DNS security

Karst Koymans & Niels Sijm

Informatics Institute

University of Amsterdam

Tuesday, September 18, 2012

1. Chain of Trust
   - How the Chain of Trust works
   - Details of RR's used in the chain of trust
2. Denial of existence
   - Concept and Resource Records
   - Details of RR's used in denial of existence
3. Resolver and protocol issues

Outline

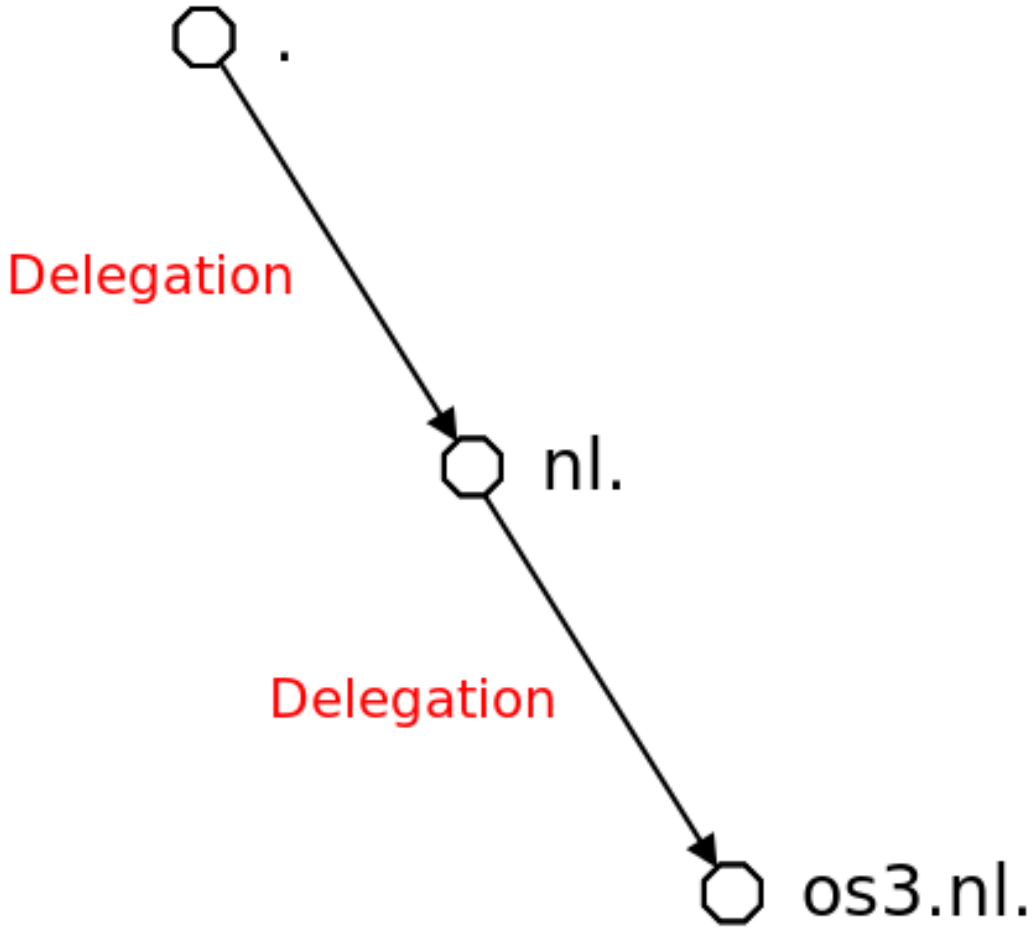
Outline

Tree walking algorithm

Validating a RRset for www.os3.nl:

- Start at the root
- Verify authenticity of delegation to nl. zone
- Verify authenticity of delegation to os3.nl. zone
- Verify authenticity of RRset of www.os3.nl

