

# Open Source eID Projects

RMLL

Frank Cornelis  
10/07/2013

# Agenda

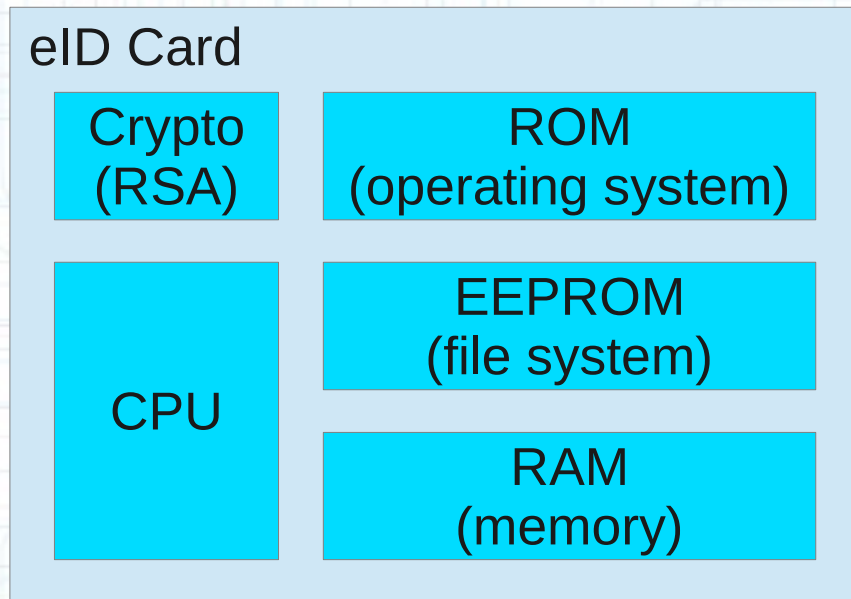
- Overview eID
- Cryptography in Java via JCA
- RSA, PKI, jTrust, eID Trust Service
- Integration levels for eID
- eID Applet
- Commons eID
- eID Identity Provider
- eID Digital Signature Service

# eID Functionality

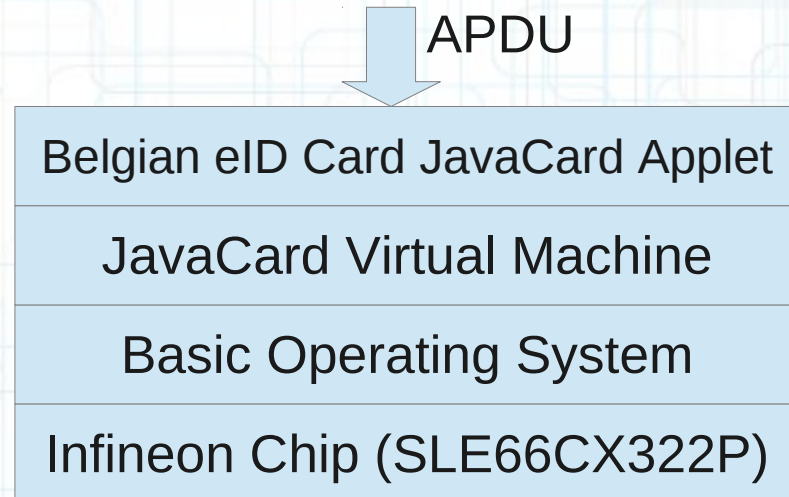
- Identification
  - Who are you?
- Authentication
  - Can you prove who you are?
- Digital signatures
  - Proof of statement made in time

# The Belgian eID Card

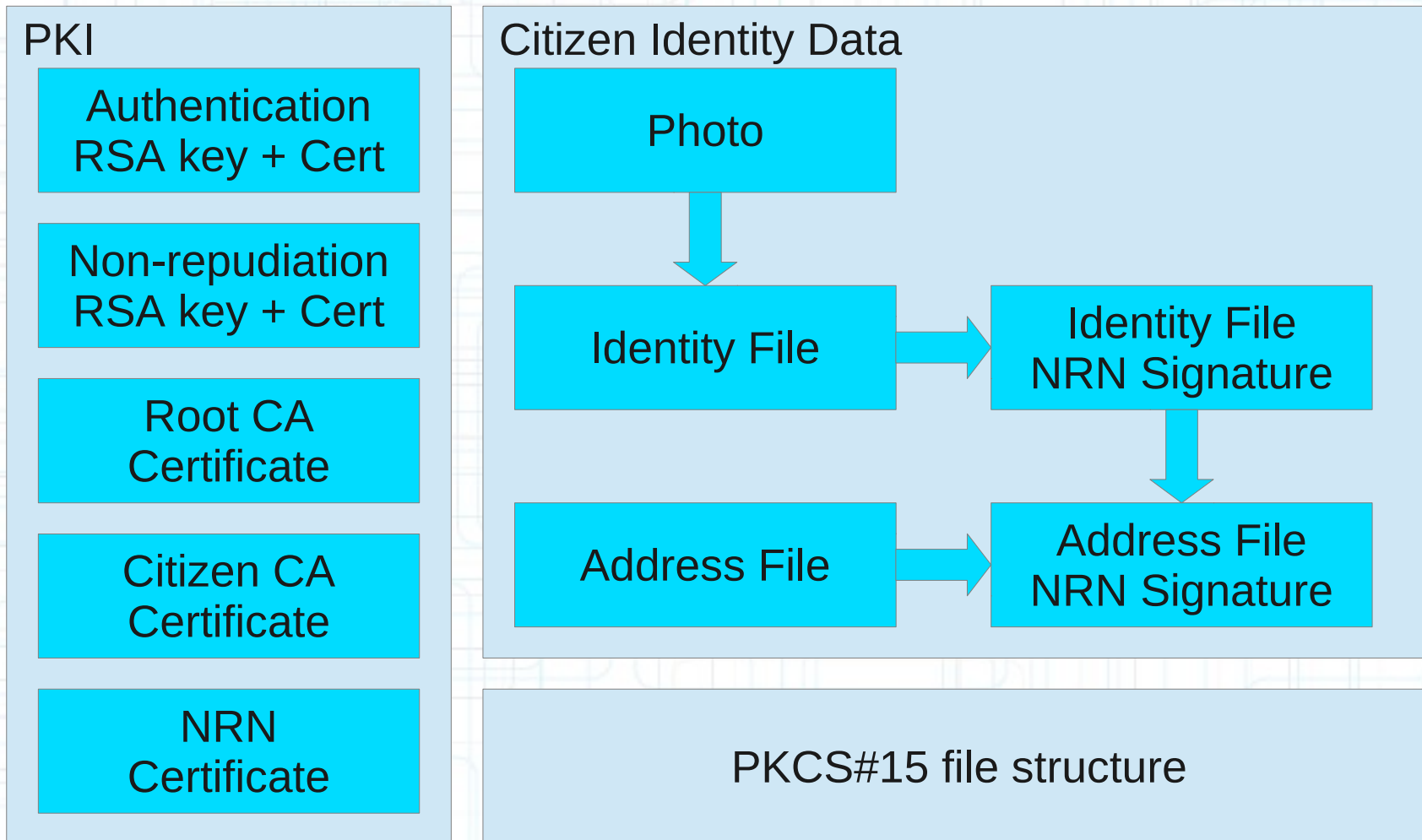
## Physical Structure



## Logical Structure



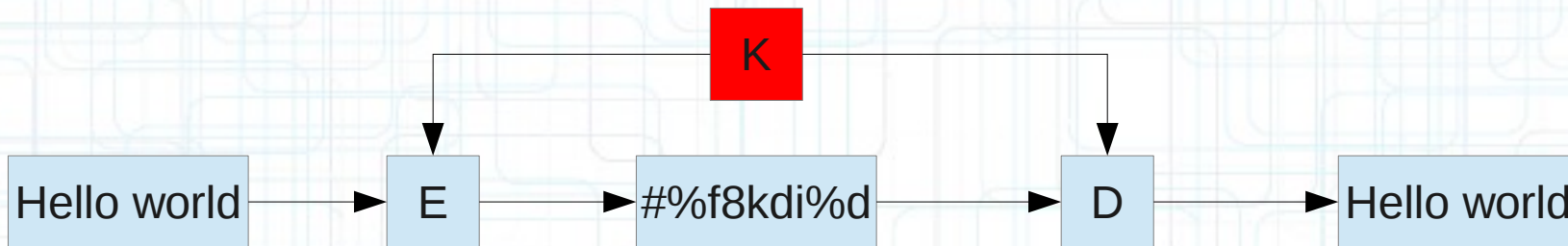
# eID Card Content



# Cryptography

- Encryption/decryption
  - Symmetric: AES
  - Asymmetric: RSA
- Digital signatures
  - RSA
- Hash functions
  - SHA256
- MAC
- Threshold crypto
- ...

# Symmetric encryption



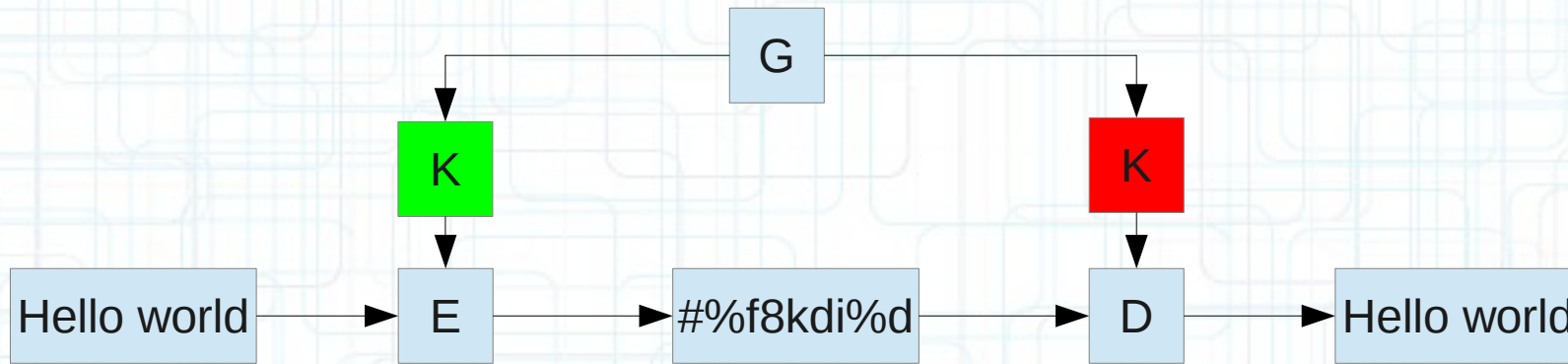
```
KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");  
keyGenerator.init(128);  
SecretKey secretKey = keyGenerator.generateKey();
```

```
byte[] message = "hello world".getBytes();
```

```
Cipher cipher = Cipher.getInstance("AES");  
cipher.init(Cipher.ENCRYPT_MODE, secretKey);  
byte[] encryptedMessage = cipher.doFinal(message);
```

```
cipher.init(Cipher.DECRYPT_MODE, secretKey);  
byte[] result = cipher.doFinal(encryptedMessage);
```

# Asymmetric encryption



```
KeyPairGenerator keyPairGenerator =  
    KeyPairGenerator.getInstance("RSA");  
keyPairGenerator.initialize(1024);  
KeyPair keyPair = keyPairGenerator.genKeyPair();
```

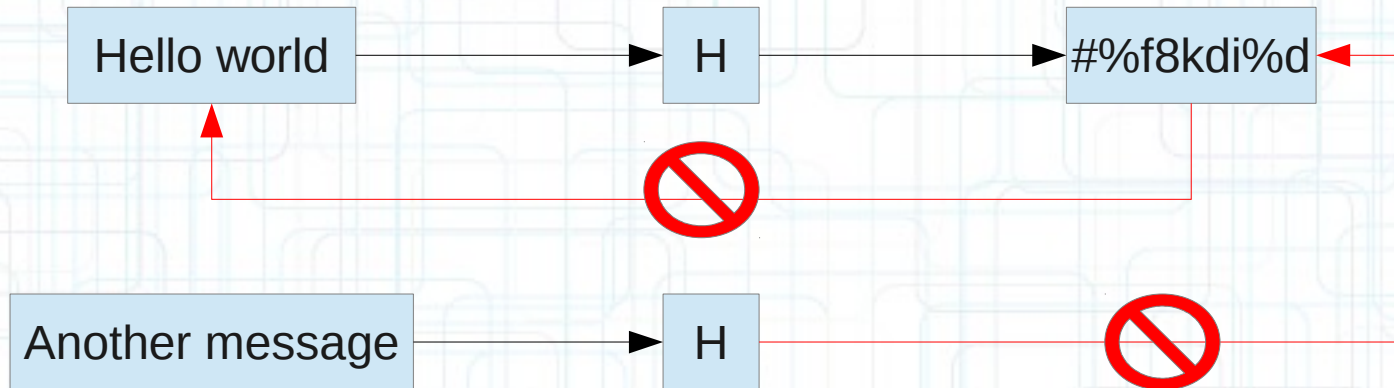
```
byte[] message = "hello world".getBytes();
```

```
Cipher cipher = Cipher.getInstance("RSA");  
cipher.init(Cipher.ENCRYPT_MODE, keyPair.getPublic());  
byte[] encryptedMessage = cipher.doFinal(message);
```

```
cipher.init(Cipher.DECRYPT_MODE, keyPair.getPrivate());  
byte[] result = cipher.doFinal(encryptedMessage);
```



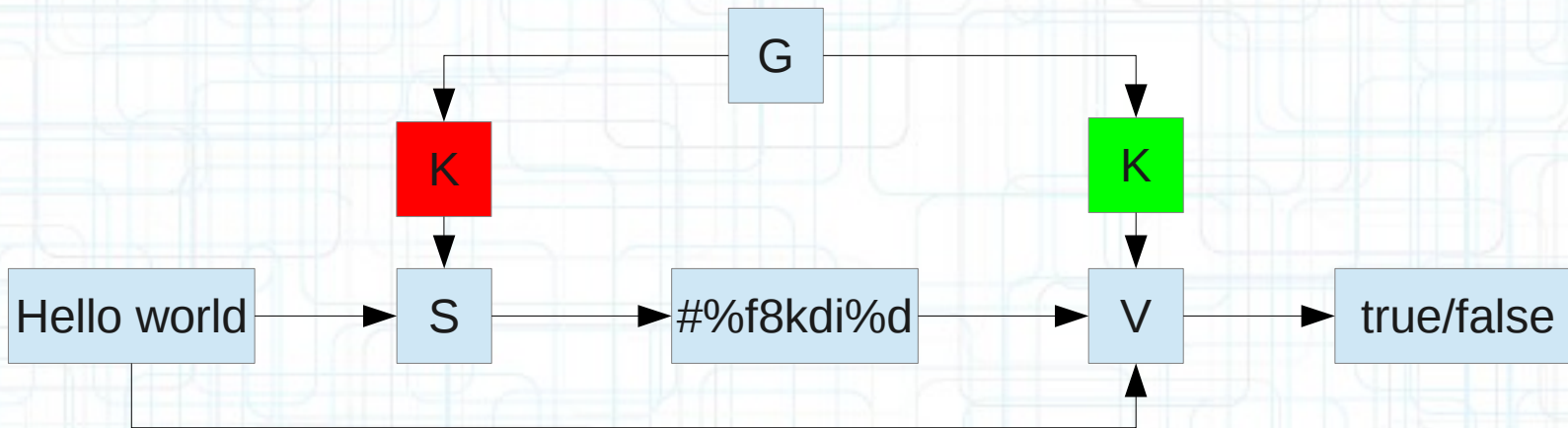
# Hash Functions



```
byte[] message = "hello world".getBytes();
```

```
MessageDigest messageDigest =  
    MessageDigest.getInstance("SHA256");  
messageDigest.update(message);  
byte[] result = messageDigest.digest();
```

# Digital Signatures



```
KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");  
keyPairGenerator.initialize(1024);  
KeyPair keyPair = keyPairGenerator.genKeyPair();
```

```
byte[] message = "hello world".getBytes();
```

```
Signature signature = Signature.getInstance("SHA1withRSA");  
signature.initSign(keyPair.getPrivate());  
signature.update(message);  
byte[] signatureValue = signature.sign();
```

```
signature.initVerify(keyPair.getPublic());  
signature.update(message);  
boolean result = signature.verify(signatureValue);
```

# RSA

group  $\langle G, \circ \rangle : \forall a \in G : a^{|G|} = e_G \Rightarrow a^{t|G|+1} = a$

$n = pq$  (Miller-Rabin)

