



OpenDNSSEC

Slides based on material from the OpenDNSSEC project team

OpenDNSSEC

What, Why, Who?

- OpenDNSSEC is a zone signer that automates the process of keeping track of DNSSEC keys and the signing of zones.

OpenDNSSEC

What, Why, Who?

- Many DNSSEC tools were lacking:
 - Good key management
 - Policy handling
 - Hardware acceleration
 - Etc.
- DNSSEC should be easy to deploy
- Increase the number of DNSSEC users
- Experience from previous DNSSEC operations

OpenDNSSEC

What, Why, Who?



OpenDNSSEC Goals

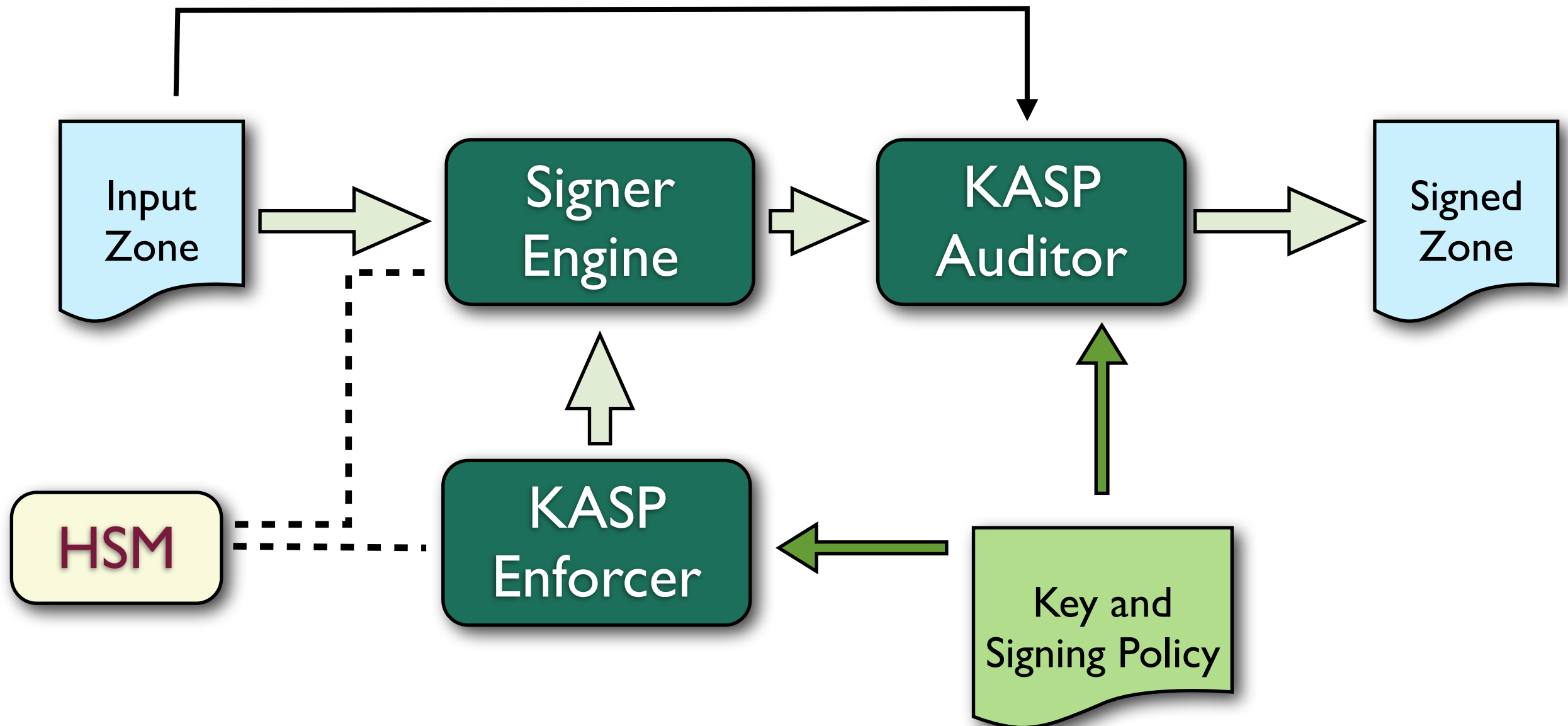
- Open source software with a BSD license
- Simple to integrate into existing infrastructure
- Key storage and hardware acceleration using PKCS#11

