



**OpenCA v1.0.2+
(ten-ten²)**



Massimiliano Pala <project.manager@openca.org>

Outline

- Basic Installation Procedures
- Initial Configuration of a new CA
- New Features Overview
 - Auto CA, Auto CRL, and Auto E-Mail
 - Browser Request & Authenticated Browser Request
 - Level of Assurance and License Agreements
- Cryptographic Updates
 - ECDSA support
 - Upgrading to SHA256

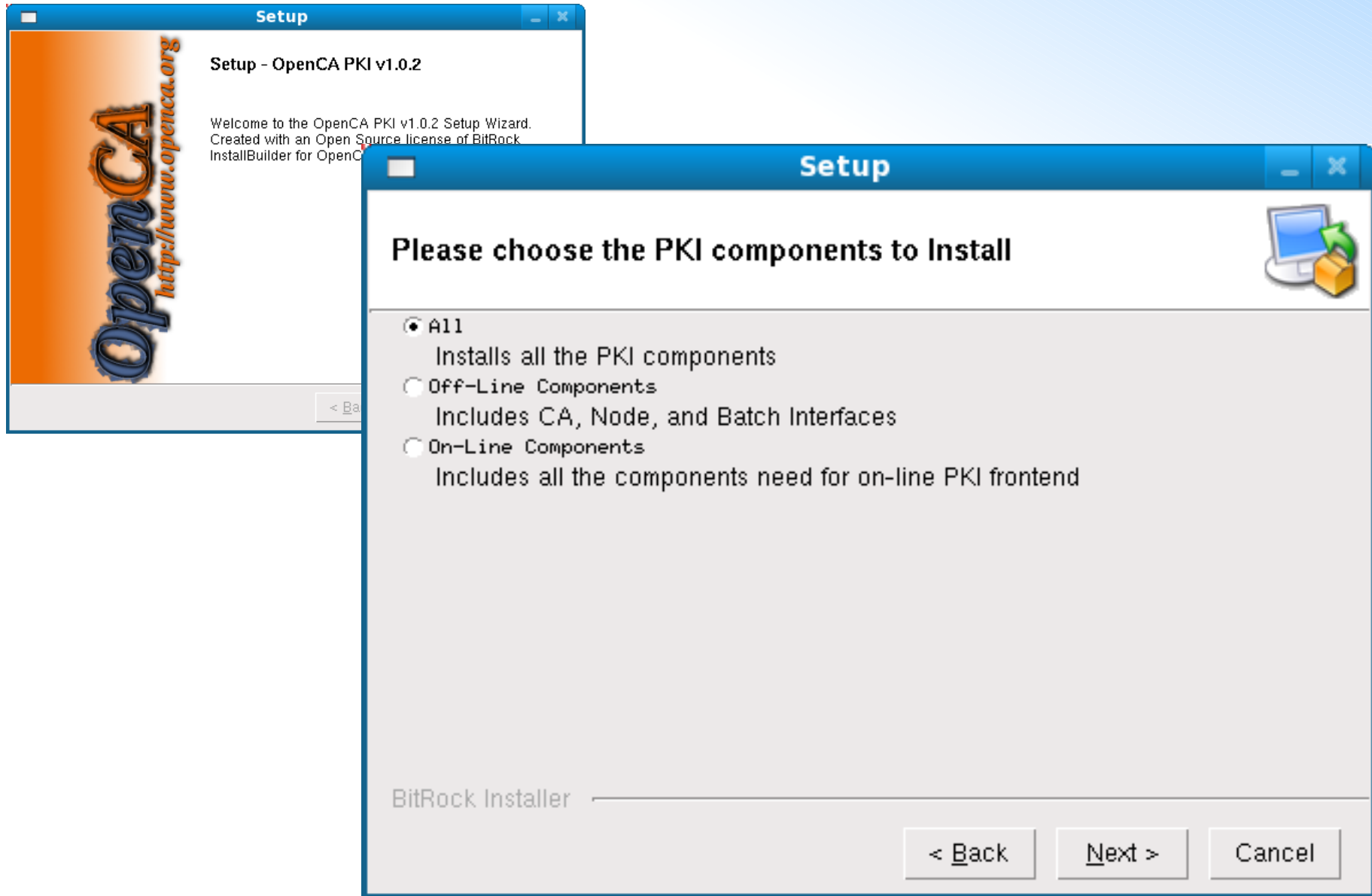


Package Installation

OpenCA - Graphical Installer

- Graphical installer for binary installation
 - Donation by BitRock
 - Available for Fedora, Ubuntu, OpenSolaris, etc...
 - Useful for 99% of common installations
 - Runs in both graphical and text mode for remote installation
- Not available for every platform
 - New distributions
 - More resources needed
 - Community support
- Dependencies
 - OpenCA-Tools

OpenCA - Graphical Installer (cont.)



