificate Authorities control this by SLAs
- If the domain is under one administrative control you might be able to enforce policy



The DNS is not a PKI (cont'd)

- The DNS does not have Certificate Revocation Lists
 - ◆ There is no way to explicitly say: Do not trust that KEY
- But it is closest to a globally secured distributed DB
 - ◆ IPsec distribution of key material
 - ◆ opportunistic keys; if there is a key in the DNS and nothing better we'll use it
 - ◆ discussions on using the DNS for key distribution
 - ◆ <keydist@cafax.se>

DS: Questions?





Key Exchange and Rollovers



Why Key Exchange

- You have to keep your private key secret
- Private key can be stolen
 - ◆ Put the key on stand alone machines or on bastion hosts behind firewalls and strong access control
- Private key reconstruction (crypto analysis)
 - ◆ random number not random
 - ◆ Leakage of key material (DSA)
 - ◆ Brute force attacks

Private Key Compromise

- Try to minimize impact
 - ◆ Short validity of signatures
 - ◆ Regular key-rollover
- Remember: KEYS do not have timestamps in them -- the SIG over the KEY has the timestamp
- Key exchange involves 2nd party:
 - ◆ State to be maintained during rollover
 - ◆ operationally more expensive

Short Signature Life Time

- Short parent signature over DS RR protects child
- Order 1 day possible

www.ripe.net. 3600 IN **SIG** A 1 3 3600 20010504144523 (
20010404144523 3112 ripe.net.
VJ+8ijXvbrTLeoAiEk/qMrdudRnYZM1VlqhN
vhYuAcYKe2X/jqYfMfjfSUrmhPo+0/GOZjW
66DJubZPmNSYXw==)

Signature expiration



Key Rollover (part 1)

- Scheduled rollover of the child's Key Signing Key
- Child replaces key-1 with key-2 and wants parent to sign it

```
$ORIGIN net.  
  
kids NS    ns1.kids  
      DS   (...) 1  
      SIG KEY (...) net.
```

old parent zone

```
$ORIGIN kids.net.  
@ NS    ns1  
  KEY (...)      (1)  
  KEY (...)      (5)  
  SIG KEY (...) kids.net. 1  
  SIG KEY (...) kids.net. 5  
ns1     A      127.0.10.3  
SIG A (...) kids.net. 5
```

old child zone

```
$ORIGIN kids.net.  
@ NS    ns1  
  KEY (...)      (1)  
a)  KEY (...)      (2)  
  KEY (...)      (5)  
  SIG KEY (...) kids.net. 1  
b)  SIG KEY (...) kids.net. 2  
  SIG KEY (...) kids.net. 5  
ns1     A      127.0.10.3  
SIG A (...) kids.net. 5
```

a) Create key 2

b) Sign key-set with key 1 and 2
and send key 2 to parent



Key Rollover (part 2)

- c) Parent generates and signs DS record
- d) Child signs his zone with only key 2, once parent updated his zone

```
$ORIGIN net.  
  
kids NS    ns1.kids  
      DS   (...) 2  
      SIG KEY (...) net.
```

```
$ORIGIN kids.net.  
  
@ NS    ns1  
  KEY   (...) 2  
  KEY   (...) 5  
  SIG KEY (...) kids.net. 2  
  SIG KEY (...) kids.net. 5  
ns1    A  127.0.10.3  
      SIG A (...) kids.net. 5
```



Timing of the Scheduled Key Rollover

- Child should not remove the old key while there are still servers handing out the old DS RR.
- The new DS will need to be distributed to the slave servers
 - ◆ max time set by the SOA expiration time
- The old DS will need to have expired from caching servers.
 - ◆ Set by the TTL of the original DS RR.
- You (or your tool) can check for the master and slave to have picked up the change.

Scheduled Key Rollover Issues

- Currently one can not distinguish between a key signing key and a zone signing key.
- Once that distinction can be made, the rollover can be fully automated.

Unscheduled Rollover Problems

- Needs out of band communication with the parent and to pre-configured resolvers
- The parent needs to establish your identity out of band again
- Your children need protection. How to protect them best? Leaving them unsecured?
- There will be a period that the stolen key can be used to generate data useful on the Internet
- There is no 'revoke key' mechanism
- Emergency procedure must be on the shelf

Key Rollover: Questions?





