



**Cynops**  
network security engineering



# Building an Open Source PKI using OpenXPKI

by Alexander Klink and Michael Bell

# OpenXPKI



1. Some theory (concepts) by Michael
2. Some practical experiences (concepts + demo) by Alex

# Concepts

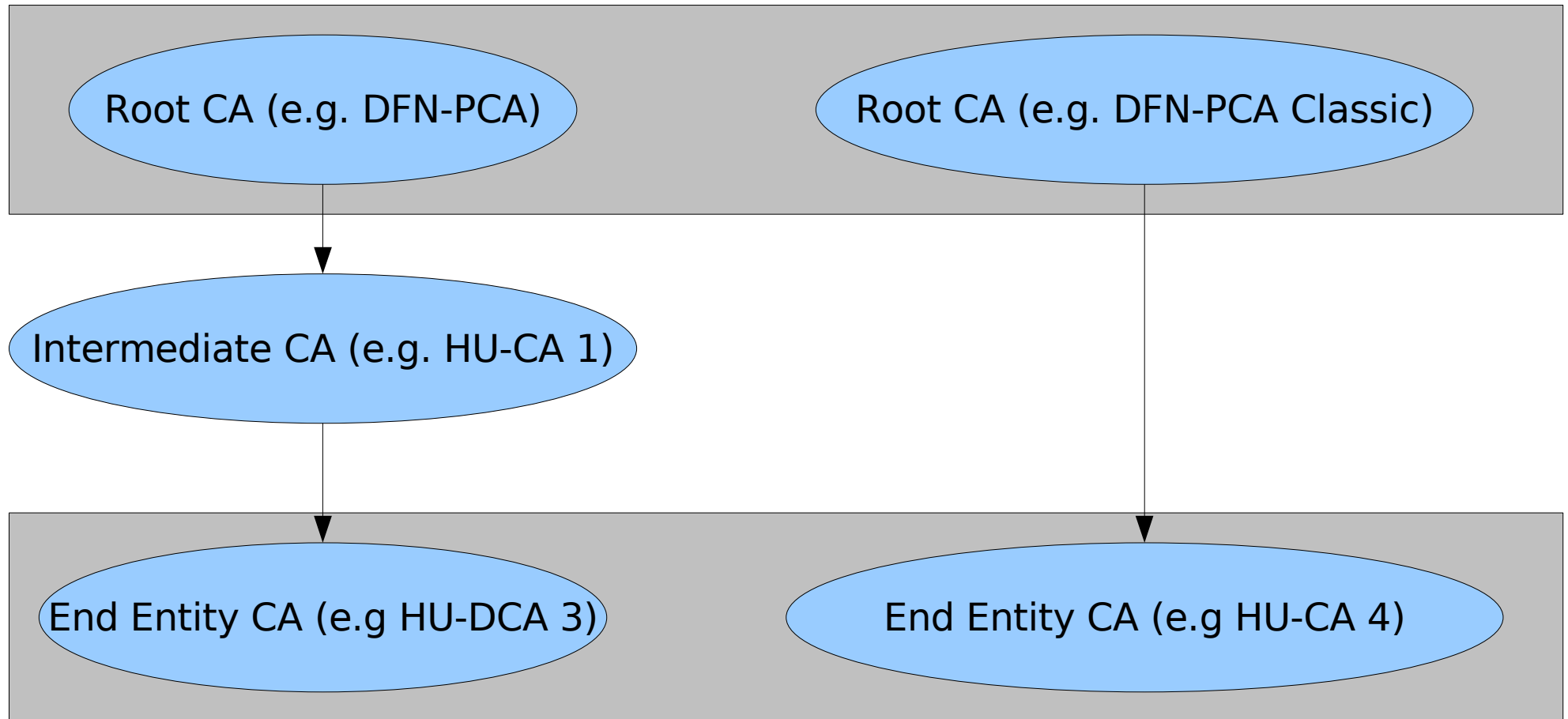


- PKI Realm
- Crypto Abstraction
- Configuration Inheritance
- Workflow

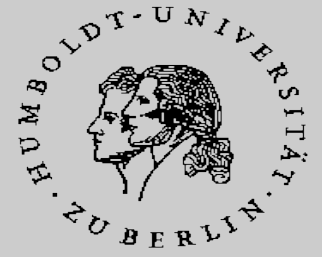
# Concepts – PKI Realm



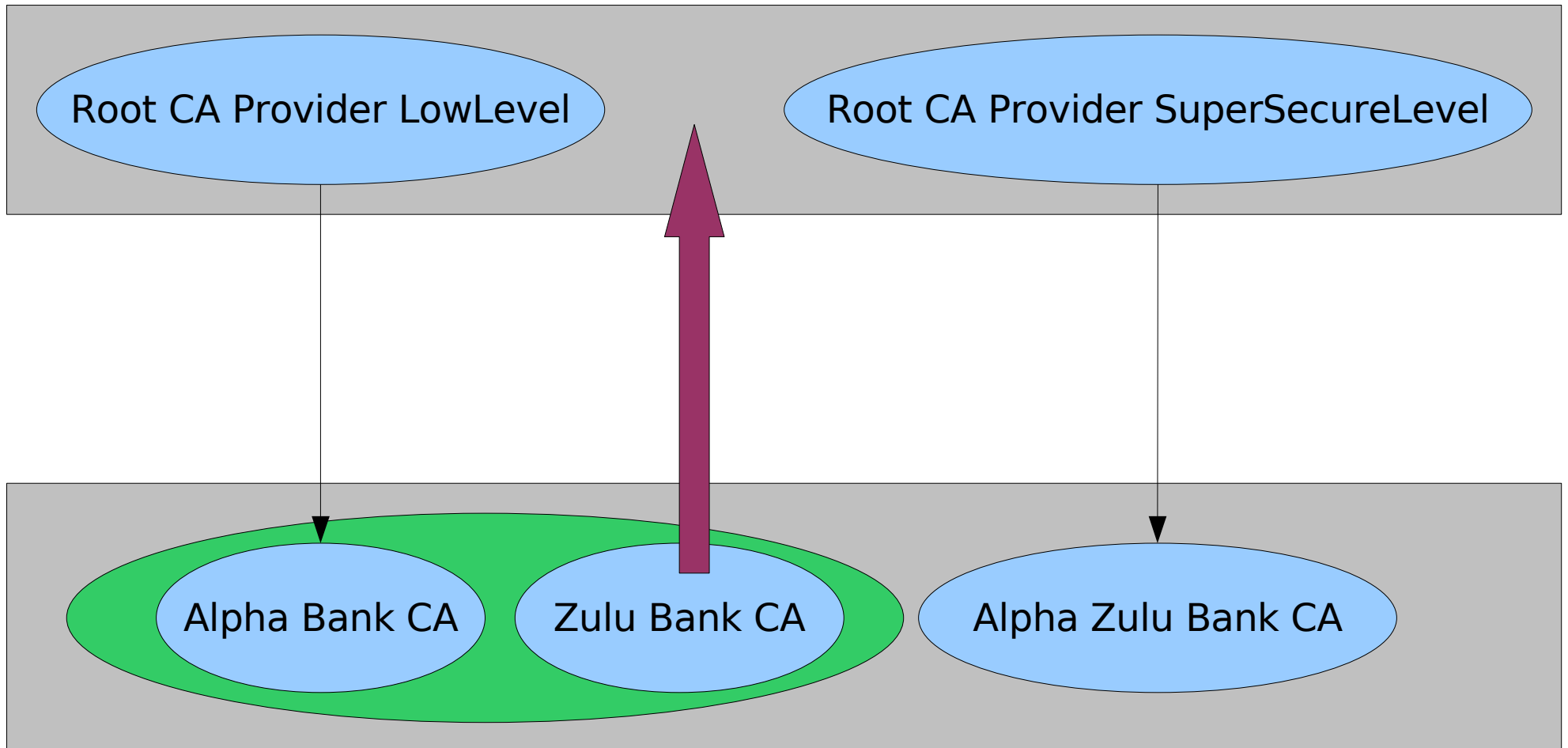
## CA Migration



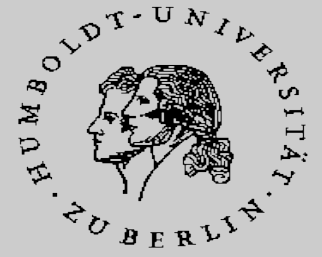
# Concepts – PKI Realm



## Merger



# Concepts – PKI Realm

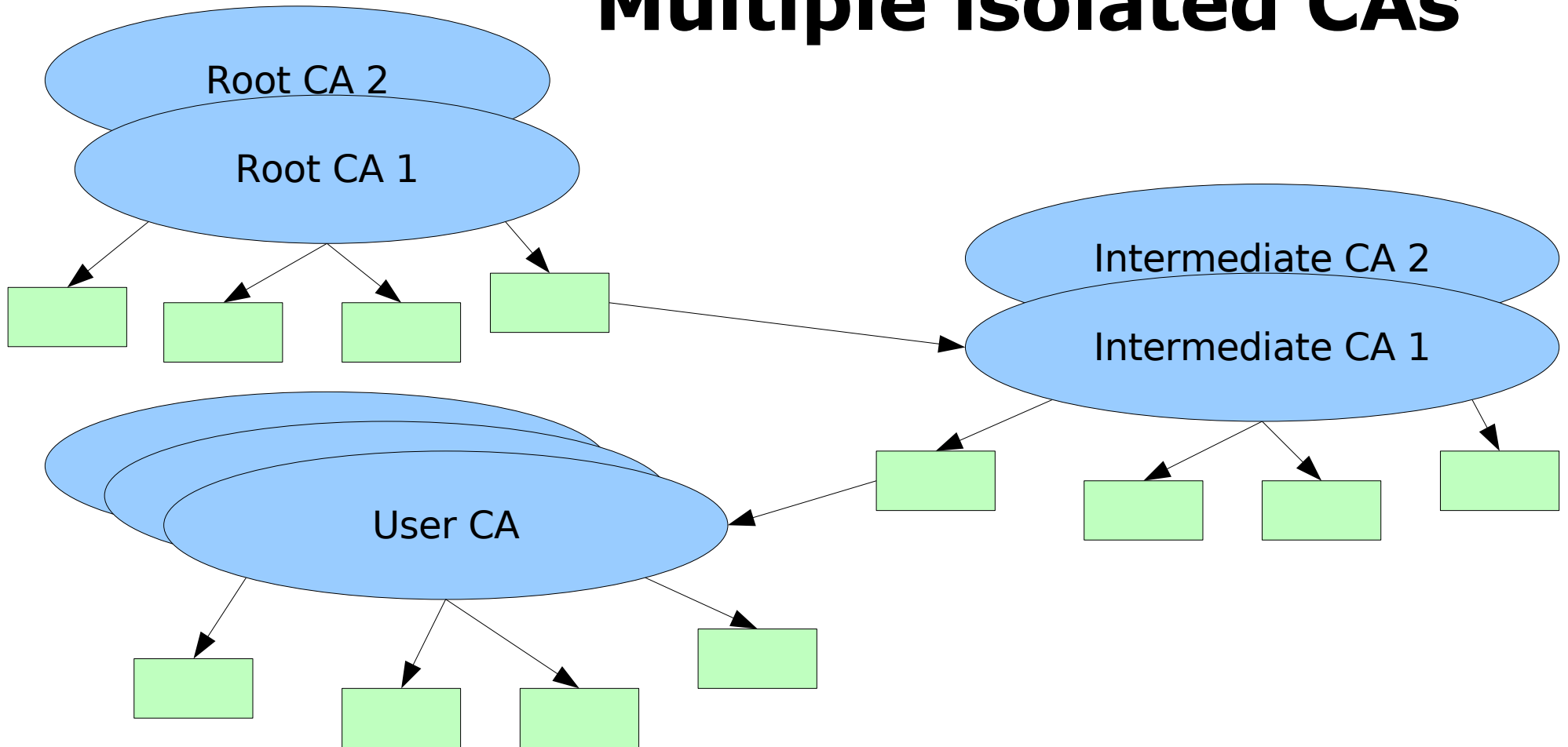


- You never want to lose certificates and CSRs because of a migration.
- Design or time-based rollovers should not influence the operations.
- How to (and why) import a PKI?
  - with/without CA key
- We simply need a playground ;-D

