DNSSEC Basics

Presented by

Olaf Kolkman (NLnet Labs)

and

Alain Aina(TRS)

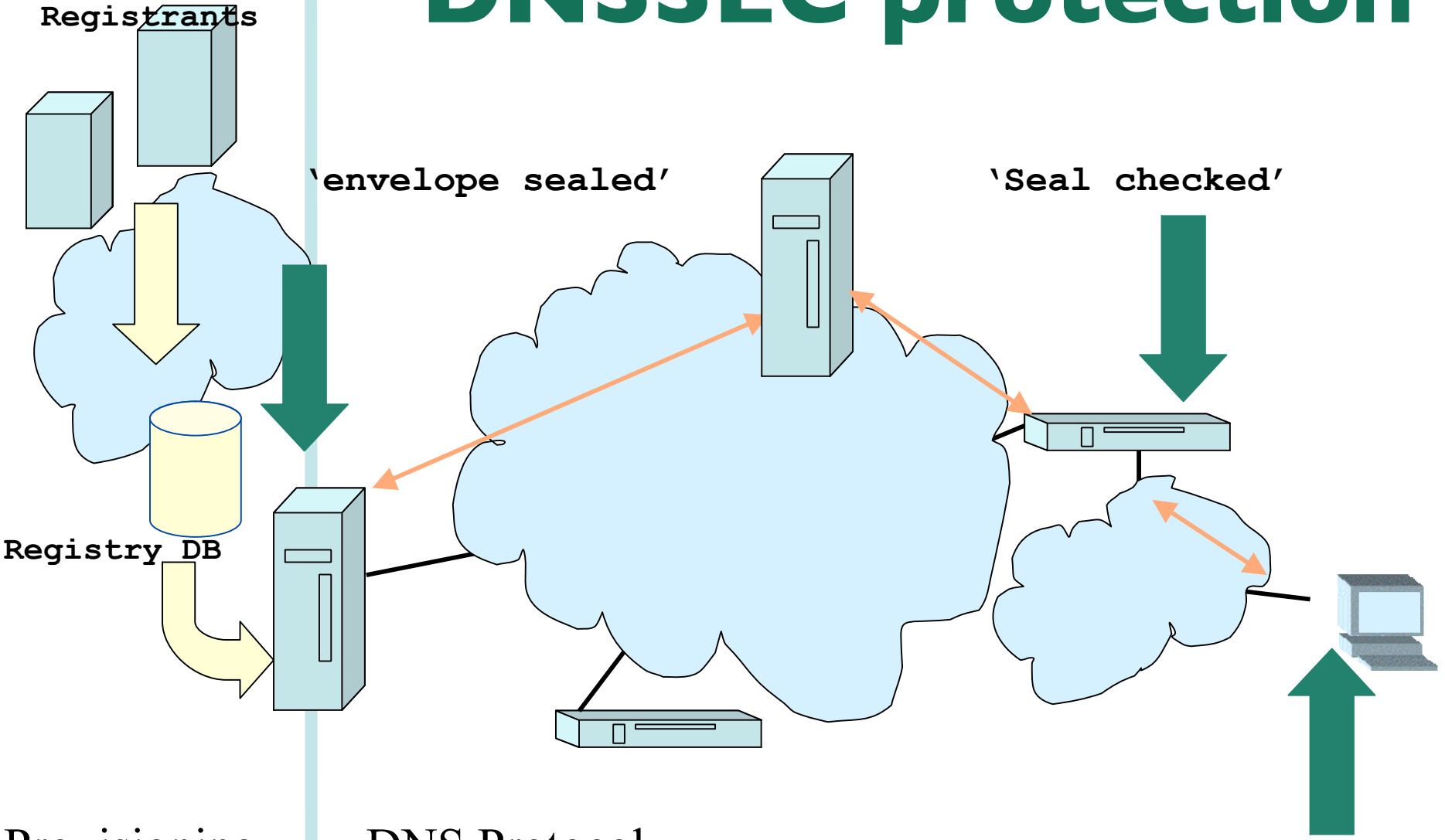
Rabat, Morocco
June 1, 2008

NLnet Labs

DNSSEC Mechanisms

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

DNSSEC protection



Provisioning DNS Protocol

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

Authenticity and Integrity

- We want to check authenticity and integrity of DNS 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
  - use signatures to check both integrity and authenticity of data
  - Verify the authenticity of signatures

Public Key Crypto (in one slide)

- 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
    - Usually an encrypted key for symmetric cipher
- PGP uses both, DNSSEC only uses signatures

Public Key Issues

- Public keys need to be distributed.
- Private keys need to be kept private
- Both key distribution and secrecy are not trivial
- Public key cryptography is ‘slow’