Extra: NXT RR and Wildcard Issues



Not just one NXT RR in your response

- If you query for data does not exist in a zone, the NXT RR provides proof of non-existence
- The principle is simple, as explained before but there is a complication: WILDCARDS.
- Wildcards are needed and will need to be secured:
 - *.1.3.e164.arpa. NAPTR (redirection to some ldap server)

Recap Wildcards

- *.ripe.net will provide an answer for:
 - ◆ Any label that is in the ripe.net zone
 - ◆ For labels not already known to exist between the query name and the wildcard domain
 - ✦ If B.X and *.X appear in the zone with origin X then a query for Z.X would return the wildcard data. The wildcard answer would not apply for question for B.X, A.B.X or X.
 - ◆ The wildcard label sorts canonically before the alphabetical data
 - X
 - *.X
 - B.X
 - A.B.X

Proving Non-existence of a wildcard (1)

- Suppose our zone looks like

f.	SOA	...
e.f	A	...
d.e.f	A	...
c.d.e.f	A	...
b.c.d.e.f	A	...
- We query for a.b.c.d.e.f.
- We will have to prove the non-existence of the possible wildcards
- How would a zone with wildcards look?

Proving Non-existence of a wildcard (2)

We have to prove that all these wildcards are NOT in the zone

f.	SOA	...
<i>*.f</i>	<i>A</i>	
e.f	A	...
<i>*.e.f.</i>	<i>A</i>	...
d.e.f	A	...
<i>*.d.e.f.</i>	<i>A</i>	...
c.d.e.f	A	...
<i>*.c.d.e.f.</i>	<i>A</i>	
b.c.d.e.f	A	...
<i>*.b.c.d.e.f</i>	<i>A</i>	

In this case you will only have to return
b.c.d.e.f NXT a
Since it already proves that there is no
closer match for the wildcard.
b.c.d.e.f cancels *.f, *.e.f, *.d.e.f , etc.

(no *.f)



Proving Non-existence of a wildcard (3)

- 5 minute exercise; increase your knowledge of responses.

Zone:

```
$ORIGIN foo.  
foo.      SOA ...  
a.foo.    A 10.0.0.1  
b.foo.    A 10.0.0.2  
e.a.c.foo. A 10.0.0.3
```

Query: f.a.c.foo.

Which NXT RRs may be expected in the answer.

NXT / wildcards: Questions?





DNSSEC - Conclusions



What Did We Learn

- DNSSEC provides a mechanism to protect DNS
- DNSSEC implementation:
 - ◆ TSIG for servers
 - ◆ SIG, KEY and NXT for data
- DNSSEC main difficulties:
 - ◆ keeping private key safe
 - ◆ distributing keys



Open Issues

(the where-shall-I-put-it slide)

DNSSEC is still a moving target...

- RFC 2535 rewrite
- NXT/OPT-IN
- Delegation Signer (DS)
- BIND development
 - ◆ Current bind snapshots have bugs in DNSSEC.
- Operational issues
 - ◆ Webfarms and keymanagement
 - ◆ NXT RR walk and privacy
- API resolver<->cache



Additional Resources

- NLnet labs maintains a list of DNSSEC resources
<http://www.nlnetlabs.nl/dnssec/>
- <http://www.dnssec.net/> is a good portal
- <http://www.ripe.net/disi/>
- dnssec-request@cafax.se

End of Part I... Questions???



Questions later:

■ <training@ripe.net>

<okolkman@ripe.net>



PART II

DNSSEC Operations Description of tools





DNSSEC Operations

- Configuration & Installation
- Securing host-host communication (TSIG)
- Securing zones
- Building a secure tree
- Miscellaneous



Configuration & installation

- TSIG requires that the time between servers is in sync (time zone!)
 - ◆ `ntpdate -b`
 - ◆ `xntpd`
- Openssl libraries required
 - ◆ <http://www.openssl.org/>
- For now, get the latest version (snapshot)
 - ◆ <ftp://ftp.isc.org/isc/bind9/snapshots/>
- Compile the source using openssl libraries:
`./configure --with-openssl`