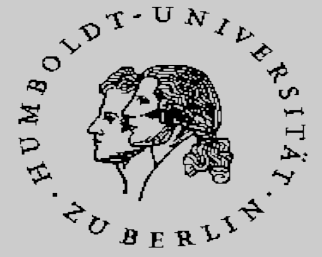
# Concepts – PKI Realm



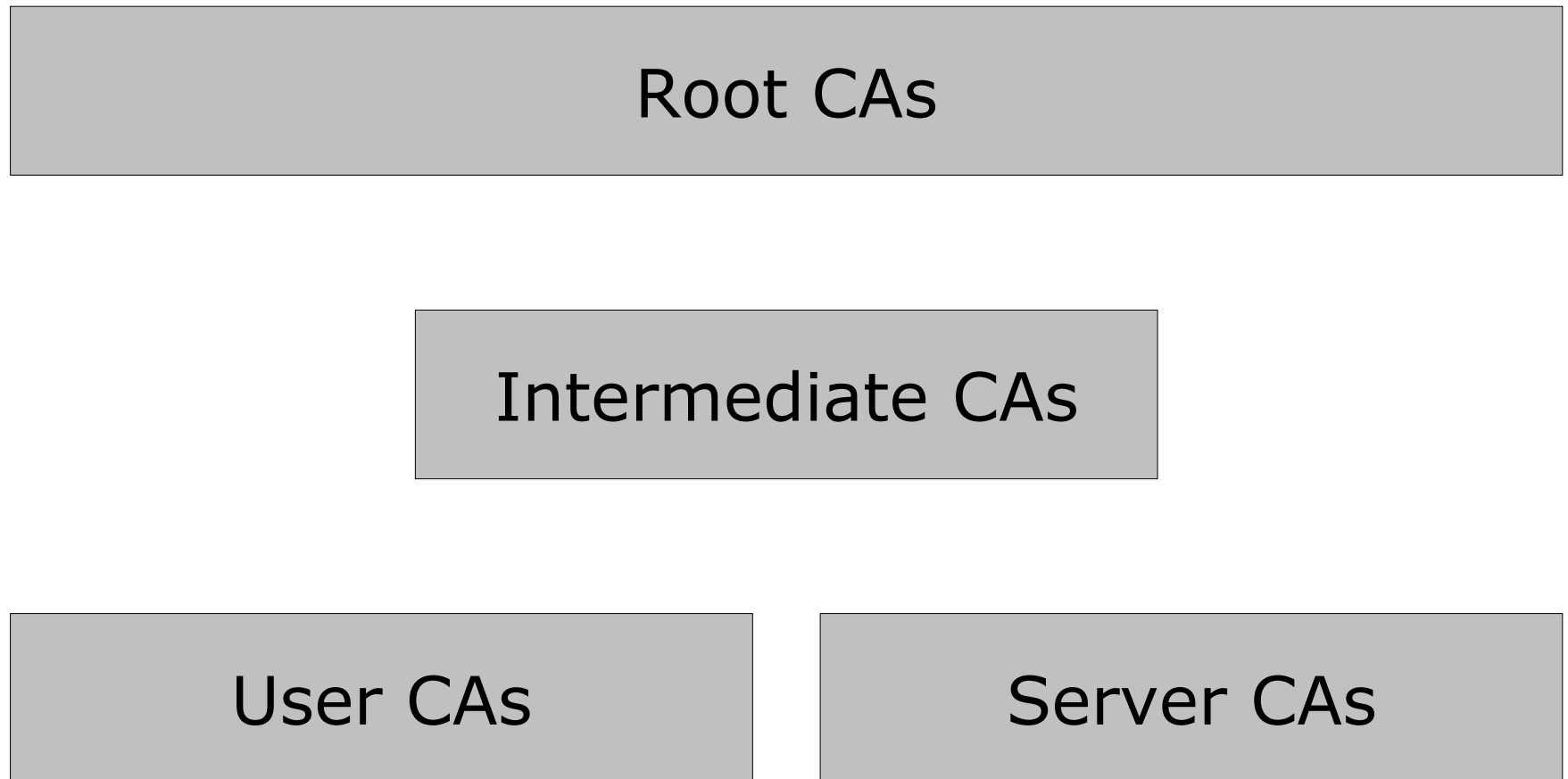
## Multiple isolated CAs



# Concepts – PKI Realm

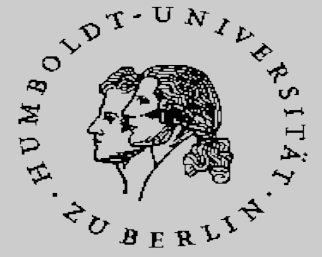


## Domain/Group Concept



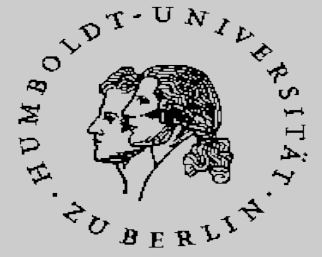


# Concepts – PKI Realm



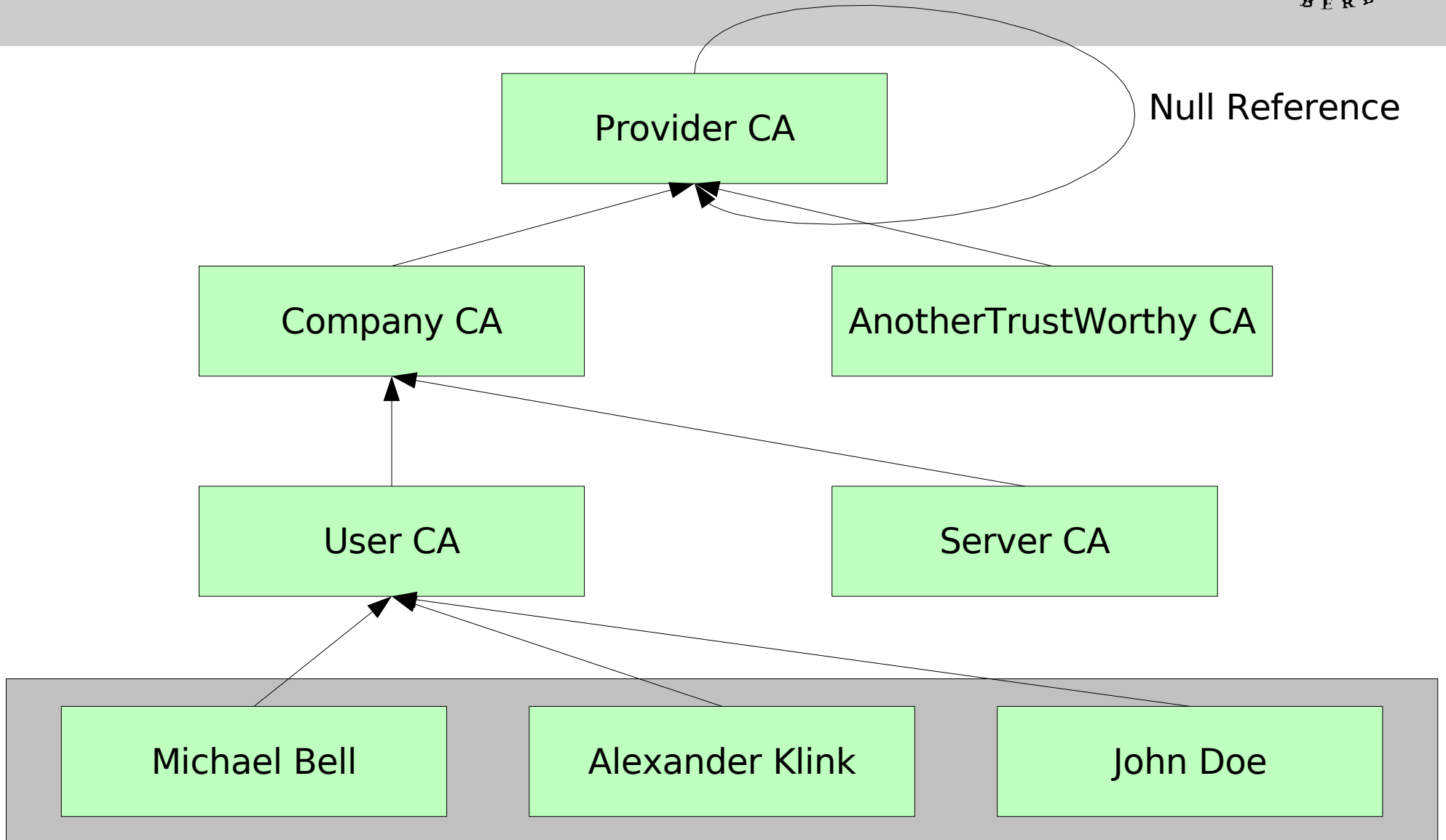
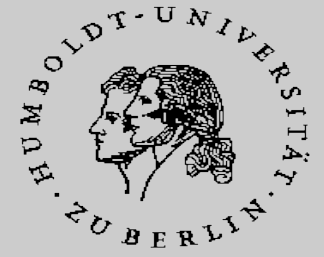
- PKI Realm Definition
  - A PKI realm is a namespace for CSRs, certificates, CA certificates, CRLs and any other PKI related information.

# Concepts – PKI Realm

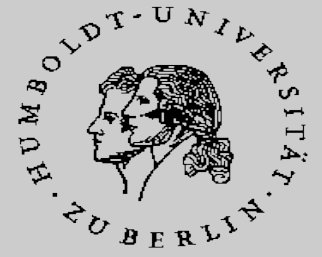


- Certificate Storage Definition
  - Every certificate exists only once.
  - Every certificate references its issuer via a SHA-1 hash of the issuer's certificate.
  - Every certificate is a member of exactly one PKI realm.

# Concepts – PKI Realm



# Concepts - PKI Realm



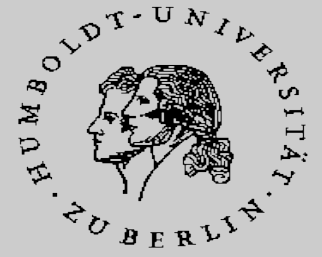
- Additional implementational sugar
  - aliases
  - special CA certificate handling

# Concepts – PKI Realm



- Alias Definition
  - An alias is a reference to a certificate.
  - An alias consists of a name, a SHA-1 hash and a PKI realm.
  - Every alias is only valid inside of exactly one PKI realm.

# Concepts – PKI Realm



- CA Handling

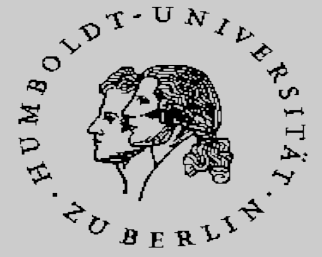
- Every CA Certificate must have set the PKI realm of its issuers PKI realm.
- The CA cert(s) inside of a PKI realm are only present as aliases.
- All self-signed (CA) certs are in a special PKI realm called "" (empty word).