Bump-in-the-Wire

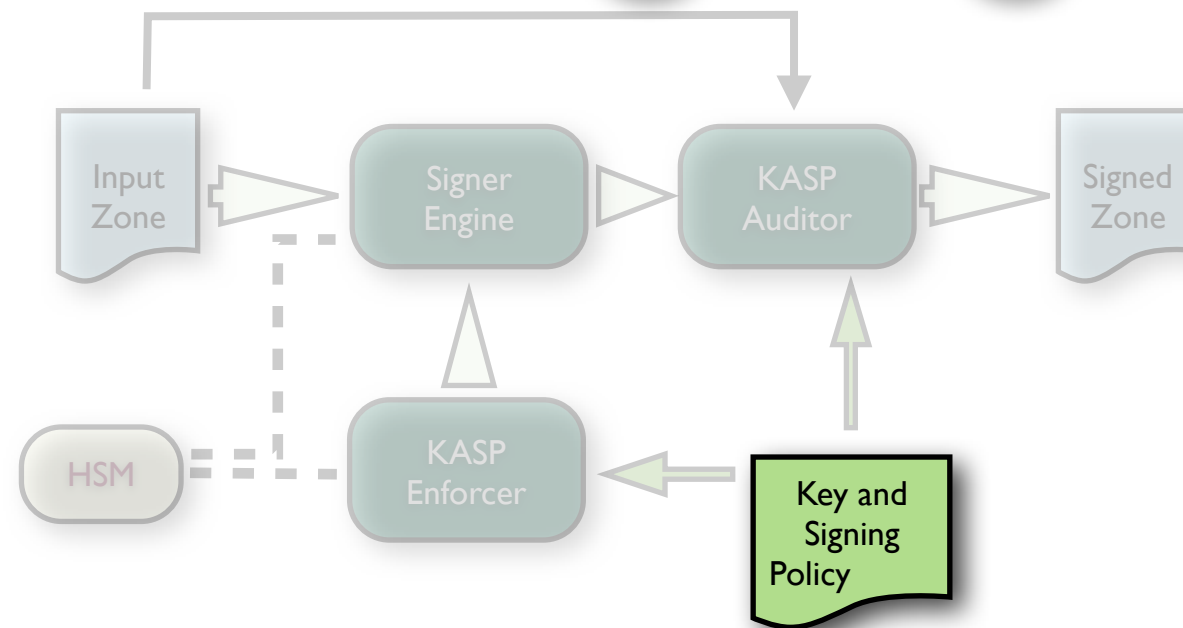
- In many cases, anticipate that OpenDNSSEC will be employed on a system between a hidden and public master.



