Deploying DNSSEC

Part II DNSSEC Mechanisms and deployment

AFRINIC-17

aalain@afrinic.net
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 is 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
DNSSEC Mechanisms

• New Resource Records
• Setting Up a Secure Zone
• Delegating Signing Authority
• DNSSEC Deployment Rollovers
New Resource Records
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:

nlnetlabs.nl. 3600 IN DNSKEY (AQOvhvXXU61Pr8sCwELcqqq1g4JJ CALG4C9EtraBKvD +vGIF/unwigfLOA Q3nHp/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
- 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 authoritative 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

$ORIGIN nlnetlabs.nl.
lab.nlnetlabs.nl. 3600 IN NS ns.lab.nlnetlabs.nl
lab.nlnetlabs.nl. 3600 IN DS

3112 5 1

239af98b923c023371b521
g23b92da12f42162b1a9
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 problem for some entities

• NSEC3
  – All RR names hashed
  – Hashed names are ordered
  – “opt-out” for unsecured delegations possibilities
Delegating Signing Authority

Chains of Trust
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 child’s zone
    • for DNSKEY & RRSIGs
  – Verify if key can be trusted
  – Generate DS RR
Walking the Chain of Trust

1. $ORIGIN .
   - DNSKEY (...) 5TQ3s... (8907) ; KSK
   - DNSKEY (...) lasE5... (2983) ; ZSK
   - RRSIG DNSKEY (...) 8907 . 69Hw9...
   - net. DS 7834 3 1ab15...
   - RRSIG DS (...) . 2983

2. RRSIG DNSKEY (...) 8907 . 69Hw9...
3. net. DS 7834 3 1ab15...
4. RRSIG DS (...) . 2983

5. $ORIGIN net.
   - net. DNSKEY (...) q3dEw... (7834) ; KSK
   - DNSKEY (...) 5TQ3s... (5612) ; ZSK
   - RRSIG DNSKEY (...) 7834 net. cMas...

6. RRSIG DNSKEY (...) 7834 net. cMas...
7. $ORIGIN foo.net.
   - foo.net. DS 4252 3 1ab15...
   - RRSIG DS (...) net. 5612

8. foo.net. DNSKEY (...) rwx002... (4252) ; KSK
   - DNSKEY (...) sovP42... (1111) ; ZSK
   - RRSIG DNSKEY (...) 4252 foo.net. 5t...

   - RRSIG A (...) 1111 foo.net. a3...
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
DNSSEC DEPLOYMENT
DNSSEC Deployment Tasks

• Key maintenance policies and tools
  - Private key use and protection
  - Public key distribution
• Zone signing and integration into the provisioning chain
• DNS server infrastructure
• Secure delegation registry changes
  - Interfacing with customers
DNSSEC Architecture modification

- Zone Creation
- Zone signer
- Primary DNS
- Secondary DNS
- DNSSEC aware servers
- Provisioning DB
- DNS and input checks
- DNSSEC aware provisioning
- Customer interfaces
Key Maintenance

• DNSSEC is based on public key cryptography
  – Data is signed using a private key
  – It is validated using a public key

Operational problems:

• Dissemination of the public key
• Private key has a ‘best before’ date
  – Keys change, and the change has to disseminate
This document presents a framework to assist writers of DNSSEC Policy and Practice Statements such as Domain Managers and Zone Operators on both the top-level and secondary level, who is managing and operating a DNS zone with Security Extensions (DNSSEC) implemented.

In particular, the framework provides a comprehensive list of topics that should be considered for inclusion into a DNSSEC Policy definition and Practice Statement.

- **ICANN DPS for root zone**
Public Key Dissemination

• In theory only one trust-anchor needed that of the root
  – How does the root key get to the end user?
  – How is it rolled?

• In absence of hierarchy, there will be many trust-anchors
  – How do these get to the end-users?
  – How are these rolled?

• These are open questions, making early deployment difficult.

