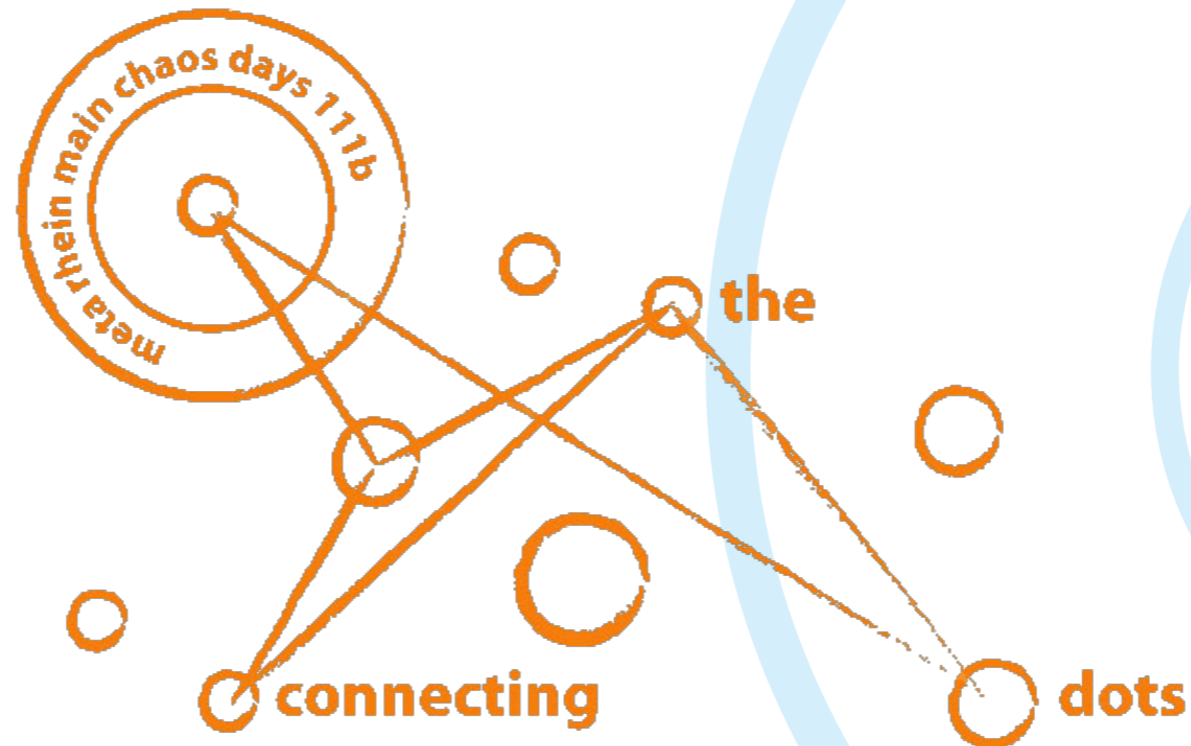
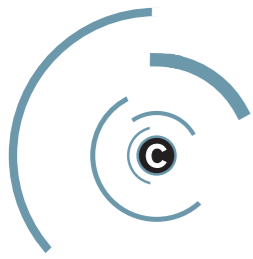


Open Source Public Key Infrastrukturen mit OpenXPKI

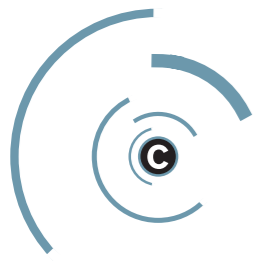
Alexander “alech” Klink, Cynops GmbH





Übersicht

- Kurze Einführung in PKI
- OpenXPKI Features und Konzepte
- Workflows am Beispiel Zertifikatsantrag
- Simple Certificate Enrollment Protocol (SCEP)
- Smartcard-Personalisierung
- Support
- Hackers wanted!



Public Key Infrastrukturen

Protecting your private bits

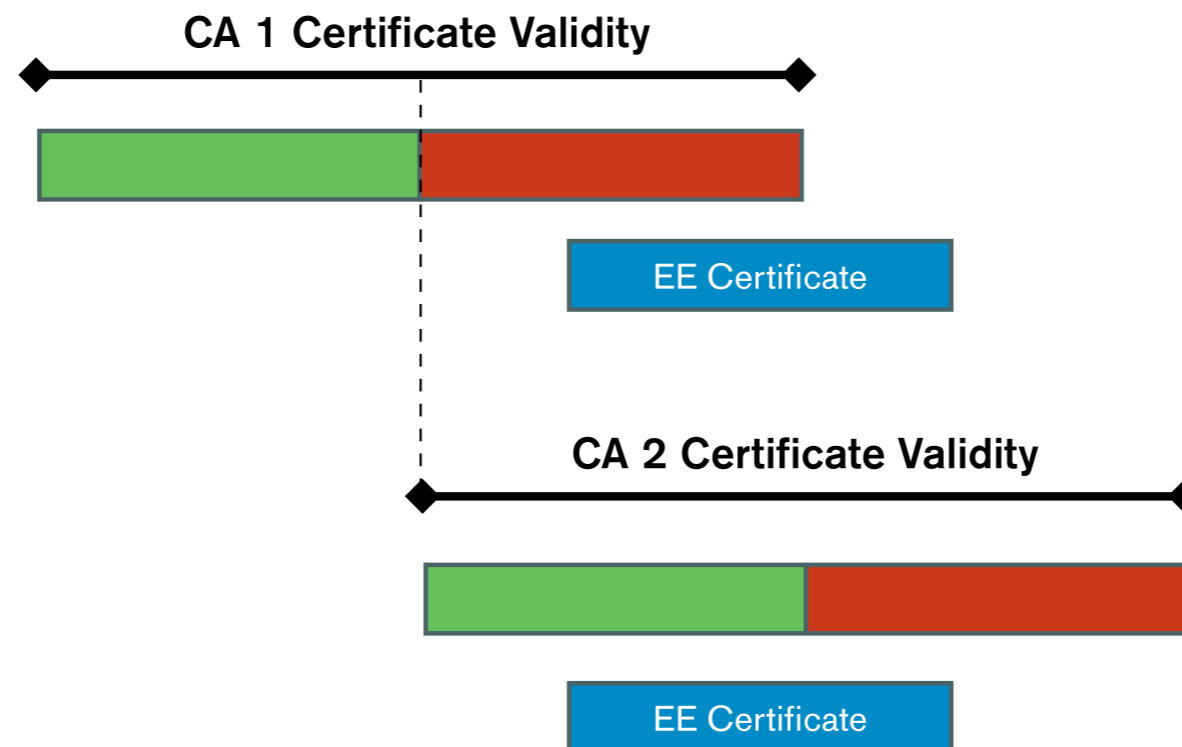
- Alice und Bob, oder auch nicht
- Certificate Authorities (CAs) stellen Zertifikate aus – an end entities (User/Server) oder CAs
- Zertifikatsantrag + Vertrauen + Freigabe = Zertifikat
- hierarchische Strukturierung von CAs