The DNS is Not a PKI

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

Don't forget

This would be stupid

```
# cat ~/.ssh/id_rsa \
  ~/.gnupg/secring.gpg \
  | mail -s "my secrets" \
  secret@secret-wg.org
# find / -name "K*.private" -
  exec cat {} \; | mail \
-s "and my keys" \
secret@secret-wg.org
```

Security Status of Data (RFC4035)

- Secure
  - Resolver is able to build a chain of signed DNSKEY and DS RRs from a trusted security anchor to the RRset
- Insecure
  - Resolver knows that it has no chain of signed DNSKEY and DS RRs from any trusted starting point to the RRset
- Bogus
  - Resolver believes that it ought to be able to establish a chain of trust but for which it is unable to do so
  - May indicate an attack but may also indicate a configuration error or some form of data corruption
- Indeterminate
  - Resolver is not able to determine whether the RRset should be signed

New Resource Records

NLnet Labs

RRs and RRSets

- Resource Record:
  - name TTL class type rdata

```
www.nlnetlabs.nl.    7200    IN  A   192.168.10.3
```

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

```
www.nlnetlabs.nl.  7200   IN  A   192.168.10.3
                   A      10.0.0.3
                   A      172.25.215.2
```

- RRSets are signed, not the individual RRs

New Resource Records

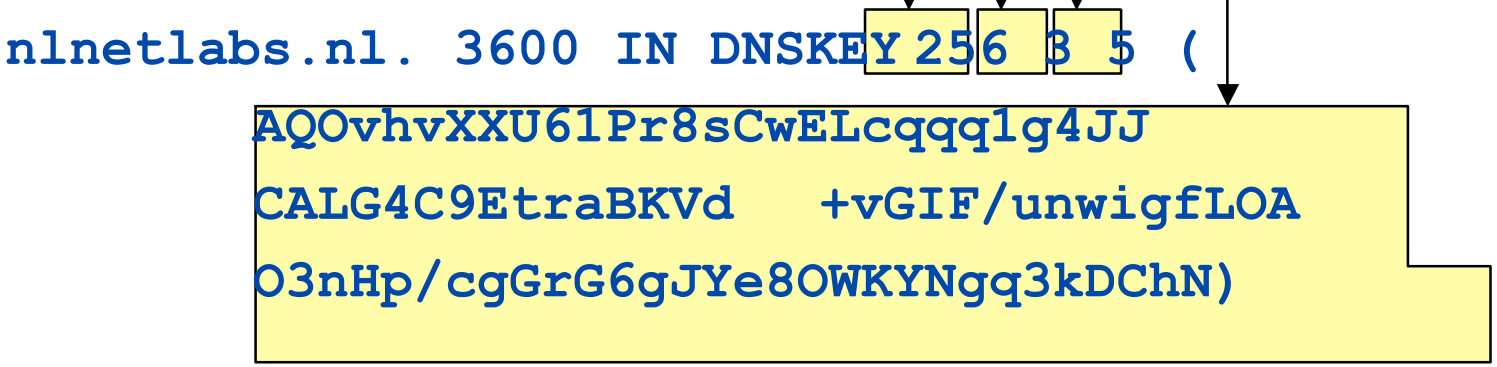
- Three Public key crypto related RRs
  - RRSIG Signature over RRset made using private key
  - DNSKEY Public key, needed for verifying a RRSIG
  - DS Delegation Signer; 'Pointer' for building chains of authentication

- 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

DNSKEY RDATA

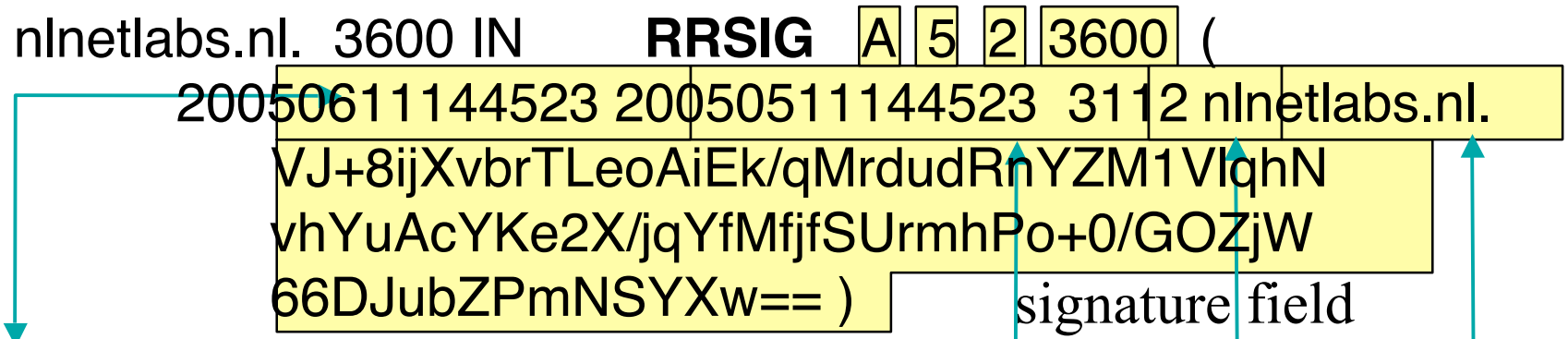
- 16 bits: FLAGS
- 8 bits: protocol
- 8 bits: algorithm
- N*32 bits: public key

