# DNSSEC

Lutz Donnerhacke

**db089309**: 1c1c 6311 ef09 d819  e029 65be bfb6 c9cb

dig +dnssec 1.6.5.3.7.5.1.4.6.3.9.4.e164.arpa. naptr

IKS service

# A protocol from better times

- An ancient protocol
- People were friendly and trustworthy
- Internet was a warm and fuzzy place
- *DNS is a protocol from admins for admins*
- Main assumption: Computers do not lie
- Idea: A hierarchical distributed database
- Store locally, read globally

# Playground to extend

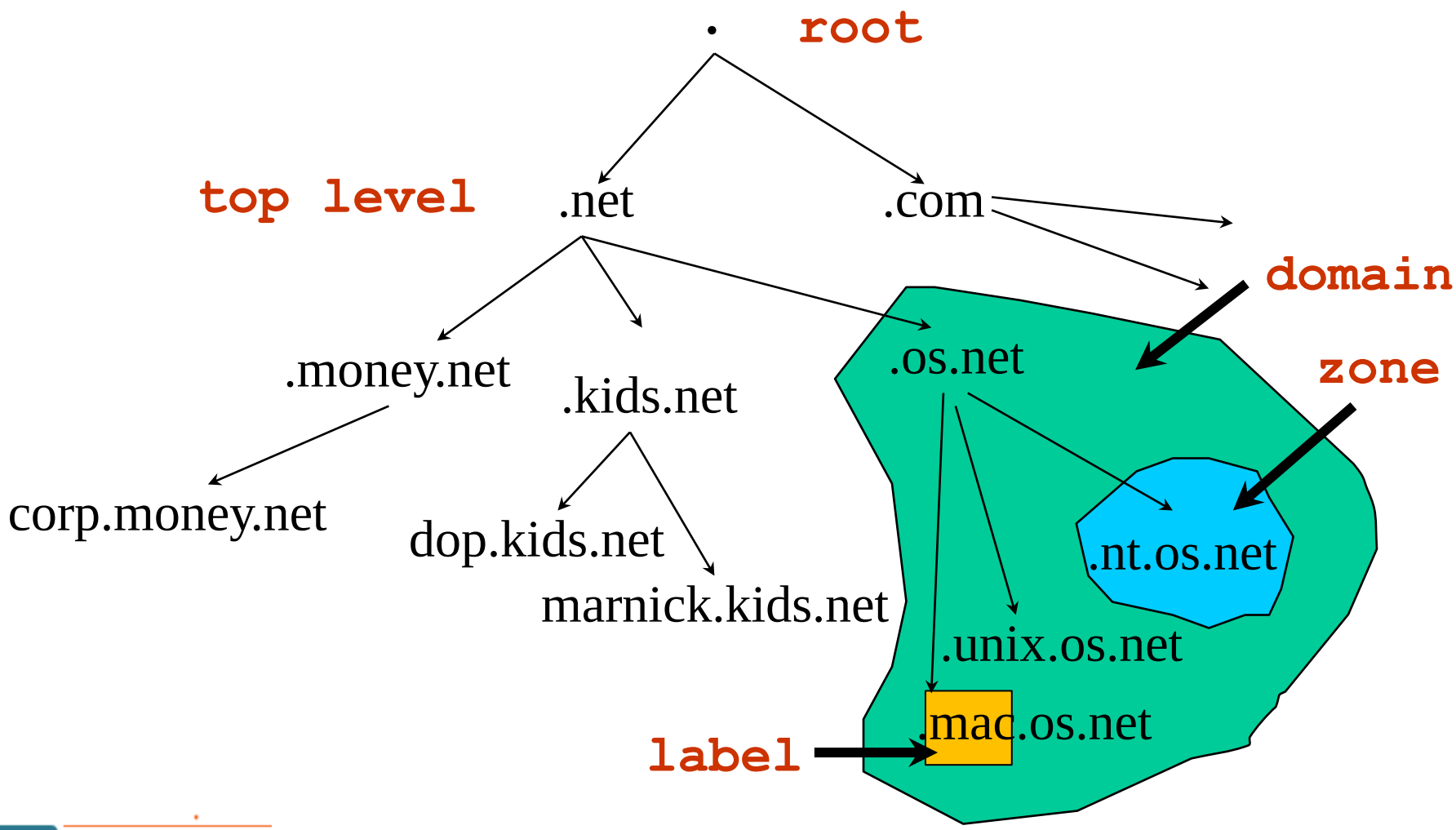
- DNS works, so use is as a container
  - http://tools.ietf.org/wg/dnsext/
- DNS scales, so push a lot of data in
  - in-addr.arpa
- DNS can be misused as a catchword repository: www.*catchword*.com
- DNS may have multiple roots, so introduce private name spaces

IKS service

# Playground to manipulate

- Push all initial requests to a payment site
- Prevent requests to *bad* sites
- Offer own search engine for NXDOMAIN
- Geolocation for efficient content delivery
- Geolocation for lawful content selection
- Provide different software updates
- Prevent worm updates

