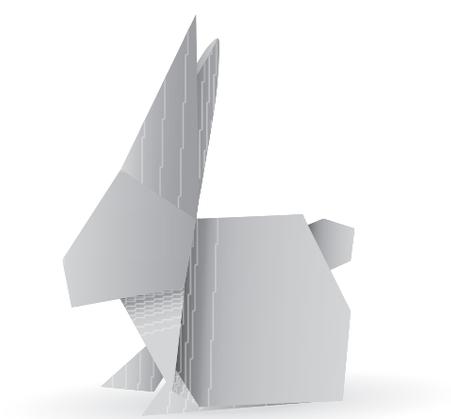
OpenXPKI

and its ecosystem



Martin Bartosch, 2015/10

WhiteRabbitSecurity

The White Rabbit put on his spectacles.

“Where shall I begin, please your Majesty?” he asked.

“Begin at the beginning,” the King said gravely,
“and go on till you come to the end: then stop.”

(Alice’s Adventures in Wonderland, Chapter 12)

10 Years of Development…

… creating the building blocks of an enterprise level
Open Source
Key Management Solution

WhiteRabbitSecurity

OpenSource Key Management

- **OpenXPKI -** The Trustcenter Software
- **clca** - Offline Root CA environment
- **CertNanny** - Automatic end entity certificate renewal
- **KeyNanny** - System level credential protection

Timeline

- 2004-10-12: first OpenCA workshop, Munich
- 2005-10-18: second OpenCA workshop, Munich
- **2005-10-20: OpenXPKI project founded**
- 2005-12-23: **CertNanny** first public release
- 2013-08-13: **clca** first public release
- 2014-08-14: **KeyNanny** first public release
- **2015-10-20: OpenXPKI 1.0**

Some Statistics, quoting OpenHub

https://www.openhub.net/p/openxpki

In a Nutshell, OpenXPKI...

... has had 6,304 commits made by 28 contributors
representing 332,264 lines of code

... is mostly written in Perl
with an average number of source code comments

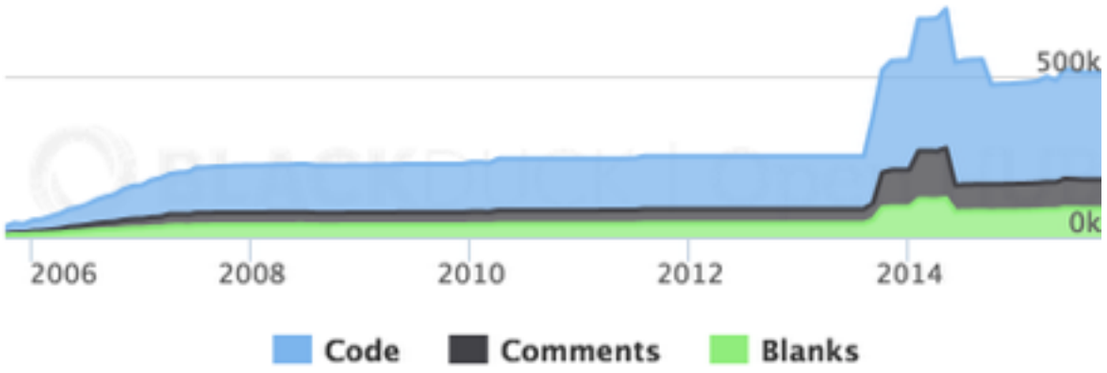
... has a well established, mature codebase
maintained by a large development team
with decreasing Y-O-Y commits

... took an estimated 86 years of effort (COCOMO model)
starting with its first commit in October, 2005
ending with its most recent commit about 1 month ago

Languages

Perl	45%	XML	16%
JavaScript	27%	10 Other	12%

Lines of Code



OpenXPKI
The Trust Center - Key Features

- Primary use case: Online Level 2 Issuance CA
- Multiple CA „Realms“ (namespaces, logical CAs)
- Seamless Issuing CA rollover (within a Realm)
- Highly configurable and flexible