Architecture

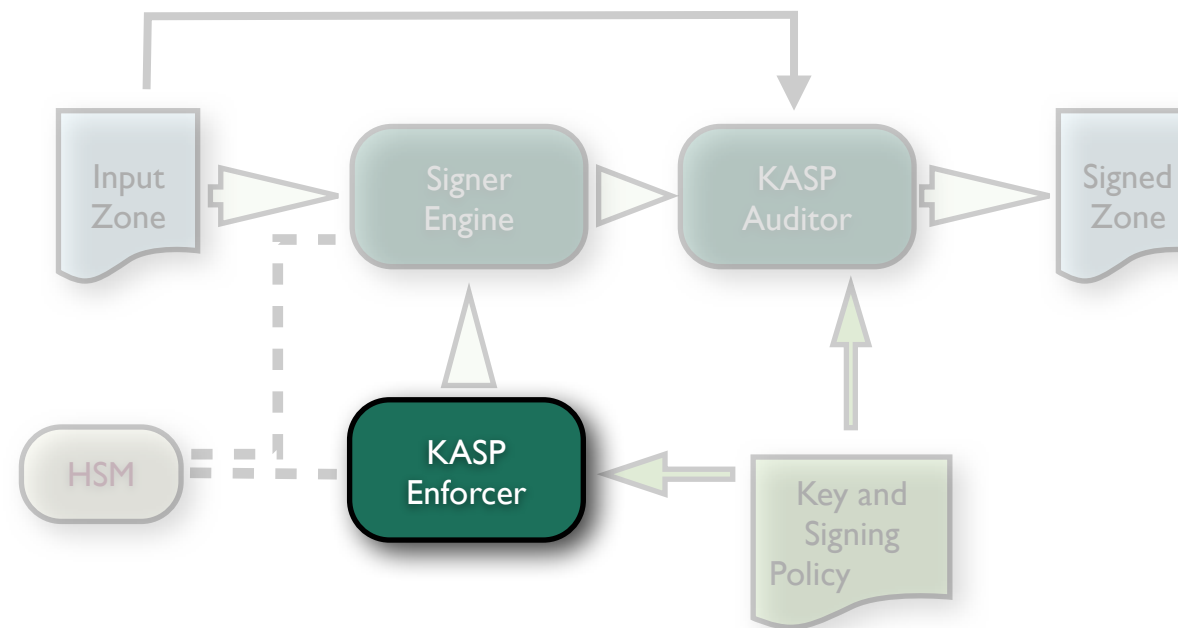


Key and Signing Policy



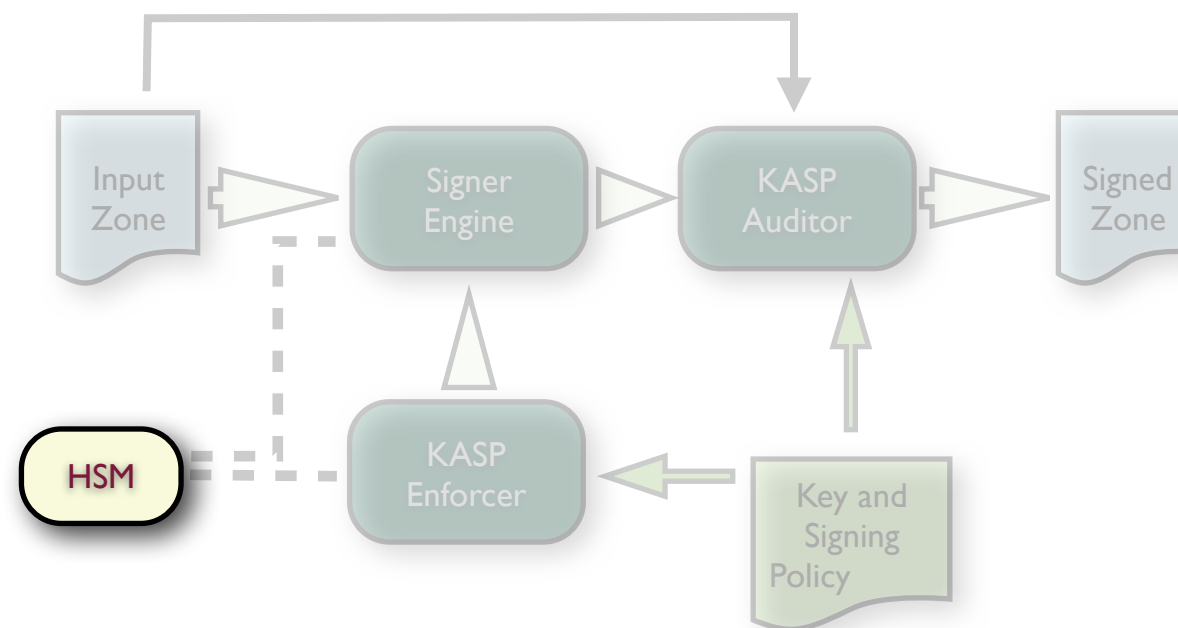
- How to sign a zone is described by a policy
- Allows choice of key strengths, algorithm, key and signature lifetimes, NSEC/NSEC3, etc.
- Can have anything between one policy for all zones to one policy per zone.

KASP Enforcer



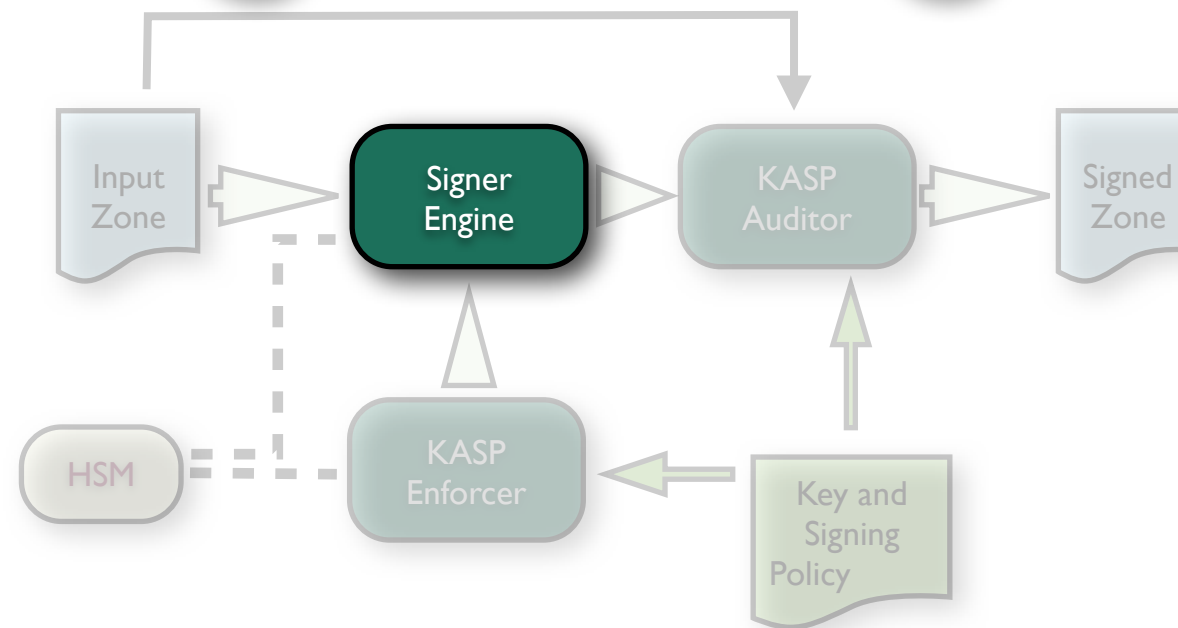
- Handles the management of keys:
 - Key creation using HSM
 - Key rolling
- Chooses the keys used to sign the zone.