# Basic definitions

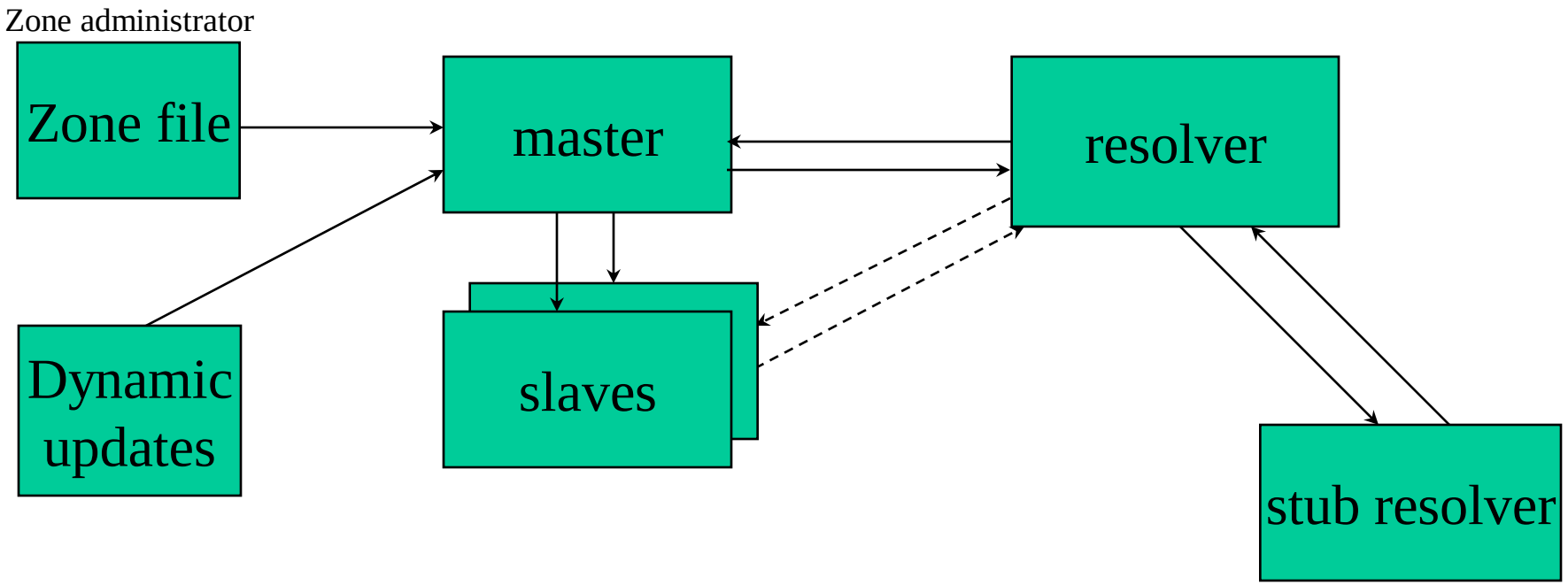


# DNS Data Flow

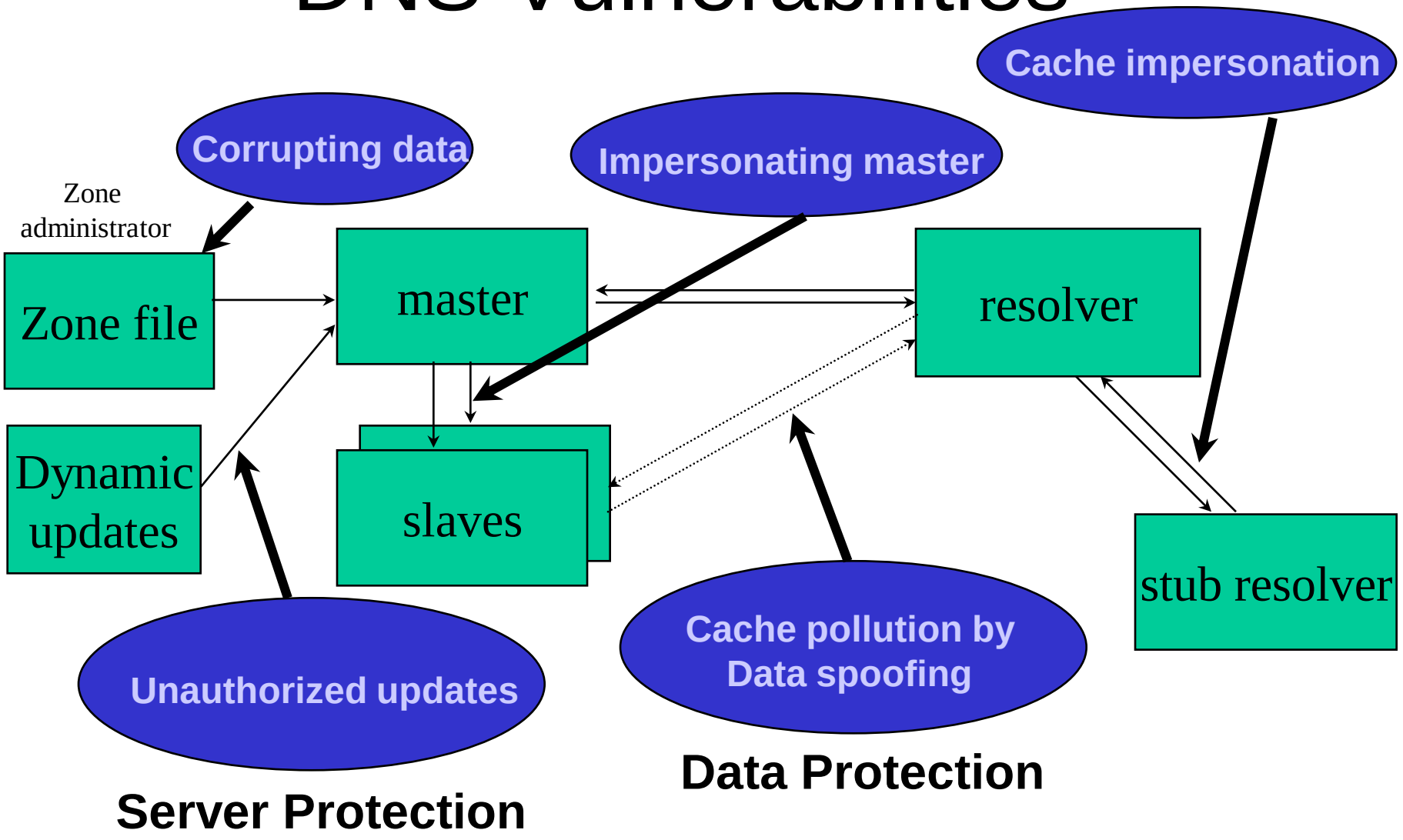
- Modelling real world data as DNS records
- Transferring data into DNS primary server
- Transferring data into DNS secondaries
- Updating meta data in parent zone
- Delivering data to recursive servers
- Processing by resolver code
- Providing structures to applications
- Interpreting data by users