# Concepts – PKI Realm

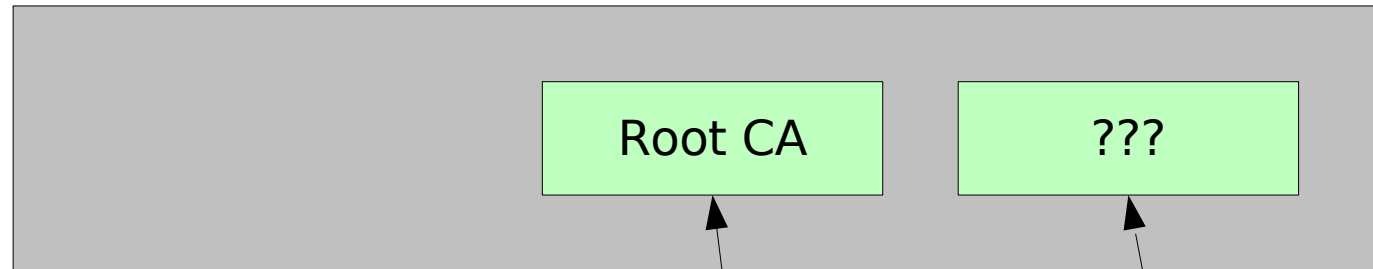


**(Too) Heavy stuff?**

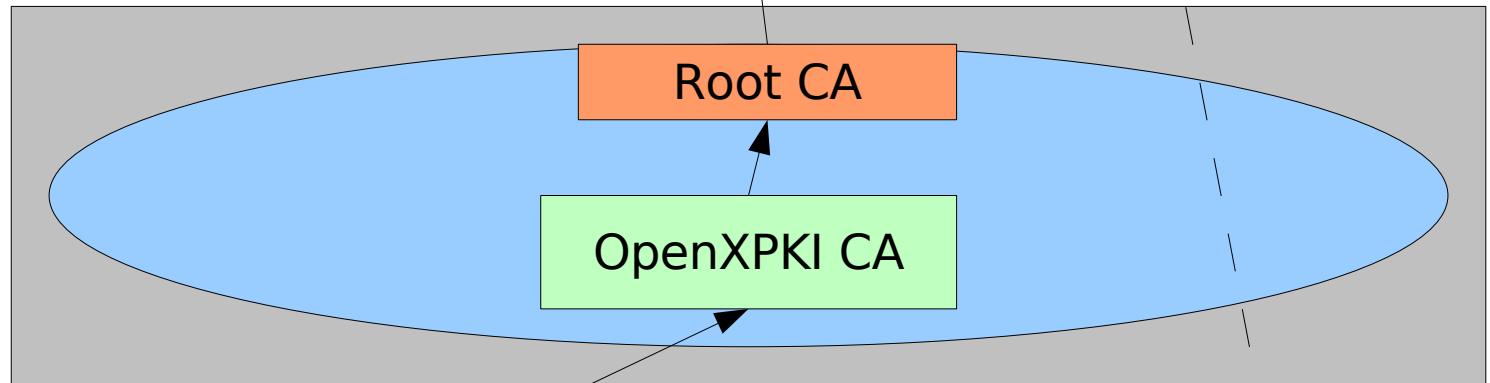
# Concepts – PKI Realm



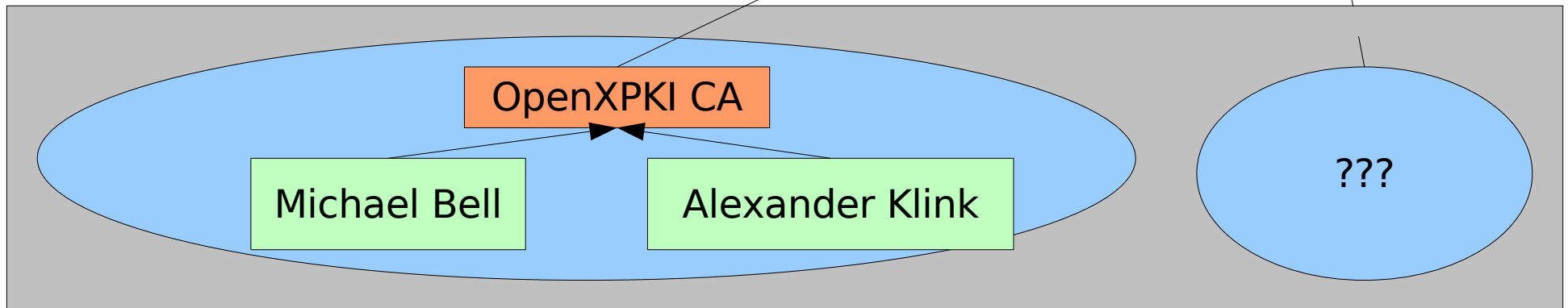
special PKI realm ""



Company CAs

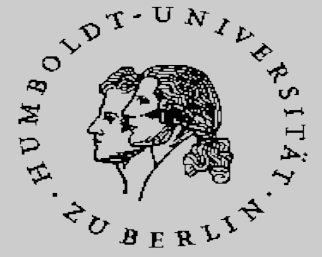


User CAs

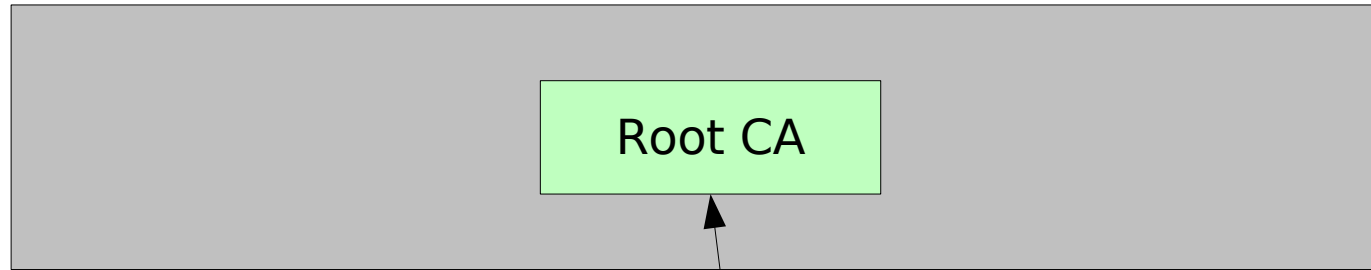




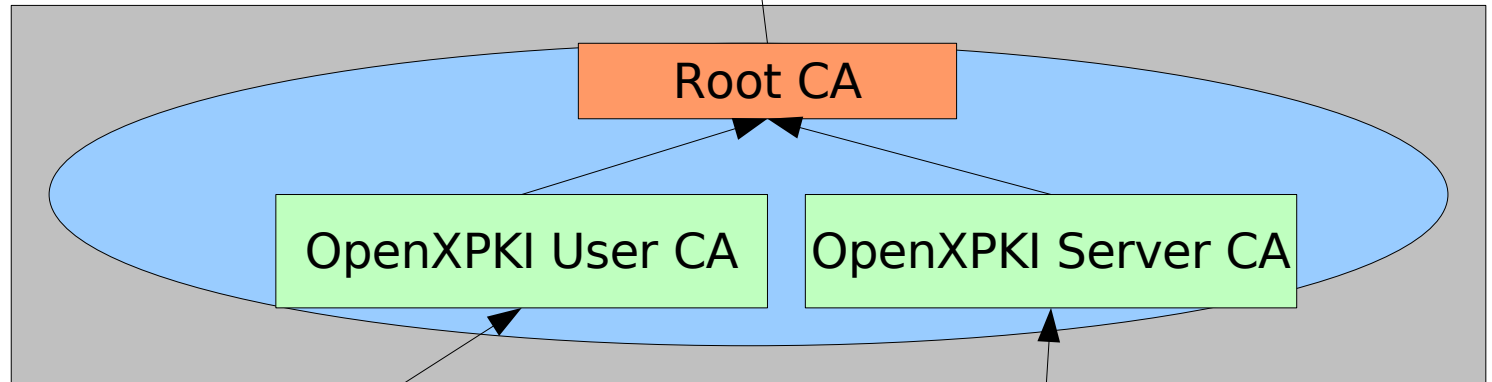
# Concepts – PKI Realm



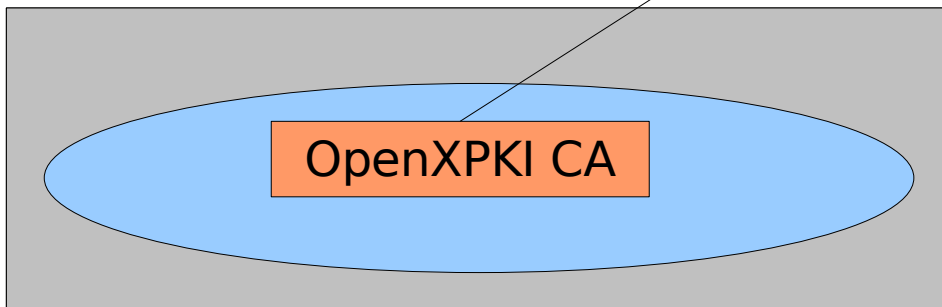
special PKI realm ""



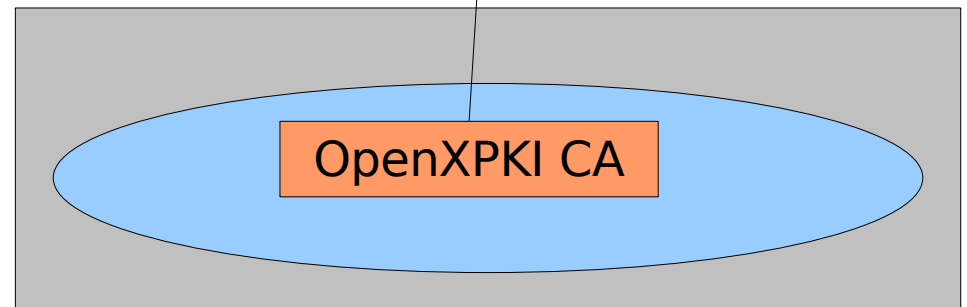
Company CAs



User CAs

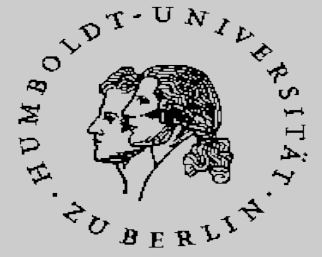


Server CAs

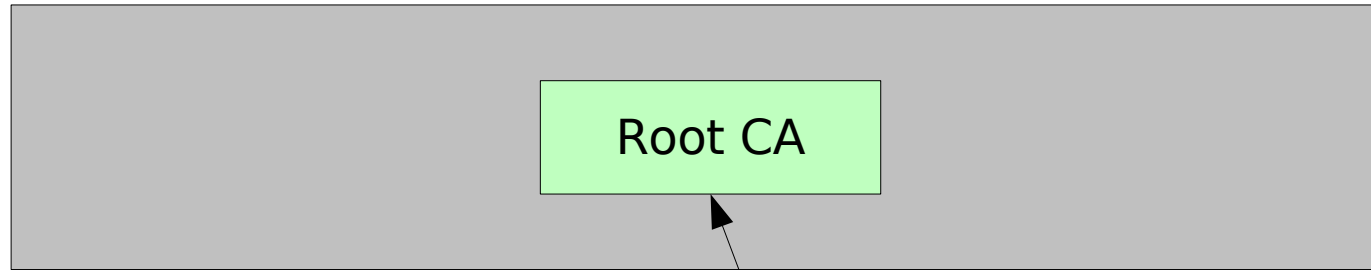


# Concepts – PKI Realm

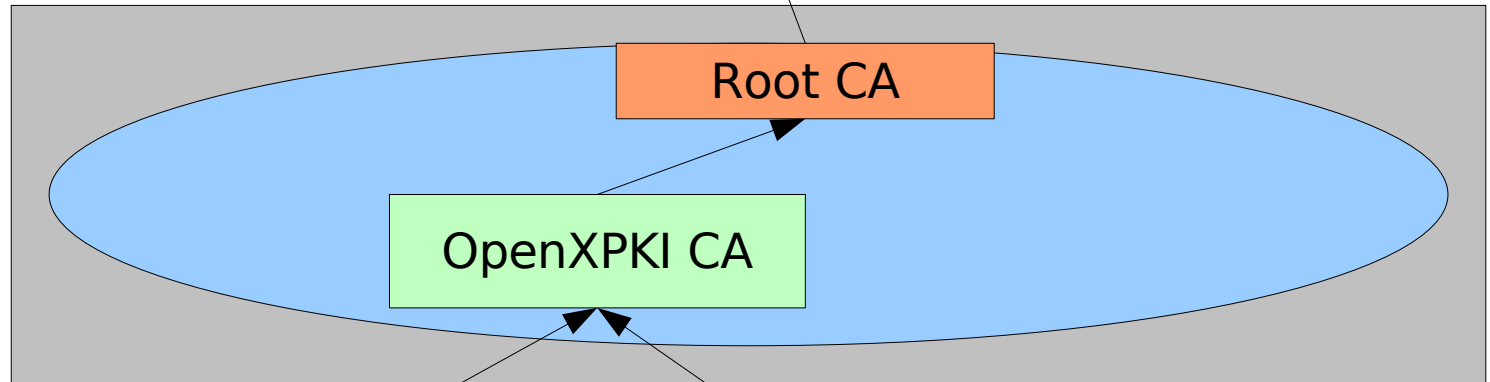
Do not try this at home!!!