• DLV registries (https://secure.isc.org/index.pl?/ops/dlv/)
Key Management

• There are many keys to maintain
  - Keys are used on a per zone basis
    • Key Signing Keys and Zone Signing Keys
  - During key rollovers there are multiple keys
    • In order to maintain consistency with cached DNS data
    • RFC4641

• Private keys need shielding
Private Key Maintenance
Basic Architecture

- Zone DB
- DNS server
- Key DB and Signer server
- Signer client
- KEY Master
- Key maintainer
Maintaining Keys and Signing Zones

• The KeyDB maintains the private keys
  – It ‘knows’ rollover scenarios
  – UI that can create, delete, roll keys without access to the key material
  – Physically secured

• The signer ties the Key DB to a zone
  – Inserts the appropriate DNSKEYs
  – Signs the the zone with appropriate keys

• Strong authentication
Infrastructure

• One needs primary and secondary servers to be DNSSEC protocol aware

• We have concerns about Firewalls/IDS/IPS on DNS packet size and EDNS0
  – http://www.icann.org/committees/security/sac016.htm

• We had a number of concerns about memory, CPU, network load
  – Research done at RIPE-NCC and published as RIPE 352
Infrastructure

• Bandwidth increase is caused by many factors
  – Hard to predict but fraction of DO bits in the queries is an important factor

• CPU impact is small, Memory impact can be calculated

• Don’t add DNSKEY RR set in additional
In the DNS the parent signs the “Delegations Signer” RR
- A pointer to the next key in the chain of trust

DNSKEY or DS RR needs to be exchanged between parent and child
Underlying Ideas

• The DS exchange is the same process as the NS exchange
  – Same authentication/authorization model
  – Same vulnerabilities
  – More sensitive to mistakes

• Integrate the key exchange into existing interfaces
  – Customers are used to these

• Include checks on configuration errors
  – DNSSEC is picky

• Provide tools
  – To prevent errors and guide customers
Key Rollover
DNSKEY in flavours

- Zone Signin Key (ZSK)
- Key Signing Key (KSK)
  - Functions as secure entry point into the zone
    - Trust-anchor configuration
    - Parental DS points to it
    - Interaction with 3rd party
- DNSKEYs are treated all the same in the protocol
- Operators can make a distinction
  - Look at the flag field: ODD (257 in practice) means SEP
Benefits of using separate keys

• Rolling KSK needs interaction, rolling ZSKs can be done almost instantaneously

• Remember KSK replacement may result in
  – Trust-anchor updates
  – Change of DS record at parent

• Allows different responsibilities
  – ZSKs may be touched day to day by junior staff
  – KSKs may only be touched by senior staff
Rolling keys instantaneously?

• Remember that in the DNS caches are at play.
  – It takes a bit of time to have new information propagate
• When you happen to get new data you would like to be able to use RRSIGs from the cache
• When you happen to get old data from the cache you would like to use new RRSIGs
• Try to make sure both old and new keys are available
• Or, try to make sure both old and new sigs are available
Timing Properties

- **Authoritative Master**: Foo TXT Old to Foo TXT New
- **Authoritative Slave**: New data synchronization
- **Caching Nameserver**: Publication of new data, Query to slave, Zone transfer, Expiration From Cache

Timeline:
- $t_0$: Publication of new data
- $t_1$: Query to slave followed by Caching
- $t_2$: Zone transfer
- $t_3$: Expiration From Cache
PRE-publish ZSK rollover

- Introduce the new DNSKEY before you start using it to sign the data.
  - ‘passive and active’ key
  - The passive key is just published, the active key is used for signing

- You could also create two signatures after introducing the key, but that would cause your zone file to grow
ZSK rollover (pre-publish)

\[
dnssec-signzone -k ksk example.com zsk1
\]

Create passive zsk2

\[
dnssec-signzone -k ksk example.com zsk2
\]