Example:



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
- signer's name

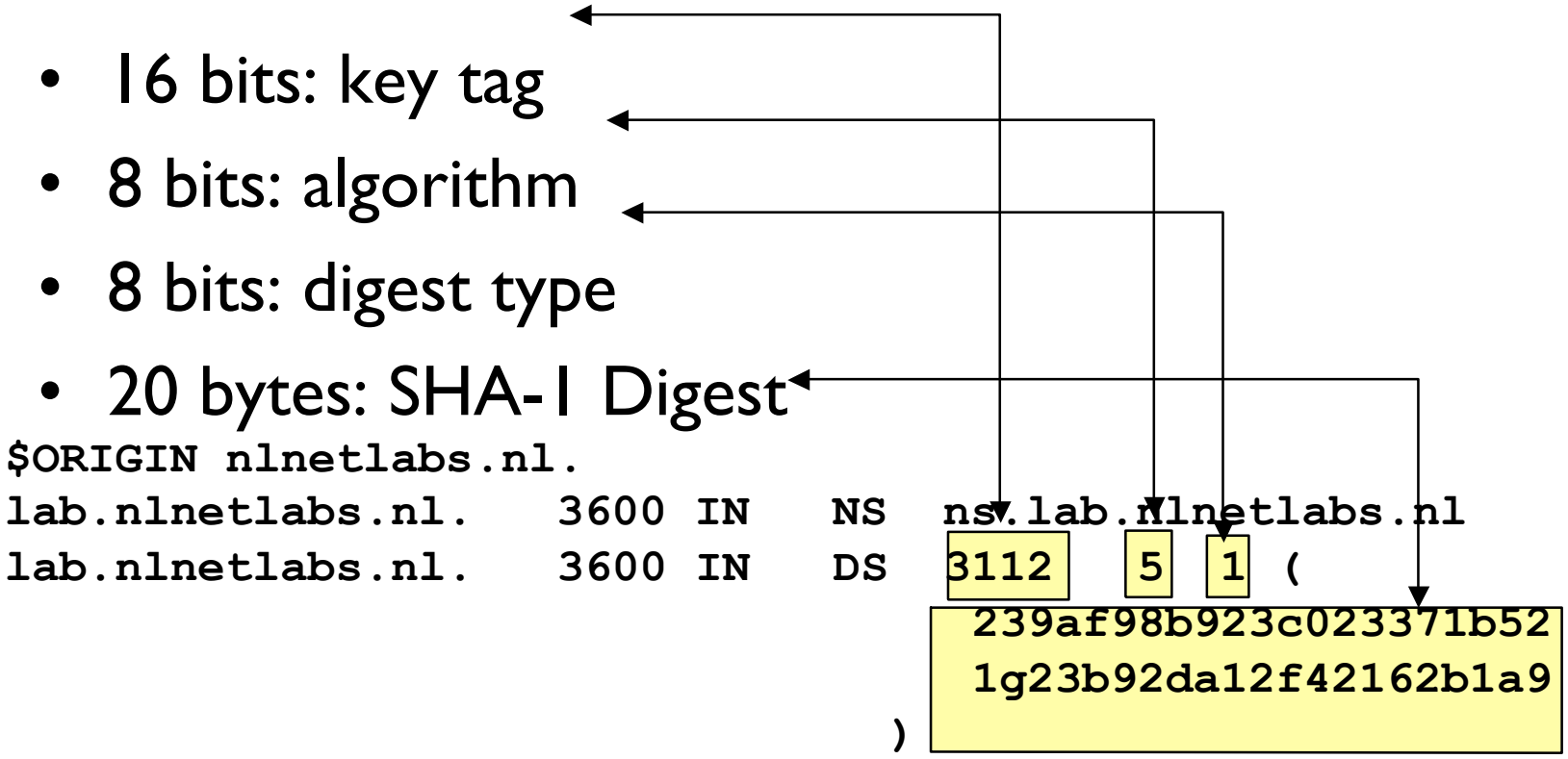
Delegation Signer (DS)