OpenXPKI
The Trust Center - Key Features

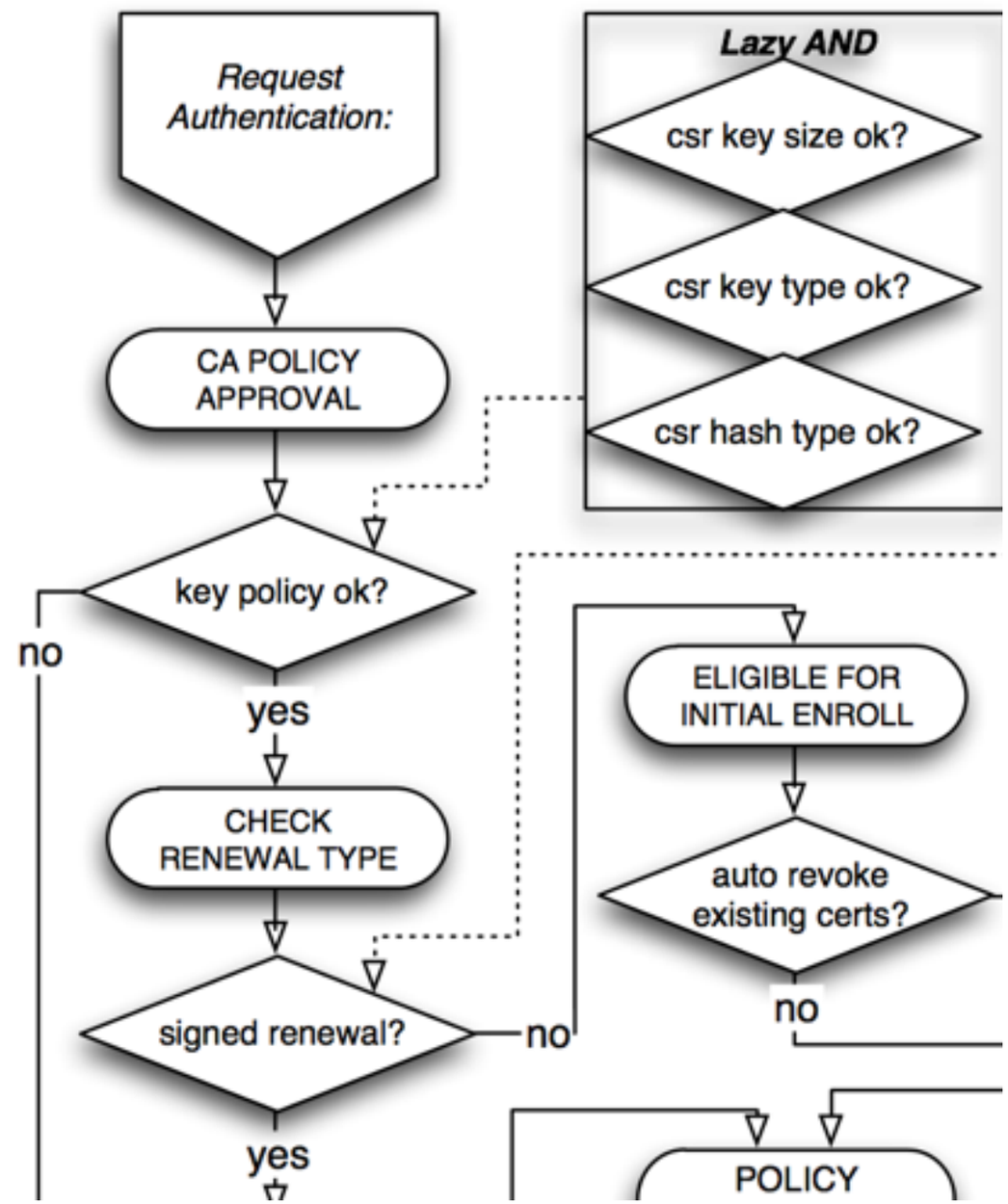
- Automation interfaces
- Simple integration of external data sources
- Crypto backend agnostic (Software keys, HSM, cascading to other CAs…)
- Workflow engine: Business logic implementation

OpenXPKI - Workflows

- Stateless crypto toolbox
- Stateless server API
- Workflows for all complex or persistent operations, e. g.
  - request certificate (GUI)
  - request revocation (GUI)
  - automatic enrollment via SCEP protocol

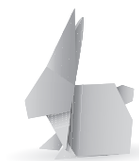
OpenXPKI - Workflows

- Workflow example: generic enrollment workflow for automatic certificate issuance
  - Primarily used by the SCEP interface
  - Configurable certificate lifecycle policy (e. g. maximum number of active certificates)
  - Authentication: configurable policy and sources
  - Authorization: configurable policy and sources



OpenXPKI - Configuration

- OpenXPKI configuration layer uses an abstraction for arbitrary data sources („Connector“)
- Connectors can be used *anywhere*, for *any* configuration item
- Replace literal configuration value with a reference to a Connector returning the value
- OpenXPKI configuration extensively supports templating (via Template::Toolkit)



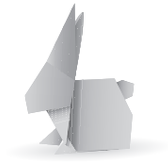
OpenXPKI - Configuration Example

- `challenge.value:` ask „scep.connectors.challenge“
- Pass workflow instance context parameter „cert_subject“ as argument