Tree walking algorithm

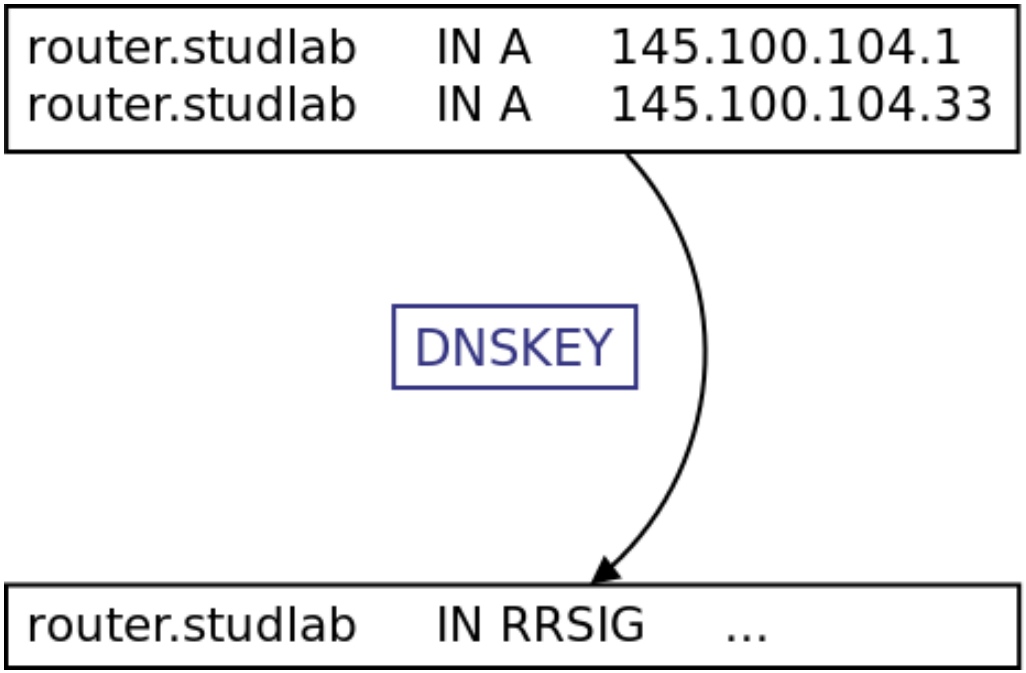


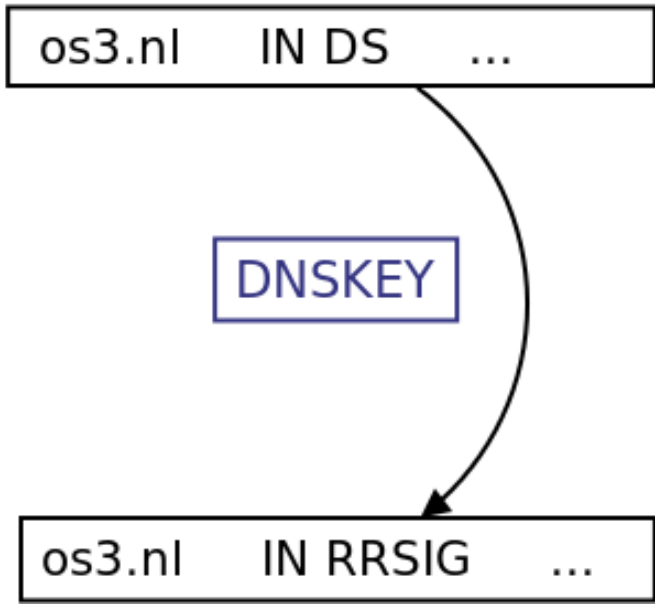
Tree walking algorithm

Resource Records used to build the Chain of Trust:

- RRSIG
  - Resource Record Signature
  - Contains signatures of RRsets
- DNSKEY
  - Stores public key of a zone
  - Used to verify signatures
- DS
  - Delegation Signer
  - Contains a hash of the DNSKEY of a child zone

Tree walking algorithm





Tree walking algorithm

Validating delegation from root to nl. zone:

1. Keys of root zone are known to resolver
2. Ask root zone for DS of nl. zone
3. Ask nl. zone for DNSKEY of nl. zone
4. Verify that DS is valid hash of DNSKEY

Tree walking algorithm

Validating delegation from root to nl. zone:

1. Keys of root zone are known to resolver
2. Ask root zone for DS of nl. zone
3. Ask nl. zone for DNSKEY of nl. zone
4. Verify that DS is valid hash of DNSKEY

Two steps are missing...