IKS service

# DNS Data flow



# DNS Vulnerabilities



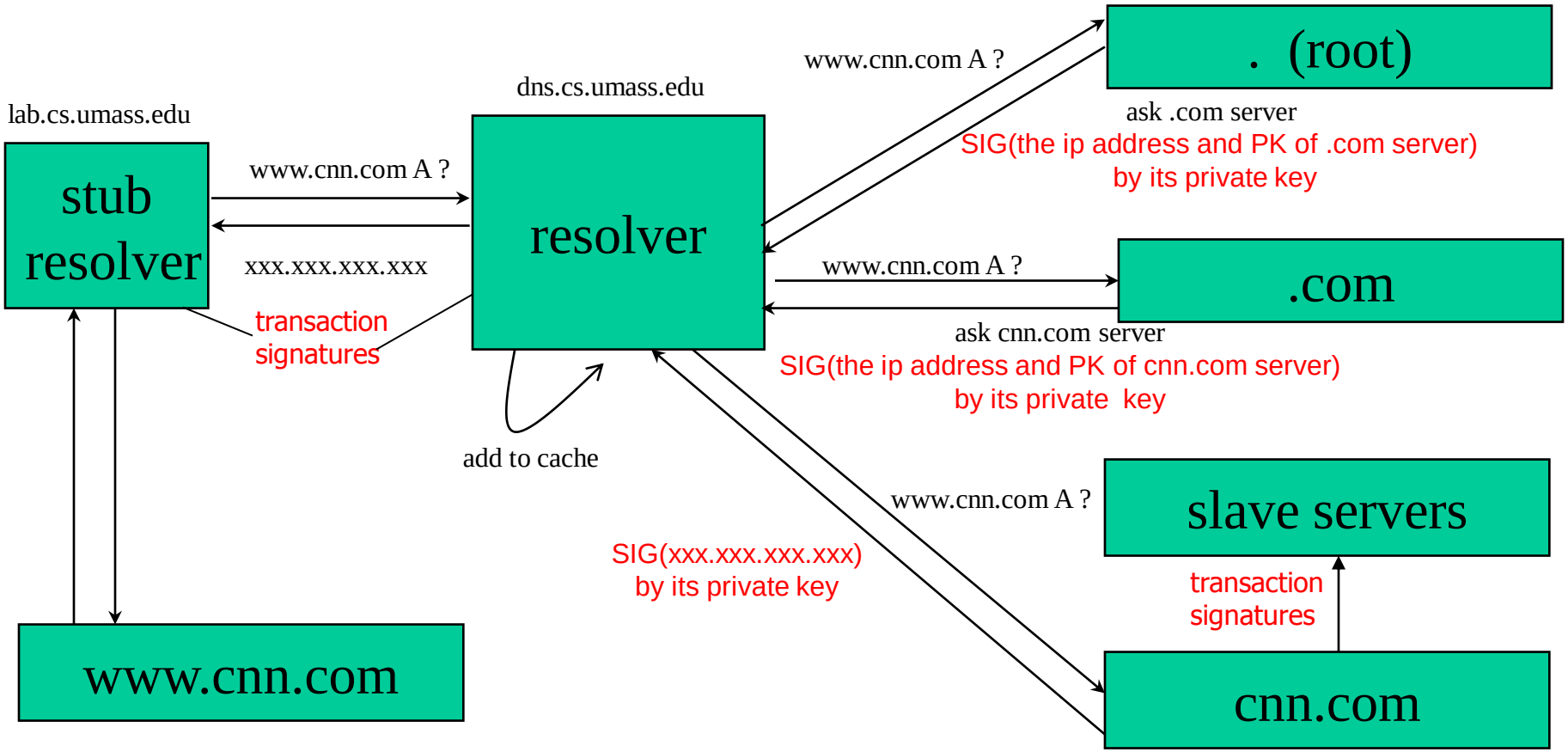
# Securing the data flow

- Two possible design goals:
  - Detect manipulation
  - Prevent wire-tapping
- Facing typical problems
  - The compatibility hydra
  - Partial roll-out
  - Satellite networks

# DNS SECurity

- Trust the primary name server data
  - Signed by the zone-c
- A framework to verify integrity
  - Signature chains up to a trust anchor
- In band key management
  - DS records in parent zone (but glue!)
- Supports caching as well as offloading
- Provides peer authentication
- Still designed by admins: NSEC(3)

# Securing the communication



http://lutz.donnerhacke.de/Projekte/DNSSEC/Livetest

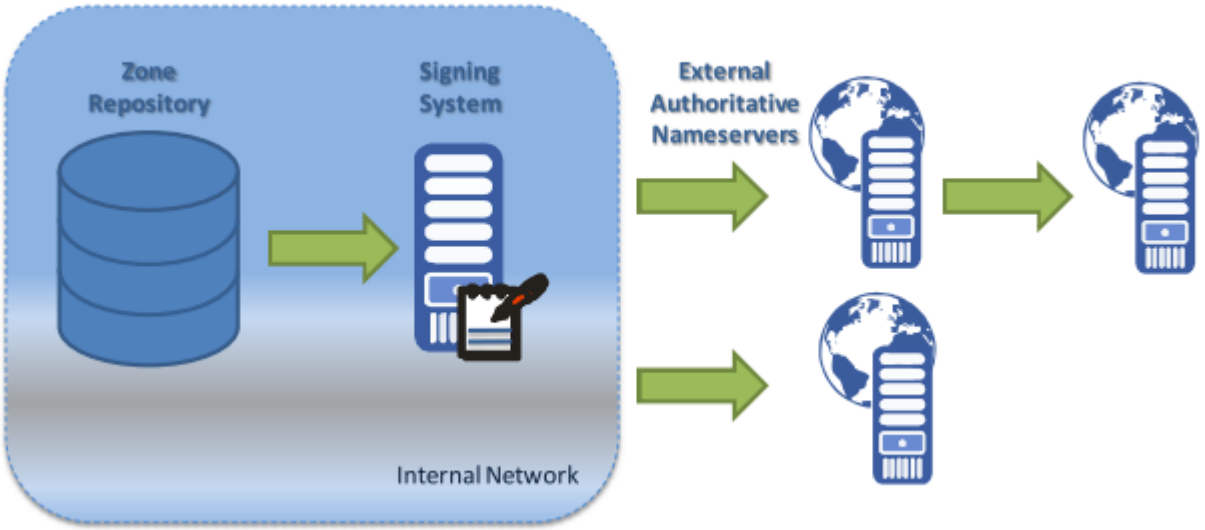
# DEMO

# Prerequisites

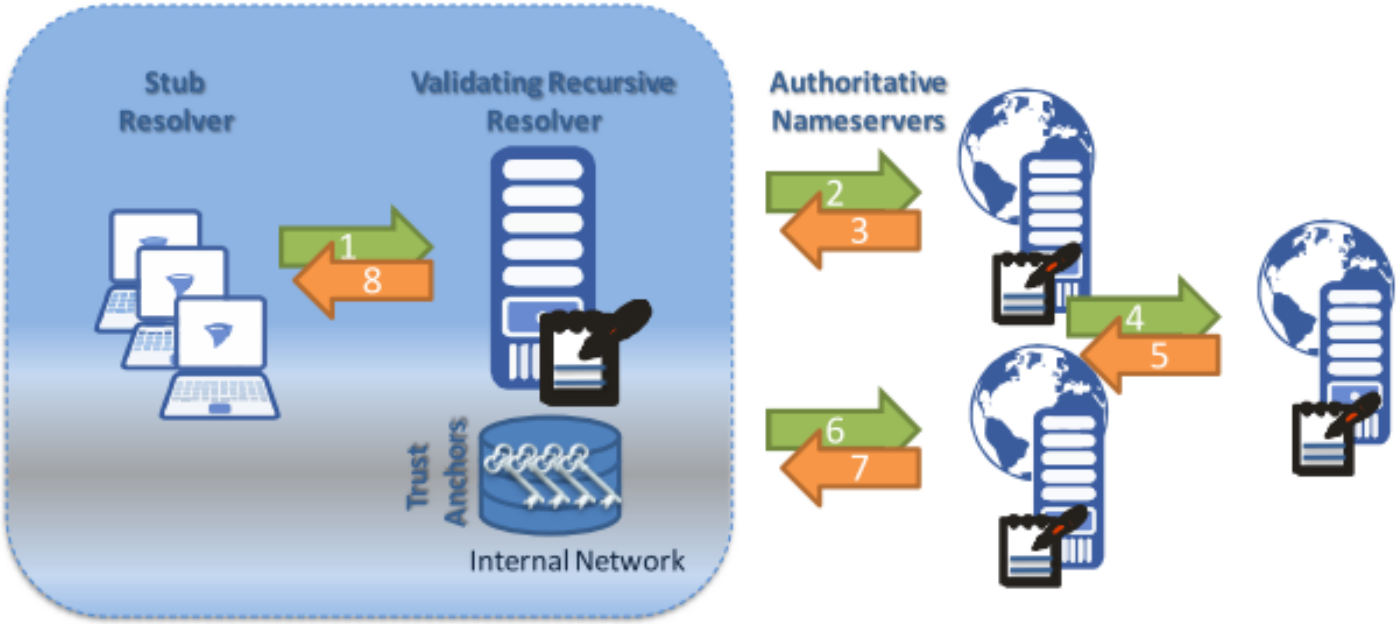
- Clean up DNS definition
  - Remove contradictional issues
  - Specify corner cases
- Define ownership of data
  - Specify glue at zone cut
  - Introduce DS in parent (long term error at Google)
- Ensure algorithm invariance
  - Parameterize and sort(!) everything
- Ease human debugging
  - Separate meta data from crypto
- Ensure backward compatibility
  - EDNS0 signalling