OpenCA - Graphical Installer (cont.)

Setup

Organization Name

Please provide the name that identifies your organization. This parameter will set several variables in the configuration files (PREFIX/etc/openssl/config.xml). An example can be 'OpenCA Labs' or 'University of Modena'. What is the name of your Organization ?

Organization Name

BitRock Installer

Setup

Password

All the administrative interfaces of OpenCA require the user to be authenticated. The default username is 'admin'. Please provide the password for the default user.

Password

Retype Password

BitRock Installer

Setup

OpenCA Support

The OpenCA Project is built around a community of many users and developers who contributed with their time, code and knowledge to the project. The OpenCA Users mailing list is the best place where to ask for help or suggestions about your installation of OpenCA.

Do you want to subscribe to the OpenCA-Users mailing list now ?

Yes
 No

BitRock Installer

< Back Next > Cancel

Installing OpenCA from Sources-1

- Source package installation
 - Suitable for many UNIX systems
 - BSD, Solaris, Linux, etc..
 - More flexible (allows many options to be set at installation time)
 - Requires more expertise in solving configuration issues
 - Does not automatically install OpenSSL (with ECDSA support)
- Dependencies
 - OpenSSL
 - OpenCA-Tools

Installing OpenCA from Sources-2

■ Example:

```
$ ./configure --prefix=/opt/openca \  
    --with-openssl-prefix=/opt/openca-openssl \  
    --with-httpd-user=apache  
...  
$ make  
...  
$ make install-offline  
...  
$ make install-online
```



Let's Install OpenCA

Applying available Patches

- Wiki Pages
 - All OpenCA Labs projects
 - Check them for the latest patches available
- OpenCA PKI wiki
 - http://mm.cs.dartmouth.edu/wiki/index.php/OpenCA_PKI_v1.0.x
 - Linked directly from OpenCA Labs website
- Applying the Patches
 - Follow the instructions on the Wiki

Installing OpenCA: what's Next ?

- Many Options to fit your CA
 - System Configuration(s)
 - Certificate Profiles and Certificate Policies
 - Service(s) configuration
 - Web Server
 - LDAP (for cert. publication)
 - Data Exchange device(s) or scripts
- Core Configuration file
 - PREFIX/etc/openca/config.xml
- Activate changes: restart OpenCA!

Starting the needed Daemons

- Needed daemons
 - HTTP server
 - Database server
 - OpenCA server
- Core Configuration file
 - PREFIX/etc/openca/config.xml
- Configuring the Self-Sign CA profile
 - PREFIX/etc/openca/openssl/openssl.cnf.template
(options from [v3_ca] section)

Configuration: Initial Options



Let's see an example...

Profile Configuration

- OpenCA uses OpenSSL configurations
- Certificate Profiles == Extensions == OpenCA Roles
- To add a new profile:
 - Add the new role on the CA interface
 - Edit the new configuration file in PREFIX/etc/openssl/extensions/
 - If needed copy the config_file to config_file.template
 - After adding a role, use the export/import to deploy the changes to other interfaces (pub/ra)
- Suggestions:
 - Use simple profiles
 - Use RFC 5280 as a reference (do not rely on intuitions!)
 - Check the OpenSSL documentation on how to specify extensions in config files

IGTF Profiles

- Required Properties (CA)
 - BasicConstraints – set to CA:TRUE
 - KeyUsage – critical
 - KeyCertSign, CrlSign
- Advised Properties (CA)
 - AuthorityKeyIdentifier – no default value (supposedly the hash of the key – which algorithm ?)
 - SubjectKeyIdentifier – no default value (supposedly the hash of the key – which algorithm ?)
- Optional Properties (CA)
 - CrlDistributionPoint – set to at least one http URI

IGTF Profiles (cont.)

