Welcome to the

DNSSEC

RIPE NCC Training Course

Why DNSSEC?

- DNS is not secure
  - Known vulnerabilities
  - People depend more and more on DNS
- DNSSEC protects against data spoofing and corruption
- Why this course:
  - To raise awareness on DNSSEC
  - To provide handles for deployment
    - Reverse delegation

Why RIPE NCC?

- Maintaining in-addr.arpa for several /8 blocks
- Involved in other DNS issues:
  - K-root name server
  - ENUM
- Interested in Internet-wide security technologies
  - **http://www.ripe.net/disi/**

Who are we?

- Trainers:
  - Know about DNSSEC
  - Not DNS server operators!

- Audience:
  - Know about DNS and want to know about DNSSEC

- please fill in the “expectations” in the questionnaire

Course Outline

- Introduction
- DNSSEC mechanisms
  - to authenticate communication between hosts
    - TSIG / SIG0
  - to establish authenticity and integrity of data
    - New RRs
    - Signing a single zone
    - Building chains of trust
    - Key exchange and key rollovers
- Operational concerns
- Conclusions

DNS: Known Concepts

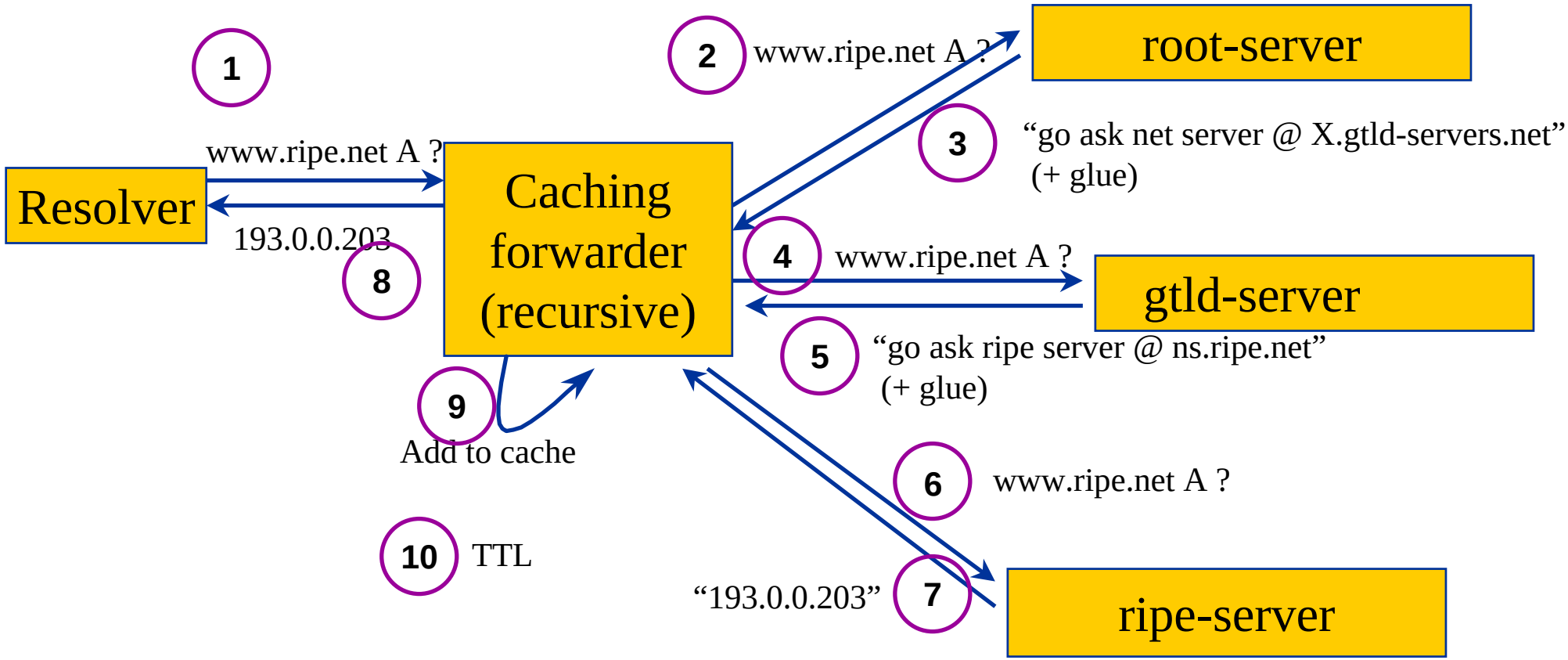
- Known DNS concepts:
  - Delegation, Referral, Zone, RRs, label, RDATA, authoritative server, caching forwarder, stub and full resolver, SOA parameters, etc
  - Don’t know? Do ask!

- Operational knowledge with BIND
  - BIND 8 or 9 named.conf, writing zone files
  - All examples based on IPv4

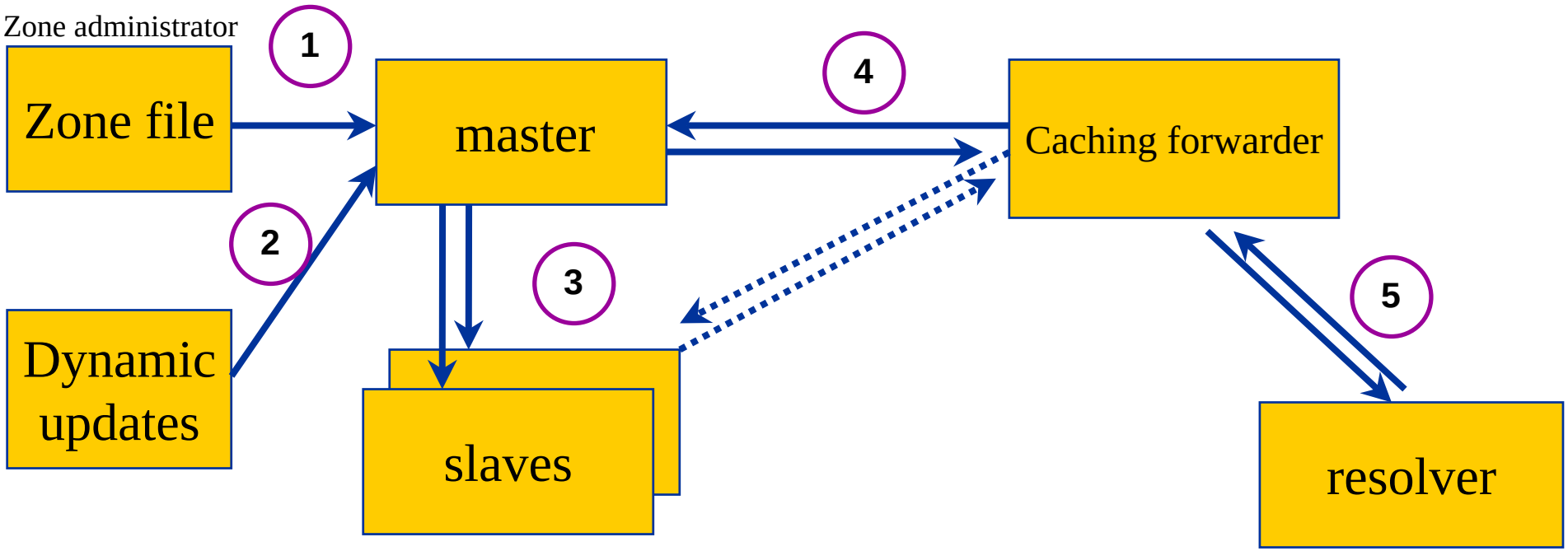
Reminder: DNS Resolving

Question:

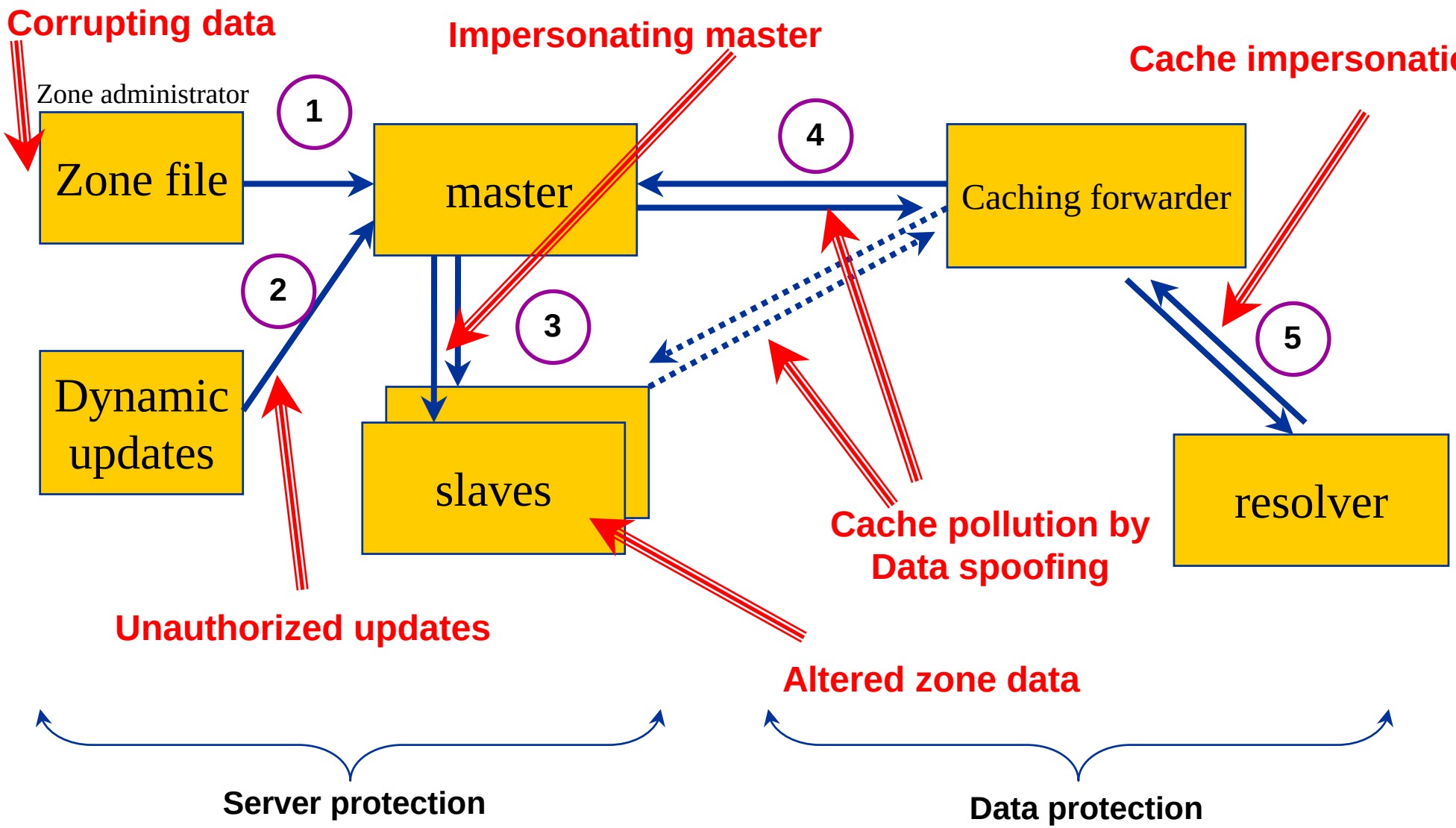
www.ripe.net A



DNS: Data Flow



DNS Vulnerabilities



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)?

DNSSEC protects..

DNSSEC protects against data spoofing and corruption

- TSIG/SIG0: 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

DNSSEC Current State

- RFC 4033
  - DNS Security Introduction and Requirements
- RFC 4034
  - Resource Records for the DNS Security Extensions
- RFC 4035
  - Protocol Modifications for the DNS Security Extensions

- March 2005
- Obsoletes RFC 2535

Configuration & Installation BIND's named

- BIND 9.3 or later supports current DNSSEC
  - **ftp://ftp.isc.org/isc/bind9/**
- TSIG requires servers to sync time (time zone!)
  - ntpdate -b
  - xntpd
- Openssl libraries required for crypto parts
  - http://www.openssl.org/
- Compile the source using openssl libraries:
  ./configure --with-openssl

Bind DNSSEC Tools

- Named
- dnssec-keygen
  - Generate keys of various types
- dnssec-signzone
  - Sign a zone
- dig
  - Troubleshoot: Usage: dig +dnssec @…
- named-checkzone & named-checkconf
  - syntax check for zonefiles and named.conf

Server/Named Configuration

- The configuration file is called “named.conf”
- Documentation in <src>/doc/arm/Bv9ARM.html

- Turn on DNSSEC in “options” statement
  - dnssec-enable yes;

- Turn on logging for troubleshooting
  - Several categories
  - Categories are processed in one or more channels
  - Channels specify where the output goes

Relevant Logging Categories

- 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

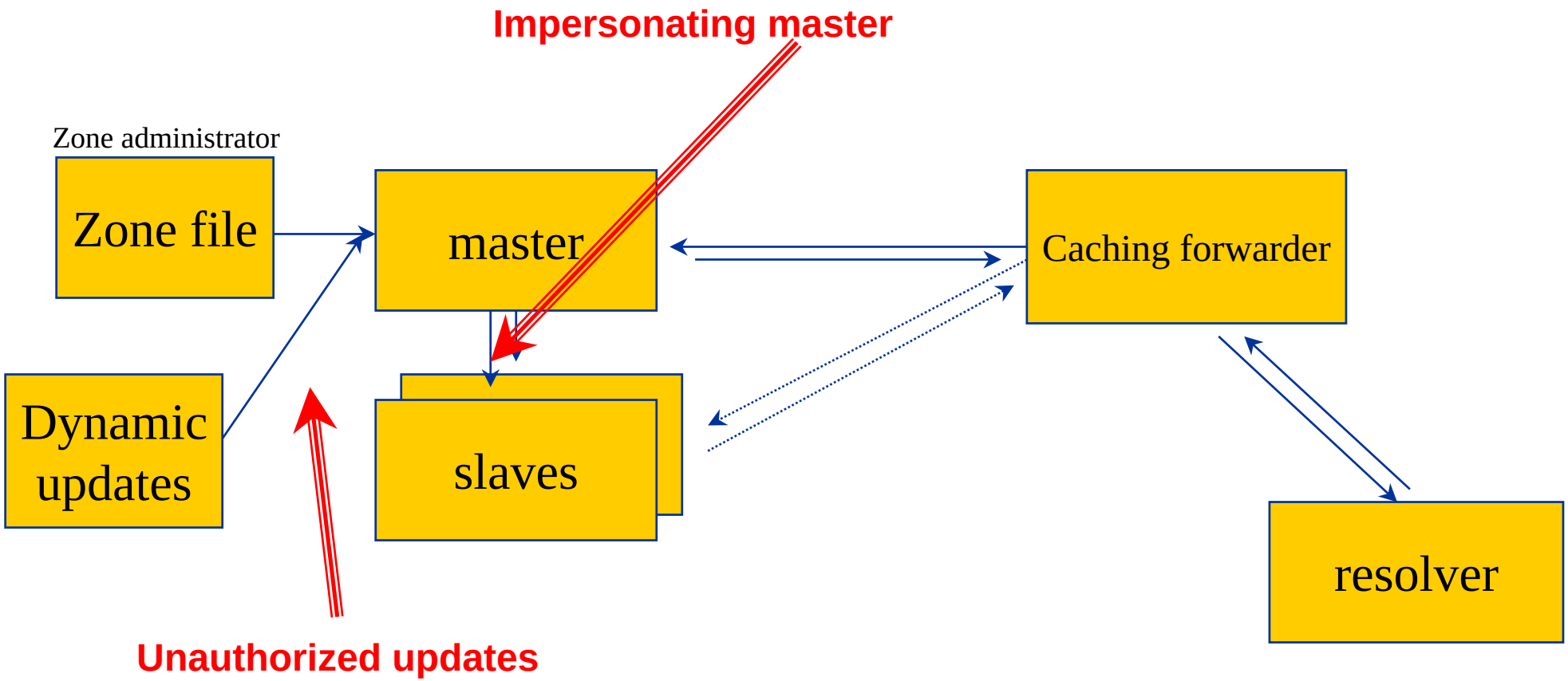
Logging Configuration Example

```
logging {
        channel query_channel {
                file "log/querylog" versions 3;
                print-time yes;
        };
        channel dnssec_log {
                file "log/dnssec" versions 2;
                print-time yes;       // add timestamp the entries
                print-category yes;   // add category name
                print-severity yes;   // add severity level
                severity debug 3;     // print debug messages
        };
        channel everything_else {
                file "log/runlog" versions 3;
                print-time yes;
                print-severity yes;
                print-category yes;
        };
        category dnssec   { dnssec_log; };
        category security { dnssec_log; everything_else; };
        category queries  { query_channel; };
        category default  { everything_else; };
};
```