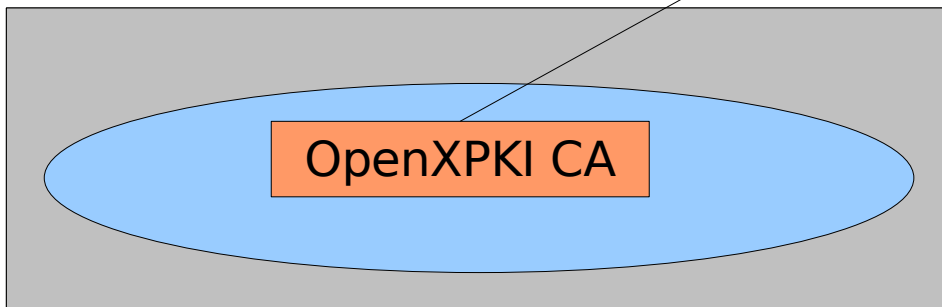
special PKI realm ""



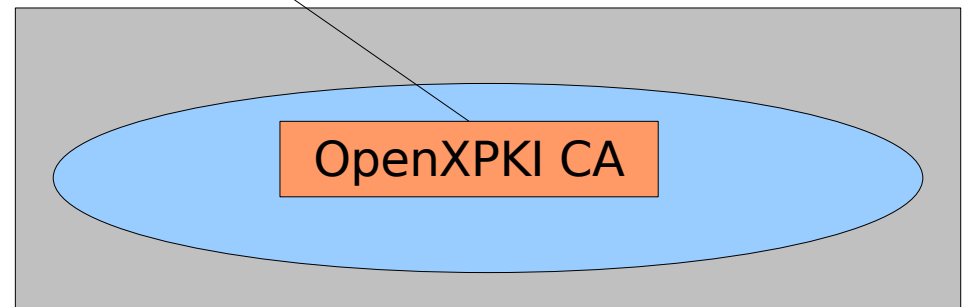
Company CAs



User CAs



Server CAs

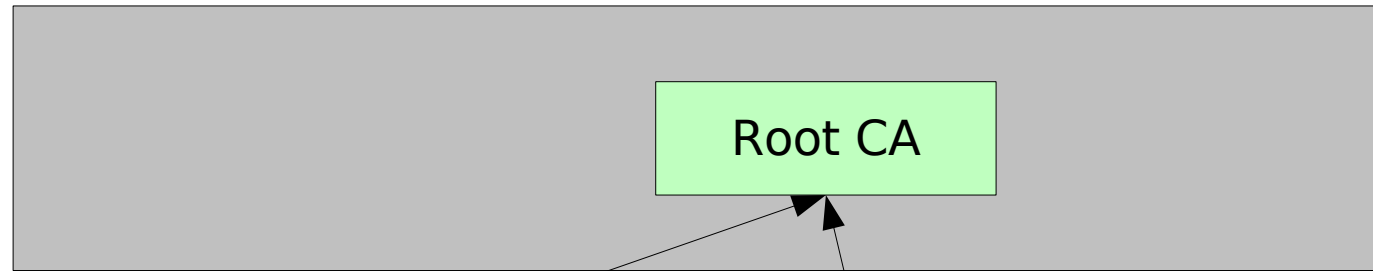


# Concepts – PKI Realm

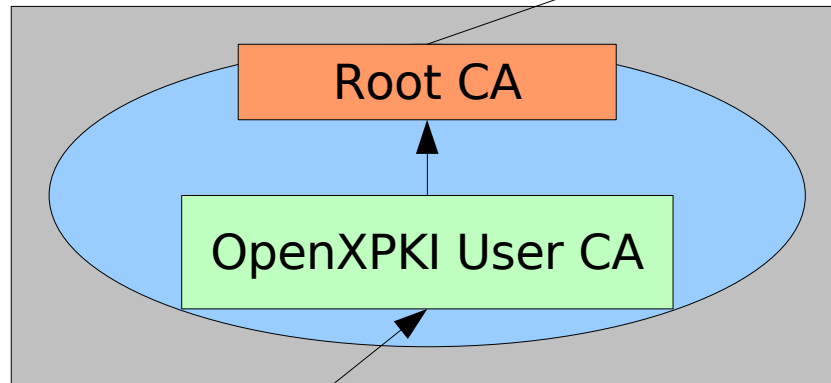
Do not try this at home!!!



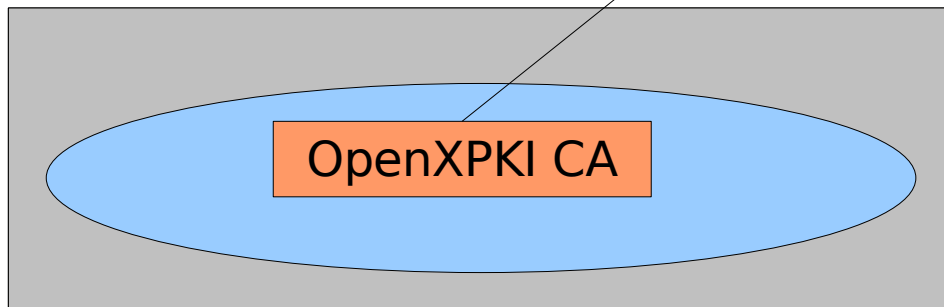
special PKI realm ""



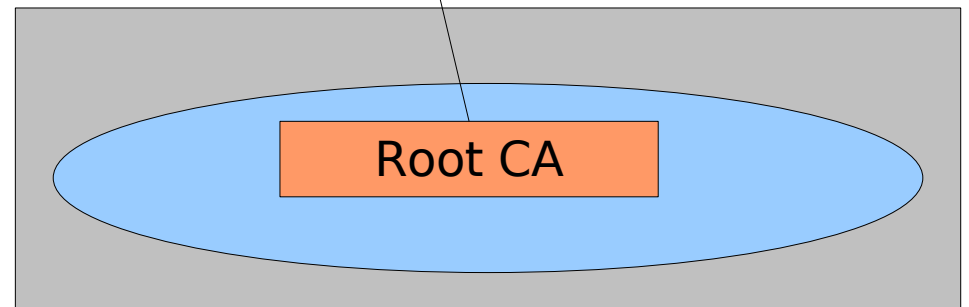
Company CAs



User CAs



Server CAs

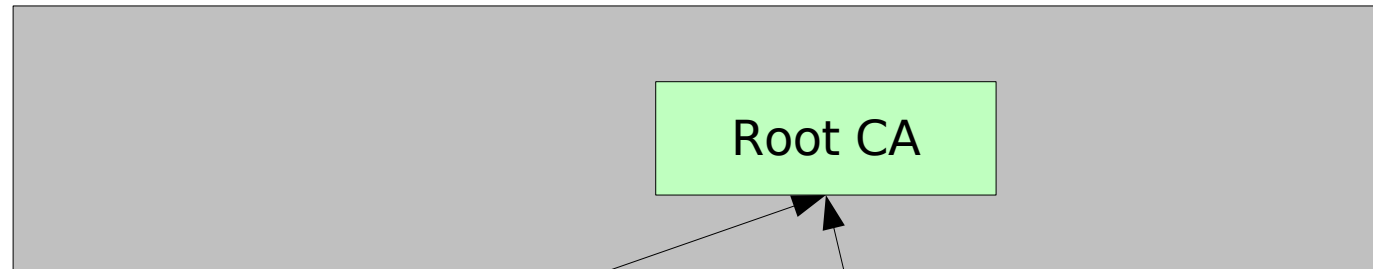


# Concepts – PKI Realm

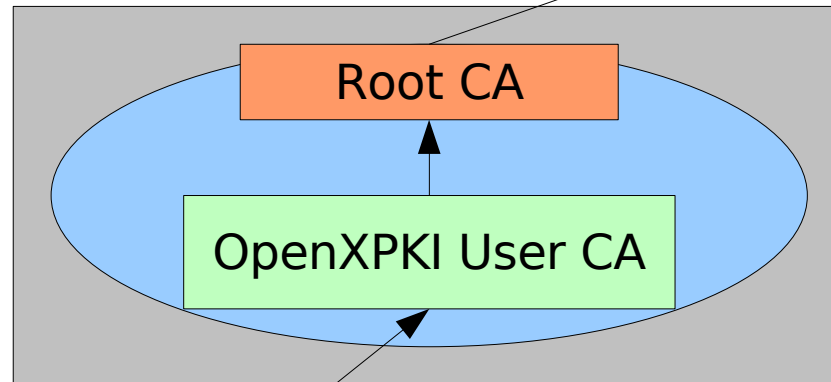
Do not try this at home^Wwork!!!



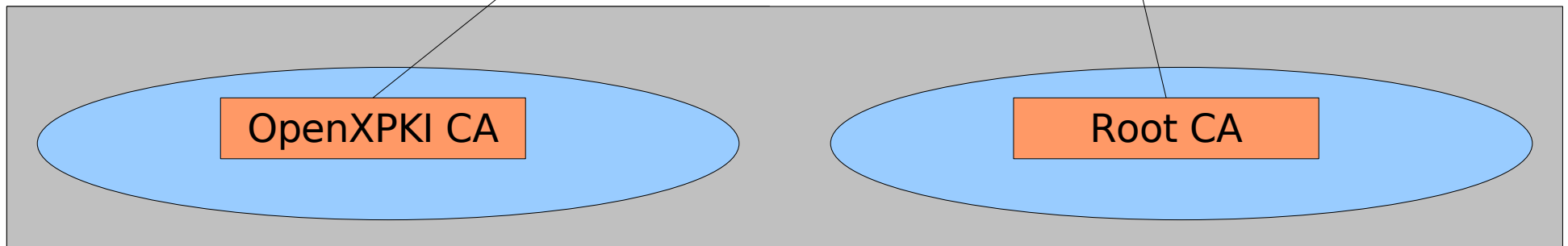
special PKI realm ""