$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n : a \perp n\}$  is a group

$|\mathbb{Z}_n^*| = \varphi(n) = (p-1)(q-1)$

$e \perp \varphi(n) \Rightarrow \exists d : ed \equiv 1 \pmod{\varphi(n)}$

public key :  $K^+ = \langle e, n \rangle$

private key :  $K^- = \langle d, n \rangle$

$\forall a \in \mathbb{Z}_n^* : c \equiv a^e \pmod{n}$

$\Rightarrow c^d \equiv (a^e)^d \equiv a^{t\varphi(n)+1} \equiv a \pmod{n}$

with cipher text  $c$

# PKCS#1

- Textbook RSA has some problems:
  - Common modulus
  - Blinding
  - Low public exponent
- PKCS#1 introduces padding, ...
- 00 01 ff ff ff ... ff ff ff 00 DigestInfo(OID, #)

```
RSAPublicKey publicKey = (RSAPublicKey)
    certificate.getPublicKey();
BigInteger signatureValueBigInteger = new
    BigInteger(signatureValue);
BigInteger messageBigInteger =
    signatureValueBigInteger.modPow(
        publicKey.getPublicExponent(),
        publicKey.getModulus());
```

$c^e \pmod n$

# ASN.1 & DER

- Abstract Syntax Notation One

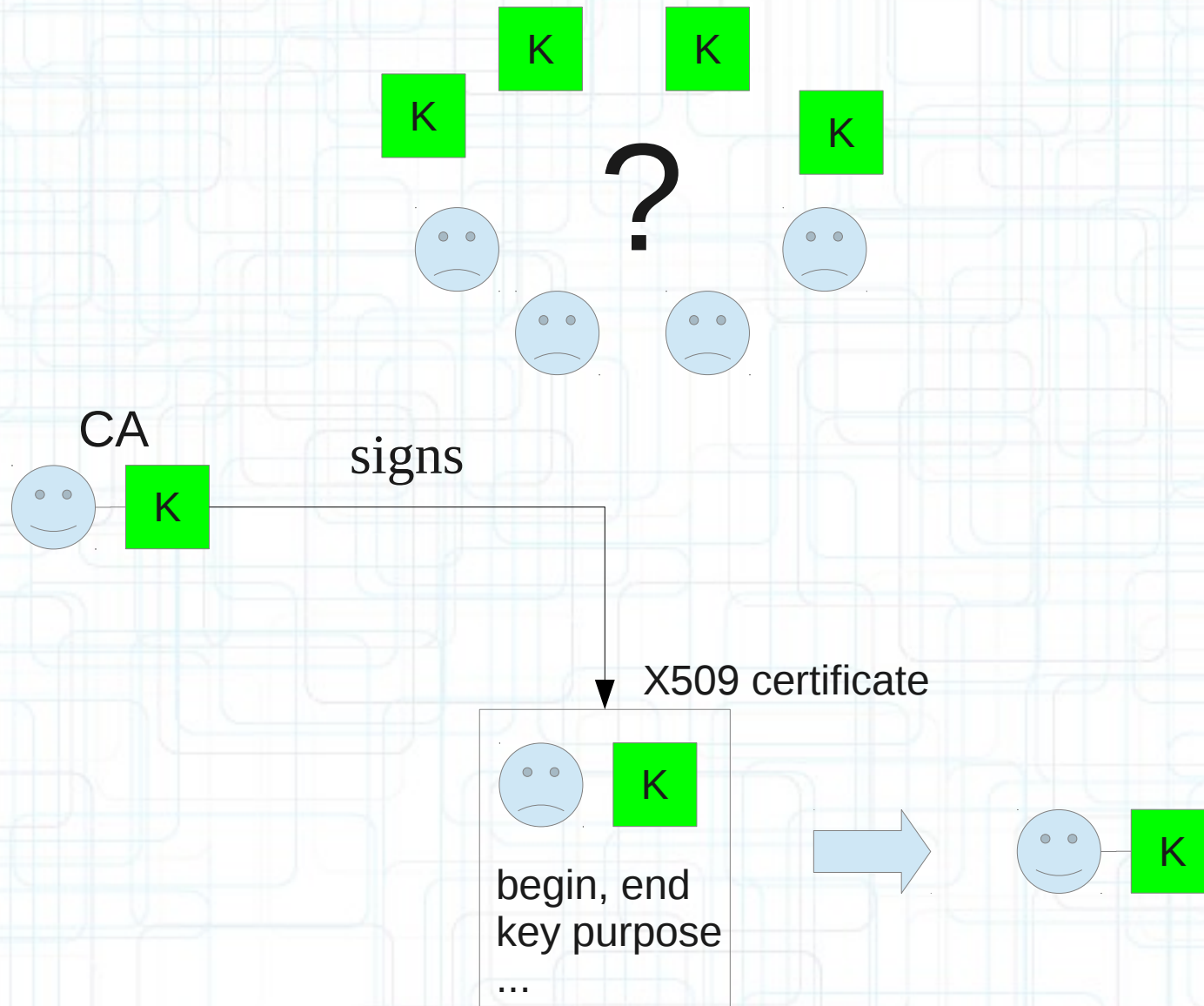
```
FullName ::= SEQUENCE {  
    Name IA5String  
    GivenName IA5String  
}
```

- Distinguished Encoding Rules

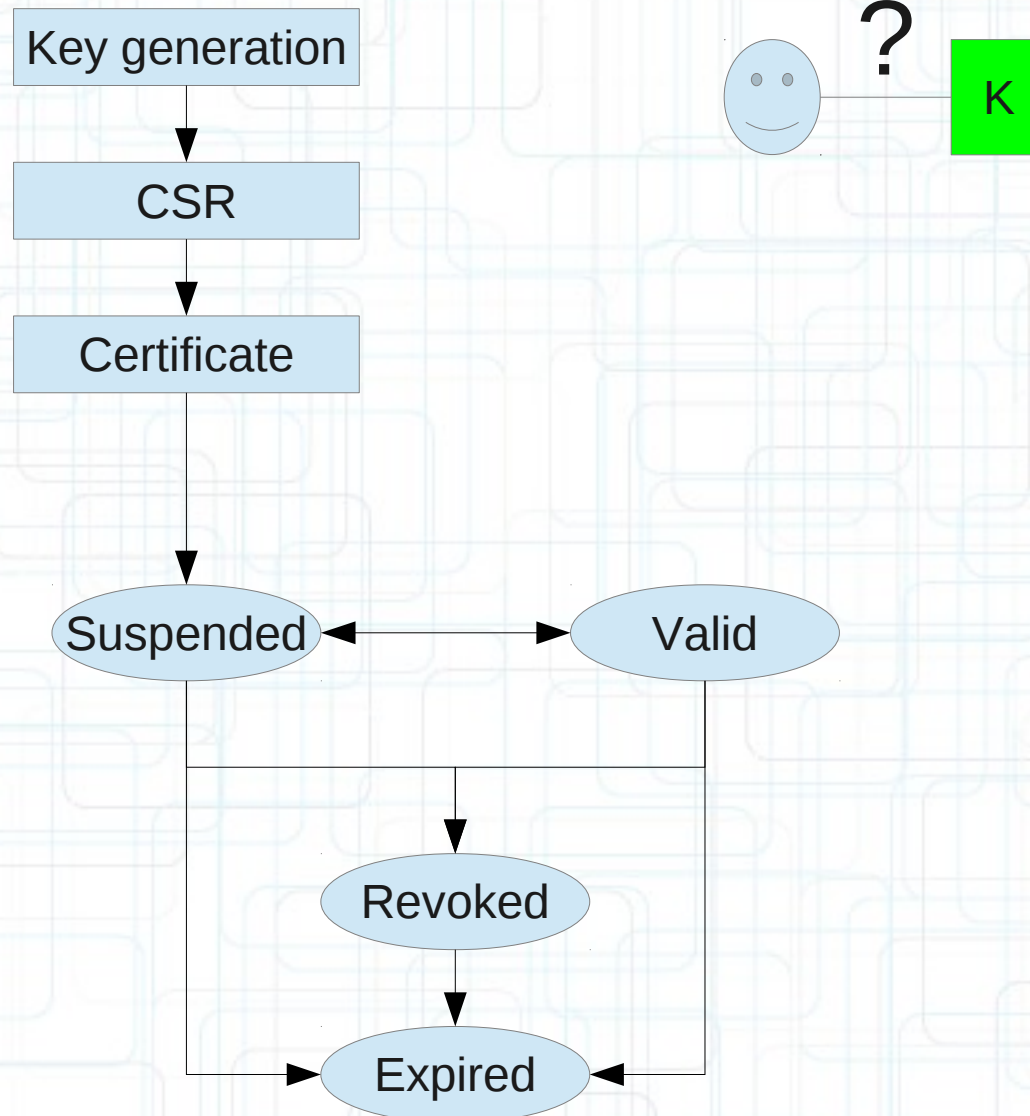
```
30 0a 16 3 "f" "o" "o" 16 3 "b" "a" "r"
```

- Implementation: BouncyCastle

# PKI



# Certificate Life Cycle

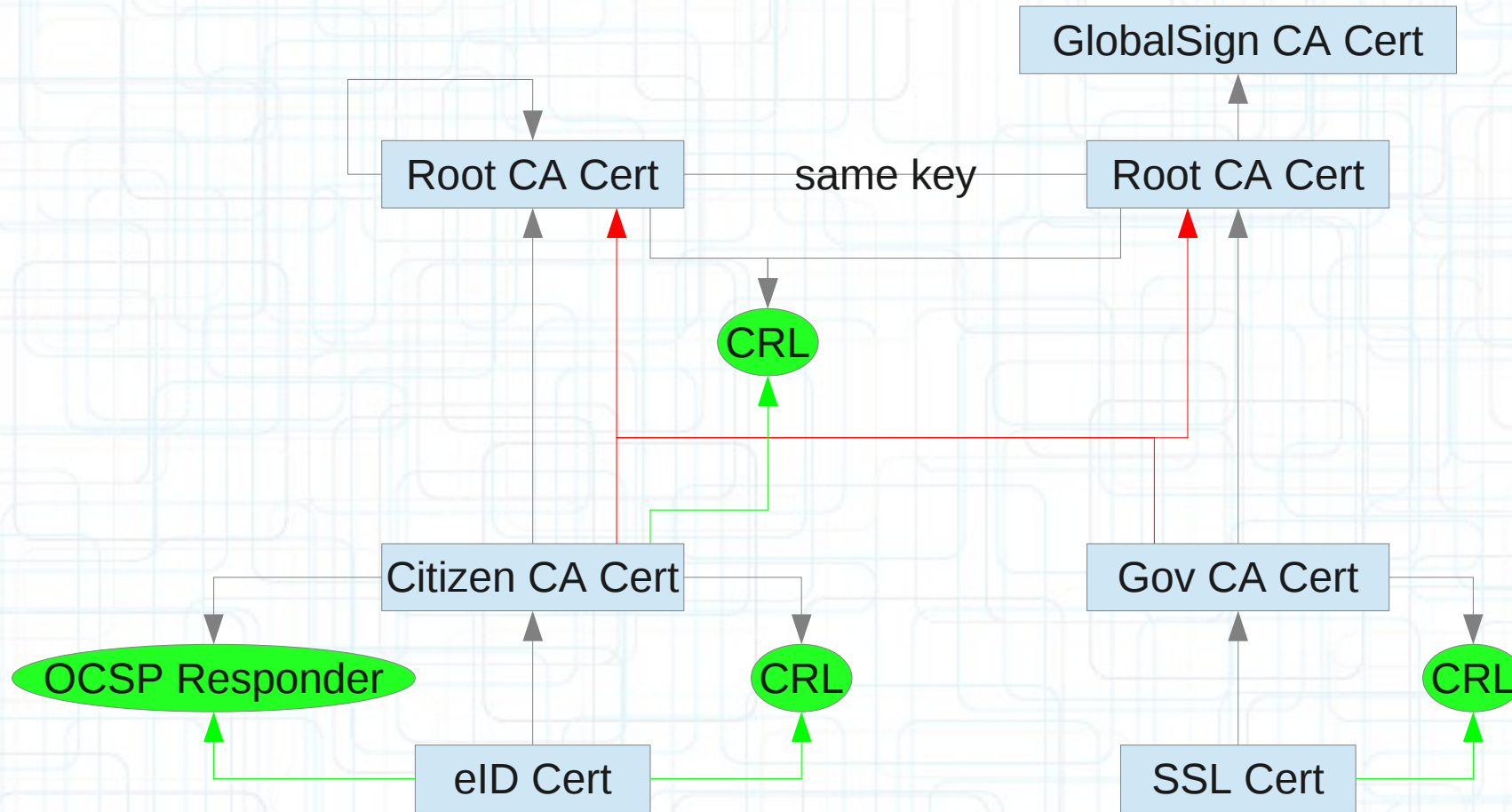


# Certificate Status

- **CRL: Certificate Revocation List**
  - Contains serial numbers of revoked certs
  - Signed by the CA
  - Issued periodically
- **Online Certificate Status Protocol**
  - Online query for certificate status
  - Signed by the CA OCSP Responder