Server/Named configuration

- The configuration file is called “named.conf”
- Documentation in <src>/doc/arm/Bv9ARM.html
- Turn on logging
 - ◆ Several categories
 - ◆ Categories are processed in one or more channels
 - ◆ Channels specify where the output goes

Logging Categories

Relevant to DNSSEC

- dnssec
 - ◆ Processing DNSSEC signed responses
- security
 - ◆ Request that are approved or not
- notify
 - ◆ Zone change notification (relevant for dynamic update environments)
- update
 - ◆ Dynamic update events



Toolbag

The complete set

- NAMED
- DNSSEC tools:
 - ◆ dnssec-keygen Generate keys of various types
 - ◆ dnssec-signzone Sign a zone
 - ◆ dig +dnssec Use dig to troubleshoot
(or host, nslookup not supported)
 - named-checkzone & named-checkconf



Toolbag: dig For trouble shooting

- **dig** is your friend, learn to use it!
 - ◆ nslookup will not be supported, host will
 - ◆ dig is low level *i.e.* not user friendly
- All pieces of information are relevant
 - ◆ Status, flags, answer section, authority section, additional section



Dig example

dig bert.secret-wg.org

```
; <<>> DiG 9.3.0s20020122 <<>> bert.secret-wg.org
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40334
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2,
    ADDITIONAL: 2

;; QUESTION SECTION:
bert.secret-wg.org.                IN      A

;; ANSWER SECTION:
bert.secret-wg.org.  36293  IN      A      193.0.0.4

;; AUTHORITY SECTION:
secret-wg.org.       170827 IN      NS      bert.secret-wg.org.
secret-wg.org.       170827 IN      NS      NS2.secret-wg.org.

;; ADDITIONAL SECTION:
bert.secret-wg.org.  36293  IN      A      193.0.0.4
NS2.secret-wg.org.  170827 IN      A      193.0.0.202

;; Query time: 89 msec
;; SERVER: 193.0.1.96#53(193.0.1.96)
;; WHEN: Wed Feb 13 11:08:36 2002
;; MSG SIZE  rcvd: 116
```

Securing host-host communication



TSIG configuration Outline

- Generate a key
- Configuring secure transfers between servers with TSIG
- Testing
- Other types of host-host communication



TSIG Toolbag: dnssec-keygen

■ Use dnssec-keygen to Generate TSIG keys

Usage:

```
dnssec-keygen -a alg -b bits -n type [options] name
```

- Use HMAC-MD5 as algorithm
- type is host
- Bitsize: 256 or larger
- Name: unique identifier
 - ◆ Suggested: **host-host.domain.foo.**
 - ◆ We use: **me-friend** because of formatting constraints

TSIG Toolbag: dnssec-keygen output

```
dnssec-keygen -a HMAC-MD5 -b 256 -n host me-friend
```

algorithm

keytag

- Kme-friend.+157+51197.private
- Kme-friend.+157+51197.key
- Private and Public Key content both the same
- TSIG should never be put in zone files!!!
(This might be confusing because of the content of
Kme-friend+157+51197.key)

```
me-friend. IN KEY 512 3 157 nEfRX9...bbPn7lyQtE=
```



TSIG configuration steps 1-3

1. Create key using DNSSEC-keygen:

```
dnssec-keygen -a HMAC-MD5 -b 256 -n HOST me-friend
```

```
Kme-friend.+157+51197
```

2. Cut-n-paste key material into named.conf

```
key "me-friend." {  
    algorithm hmac-md5;  
    secret  
        "nEfRX9jxOmzsby8VKRgDWEJorhyNbjt1ebbPn71yQtE=" ;  
};
```

3. Communicate this with your partner (off band, PGP...)

TSIG configuration step 4

4. Configure your server to require the key for zone transfers

- ◆ Use the key statement to configure the key
- ◆ Use the allow-transfer statement in the zone statement to indicate which keys are allowed transfer

```
zone "ripe.net" {  
    type master;  
    file "zones/ripe.net."  
    allow-transfer { key me-friend ; }  
    notify yes;  
};
```