Questions?

Securing Host-Host Communication

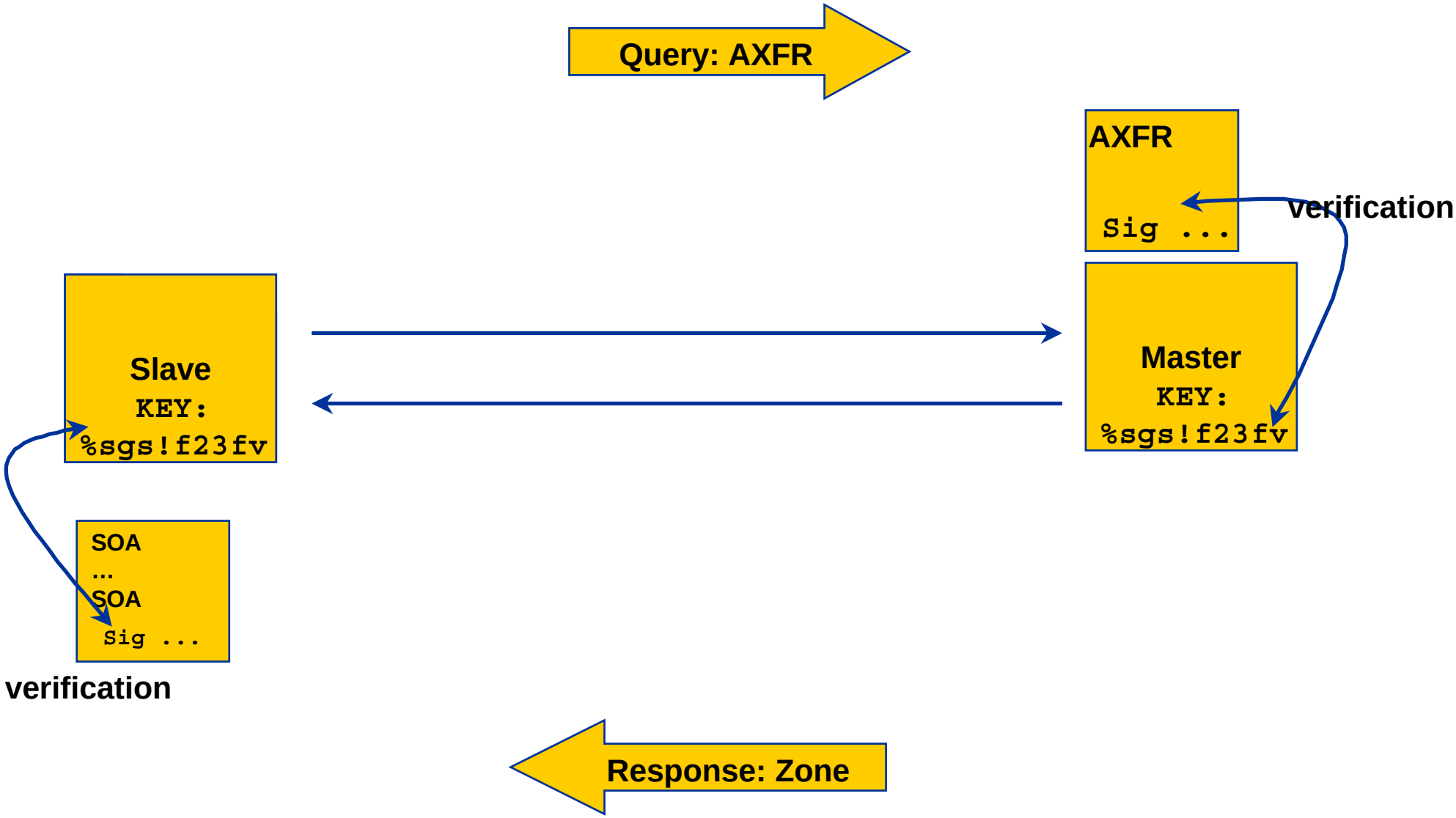
TSIG Protected Vulnerabilities



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
  - DNS question or answer & timestamp
- Traffic signed with “shared secret” key
- Used in configuration, **NOT** in zone file

TSIG example



TSIG for Zone Transfers

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

Generate TSIG Secret

```
dnssec-keygen -a <alg> -b <bits> -n <type> [options] <keyname>
```

- algorithm: HMAC-MD5
- '-r /dev/urandom' might be needed
- Bits: 256 or larger
- type: host
- Name: unique identifier
  - Suggested: **`master-slave.zone.name.`**
  - "**`me-friend.`**" **used as example because it is short**

- TSIG secret can be generated differently (base64 encoding)

TSIG dnssec-keygen Output

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


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

- Private and Public Key contain the same key
- TSIG should never be put in zone files!!!
  - might be confusing because it looks like RR:

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

Master Server: named.conf

- "**Key**" statement to configure key

```
key ”me-friend." {
      algorithm hmac-md5;
      secret “nEfRX9jxOmzsby8VKRgDWEJorhyNbjt1ebbPn7lyQtE=";
   };
```

- "allow-transfer" option in zone statement indicates which keys are allowed transfer
  - Can be combined with IP based restrictions

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

Slave Servers: named.conf

- **"key"** statement to configure the key

```
key ”me-friend." {
        algorithm hmac-md5;
        secret
  “nEfRX9jxOmzsby8VKRgDWEJorhyNbjt1ebbPn7lyQtE=";
};
```

- **"server"** statement to indicate key used
  - Zone configuration doesn’t change on slave server

```
server 192.168.10.1 {
        keys {me-friend.; };
};
```

Testing & Troubleshooting: dig

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

```
$ dig @10.0.53.204 example.net AXFR \
    -k Kme-friend.+157+51197.key
```

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

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

Importance of the Time Stamp

- TSIG/SIG0 signs a complete DNS request / response with time stamp
  - to prevent replay attacks
  - currently hardcoded at 5 minutes
- 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!!!