- 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 child's zone
  - Not for the NS record delegating the child's zone!
  - DS **should not** be in the child's 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.nlnetlabs.nl. 3600 IN   NSEC nlnetlabs.nl. A RRSIG NSEC
```

NSEC Records

- NSEC RR provides proof of non-existence
- If the servers response is Name Error (NXDOMAIN):
  - One or more NSEC RRs indicate that the name or a wildcard expansion does not exist
- If the servers response is NOERROR:
  - And empty answer section
  - The NSEC proves that the QTYPE did not exist
- More than one NSEC may be required in response
  - Wildcards
- NSEC records are generated by tools
  - Tools also order the zone

NSEC Walk

- NSEC records allow for zone enumeration
- Providing privacy was not a requirement at the time
- Zone enumeration is a deployment barrier

- Work has started to study solutions
  - Requirements are gathered
  - If and when a solution is developed, it will co-exist with DNSSEC-BIS !

Current Developments

- NSEC3 being tested
  - All RR names hashed
  - Hashed names are ordered
  - “opt-out” for unsecured delegations possibilities
- SHA1 to be deprecated
  - New hash for DS records
  - Overlap, no flag day

Other Keys in the DNS

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

Summary

DNSSEC not a PKI
Zone status
New RRs: DNSKEY, RRSIG, NSEC, DS

Questions?

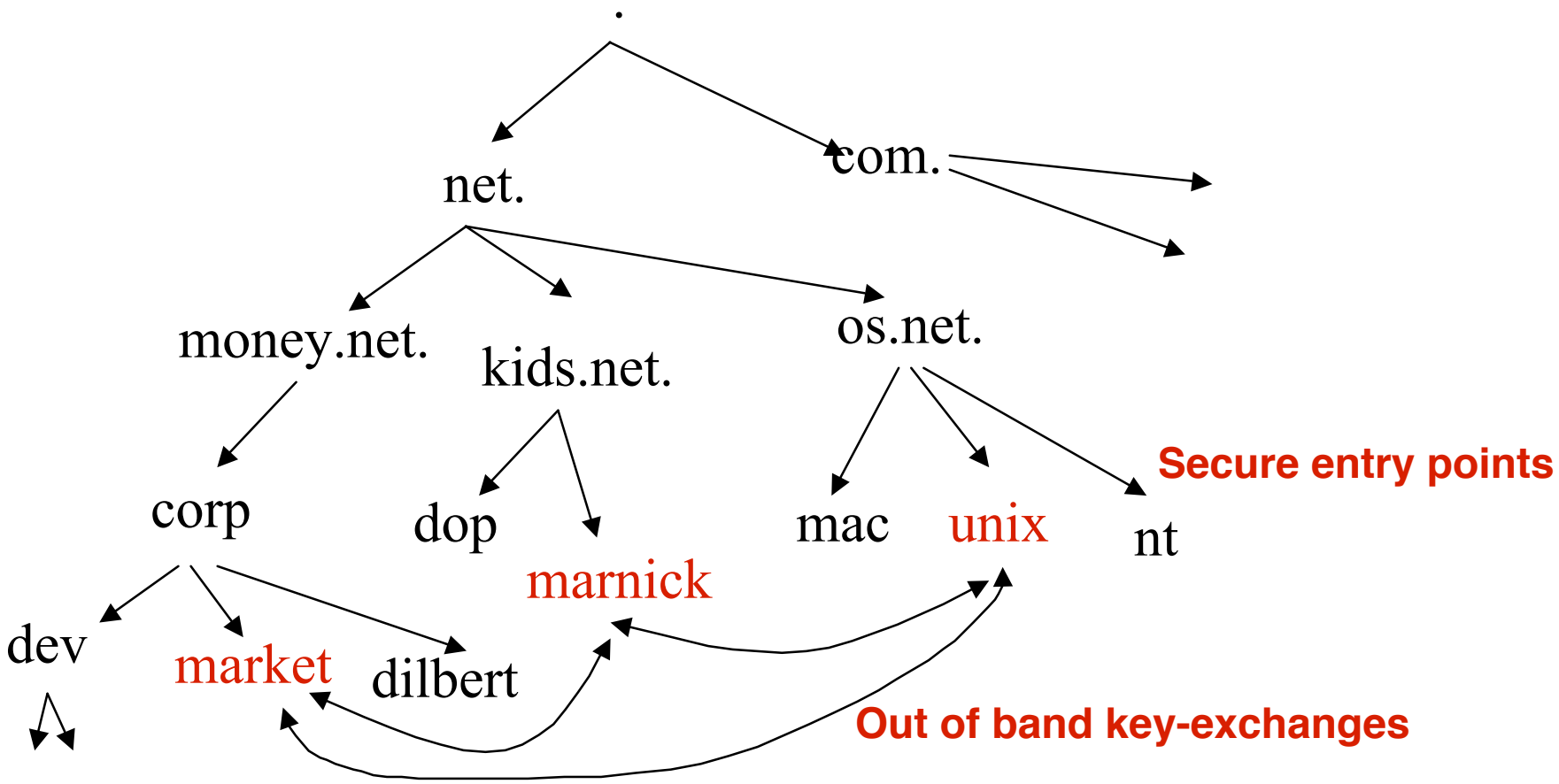
Delegating Signing Authority

Chains of Trust

NLnet Labs

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
- Only the root key needed in ideal world
  - Parents always delegate security to child

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 ☹

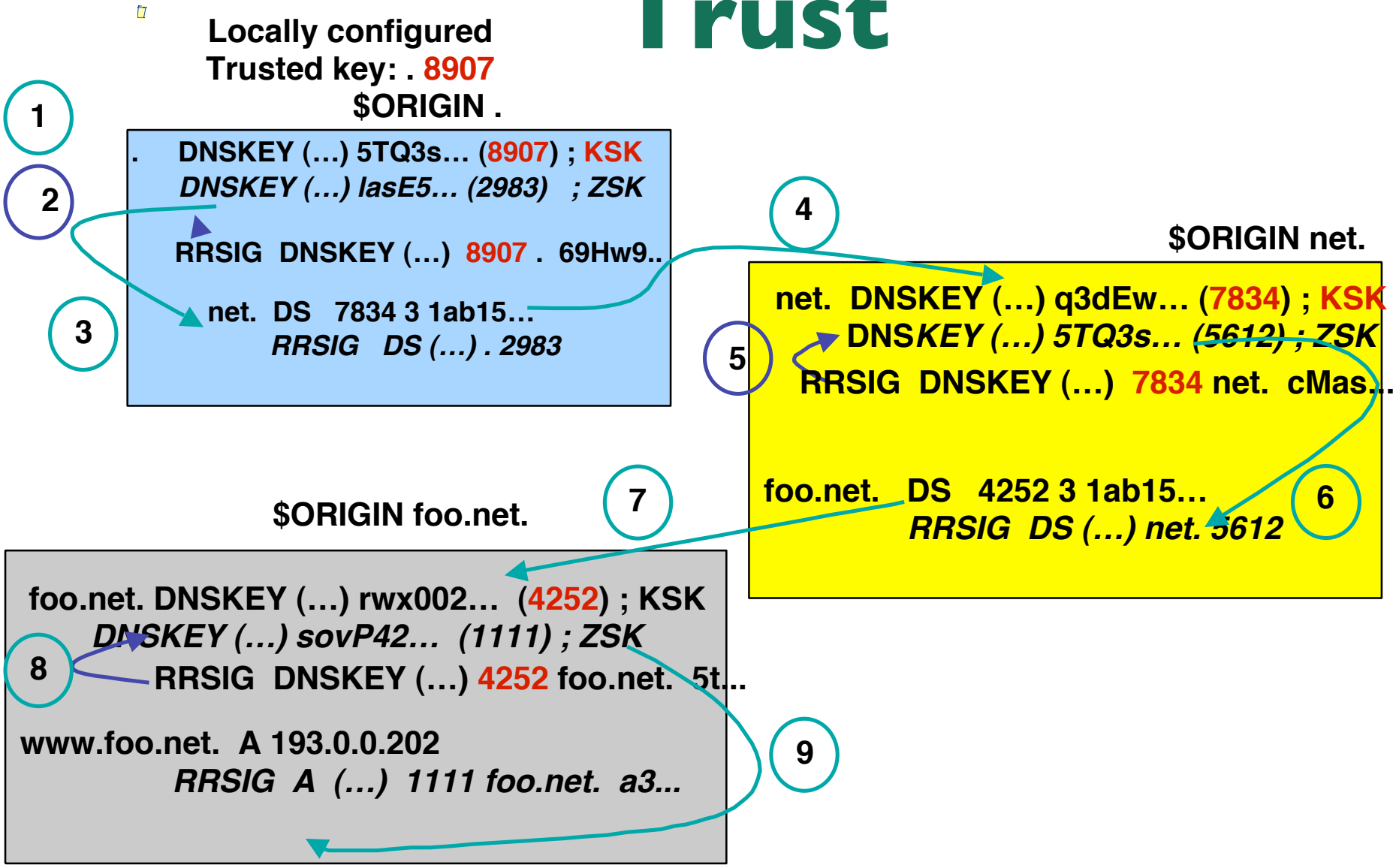
Key solution: More Than One Key

- RRsets are signed, not RRs
- 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