■ Required Properties (EE)

- KeyUsage – critical
 - DigitalSignature, KeyEncipherment (dataEncipherment)
- ExtendedKeyUsage – non critical
 - ClientAuth (serverAuth for servers)

■ Advised Properties (EE)

- BasicConstraints – set to CA:FALSE
- CrlDistributionPoint – set to at least one http URI
- CertificatePolicies – IGTF Profile OID + Issuer OID
- SubjectAltName – email (EE) or dnsName (Server)

■ Optional Properties (EE)

- NsComment, nsPolicyURL, nsRevocationURL, AKI, SKI, AIA, IAN



CA Interface On-line Daemons

Online Daemons

- On-Line daemons to ease Grid operations
 - On-Line CA support
- Supported CA Operations
 - Automatic Certificate Issuing
 - Automatic CRL issuing
- Supported RA Operations
 - Automatic E-Mail warning to Users

Auto Certificate Issuing

- CA Interface

- CA Operations -> Auto Certificate Issuing -> Enable

- RA Options

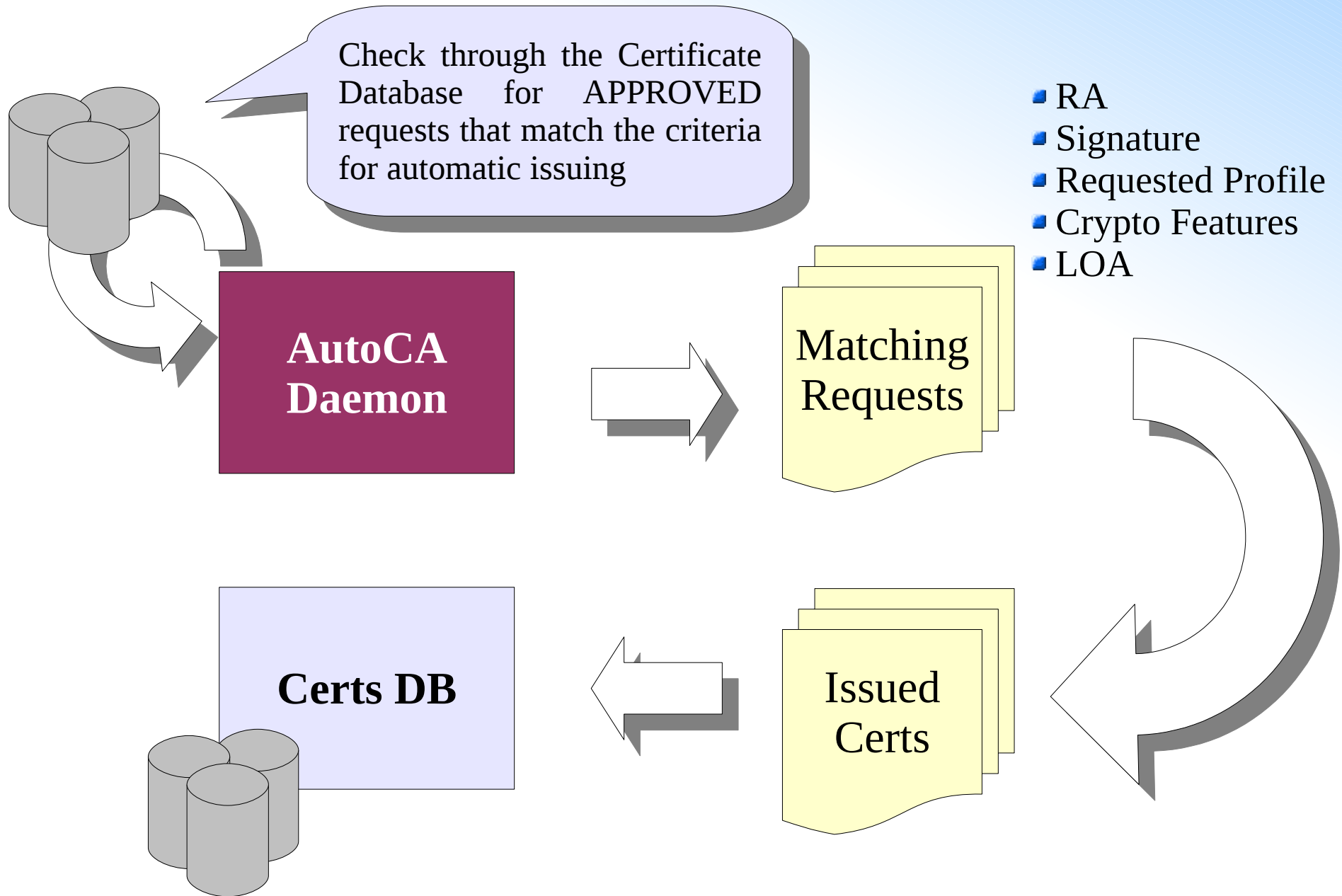
- Registration Authority (Requested by the User or setup in the request)
- RA Operator that approved the request
- Signed Requests only