TSIG configuration step 5

5. Have your partners configure their servers to use the key when talking to you
- ◆ Use the **key** statement to configure the key
 - ◆ Use the **server** statement to indicate which key is needed for communication with that server

```
server 192.168.10.1 {  
    keys {me-friend; };  
};  
zone "ripe.net" {  
    type slave;  
    masters { 192.168.10.1;};  
    file "slaves/ripe.net";  
};
```




TSIG Troubleshooting: dig

- You can use dig to check TSIG configuration
- `dig @<server> <zone> AXFR -y name:key`

```
$ dig @193.0.0.202 ripe.net AXFR \  
-y me-friend:nE1Gw6fpW...tE=
```

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

```
security: error: client 193.0.0.182#1228: zone transfer 'ripe.net/IN' denied
```



Using TSIG to protect dynamic updates

- You can use TSIG or SIG0 to protect your dynamic updates
 - ◆ Detailed howto at: Secure dynamic update HOWTO on ops.ietf.org
- Steps for TSIG dynamic update of forward tree:
 - ◆ Configure your TSIG key into `/etc/dhclient.conf` and specify the FQDN
 - ◆ Configure `named.conf` to allow updates using the key

Securing zones



Setting up a secure zone Outline

*We now focus on setting up a locally secured zone.
e.g. for use in a corporate environment*

- *Resolver issues* ←
- *Generating key*
- *Signing the zone*

Resolving in a secured DNS environment

A few remarks

- DNSSEC is not in POSIX yet (e.g. `gethostbyname()` or `getnameinfo()`)
- SIG verification is (only) done by caching forwarders
- To test DNSSEC setups, you have to work with `dig`, or use the BIND `lwresolver` library
- Alternatively: write some tools in PERL (`Net::DNS` and `Net::DNS::SEC`)



Setting up a verifying resolving name server

- You want to verify the content of a zone:
 - ◆ Get the public (key signing) key and check, out of band, that this key belongs to the zone owner.
 - ◆ Configure the key in your resolving nameservers.
- We will configure the key of secret-wg.org in our verifying resolving name server



Configuring verifying resolving name servers

- In the forwarder you configure the keys you trust as a secure entry point

```
trusted-keys {  
    "." 256 3 1 "Abc12...zZ";  
    "SECRET-WG.ORG" 256 3 1 "AQ...QQ=="  
};
```

- Sys-admins of resolving name servers should verify authenticity of **trusted-keys** before putting them in zone files



Testing a verifying forwarder using dig

dig +dnssec [@server] record [TYPE]

- Answer Flags are relevant
- Example query to a authoritative nameserver

```
; <<>> DiG 9.1.1 <<>> +dnssec @193.0.0.202
    www.ripencc.dnssec.nl.nl
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1947
;; flags: qr aa rd; QUERY: 1, ANSWER: 4, AUTHORITY: 3,
    ADDITIONAL: 4
```

Recursion desired (but not available, RA is not set)

authoritative answer



Testing a verifying forwarder dig: an example

```
; <<>> DiG 9.3.0s20020122 <<>> +dnssec @127.0.0.1 secret-wg.org
NS
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31630
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 3, AUTHORITY: 0,
    ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version:    0, udp=    4096
;; QUESTION SECTION:
;secret-wg.org.      IN NS

;; ANSWER SECTION:
secret-wg.org.      600 IN NS ns2.secret-wg.org.
secret-wg.org.      600 IN NS bert.secret-wg.org.
secret-wg.org.      600 IN SIG NS 1 2 600 20020314134313
                        20020212134313 47783 secret-wg.org.
DVC/ACejHtZy1ifpS6VSSqLa15xPH6p33HHmr3hC7eE6/QodM6fBi5z3
fsLhbQuuJ3pCEdi2bu+A0duuQlQMihPvrkYia4bKmoyyvWHwB3jcyFhW
1V4YOzX/fgkLUmu8ysGOiD9C0CkSvNSE6rBCzUa3hfkksHt4FBsuA1oQ
yoc=
```



Troubleshooting client side

- Dig returns status: SERVFAIL
- First try without **+dnssec**
- Also try with **+dnssec +cdflag**
 - ◆ Checking is disabled. Data directly forwarded
- Be ready for some interesting troubleshooting



