OpenDNSSEC

Design

Table of Contents

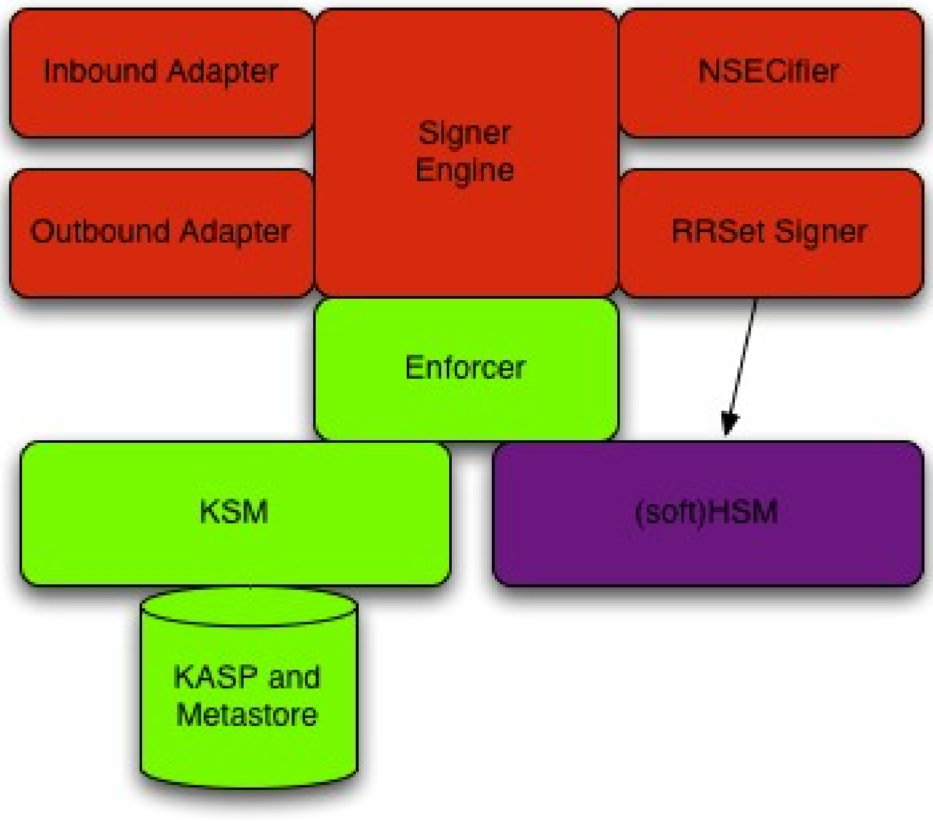
Introduction

This document attempts to describe a design for the OpenDNSSEC software.

At the moment it just describes how I think the whole system fits together. This is based on my own thoughts and expectations, the existing OpenDNSSEC wiki, and various conversations that I seem to remember having at various IETF meetings... It is intended only as a starting point for further discussion.

Architecture

This diagram is a modified version of that found on the wiki. It shows the components that are expected to form part of the OpenDNSSEC software. The components are described in more detail later in the components section. Where one component touches another or there is an arrow there will be some kind of interface or API between them. This is described in the Interfaces section.



Components

There are several components shown in the architecture diagram. These are described below.

1. Enforcer

The Enforcer controls the signing process by reading the KASP and meta-store databases and deciding what needs doing. It calls the signer engine when necessary and passes it all the information needed to run. The Enforcer is the only part of this system that talks to the database.

The Enforcer also takes care of key management, generating, rolling and destroying keys as needed.

2. KSM

This is a library that can be used to build applications that need to access KASP and/or the meta-store. Such applications may include the Enforcer, a command line key management utility, nagios plugins etc.

3. KASP and Meta-store Database

KASP stores the policies that will be enforced for signing a zone or group of zones.

The meta-store holds information about the state of the signed zones.

They may be separate databases, schemas or just different tables in the same schema.

See you database schema section for more information.

4. Signer Engine

This is called by the Enforcer and signs the zone. It reads the unsigned zone using an adapter that allows it to read a file or perform an AXFR request. It "publishes" the signed data in a similar way.