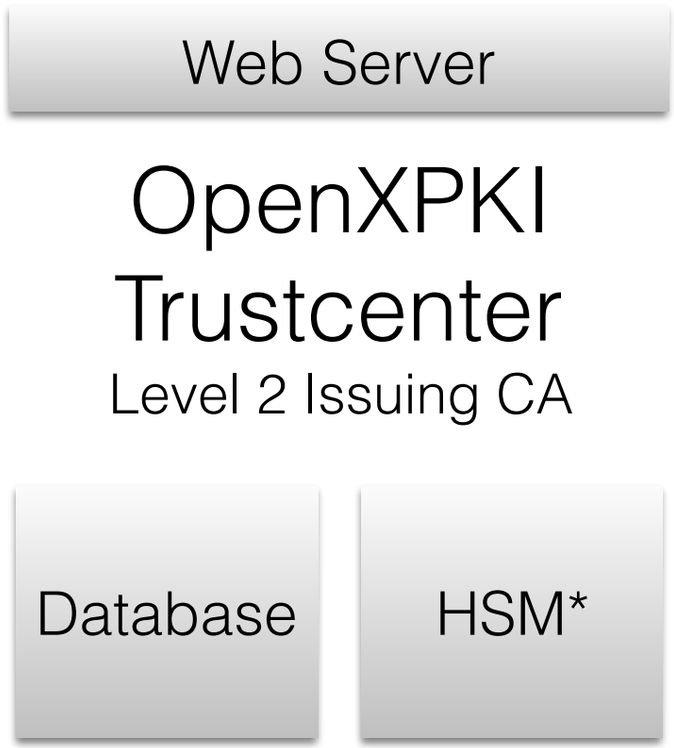
```
scep:
  challenge:
    mode: bind
    value@: connector:scep.connectors.challenge
    args:
    - "[% context.cert_subject %]"

connectors:
  challenge:
    class: Connector::Builtin::Authentication::Password
    LOCATION: /etc/openxpki/openxpki/passwd.txt
```