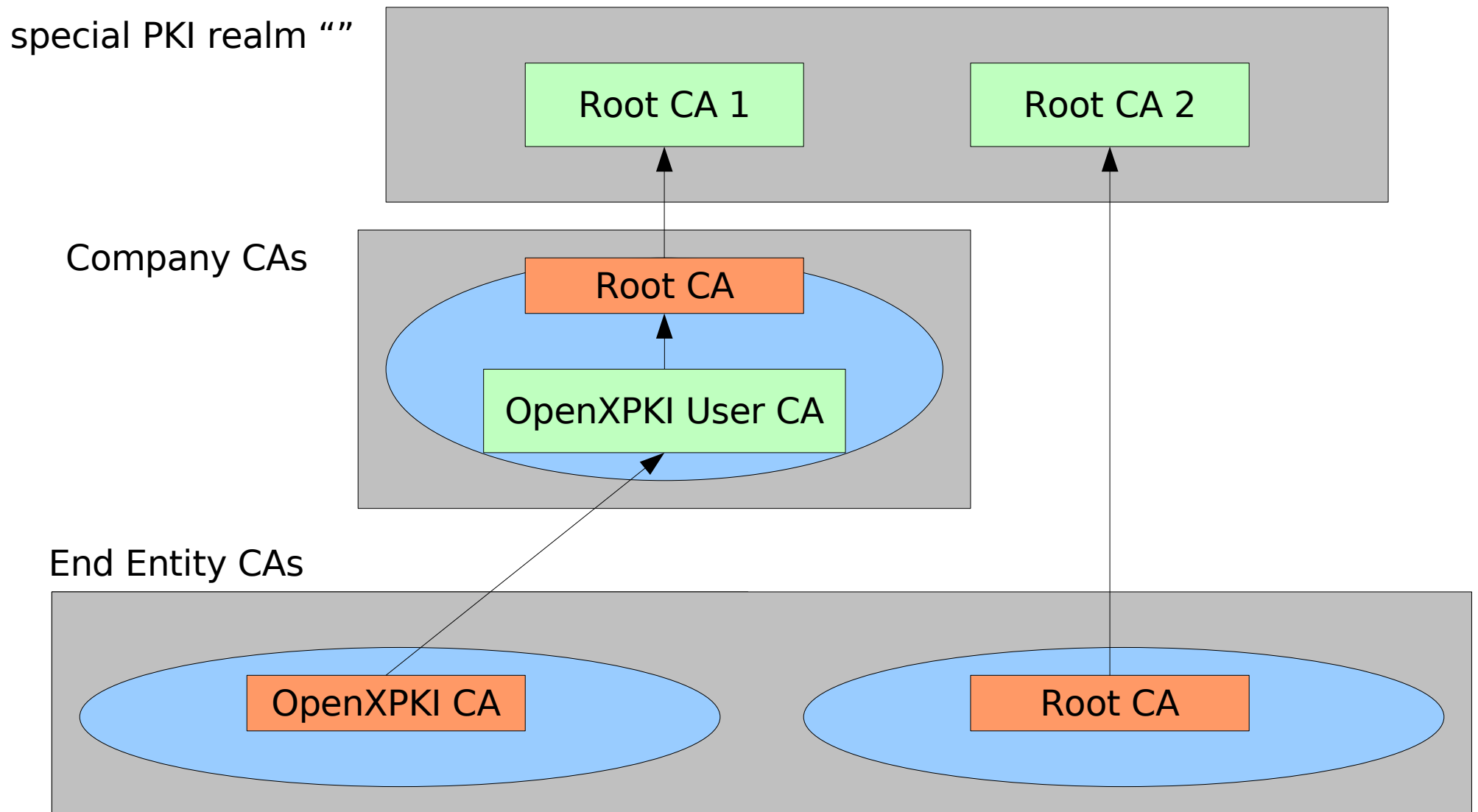
Company CAs



End Entity CAs

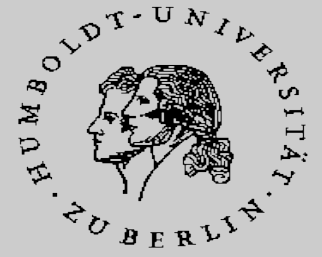


# Concepts – PKI Realm



# Concepts

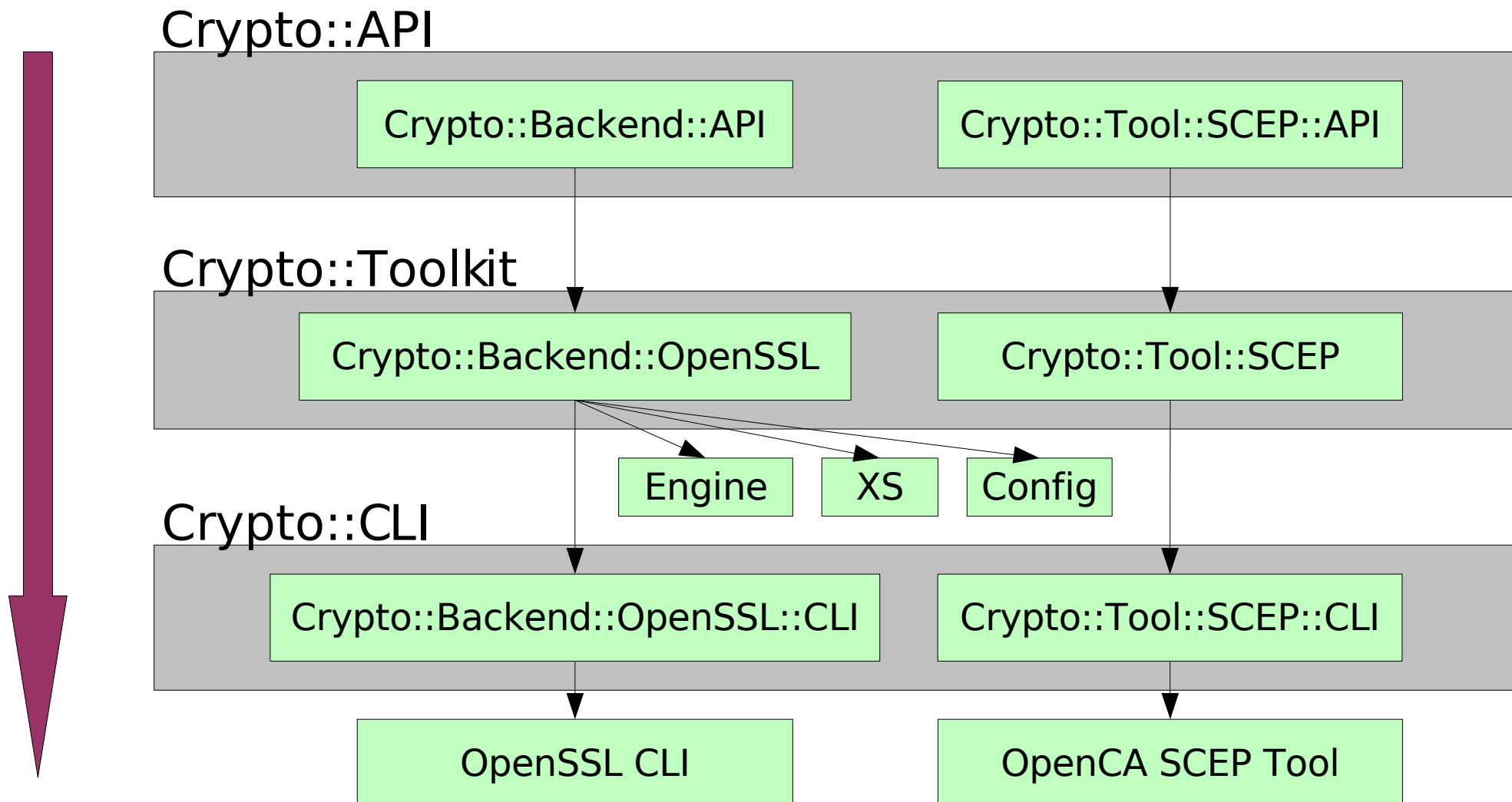
## Crypto Abstraction



- OpenSSL vs. OpenTLS vs. NSS
- patched toolkits (e.g. for GOST)
- different protection classes of used keys  
incl.  $m$  out of  $n$  passphrases for software  
keys (e.g. CA, SCEP, key backup)
- Hardware Security Modules (HSM)

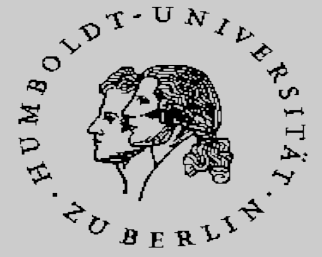
# Concepts

## Crypto Abstraction



# Concepts

## Crypto Abstraction

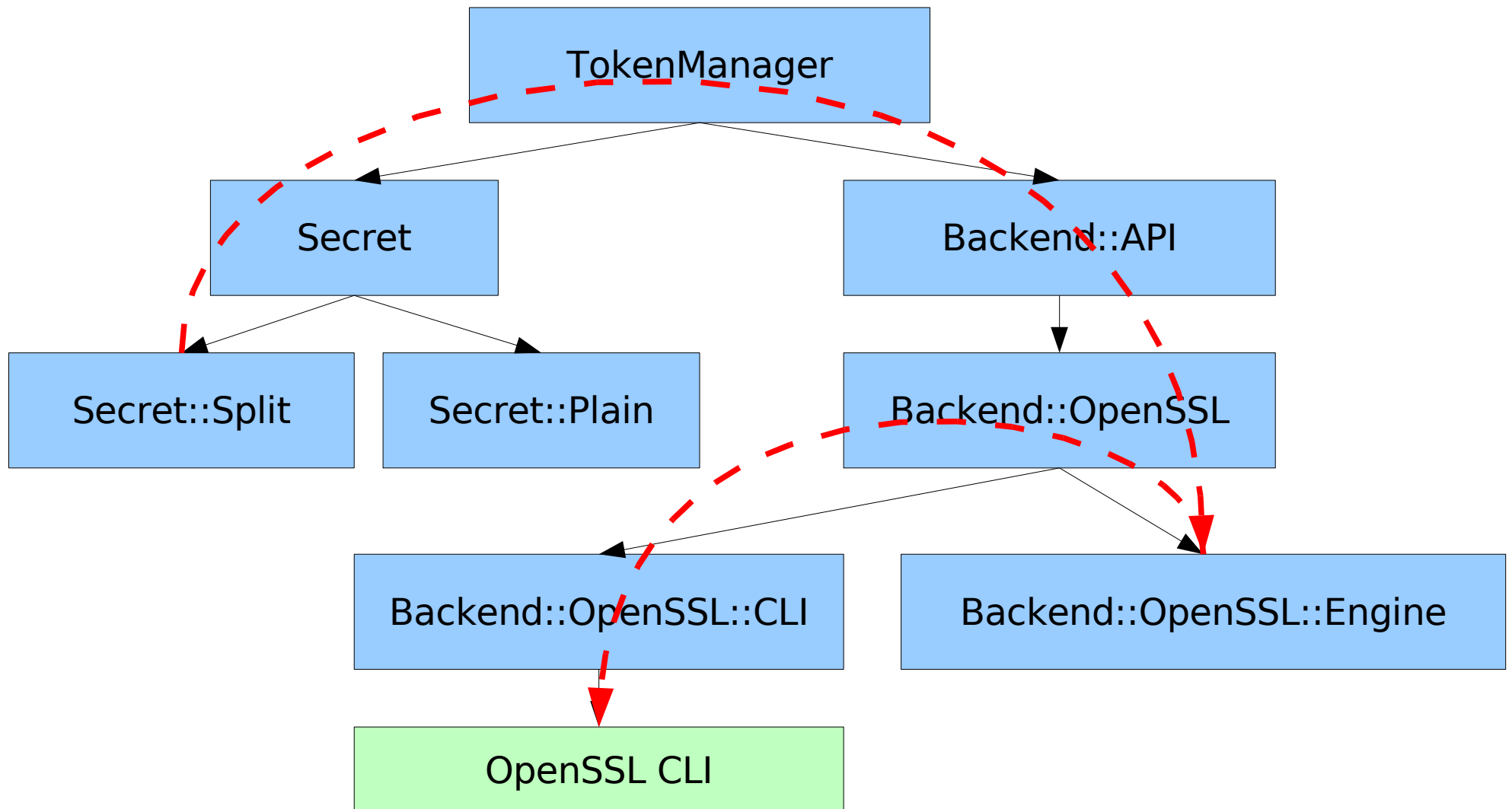
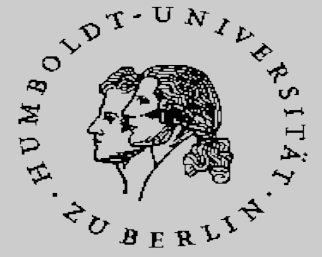


- Example: crypto algorithms
  - RSA
  - DSA/ECDSA
  - GOST algorithms
- Example: passphrase handling
  - hard coded
  - (splitted) plain
  - m of n as described by Adi Shamir
  - HSMs



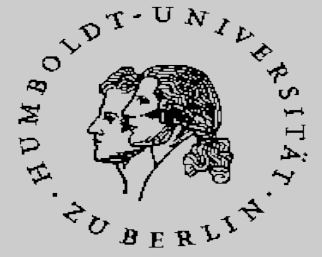
# Concepts

## Crypto Abstraction



# Concepts

## Config Inheritance



- Problems
  - many repetitions
  - new CA with only minimal changes
  - basic config and only small local additions
- Solution
  - configuration inheritance
  - attribute id ::= identify local nodes
  - attribute super ::= path to the parent

# Concepts

## Config Inheritance



```
<common>
```

```
  <test id="default">
```

```
    <value>My Test</value>
```

```
  </test>
```

```
<common>
```

```
<deep>
```

```
  <test super="../common/test{default}"/>
```

```
  <test super="/common/test"/>
```

```
</deep>
```

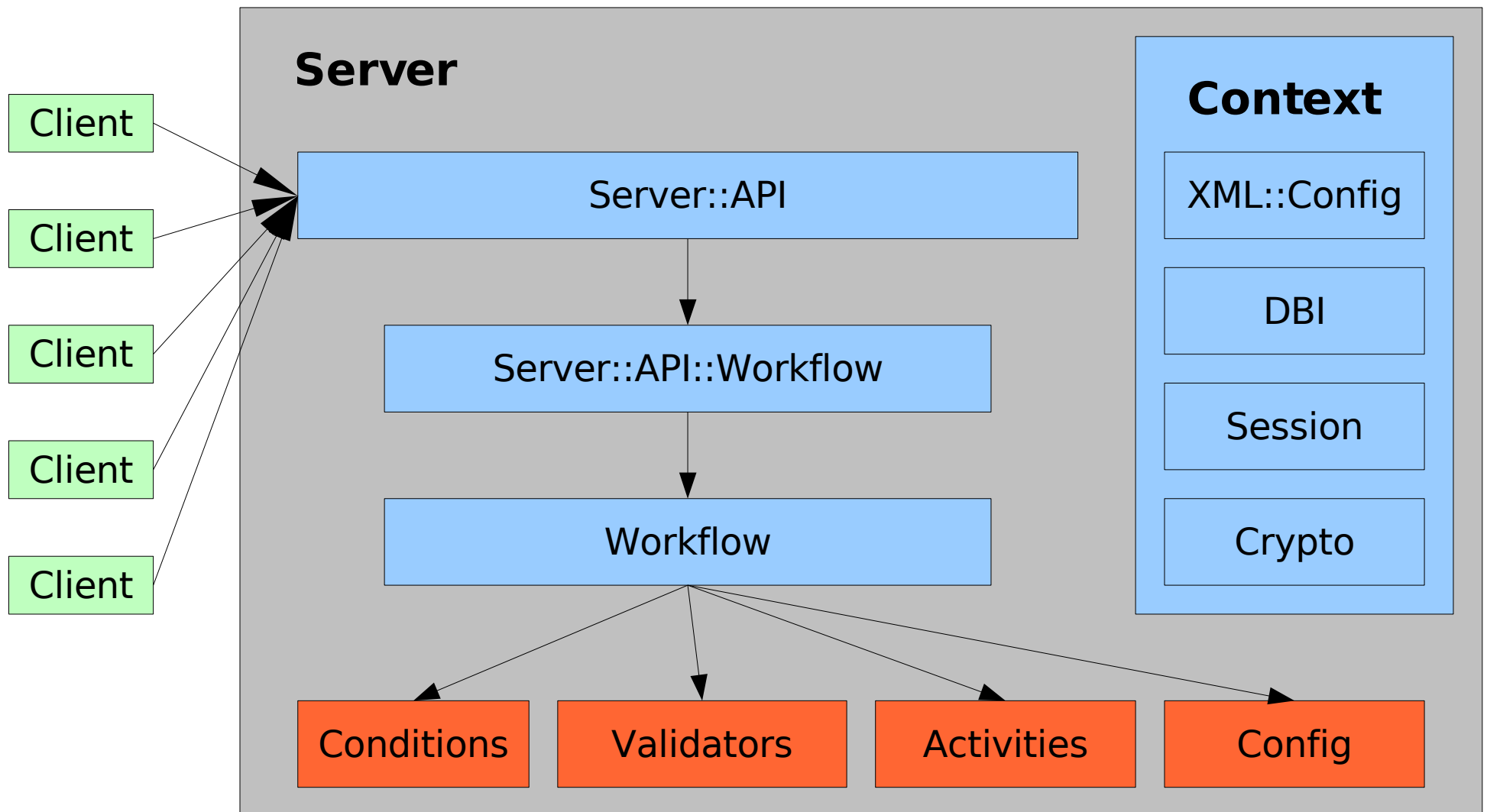
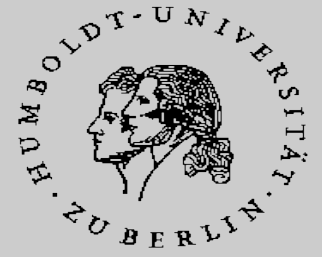
# Concepts - Workflow



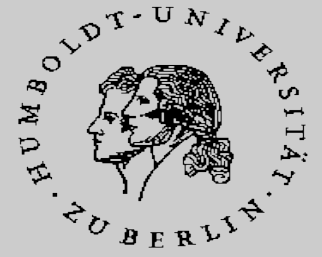
- Which problems are the most common ones with trustcenter software?
- Cool demo systems, BUT ...
  - We need the following customization(s) ...
  - We have an ERP/HR/CRM system XYZ ...
  - My business process is the following one ...