# eID PKI Infrastructure

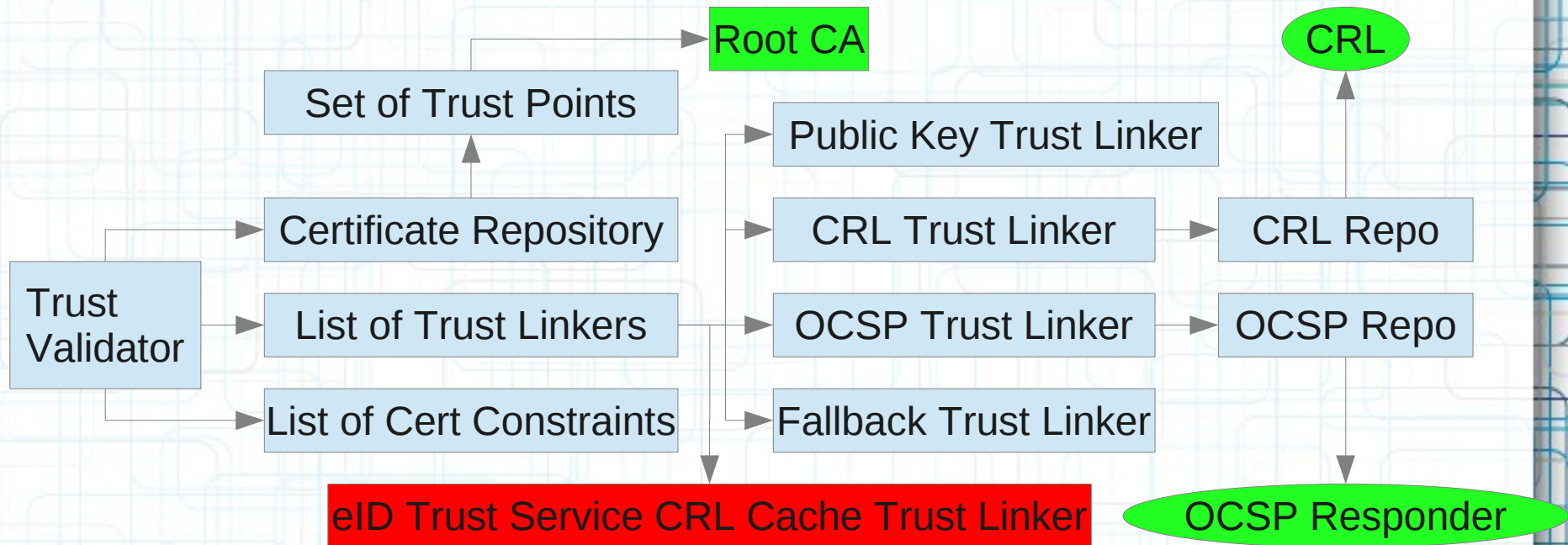


# X509 Validation: jTrust

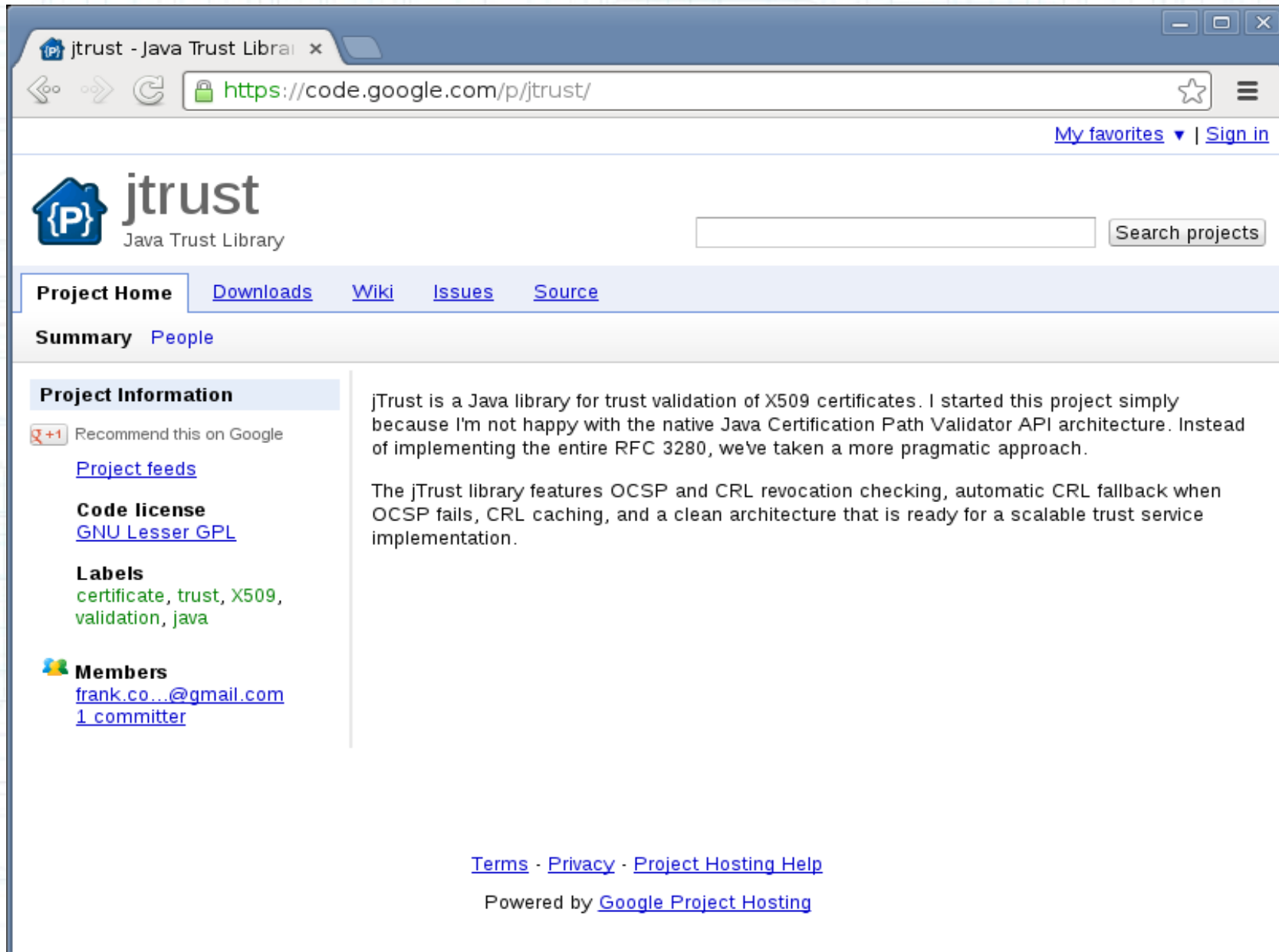
- Alternative to Java Cert Path API
- Java library with flexible architecture
- Readable code

```
Certificate[] authnCertificateChain = ...  
  
Security.addProvider(new BouncyCastleProvider());  
  
TrustValidator trustValidator =  
    BelgianTrustValidatorFactory.createTrustValidator();  
  
trustValidator.isTrusted(authnCertificateChain);
```

# jTrust Architecture



# X509 Validation: jTrust



The screenshot shows a web browser window with the address bar displaying `https://code.google.com/p/jtrust/`. The page title is "jtrust - Java Trust Library". The main content area features the "jtrust" logo and the text "Java Trust Library". Below the logo is a search bar with the text "Search projects". The navigation menu includes "Project Home", "Downloads", "Wiki", "Issues", and "Source". The "Summary" section is active, showing "Project Information". The "Project Information" section includes a "Recommend this on Google" button, "Project feeds", "Code license" (GNU Lesser GPL), "Labels" (certificate, trust, X509, validation, java), and "Members" (frank.co...@gmail.com, 1 committer). The main text describes the project as a Java library for trust validation of X509 certificates, mentioning the RFC 3280 and a pragmatic approach. The footer contains links for "Terms", "Privacy", and "Project Hosting Help", and a note "Powered by Google Project Hosting".

jtrust - Java Trust Library

<https://code.google.com/p/jtrust/> My favorites | Sign in


**jtrust**  
Java Trust Library

Search projects

[Project Home](#) [Downloads](#) [Wiki](#) [Issues](#) [Source](#)

Summary [People](#)


**Project Information**

 Recommend this on Google

[Project feeds](#)

**Code license**  
[GNU Lesser GPL](#)

**Labels**  
certificate, trust, X509, validation, java

 **Members**  
[frank.co...@gmail.com](#)  
[1 committer](#)

jTrust is a Java library for trust validation of X509 certificates. I started this project simply because I'm not happy with the native Java Certification Path Validator API architecture. Instead of implementing the entire RFC 3280, we've taken a more pragmatic approach.

The jTrust library features OCSP and CRL revocation checking, automatic CRL fallback when OCSP fails, CRL caching, and a clean architecture that is ready for a scalable trust service implementation.

[Terms](#) - [Privacy](#) - [Project Hosting Help](#)

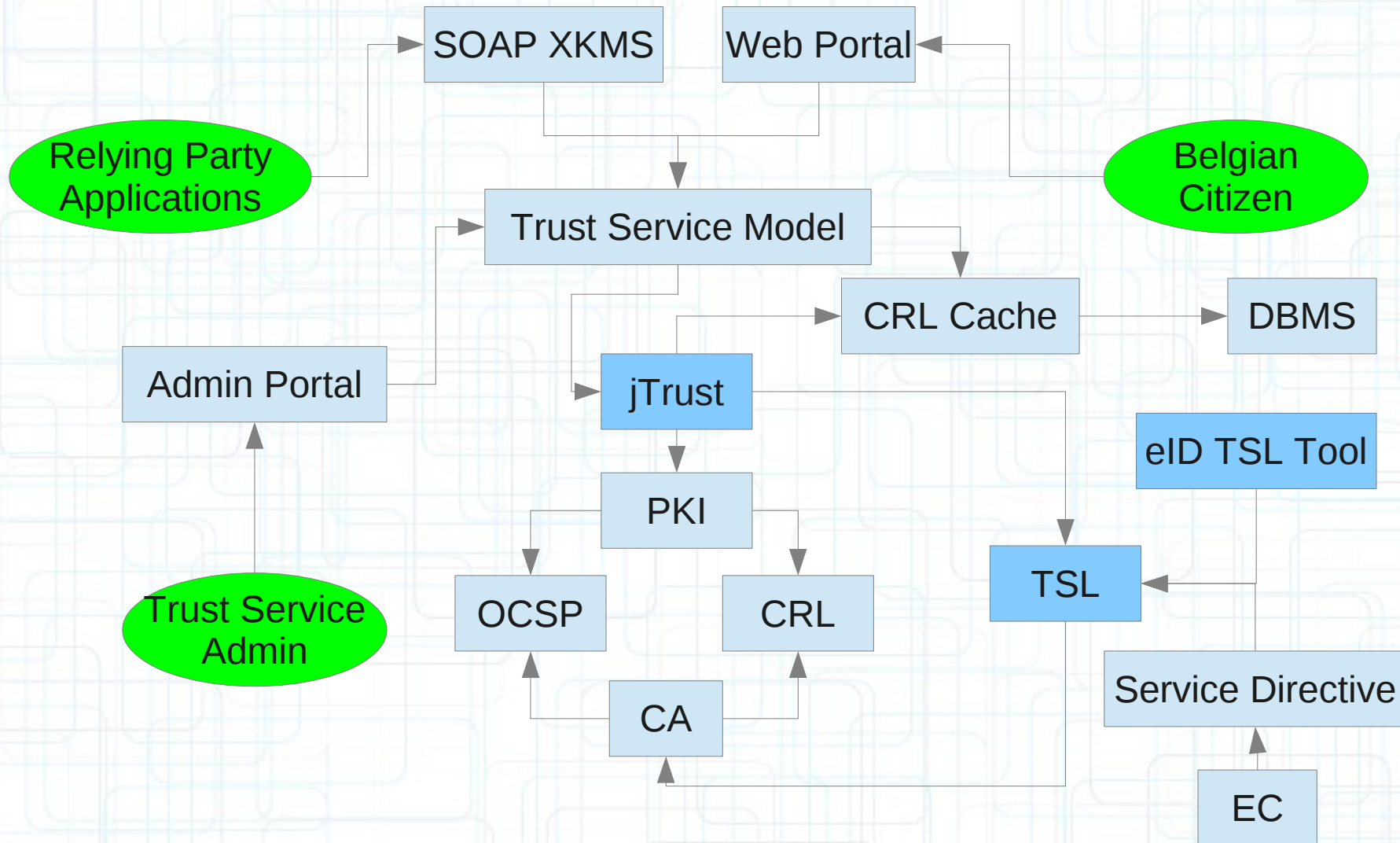
Powered by [Google Project Hosting](#)

# X509 Validation: Trust Service

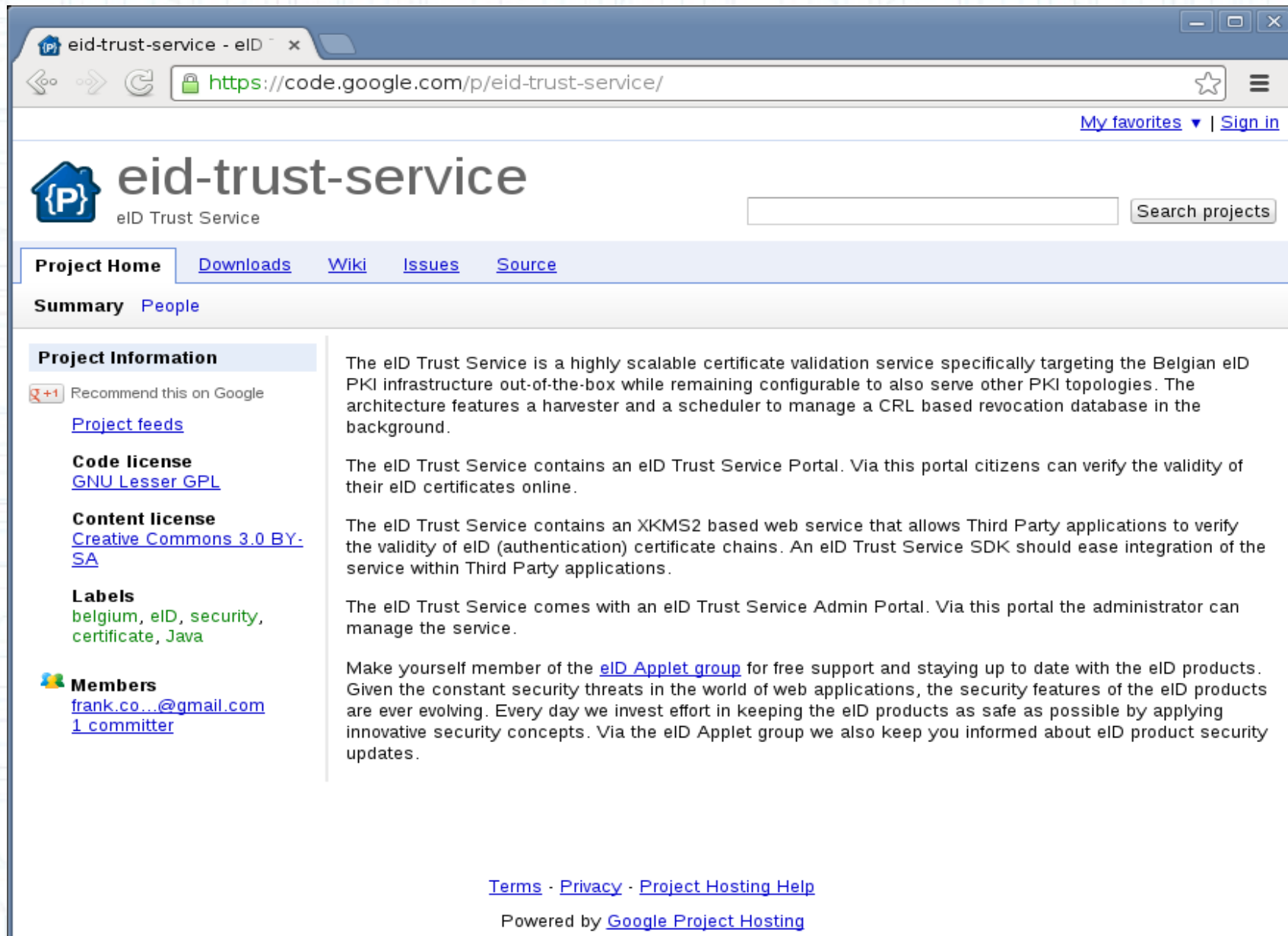
- jTrust extension: CRL cache (Java EE)
- XKMS2 web service interface
- Java SDK

```
List<X509Certificate> authnCertificateChain = ...  
  
XKMS2Client client = new XKMS2Client(  
    "https://www.e-contract.be/eid-trust-service-ws/xkms2");  
  
client.validate("BE-AUTH", authnCertificateChain);
```

# eID Trust Service Architecture



# X509 Validation: Trust Service



The screenshot shows a web browser window displaying the Google Code project page for 'eid-trust-service'. The browser's address bar shows the URL 'https://code.google.com/p/eid-trust-service/'. The page header includes the project name 'eid-trust-service' and the subtitle 'eID Trust Service'. Below the header, there are navigation links for 'Project Home', 'Downloads', 'Wiki', 'Issues', and 'Source'. The main content area is divided into two columns. The left column contains 'Project Information' with links for 'Recommend this on Google', 'Project feeds', 'Code license' (GNU Lesser GPL), 'Content license' (Creative Commons 3.0 BY-SA), 'Labels' (belgium, eID, security, certificate, Java), and 'Members' (frank.co...@gmail.com, 1 committer). The right column contains three paragraphs of text describing the service's capabilities, the eID Trust Service Portal, and the eID Trust Service Admin Portal. At the bottom of the page, there are links for 'Terms', 'Privacy', and 'Project Hosting Help', and a note that the project is 'Powered by Google Project Hosting'.

eid-trust-service - eID x

https://code.google.com/p/eid-trust-service/

My favorites | Sign in

## eid-trust-service


eID Trust Service

Search projects

[Project Home](#) [Downloads](#) [Wiki](#) [Issues](#) [Source](#)

Summary [People](#)

### Project Information


 Recommend this on Google

[Project feeds](#)

**Code license**  
[GNU Lesser GPL](#)

**Content license**  
[Creative Commons 3.0 BY-SA](#)

**Labels**  
belgium, eID, security, certificate, Java

 **Members**  
[frank.co...@gmail.com](#)  
[1 committer](#)

The eID Trust Service is a highly scalable certificate validation service specifically targeting the Belgian eID PKI infrastructure out-of-the-box while remaining configurable to also serve other PKI topologies. The architecture features a harvester and a scheduler to manage a CRL based revocation database in the background.

The eID Trust Service contains an eID Trust Service Portal. Via this portal citizens can verify the validity of their eID certificates online.

The eID Trust Service contains an XKMS2 based web service that allows Third Party applications to verify the validity of eID (authentication) certificate chains. An eID Trust Service SDK should ease integration of the service within Third Party applications.

The eID Trust Service comes with an eID Trust Service Admin Portal. Via this portal the administrator can manage the service.