- Request Details

- Requested Role
- Level of Assurance

- Accepted Algorithms and Key Sizes

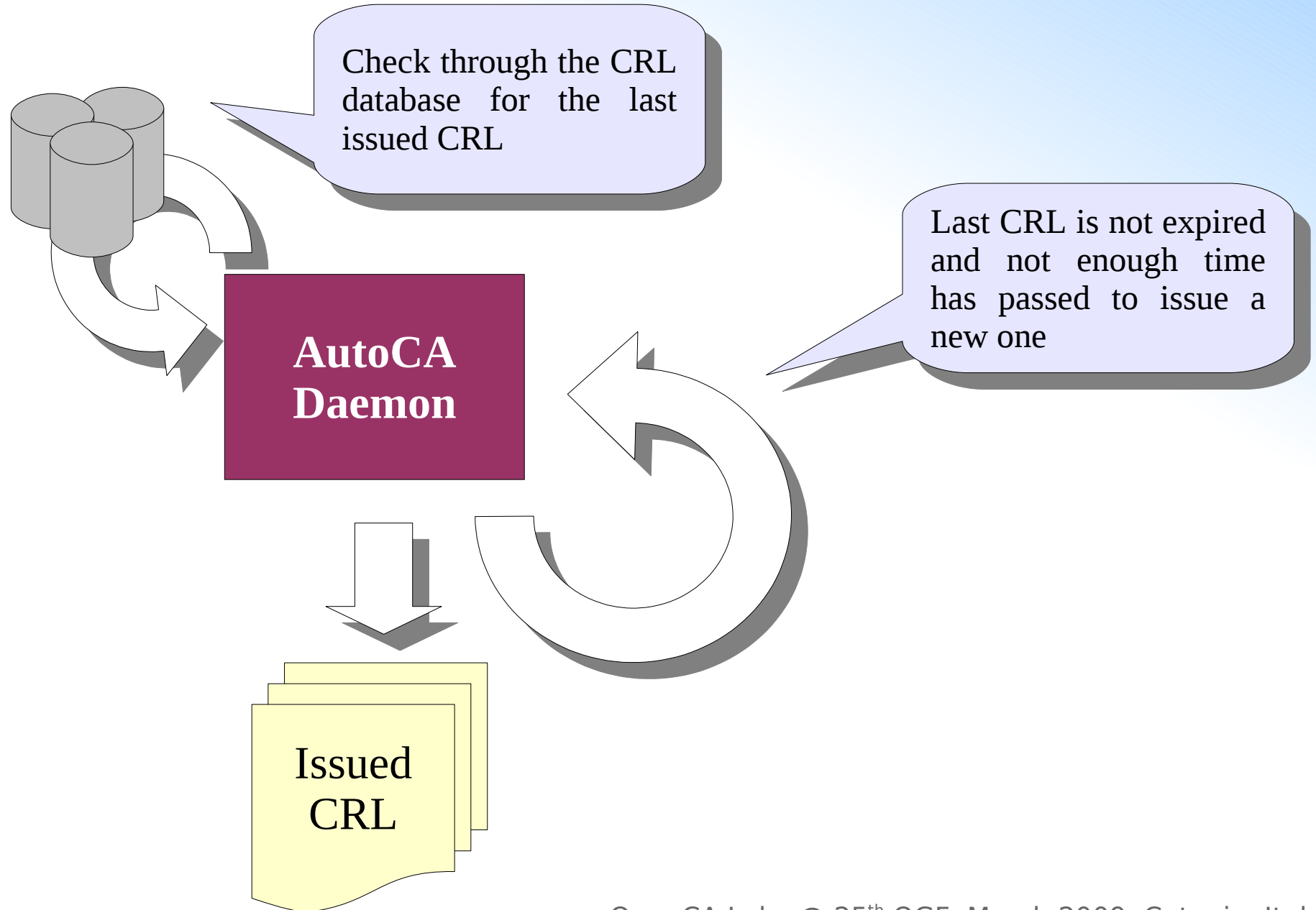
Auto Certificate Issuing



Auto CRL Issuing System

- CA Interface
 - CA Operations -> Auto CRL Issuing -> Enable
- CRL Options
 - Issue CRL Every Period
 - CRL Validity is Period
 - CRL Extension
- Where Period can be:
 - Days, Hours, Minutes, Seconds (for Issuing Period)
 - Days, Hours (for Validity – Limitation of OpenSSL)