5. RRSet Signer

This receives RRSets from the signer and passes them to the HSM to be signed. It returns the signature to the Signer to include in the zone.

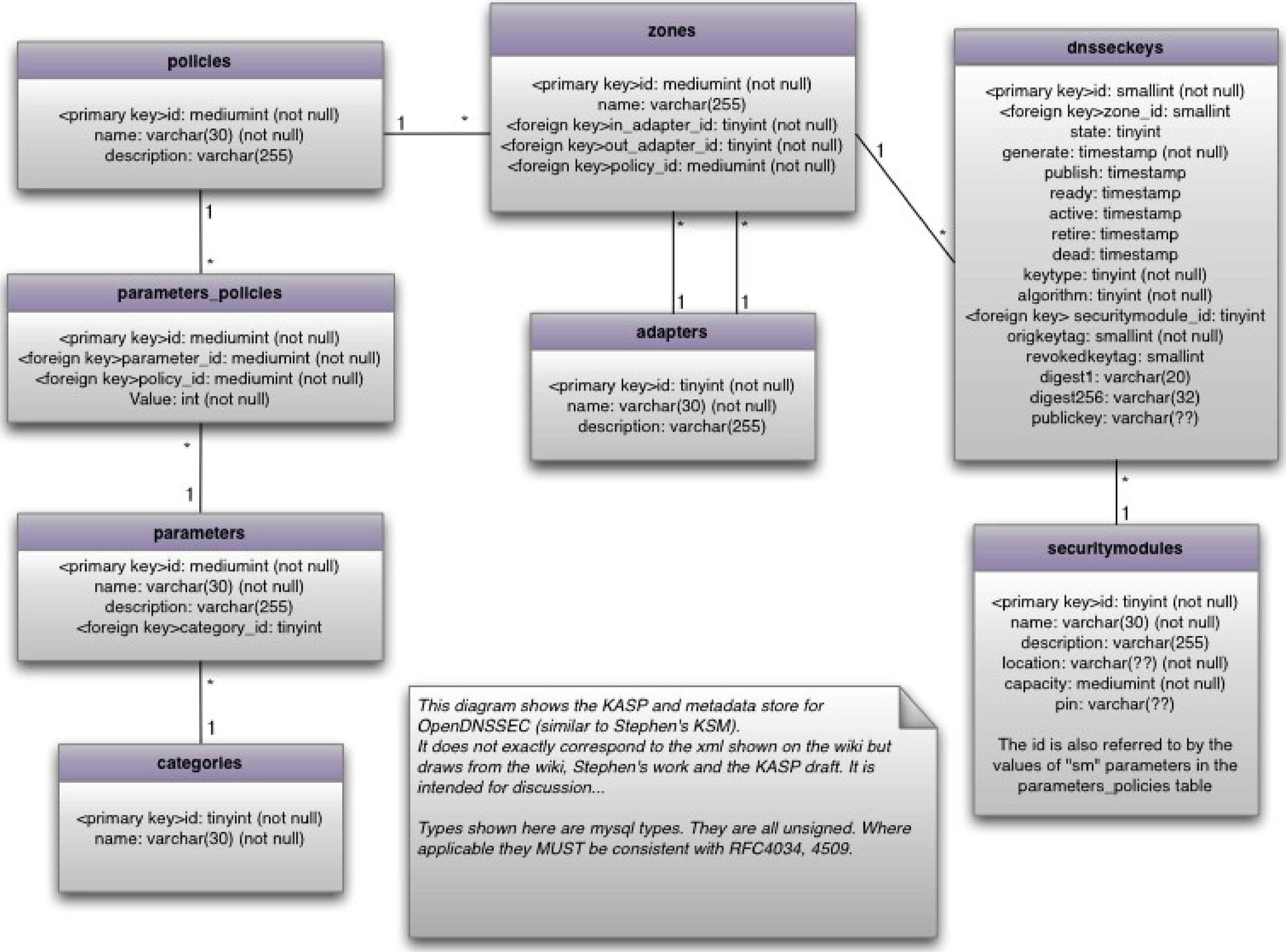
6. NSECifier

This generates NSEC or NSEC3 Rrs for inclusion in the zone

7. Adapters

Database Schema

The database schema shown below is a draft of a schema developed to explore the data that needs to be stored for OpenDNSSEC.