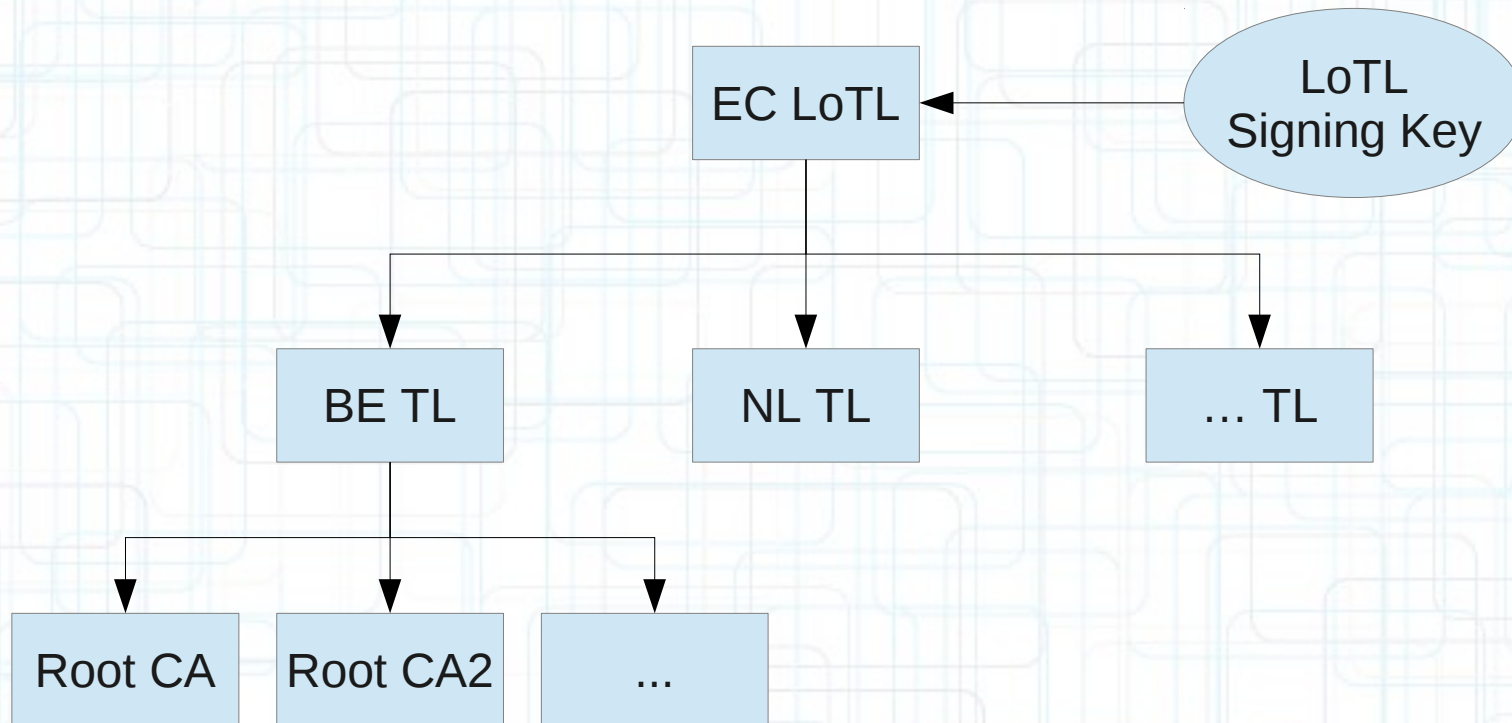
Make yourself member of the [eID Applet group](#) for free support and staying up to date with the eID products. Given the constant security threats in the world of web applications, the security features of the eID products are ever evolving. Every day we invest effort in keeping the eID products as safe as possible by applying innovative security concepts. Via the eID Applet group we also keep you informed about eID product security updates.

[Terms](#) - [Privacy](#) - [Project Hosting Help](#)

Powered by [Google Project Hosting](#)

# Bootstrapping Trust

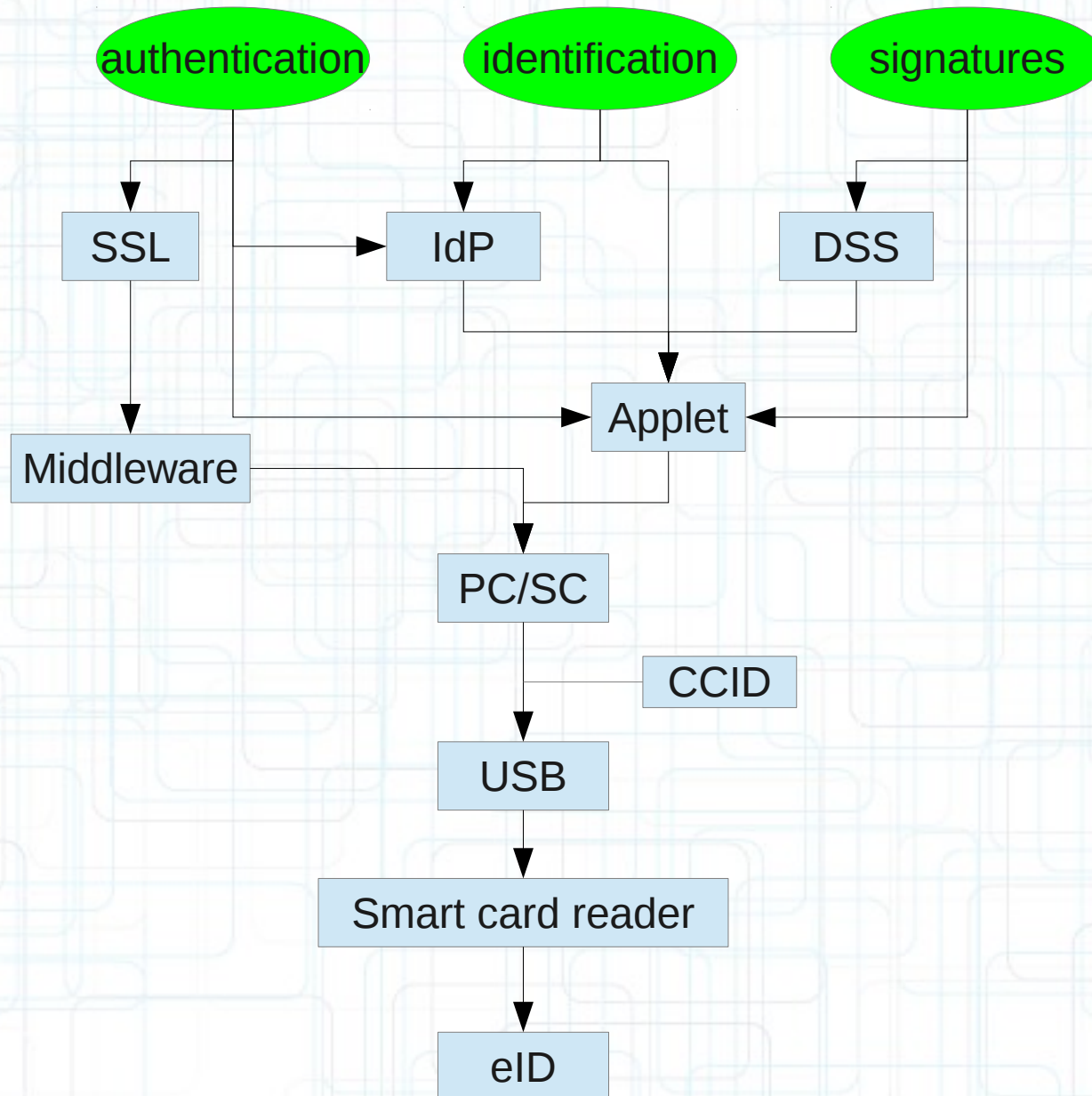
- Trusted Lists & List of Trusted Lists (LoTL)
- Dynamic updating of the EU trust realm
- Bootstrapping reduced to a single key



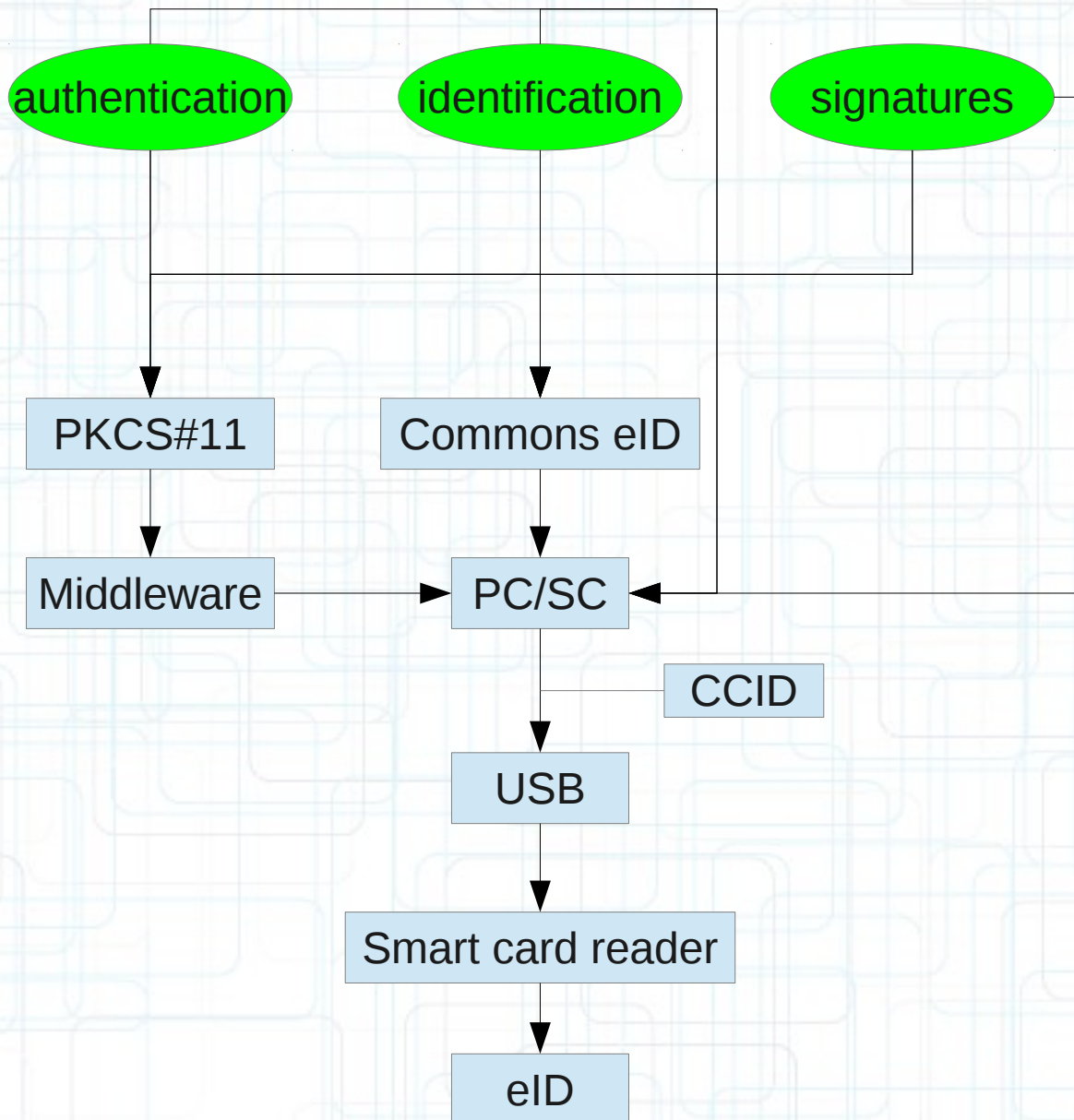




# eID Web Integration



# eID Desktop Integration



# PC/SC

```
TerminalFactory terminalFactory = TerminalFactory.getDefault();
CardTerminals cardTerminals = terminalFactory.terminals();
CardTerminal cardTerminal = cardTerminals.list().get(0);
Card card = cardTerminal.connect("T=0");
CardChannel cardChannel = card.getBasicChannel();

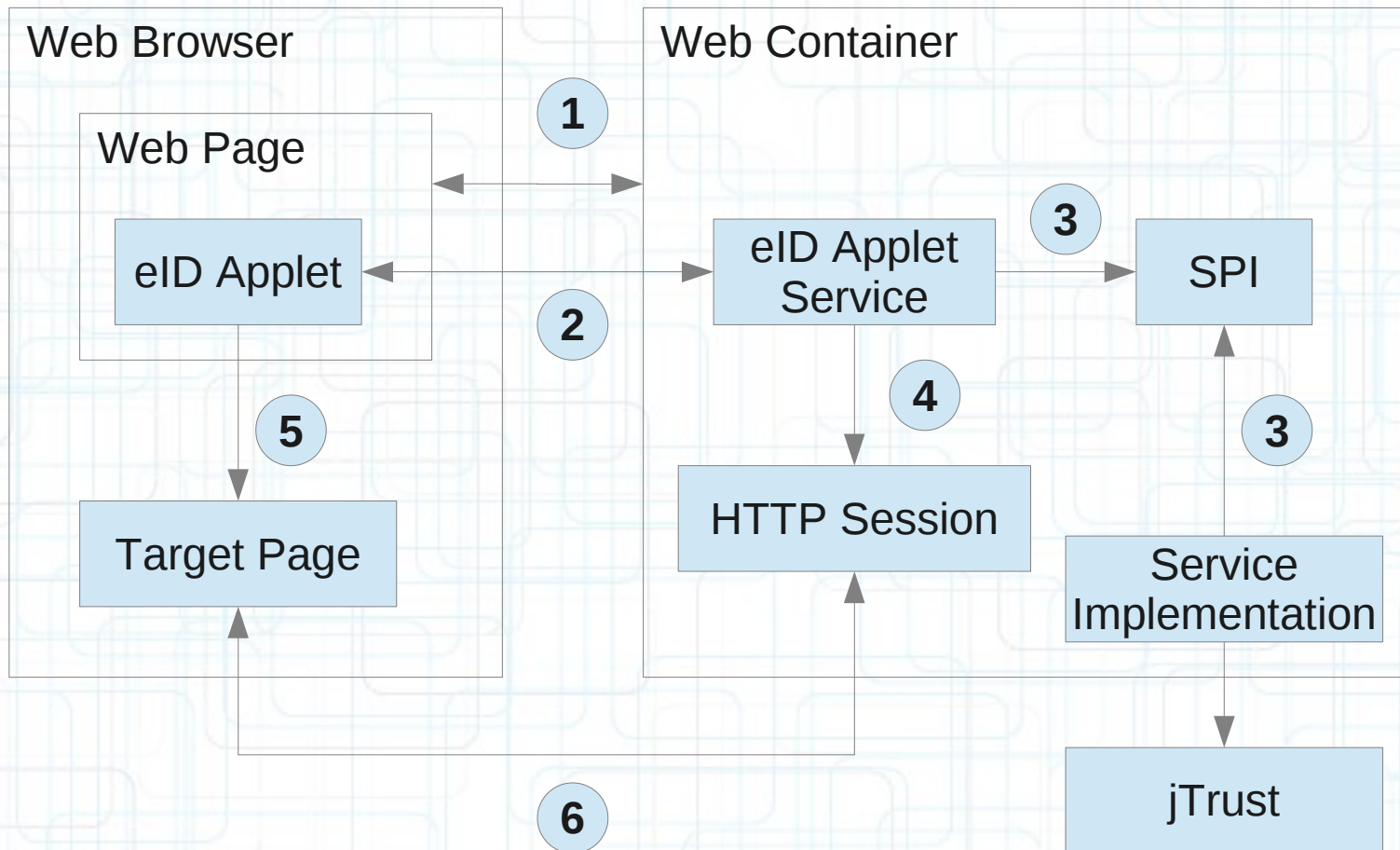
// select file
cardChannel.transmit(new CommandAPDU(0x00, 0xA4, 0x08, 0x0C,
    new byte[] { 0x3F, 0x00, (byte) 0xDF, 0x01, 0x40, 0x35 }));

ByteArrayOutputStream baos = new ByteArrayOutputStream();
int offset = 0;
ResponseAPDU responseApu;
do {
    // read binary
    responseApu = cardChannel.transmit(new CommandAPDU(0x00,
        0xB0, offset >> 8, offset & 0xFF, 0xff));
    baos.write(responseApu.getData());
    offset += responseApu.getData().length;
} while (responseApu.getData().length == 0xff);

BufferedImage photo =
    ImageIO.read(new ByteArrayInputStream(baos.toByteArray()));
JOptionPane.showMessageDialog(null, new ImageIcon(photo));
```



# eID Applet

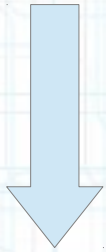


- eID Applet Service targets Java EE servlet container only

# eID Applet Example

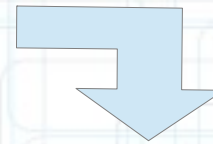
## identify-the-user.html

```
<script src="https://www.java.com/js/deployJava.js"></script>
<script>
  var attributes = {
    code : 'be.fedict.eid.applet.Applet.class',
    archive : 'eid-applet-package-1.1.0.Beta4.jar',
    width : 600,
    height : 300
  };
  var parameters = {
    TargetPage : 'identification-result-page.jsp',
    AppletService : 'applet-service',
  };
  var version = '1.6';
  deployJava.runApplet(attributes, parameters, version);
</script>
```