**... AND THEN THE HORRORS BEGIN ...**

# Concepts - Workflow

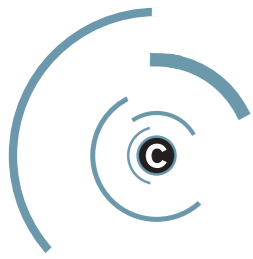


# Concepts – Workflow



How does this design help?

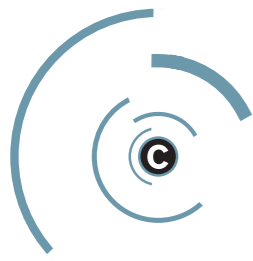
**Please give Alex a chance ;-D**



# Structure

## from abstract to concrete

- Workflows – example Certificate Request
- Simple Certificate Enrollment Protocol (SCEP)
- Smartcard personalization
- Support
- Hackers wanted!

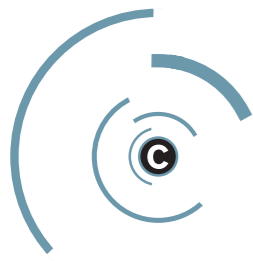


# Workflows

## Example: Certificate Request

- How does it look from a user perspective?
- Certificate Request using a web interface
  - browser-based request (SPKAC/XEnroll)
  - PKCS#10 upload
  - key generation on the CA
- Let's have a look

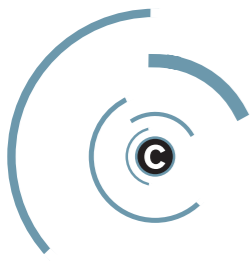




# Workflows

## Certificate Request: User Experience

The screenshot shows a web browser window titled 'OpenXPKI' with the address bar containing 'http://127.0.0.1:8080/service/create\_csr/index.html'. The page header features the 'OpenXPKI' logo and a search bar. A navigation menu includes 'CA Info', 'New Request (CSR)', 'Search certificate(s)', 'Language', and 'Logout'. The main content area is titled 'Choose a certificate profile' and contains the instruction 'Please choose the certificate profile for your certificate.' Below this is a dropdown menu set to 'TLS Server profile', with 'OK' and 'reset' buttons. The footer of the page displays '© 2006 The OpenXPKI Project'. The browser's status bar at the bottom left shows the word 'Fertig'.



# Workflows

## Certificate Request: User Experience

The screenshot shows a web browser window titled 'OpenXPKI' with the address bar containing 'http://127.0.0.1:8080/service/create\_csr/index.html'. The page header features the 'OpenXPKI' logo and a search bar. A navigation menu includes 'CA Info', 'New Request (CSR)', 'Search certificate(s)', 'Language', and 'Logout'. The main content area is titled 'Select the type of key generation' and contains the following text: 'Please choose one of the following available options for the key generation. This is important because this influences the next steps. The options mean quite different things. So please read the descriptions carefully.' Below this text is a dropdown menu with 'SPKAC' selected, and a list of options: 'SPKAC', 'Microsoft Internet Explorer Serverside key generation', and 'PKCS#10 Automatic browser detection'. A paragraph explains that the 'Automatic browser detection' option is used for Microsoft Internet Explorer. Below this is a section for 'SPKAC' with a detailed description: 'SPKAC was originally designed by Netscape. Today this format is used by several different browsers. Such browsers are Netscape, Mozilla, Firefox and Opera. If you have such a browser and you want to create the key using your client then please use this type of key generation.'



# Workflows

## Certificate Request: User Experience

The screenshot shows a web browser window titled "OpenXPKI" with the URL `http://127.0.0.1:8080/service/create_csr/index.html`. The page header features the "OpenXPKI" logo and a search bar. A navigation menu includes "CA Info", "New Request (CSR)", "Search certificate(s)", "Language", and "Logout".

### Creating the name of the certificate

The name of the certificate is built from the values of the following form fields. Please enter the required information. If you are not sure about what to enter then please read the descriptions of the fields at the end of this page.

uid =  + Common name =   
Organizational Unit =   
Organization = OpenXPKI  
dc = org

### Descriptions of the form fields

**Common name**  
The common name can be used to identify persons or objects by its name. Examples are John Doe or `www.openxпки.org`.

Fertig

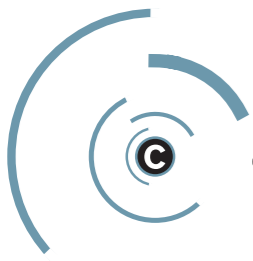


# Workflows

## Certificate Request: User Experience

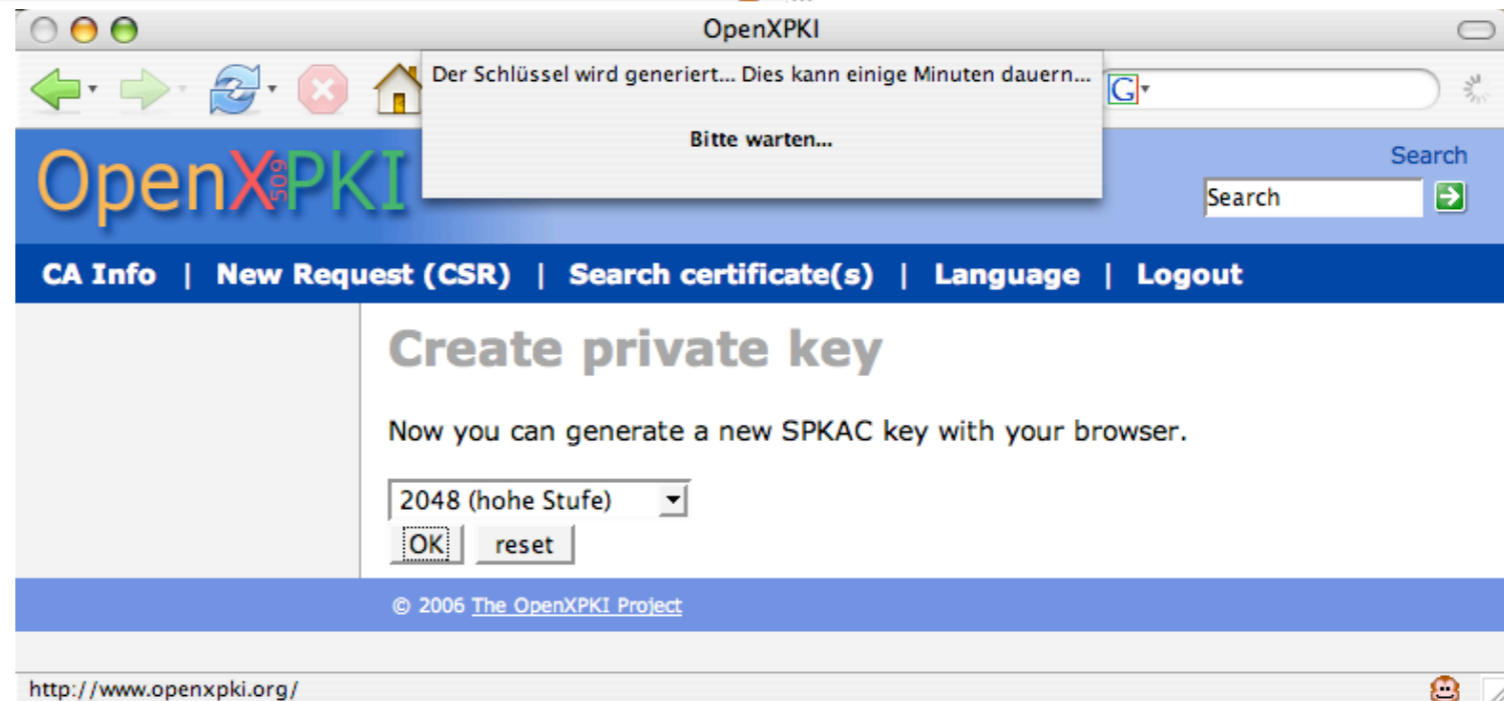
The screenshot shows a web browser window titled 'OpenXPKI' with the URL 'http://127.0.0.1:8080/service/create\_csr/index.html'. The page header includes the 'OpenXPKI' logo and a search bar. A navigation menu contains links for 'CA Info', 'New Request (CSR)', 'Search certificate(s)', 'Language', and 'Logout'. The main content area is titled 'Subject alternative name' and contains a text block explaining that certificates can have multiple alternative names, such as DNS names, IP addresses, or email addresses. Below this text is a form with several rows, each with a dropdown menu and a text input field. The first row has 'emailAddress' selected and 'a.klink@cynops.de' entered. The other rows are empty. At the bottom of the form are 'OK' and 'reset' buttons. The status bar at the bottom left shows 'Fertig'.

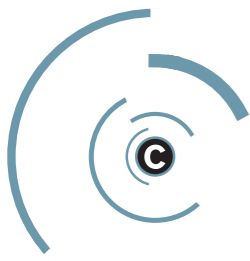
emailAddress	a.klink@cynops.de
emailAddress	
Microsoft UPN	
Microsoft GUID	
DNS	
IP	
IP	
URI	
Registered ID	
DirName	