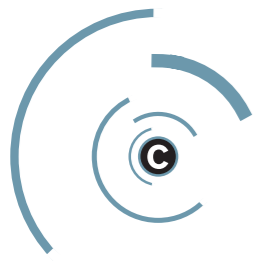


OpenXPKI features

All the best, for free

- Enterprise Features, nicht die typische 5-Minuten CA für das Heim-VPN
- CA rollover für kontinuierlichen Betrieb

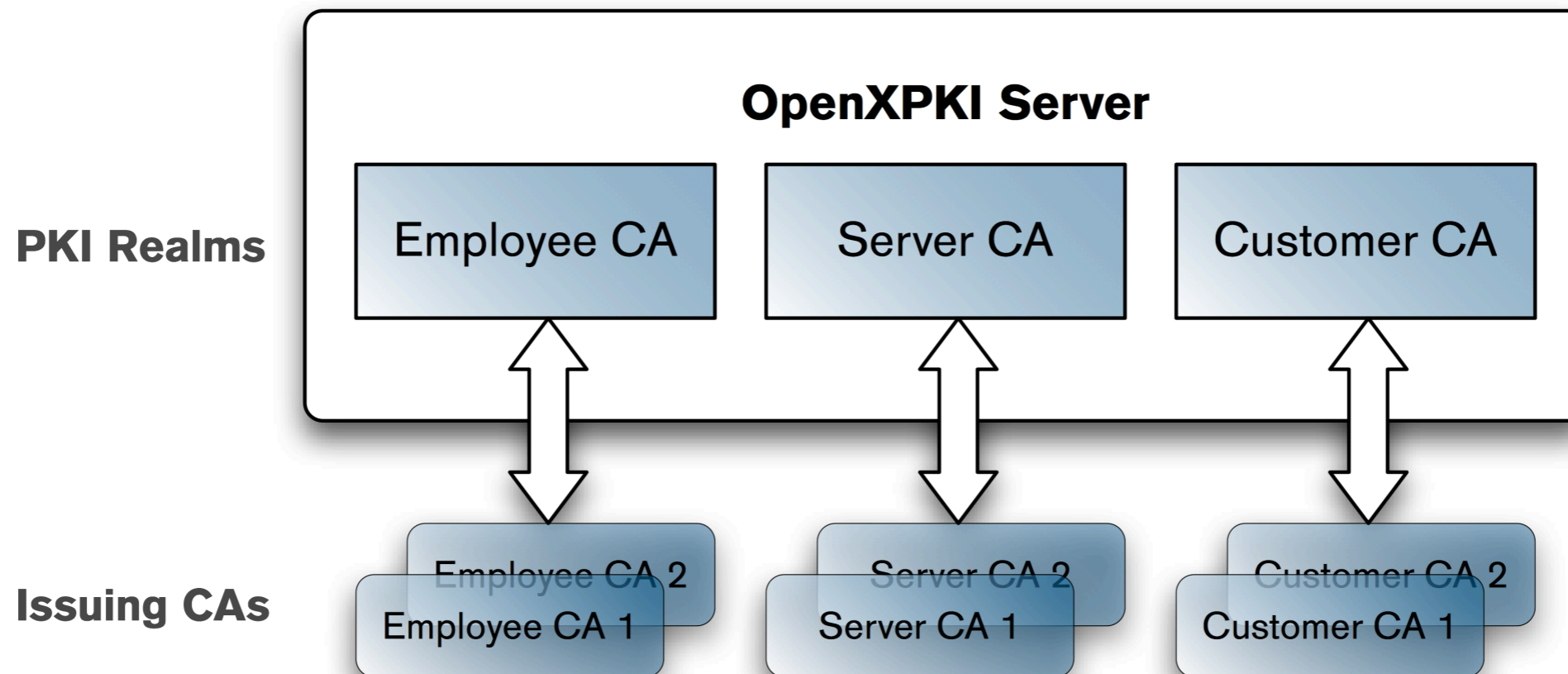


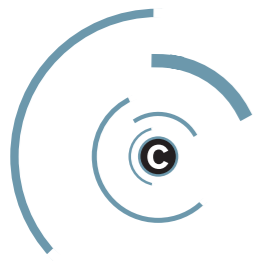


OpenXPKI features

All the best, for free

- PKI realms \approx Multimandantenfähigkeit

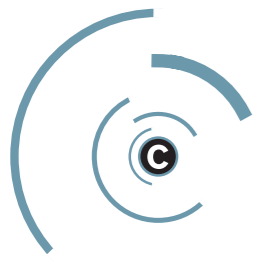




OpenXPKI Features

All the best, for free

- Support für professionelle Backends: HSM, RDBMS, LDAP, ...
- I18N: Englisch, Deutsch, Russisch
- Konfigurationsvererbung (`<tag super="../../foo">`)
- Request-Tracker-Anbindung
- Flexibilität durch Workflows
- Interfaces: Web, SCEP, Perl
- Smartcard-Personalisierung