Auto CRL Issuing



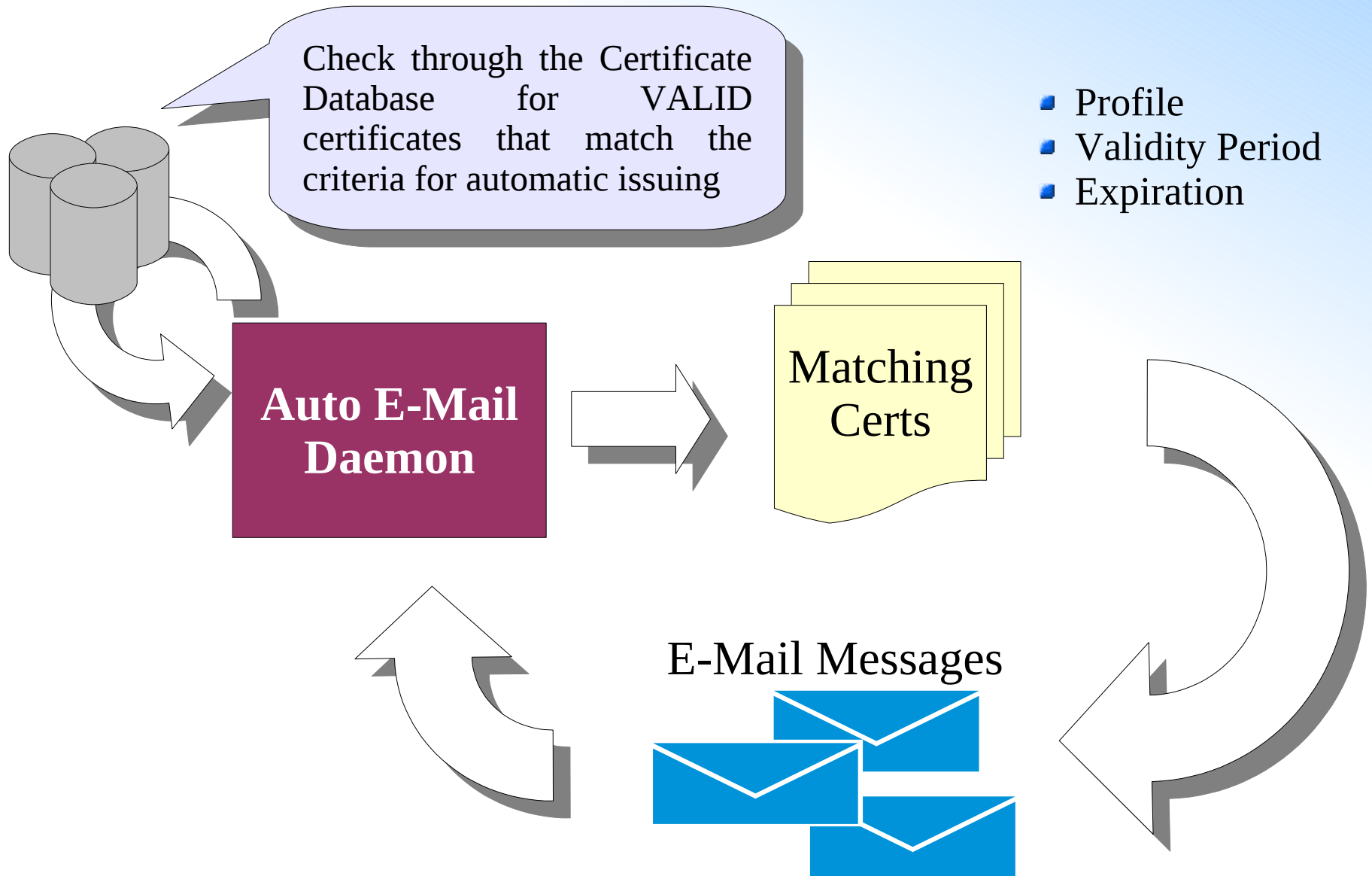


RA Interface On-line Daemons

Auto E-Mail System

- RA Interface
 - RA Operations -> Auto E-Mail -> Enable
- Daemon Process Options
 - Check Period for outgoing mail
 - Days, Hours, Minutes, Seconds
- E-Mail Options
 - Warn for Expiring Certificates
 - Filter Certificates by:
 - Role, LOA, Validity Period (minimum lifespan)
 - Up to two warning E-Mails
 - Certificates expiring within (Days, Hours)