IKS service

# Signing System Architecture



# Validation steps




Top-Down: Resolve and check later. On error: SERVFAIL
Bottom-Up: Check every response on arrival. On error: Try others.

http://dnsviz.org/

http://lutz.donnerhacke.de/Projekte/DNSSEC/Livetest

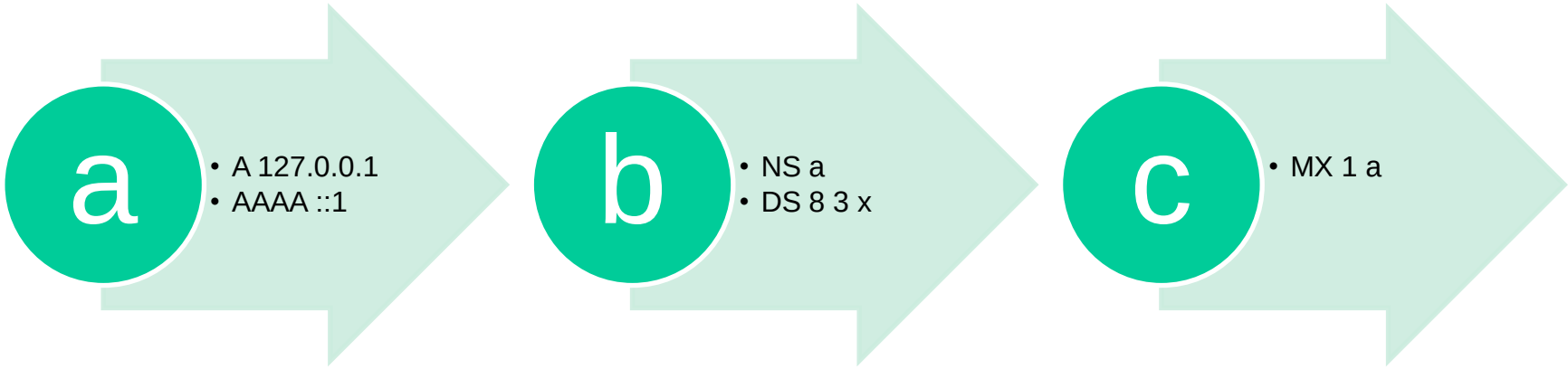
# LAB TIME

# Proof of non-existence

- *Precomputed* answers to unknown queries
- missing record type for label
  - Show all existing records of label
- missing label
  - Show half open interval containing query
- missing wildcard
  - Show half open interval containing *
- Zone walker
  - Salt and hash all the labels (NSEC3)
  - Compute NSEC on the fly (trade in CPU)

IKS service

# Proof of non-existence




```
@ NSEC a SOA NS NSEC RRSIG
a NSEC b A AAAA NSEC RRSIG
b NSEC c DS NSEC RRSIG
c NSEC @ MX NSEC RRSIG
```

http://dnsviz.org/

http://lutz.donnerhacke.de/Projekte/DNSSEC/Livetest

# LAB TIME

# Trust anchor management

- In an ideal world everything down from the root is signed
- Many roots: Trust Anchor Repositories
- Unattended roll-overs: RFC 5011
- Manual trust anchors: Edit files on disk
- Automatic trust anchors: DLV
- Open question: Precedence of sources

# Unexpected trust anchors

- Zone not delegable
  - Private name space (local, internal)
  - Reserved address space (private, CGN, local)
- Zone can’t be signed
  - System unable to handle DNSSEC
  - Internal dynamic updates
- Local trust anchor repository necessary
  - Contains keys and negative anchors

# Management hurdles

- RRSIGs time out (not keys!)
  - Resign early, resign often (worst cast ./NS)
  - Use jitter to prevent DNS storms (Why?)
  - Requires keys on public systems to sign zone
- Key rollovers
  - Distinguish between **work** and **management**
  - Limit lifetime to limit misuse if keys are lost
  - Work keys (ZSK) signed by (offline) KSK
  - KSK referenced by DS in parent zone (root?)

# Outsourcing

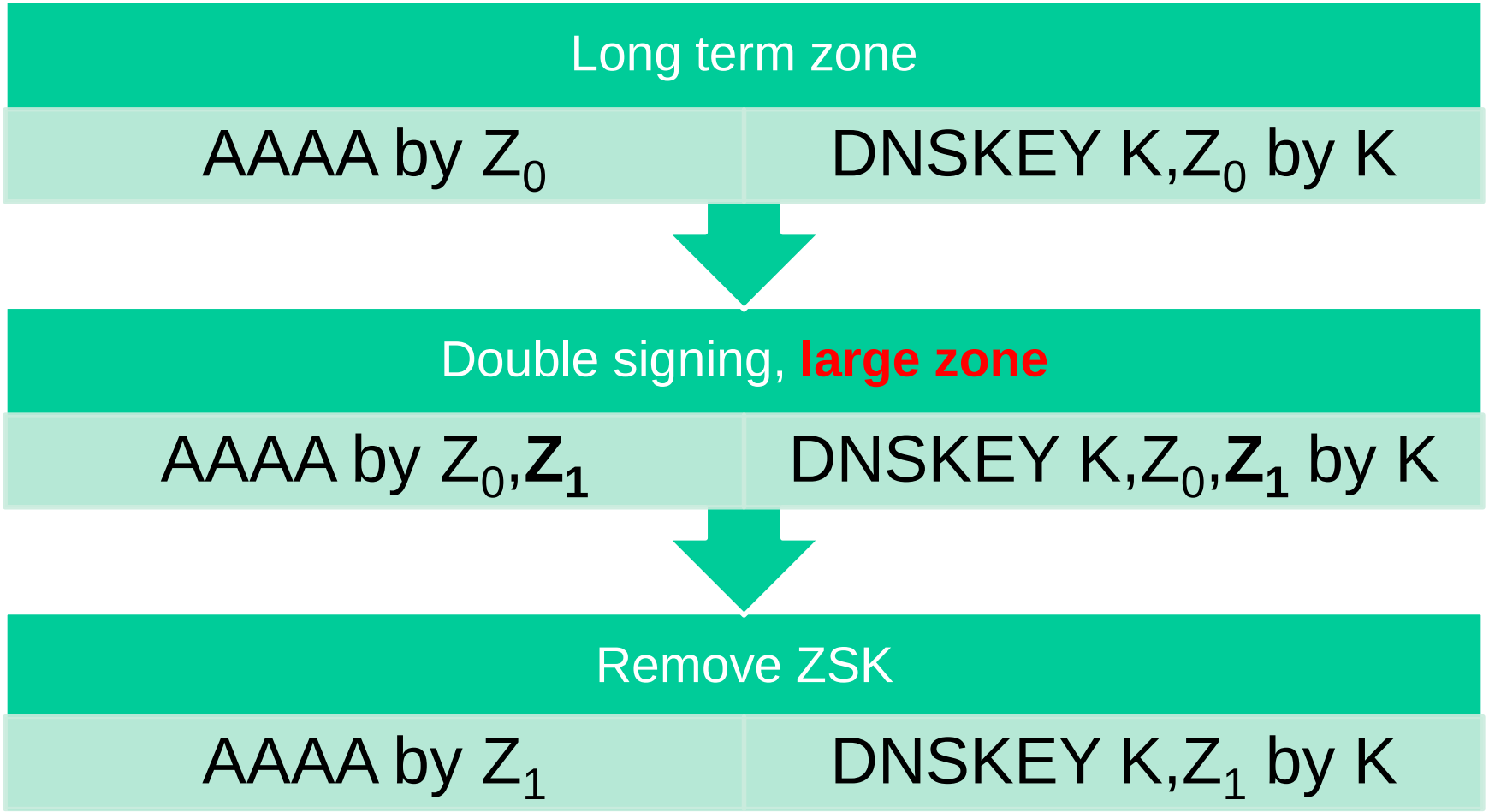
- The management way
  - Buy an appliance, install, be happy, pay fees
- Inside a remote signer appliance
  - AXFR unsigned zone into the appliance
  - Sign and resign the zones
  - NOTIFY changes to external name servers
  - Handle key rollovers automatically
  - Needs special privileges for the registry API
  - Keeps keys internally, only TSIG connections