Workflows

Beispiel Zertifikatsantrag

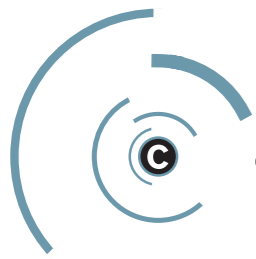
- Wie sieht es für den User aus?
- Zugriff über ein Webinterface
 - Erstellung durch Browser (SPKAC/XEnroll)
 - PKCS#10-Upload
 - Schlüsselgenerierung auf der CA
- Und so sieht's aus (ungefähr) ...



Workflows

Zertifikatsantrag: User Interface

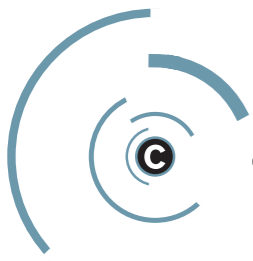
The screenshot shows a web browser window titled "OpenXPKI - Vimperator". The browser tab is labeled "OpenXPKI". The page header features the "OpenXPKI" logo on the left and navigation links "Feedback | About a | User | Test Dummy CA" on the right, along with a search box containing the text "Search". A dark blue navigation bar contains the links "Home | Request | Download | Search | Logout". The main content area is titled "Choose a role" and includes the instruction "Please choose the role for which you would like to request a certificate." Below this is a dropdown menu currently set to "Web Server", and two buttons labeled "OK" and "reset". A sidebar on the left lists menu items: "Certificate Signing Request (CSR)", "Smartcard personalization", "Certificate Revocation Request (CRR)", and "Bulk certificate request". The footer contains the copyright notice "© 2006-2008 The OpenXPKI Project". The browser's address bar at the bottom shows the URL "https://127.0.0.1:8443/service/create_csr/index.html?;__menu_level=1;__1" and includes status icons for [1/1], All, 127.0.0.1:8443, a lock icon, a smiley face icon, and a search icon.



Workflows

Zertifikatsantrag: User Interface

The screenshot shows a web browser window titled "OpenXPKI - Vimperator". The browser tab is labeled "OpenXPKI". The page header features the "OpenXPKI" logo and navigation links: "Feedback | About a | User | Test Dummy CA". A search bar is present with a green arrow button. Below the header is a dark blue navigation bar with links: "Home | Request | Download | Search | Logout". The main content area is titled "Choose a naming style" and contains the text: "Please choose a style for the certificate's name. Depending on the style, you'll be asked for more or less information which will be included in the certificate." Below this text is a dropdown menu with three options: "Basic TLS style" (selected), "Basic TLS style", and "Advanced naming style". The browser's address bar shows the URL: "https://127.0.0.1:8443/service/create_csr/index.html [+]. The status bar at the bottom indicates "127.0.0.1:8443" and includes security icons.



Workflows

Zertifikatsantrag: User Interface

OpenXPKI - Vimperator

OpenXPKI

Feedback | About
a | User | Test Dummy CA

Search

Home | Request | Download | Search | Logout

Certificate Signing Request (CSR)
Smartcard personalization
Certificate Revocation Request (CRR)
Bulk certificate request

Select the way of the key generation

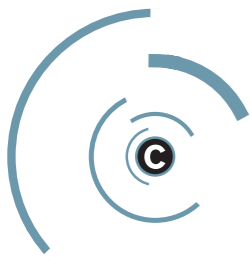
Please choose one of the following available options for the key generation. This is important because this influence the next steps. The options can mean really different things. So please read the descriptions carefully.

SPKAC
SPKAC
Microsoft Internet Explorer Serverside key generation
PKCS#10
Automatic browser detection

This type of key generation is used if you use a Microsoft Internet Explorer and you want to use your browser to generate the key on your client side.

https://127.0.0.1:8443/service/create_csr/index.html [+] [1/1] All 127.0.0.1:8443

-- INSERT --



Workflows

Zertifikatsantrag: User Interface

The screenshot shows a web browser window titled "OpenXPKI - Vimperator". The page header includes the "OpenXPKI" logo, navigation links for "Feedback", "About", "a", "User", and "Test Dummy CA", and a search box. A blue navigation bar contains links for "Home", "Request", "Download", "Search", and "Logout".

The main content area is titled "Creating the name of the certificate". It contains the following text: "The name of the certificate is build 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."

The form fields are:

- Hostname:
- Port (optional):

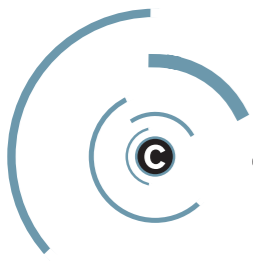
Buttons for "OK" and "reset" are located below the form fields.

Below the form is a section titled "Descriptions of the form fields":

- Hostname: This is the (fully qualified) hostname
- Port: The (TCP) port that is used with the certificate

The footer of the page contains the copyright notice: "© 2006-2008 The OpenXPKI Project".

The browser's address bar shows the URL: `https://127.0.0.1:8443/service/create_csr/index.html`. The status bar at the bottom right displays "[1/1] All 127.0.0.1:8443" along with security and navigation icons.