HSM



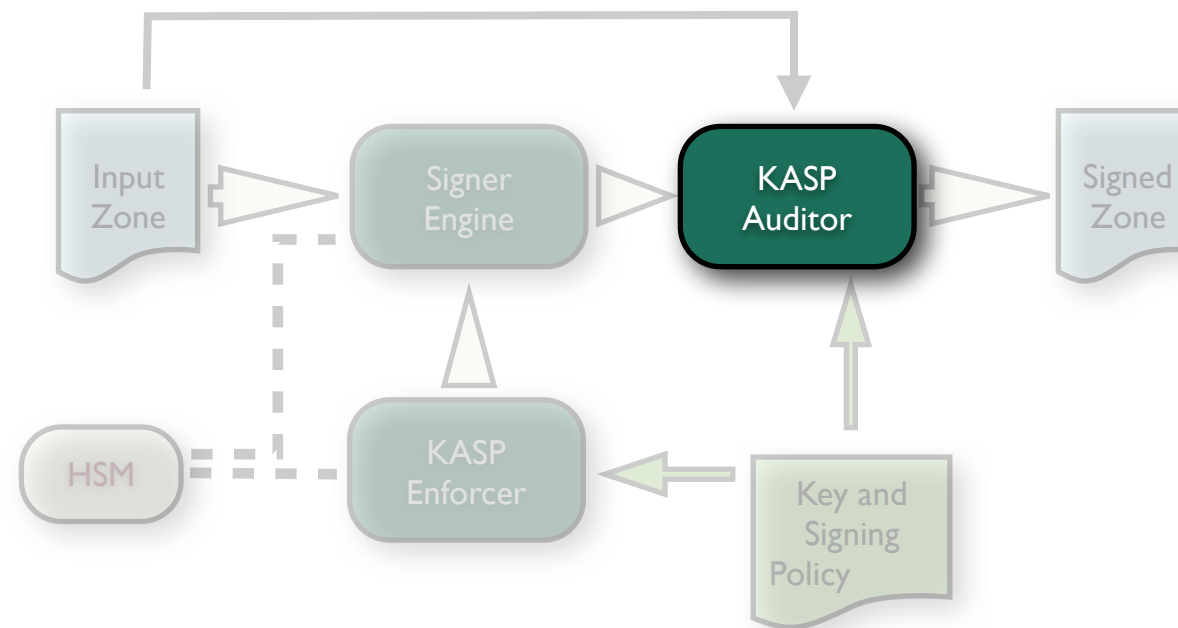
- Hardware Security Module
 - Stores the keys.
 - Hardware acceleration to sign records
- Standard interface via PKCS#11 API - abstracted within OpenDNSSEC into libhsm.
- SoftHSM available with OpenDNSSEC: software emulation of the HSM.

Signer Engine



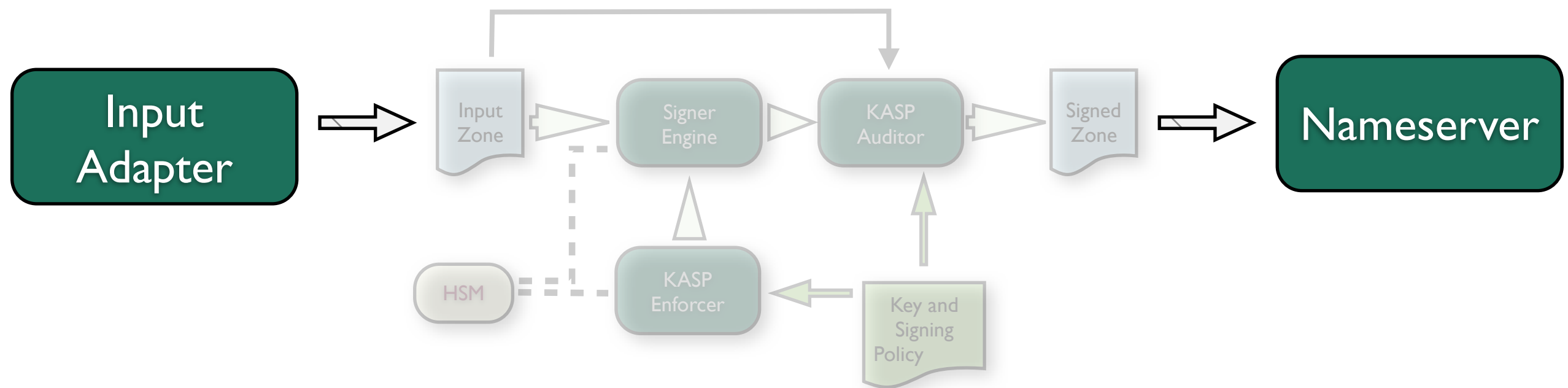
- Automatic signing of the zones
 - Can reuse signatures that are not too old
 - Can spread signature expiration time over time (jitter)
- Maintains NSEC/NSEC3 chain.
- Updates SOA serial number.