Authenticating Servers Using SIG0

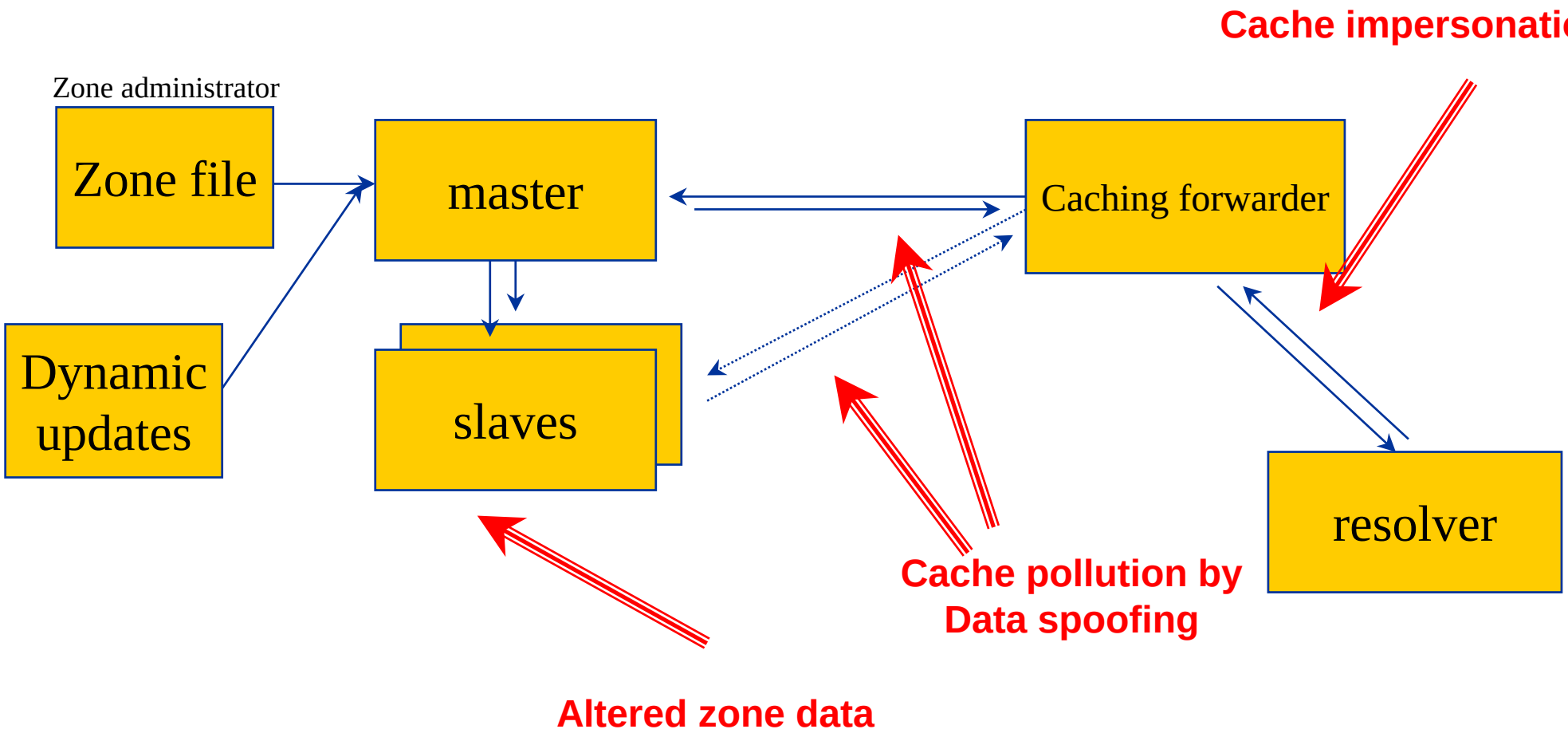
- Alternatively its possible to use SIG0
  - Not widely used yet
  - Works well in dynamic update environment
- Public key algorithm
  - Authentication against a public key published in the DNS
- SIG0 specified in RFC 2931

Questions?

DNSSEC Mechanisms

- New Resource Records
- Setting up a Secure Zone
- Delegating Signing Authority
- Key Rollovers

Vulnerabilities protected by DNSKEY / RRSIG / NSEC



DNSSEC hypersummary

- 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

The DNS is not a Public Key Infrastructure (PKI)

- All key procedures are based on local policy
- A PKI is as strong as its weakest link
  - Certificate Authorities control this by SLAs
- The DNS does not have Certificate Revocation Lists
- If the domain is under one administrative control you might be able to enforce policy

Public Key Crypto

- Key pair: a private (secret) key and a corresponding public key
- Simplified:
  - If you know the public key, you can verify a signature created with the private key
  - If you know the public key, you can encrypt data that can only be decrypted with the private key
- DNSSEC only uses signatures
  - PGP uses both methods

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

Zone status terminology (RFC3090)

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

New Resource Records

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

Other Keys in the DNS

- DNSKEY RR should only be used for DNSSEC
  - keys for other applications should use other RR types
- CERT
  - For X.509 certificates
- Application keys under discussion/development
  - IPSECKEY
  - SSHFP

RR's and RRsets

- Resource Record:
  - name TTL class type rdata

```
www.ripe.net.     7200   IN    A      192.168.10.3
```

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

```
www.ripe.net.     7200   IN    A      192.168.10.3
                                A      10.0.0.3
                                A      172.25.215.2
```

- RRsets are signed, not the individual RRs

DNSKEY RDATA

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

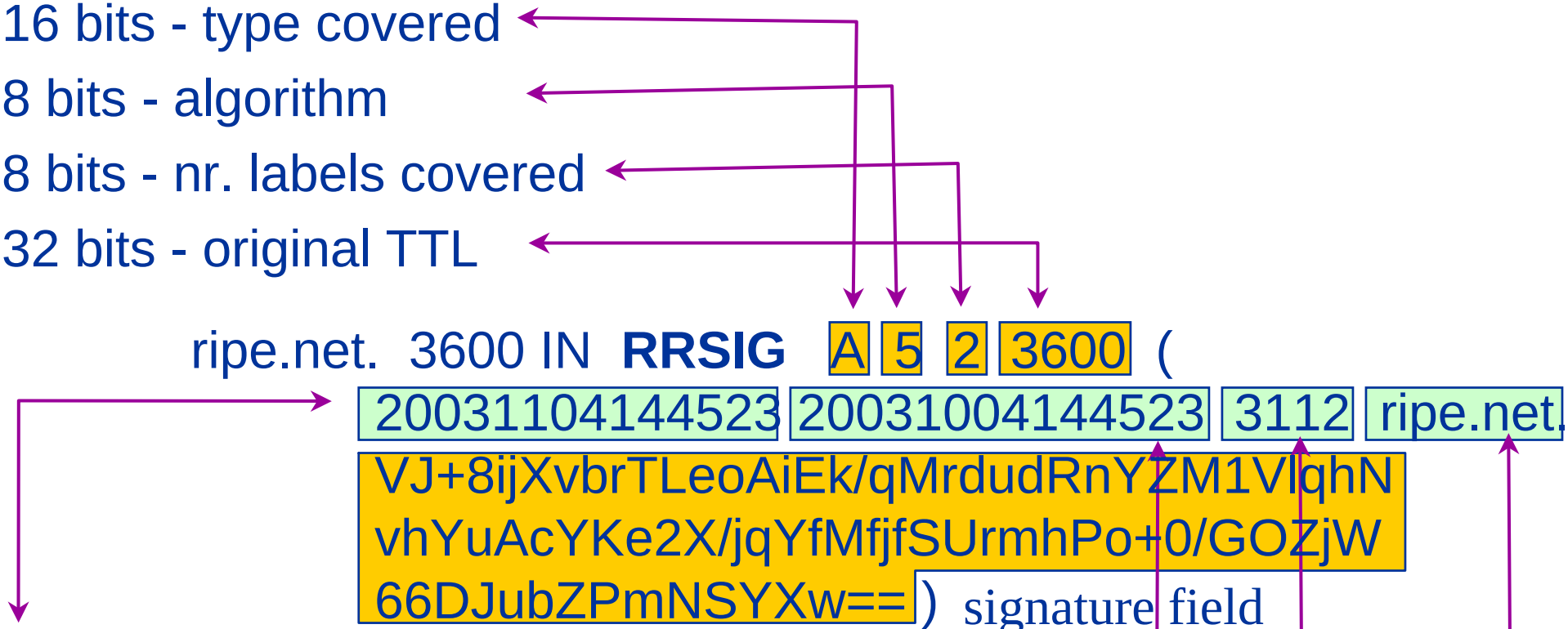
Example:

ripe.net. 3600 IN **DNSKEY** 256 3 5 (

AQOvhvXXU61Pr8sCwELcqqq1g4JJ
CALG4C9EtraBKVd+vGIF/unwigfLOA
O3nHp/cgGrG6gJYe8OWKYNgq3kDChN)

RRSIG RDATA

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



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

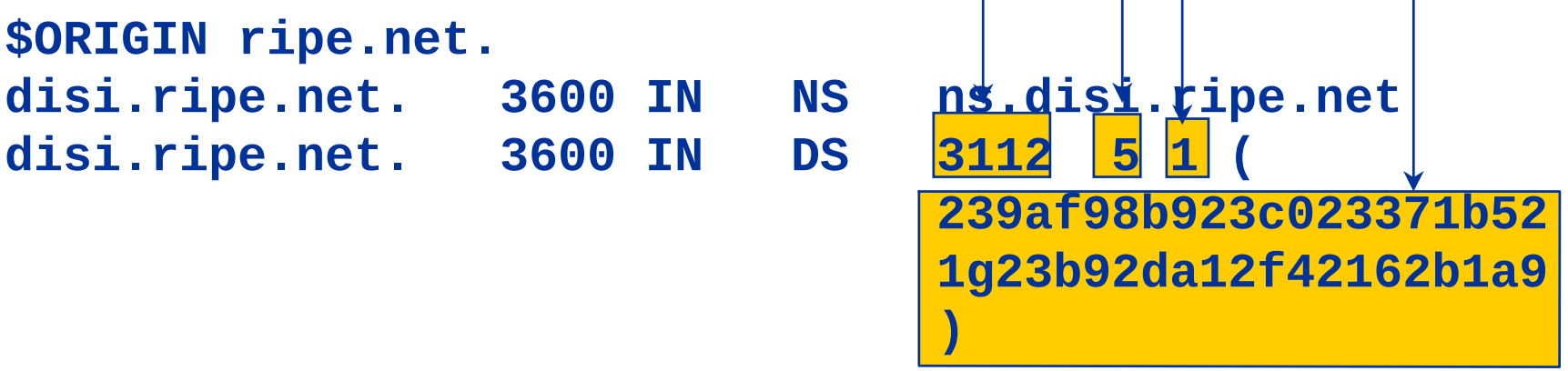
Delegation Signer (DS)

- 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

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



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
- N*32 bit type bit map
- Used for authenticated denial-of-existence of data
  - authenticated non-existence of TYPEs and labels

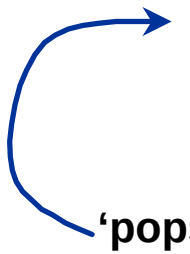
- Example:

```
www.ripe.net. 3600 IN   NSEC ripe.net. A RRSIG NSEC
```

NSEC Record example

(RRSIG records removed for brevity)

```
$ORIGIN ripe.net.
@   SOA       …..
            NS      NS.ripe.net.
            DNSKEY        …..
            NSEC   mailbox.ripe.net. SOA NS NSEC DNSKEY RRSIG
mailbox     A       192.168.10.2
            NSEC    www.ripe.net.  A NSEC RRSIG
WWW         A       192.168.10.3
            TXT     Public RIPE & RIPE NCC webserver
            NSEC     ripe.net. A NSEC RRSIG TXT
```



‘popserver’ is missing

- Query for “popserver.ripe.net” would return:

```
aa bit set   RCODE=NXDOMAIN
authority: mailbox.ripe.net.  NSEC www.ripe.net.  A NSEC RRSIG
```

- Query for “`www.ripe.net MX`” would return: an empty answer section and the “`www NSEC`” record in the authority section

NSEC records

- If your query for data does not exist in a zone, the NSEC RR provides proof of non-existence

- If after a query the response is:
  - NXDOMAIN: One or more NSEC RRs indicate that the name or a wildcard expansion does not exist
  - NOERROR and empty answer section: The NSEC TYPE array proves that the QTYPE did not exist
- More than 1 NSEC may be required in response
  - wildcards
- NSEC records are generated by tools
  - they also lexicographically order the zone

Questions?

Setting up a secure Zone

Securing a Zone

1. Generate keypair
   - include public key (DNSKEY) in zone file

2. Sign your zone; signing will:
   - sort the zone
   - Insert:
     - NSEC records
     - RRSIG records (signature over each RRset)
     - DS records (optional)
   - generate key-set file (can be used later)

Securing a Zone - continued

3. Publish Signed Zone

4. Configure Forwarding Resolver

5. Test

6. Distribute your public key (DNSKEY) to those that need to be able to trust your zone
   - Key-set or DS-set for Parent

Setting up a secure Zone

Generating keys

Toolbag: dnssec-keygen

- **dnssec-keygen to generate keys**

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

- algorithm: RSASHA1 (or RSA or DSA)
- Bitsize: depends on algorithm, key function, paranoia level
- type: zone
- Name: zone you want to sign

- ‘-r /dev/urandom’ might be needed

Creating keys

```
$dnssec-keygen -a RSASHA1 -b 1024 -n zone example.net.
Kexample.net.+005+20704
$
```

- 2 files are created:
  - `Kexample.net.+005+20704.key`
    - contains the public key
    - should go into the zone file
  - `Kexample.net.+005+20704.private`
    - contains the private key
    - **should be kept secret!!!**

Setting up a secure Zone

Signing & publishing

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

Preparing the zonefile

- include the public keys in the zonefile:
  - `cat Kexample.net.+005+20704.key >> example.net`
- Use named-checkzone
- Increase the SOA serial number
  - **Always increase the SOA serial before signing!**

Sign the zone

```
dnssec-signzone [options] zonefile [ZSK's]
```

- If zonefile name is not zone name:
  - use `-o <origin>` option
- Signed zonefile is called "*zonefilename*.signed"
- Keyset is created as a bonus…
  - ready to go to parent
- To create DS records from keyset files:
  - use `-g` option

Publishing the signed zone

- Edit named.conf:

```
zone ”example.net" {
        type master;
        file "zones/example.net.signed";
        allow-transfer { 10.1.2.3 ;
                         key mstr-slave.example.net.; };
        notify yes;
};
```

- Use named-checkconf
- Reload zone
- Test

Setting up a secure Zone

Resolver configuration

Setting up a verifying resolving name server

- To verify the content of a zone:
  - Get the public (key signing) key and check that this key belongs to the zone owner

- Configure the keys you trust as secure entry points in named.conf

```
trusted-keys {
        "example.net." 256 3 1 "AQ…QQ==";
        };
```

Setting up a secure Zone

Testing the secure zone

Testing a verifying forwarder

`dig +dnssec [@server] record [TYPE]`

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

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

authoritative answer
Not authenticated!