At least TTL DNSKEY RRs
# ZSK rollover (pre-publish)

<table>
<thead>
<tr>
<th>Initial</th>
<th>new DNSKEY</th>
<th>New RRSIGs</th>
<th>DNSKEY removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOA0</td>
<td>SOA1</td>
<td>SOA2</td>
<td>SOA3</td>
</tr>
<tr>
<td>RRSIG10 (SOA0)</td>
<td>RRSIG10 (SOA1)</td>
<td>RRSIG11 (SOA2)</td>
<td>RRSIG11 (SOA3)</td>
</tr>
<tr>
<td>DNSKEY 1</td>
<td>DNSKEY 1</td>
<td>DNSKEY 1</td>
<td>DNSKEY 1</td>
</tr>
<tr>
<td>DNSKEY 10</td>
<td>DNSKEY 10</td>
<td>DNSKEY 10</td>
<td>DNSKEY 11</td>
</tr>
<tr>
<td>DNSKEY 11</td>
<td>DNSKEY 11</td>
<td>DNSKEY 11</td>
<td></td>
</tr>
<tr>
<td>RRSIG1 (DNSKEY)</td>
<td>RRSIG1 (DNSKEY)</td>
<td>RRSIG1 (DNSKEY)</td>
<td>RRSIG1 (DNSKEY)</td>
</tr>
<tr>
<td>RRSIG10 (DNSKEY)</td>
<td>RRSIG10 (DNSKEY)</td>
<td>RRSIG11 (DNSKEY)</td>
<td>RRSIG11 (DNSKEY)</td>
</tr>
</tbody>
</table>
### ZSK rollover(double signature)

<table>
<thead>
<tr>
<th></th>
<th>SOA0</th>
<th>SOA1</th>
<th>SOA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRSIG10 (SOA0)</td>
<td>RRSIG10 (SOA1)</td>
<td>RRSIG11 (SOA2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RRSIG11 (SOA1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNSKEY1</td>
<td>DNSKEY1</td>
<td>DNSKEY1</td>
<td>DNSKEY11</td>
</tr>
<tr>
<td>DNSKEY10</td>
<td>DNSKEY10</td>
<td></td>
<td>DNSKEY11</td>
</tr>
<tr>
<td></td>
<td>DNSKEY11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RRSIG1 (DNSKEY)</td>
<td>RRSIG1 (DNSKEY)</td>
<td>RRSIG1 (DNSKEY)</td>
<td>RRSIG1 (DNSKEY)</td>
</tr>
<tr>
<td>RRSIG10 (DNSKEY)</td>
<td>RRSIG10 (DNSKEY)</td>
<td>RRSIG11 (DNSKEY)</td>
<td>RRSIG11 (DNSKEY)</td>
</tr>
<tr>
<td></td>
<td>RRSIG11 (DNSKEY)</td>
<td>RRSIG11 (DNSKEY)</td>
<td></td>
</tr>
</tbody>
</table>
KSK rollover

• You are dependent on your parent.
  – You cannot control when the parent changes the DS RR
• Use the old KSK until the old DNS had time to expire from caches
• Double signature or pre-publish rollover
KSK rollover

Create ksk2 and send to parent

Remove ksk1

At least TTL DS RRs
# KSK rollover

<table>
<thead>
<tr>
<th>Initial</th>
<th>New DNSKEY</th>
<th>DS Change</th>
<th>DNSKEY removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOA0</td>
<td>SOA0</td>
<td>SOA1</td>
<td>SOA1</td>
</tr>
<tr>
<td>RRSIGpar(SOA0)</td>
<td>RRSIGpar(SOA0)</td>
<td>RRSIGpar(SOA1)</td>
<td>RRSIGpar(SOA1)</td>
</tr>
<tr>
<td>DS1</td>
<td>DS1</td>
<td>DS2</td>
<td>DS2</td>
</tr>
<tr>
<td>RRSIGpar(DS)</td>
<td>RRSIGpar(DS)</td>
<td>RRSIGpar(DS)</td>
<td>RRSIGpar(DS)</td>
</tr>
</tbody>
</table>