# Workflows

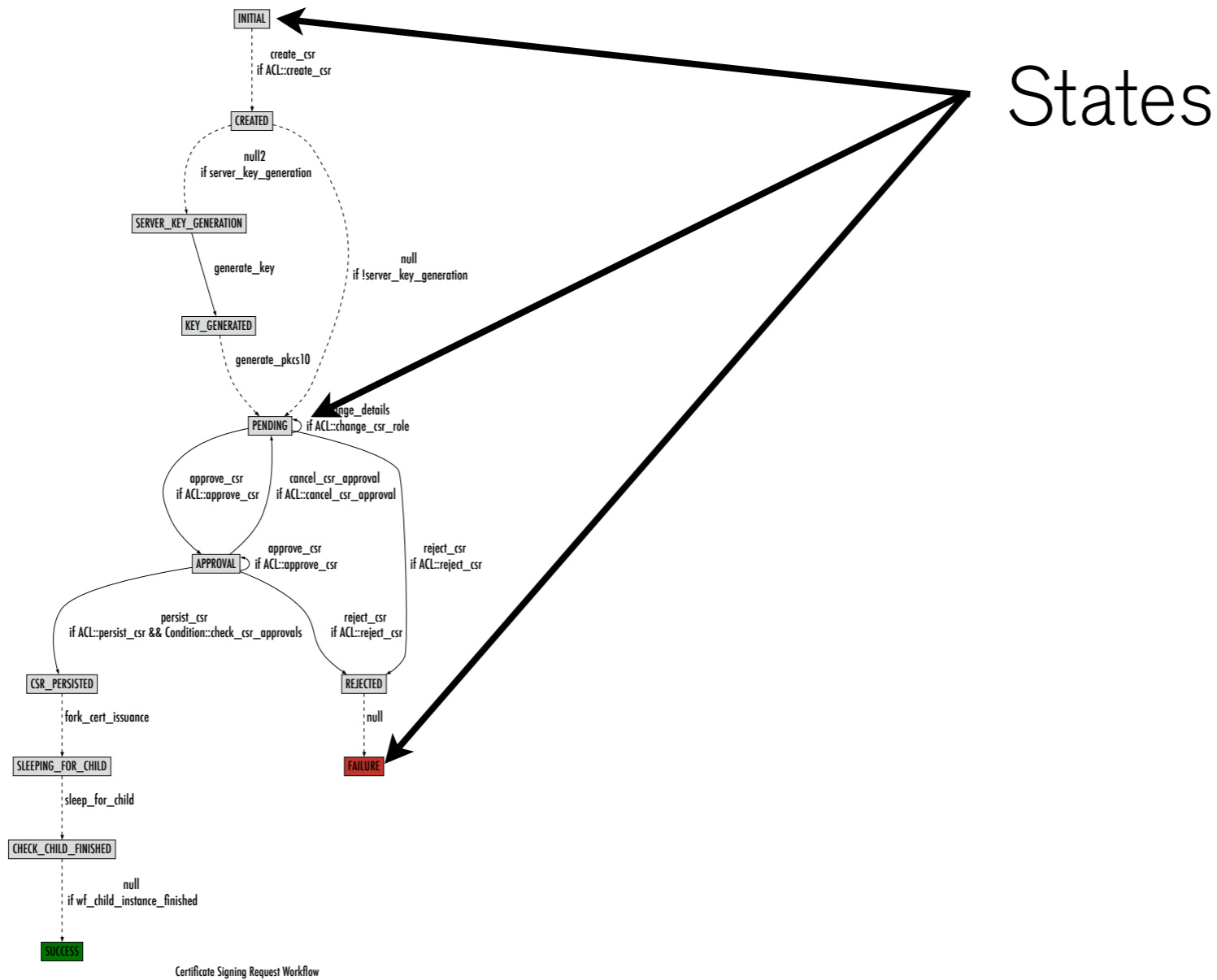
## Certificate Request: User Experience

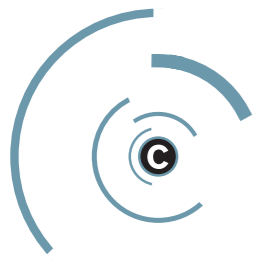




# Workflows

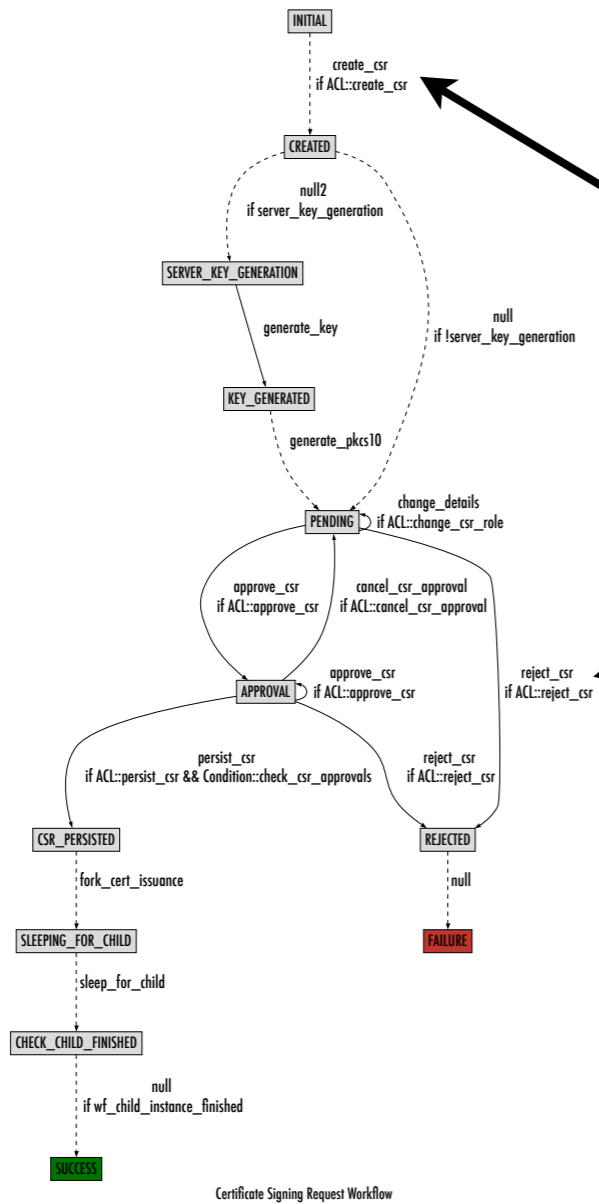
## Abstract: visualization using GraphViz



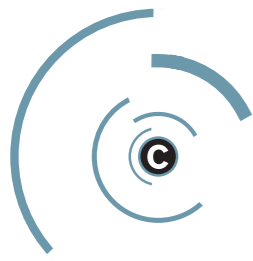


# Workflows

## Abstract: visualization using GraphViz

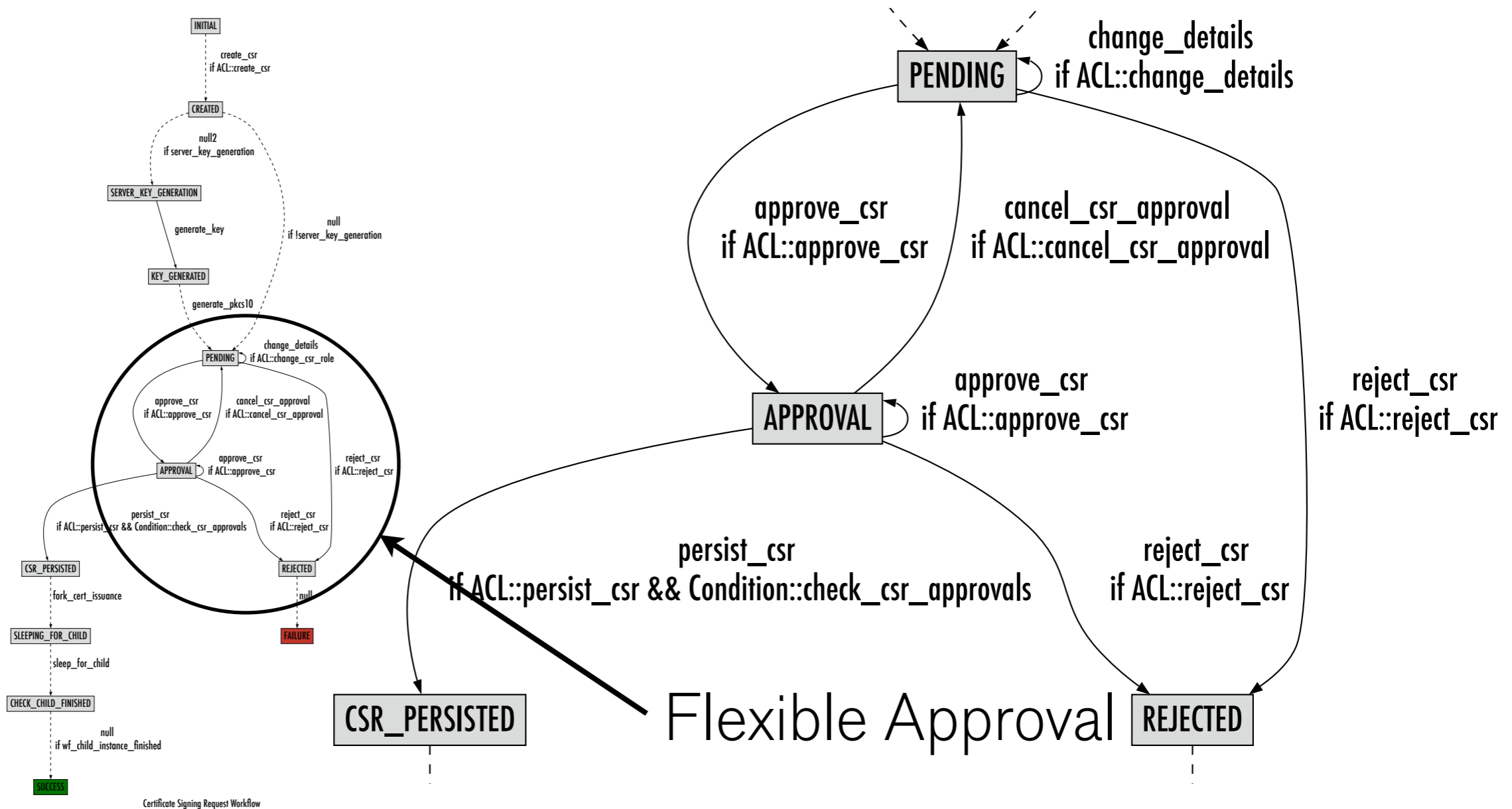


Activities (if condition)

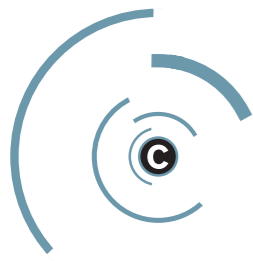


# Workflows

## Abstract: visualization using GraphViz







# Workflows

## more concrete: XML configuration

```
50 <state name="APPROVAL">
51   <action name="persist_csr"
52     resulting_state="CSR_PERSISTED">
53     <condition name="ACL::persist_csr"/>
54     <condition name="Condition::check_csr_approvals"/>
55   </action>
56   <action name="approve_csr"
57     resulting_state="APPROVAL">
58     <condition name="ACL::approve_csr"/>
59   </action>
60   <action name="cancel_csr_approval"
61     resulting_state="PENDING">
62     <condition name="ACL::cancel_csr_approval"/>
63   </action>
64   <action name="reject_csr"
65     resulting_state="REJECTED">
66     <condition name="ACL::reject_csr"/>
67   </action>
68 </state>
```

a snippet from workflow\_def\_certificate\_signing\_request.xml

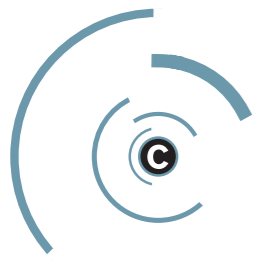


# Workflows

## at the bottom: Perl & Workflow.pm

```
18 sub execute {
19     my $self      = shift;
20     my $workflow  = shift;
21     my $context   = $workflow->context();
22
23     my $type      = $context->param('csr_type');
24     my $profile   = $context->param('cert_profile');
25
26     [ ... ]
27
28     $dbi->insert(
29         TABLE => 'CSR',
30         HASH   => {
31             'PKI_REALM'    => $pki_realm,
32             'CSR_SERIAL'   => $csr_serial,
33             'DATA'         => $data,
34             [ ... ]
35         },
36     );
37     $dbi->commit();
38     $context->param('csr_serial' => $csr_serial);
39 }
```

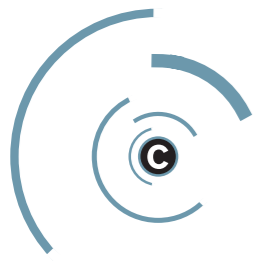
a snippet from Workflow/Activity/CSR/PersistRequest.pm



# SCEP

## What is it all about?

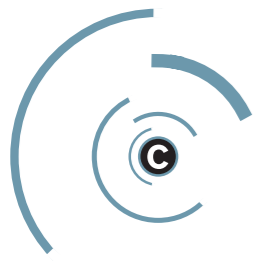
- SCEP = Simple Certificate Enrollment Protocol
- IETF draft initiated by Cisco
- Protocol to automatically enroll for or renew certificates
- PKCS#10 in PKCS#7 over HTTP (so it could be worse)
- Implemented in many hardware devices



# SCEP

## The motivation

- Alternative: doing it manually
  - request certificate, install, note down expiry date in (probably at least next year's) calendar
  - come expiry date, request new certificate, install
  - repeat, don't make mistakes
  - pretty unsuitable for larger installations



# SCEP

## The “user” perspective: sscep

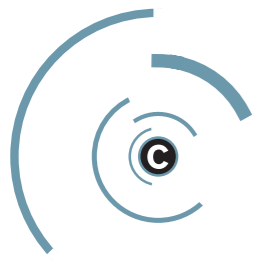
Request CA certificate from SCEP server

```
trinidad:~ klink$ sscep getca -u http://127.0.0.1:8042/cgi-bin/scep -c cacert
sscep: requesting CA certificate
sscep: valid response from server

sscep: found certificate with
  subject: /DC=org/DC=OpenXPKI/OU=Development/UID=scep/CN=SCEP Testserver
  issuer: /C=DE/O=Local Test/OU=PKI/CN=Local Test Root DUMMY CA 2
  usage: Key Encipherment
  MD5 fingerprint: 98:05:67:81:D9:DA:70:77:AD:E6:14:30:ED:67:E6:76
sscep: certificate written as cacert-0

sscep: found certificate with
  subject: /C=DE/O=Local Test/OU=PKI/CN=Local Test Root DUMMY CA 2
  issuer: /C=DE/O=Local Test/OU=PKI/CN=Local Test Root DUMMY CA 2
  usage: Certificate Sign, CRL Sign
  MD5 fingerprint: 59:ED:FE:F5:F3:94:21:B2:71:8F:D1:B6:6A:1C:C6:3B
sscep: certificate written as cacert-1
trinidad:~ klink$
```

verify fingerprints  
(out of band)



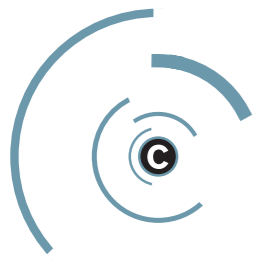
# SCEP

## The “user” perspective: `sscep`

Send Certificate Signing Request to CA

```
trinidad:~ klink$ sscep enroll -u http://127.0.0.1:8042/cgi-bin/scep -c cacert-0  
-k client_key.pem -r client_req.pem -l my_certificate.pem -t 30 -n 1  
sscep: sending certificate request  
sscep: valid response from server  
sscep: pkistatus: PENDING
```