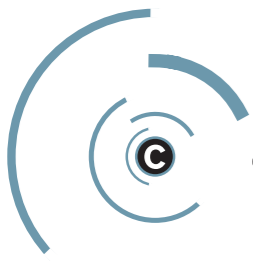
Workflows

Zertifikatsantrag: User Interface

The screenshot shows a web browser window titled "OpenXPKI - Vimperator". The browser tab is labeled "OpenXPKI". The page header features the "OpenXPKI" logo and navigation links: "Feedback | About a | User | Test Dummy CA". A search bar is present with the text "Search" and a green arrow button. Below the header is a blue navigation bar with links: "Home | Request | Download | Search | Logout".

The main content area is titled "Additional information" and contains the instruction: "Please enter now some additional information." Below this, there are two input fields: "Phone" (a text input box) and "Comment" (a larger text area). At the bottom of the form are two buttons: "OK" and "reset".

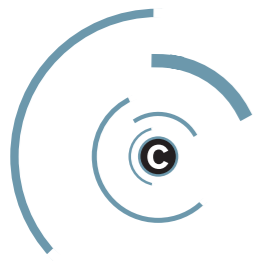
The browser's address bar at the bottom shows the URL: "Link: https://127.0.0.1:8443/service/info/download.html?;__menu_leve [1/1] All 127.0.0.1:8443".