Tree walking algorithm

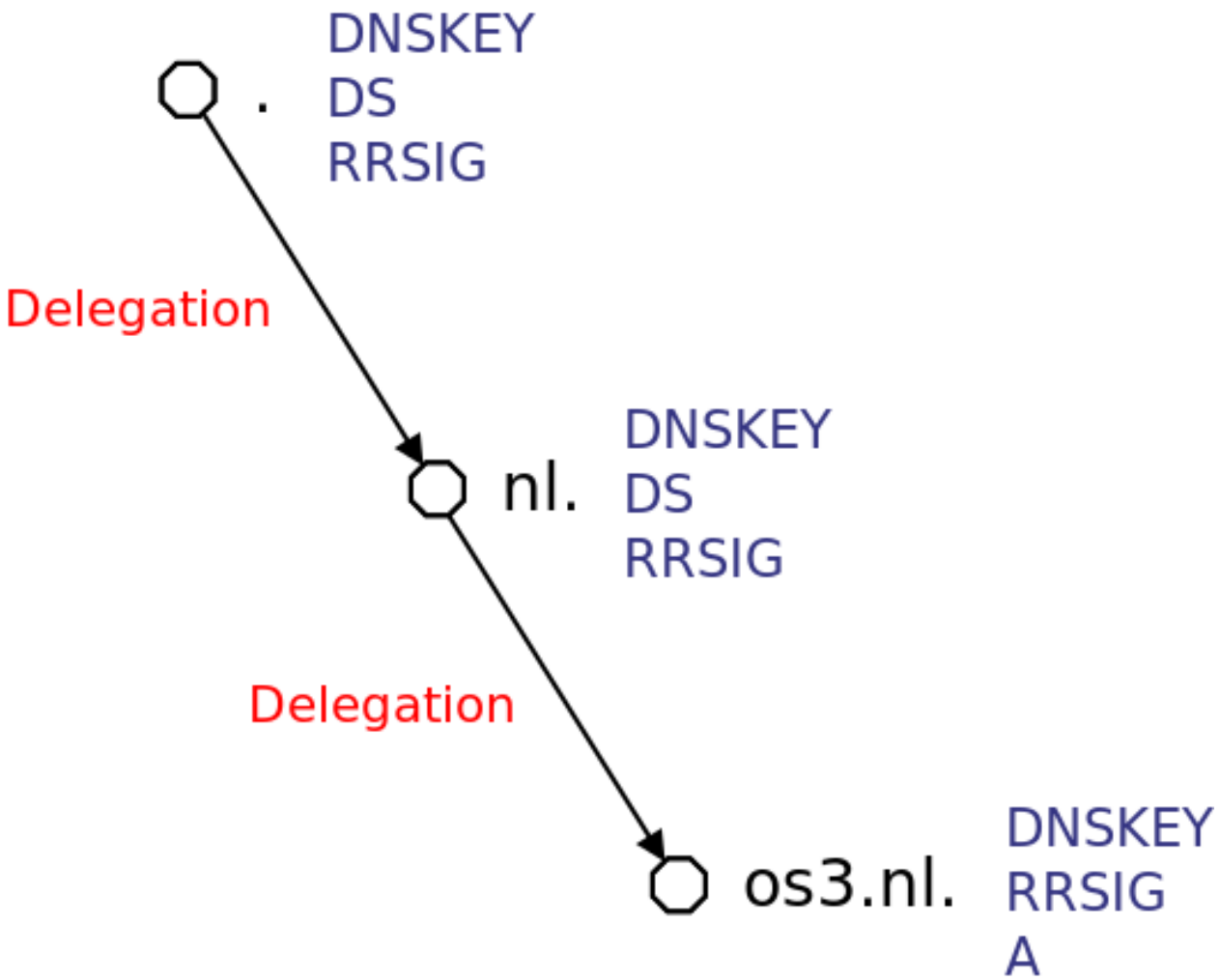
Validating delegation from root to nl. zone:

1. Keys of root zone are known to resolver
2. Ask root zone for DS of nl. zone
3. Ask nl. zone for DNSKEY of nl. zone
4. Verify that DS contains a valid hash of DNSKEY

Two steps are missing...