policies
<primary key>id: mediumint (not null)
name: varchar(30) (not null)
description: varchar(255)
zones
<primary key>id: mediumint (not null)
name: varchar(255)
<foreign key>in_adapter_id: tinyint (not null)
<foreign key>out_adapter_id: tinyint (not null)
<foreign key>policy_id: mediumint (not null)
dnsseckeys
<primary key>id: smallint (not null)
<foreign key>zone_id: smallint
state: tinyint
generate: timestamp (not null)
publish: timestamp
ready: timestamp
active: timestamp
retire: timestamp
dead: timestamp
keytype: tinyint (not null)
algorithm: tinyint (not null)
<foreign key> securitymodule_id: tinyint
origkeytag: smallint (not null)
revokedkeytag: smallint
digest1: varchar(20)
digest256: varchar(32)
publickey: varchar(??)
parameters_policies
<primary key>id: mediumint (not null)
<foreign key>parameter_id: mediumint (not null)
<foreign key>policy_id: mediumint (not null)
Value: int (not null)
adapters
<primary key>id: tinyint (not null)
name: varchar(30) (not null)
description: varchar(255)
parameters
<primary key>id: mediumint (not null)
name: varchar(30) (not null)
description: varchar(255)
<foreign key>category_id: tinyint
securitymodules
<primary key>id: tinyint (not null)
name: varchar(30) (not null)
description: varchar(255)
location: varchar(??) (not null)
capacity: mediumint (not null)
pin: varchar(??)
The id is also referred to by the values of "sm" parameters in the parameters_policies table
categories
<primary key>id: tinyint (not null)
name: varchar(30) (not null)
1
*
1
*
*
1
*
1
*
*
1
1
1
*
*
1

This diagram shows the KASP and metadata store for OpenDNSSEC (similar to Stephen's KSM).
It does not exactly correspond to the xml shown on the wiki but draws from the wiki, Stephen's work and the KASP draft. It is intended for discussion...

Types shown here are mysql types. They are all unsigned. Where applicable they MUST be consistent with RFC4034, 4509.

Interfaces between components

1. Enforcer - KSM

The enforcer is written using the KSM API - This exists in part and needs extending to be consistent with the OpenDNSSEC Schema described elsewhere. It needs documenting.

2. KSM – Database

This uses database access function that are wrappers around an appropriate database API. Currently this uses MySQL.

3. Enforcer – Signer

In the first version this will rely on the signer engine having a command line interface. that can be called by the Enforcer. Similar to that currently used by ldns-signzone but extended to account for the additional options needed.

4. Signer Engine – NSECifier

5. Signer Engine – RRSet Signer

6. Signer Engine – Adapter

7. RRSet Signer – (soft)HSM

The RRSet Signer is written using PKCS11 API to perform signing operations. It will dynamically link to the necessary PKCS11 library specified for the HSM in the KASP database. (The location of the library will be passed by the Enforcer to the Signer Engine and on to the RRSet Signer.)

Plan of work

Work will begin by developing a prototype of the system that can perform all the operations needed but not necessarily in the most optimal or best performing way. This prototype will be used to explore how the system should work and will form the bases of a full implementation in the future.

Approximate timelines are end of March 2009 for the prototype and Summer 2009 for the full implementation.

Current status of Components

Enforcer

John Dickinson has some initial code.

KSM

Stephen Morris wrote a command line utility that utilised a database similar to the KASP concept used by OpenDNSSEC. This utility already has a KSM component.

John Dickinson has started modifying that KSM component so that it can be used by the Enforcer.

Signer

This is likely to be based on ldns-signzone. PKCS11 support needs adding to LDNS???????

SoftHSM

Rickard has written this and working code exists on the OpenDNSSEC trac site.

Vision for final system

The final implementation may be a single daemon or set of applications that work together.

Outstanding Issues

1. If the signer is to be able to do XFR to obtain and serve zone data then it needs to maintain state – How will this work???