Workflows

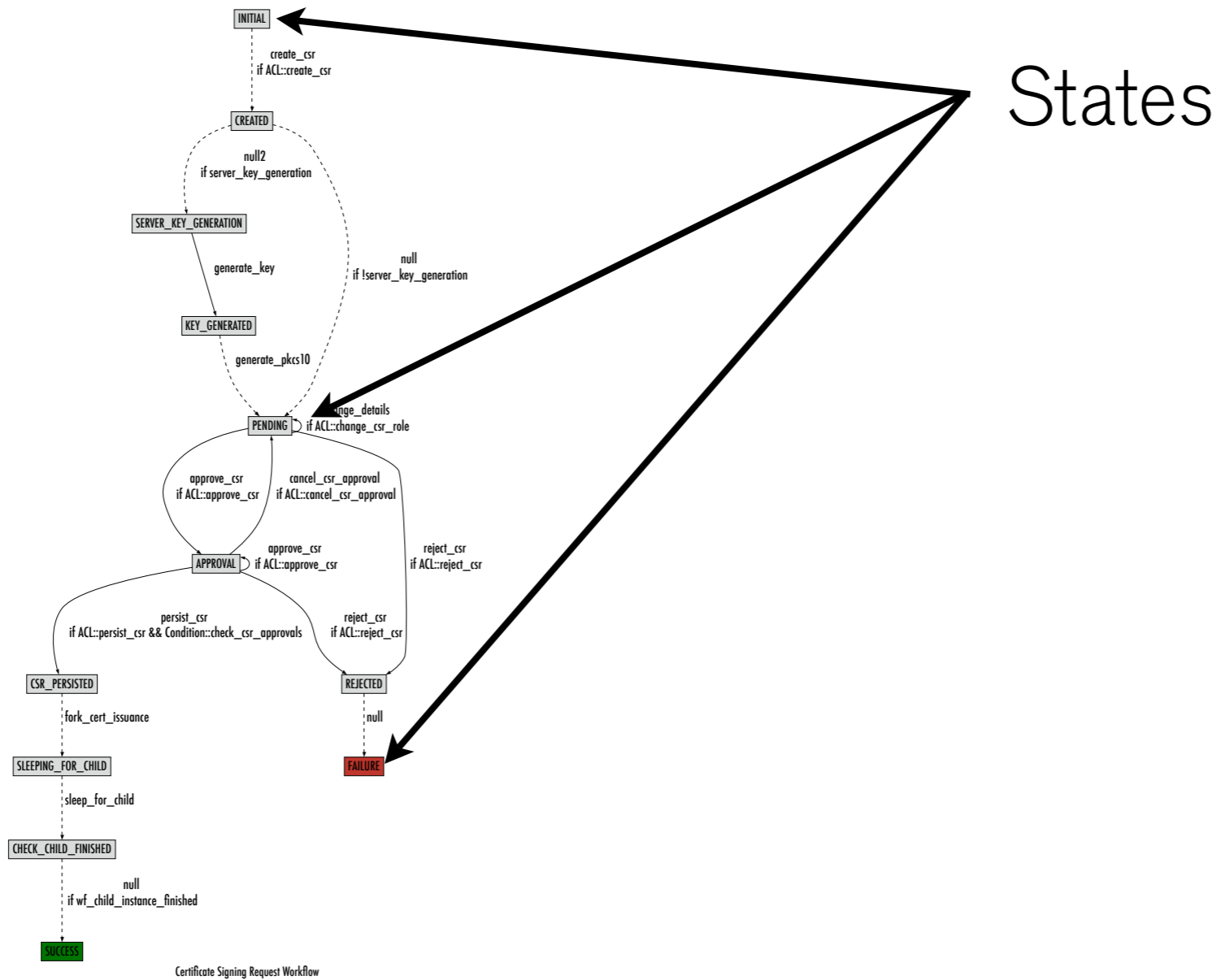
Zertifikatsantrag: User Interface

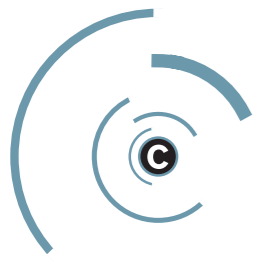




Workflows

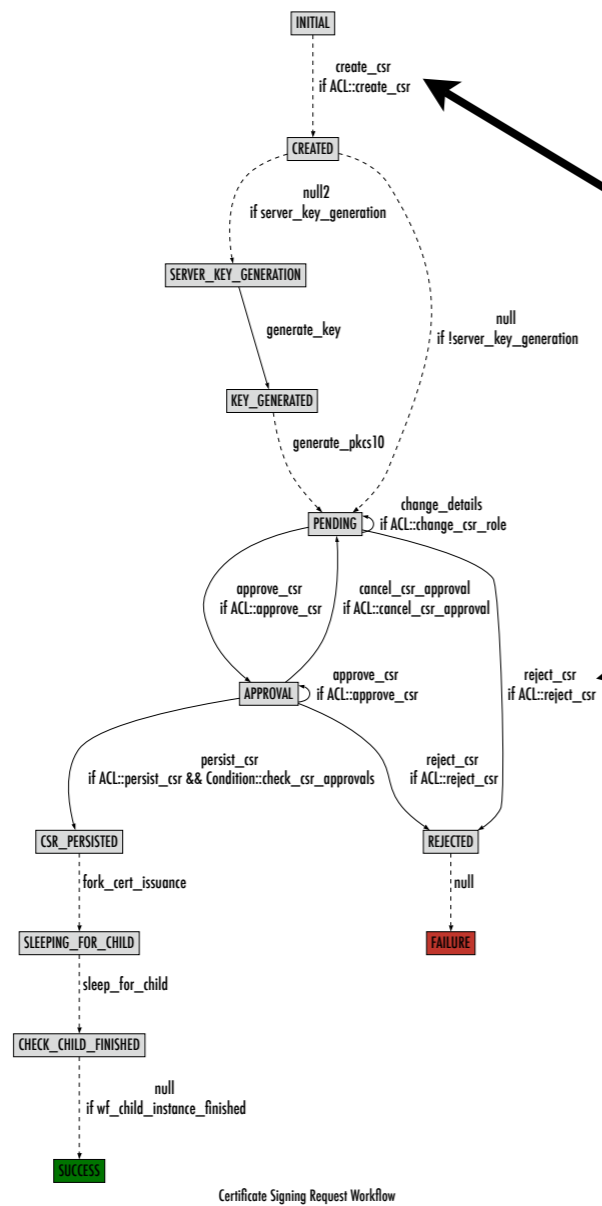
Visualisierung mit GraphViz



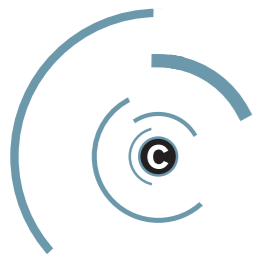


Workflows

Visualisierung mit GraphViz

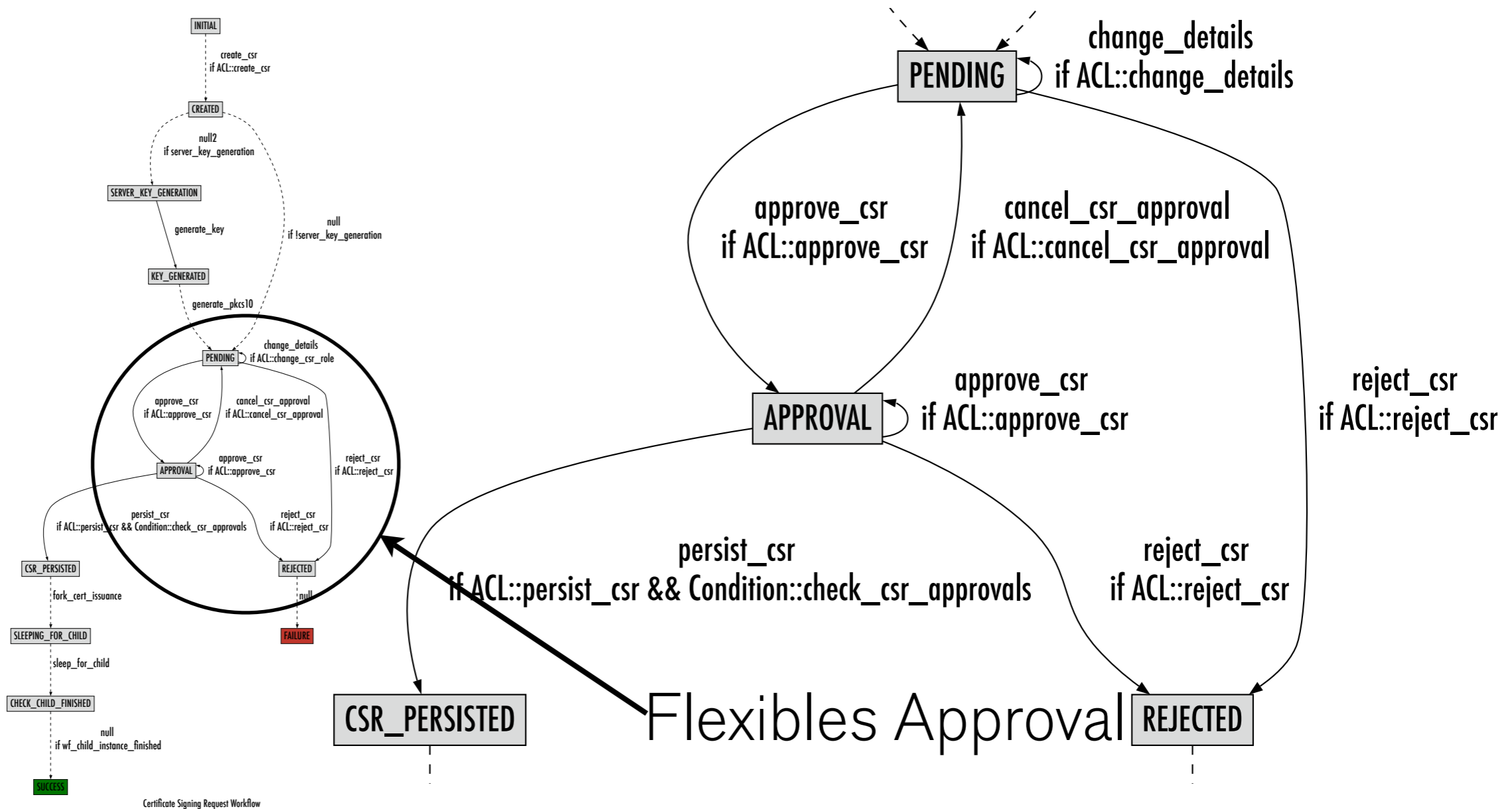


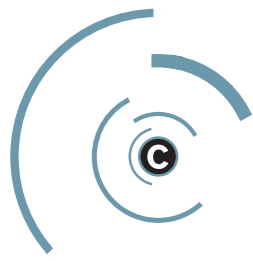
Activities (if condition)



Workflows

Visualisierung mit GraphViz



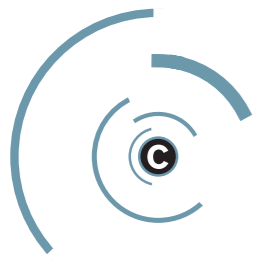


Workflows

XML-Konfiguration

```
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

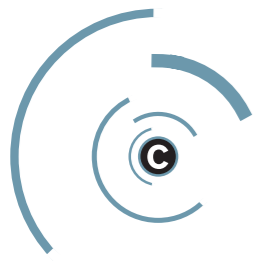


Workflows

ganz unten: 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

Worum geht's?

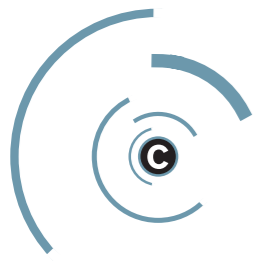
- SCEP = Simple Certificate Enrollment Protocol
- IETF-Draft ausgehend von Cisco
- Protokoll um automatisch Zertifikate zu beantragen oder zu erneuern