5. Retrieve RRSIG for root zone, and
6. verify authenticity of DS

Tree walking algorithm



Tree walking algorithm

Validating delegation from root to nl. zone:

1. Retrieve DNSKEY for root zone
   - Stored in resolver as secure starting point
   - Can also be retrieved from root nameservers:
     - `dig . DNSKEY`
2. Ask root zone for DS of nl. zone
   - Contains a hash of the DNSKEY of nl. zone
   - Stored in root zone
     - `dig nl DS`

Tree walking algorithm

Validating delegation from root to nl. zone:

3. Ask nl. zone for DNSKEY of nl. zone
   - `dig nl DNSKEY`
4. Verify that DS contains a valid hash of DNSKEY
   - Create hash of DNSKEY, and
   - compare to hash in DS record

Tree walking algorithm

Validating delegation from root to nl. zone:

5. Retrieve RRSIG for root zone
   - Stored in root zone
   - `dig . RRSIG`
6. Use DNSKEY to verify signature in RRSIG matches data in DS record

DNSSEC-bis chain of trust (1)

- To validate a resource record set RRset
  - Validate RRSIG(RRset)
    - by using a ZSK (zone signing key) from the DNSKEY RRset
  - Validate RRSIG(DNSKEYset)
    - by using a KSK (key signing key) from that same DNSKEY RRset
  - Validate the KSK
    - by using a DS (present in the parent zone)

      which contains a hash of the KSK
    - the KSK used is called a SEP (Secure Entry Point)

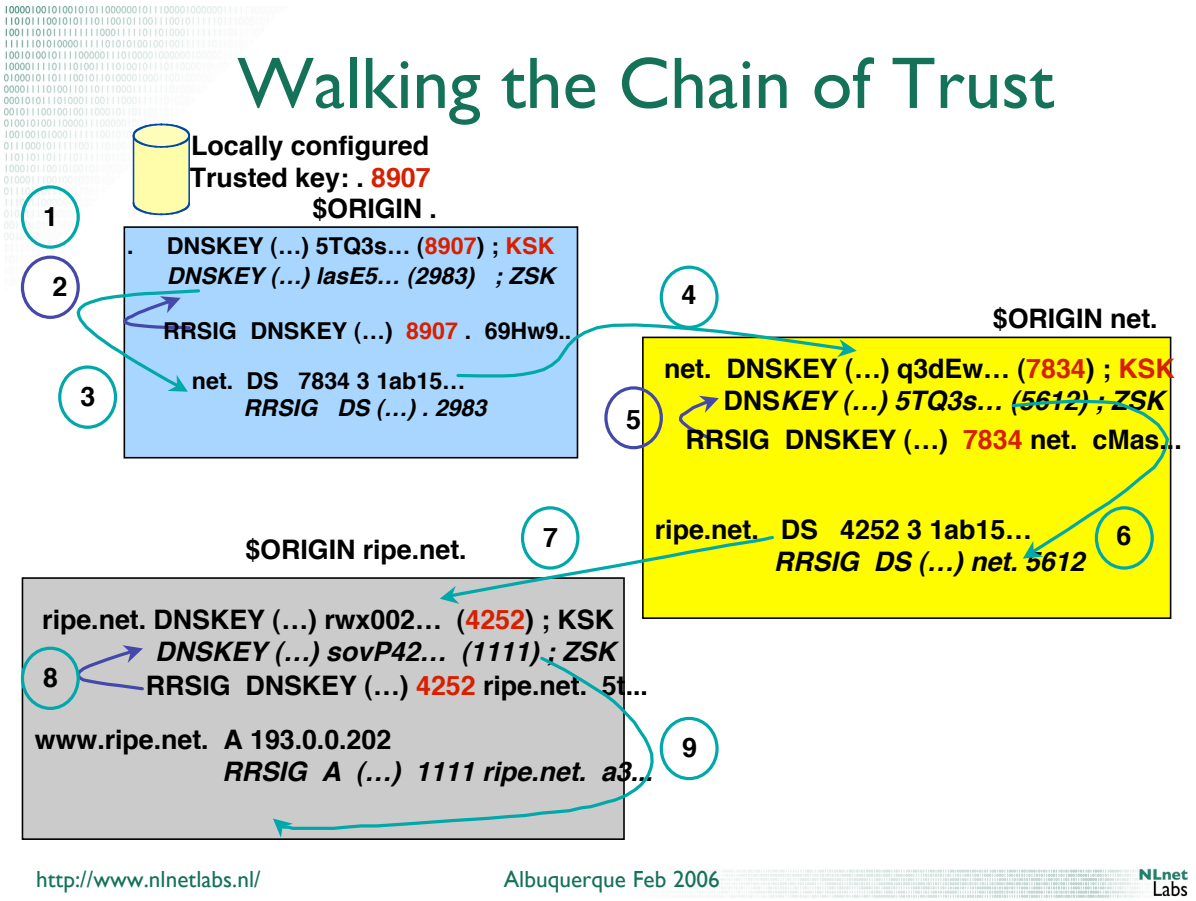
DNSSEC-bis chain of trust (2)