Architecture Patterns and Use Cases

WhiteRabbitSecurity

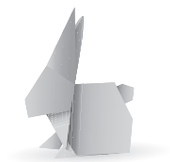


OpenXPKI Single Node

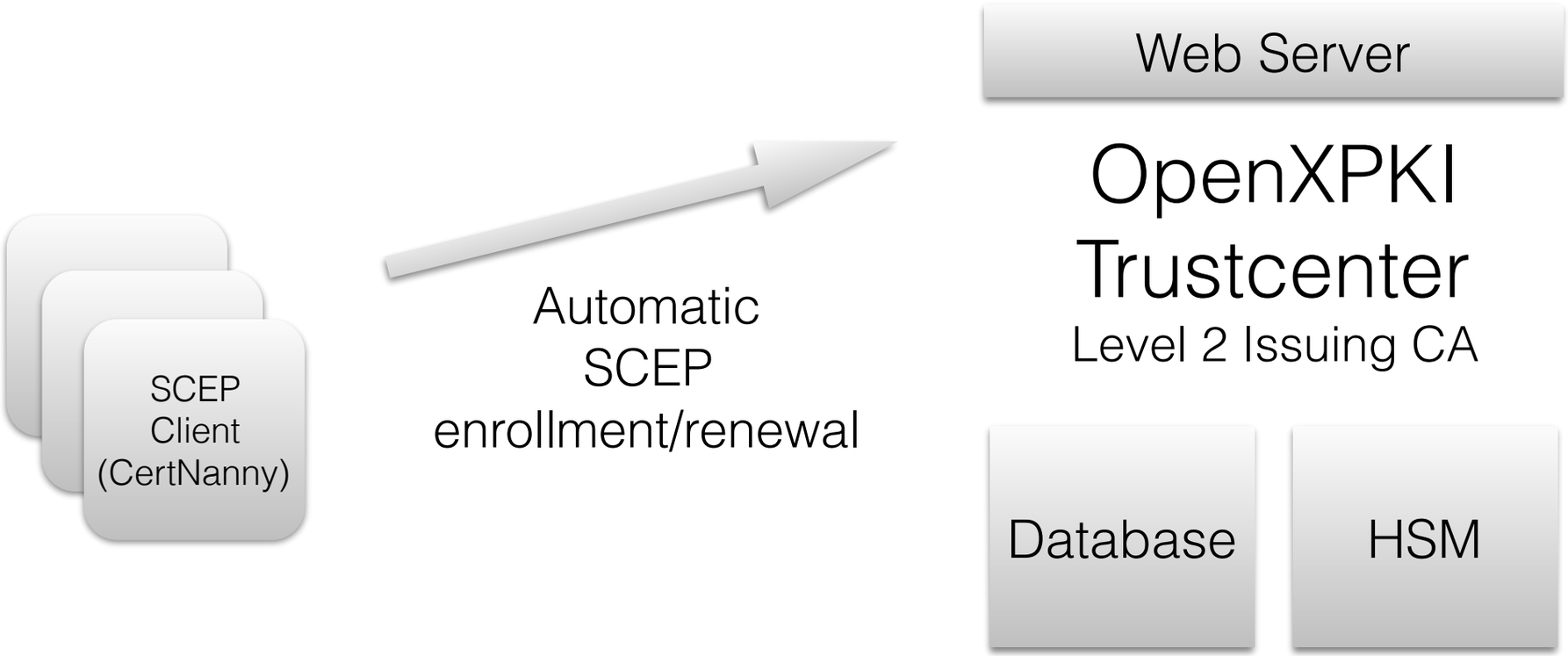


* HSM is optional

WhiteRabbitSecurity



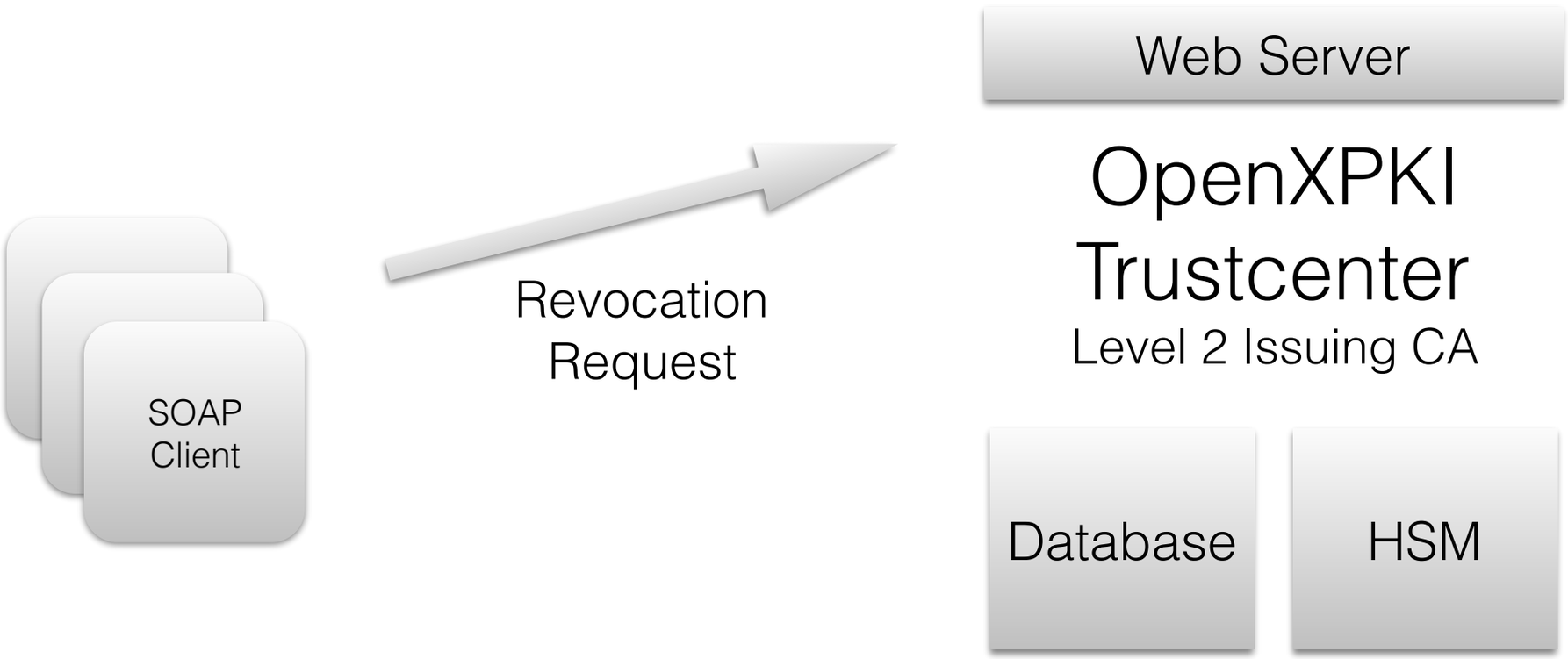
OpenXPKI providing automatic enrollment/renewal