Auto E-Mail Issuing





Certificate Request Configuration

Browser Request

- Unified Request for different browsers
 - Mozilla, Firefox, Konqueror, Opera, IE6, IE7
 - Support Server Side Key and Request Generation
- XML configuration
 - PREFIX/etc/openca/browser_request.xml
 - Different Sections supported
 - User, Certificate, Keygen, Agreement
 - Subsections (Groups of Input fields)
 - Base DN added automatically to the request
 - Final Request Status (eg., NEW, APPROVED)
- Full Input fields configuration

Browser Request (cont.)

- **\$EXEC::[function]** - Executes a function and uses the output to populate the input object
 - **loadDataSources()** - generates the list of the configured datasources in *datasources.xml.template* (**)
 - **loadRoles()** - generates the list of Roles (or profiles)
 - **loadLoa()** - generates the list of Levels Of Assurance
 - **loadKeygenMode()** - generates the list of Key Generation Modes allowed for the currently used browser (check the *loa.xml* config file as well)
 - **loadKeyTypes()** - generates the list of allowed Key Types. Currently supported are RSA, DSA, ECDSA; the list can be shorter depending on the capabilities of the browser and the type of current request.
 - **loadKeyStrengths()** - generates the list of allowed Key Strengths (check the *loa.xml* config file for more explanation)

Browser Request (cont.)

- **\$DATA::[fieldname]** - substitute the value with the FIELD value gathered from the chosen datasource
- Use the **<map></map>** option in the datasources configuration file to map names across different datasources
(PREFIX/etc/openca/datsources.xml.template)
- Mapping options
 - Name – name of the original field in the DS
 - Value – name of the field name to be mapped to

Browser Request (cont.)

- In the request configuration file we can use `$DATA::userIdentifier` and map it to different names from different DS (e.g., `userid`, `uid`, etc.)

```
...
<map>
  <!-- Each Entry has two properties:
        * name - the name of the attribute in the ldap
        * value - the name of the attribute it has to be
                mapped to
        -->
  <entry>
    <name>dndUid</name>
    <value>userIdentifier</value>
  </entry>
</map>
...
```

Browser Request (Cont.)

```
...  
<input>  
  <!-- Name of the Input Field -->  
  <name>loa</name>  
  <!-- Label displayed to the User -->  
  <label>Level of Assurance</label>  
  <!-- Info Image and Link (right of Input) -->  
  <info img="bulb.png">?cmd=viewLoas</info>  
  <!-- Type of input (eg., textfield, select,  
        popup_menu, checkbox, etc...) -->  
  <type>select</type>  
  <charset>UTF8_LETTERS</charset>  
  <value></value>  
  <minlen>1</minlen>  
  <required>YES</required>  
</input>  
...
```

Authenticated Browser Request

- Same Characteristics as the normal Request
- Requires authentication to a Data Source
 - Currently Supported only LDAP
- XML configuration
 - PREFIX/etc/openca/auth_browser_request.xml.template
 - Uses Datasources to determine the list of available data source for authentication and data retrieval purposes
- **\$DATA::[FIELD]** - substitute the value with the FIELD value gathered from the chosen datasource:
 - e.g., 'givenName' available as \$DATA::giveName.

Authenticated Browser Request

- Same Characteristics as the normal Request
- Requires authentication to a Data Source
 - Currently Supported only LDAP
 - Retrieves data from the Data Source after authentication

Firefox 3 Certificate Request (Linux) - Login

Please enter your credentials in order to verify your identity.

Login Information

Login	<input type="text" value="pala"/>
Password	<input type="password" value="....."/>
Home Institution	<input type="text" value="Dartmouth College"/>



Back

Continue



LOA and License Agreement

LOAs and License Agreements

■ Level of Assurance

- Different LOAs require different request properties
- Different LOAs require different user behaviors

■ Configuring LOAs

- PREFIX/etc/openca/loa.xml

■ Three sub-sections

- Requires – defines the requirements for the LOA
- Agreement – points to the Agreement file to be displayed during the request process
- Cert (i.e. CP and CPS) – defines the CP/CPS extensions to be used for the LOA (typically the pointer to the CP/CPS)