- Continue validating one level higher in the hierarchy
  - Use DS as RRset and iterate
- Use trusted anchors for DNSKEY's or DS's to terminate
  - for instance for checking root zone keys
- Authority around a cut is now as follows
  - NS and DNSKEY are authoritative on

    the child side of the cut (zone apex)
  - DS is authoritative on

    the parent side of the cut (delegation point)

Chain of Trust illustration



(Source: Olaf Kolkman, RIPE NCC, NLnet Labs)

Outline

RRSIG record example from RFC 4034

```
host.example.com. 86400 IN RRSIG A 5 3 86400 20030322173103 (
                                    20030220173103 2642 example.com.
                                    oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr
                                    PYGv07h108dUKGMeDPKijVCHX3DDKdfb+v6o
                                    B9wfuh3DTJXUAfI/M0zm0/zz8bW0Rznl803t
                                    GNazPwQKkRN20XPXV6nwwfoXmJQbsLNrLfkG
                                    J5D6fwFm8nN+6pBzeDQfsS3Ap3o= )
```

RRSIG record content

Part	Meaning
Type covered	Record type this RRSIG is about
Algorithm	Signature algorithm used; 5 is RSA/SHA-1
Labels	Number of labels of owner (without root)
TTL	Original time to live
Expiration and Inception	Signature validity date bounds
Key Tag	To help find (not identify) the signing key
Signer's Name	Owner name of zone and key to use
Signature	Signature (in Base64)

DNSKEY record example from RFC 4034

```
example.com. 86400 IN DNSKEY 256 3 5 ( AQPSKmynfzW4kyBv015MUG2DeIQ3
                                       Cbl+BBZH4b/0PY1kxkmvHjcZc8no
                                       kfzj31GajIQKY+5CptLr3buXA10h
                                       WqTkF7H6RfoRqXQeogmMHfpftf6z
                                       Mv1LyBUgia7za6ZEzOJBOztyvhjL
                                       742iU/TpPSEDhm2SNKLijfUppn1U
                                       aNvv4w== )
```

DNSKEY record content

Part	Meaning
Flags	Zone key (KSK+ZSK); Secure entry point (KSK)
Protocol	Always 3 (for backward compatibility with KEY RR)
Algorithm	Signature algorithm used; 5 is RSA/SHA-1
Public key	Key used for signing (in Base64)