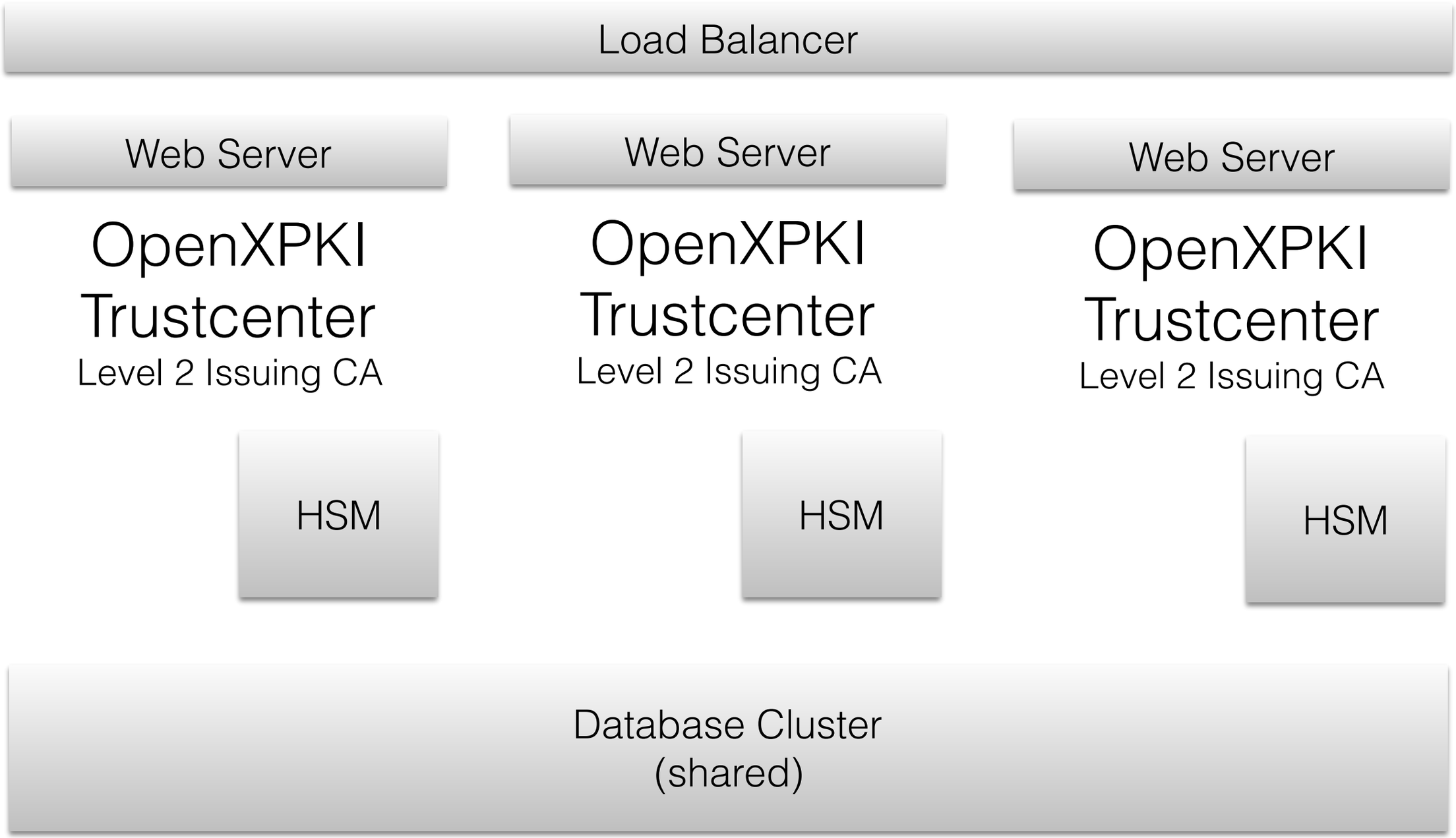
```
authorized_signer_on_behalf:
    technicians:
        subject: CN=.*DC=SCEP Signer CA,DC=mycompany,DC=com
        profile: I18N_OPENXPKI_PROFILE_SCEP_SIGNER
    blackbox:
        identifier: JNHN5Hnje34HcltluuzooKVqxss

challenge:
   value: SecretChallenge

renewal_period: 000014
replace_period: 05
revoke_on_replace:
    reason_code: keyCompromise
    invalidity_time: +000014

eligible:
    initial:
       value: 0
    renewal:
       value: 1

policy:
    allow_anon_enroll: 0
    allow_man_authen: 1
    allow_man_approv: 1
    max_active_certs: 1
    allow_expired_signer: 0
    auto_revoke_existing_certs: 1
    approval_points: 1


# Mapping of names to OpenXPKI profiles to be used with the
# Microsoft Certificate Template Name Ext. (1.3.6.1.4.1.311.20.2)
profile_map:
    pc-client: I18N_OPENXPKI_PROFILE_USER_AUTHENTICATION

subject_style: enroll
```

SCEP configuration example (incomplete)