Troubleshooting Server side

- Turn on logging. Category “dnssec” with severity debug 3 gives you appropriate hints
- Debug output is a little detailed
 - ◆ On the next page is an example where we corrupted the trusted-key
 - ◆ It is not directly obvious what the problem is
 - ◆ We edited the output a little so that it fits on a slide



Example debugging output

```
validating secret-wg.org KEY: starting
validating secret-wg.org KEY: attempting positive response validation
validating secret-wg.org KEY: keyset was self-signed but not
                                preconfigured
validating secret-wg.org KEY: no valid signature found
validating secret-wg.org KEY: falling back to insecurity proof
validating secret-wg.org KEY: insecurity proof failed
validator @0x81e6900: dns_validator_destroy
validating secret-wg.org NS: in fetch_callback_validator
validating secret-wg.org NS: fetch_callback_validator: got no valid SIG
validator @0x81e1d00: dns_validator_destroy
```

Setting up a secure zone Outline

- *Resolver issues*
- *Generating key*
- *Signing the zone*





Toolbag: dnssec-keygen

- **Use dnssec-keygen to Generate zone keys**

Usage:

```
dnssec-keygen -a alg -b bits -n type [options] name
```

- Use RSA or DSA as algorithm
- type is zone
- Bitsize: depends...
- Name: the name of the zone you want to sign

Toolbag: dnssec-signzone

Usage:

`dnssec-signzone [options] -k [keysignkeys] zonefile [keys]`

- If the name of your zonefile is not the name of the zone then use the `—o <origin>` option
- You might need the `'-r /dev/urandom'` option on your OS
- Choose which key to use as zone signing and which key to use as key signing key.
 - ◆ There is no distinction in filename or RR content to distinguish between the two types of keys yet
- Your keyset is extracted as a bonus... ready to go to parent



Signing a Zone 1 Creating the KEY

- `dnssec-keygen -a RSASHA1 -b 1024 -n zone secret-wg.org`

`Ksecret-wg.org.+005+20704`

- `Ksecret-wg.org.+005+20704.key` contains the public key.
- `Ksecret-wg.org.+005+20704.private` should be kept secret

Creating the key 2

- You will need to generate multiple keys and designate their use as zone or key signing keys.
- The zone signing key does not need to be as long the key signing key.
- The zone signing key can be rolled-over without interaction with parent or children.
- The key signing key will be configured in resolving name servers (and maybe later uploaded to your parent)
- There is no way (yet) to distinguish between the two.

Setting up a secure zone Outline

- *Resolver issues*
- *Generating key*
- *Signing the zone*



Signing a Zone 2

Signing the Zone

- include the keys for the apex into your zone:
 - ◆ `cat Ksecret-wg.org.+001+20704.key >> secret-wg.org`
 - ◆ `cat Ksecret-wg.org.+001+16748.key >> secret-wg.org`
 - ◆ `cat Ksecret-wg.org.+001+86491.key >> secret-wg.org`
- Increase the SOA serial number
 - ◆ **Always increase the SOA serial before signing!**
- Sign the zone:

```
dnssec-signzone -k Ksecret-wg.org.+001+20704 \
secret-wg.org Ksecret-wg.org.+001+16748 \
Ksecret-wg.org.+001+86491
```

Key signing keys

Zone signing keys



Signing a Zone 2 Publishing Zone

- Publish the zone:

```
zone "secret-wg.org" {  
    type master;  
    file "zones/secret-wg.org.signed";  
    allow-transfer { 10.1.2.3 ;  
                    key mstr-slave.secret-wg.org.; };  
    notify yes;    // Don't forget to increase serial  
    before signing  
};
```

- RNDNC reload and test.

Notes on secured zones

- Only those records for which the server is authoritative for are signed
 - ◆ NS records in the APEX are signed
 - ◆ Delegating NS records and GLUE are not signed
 - ◆ DS RRs are set

Building a secure tree

Task 3

Parent-Child interaction

- With secured islands there is a problem with key distribution
- Use the DNS itself to distribute keys; once authenticity is established for one key you can use that key to establish authenticity of other keys
- In an ideal world:
 - ◆ You would only configure one key (the root key)
 - ◆ Delegate trust from parent to child

DS RRs for delegation.