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

Testing a verifying forwarder dig: an example

```
; <<>> DiG 9.3.0s20020122 <<>> +dnssec @127.0.0.1 example.net 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:
;example.net.  IN NS

;; ANSWER SECTION:
example.net.          600 IN NS ns1.example.net.
example.net.          600 IN NS ns2.example.net.
example.net.          600 IN SIG NS 1 2 600 20020314134313
                             20020212134313 47783 example.net.
      DVC/ACejHtZylifpS6VSSqLa15xPH6p33HHmr3hC7eE6/QodM6fBi5z3
      fsLhbQuuJ3pCEdi2bu+A0duuQlQMiHPvrkYia4bKmoyyvWHwB3jcyFhW
      lV4YOzX/fgkLUmu8ysGOiD9C0CkSvNSE6rBCzUa3hfkksHt4FBsuA1oQ
      yoc=
```

"use DNSSEC, if you can"

Authenticated Data

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 (partial)

validating sub.tld KEY: in dsvalidated

validating sub.tld KEY: dsset with trust 7

validating sub.tld KEY: verify rdataset: success

validating sub.tld KEY: marking as secure

validator @0x81b53d0: dns_validator_destroy

validating b1.sub.tld A: in fetch_callback_validator

validating b1.sub.tld A: keyset with trust 7

validating b1.sub.tld A: resuming validate

validating b1.sub.tld A: verify rdataset: success

validating b1.sub.tld A: marking as secure

validator @0x81b9e70: dns_validator_destroy

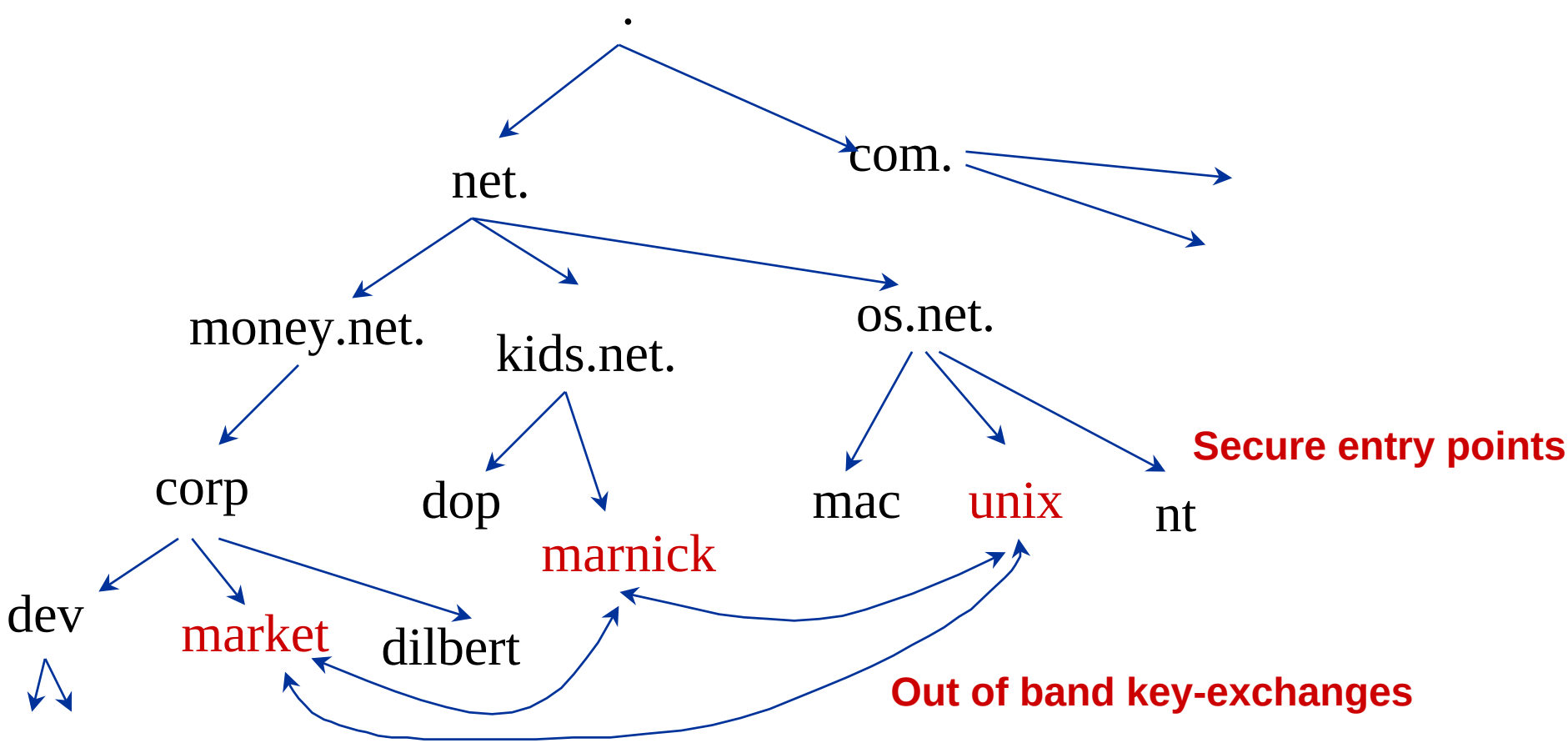
Questions?

Delegating Signing Authority

chains of trust

Locally Secured Zones

- Key distribution does not scale!



Using the DNS to Distribute Keys

- Secured islands make key distribution problematic
- Distributing keys through DNS:
  - Use one trusted key to establish authenticity of other keys
  - Building chains of trust from the root down
  - Parents need to sign the keys of their children
- Ideal world:
  - only the root key needs to be configured
  - parents always delegate security to child

DS RRs for delegation

- Parent is authoritative for the DS record
  - It should not appear in the child's zonefile
- DS resource records are used for Delegation of Security
- DS is not backwards compatible with RFC2535
- Eases resigning
  - parent can sign often => short signature lifetime => shorter impact time when key gets compromised

Key problem

- Interaction with parent administratively expensive
  - Should only be done when needed
  - Bigger keys are better
- Signing zones should be fast
  - Memory restrictions
  - Space and time concerns
  - Smaller keys with short lifetimes are better

Key functions

- Large keys are more secure
  - Can be used longer ☺
  - Large signatures => large zonefiles ☹
  - Signing and verifying computationally expensive ☹
- Small keys are fast
  - Small signatures ☺
  - Signing and verifying less expensive ☺
  - Short lifetime ☹