OpenXPKI SOAP API

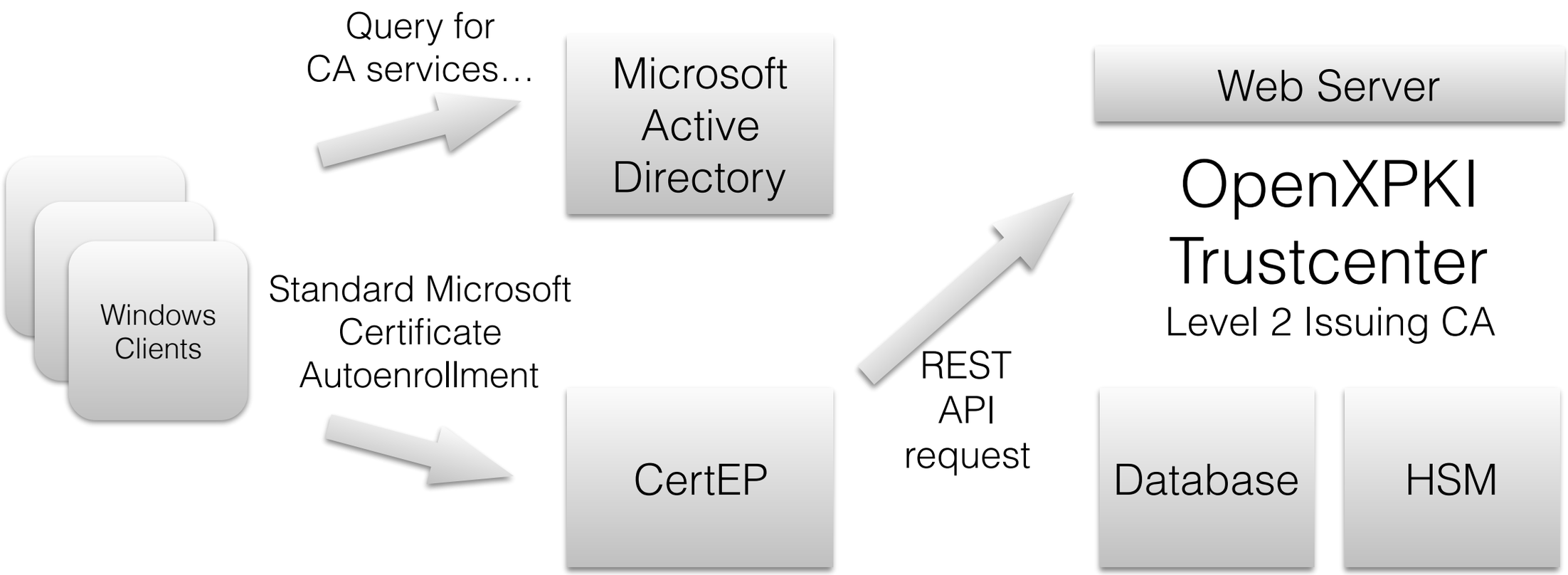


OpenXPKI Cluster Setup



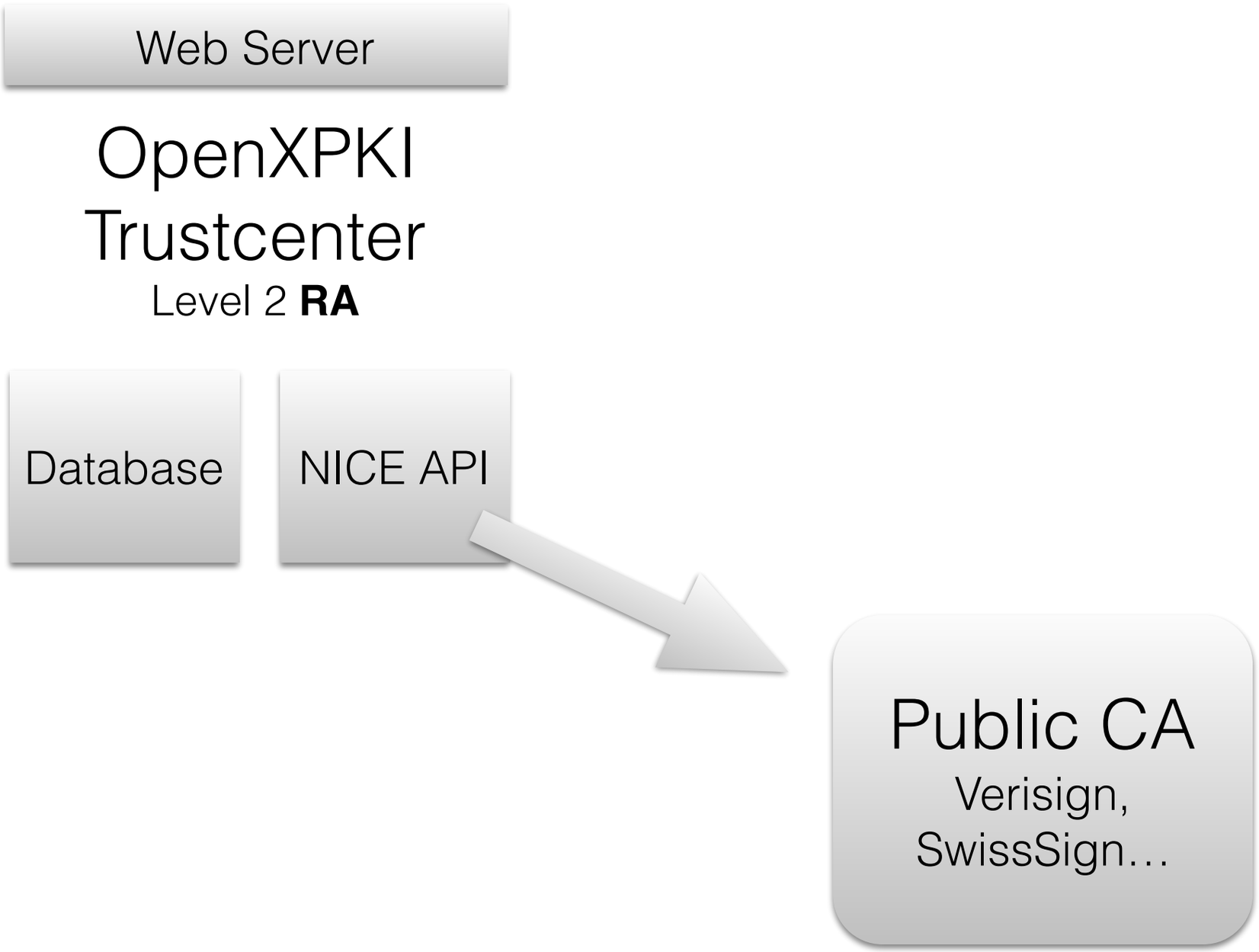
WhiteRabbitSecurity

OpenXPKI as Microsoft CA replacement using CertEP

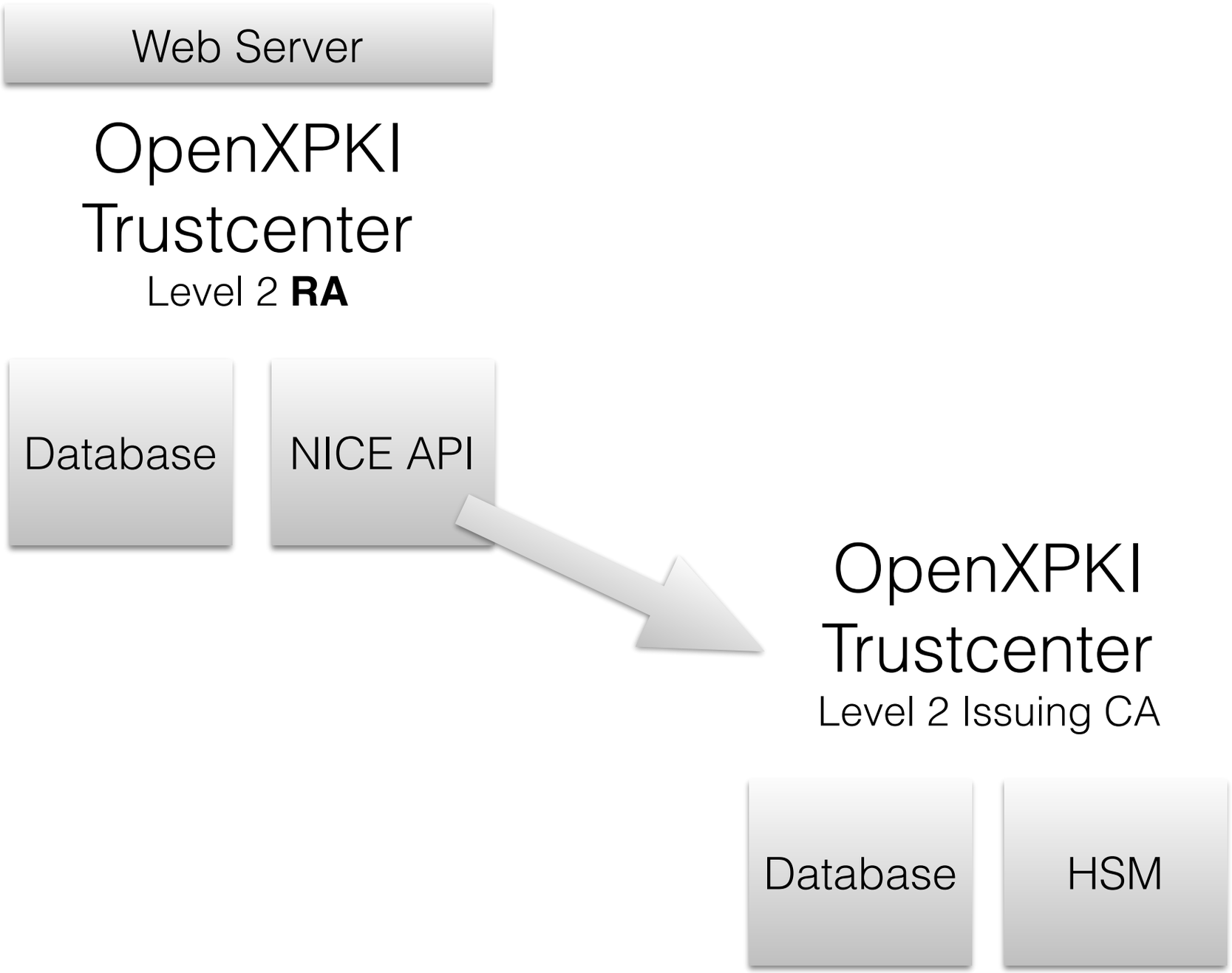


Certificate Enrollment Proxy
(Commercial Third Party Product)
Secardeo GmbH
http://www.secardeo.de/produkte/certep/

OpenXPKI using Public CAs as backend



OpenXPKI chaining/cascade (planned)



WhiteRabbitSecurity

clca

Offline Root CA environment

WhiteRabbitSecurity

clca
Root CA Environment

- Root CA: high security requirements, low usage frequency
- Typical tasks
  - Issue Root CA Certificates
  - Issue Root CA CRLs
  - Issue Level 2 Issuing CA Certificates
  - Perform administrative HSM operations (e. g. key generation)

clca
Root CA Environment

- clca itself is a bash shell script, wrapping OpenSSL
- implements simple commands:
  `clca initialize` - create self-signed root
  `clca issue_crl` - issue Root CA CRL
  `clca certify` - sign sub-ca request

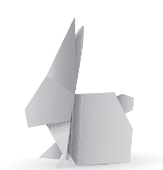
clca
Root CA Environment

- include scripts for creating a bootable Linux Memory Stick (or CD)
- include clca script and other necessary tools