**Parent:**

<table>
<thead>
<tr>
<th>SOA0</th>
<th>SOA1</th>
<th>SOA1</th>
<th>SOA1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRSIGpar(SOA0)</td>
<td>RRSIGpar(SOA0)</td>
<td>RRSIGpar(SOA1)</td>
<td>RRSIGpar(SOA1)</td>
</tr>
<tr>
<td>DS1</td>
<td>DS1</td>
<td>DS2</td>
<td>DS2</td>
</tr>
<tr>
<td>RRSIGpar(DS)</td>
<td>RRSIGpar(DS)</td>
<td>RRSIGpar(DS)</td>
<td>RRSIGpar(DS)</td>
</tr>
</tbody>
</table>

**Child:**

<table>
<thead>
<tr>
<th>SOA0</th>
<th>SOA1</th>
<th>SOA1</th>
<th>SOA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRSIG10(SOA0)</td>
<td>RRSIG10(SOA1)</td>
<td>RRSIG10(SOA1)</td>
<td>RRSIG10(SOA2)</td>
</tr>
<tr>
<td>DNSKEY1</td>
<td>DNSKEY1</td>
<td>DNSKEY1</td>
<td>DNSKEY2</td>
</tr>
<tr>
<td>DNSKEY2</td>
<td>DNSKEY2</td>
<td>DNSKEY2</td>
<td>DNSKEY2</td>
</tr>
<tr>
<td>DNSKEY10</td>
<td>DNSKEY10</td>
<td>DNSKEY10</td>
<td>DNSKEY10</td>
</tr>
<tr>
<td>RRSIG1(DNSKEY)</td>
<td>RRSIG1(DNSKEY)</td>
<td>RRSIG1(DNSKEY)</td>
<td>RRSIG2(DNSKEY)</td>
</tr>
<tr>
<td>RRSIG2(DNSKEY)</td>
<td>RRSIG2(DNSKEY)</td>
<td>RRSIG2(DNSKEY)</td>
<td>RRSIG2(DNSKEY)</td>
</tr>
<tr>
<td>RRSIG10(DNSKEY)</td>
<td>RRSIG10(DNSKEY)</td>
<td>RRSIG10(DNSKEY)</td>
<td>RRSIG10(DNSKEY)</td>
</tr>
</tbody>
</table>
Planning for emergency Keys rollovers

• A compromised Key can be used as long as a valid trust chain exists
  – As long as a signature over the compromised key in the trust chain is valid
  – As long as a parental DS points to the compromised key
  – As long as the key is anchored in resolvers and used as SEP

• Tradeoff between abuse of the compromised key and cached data validation

• Needs a documented procedure ready
KSK compromising

• The DS pointing to the key or the TA should be replaced as soon as possible

• Keep the chain of trust
  – Introduce a new KSK into the key set, keep the compromised key in the key set
  – Sign the key set with short validity period
    • Signature should expire shortly after the DS appears in parent zone and old DSes has expired from cache
  – Upload the DS for the new key to the parent
  – Follow the procedure for normal KSK rollover
  – Remove the compromised key and re-sign the key set to the normal validity period
KSK compromising

• Breaking the chain of trust

• Two methods to break the chain
  – By removing the key in child zone, re-sign key set and send the DS to the parent
    • Zone is bogus and attackers zone valid
  – By removing the DS from Parent zone and the key from child zone
    • Zone insecure

• If a TA is compromised
  – Resolvers should be notified
  – New key distributed and authenticated out-of-band
ZSK compromise

• No child/parent interactions needed
  – Zone should be re-signed with a new ZSK as soon as possible
  – Immediate disappearance of the compromise key can lead to validation problems
  – Until signature expired on the compromised key, the domain may still be at risks.
Questions ????