- PKCS#10 in PKCS#7 über HTTP (könnte also schlimmer sein ...)
- In vielen Hardware-Devices eingebaut



SCEP

Die Motivation

- Alternative: manueller Prozess:
 - Zertifikat anfordern, installieren, Ablaufdatum in den Kalender eintragen (wahrscheinlich mindestens den vom nächsten Jahr)
 - Vor dem Ablaufdatum: gleiche Prozedur
 - wiederholen ...
 - ziemlich unmöglich für große Installationen



SCEP

... unter Unix mit sscep

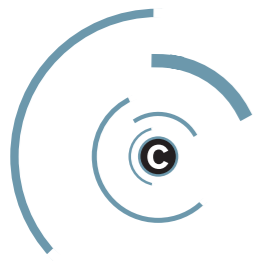
CA-Zertifikate vom SCEP-Server anfordern

```
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$
```

Fingerprints
überprüfen
(out of band)



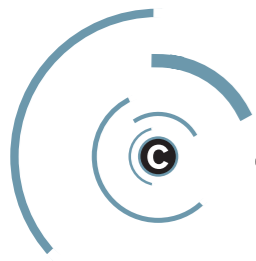
SCEP

... unter Unix mit `sscep`

Zertifikatsantrag an die CA schicken

```
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
```

Mit einem vorher generierten Schlüssel und PKCS#10-File



SCEP

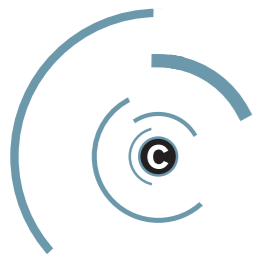
... unter Unix mit sscep

Zertifikatsantrag an die CA schicken