KASP Auditor



- Checks that the signer and enforcer work the way they are supposed to, e.g.
 - Non DNSSEC RRs are not added or removed
 - Policy is being followed
- Can stop zone distribution if needed.
- Written by a different person and in a different language (Ruby).

Input and Output Adapters



- Input adapter supplied as part of OpenDNSSEC - accepts AXFRs, responds to NOTIFYs.
- Output adapter not supplied - any preferred nameserver can be used (BIND, NSD, etc.)
- Can configure command to be used to reload zone.

Development

- Modular nature
 - Each participant could work separately
 - Agree the interfaces
- Regular phone conferences
- Code checking with Coverity Prevent
- Code reviews before releases

Testing

- Modular nature
 - Each module has unit tests
- Functional testing performed by SIDN
- Performance testing by SIDN and IIS

Running OpenDNSSEC

Read the Documentation

- <http://www.opendnssec.org/documentation/using-opendnssec/>
- OpenDNSSEC is powerful

Configuring OpenDNSSEC

- **conf.xml**
Used for overall configuration of the system
- **kasp.xml**
Defines the various policies for signing zones
- **zonelist.xml**
Zones that will be signed, tying them to a policy in kasp.xml
- **zonefetch.xml**
for transferring/fetching zones that are to be signed

P[n]Y[n]M[n]DT[n]H[n]M[n]S

- OpenDNSSEC is about durations (periods), not about absolute times.
- The format of periods is as above
 - PIDT12H is 1 day and 12 hours
 - No clue about Gregorian Calendar
 - P1M is considered 1 month (always 31 days)
 - P1Y is considered 1 year (always 365 days)

CONF.XML

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!-- $Id: conf.xml.in 1143 2009-06-24 12:10:40Z jakob $ -->
```

- Preamble... its what you get when you use XML

Configuration

- Configuration contains
 - RepositoryLists
 - Common
 - Enforcer
 - Signer
 - Auditor

```
<Configuration>
  <RepositoryList>
    ....
  <RepositoryList>
  <Common>
    ....
  </Common>
  <Enforcer>
    ....
  </Enforcer>
  <Signer>
    ....
  </Signer>
  <Auditor>
    ....
  </Auditor>
</Configuration>
```

Repository List

name, also used in
kasp.xml

```
<RepositoryList>
  <Repository name="SoftHSM">
    <Module>/usr/local/lib/libsofthsm.so</Module>
    <TokenLabel>OpenDNSSEC</TokenLabel>
    <PIN>1234</PIN>
  </Repository>
</RepositoryList>
```

- Defines where private keys live
 - You need at least one but you can have more
 - HSM interface available
 - SoftHSM if you do not have a board.
 - Each private key repository is listed as an <repository> element

Sideline: SoftHSM

The (Soft)HSM

- PKCS#11 is the industry API to HSMs
- Private keys are assumed to be stored in HSMs
- The SoftHSM is distributed separately; for those who do not have HSM (most users)
- HSMs can be tricky read the documentation about the <Capacity> and the <RequireBackup> elements
- <http://www.opensssec.org/documentation/hsm-buyers-guide/>

Using Soft HSM

- SOFTHSM_CONF or /etc/softhsm.conf
- You need a key: initialize one or import one

```
softhsm --init-token --slot 0 --label "My token 1"
```

or

```
softhsm --import key1.pem --slot 1 --label "My key" --id A1B2 --pin 123456
```

and you can convert your current bind keys:

```
softhsm-keyconv --topkcs8 --in Kexample.com.+007+05474.private --out rsa.pem
```


Back to CONF.XML

Common

```
<Common>
  <Logging>
    <Syslog><Facility>local0</Facility></Syslog>
  </Logging>

  <PolicyFile>/etc/opensnssec/kasp.xml</PolicyFile>
  <ZoneListFile>/etc/opensnssec/zonelist.xml</ZoneListFile>

  <!--
    <ZoneFetchFile>/etc/opensnssec/zonefetch.xml</ZoneFetchFile>
  -->
</Common>
```

- This element provides pointers to other configuration files and some settings shared by all components such as logging

Enforcer

```
<Enforcer>
  <!--
    <Privileges>
      <User>opendnssec</User>
      <Group>opendnssec</Group>
    </Privileges>
  -->

  <Datastore><SQLite>/var/opendnssec/kasp.db</SQLite></
Datastore>
  <Interval>PT3600S</Interval>
</Enforcer>
```

Users/Group under which
the process runs

- The Enforcer needs to keep state in a “Datastore” this can be an SQLite or MySQL database
- The Interval is the interval by which the enforcer will do ‘its thing’. That interval should be well below the key lifetimes defined in kasp.xml

Signer configuration