- Parent is authoritative for the DS record
 - ◆ It may not appear in the child's apex
- Simplifies KEY exchange
- Eases resigning
 - ◆ parent can sign often ➡ short signature lifetime ➡ shorter impact time when key gets compromised



Delegation of authority

- DS are used for delegation of security
- DS only implemented in bind9.3s20020722 or later
- DS will be backwards incompatible with 2535, ie not work with bind9.1, bind9.2 etc



DS and key exchanges outline

- *Building a secure tree*
 - ◆ *Initial key exchange*
 - ◆ *Regular rollover*

Key exchange 1

Initial Key exchange

- The procedure is the same as signing a 'locally secured zone'
- Before starting publish the key signing key in your zone
- Child will need to upload a key signing key to the parent. The procedure is dependent of the parent policy.
- Procedure will usually have an off-band verification stage.



Key exchange 2

Initial Key exchange parent's considerations

- After the key has been uploaded by the child you have to do an out-of band verification
 - ◆ Check if the key is in the zone.
 - ◆ Check the self-signature; zone owner has private key.
 - ◆ Check if the zone owner initiated the key-exchange.
 - ✦ Phone call, fax, pre-exchanged PGP/CERTs etc
 - ✦ Mail to contact information
 - ✦ Mail to SOA address
 - ◆ Generate DS RRs from the key RRs
 - ✦ Done automatically by `dnssec-signzone` if the key is stored in a file called `keyset-<child domain name>`

Key exchange 3

Parental considerations 2

- You may want to design a database that maintains the trust information and implements your policy.
- A key revocation procedure will probably need the same authentication (or stronger) than the

Key exchange 4

The parental hand work

```
cd /export/zonedb/           (contains zone file and private keys)
cat Ksub.tld.+5+12345.key > keyset-sub.tld.
dnssec-signzone -k Ktld.+5+3215 tld
Ktld.+5+9823 (3215 is key signing key, 9823 is zone signing key)
```

- The signer will automatically generate the DS RRs.
- Alternatively you can use a tool to generate the DS RRs from your key database.

Regular Rollover

- Child generates new zone signing key and signs with two keys.
- Query for the parental DS and remember the TTL you will need it later
- `dnssec-signzone -k Ksub.tld.+5+12345.key -k Ksub.tld.+5+67890.key`
- Upload the new key to the parent. The parent will generate a new DS RR.
- Check if all parental servers (slaves and masters) have picked up the change, wait another TTL before you remove the old key.

Miscellaneous (and conclusions)

Key exchange and Key rollover

- Upload your key to parent (first key exchange)
 - ◆ procedure is registry dependent
- Key rollover Task
 - ◆ Generate a new key
 - ◆ Publish new key in your zone file and sign with old and new key
 - ◆ Don't forget to inform those resolvers that need you as a secure island (**trusted-keys** configuration)
 - ◆ Trigger the registry (push or pull)
 - ◆ Check availability of SIG over new DS record at parent
 - ◆ Remove old key



Back at the ranch

- Design a secure architecture
- Design a key exchange procedure
- Resign your zone regularly
- Automate the process (cron and Makefiles)
- Have an emergency procedure in place



Net::DNS::SEC Shameless plug

- Net::DNS security extensions at www.ripe.net/disi and on CPAN
- May be useful while troubleshooting or building tools
- Introduces Net::DNS::RR::SIG, Net::DNS::RR::KEY, Net::DNS::RR::NXT and Net::DNS::RR::DS

Example creating and verifying signature:

```
$keypath="/home/olaf/Kbla.foo.+001+60114.private";  
$sigrr= create Net::DNS::RR::SIG(\@datarrset, $keypath);  
$sigrr->verify(\@dataset,$keyrr)||croak $sigrr->vrfyerrstr;
```

Feedback

- Give us feedback on DNSSEC operations
 - ◆ Which tools would make your life easier
 - ◆ Why are you (un)successful with deployment
 - ◆ What kind of information would ease your tasks

- Slides and other DNSSEC material at:
www.ripe.net/training/dnssec/

- Feedback on this tutorial.
 - ◆ Suggestions can always be mailed to training@ripe.net.

Questions