LOA Configuration (cont.)

```
<loa>
  <level>1</level>
  <name>Low</name>
  <description> ... </description>
  <requires>
    <!-- The requires defines the accepted algo+bits -->
    <strength>
      <name>Weak</name>
      <allowed>RSA+512</allowed>
    </strength>
    <!-- Keygen:
      * mode - generation mode (eg., Browser or Server).
    -->
    <keygen>
      <mode>Server (Our Systems)</mode>
    </keygen>
  </requires>
  <agreement>...</agreement>
  <cert>...</cert>
</loa>
```

LOA Configuration (cont.)

```
<cert>
  <ext>
    <name>certificatePolicies</name>
    <!-- list all the policy OIDs here -->
    <CP>
      <value>1.2.3.3.4</value>
      <value>@psec</value>
    </CP>
    <section>
      <name>psec</name>
      <policy_ID_tag> policyIdentifier</policy_ID_tag>
      <!-- you can have multiple CPS URIs exmple -->
      <CPS>
        <URI>
          CPS.1 ="http://host/pki/pub/cps/rudimentary"
        </URI>
      </CPS>
    </section>
  </ext>
</cert>
```

Cryptographic Updates

■ Support for ECDSA

- Only with ECDSA enabled openssl
- No encryption is supported (as in DSA)
- Smaller Keysizes (128 -> 521 bits)
- Subset of curves supported (standard curves – NIST)
- No browsers support ECDSA requests
 - use Server Side generation
- Support for ECDSA certificates in browsers still to be fully tested

■ Support for SHA256

- Enabled by default on OpenCA 1.0.x
- Absolutely needed – 2010 is deadline for SHA1!

OpenCA: Next Steps (cont.)

- Integrating openca-tools with libpki
 - Take advantage of the extended support of algorithms and token interface
 - Easier integration with HSMs
 - Porting to mobile devices (eg., Smart Phones)

- Start of the OpenCA-DigiSign project
 - Beginning in April 2009
 - New C-Based codebase
 - Integrated with LibPKI
 - Commercial Support available



LibPKI

LibPKI

- Easy-to-use PKI Library
 - Easy to integrate PKI functionalities (Certificate Management) in applications
 - Easier integration with HSMs
 - Provides native PKCS#11 support
 - ANSI C (portable on any UNIX system)
 - Uses OpenSSL
 - Future integration with other CSP
 - Will be used as crypto lib for all OpenCA Projects
 - OCSP Server, PRQP Server, OpenCA-NG
 - Native support for PRQP
 - Provides pki-tool and programming examples

LibPKI usage Example

```
#include <libpki/pki.h>

int main(int argc, char *argv[] ) {
    PKI_TOKEN *tk1 = NULL;
    PKI_X509_REQ *req = NULL;
    PKI_X509_CERT *cert = NULL;

    /* Allocate the Data structures for a PKI_TOKEN */
    tk1 = PKI_TOKEN_new_null()
    /* Init the TOKEN by loading the argv[1] config file */
    PKI_TOKEN_init ( tk1, NULL, argv[1]);
    /* Loads a Request from URI */
    req = PKI_X509_REQ_get ( "file://req.pem", NULL, NULL );
    /* Set the request inside the TOKEN (tk1) */
    PKI_TOKEN_set_req( tk1, req );
    /* Issue a Certificate by using the TOKEN */
    cert = PKI_TOKEN_issue_cert ( tk1, "CN=Test", "0",
                                   60, NULL, NULL);

    Return (1);
}
```