## web.xml

```
<servlet>
  <servlet-name>AppletServiceServlet</servlet-name>
  <servlet-class>be.fedict.eid.applet.service.AppletServiceServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>AppletServiceServlet</servlet-name>
  <url-pattern>/applet-service</url-pattern>
</servlet-mapping>
```



## identification-result-page.jsp

```
<%@page import="be.fedict.eid.applet.service.Identity"%>
<html>
<body>
  <%=((Identity) session.getAttribute("eid.identity")).name%>
</body>
</html>
```



# eID Applet


eID Applet Beta Site - Mozilla Firefox 4.0 Beta 12

File Edit View History Bookmarks Tools Help

eID Applet Beta Site

e-contract.be https://www.e-contract.be/eid-applet-beta/ Google

FedICT Mail eid-applet Slashdot Java.net Redmine eID dev

**fedict**  eID Applet Beta Site  
@-novating government

Home **Tests** Feedback Results Developers Admin

**eID Applet Identification Test**

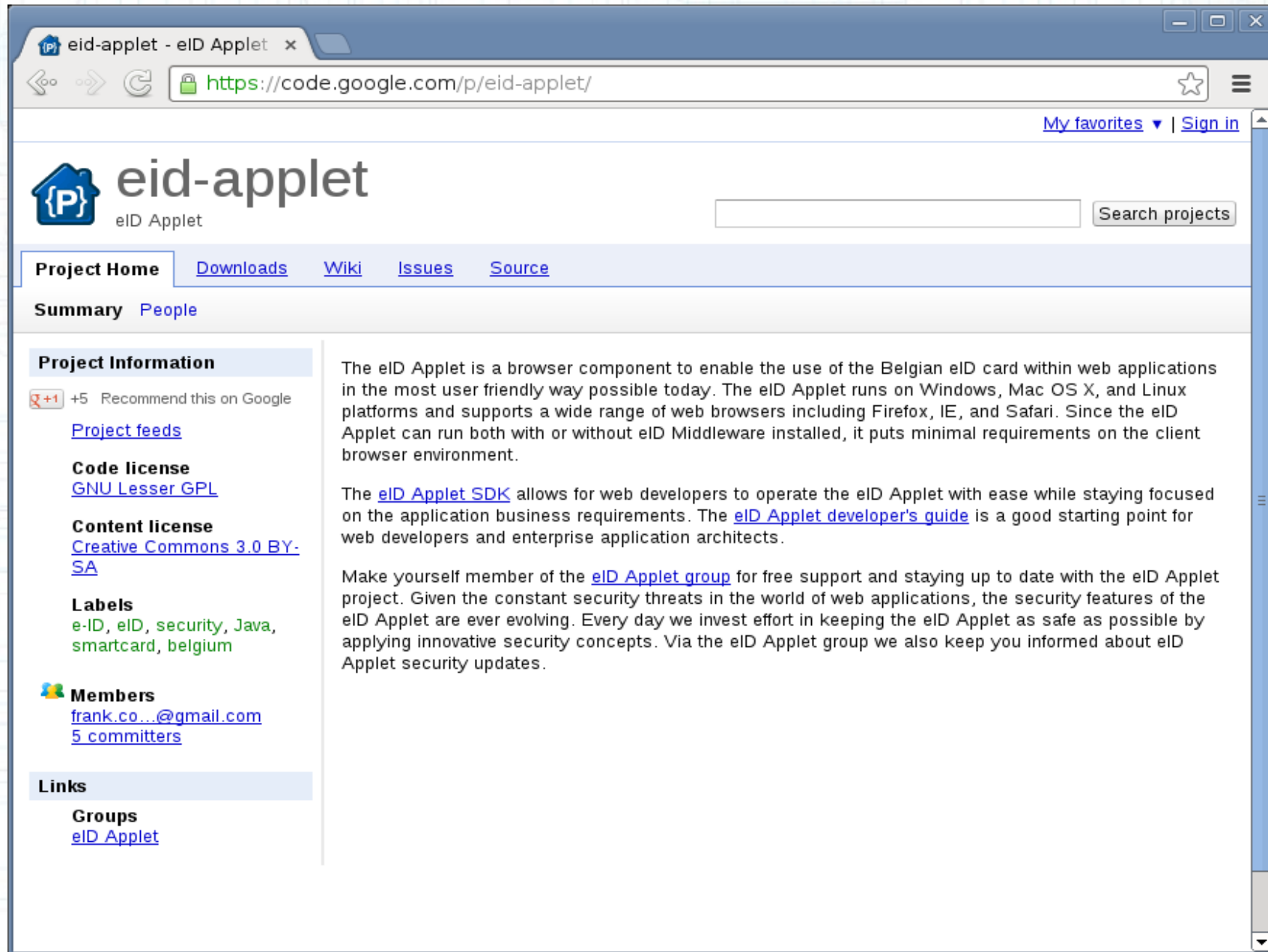
Via this page you can test the eID Applet Identification functionality.

**Please insert your eID card...**

Progress bar: [ ]

Details >>

# eID Applet



The screenshot shows a web browser window with the address bar displaying <https://code.google.com/p/eid-applet/>. The page title is "eid-applet" and the subtitle is "eID Applet". The navigation menu includes "Project Home", "Downloads", "Wiki", "Issues", and "Source". The "Summary" section is active, showing "Project Information", "Code license", "Content license", "Labels", "Members", and "Links".

**Project Information**

+5 Recommend this on Google

[Project feeds](#)

**Code license**  
[GNU Lesser GPL](#)

**Content license**  
[Creative Commons 3.0 BY-SA](#)

**Labels**  
e-ID, eID, security, Java, smartcard, belgium

**Members**  
[frank.co...@gmail.com](#)  
[5 committers](#)

**Links**  
**Groups**  
[eID Applet](#)

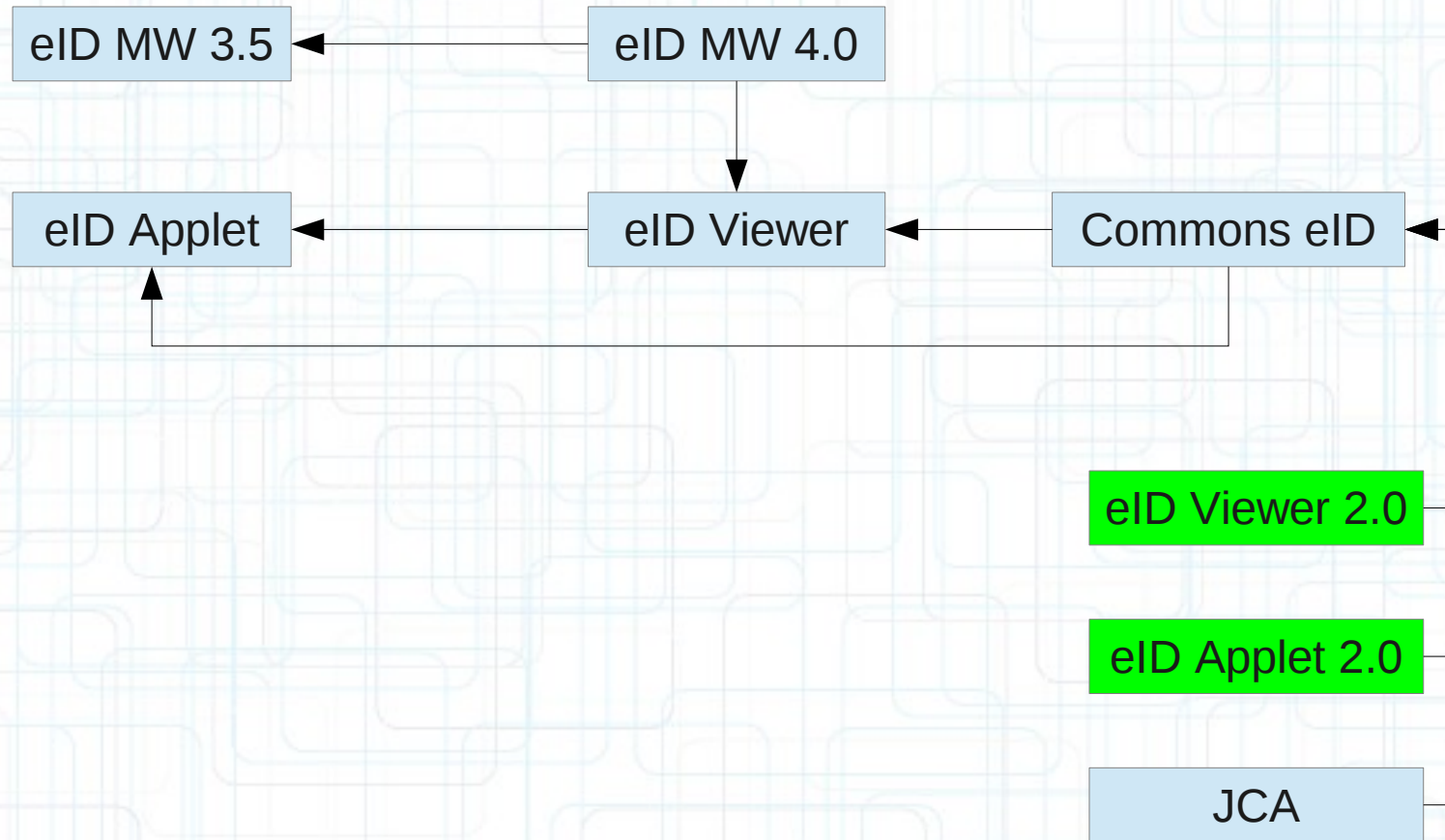
The eID Applet is a browser component to enable the use of the Belgian eID card within web applications in the most user friendly way possible today. The eID Applet runs on Windows, Mac OS X, and Linux platforms and supports a wide range of web browsers including Firefox, IE, and Safari. Since the eID Applet can run both with or without eID Middleware installed, it puts minimal requirements on the client browser environment.

The [eID Applet SDK](#) allows for web developers to operate the eID Applet with ease while staying focused on the application business requirements. The [eID Applet developer's guide](#) is a good starting point for web developers and enterprise application architects.

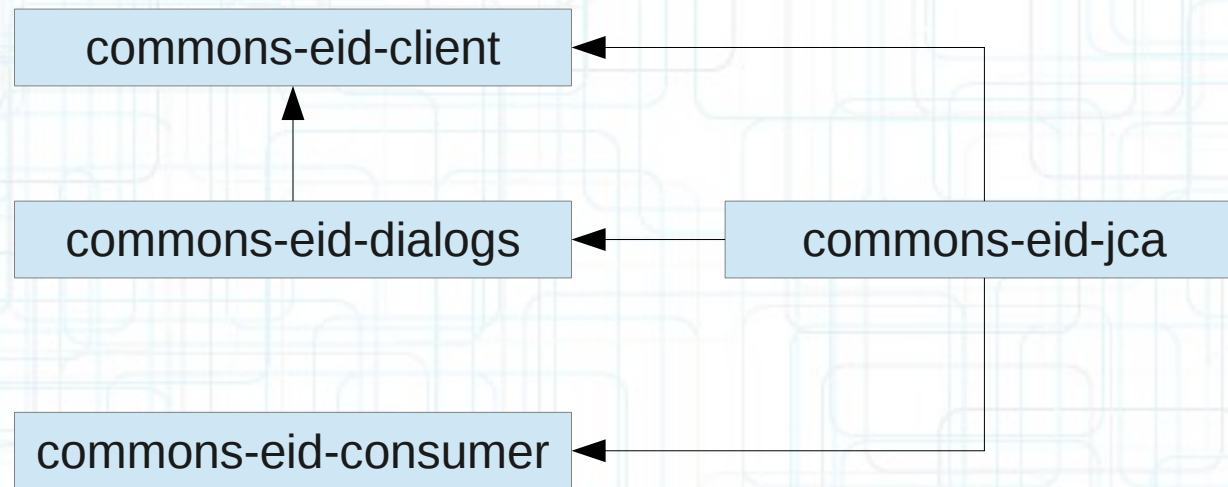
Make yourself member of the [eID Applet group](#) for free support and staying up to date with the eID Applet project. Given the constant security threats in the world of web applications, the security features of the eID Applet are ever evolving. Every day we invest effort in keeping the eID Applet as safe as possible by applying innovative security concepts. Via the eID Applet group we also keep you informed about eID Applet security updates.



# Commons eID



# Commons eID Components



- Desktop: commons-eid-jca, or lower-level
- Client-Server:
  - Client: commons-eid-client, dialogs
  - Server: commons-eid-consumer

# BeID JCA Security Provider

- KeyStore
  - Load key material
- Signature
  - Targeting eID key material
- SecureRandom
  - eID based hardware secure random
- X509KeyManager
  - eID based mutual SSL

```
Security.addProvider(new BeIDProvider());
```

# JCA KeyStore

```
Security.addProvider(new BeIDProvider());

KeyStore keyStore = KeyStore.getInstance("BeID");
keyStore.load(null);

PrivateKeyEntry privateKeyEntry =
    (PrivateKeyEntry)
    keyStore.getEntry("Authentication", null);
PrivateKey privateKey =
    privateKeyEntry.getPrivateKey();

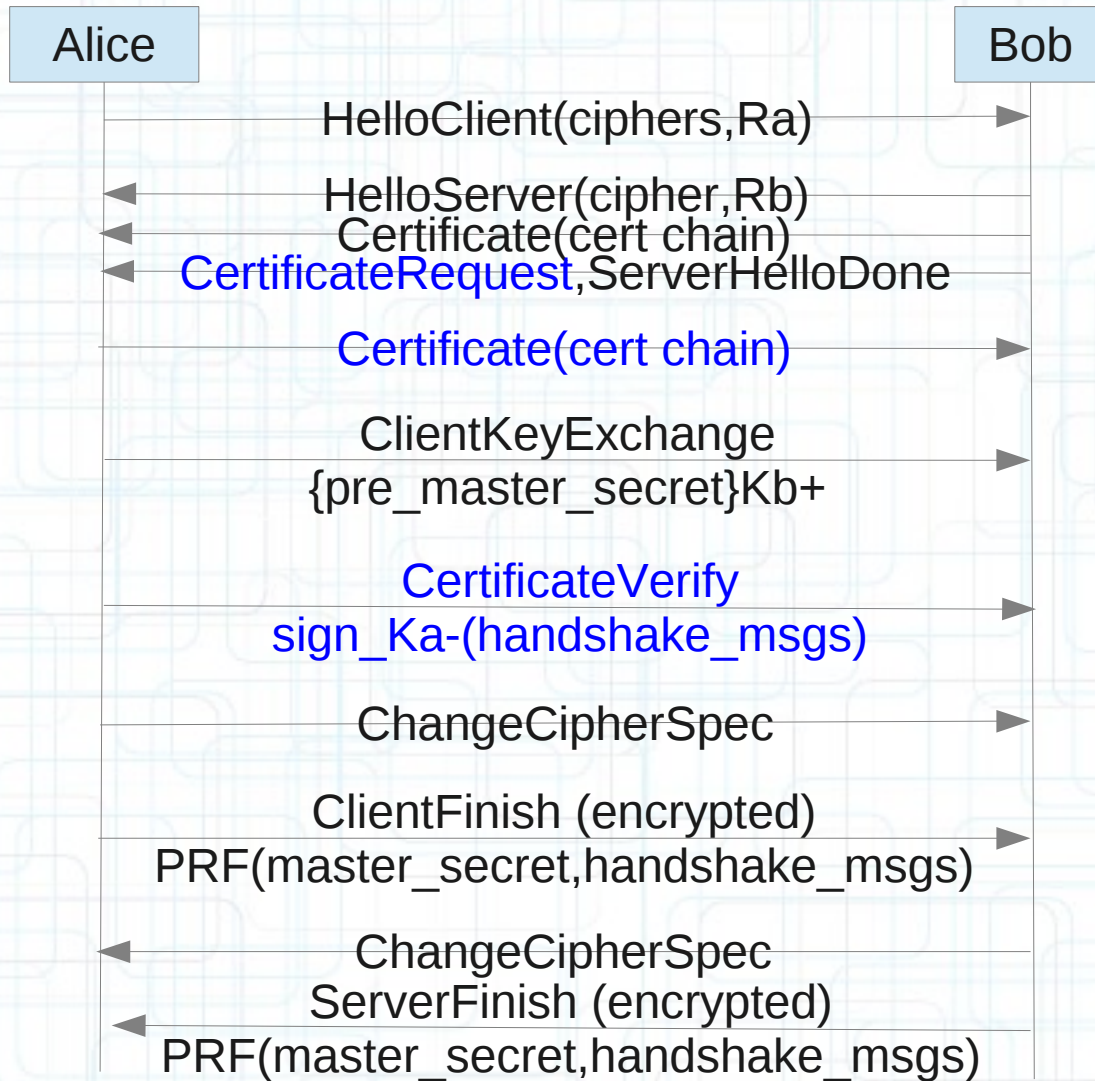
X509Certificate certificate = (X509Certificate)
    privateKeyEntry.getCertificate();
PublicKey publicKey = certificate.getPublicKey();
```