- persistent home directory on memory stick contains CA data
- optional: include HSM drivers and set up infrastructure

CertNanny

Fully automatic certificate renewal

WhiteRabbitSecurity



CertNanny
Certificate Management Automation

- Basic assumption: CA side certificate monitoring does not work
- Client side agent monitors all keystores on end entity system
- Must be configured once by an experienced (sic!) admin
- Monitors certificates and automatically renews via SCEP

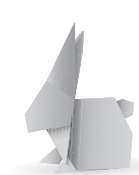
CertNanny
Certificate Management Automation

- Renewal mode: sign request with existing (old) private key
- Initial Enrollment mode: sign request with preinitialization key or embed challenge password
- Once new certificate is downloaded, composes new keystore from scratch, replaces existing one
- Maintenance-free if correctly configured

KeyNanny

Server credential management

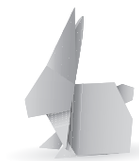
WhiteRabbitSecurity



KeyNanny
Server Credential Management

- Server systems need sensitive information in configuration files
  Example: database passwords, LDAP bind credentials, software private key passwords

- Problem: how to securely manage system configuration if it contains sensitive information?



KeyNanny
Server Credential Management

- Naive approach: „encrypt passwords“ in config…
- … with a password which is stored in…
- … the configuration? Or the application?
- Essentially this is *obfuscation*, not encryption
- Problem cannot be solved without hardware crypto

KeyNanny
Server Credential Management

- Unix Daemon that can be nicely asked for credentials by the application…
- … via a Unix Domain Socket, protected by Unix Permissions.
- KeyNanny uses one (or more) certificates for credential encryption. Support hardware crypto (HSM).
- Keep credential tuples on disk encrypted via CMS/ PKCS#7
- Decrypt when asked, return password to application

KeyNanny
Server Credential Management

- Direct access to KeyNanny:
  - application directly accesses KeyNanny via Socket protocol (requires native application support)
- Indirect: render config to temp file on memory file system
  - Store application configuration template on disk. Replace passwords with placeholders
  - On KeyNanny startup render configuration file from template, replacing all password references found
  - On KeyNanny shutdown unmount memory file system, destroying the configuration file

Thank You!

WhiteRabbitSecurity