IKS service

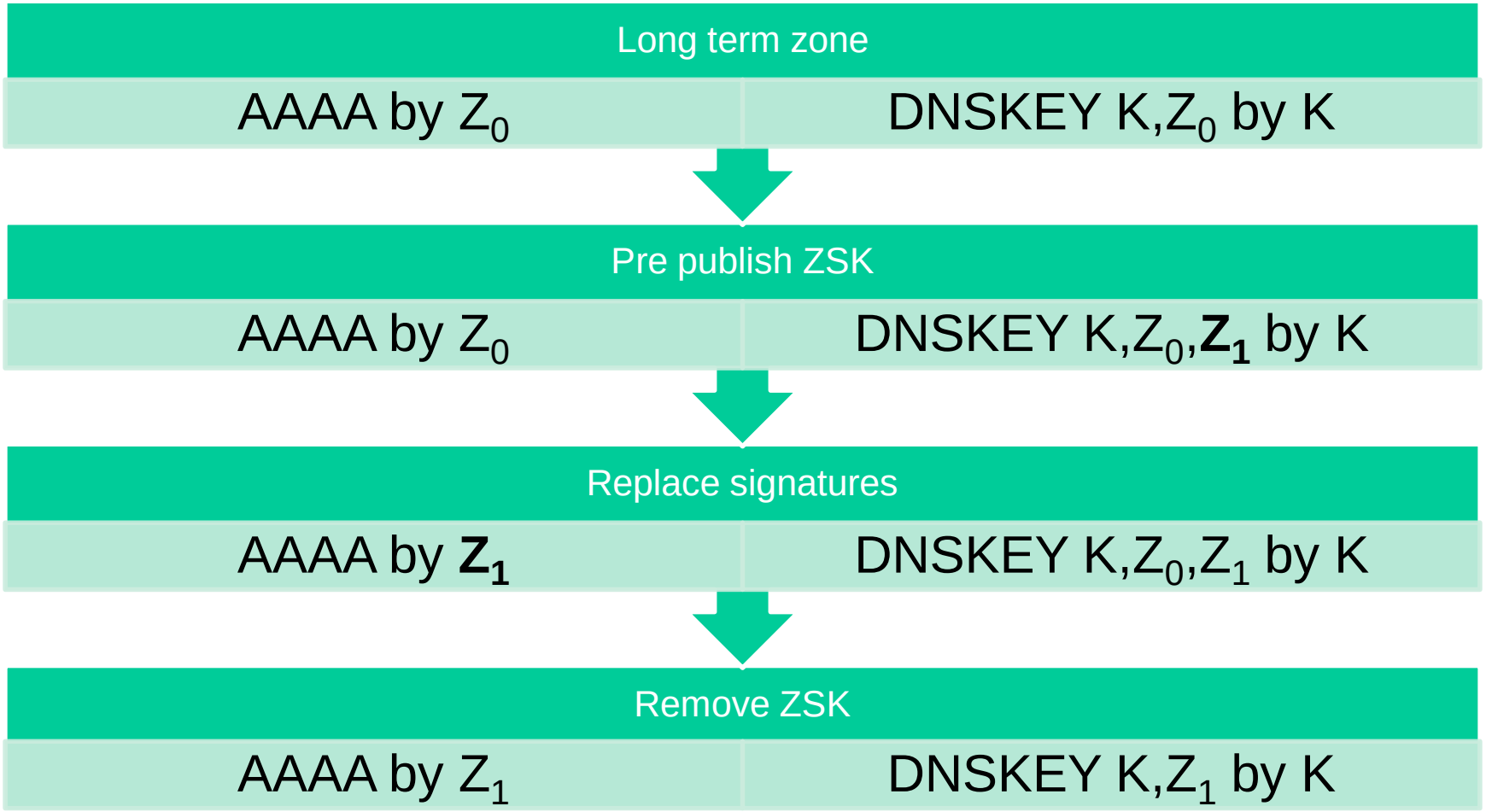
# Key rollover

- Why change ZSK?
  - Key might be exposed on public system
  - Can be computed from $e^2$ related signatures
- Why change KSK?
  - Practice operational procedures
  - HSM needs to be replaced
  - Customer changes registrar or reseller
  - Be prepared for legal action (customer, LEA)