# JCA Signatures

- Supported algorithms:
  - SHA 1/224/256/384/512
  - NONE
  - RIPEMD 128/160/256
- Delegate init: SupportedKeyClasses

```
Signature signature =  
    Signature.getInstance("SHA256withRSA");  
signature.initSign(privateKey);  
assertTrue(signature.getProvider() instanceof  
    BeIDProvider);  
  
final byte[] toBeSigned =  
    "hello world".getBytes();  
signature.update(toBeSigned);  
final byte[] signatureValue = signature.sign();
```

# Mutual SSL



**Ra:** random by A  
**Rb:** random by B  
**Kb+:** public key of B  
**pre\_master\_secret:** random by A  
**Ka-:** private key of A  
**PRF:** pseudo-random function

# Commons eID: SSL

```
Security.addProvider(new BeIDProvider());  
KeyManagerFactory keyManagerFactory =  
    KeyManagerFactory.getInstance("BeID");
```

```
BeIDManagerFactoryParameters spec = new  
    BeIDManagerFactoryParameters();  
spec.setAutoRecovery(true);  
spec.setCardReaderStickiness(true);  
keyManagerFactory.init(spec);
```

```
SecureRandom secureRandom = new SecureRandom();  
sslContext.init(  
    keyManagerFactory.getKeyManagers(), null,  
    secureRandom);
```

```
SSLSocketFactory sslSocketFactory =  
    sslContext.getSocketFactory();
```

# Commons eID Identification

- Via lower-level API:
  - Synchronous
  - Asynchronous: event driven

```
BeIDCards beIDCards = new BeIDCards();  
BeIDCard beIDCard = beIDCards.getOneBeIDCard();
```

```
byte[] photoData =  
    beIDCard.readFile(FileType.Photo);
```

```
BufferedImage photo = ImageIO.read(  
    new ByteArrayInputStream(photoData));  
JOptionPane.showMessageDialog(null,  
    new ImageIcon(photo));
```



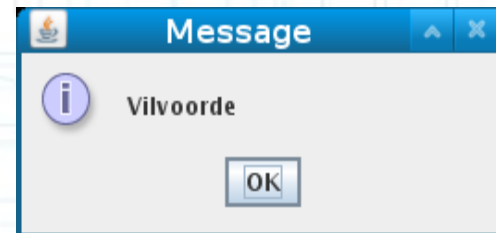


# Commons eID: events

```
BeIDCardManager beIDCardManager = new BeIDCardManager();
final JLabel label = new JLabel();
beIDCardManager.addBeIDCardEventListener(new BeIDCardEventsListener() {
    @Override
    public void eIDCardRemoved(CardTerminal cardTerminal, BeIDCard card) {
        label.setText("insert card...");
    }

    @Override
    public void eIDCardInserted(CardTerminal cardTerminal, BeIDCard card) {
        try {
            Address address = TlvParser.parse(
                card.readFile(FileType.Address), Address.class);
            label.setText(address.municipality);
        } catch (Exception e) {
            label.setText("error");
        }
    }

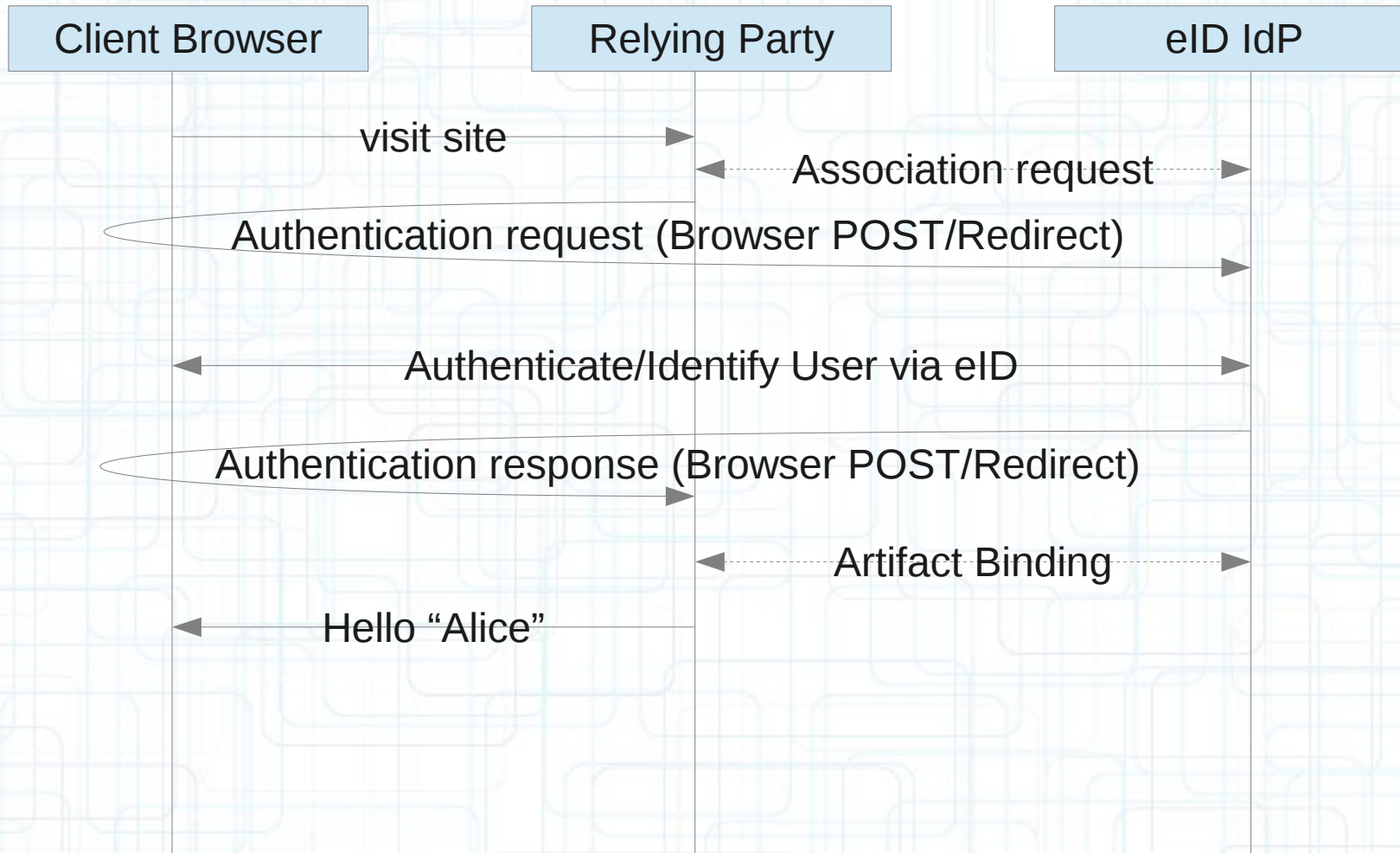
    @Override
    public void eIDCardEventsInitialized() {
    }
});
beIDCardManager.start();
JOptionPane.showMessageDialog(null, label);
```



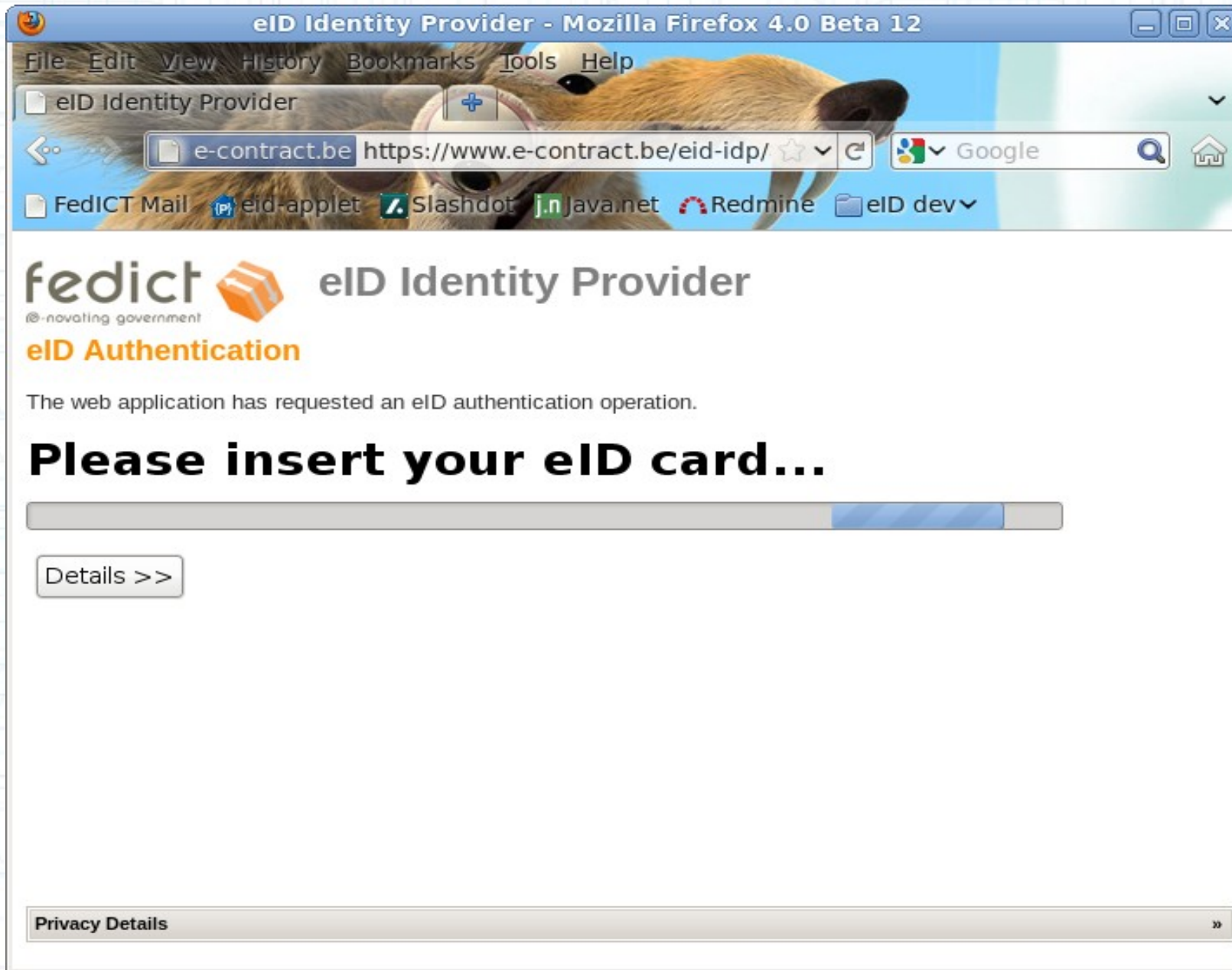
# eID Identity Provider

- Supports different OPEN authentication protocols:
  - OpenID 2.0: PHP, Drupal, ...
  - SAML2 Browser POST: Java EE, ...
  - WS-Federation: ASP.NET, ...
- Offers 3 eID based flows:
  - Identification
  - Authentication
  - Identification combined with authentication
- Configurable Relying Parties via admin console
- Comes in JBoss AS 6.1 distributions:
  - MySQL, PostgreSQL, Oracle

# Authentication Protocols



# eID Identity Provider




eID Identity Provider - Mozilla Firefox 4.0 Beta 12

File Edit View History Bookmarks Tools Help

eID Identity Provider

e-contract.be <https://www.e-contract.be/eid-idp/> Google

FedICT Mail eid-applet Slashdot java.net Redmine eID dev

**fedict**  **eID Identity Provider**  
@-novating government

**eID Authentication**

The web application has requested an eID authentication operation.

**Please insert your eID card...**

Progress bar: [-----|-----]

[Details >>](#)


[Privacy Details](#) »

# eID Identity Provider

The screenshot shows a web browser window with the address bar displaying `https://code.google.com/p/eid-idp/`. The page features the project logo, navigation tabs for 'Project Home', 'Downloads', 'Wiki', 'Issues', and 'Source', and a search bar. The main content area is divided into a left sidebar and a main text area. The sidebar includes 'Project Information' with a recommendation count, 'Project feeds', 'Code license' (GNU Lesser GPL), 'Content license' (Creative Commons 3.0 BY-SA), 'Labels' (eID, e-ID, security, Java, belgium, identityprovider), and 'Members' (frank.co...@gmail.com, 3 committers). The main text area contains a description of the eID Identity Provider, a list of supported protocols, a scope statement, an implementation note, and a call to action to join the eID Applet group.

eid-idp - eID Identity Pr x

<https://code.google.com/p/eid-idp/> My favorites | Sign in

 **eid-idp**  
eID Identity Provider

Search projects

**Project Home** Downloads Wiki Issues Source

Summary People

**Project Information**

+2 Recommend this on Google

[Project feeds](#)

**Code license**  
[GNU Lesser GPL](#)

**Content license**  
[Creative Commons 3.0 BY-SA](#)

**Labels**  
eID, e-ID, security, Java, belgium, identityprovider

**Members**  
[frank.co...@gmail.com](#)  
[3 committers](#)

The eID Identity Provider is a simple IdP using the eID as authentication token. The eID IdP supports different authentication protocols:

- SAML v2 browser POST profile
- OpenID 2.0 with AX, PAPE, UI extension support
- WS-Federation

The scope of this project is not to create an IDM product. The eID IdP limits itself to the eID card and the attribute set that this authentication token offers. The eID IdP has no local attribute store, nor provides support for other tokens but the eID card.

The eID IdP is implemented as a Java EE application. The eID IdP is using the eID Applet to perform the entity authentication. The eID Trust Service is used for validation of the authentication certificate chain of the citizen.

The eID IdP comes with an eID IdP SDK to ease the task of developers on integrating the eID IdP functionality in web application.

Make yourself member of the [eID Applet group](#) for free support and staying up to date with the eID products. Given the constant security threats in the world of web applications, the security features of the eID products are ever evolving. Every day we invest effort in keeping the eID products as safe as possible by applying innovative security concepts. Via the eID Applet group we also keep you informed about eID product security updates.

# WebScarab Plugins

The screenshot displays the WebScarab application window. The title bar reads "WebScarab". The menu bar includes "File", "View", "Tools", and "Help". The main toolbar contains several tabs: "Scripted", "Fragments", "Fuzzer", "Compare", "Search", "SAML", "OpenID", "WS-Federation", "Summary", "Messages", "Proxy", "Manual Request", "Spider", "Extensions", "XSS/CRLF", and "SessionID Analysis". Below the toolbar, there are sub-tabs for "SAML Browser POST Profile Messages", "Signature Attacks", "Injection Attacks", and "Replay Attacks".