Using a previously generated private key and PKCS#10



# SCEP

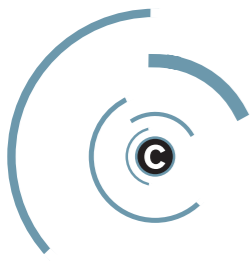
## The “user” perspective: sscep

### Send Certificate Signing Request to CA

```
trinidad:~ klink$ sscep enroll -u http://127.0.0.1:8042/cgi-bin/scep -c cacert-0
-k client_key.pem -r client_req.pem -l my_certificate.pem -t 10
sscep: sending certificate request
sscep: valid response from server
sscep: pkistatus: PENDING
sscep: requesting certificate (#1)
sscep: valid response from server
sscep: pkistatus: PENDING
sscep: requesting certificate (#2)
sscep: valid response from server
sscep: pkistatus: PENDING
sscep: requesting certificate (#3)
sscep: valid response from server
sscep: pkistatus: SUCCESS
sscep: certificate written as my_certificate.pem
trinidad:~ klink$
```

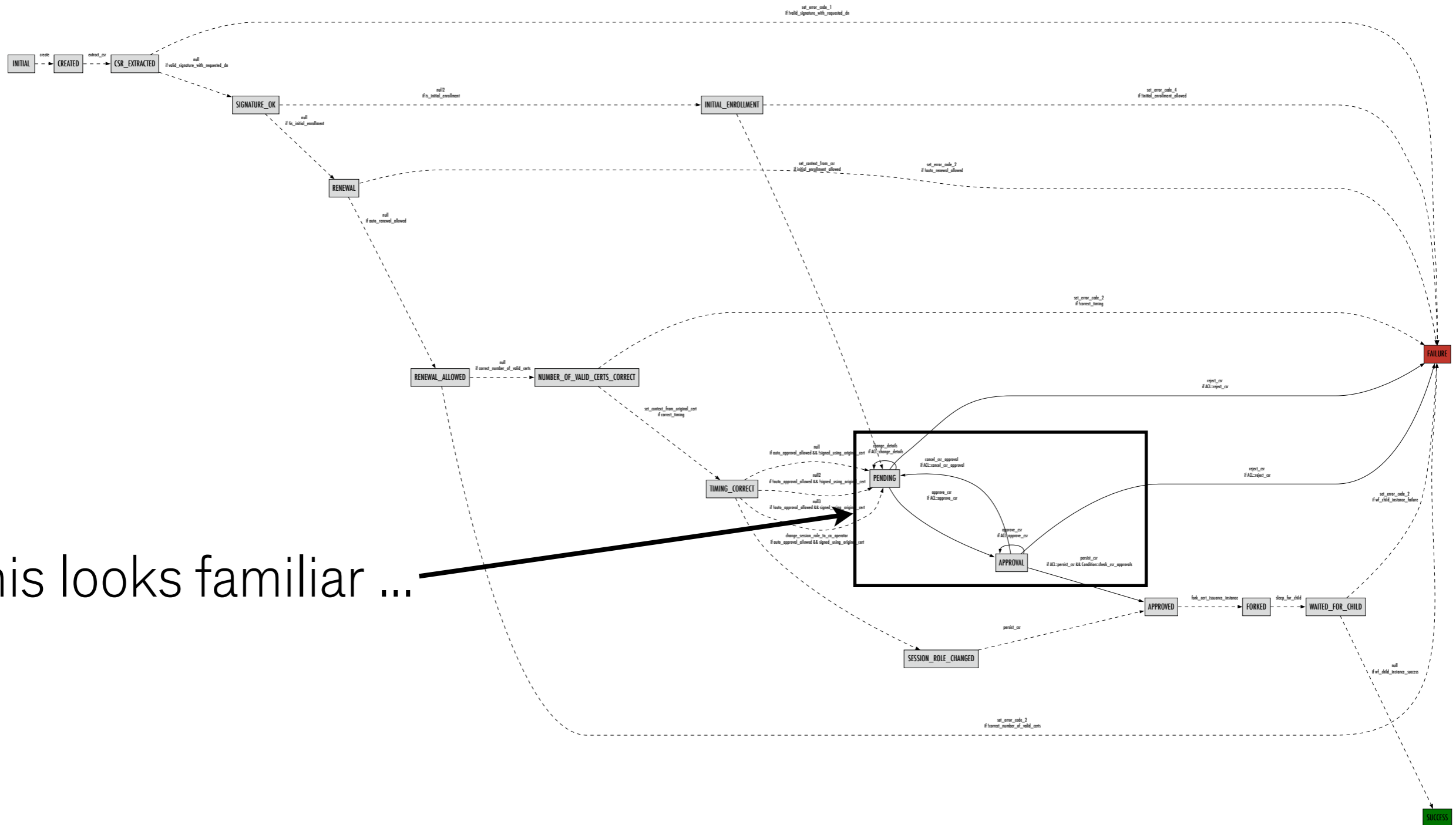
...wait for approval

... done!



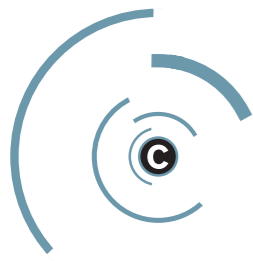
# SCEP

## Workflows revisited



this looks familiar ...

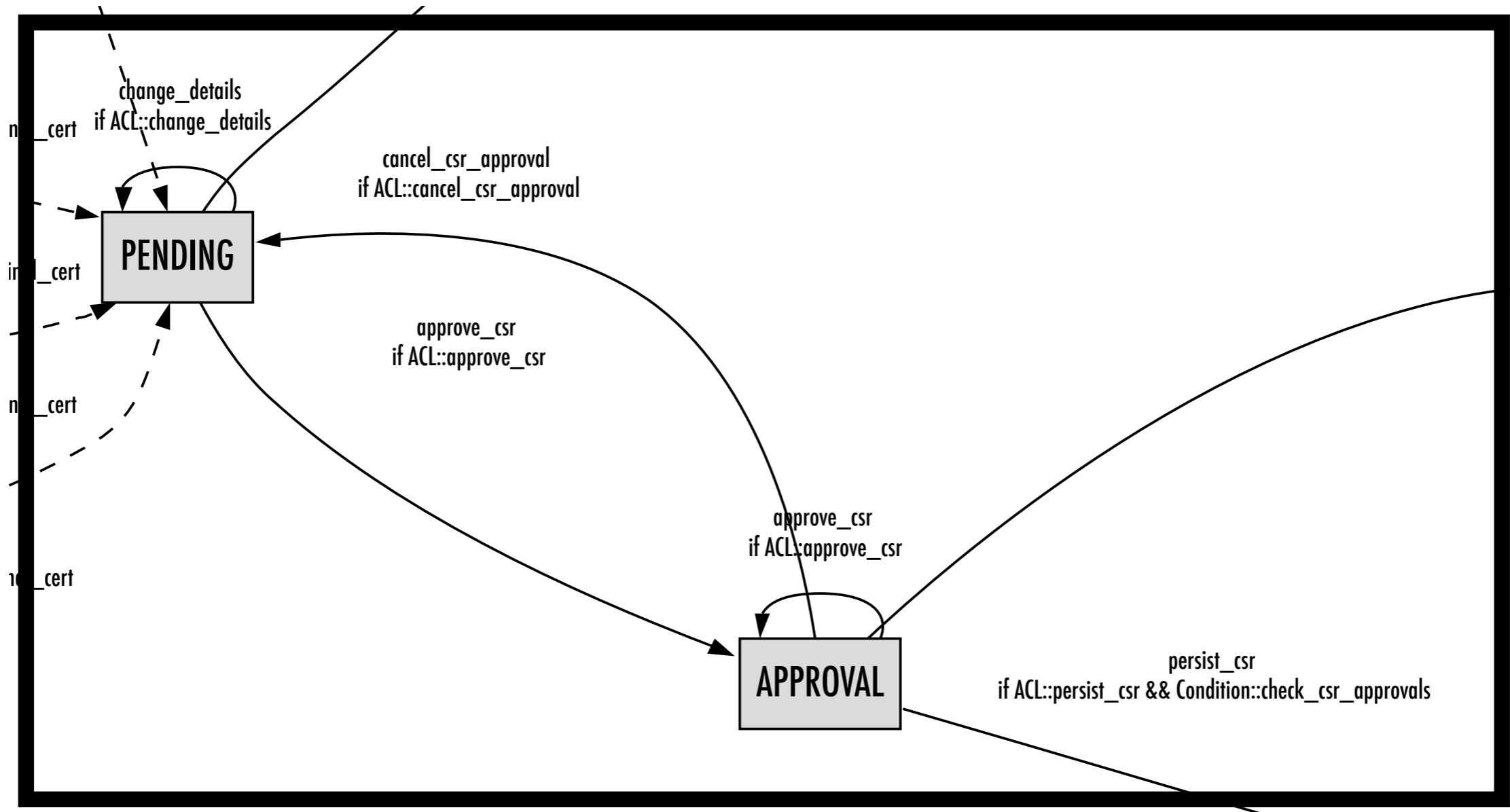




# SCEP

## Workflows revisited

Approval again:





# SCEP

## Alternatives

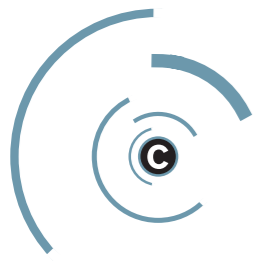
- PKIX-CMP (Certificate Management Protocol)
  - standardized in RFC 4210, 4211, but not widely used ...
- CMC (CM over CMS, no transport defined)
  - RFC 2797, used by Microsoft CA with COM/DCOM as proprietary transport
  - we would like to see support for that to be able to easily replace a MS CA for domain controller enrollment



# SCEP

## (Santa's?) Little helper: CertNanny

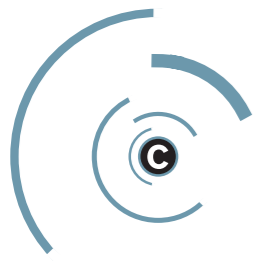
- Automatic renewal and keystore modification system (SCEP + signature using “old” certificate)
- Available on Unix (tested: Linux, AIX, Solaris, Darwin), Win32, Tandem NonStop
- Modifies OpenSSL keys, PKCS#8, Java Keystores, IBM GSKit, Windows Certificate Store
- widely deployed within a large financial institution



# Smartcard personalization

## What do we need smartcards for?

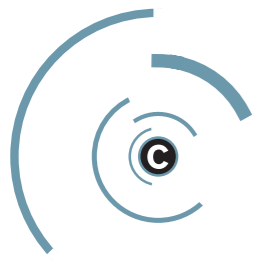
- Authentication:
  - Smartcard logon
  - (W)LAN authentication (802.1x)
  - Fileserver encryption (outsourcing!)
- (E-Mail)-Encryption
  - you have to think about key recovery



# Smartcard personalization

## Why a self-service personalization?

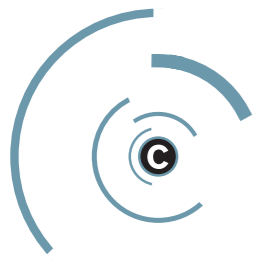
- To get a huge set of smartcards in a working state
  - Do It Yourself (DIY)
  - Let the users do it
- Obvious choice :-)
  - User only needs IE, Smartcard drivers and some time → screencast



# Support

## If RTFM does not help ...

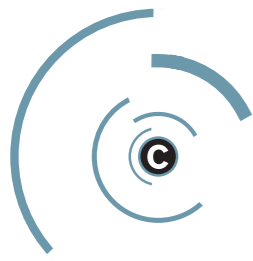
- Actually, the fine manual is not yet finished ...
- ... but even when it is, OpenXPKI is still quite a complex piece of software
- Support via mailing list: [openxpki-users@lists.sf.net](mailto:openxpki-users@lists.sf.net)
- Shameless plug: Cynops GmbH offers commercial support as well



# Hackers wanted

## Use it, break it, enhance it ...

- We are always looking for more people to help
  - Developers
  - Bug<sup>^</sup>H<sup>^</sup>H<sup>^</sup>H feature reporters :-)
  - people to WTFM (just kidding)
  - Code auditors
  - Users of hardware that talks SCEP
- Talk to us now, or later: [openxpki-devel@lists.sf.net](mailto:openxpki-devel@lists.sf.net)



# Questions?

## Comments? Confusion?

- Now is the time to ask ....
- But later is fine too:
  - Alex – DECT 2412, ak-23c3@cynops.de
  - Michael – michael.bell@cms.hu-berlin.de
- Thanks for your time!