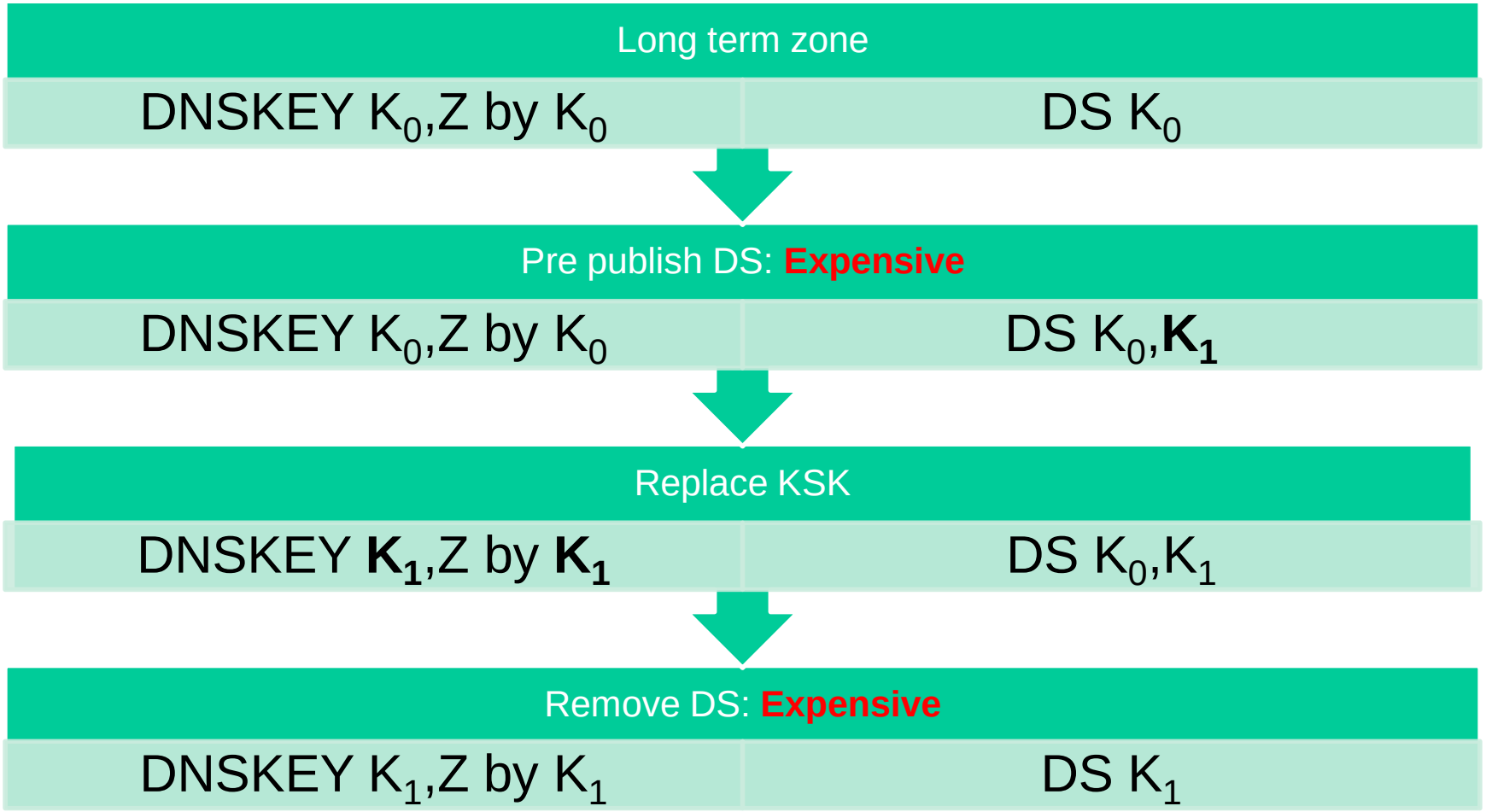
# Roll Key, Roll



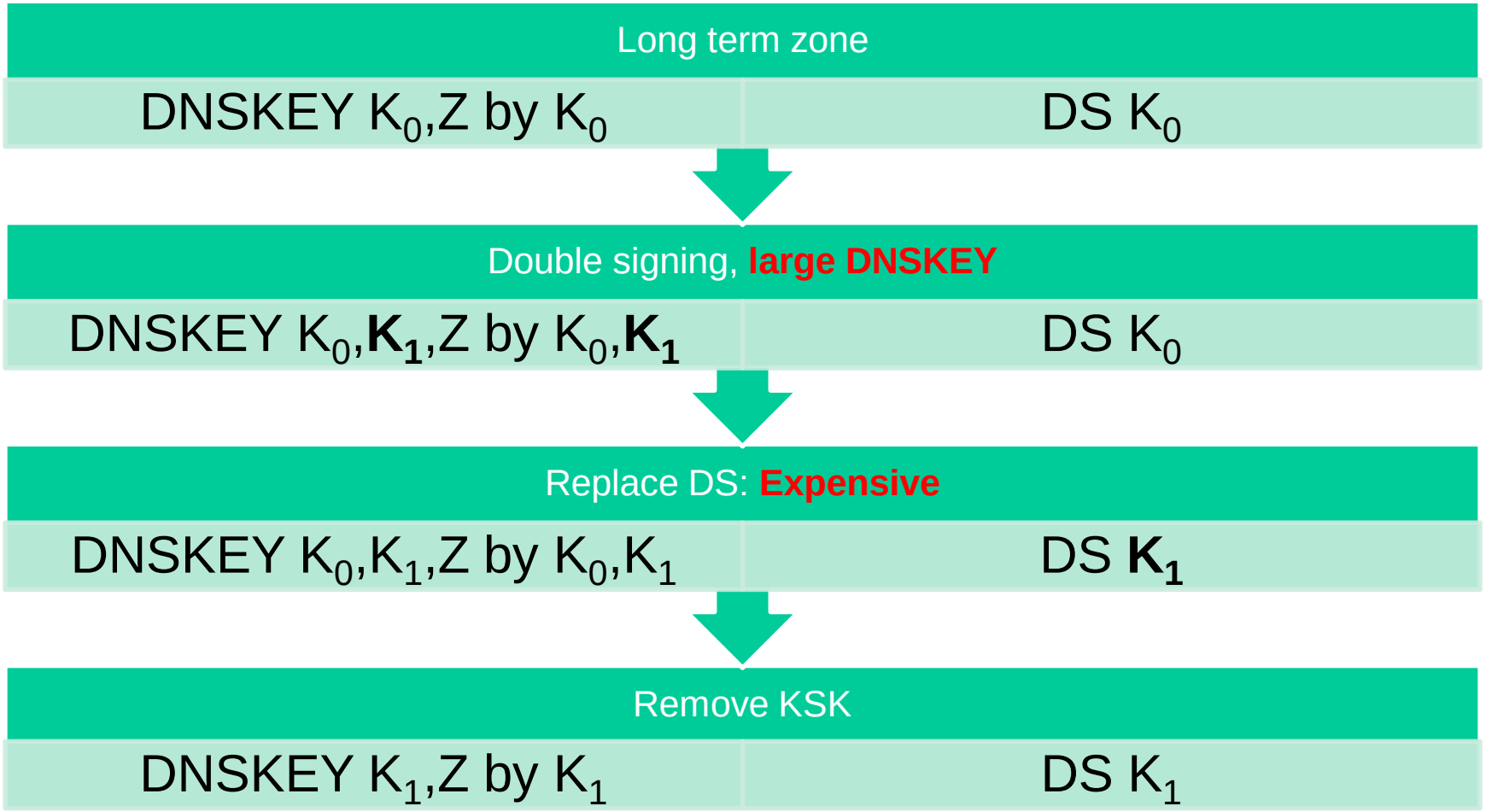
# Roll Key, Roll



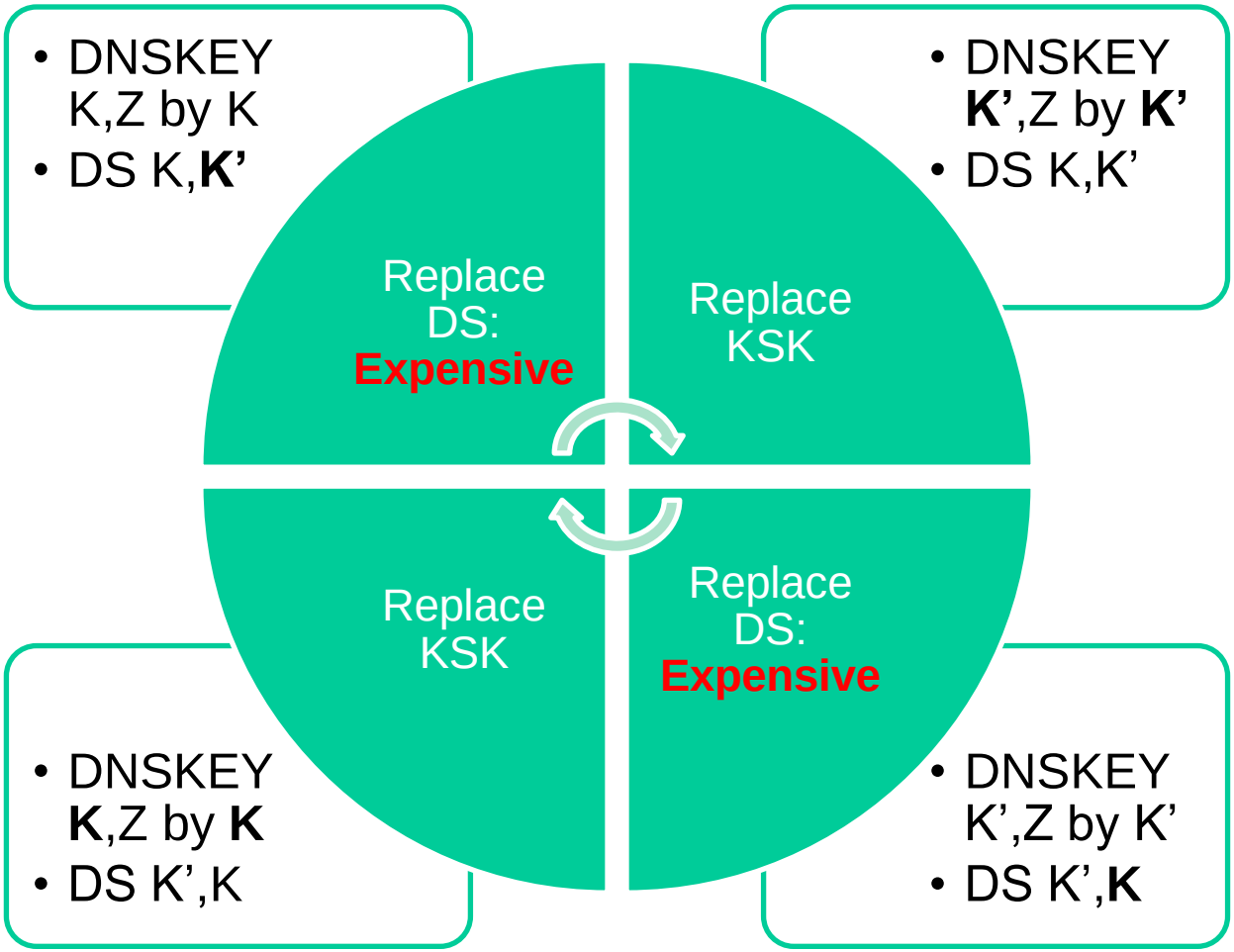
# Roll Key, Roll



# Roll Key, Roll



# Roll Key, Roll

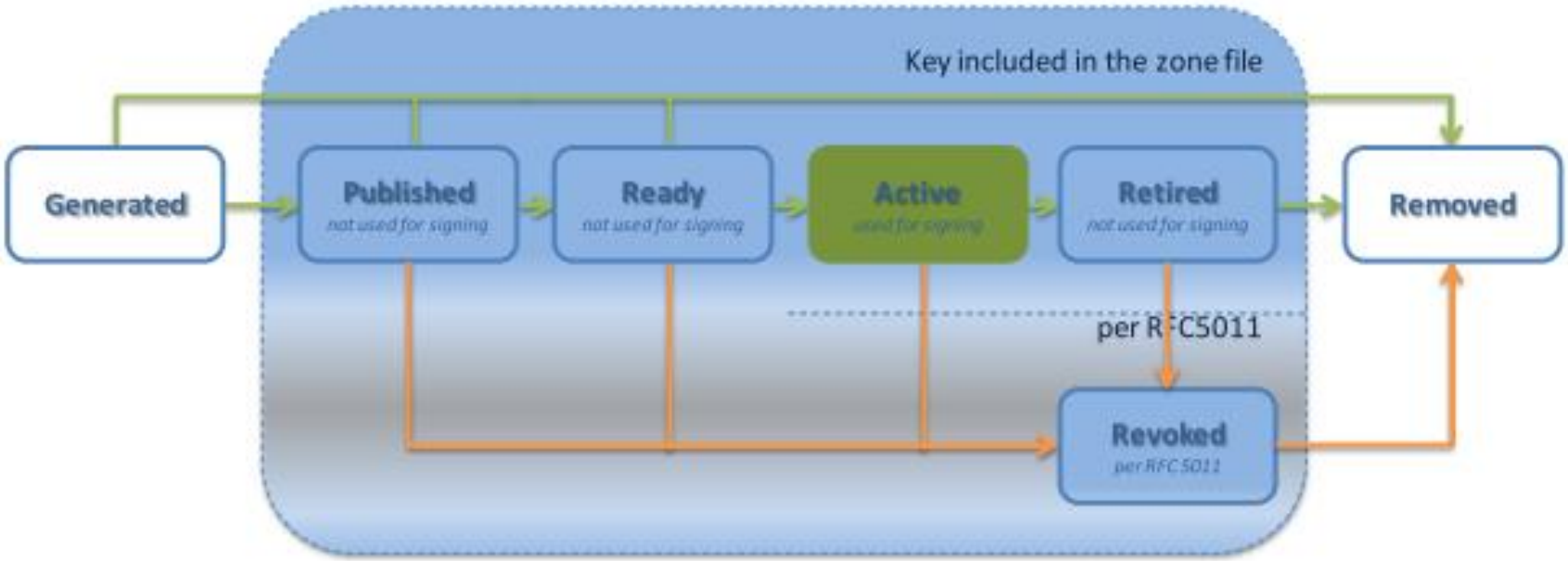


# External references

- Public keys can be copied
  - Everyone can make a TAR
- Trust Anchor Repositories are useful
  - Quite common in ITIL driven companies
  - Procedure describes only direct key exchange
- Parent zone not trustworthy
  - Registrar fuckup causes loss of zone
  - Registrant’s access token can be stolen
- RFC 5011 defines automatic update