The main message log table is as follows:

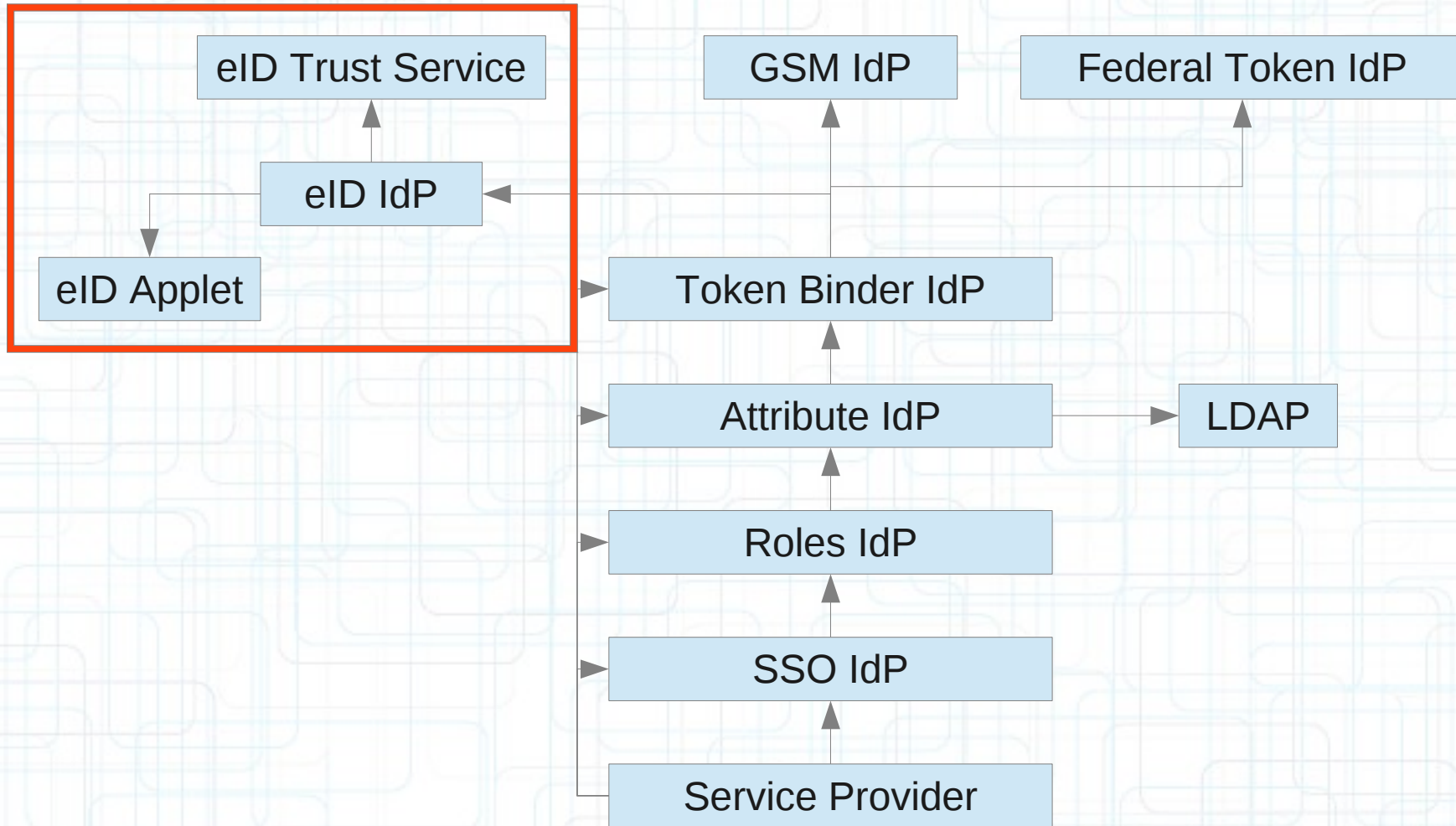
| ID | Date        | Method | Host         | Path         | Parameters  | Status     | Origin | Tag | Size | SAML Type |
|----|-------------|--------|--------------|--------------|-------------|------------|--------|-----|------|-----------|
| 5  | Dec 20, ... | POST   | https://w... | /eid-idp/... | ?languag... | 302 Mov... | Proxy  |     |      | Request   |
| 28 | Dec 20, ... | POST   | http://w...  | /eid-idp/... |             | 302 Mov... | Proxy  |     |      | Response  |

Below the message log, there are tabs for "Encrypted Attributes", "HTML Form", "Relay State", "Analysis", and "About". The "Encrypted Attributes" tab is active, showing a table with columns "Raw", "Text", "XML", "Protocol Signature", and "Attributes". The "Attributes" sub-tab is selected, displaying a table of attribute details:

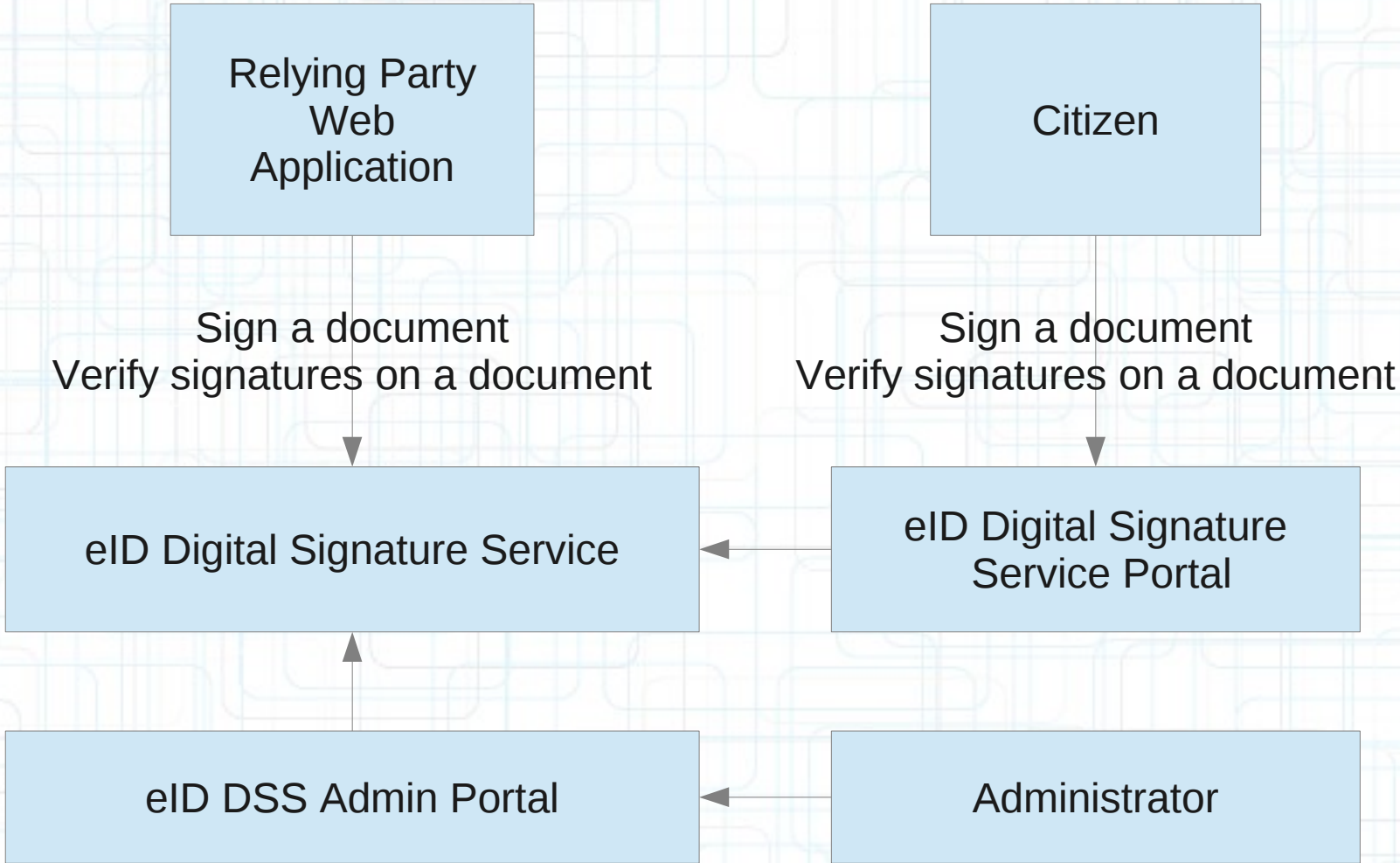
| Name  | Value                                  |
|---|--|
| be:fedict:eid:idp:nationality                                     | Belg                                   |
| be:fedict:eid:idp:pob   | Dendermonde                            |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/locality    | Vilvoorde                              |
| be:fedict:eid:idp:photo   | /9j/4AAQSkZJRgABAgEBLAEsAAD/2wBDABA... |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname     | Cornelis                               |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/gender      | 1                                      |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname   | Frank Henri                            |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/dateofbirth | 1979-10-25T00:00:00.000Z               |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/postalcode  | 1800                                   |
| be:fedict:eid:idp:age   | 32                                     |

A green status bar at the bottom of the application window indicates "Used 136.88 of 896.0MB".