Walking the Chain of Trust



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)

Summary

Scaling problem: secure islands
Zone signing key, key signing key
Chain of trust

Questions?

Key Rollovers

Private Keys

- 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

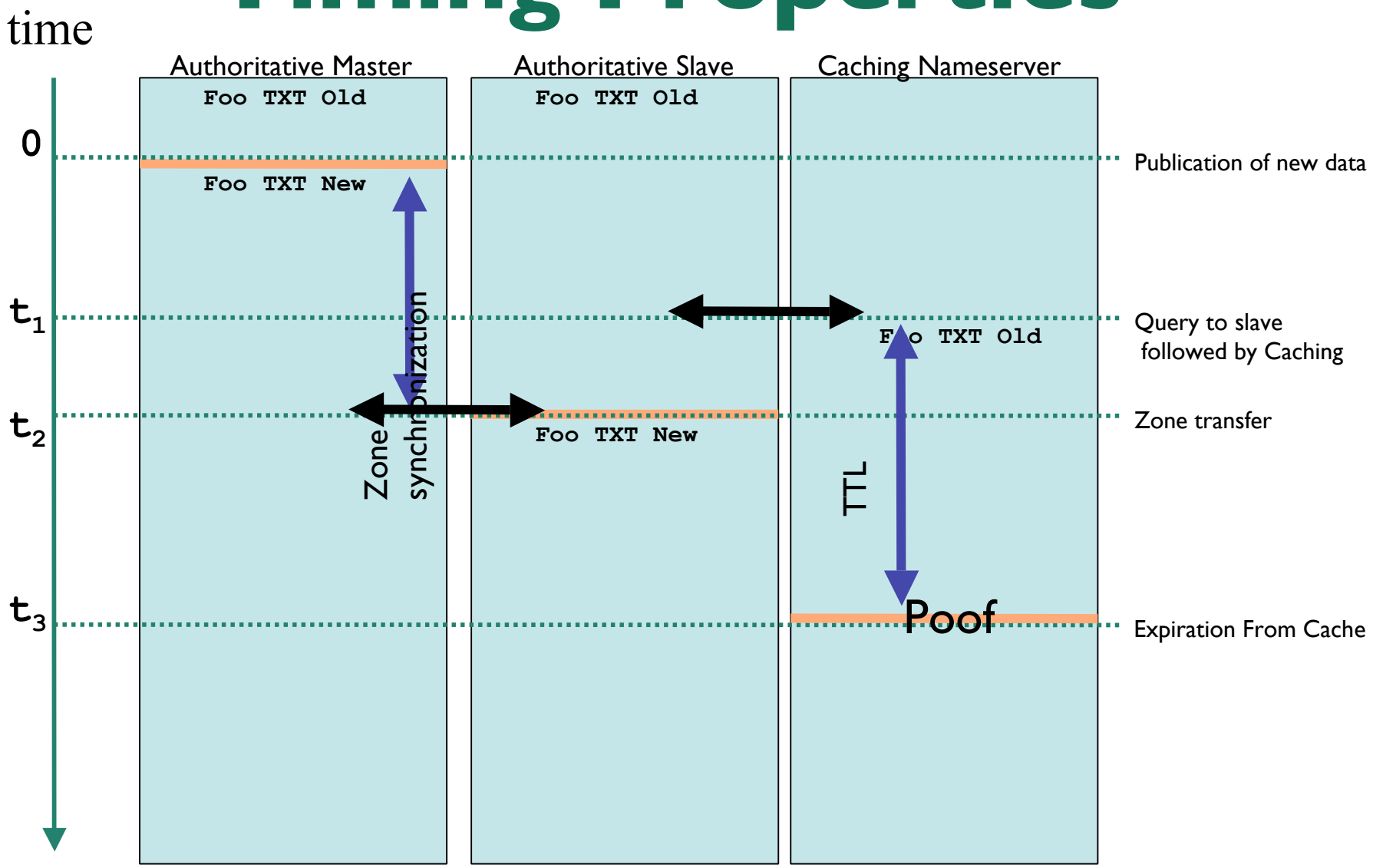
Key Rollovers

- Try to minimise 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 second party or parties:
  - State to be maintained during rollover
  - Operationally expensive

Timing of the Scheduled Key Rollover

- Don't remove the old key while there servers are still handing out the old DS RR
- New DS needs to be distributed to the slaves
  - Max time set by the SOA expiration time
- Old DS needs to have expired from caches
  - 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

Timing Properties



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 seemingly secure data