DS record example from RFC 4034 and 4509

```
dskey.example.com. 86400 IN DNSKEY 256 3 5 ( AQOeiiR0GOMYkDshWoSKz9Xz
                                             fwJr1AYtsmx3TGkJaNXVbfi/
                                             2pHm822aJ5iI9BMzNXxeYCmZ
                                             DRD99WYwYqUSdjMmmAphXdvx
                                             egXd/M5+X7OrzKBaMbCVdFLU
                                             Uh6DhweJBjEVv5f2wwjM9Xzc
                                             nOf+EPbtG9DMBmADjFDc2w/r
                                             ljwvFw==
                                           ) ;  key id = 60485

dskey.example.com. 86400 IN DS 60485 5 1 (
               2BB183AF5F22588179A53B0A98631FAD1A292118 ) ; SHA-1

dskey.example.com. 86400 IN DS 60485 5 2 (
               D4B7D520E7BB5F0F67674A0C
               CEB1E3E0614B93C4F9E99B83
               83F6A1E4469DA50A ) ; SHA-256
```

DS record content

Part	Meaning
Key Tag	To help find (not identify) the signing key
Algorithm	Signature algorithm of the signing key
Digest Type	Hashing algorithm used; 1 is SHA-1, 2 is SHA-256
Digest	Sequence of case-insensitive hexadecimal digits

Outline

Outline

Positive and negative answers

- Proving existence (and authenticity) is elaborate, but "easy"
  - It uses standard public key cryptography signing mechanisms
- Proving non-existence is much more difficult
  - There is nothing to sign
  - Empty non-terminals
  - Wildcards

Methods for denial of existence

- Doing a zone transfer
  - Only possible for trusted clients
- On the fly (realtime) signing of negative replies
  - Often too compute intensive
- Enumerating the zone
  - Makes "zone walking" possible
    - Unless the real names are hidden somehow

More DNSSEC-bis resource records

- NSEC: authenticated denial of existence
  - Uses canonical ordering to list domain names in a zone
    - Lexicographic ordering for labels by considering them as "unsigned left-justified octet strings"
    - Lexicographic ordering for domain names as sequence of labels, in fact walking the domain name tree in preorder traversal
- NSEC3 introduced later to frustrate zone enumeration
  - All domain names are disguised as hashes
  - One can still count the number of domain names

Outline

NSEC record example from RFC 4034

```
alfa.example.com. 86400 IN NSEC host.example.com. (
                                A MX RRSIG NSEC TYPE1234 )
```

NSEC record content

Part	Meaning
Next Domain Name	Defined by a standard ordering
Type Bit Maps	Types present at the current owner

- This mechanism can be used to enumerate a complete zone...
  - ...which is why certain organisations refuse to implement DNSSEC-bis
- NSEC chains wrap around and fork or join

  at a delegation point or zone apex
- NSEC3 RR's can anonymise the NSEC records

NSEC3 record example from RFC 5155

```
q04jkcevqvmu85r014c7dkba38o0ji5r.example. NSEC3 1 1 12 aabbccdd (
                        r53bq7cc2uvmubfu5ocmm6pers9tk9en A RRSIG )
```

NSEC3 record content

Part	Meaning
Hash Algorithm	1 for SHA-1
Flags	Opt-out for skipping unsigned delegations
Iterations	Number of times hash is calculated
Salt Length	0-255 (not represented in text format)
Salt[1]	Prevents dictionary attacks
Hash Length	1-255 (not represented in text format)
Next Hashed Owner Name	(binary data)
Type Bit Maps	Types present at the current owner

[1] Hash Algorithm, Flags, Iterations and Salt are communicated to slave servers via the NSEC3PARAM Resource Record in the zone apex.

Outline

Protocol flags

- DO (DNSSEC OK)
  - Indicate that you want all DNSSEC-related information like RRSIG and NSEC(3) for normal queries, if available
  - Presupposes use of EDNS0
- CD (Checking Disabled)
  - From resolver to (recursive) name server
  - Indicate that you want to do checking yourself
  - Also insecure (invalid) data is passed along
- AD (Authentic Data)
  - From recursive name server to resolver
  - Data has been verified successfully
  - In most cases not set by authoritative server
    - Checking is done at the resolver side

Security status of RR's

- Secure
  - Chain of trust exists and signature is valid
- Insecure
  - Chain of trust exists and signature is invalid
  - Results in SERVFAIL
- Bogus
  - Chain of trust should exist but is broken
  - Results in SERVFAIL
- Indeterminate
  - Chain of trust does not exist