```
<Signer>
  <!--<Privileges><User>opendnssec</User><Group>opendnssec</Group>
    </Privileges>
  -->
  <WorkingDirectory>/var/opendnssec/tmp</WorkingDirectory>
  <WorkerThreads>8</WorkerThreads>
<!--
  <NotifyCommand>/usr/local/bin/my_nameserver_reload_command</NotifyCommand>
-->
</Signer>
```

- The Signer will need a place to put temporary files and may start multiple threads (its the component that does most of the work)
- After the signer is done you may want to kick your name server for a reload

Auditor configuration

```
<Auditor>  
  <!--<Privileges><User>opensnsec</User><Group>opensnsec</Group>  
    </Privileges>  
  -->  
    <WorkingDirectory>/var/opensnsec/tmp</WorkingDirectory>  
</Auditor>
```

- The Auditor will also need a place to put temporary files

KASP.conf

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!-- $Id: kasp.xml.in 1154 2009-06-24 13:02:25Z jakob $ -->
```

- **Key and Signature Policy** is documented in here

KASP

- KASP contain 1 or more Policies
- Policy contains
 - Description
 - Signatures
 - Denial
 - Keys
 - Zones
 - Parent
 - Audit

```
<KASP>
  <Policy>
    <Description>
      ....
    </Description>
    <Signatures>
      ...
    </Signatures>
    <Denial>
      ...
    </Denial>
    <Keys>
      ...
    </Keys>
    <Zones>
      ...
    </Zones>
    <Parent>
      ...
    </Parent>
    <Audit/>
  </Policy>
  <Policy>
    ....
  </Policy>
</KASP>
```

Signatures

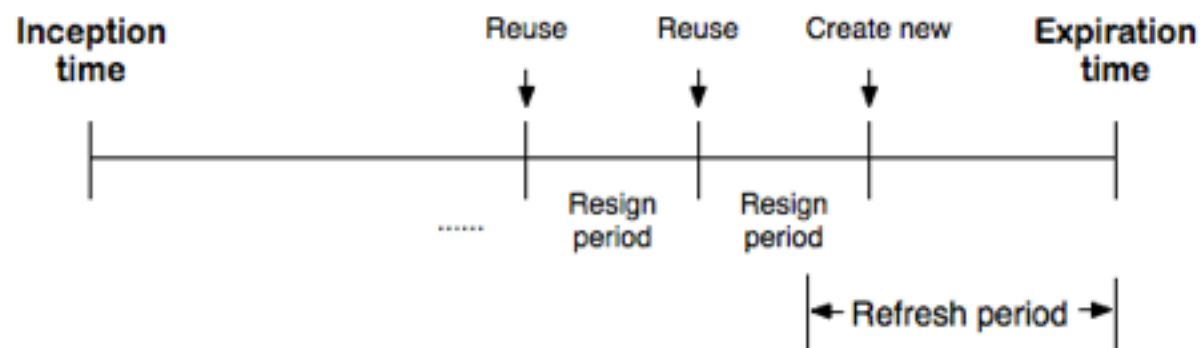
```
<Signatures>
  <Resign>PT2H</Resign>
  <Refresh>P3D</Refresh>
  <Validity>
    <Default>P7D</Default>
    <Denial>P7D</Denial>
  </Validity>
  <Jitter>PT12H</Jitter>
  <InceptionOffset>PT300S</InceptionOffset>
</Signatures>
```

For 'regular data'

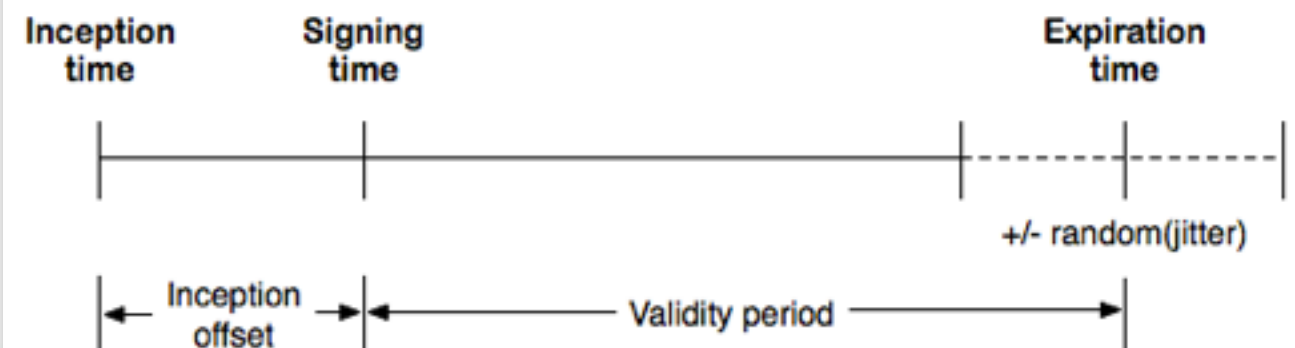
For NSEC/NSEC3

- Signatures defines timing parameters for signature creation
- Signer runs each <resign> interval
- When the signer runs during the <Refresh> interval signatures are re-generated