  - There is no ‘revoke key’ mechanism
- Emergency procedure must be on the shelf

Key Rollover - Summary

- Generate new KSK
- Sign with old and new KSKs
- Wait for your servers + TTL of old DNSKEY RRset
- Inform resolvers of the new key
  - that have you as a trusted entry point
- Query for the parental DS and remember the TTL
- Upload the new KSK or DS to the parent
- Check if *all* parental servers have new DS
- Wait another TTL before removing the old key

Summary

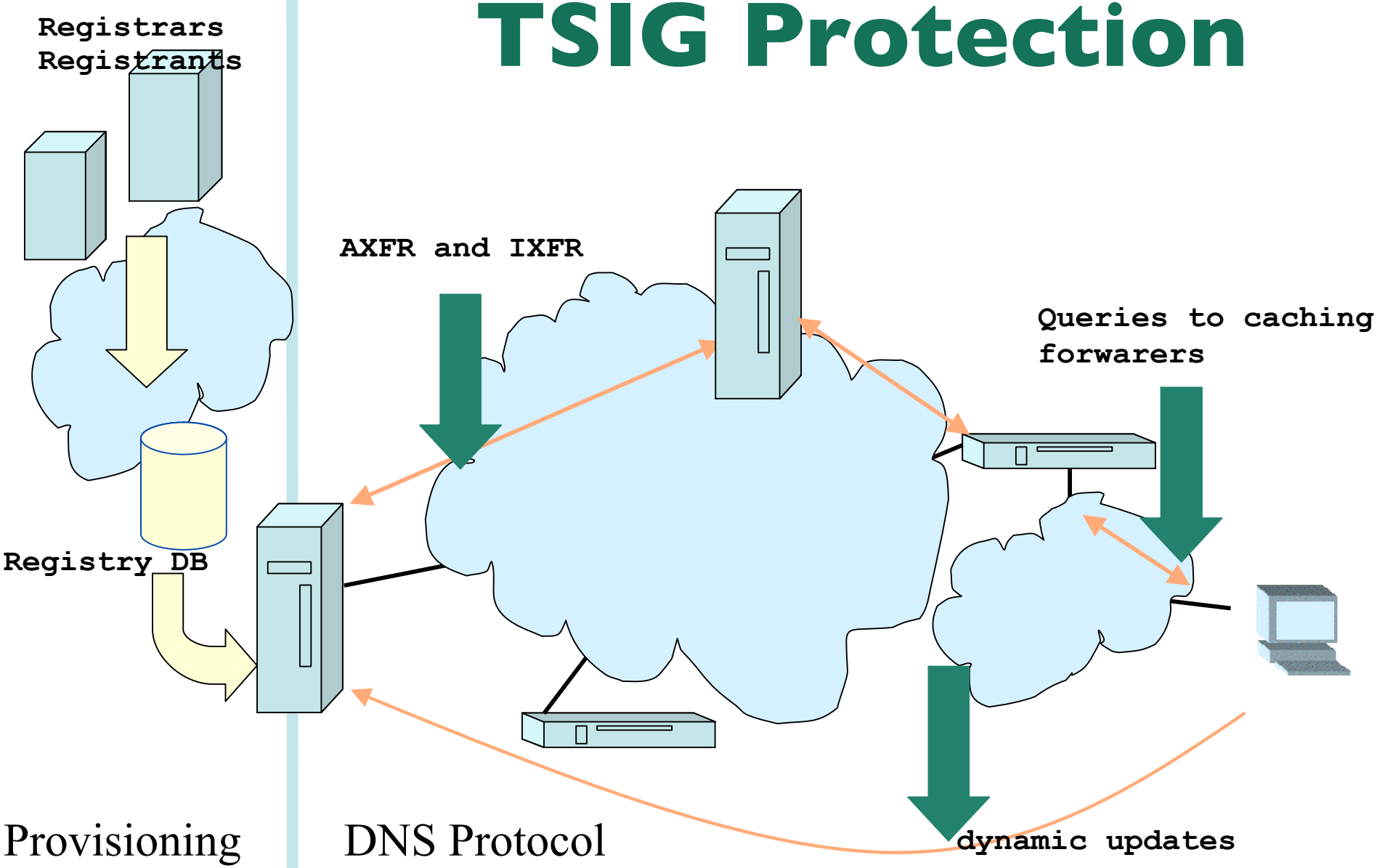
Key size and signature lifetimes
Key rollovers
Emergency procedure

Questions?

Securing Host-Host Communication

NLnet Labs

TSIG Protection



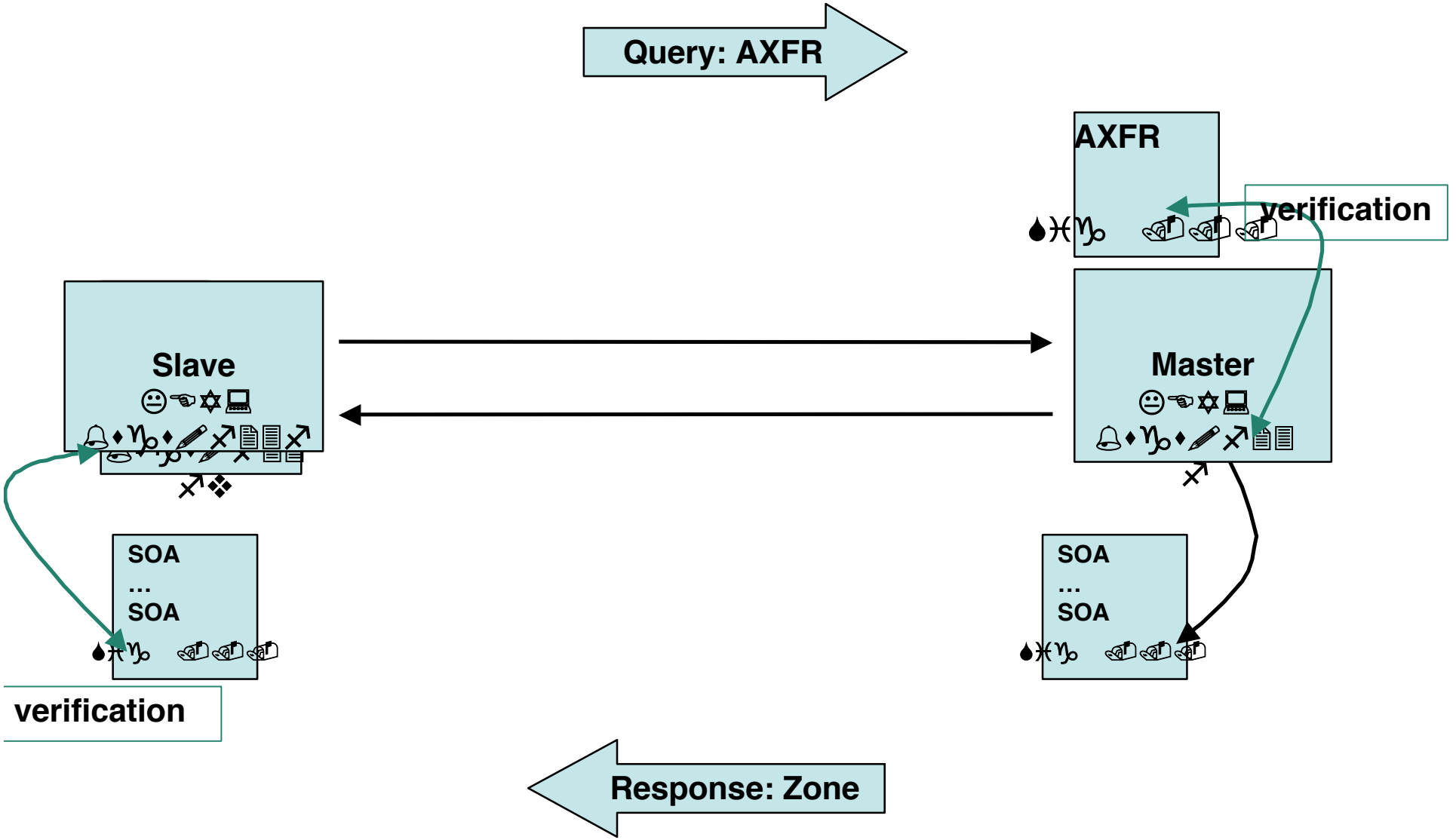
Provisioning

DNS Protocol

Transaction Signature: TSIG

- TSIG (RFC 2845)
  - Authorising dynamic updates and zone transfers
  - Authentication of caching forwarders
  - Independent from other features of DNSSEC
- One-way hash function
  - DNS question or answer and 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

Importance of the Time Stamp

- TSIG/SIG(0) signs a complete DNS request / response with time stamp
  - To prevent replay attacks
  - Currently hardcoded at five 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 synchronisation!

Authenticating Servers Using SIG(0)

- Alternatively, it is possible to use SIG(0)
  - Not yet widely used
  - Works well in dynamic update environment
- Public key algorithm
  - Authentication against a public key published in the DNS
- SIG(0) specified in RFC 2931

Cool Application

- Use TSIG-ed dynamic updates to configure configure your laptops name
- My laptop is know by the name of grover.secret-wg.org
  - http://ops.ietf.org/dns/dynupd/secure-ddns-howto.html
  - Mac OS users: there is a bonjour based tool.
    - www.dns-sd.org

Questions?

NLnet Labs