IKS service

# Timeframe considering TARs

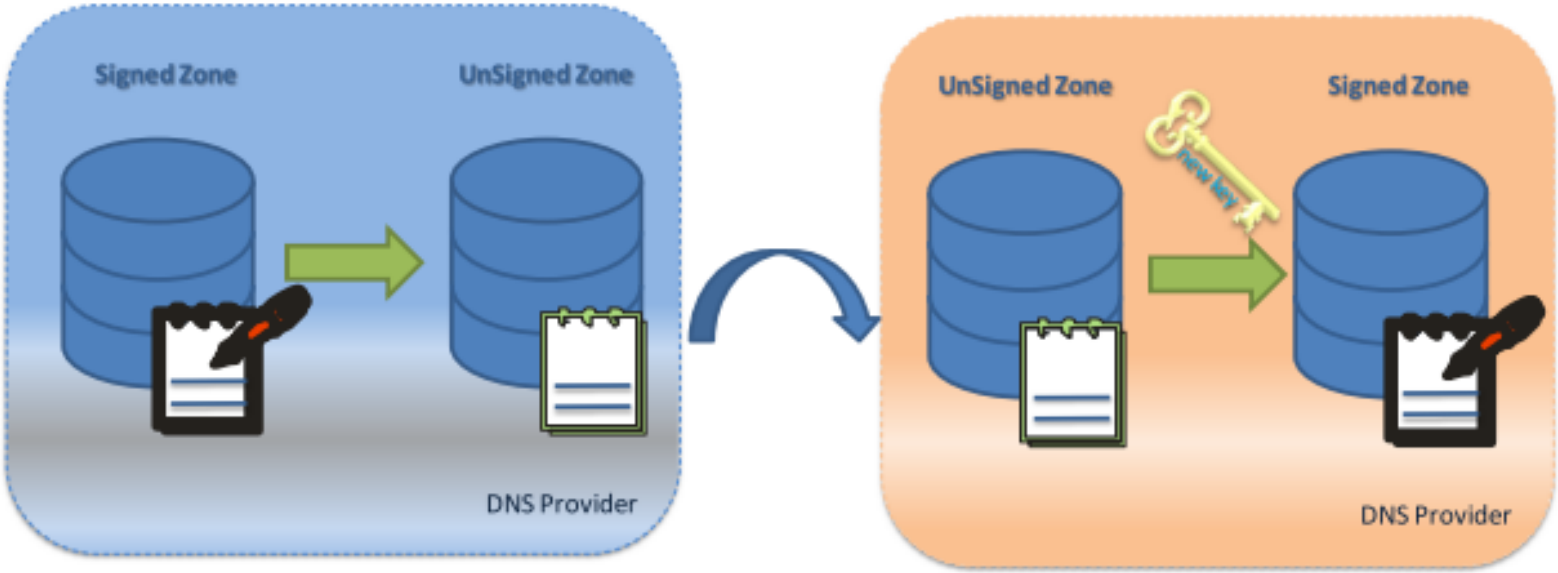


# Change of registrar

- Customer wants to change everything
  - DNSSEC renders changes as BOGUS
  - Lock in scenario, lot of “after contract” work
- Old and new operators need to cooperate
  - Old one prepares zone or parent for transfer
    New one removes old references later
  - Old one keeps name server longer in old state
    New one does the change on the registry and changes name servers weeks after transfer

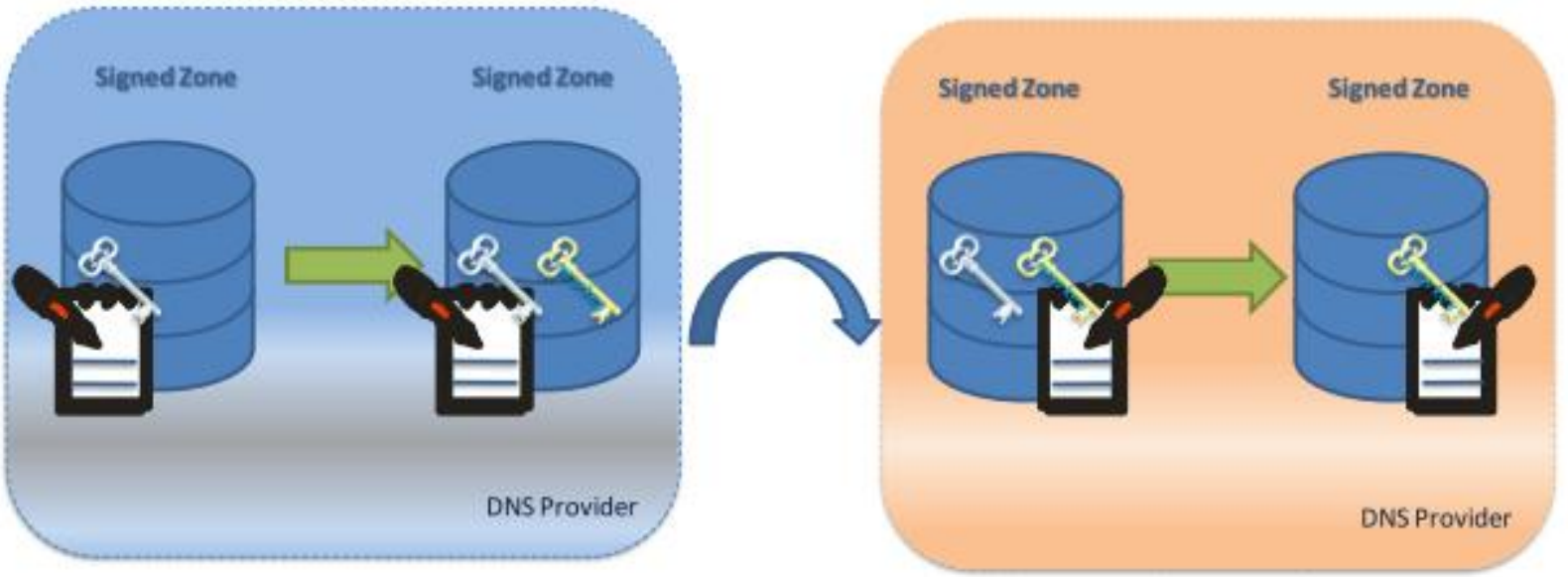
IKS service

# Insecure transfer




Registries require every registrar to be able to remove DNSSEC at least.

# Secure transfer




Registrars need to cooperate before and after transfer.

# Transfer considerations

- Caches have old and new information
  - All keys needs to be known for a long time
  - Put new keys in old zone long before transfer
  - Keep old keys in new zone long after transfer
- Change DS once or twice?
  - Prepublished KSKs: change DS on transfer
  - Otherwise add new DS long before transfer
    Keep old DS long after transfer
- Without cooperation, zone will *fail*
  - Old operator can only limit TTLs before transfer
  - On failure, customers tend to sue you for old keys

# The last mile

- In an ideal world, apps use a new API
  - Error messages might become helpful
  - Validation errors are SERVFAIL
- Resolver offloading
  - Provide validated data with AD
  - Allow validator chaining with CD
  - Question: Provide bogus data at all?
- Attacks on the last mile even for LEAs

# Finally gain benefits

- *DNSSEC adds trust to DNS*
- DNS as a hierarchical distributed DB
- Manage your SSHFPs centrally
- Manage your CERTs distributed
- Manage your OpenPGP keys distributed
- Do not deliver poisoned data to clients
- Validate late, validate centrally

IKS service

# Real wold usage

- Microsoft does it
  - AD can insist on DNSSEC
  - Uses IPSec for last mile security
  - Derived IPSec from DNSSEC
- DNSSEC based IPSec = VPN
  - DirectAccess: IPv6 and DNSSEC for setup
  - Build automatic VPNs using DNS policy
- Use case: Thousands of offices worldwide

IKS service

# Generalize Benefits

- Quick hack drawbacks:
  - A new DNS record per application?
  - How about same protocols on different ports?
- Generalized approach: DANE
  - _<port>._<proto>.<fqdn> TLSA <cert>
  - Can crypt any TCP stream with TLS
  - Surprise: Works nice with CNAMEs as well
  - Where does the trust come from?
    Can it replace CAs?

IKS service

# Did you sign your zones?

Why not?