Key solution: more than one key

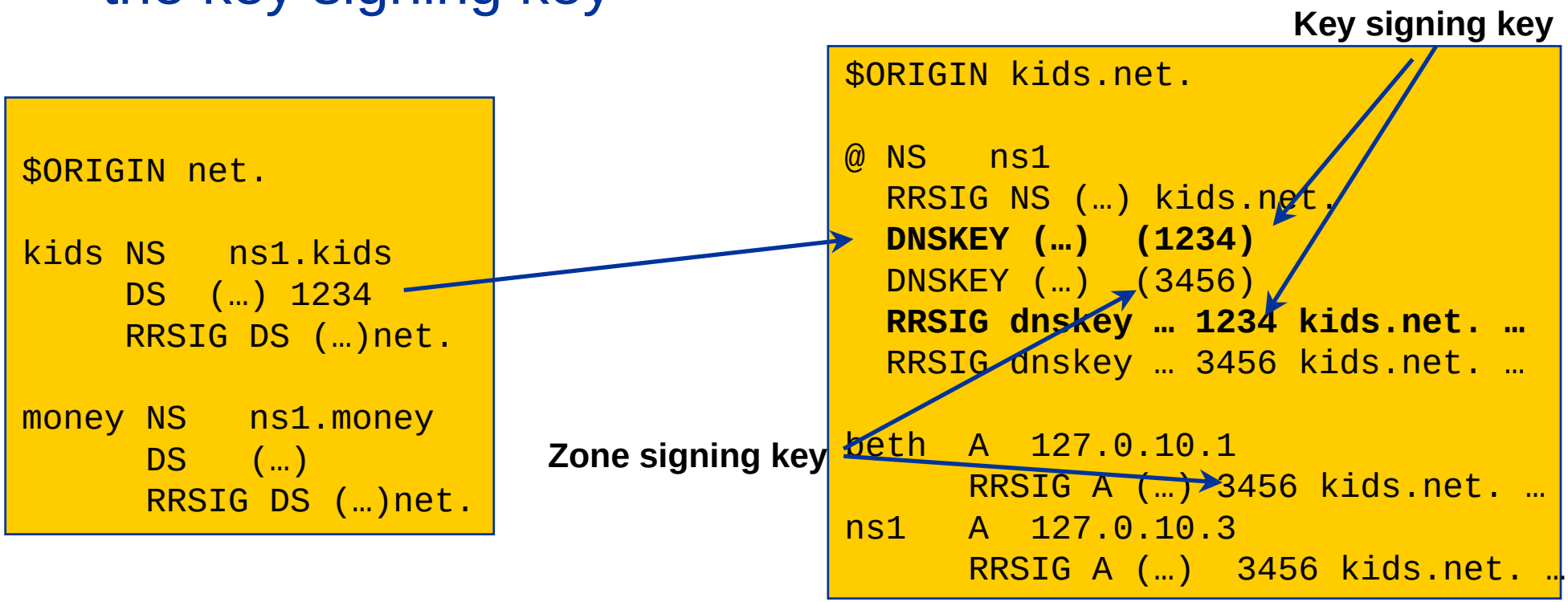
- RRsets are signed, not RR's
- DS points to specific key
  - Signature from that key over DNSKEY RRset transfers trust to all keys in DNSKEY RRset
- Key that DS points to only signs DNSKEY RRset
  - Key Signing Key (KSK)
- Other keys in DNSKEY RRset sign entire zone
  - Zone Signing Key (ZSK)

Initial Key Exchange

- Child needs to:
  - Send key signing keyset to parent
- Parent needs to:
  - Check childs zone
    - for DNSKEY & RRSIGs
  - Verify if key can be trusted
  - Generate DS RR

Delegating Signing Authority

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



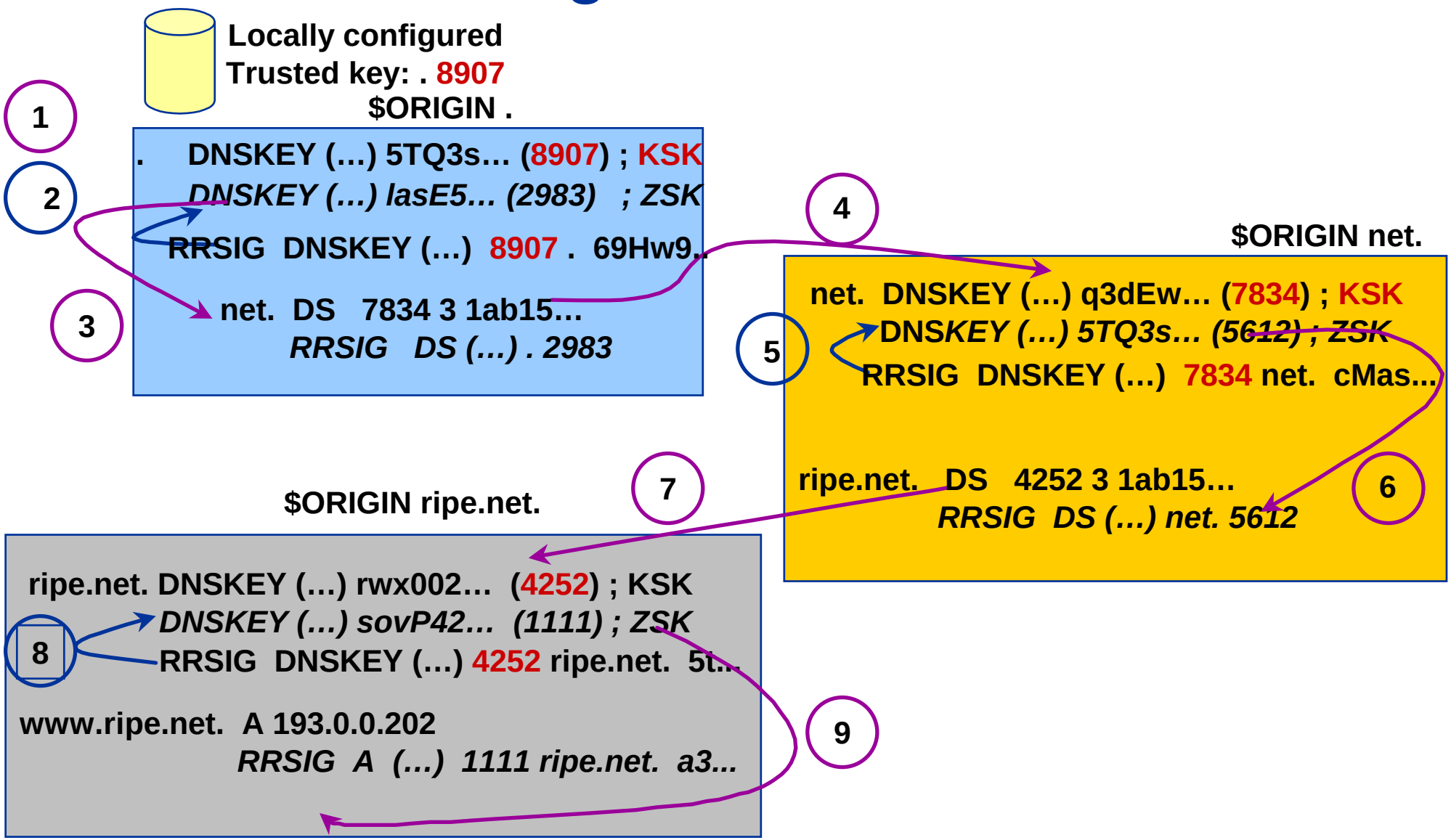
- The parent is authoritative for the DS RR of its children

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

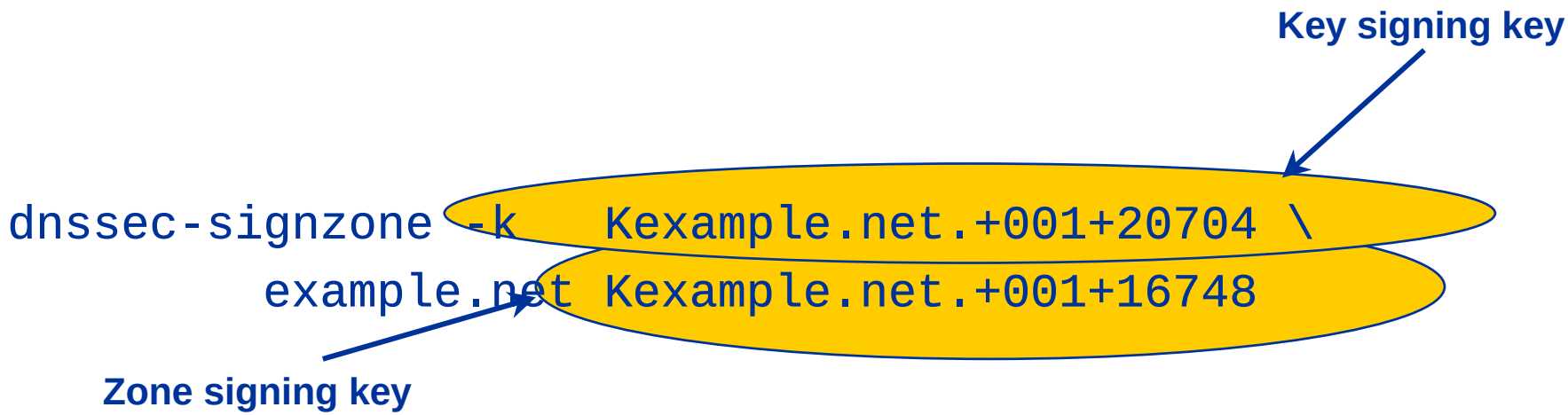
  or
  - DS or DNSKEY records can be trusted if exchanged out-of-band and locally stored (Secure entry point)