LibPKI::pki-tool

```
# Provides information about the token
$ pki-tool info -hsm etoken-pkcs11

# clears the contents of the device/token
$ pki-tool clear -hsm etoken-pkcs11

# Generates a keypair and a request (req.pem)
$ pki-tool genreq -hsm etoken-pkcs11 -newkey -bits 1024 \
  -label "id://OpenCA Test" -subject "DC=org, DC=openca, \
  O=OpenCA, OU=Internet, CN=localhost" -out req.pem

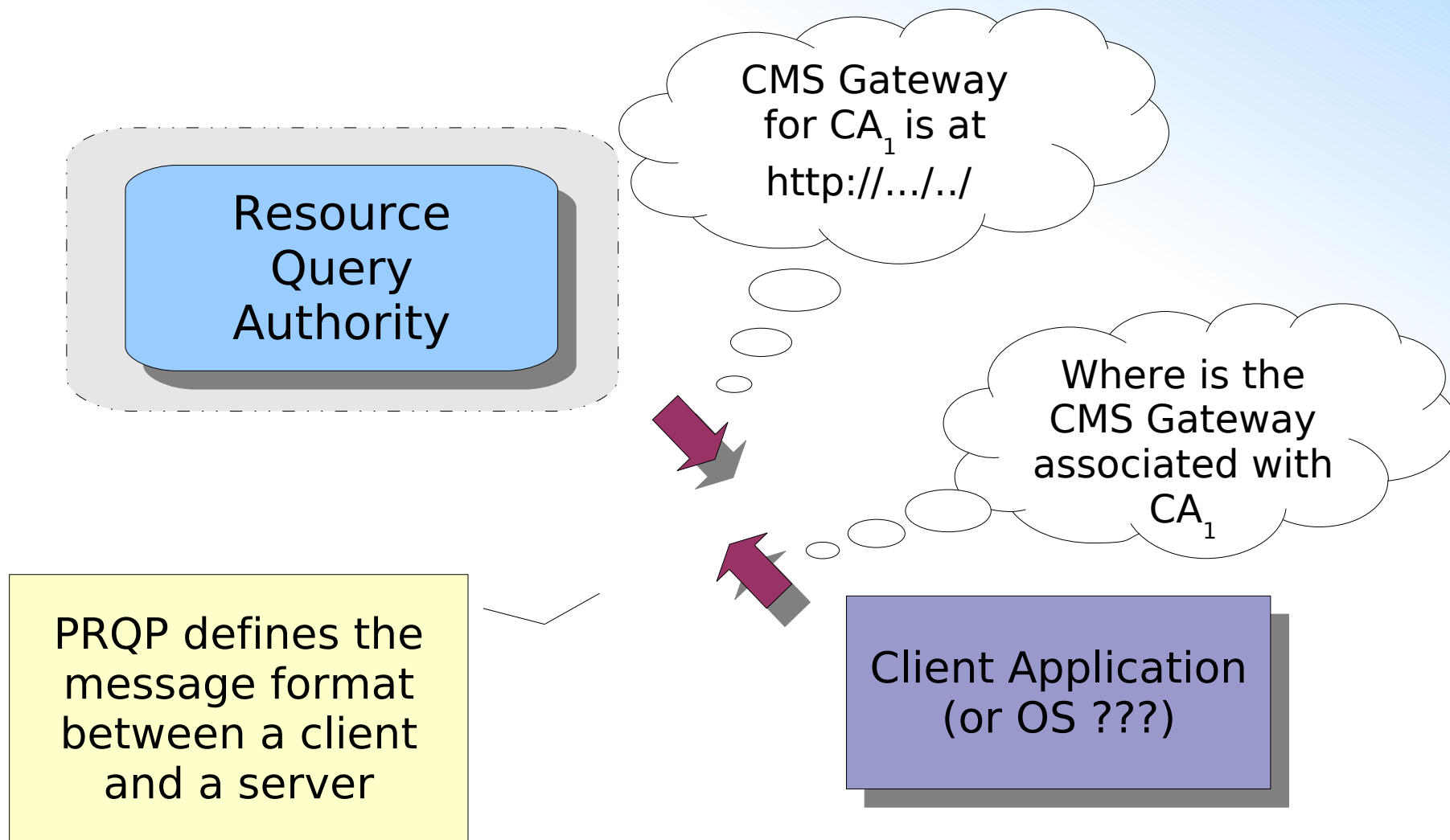
# Generates a self-signed certificate (cert.pem)
$ pki-tool gencert -hsm etoken-pkcs11 -in req.pem \
  -out cert.pem -label "id://OpenCA Test" -selfsign

# Imports the Self-Signed certificate into the token
$ pki-tool import -hsm etoken-pkcs11 -type usercerts \
  -label "id://OpenCA Test" \
  -in cert.pem
```



PKI Resource Query Protocol (PRQP)

PKI Resource Discovery Protocol



OpenCA-PRQPD

- Provides a server
 - Uses LibPKI for PRQP and PKI operations
- Includes the PRQP client (pclient)

```
$ bin/pclient -cacert <cacert-file> -connect <URI> \  
[ -service <service> ... ]
```

- Usage example:

```
$ bin/pclient -cacert openca.pem \  
-connect http://prqp.openca.org: 830 \  
-service endorsedTA \  
-service timeStamping
```



On-Line Demo...

Questions



Questions



Thank You!

contribute

contribute

contribute

contribute

contribute

contribute

contribute

contribute

contribute

contribute

contribute

contribute



OpenCA References

- OpenCA HomePage:

<https://www.openca.org>

- Main Contact:

project dot manager at openca dot org