```
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$
```

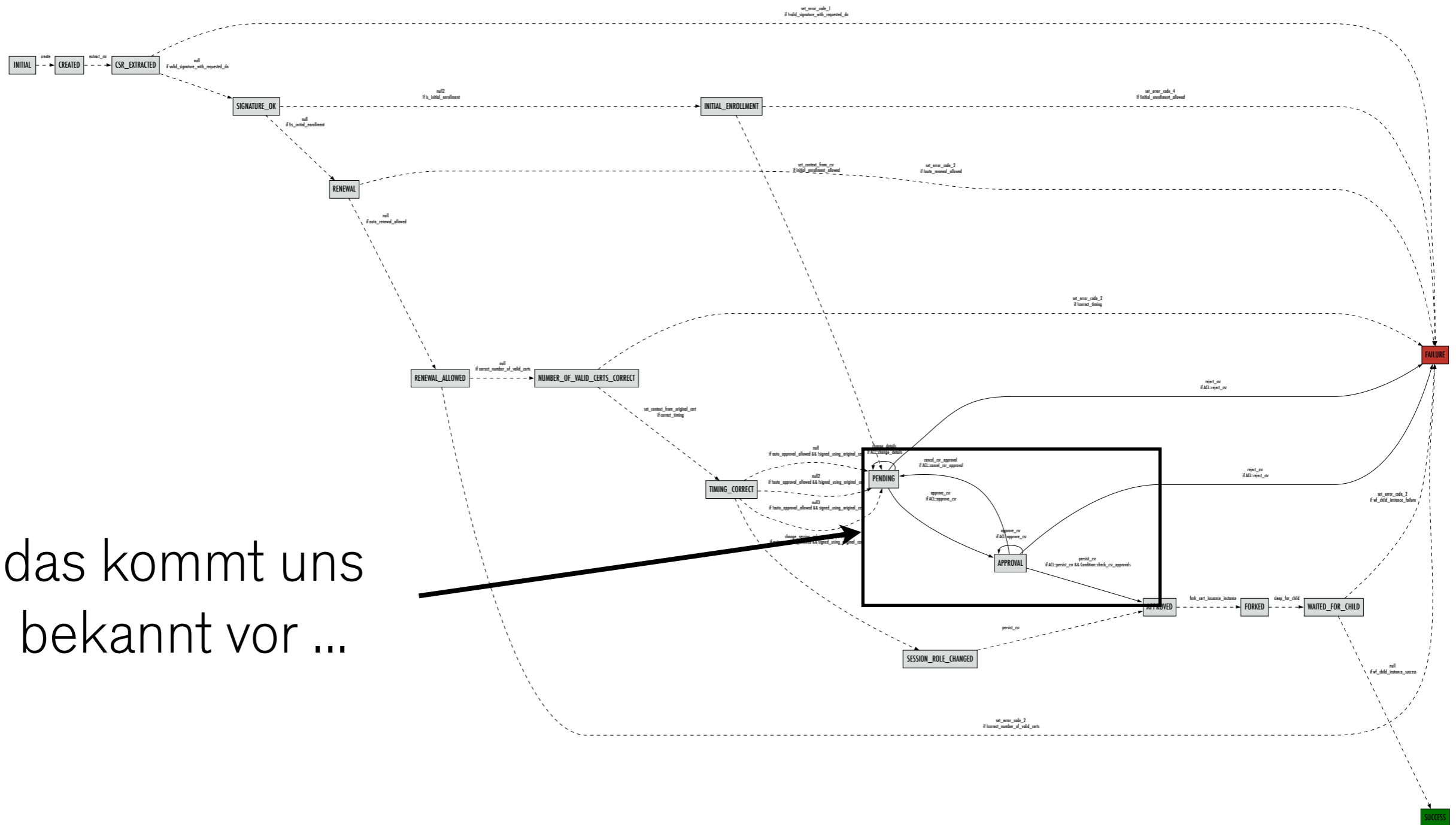
... auf Freigabe
warten

... fertig!

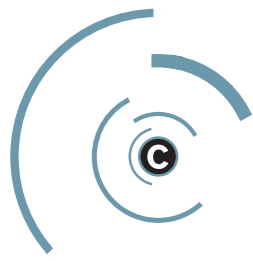


SCEP

... schon wieder Workflows



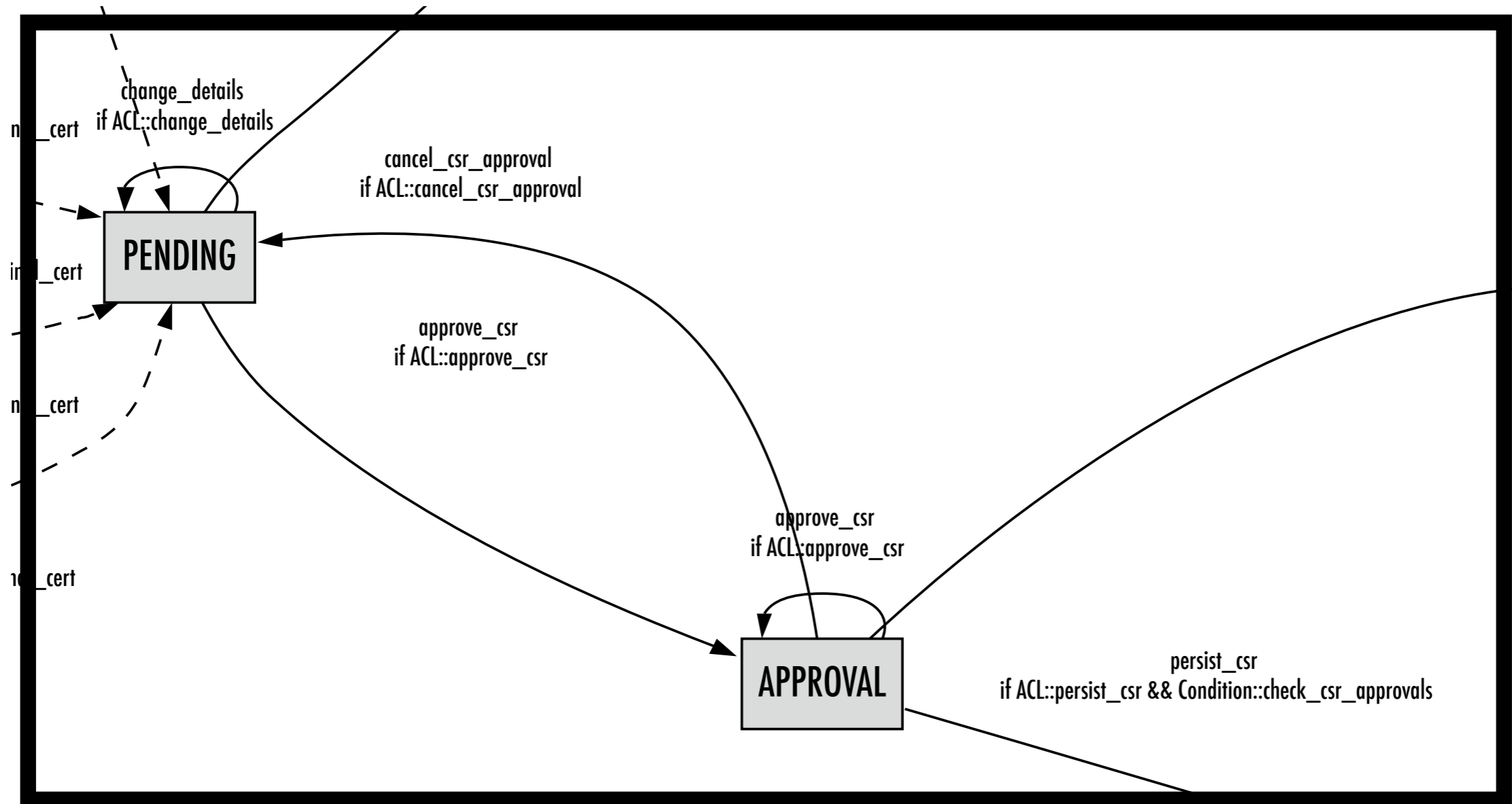
das kommt uns bekannt vor ...

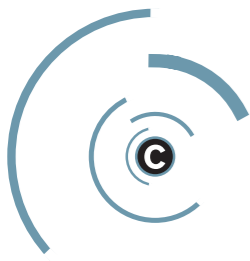


SCEP

... schon wieder Workflows

Approval:

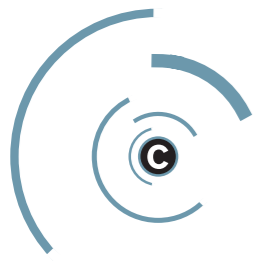




SCEP

Alternativen

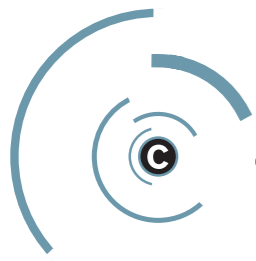
- PKIX-CMP (Certificate Management Protocol)
 - standardisiert in RFC 4210, 4211, aber sehr wenig benutzt ...
- CMC (CM over CMS, kein Transport definiert)
 - RFC 2797, wird von Microsoft CA mit COM/DCOM als proprietärem Transport genutzt
 - Support in OpenXPKI wäre super, da wir so einfach eine MS CA für Domain-Controller-Enrollment ersetzen könnten



SCEP

Hilfswerkzeug: CertNanny

- Automatisches Renewal von Zertifikaten und Austausch im Keystore (SCEP + Signatur mit dem “alten” Zertifikat)
- Verfügbar auf Unix (getestet: Linux, AIX, Solaris, Darwin), Win32, Tandem NonStop
- Arbeitet mit OpenSSL-Schlüsseln, PKCS#8, Java Keystores, IBM GSKit, Windows Certificate Store, PKCS#12
- weit verbreitet in einer großen Bank

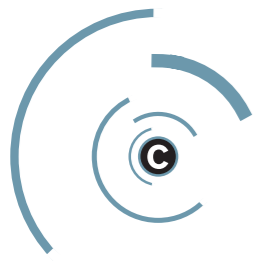


Smartcard-Personalisierung

Wofür eigentlich Smartcards?

- Authentisierung:
 - Smartcard logon
 - (W)LAN (802.1x)
 - Fileserver-Verschlüsselung (outsourcing!)
- (E-Mail)-Encryption
 - ...aber: Problem Key Recovery

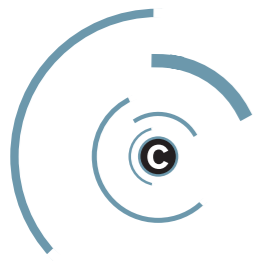