# Identity and Access Management

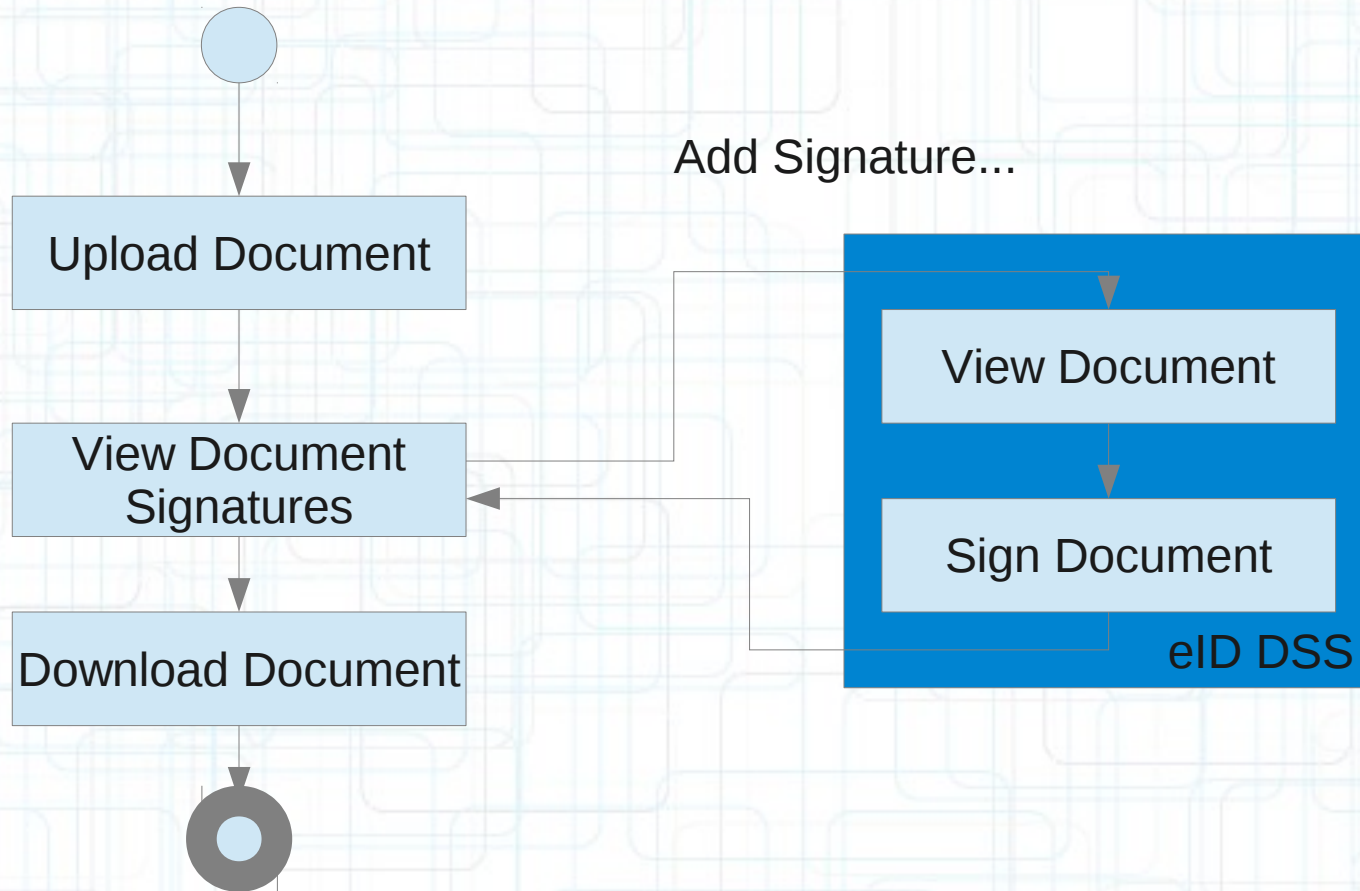


# eID DSS



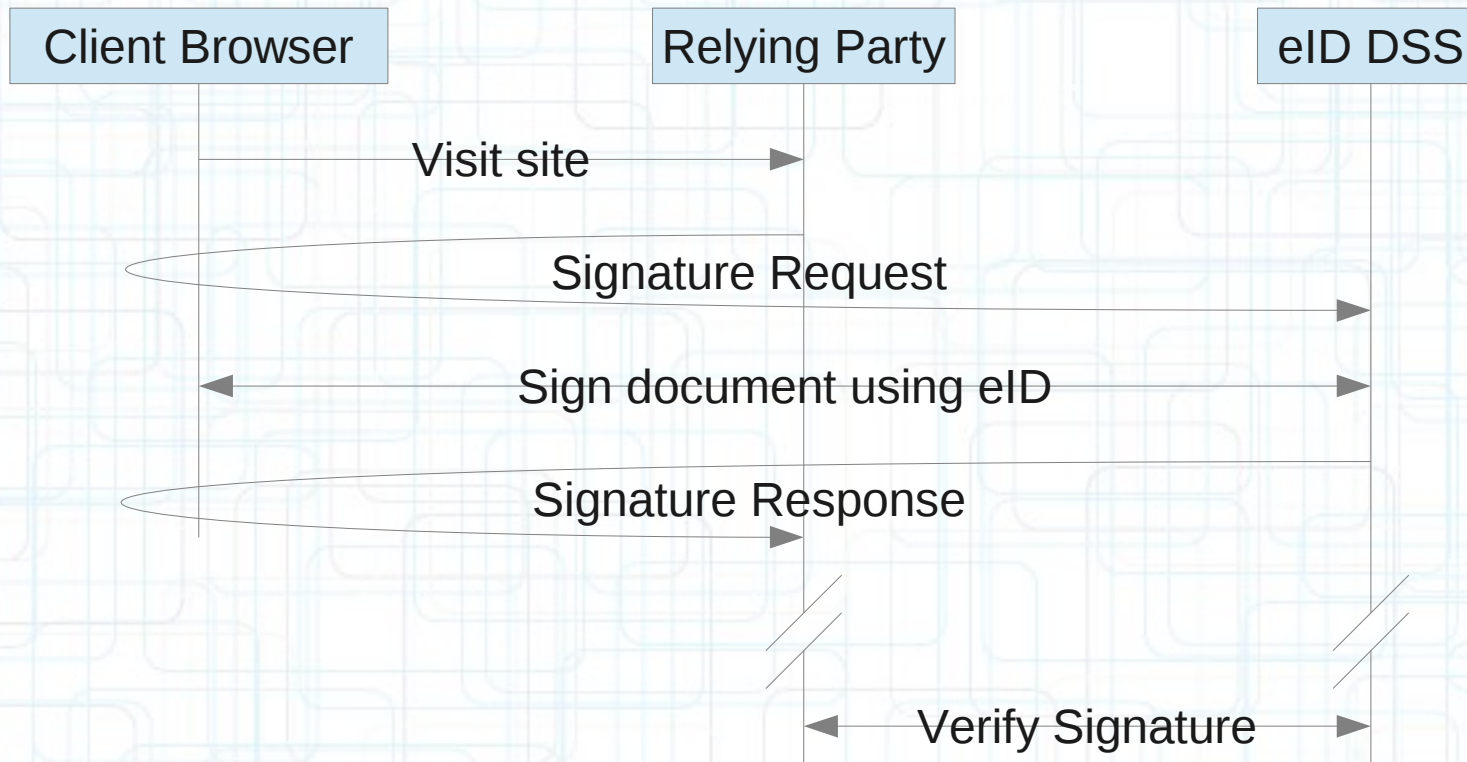


# eID DSS Work flow



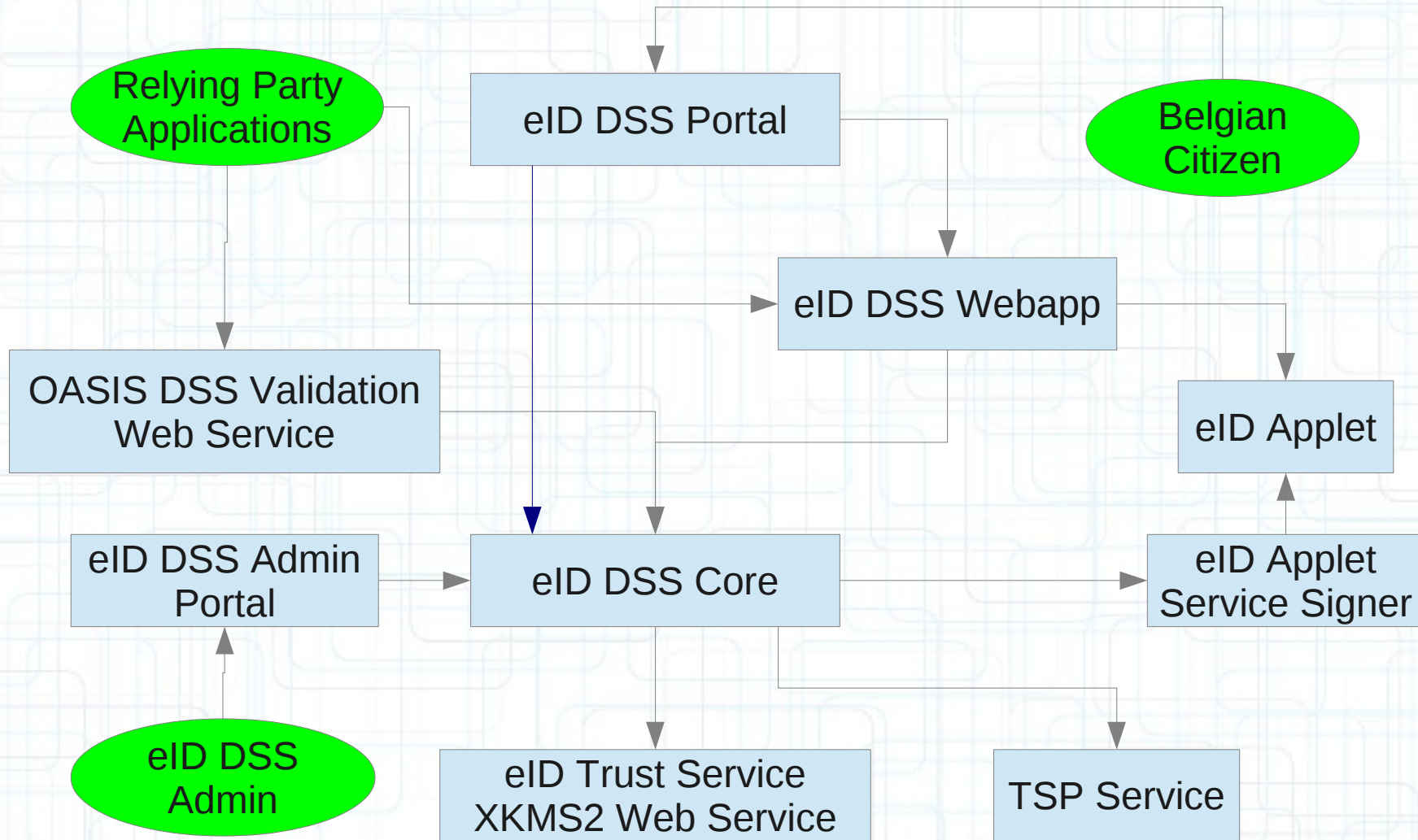
- Intuitive handling of multiple co-signatures

# eID DSS Protocol Flow



- Verification: OASIS DSS SOAP Web Service
- Creation: proprietary protocol for the moment
- Work in progress on OASIS DSS-X Local Signature Computation DSS Profile

# eID DSS Architecture



# Signature Types: EU Directive 1999/93/EC

Electronic Signature

Advanced Electronic Signatures

Qualified Electronic Signatures

Qualified Electronic Signatures with SSCD

Digital Signatures

QC

eID

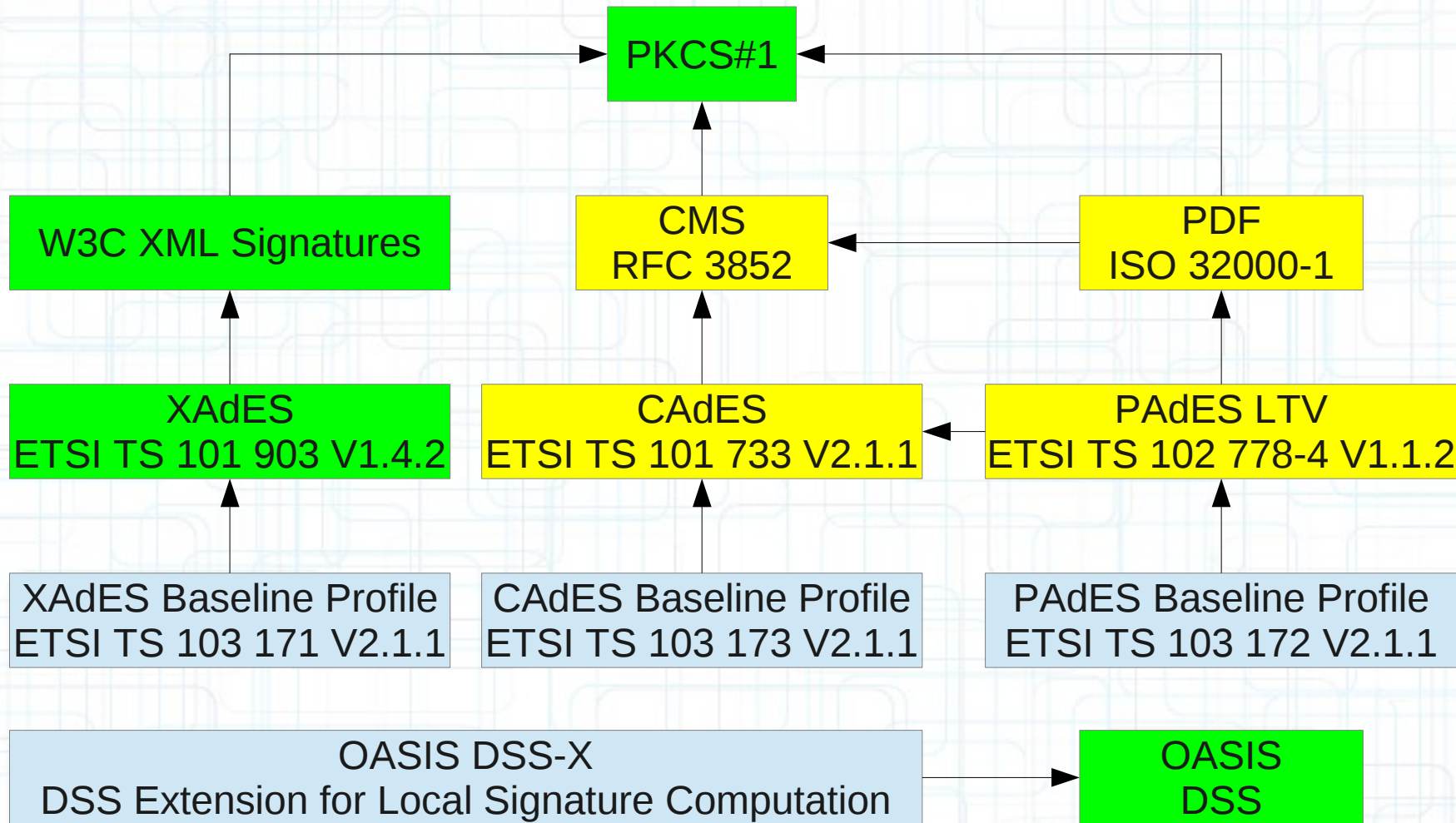
- Did Alice sign document D at time t ???

# eID Signatures

|                    | Authentication                           | Non-repudiation  |
|--------------------|--|--|
| Signature Verifier | Signature verifier = signature requestor | Signature verifier most likely not the signature requestor/creator.                      |
| Verification Time  | Instantly                                | Most likely long after signature creation, certificate might already be revoked/expired. |
| Legal aspects      | None required                            | Court might assign an expert to analyze the signature.                                   |

- Advanced Electronic Signatures thus require an open and self-contained signature format.
- Availability of independent implementations.

# Signature Formats



# eID DSS Document Formats

- ODF documents
  - Native ODF signatures: XAdES-X-L v1.4.2
  - Valid signatures in OpenOffice 3.2
- OOXML documents
  - Native OOXML signatures: XAdES-X-L v1.4.2
  - Valid signatures in MS Office 2007/2010
- XML documents
  - Co-signatures: XAdES-X-L v1.4.2
- ZIP container

# Signature Format versus Document Format

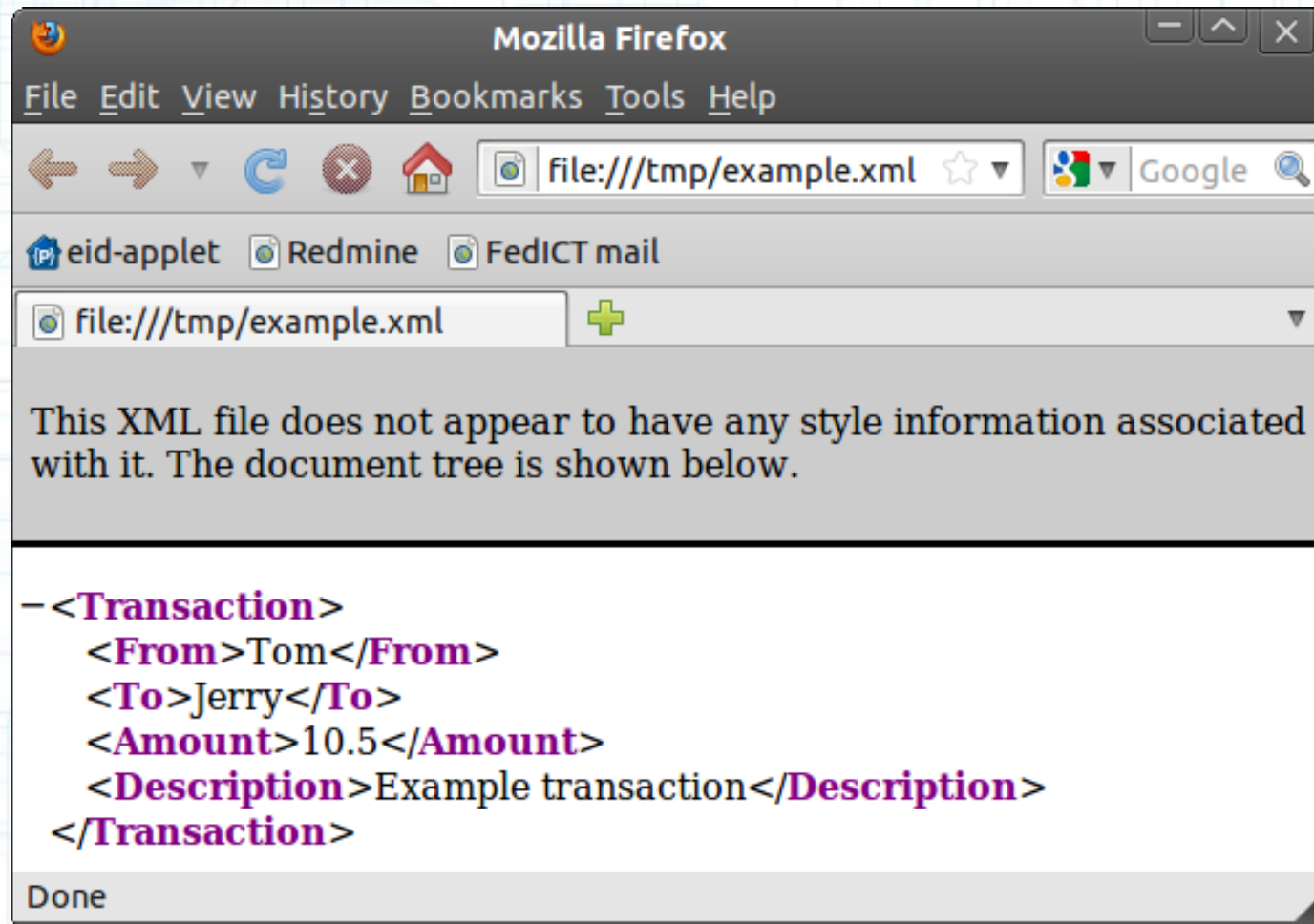
|        | XAdES     | CAdES     | PAdES     |
|--------|-----------|-----------|-----------|
| XML    | Native    | Container | Container |
| ODF    | Native    | Container | Container |
| OOXML  | Native    | Container | Container |
| Binary | Container | Native    | Container |
| PDF    | Container | Container | Native    |

- Native: document format has native support for the signature format.
- Container: the document format has no support for the signature format. Thus we need to construct a document container suited for the given signature format.
- Only XAdES is versatile towards doc formats.



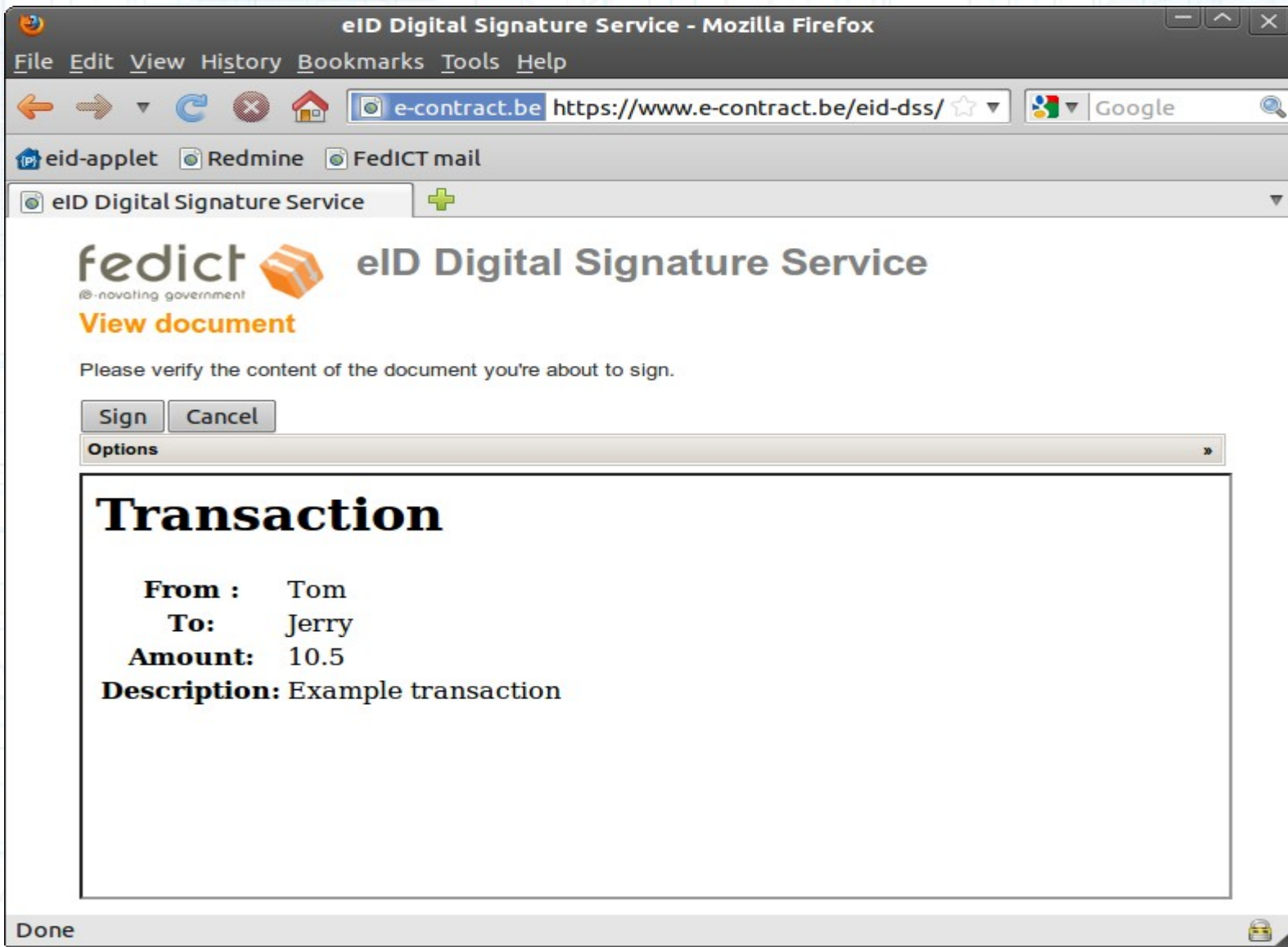
# eID DSS: XML document format

- Business Domain Specific Language in XML
- Example: a financial transaction



# eID DSS: XML document format

- The application uses eID DSS to sign the XML



# Q&A

- <https://www.e-contract.be>
- <https://code.google.com/p/eid-applet/>
- <https://code.google.com/p/eid-idp/>
- <https://code.google.com/p/eid-dss/>
- <https://code.google.com/p/eid-trust-service/>
- <https://code.google.com/p/commons-eid/>
- <https://code.google.com/p/eid-mw/>