Walking the Chain of Trust



Signing the zonefile - with KSK

- Sign the zone:



- Choose ZSK and KSK
- Other options not changed

Questions?

Key Considerations

Private Key Compromise

- 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

Key Rollovers

- Try to minimize impact
  - Short validity of signatures
  - Regular key-rollover
- Remember: DNSKEYs do not have timestamps
  - the RRSIG over the DNSKEY has the timestamp
- Key rollover involves 2nd party or parties:
  - State to be maintained during rollover
  - operationally expensive

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
     RRSIG DS (…)net.
```

parent zone

```
$ORIGIN kids.net.
@ NS   ns1
   DNSKEY (…)  (1)
a) DNSKEY (…)  (2)
   DNSKEY (…)  (5)
   RRSIG KEY (…) kids.net. 1
b) RRSIG KEY (…) kids.net. 2
   RRSIG KEY (…) kids.net. 5
ns1   A  127.0.10.3
   RRSIG 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
     RRSIG DS (…)net.
```

```
$ORIGIN kids.net.

@ NS   ns1
  DNSKEY (…) 2
  DNSKEY (…) 5
  RRSIG KEY (…) kids.net. 2
  RRSIG KEY (…) kids.net. 5
ns1   A  127.0.10.3
  RRSIG 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 if the master and slave have picked up the change

Unscheduled Rollover Problems

- Needs out of band communication
  - with parent and pre-configured resolvers
- The parent needs to establish your identity again
- How to protect child delegations?
  - 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 - Summary

1. Generate new KSK
2. Sign with old and new KSKs
3. Inform any resolvers that have you as a trusted entry point of the new key
   - `trusted-keys` configuration
4. Query for the parental DS and remember the TTL
   - you will need it later
5. Upload the new KSK to the parent
   - The parent will generate a new DS RR.
6. Check if ***all*** parental servers (slaves and masters) picked up the change
7. Wait another TTL before removing the old key

Questions?

Operational Concerns

.nl trial 2003

- ~800.000 delegations
- 40 MB unsigned, 350+ MB signed
  - .com grows to 10 Gigabyte!
  - unsigned .se ~9 MB, signed .se ~32 MB
- Daily signing: ~1.5 hours
- Loading: ~15 minutes
- Very few NXT (NSEC) walks (rate limiting)

Tips…

- Sort zone before signing
  - speeds up the signing process
- If signed zone > 3 Gigabytes: 64 bit architecture
- DNSSEC Deployment Working Group
  - http://www.sdl.sri.com/other/dnssec/
- DNSSEC deployment in .nl
  - http://www.nlnetlabs.nl/dnssec/

Signing the Root

- 3 Organisations check root zone; who signs?
  - IANA/ICANN
  - Department of Commerce
  - Verisign
- How many keys?
  - N of M system for trust?

Resolver Issues

- 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)

End User Side

- Local verifying/recursive server trusted?
  - TSIG for queries?
  - IPSec?
  - Enhance stub resolver functionality?
- How much information needed?
  - AD bit enough?
  - Verifier built in to programs?

Wish List

- Public and private key management tools
- Provisioning tools
- Secure Islands public keys distribution

- API & protocol to communicate validation results
- Killer App that relies on DNSSEC

- Documentation/training/tools in order to reduce costs

Conclusions

What Did We Cover

- DNSSEC provides a mechanism to protect DNS
- DNSSEC implementation:
  - TSIG for communication between servers
  - RRSIG, DNSKEY and NSEC for data
  - DS for delegating trust
- DNSSEC main difficulties:
  - Key distribution
  - Chicken & Egg

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

Thank You!

- Please
  - Fill out questionnaire
  - Return badges for recycling
  - Pick up your certificate

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

- Feedback on this tutorial.
  - Suggestions: **training@ripe.net**

Resources

Reference:

- <sourcedir>/doc /arm/Bv9ARM.html
- DNS and BIND, Albitz & Liu, O' Reilly & Associates
- FAQ: http://www.nominum.com/resources/faqs/bind-faqs.html

RFCs:

- <sourcedir>/doc/rfc/
- http://www.ietf.org
- ftp://ftp.ripe.net/rfc/

Drafts:

- http://www.ietf.org
- ftp://ftp.ripe.net/internet-drafts/

Additional Resources

- http://www.nlnetlabs.nl/dnssec/
- http://www.dnssec.net/
- http://www.ripe.net/disi/
- Papers from the 5th USENIX UNIX Security Symposium, Salt Lake City, Utah, June 1995
  - P. Vixie: DNS and BIND Security Issues
    - http://www.usenix.org/publications/library/proceedings/security95/vixie.html
  - S. Bellovin: Using the DNS for Break-ins
    - http://www.usenix.org/publications/library/proceedings/security95/bellovin.html

Related mailing lists

- dnssec@cafax.se
  - operators and developers working on dnssec
- namedroppers@ops.ietf.org
  - DNSEXT IETF working group (DNS protocol development)
- dnsop@cafax.se
  - DNSOP IETF working group (operational DNS issues)
- techsec@ripe.net
  - RIPE Technical Security working group
- dns-wg@ripe.net
  - RIPE DNS working group

TSIG for dynamic updates

- You can use TSIG or SIG0 to protect your dynamic updates
- Detailed howto at: http://ops.ietf.org
  - title: “Secure dynamic DNS howto”

- 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

TSIG for dynamic updates: client side

- /etc/dhclient.conf

```
send fqdn.fqdn “laptop.example.net.";
send fqdn.encoded on;       # send in dns wire format
send fqdn.server-update off;
                    # tell dhcp server not to update A

key me-friend. {
    algorithm HMAC-MD5;
    secret  "ic…==";
    }

zone example.net. {
    primary 193.0.0.4;
    key me-friend.;
    }
```

TSIG for dynamic updates: server side

- /etc/named.conf:

```
key me-friend. {
    algorithm HMAC-MD5;
    secret "ic…==";
    };

zone “example.net" {
      type master;
      file "zones/example.net.signed";
      notify yes;
      allow-transfer { key tsig.example.net.; };
      update-policy {
      grant  me-friend. name laptop.example.net ;
      };
};
```

- Author: Arno Meulenkamp (arno@ripe.net)
- Additions and editing by: Vesna Manojlovic (BECHA@ripe.net)
  Filiz Yilmaz (filiz@ripe.net)
  Tim McGinnis (mctim@ripe.net)
- Based on material by: Olaf M. Kolkman (okolkman@ripe.net)
- Material available at: http://www.ripe.net/training/dnssec/

RIPE NCC
Singel 258
NL-1016 AB AMSTERDAM