Smartcard-Personalisierung

Warum Self-Service-Personalisierung?

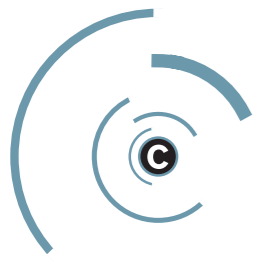
- Eine große Anzahl Karten in einen funktionsfähigen Zustand bringen:
 - selbst machen ...
 - ... oder es die Benutzer machen lassen
- Da fällt die Entscheidung leicht :-)
 - User braucht nur IE, Smartcard-Treiber und ein bisschen Zeit → screencast



Die Zukunft von OpenXPKI

Releases und neue Ideen

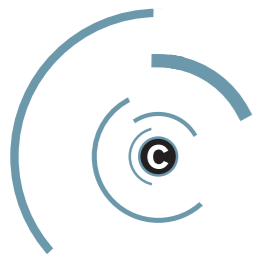
- Erstes Release “real soon now” (dieses Jahr)
- Dokumentation (Wiki statt DocBook)
- Testen, testen, testen
- mittelfristig:
 - Anbindung an Tivoli, Nagios, ...
 - Cluster-Support
- Neue Live-CD-Version



Support

Wenn RTFM nicht weiterhilft ...

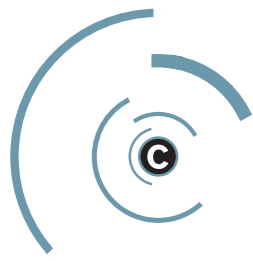
- Leider ist die Dokumentation noch nicht fertig ...
- ... aber selbst wenn sie es irgendwann ist, ist OpenXPki immer noch ziemlich komplex
- Support über die Mailingliste: openxpki-users@lists.sf.net, Sourceforge Tracker
- #openxpki im Freenode-Netz
- Cynops GmbH bietet auch kommerziellen Support an ... :-)



Hackers wanted

Use it, break it, enhance it ...

- Wir sind immer auf der Suche nach neuen Leuten
 - Entwicklern
 - Bug^H^H^H Feature-Reportern :-)
 - Leute, die gerne WTFM betreiben
 - Leute, die gerne Exploits finden
 - Leute, die SCEP-Hardware besitzen
- Gleich, oder später: openxpki-devel@lists.sf.net



Fragen?

Kommentare? Konfusion?

- Jetzt ist Zeit dafür ...
- Aber gerne auch noch später:
 - hier oder via a.klink@cynops.de
- Danke für Eure Zeit!