Reuse of signatures



Signature lifetime



Denial

```
<Denial>
  <NSEC3>
    <OptOut/>
    <Resalt>P100D</Resalt>
    <Hash>
      <Algorithm>1</Algorithm>
      <Iterations>5</Iterations>
      <Salt length="8"/>
    </Hash>
  </NSEC3>
</Denial>
```

- Denials defines parameters for Denial of Existence
- This example is for NSEC3
 - Set the Resalt and Iterations values according to paranoid level (Read RFC 5155)
- OptOut is only useful over delegations. Remove from your config.
- Use <NSEC/> instead of <NSEC3> if your zone contains trivial hostnames (www, maildrop, fruitnames, dhcpXYZ, etc)

Keys: common

<KEYS>

<TTL>PT3600S</TTL>

<RetireSafety>PT3600S</RetireSafety>

<PublishSafety>PT3600S</PublishSafety>

<ShareKeys/>

<Purge>P14D</Purge>

.....

These are common parameters
to KSKs and ZSKs

Use one slot in your HSM

- The KEYS element defines the lifetimes of keys (typically a key is used for multiple Signature periods)
- the TTL ends up in the RRset
- Retire and Publish Safety are safety margins for during key rollover
- Purge is when to remove keys for once and for all

Keys: KSK

```
<KEYS>
    . . . . .
    <KSK>
      <Algorithm length="2048">7</Algorithm>
      <Lifetime>P1Y</Lifetime>
      <Repository>softHSM</Repository>
      <Standby>1</Standby>
      <!-- <ManualRollover/> -->
    </KSK>
    . . . . .
```

- KSK sets KSK parameters for the current policy
- Repository: refers to a repository in conf.xml
- <ManualRollover/>:
ods-ksmutil key ksk-roll --zone example.com --keytag 24277
 - Manual Rollover is needed when updating your parents DS
 - Involves multiple steps

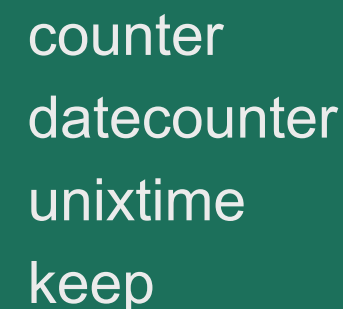
Keys: ZSK

```
<KEYS>
    .....
    <ZSK>
        <Algorithm length="1024">7</Algorithm>
        <Lifetime>P30D</Lifetime>
        <Repository>softHSM</Repository>
        <Standby>1</Standby>
    </ZSK>
</KEYS>
```

- ZSK sets ZSK parameters for the current policy
- Standby keys are present in the keyset but not used for signing. This allows them to propagate into caches so that when the keys are used to produce signatures those can be validated with DNS KEY RRs that are already in caches.

Zone

```
<Zone>
  <PropagationDelay>PT43200S</PropagationDelay>
  <SOA>
    <TTL>PT3600S</TTL>
    <Minimum>PT3600S</Minimum>
    <Serial>unixtime</Serial>
  </SOA>
</Zone>
```



counter
datecounter
unixtime
keep

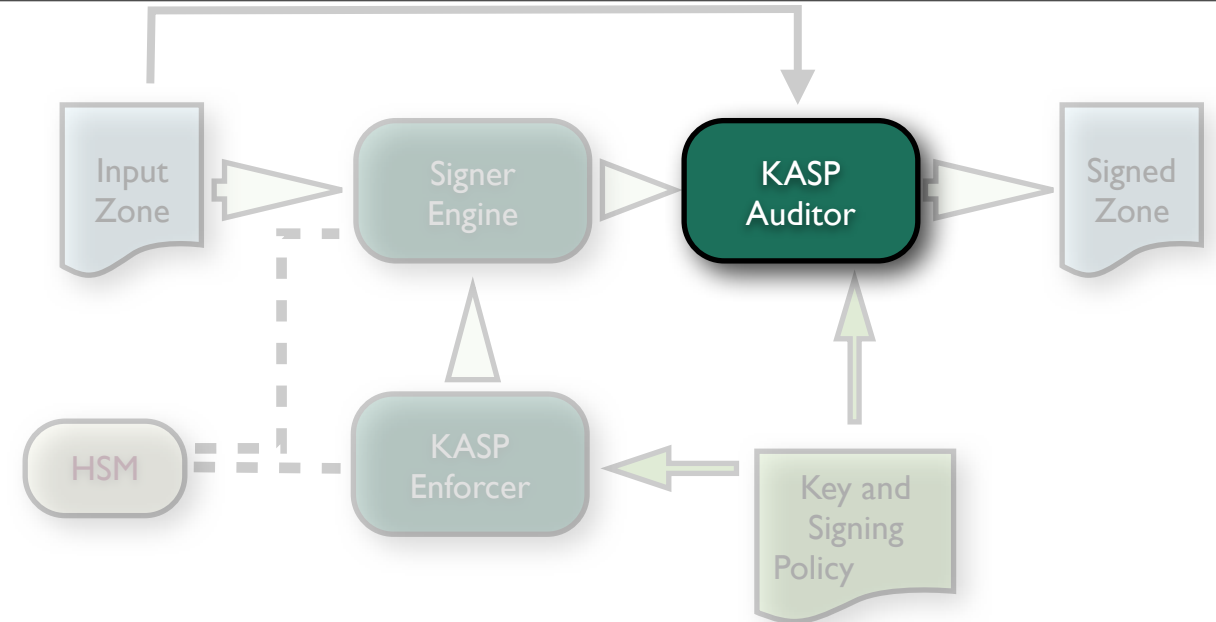
- The propagation delay is the time it takes for a zone to get to the complete set of name servers
 - Should be larger than the SOA refresh and not be larger than the SOA expiry parameter
- SOA RRs TTL and 'Minimum TTL' and Serial parameters are maintained by OpenDNSSEC

Parent

```
<Parent>
  <PropagationDelay>PT9999S</PropagationDelay>
  <DS>
    <TTL>PT3600S</TTL>
  </DS>
  <SOA>
    <TTL>PT3600S</TTL>
    <Minimum>PT3600S</Minimum>
  </SOA>
</Parent>
```

- Parent timing is important for maintaining the Chain of Trust.
- Look at the parental parameters and configure them in here
- Note that your parent may change its settings so now and then

<Audit/>



- If this element is present than all zones according to the current policy will be ‘audited’ after they are signed
 - May take a long time
 - May run out of memory
- Independent code path: provides some insurance about properly signed content
 - Is not always that liberal in parsing ‘exotic’ RRs

Now What?

- We configured conf.xml and kasp.xml
- Remember that you can have multiple policies in the KASP
 - One hsm slot serving 100 static zones with 1 private key for zone signing
 - A SoftHSM for zone signing and a KSK for key signing
 - Zones with or without parents
 - Zones with different parents (.org and .nl)
- We have to tie the policies defined in kasp.xml to the zones we want to sign

zoneList.XML

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!-- $Id: zonelist.xml.in 1147 2009-06-24 12:18:17Z jakob $ -->
```

- Preamble... its what you get when you use XML

zonelist

- Zonelist contains Zones
- Zones contain
 - Policy
 - SignerConfiguration
 - Adapters

```
<Zonelist>
  <Zone name="bla">
    <Policy>
      ...
    </Policy>
    <SignerConfiguration>
      ....
    </SignerConfiguration>
    <Adapters>
      ....
    </Adapters>
  </Zone>
  <Zone name="foo">
    ....
  </Zone>
</Zonelist>
```

First Elements in Zone

```
<Zonelist>
  <Zone name="example.com">
    <Policy>Default</Policy>
    <SignerConfiguration>
      /var/opendnssec/signconf/example.com.xml
    </SignerConfiguration>

  </Zone>
  ....
```

Name of the zone

Name of policy defined in KASP

- SignerConfiguration contains the name of a file used for passing along data between the 'enforcer' and the 'signing engine' must be unique
 - The file is temporary, don't muck with it unless you want to break things badly

The last element in Zone

```
<Zonelist>
  <Zone name="example.com">
    <Adapters>
      <Input>
        <File>/var/opendnssec/unsigned/example.com</File>
      </Input>
      <Output>
        <File>/var/opendnssec/signed/example.com</File>
      </Output>
    </Adapters>
  </Zone>
  . . . .
```

- Adapters are what gets data in and out of the signer
- Currently only file reading and writing, but OpenDNSSEC will inovate

Finishing your configuration

ods-ksmutil setup

(and then purge this command from your mind)
(oh, did you initialize your (soft)HSM?)

Start the daemons

- **ods-control start** and **ods-control stop**
- Start and stop the two daemons
 - ods-signerd
 - ods-enforcerd

Updating your configuration

- Edit the zones and run
ods-ksmutil update conf|zonelist|kasp|all

Forcing a zone to be signed

ods-signer sign example.com

Play around!!!

- Read the documentation at:
- <https://wiki.opendnssec.org/display/DOCS/OpenDNSSEC+Documentation+Home>
- Report Bugs!

Active development

- Work towards a true bump in the wire
- Enforcer NG (expected in v2.0, July)
- Signer NG input output modules in v1.4 (now in alpha)